A hybrid systems approach to identification of nonlinear biochemical processes

Lindenaar, D.J.W.

Award date:
2006

Eindhoven University of Technology
Department of Electrical Engineering
Control Systems

TU/e technische universiteit eindhoven

A Hybrid Systems Approach to Identification of Nonlinear Biochemical Processes

by Daniel J.W. Lindenaar

Master of Science thesis

Project period: May 2005 - February 2006
Report number: 06A/04
Commissioned by: TUE / MBS-CS

Supervisors:
Ir. Mark W.J.M. Musters
Dr. Ir. Natal A.W. van Riel
Prof. Dr. Ir. Paul P.J. van den Bosch

Additional Commission members:
Prof. Dr. Peter A.J. Hilbers (TU/e)

Abstract

Biochemical processes are usually nonlinear and modeled using nonlinear differential equations. In classic modeling, the usual approach is to linearize these nonlinear differential equations in some work point. For biochemical processes, this approach is not feasible, not only due to the nature of the system's working range, but also due to the fact that these nonlinearities are often essential to the actual functioning of the system. Parameter identification on nonlinear systems is not an easy task. The most striking problems are the high sensitivity to measurement noise and the necessity to have good a-priori estimates of the parameters.

A way to get round the problem of having a nonlinear system is to take many linearizations to capture the nonlinear behavior. When applying this idea to a system of nonlinear differential equations with a piecewise affine (PWA) system. This allows the system to be identified using linear optimization techniques and due to the simpler model structure, some general approaches can be developed. The major drawbacks of taking this approach are an increase in number of parameters and the problem of clustering measured data into the modes of the PWA system. In this paper, an approach to piecewise linearization is presented as well as methods for parameter identification on these systems.

Often a lot of information is available on biochemical systems, which, other than a-posteriori verification, is unusable in identification. In biochemistry, identification is usually hampered due to small amounts of data and high noise levels. Therefore, a lot could be gained if this a-priori information would be used during identification. A method is presented by which this a-priori information is converted to information which can be used in identification. This method is based on a qualitative abstraction of the PWA systems, which can be found from nonlinear systems using piecewise linearization. This qualitative abstraction method is presented along with a method to put this abstraction to use on incorporating a-priori information into the identification process.

Acknowledgements

Hereby I would like to thank Ir. Mark Musters for the support and inspiration he provided in his daily supervision during the time of my graduation at the Control Systems group.
Also thanks go out to Prof. Paul van den Bosch and Dr. Natal van Riel for the fruitful discussions, especially during the final few months.

Contents

Chapter 1

Introduction

Biological systems are generally very complex. Research in biology focuses primarily on understanding the individual components of systems, neglecting the interaction between different parts which generate the system's behavior. To understand the full functioning of a biological system, knowledge about each individual element of the system is insufficient. Interaction between subsystems is at least as important. The complexity of both the subsystems and the interaction between subsystems hampers the analysis of the system as a whole. Mathematical modeling of these systems as a tool to aid analysis has increasingly gained attention from biologists during the last few years. This resulted in a novel field of research, which is known as systems biology.

In the field of systems biology the white-box modeling approach is commonly used. White-box modeling can lead to very accurate models, but only if the subsystems are known and modeled accurately. The parameters of these models often have a physiological meaning. In systems biology often nonlinear continuous time differential equations are used to model the behavior of subsystems, which leads to a complex nonlinear description of the whole system. These nonlinearities complicate analysis due to the properties of nonlinear parameter identification.

Nonlinear parameter identification is accompanied by a number of problems. First, good initial guesses for all parameters are necessary to come to a good identification result. Biological measurement data is often heavily contaminated with noise, which contributes to the formation of local or global minima. This could lead to identification results describing the realization of the measurement noise more than the behavior of the actual system.

As a good alternative to the white-box approach, a grey-box approach can be used. This method still allows for the use of much of the available information on the system, while a simpler model is sought which describes the system only sufficiently accurate for the model's purpose. In this simpler model, the physiological meaning of the parameters is lost, as well as the

distinguishable parts.

One implementation of grey-box modeling is linearization of the nonlinear parts of the differential equations in some working point. This reduces the model to a set of linear differential equations. For biological systems this approach is usually not feasible due to the fact that the differential equations are usually too strongly nonlinear over a certain working range to be linearized.

By using a hybrid systems approach the problem of nonlinearity in the working area can be overcome. This approach has recently been introduced in [6]. By dividing the working area into smaller pieces and linearizing the nonlinearities in each of these pieces, a good description of the nonlinear behavior can be found while yielding simple linear differential equations. The resultant system is now divided into several subsystems whereby each subsystem has a set of linear differential equations. A system like this can be described by a hybrid systems model.

In biological systems, there usually is an abundance of qualitative information about the behavior of the system. This information is usually not usable for identification purposes other than a-posteriori validation. Recent research on qualitative simulation [1, 2, 3] has shown that qualitative information can be used to derive a system's behavior. This simulation algorithm is based on a small class of hybrid systems, which can be used to describe certain biological systems which exhibit clear switch-like behavior. In this paper, the simulation algorithm is changed to allow for a much broader class of hybrid systems. Also, the use of the algorithm is inverted so that a-priori information on the behavior of the system is used to validate restrictions on parameter values.

This paper discusses a general method to use a hybrid systems model to describe complex and nonlinear biological systems. In chapter 2 the use of hybrid systems to describe nonlinear equations is introduced. A method to use as much a-priori information as possible is introduced in chapter 4 as well as a method to perform the final parameter estimation.

Chapter 2

Hybrid Systems Approximation of Nonlinearities

In biology, the use of hybrid systems is an emerging field. Typical applications are limited to systems in which clear switching behavior is observed, such as genetic regulatory networks [2, 3], signal transduction pathways [7] and opening and closing of membrane channels [9]. Another use of hybrid systems will be presented here.

In classic modeling and control, nonlinearities are linearized in a certain working point using a first order Taylor approximation. Most nonlinearities in biological systems are essential to the dynamical behavior of the system. Usually these nonlinearities are too strongly nonlinear in the working range to be linearized without large deviations from the original function. Therefore linearization in a single working point is not an option. A solution to this problem is to use a piecewise linearization.

When doing piecewise linearization, the working range is divided into multiple intervals and the nonlinear function is linearized separately for each of these intervals. As the union of these intervals equals the original working range, aggregating the linearizations results into a piecewise affine (PWA) function, which will describe a piecewise linearization of the nonlinear function. When using this method on a system of nonlinear differential equations, the resulting piecewise linearized system becomes a PWA system, which is a class of hybrid systems.

2.1 Problems Introduced by Nonlinear Parts of Differential Equations

Identification of nonlinear systems poses a set of problems. Nonlinear least squares optimization is known to be sensitive to noise introduced in measurement. The noise will cause local minima in the cost function, which makes accurate parameter estimation difficult. Also in nonlinear least squares optimization, local minima are likely to exist even without any measurement noise. In traditional nonlinear methods, the only way to guarantee convergence to the global minimum is to have a very good initial estimate. This initial estimate is often not available, because this is the goal of the whole identification procedure.

Another downside to traditional nonlinear optimization is that, due to the fact that convergence cannot be guaranteed, finding the global minimum is either uncertain or requires massive computational power to achieve.

2.2 A Subclass of Piecewise Affine Systems

To enable further analysis of the results of piecewise linearization of a set of nonlinear differential equations, a suitable class of piecewise affine functions and resulting piecewise affine systems will be introduced. Also, this class is chosen such that qualitative analysis is possible, as presented in chapter 3.

First some definitions:

Definition 1 *An affine function is defined as:*

$$f(x) = k_0 + k_1 \cdot x \tag{2.1}$$

Definition 2 *A piecewise affine function can be defined as:*

$$f(x) = \begin{cases} k_{0,0} + k_{1,0} \cdot x & \text{for } x \in [\min(x), \theta_1) \\ k_{0,1} + k_{1,1} \cdot x & \text{for } x \in (\theta_1, \theta_2) \\ \vdots \quad \vdots & \vdots \\ k_{0,M} + k_{1,M} \cdot x & \text{for } x \in (\theta_M, \max(x)] \end{cases} \tag{2.2}$$

Where $k_{n,m}$ are parameters, θ_m are thresholds i.e. values at which the range for x is divided, and M the number of thresholds leading to $M+1$ intervals.

Note that $f(x)$ is left undefined on $x \in \{\theta_1, \theta_2, \cdots, \theta_M\}$. This is not desirable when using this system in simulation or other analysis, however, it is a necessary condition to allow for the qualitative simulation of the final system, as described in chapter 3. If a choice must be made, due to problems introduced by the undefined area, it should be made dependent on the specific system under scrutiny, keeping the changes introduced by this choice in mind. This paper will not go into the details of this procedure.

Most models of biochemical processes deal with more than one state variable. With the introduction of multiple state variables, the number of modes increases exponentially. For example, a system of three state variables and a single threshold for each state variable already results in eight modes and therefore eight different sets of three differential equations.

The class of hybrid systems will be restricted to a special subclass of the PWA class. This subclass is characterized by a subdivision of the state space by planes. These planes can be described by $x_n = \theta_m^n$. Such subdivision leads to rectangular or (hyper)cubic areas in the state space for each mode of the system. The choice of this class leads to a much simpler qualitative analysis as presented in chapter 3. Also each state variable's differential equation can be written down separately and there is a simplified notation scheme for this.

The restrictions imposed by this class are acceptable. In practice this class of systems would be found for some nonlinear systems, because nonlinearities frequently depend on just a single state variable. In case more than a single state variable is involved, a reasonable approximation can often still be found, as will be shown in an example later.

This subclass of PWA systems can be written in a compact way by using step functions.

Definition 3 *The step function is defined as:*

$$s^+(x,\theta) = \begin{cases} 0 & \text{if } x < \theta \\ 1 & \text{if } x > \theta \end{cases} \tag{2.3}$$

Again, the value of this step function is left undefined on $x = \theta$. If a definition is necessary, it could be done like:

Definition 4 *The step function with a value at $x = \theta$ is defined as:*

$$s^+(x,\theta) = \begin{cases} 0 & \text{if } x < \theta \\ \alpha & \text{if } x = \theta, \alpha \in [0,1] \\ 1 & \text{if } x > \theta \end{cases} \tag{2.4}$$

By choosing an α to be either zero or one, the point $x = \theta$ can be added to the left or right side of the threshold. Also, an α other than zero or one can be chosen, to allow for certain behavior at the threshold, depending on the system under scrutiny.

With this step function the PWA function of equation 2.2 can be rewritten as:

$$\begin{aligned} f(x) &= l_{0,0} + l_{1,0} \cdot x + s^+(x,\theta_1) \cdot (l_{0,1} + l_{1,1} \cdot x) + \ldots \\ &\quad + s^+(x,\theta_M) \cdot (l_{0,M} + l_{1,M} \cdot x) \end{aligned} \tag{2.5}$$

The parameters $l_{n,m}$ relate to the parameters in equation 2.2 as

$$l_{n,m} = \begin{cases} k_{n,m} - l_{n,m-1} & \text{if } n > 0 \\ k_{n,m} & \text{if } m = 0 \end{cases} \tag{2.6}$$

Expanding this notation to vectors of state variables, vectors $\vec{x'}$ and $\vec{l_m^n}$ are introduced.

$$\vec{x'} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_N \end{pmatrix} \quad \text{and} \quad \vec{l_m^n} = \begin{pmatrix} l_{0,m}^n & l_{1,m}^n & \cdots & l_{N,m}^n \end{pmatrix} \tag{2.7}$$

In these vectors, N is the number of state variables and $1 \leq n \leq N$ the index of a state variable and $n = 0$ refers to the offset part of the affine functions. m is the index of an interval of the PWA function with $0 \leq m \leq M_n$ and M_n the number of intervals of x_n. A set of binary functions s_m^n needs to be introduced which describe whether a certain parameter vector l_m^n is active:

Definition 5 *The set of functions $s_n^m(\vec{x}) : \mathbb{R}^N \rightarrow \{0,1\}$ are logical combinations of step functions as in definition 3. Any logical operator may be used, e.g. negation, OR'ing, AND'ing etc.*

If the step functions are used as defined in definition 4, the definition of s_n^m transforms into:

Definition 6 *The set of functions $s_n^m(\vec{x}) : \mathbb{R}^N \rightarrow [0,1]$ are combinations of step functions as in definition 4. Multiplication and averaging can be used to combine the step functions, but it should be observed that $0 \leq s_n^m \leq 1$. The multiplication takes the place of AND and the averaging the place of OR from the previous definition.*

With these definitions, the whole class of PWA functions used in this paper can be described by

$$\begin{array}{lcl} \dot{x}_1 & = & \vec{l_0^1} \cdot \vec{x'} + s_1^1(\vec{x}) \cdot \vec{l_1^1} \cdot \vec{x'} + \ldots + s_1^{M_1}(\vec{x}) \cdot \vec{l_1^{M_1}} \cdot \vec{x'} \\ \dot{x}_2 & = & \vec{l_0^2} \cdot \vec{x'} + s_1^2(\vec{x}) \cdot \vec{l_1^2} \cdot \vec{x'} + \ldots + s_2^{M_2}(\vec{x}) \cdot \vec{l_2^{M_2}} \cdot \vec{x'} \\ \vdots & & \vdots \qquad\qquad\qquad\qquad \vdots \\ \dot{x}_N & = & \vec{l_0^N} \cdot \vec{x'} + s_1^N(\vec{x}) \cdot \vec{l_1^N} \cdot \vec{x'} + \ldots + s_N^{M_N}(\vec{x}) \cdot \vec{l_N^{M_N}} \cdot \vec{x'} \end{array} \tag{2.8}$$

2.3 PWA Approximation for Known Nonlinear Functions

In table 2.1 a number of PWA approximations for nonlinearities commonly found in differential equations of biological systems are shown. The equation 2.10 in known as the Michaelis-Menten equation and equation 2.11 as the Hill curve. A better approximation can be achieved by using more than one or two pieces. In identification, however, this would lead to fewer and fewer data points per interval and an increasing number of parameters leading to worse identification results. Therefore the choice of numbers of

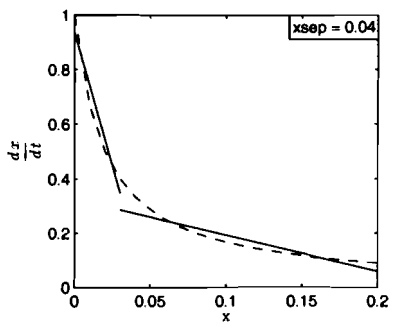
$$\frac{V}{k_m + x_1} \Rightarrow \vec{l_0}\vec{x'} + s^+(x_1, \theta)(\vec{l_1}\vec{x'}) \tag{2.9}$$

$$\frac{Vx_1}{k_m + x_1} \Rightarrow \vec{l_0}\vec{x'} + s^+(x_1, \theta)(\vec{l_1}\vec{x'}) \tag{2.10}$$

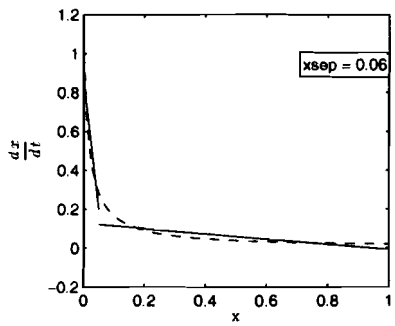
$$\frac{Vx_1^n}{k_m^n + x_1^n} \Rightarrow \vec{l_0}\vec{x'} + s^+(x_1, \theta_1)(\vec{l_1}\vec{x'}) + s^+(x_1, \theta_2)(\vec{l_2}\vec{x'}) \tag{2.11}$$

$$x_1 x_2 \Rightarrow \vec{l_0}\vec{x'} + s^+(x_1, \theta)(\vec{l_1}\vec{x'}) + s^+(x_2, \theta)(\vec{l_2}\vec{x'}) \tag{2.12}$$

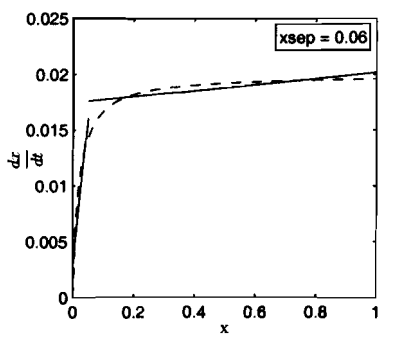
Table 2.1: Some approximations of nonlinear functions with PWA functions



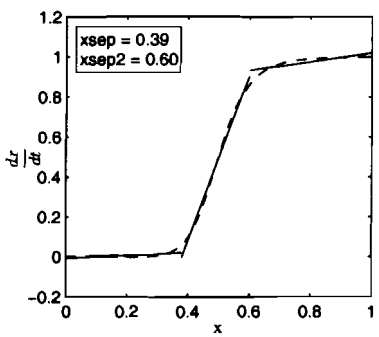
(a) equation 2.9, $x \in (0, 0.2)$



(b) equation 2.9, $x \in (0, 1)$

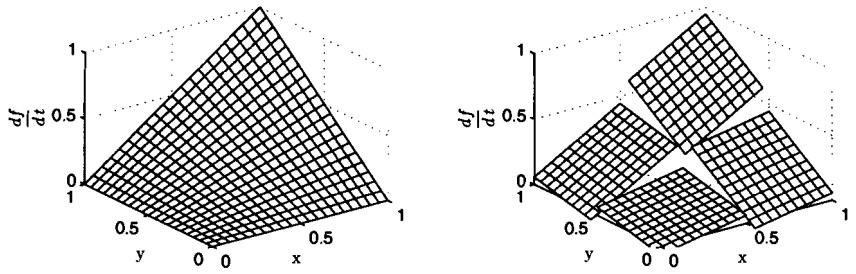


(c) equation 2.10



(d) equation 2.11

Figure 2.1: Conversions from nonlinear to PWA



(a) Nonlinear (b) PWA approximation

Figure 2.2: Example conversion for equation 2.12

intervals is a tradeoff between the accuracy of linearization and the number of data points per interval.

In figure 2.1, some examples of the nonlinear curves with the respective PWA approximations are shown. From figure 2.1(a) and figure 2.1(b), it can be seen that the total range over which approximation is made, is of importance to the final optimal fit of PWA function.

In figure 2.2 the approximation of equation 2.12 is shown. This shows the ability to describe functions of two variables with the presented class of PWA approximations. It can be observed that the approximation is not very accurate close to the thresholds, especially at the intersection of both thresholds. More accuracy is, however, not always necessary. If the uncertainty of the parameters in the nonlinear function is high, as is usually the case in biological systems, any extra accuracy is lost in deviations caused by these uncertainties. Also, in identification, putting a lower weight on data points near the thresholds can diminish the influence of these deviations on estimation results.

To find the parameter values of the PWA approximations shown in figure 2.1, a least squares optimization is carried out, minimizing a cost function $J(\vec{\theta},\vec{l}) = \left\| f(\vec{x},\vec{\theta},\vec{l}) - g(\vec{x}) \right\|^2$ where $f(\vec{x},\vec{\theta},\vec{l})$ is the PWA approximation and $g(\vec{x})$ is the original nonlinear function. If carried out on a computer, this optimization is difficult, because the cost function becomes a piecewise constant function due to the sampling of $\vec{x}$. Therefore a specialized algorithm for finding optimal thresholds is needed. To achieve certainty on whether the global minimum is found when minimizing the cost function with respect to $\vec{\theta}$, an exhaustive search of the state space must be carried out.

For a given $\vec{\theta}$, $J(\vec{x},\vec{\theta},\vec{l})$ can be minimized using linear least squares optimization. Therefore this is a very fast process and an exhaustive search is feasible for fairly large amounts of data and thresholds. More information on this exhaustive search method is presented in section 4.1.

The class of PWA models is also known under the term threshold autore-

gressive (TAR) models. There are some algorithms designed for this class of systems, but these are all targeted on different areas of research and have not been considered here.

An alternative algorithm for finding threshold positions has been devised. However, this algorithm is only able to find the global minimum if the steps between adjacent constant pieces of the cost function with changes in $\vec{\theta}$ are sufficiently small. As this can only be achieved by dense sampling and practical absence of noise, this algorithm can probably only be used to find threshold positions for transforming a known nonlinear function into its PWA equivalent.

As an example, this algorithm for fitting a two-piece PWA function onto a nonlinear function $f(x)$ for a range $\chi = [x_{min}, x_{max}]$ is shown. This is the algorithm used to find the approximations shown in figure 2.1(a-c). For the approximation of equation 2.11 and figure 2.2, an exhaustive search algorithm was used.

First a cost function J is defined:

$$J(\theta, \vec{l^0}, \vec{l^1}) = \int_{x \in \chi} \left(\vec{l^0} \cdot \vec{x'} + s^+(x, \theta) \cdot \vec{l^1} \cdot \vec{x'} - f(x) \right)^2 dx \tag{2.13}$$

This cost function is then minimized to obtain values for θ, $\vec{l^0}$ and $\vec{l^1}$ simultaneously:

$$\left\{ \theta, \vec{l^0}, \vec{l^1} \right\} = \arg \min_{\theta, \vec{l^0}, \vec{l^1}} J(\theta, \vec{l^0}, \vec{l^1}) \tag{2.14}$$

To implement this algorithm on a computer, the integrals in equation 2.13 must be replaced by sums over a set of sample points in χ. Now the cost function becomes a piecewise constant function. To find the minimum for this piecewise constant function an algorithm was devised based on the idea of binary search resulting in a bisection-like algorithm.

The algorithm starts by setting the range of possibly optimal threshold positions to the range of the state variable:

$$\begin{array}{rcl} \theta_{max} & = & \max(x) \\ \theta_{min} & = & \min(x) \end{array} \tag{2.15}$$

The optimal solution θ is then known to be conform to $\theta_{min} \leq \theta \leq \theta_{max}$. To find θ, the range $[\theta_{min}, \theta_{max}]$ is recursively reduced, until no data points lie in that range. To achieve this, the following algorithm is used:

1. Select two trial thresholds θ_1 and θ_2 so that both thresholds are respectively at $\frac{1}{4}$ and $\frac{3}{4}$ into the test range:

$$\begin{array}{rcl} \theta_1 & = & \theta_{min} + \frac{1}{4}(\theta_{max} - \theta_{min}) \\ \theta_2 & = & \theta_{min} + \frac{3}{4}(\theta_{max} - \theta_{min}) \end{array} \tag{2.16}$$

2. Calculate the minimal cost function J_1 and J_2 when using these two fixed thresholds:

$$\begin{aligned} J_1 &= \min_{\vec{l^0},\vec{l^1}} J(\theta_1, \vec{l^0}, \vec{l^1}) \\ J_2 &= \min_{\vec{l^0},\vec{l^1}} J(\theta_2, \vec{l^0}, \vec{l^1}) \end{aligned} \tag{2.17}$$

3. If $J_1 > J_2$, the optimal threshold is closer to θ_2 than to θ_1, so the range $[\theta_{min}, \theta_1]$ is discarded by setting $\theta_{min} = \theta_1$. If $J_1 < J_2$, the optimal threshold is probably closer to θ_1 than to θ_2, so the range $[\theta_2, \theta_{max}]$ is discarded by setting $\theta_{max} = \theta_2$. If the cost-function is sufficiently smooth, the invariant $\theta_{max} \leq \theta \leq \theta_{min}$ will still hold.

4. If $\theta_{max} - \theta_{min} > \delta$, with δ the smallest interval between two consecutive samples in x, continue at item 1.

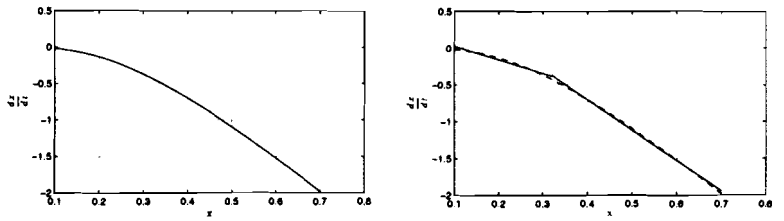
5. If there is no data point in $[\theta_{min}, \theta_{max}]$, θ can be chosen arbitrarily in that range and the algorithm finishes.

6. A data point $x(k)$ lies in $[\theta_{min}, \theta_{max}]$. Now choose a θ_1, θ_2 such that $\theta_{min} < \theta_1 < x(k) < \theta_2 < \theta_{max}$. Now calculate

$$\begin{aligned} J_1 &= \min_{\vec{l^0},\vec{l^1}} J(\theta_1, \vec{l^0}, \vec{l^1}) \\ J_2 &= \min_{\vec{l^0},\vec{l^1}} J(\theta_2, \vec{l^0}, \vec{l^1}) \end{aligned} \tag{2.18}$$

   If $J_1 < J_2$, $\theta = \theta_1$, otherwise $\theta = \theta_2$. The algorithm finishes.

Note that if the precondition of sufficiently small steps in the cost function with respect to $\vec{\theta}$ is not met, there's no guarantee of convergence to a global minimum.



(a) Shape of $\frac{V_{\text{SERCA}}(x)}{B(x)}$

(b) Shape of $\frac{V_{\text{SERCA}}(x)}{B(x)}$ with PWA approximation

Figure 2.3: Conversion of nonlinear function

2.4 Example: Identification Procedure for SERCA Pump

Calcium plays an important role in muscle contractions. One of the mechanisms at play is the sarcoplasmatic reticulum into which calcium is pumped and periodically released. This pump is called the SERCA pump.

The kinetics of the SERCA pump are modeled as presented in [8]:

$$V_{\text{SERCA}}(c_i) = \frac{V_{max,\text{SERCA}} \cdot c_i^2}{c_i^2 + K_{m,\text{SERCA}}^2} \tag{2.19}$$

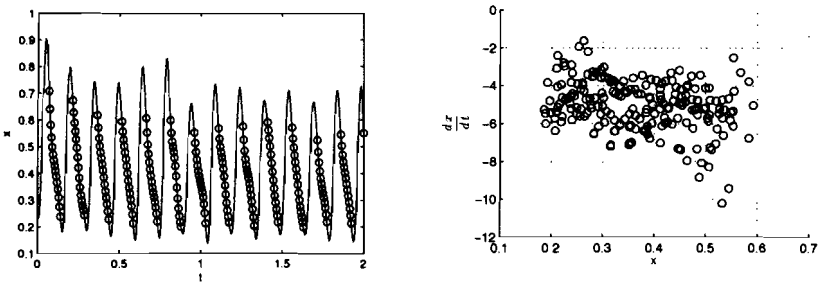
and the system with no release active:

$$\dot{x} = \frac{V_{leak} + V_{\text{SERCA}}(x)}{B(x)} \tag{2.20}$$

The goal of parameter identification is to find values for $V_{max,\text{SERCA}}$ and $K_{m,\text{SERCA}}$. Due to the term $B(x)$ in the differential equation, which is a nonlinear approximation of influences of buffering of calcium, no direct conversion to a PWA system as proposed in section 2.3 is possible. A visual inspection using parameter values with the right order of magnitude shows that V_{leak} is small enough to be neglected and that $\frac{V_{\text{SERCA}}(x)}{B(x)}$ can be approximated with a PWA function with one threshold, see figure 2.3. The differential equation in equation 2.20 transforms into

$$\dot{x} = l_0 + l_1 \cdot x + s^+(x, \theta_1)(l_2 + l_3 \cdot x) \tag{2.21}$$

From inspection of figure 2.3(b) some information can directly be derived: $l_0 \geq 0$, $l_2 > 0$, $l_1 \leq 0$, $l_3 < 0$. Further a-priori information is not available and, due to the simplicity of the PWA function, not required in this case. Qualitative simulation does not reveal anything new either, because the system is one-dimensional with only two regular domains.



(a) Measurement data, usable data is encircled

(b) Approximate derivative of measurement data

Figure 2.4: Measurement data

The measurement data is only partially usable due to the fact that during calcium release, the influence of the SERCA pump on the concentrations is too insignificant to be measured accurately. Therefore only data is used at times when no release is active and the only influence in calcium concentration is the SERCA pump. In figure 2.4(a) the encircled data points are used for identification. In figure 2.4(b), the approximate derivative is shown. It can be seen that apart from being noisy, there is some offset in the data, not accounted for in the kinetics of the SERCA pump. Therefore, the constraints are relaxed during this identification to allow for this offset. Identification results in a threshold at $\theta = 0.46$ and a parameter vector of:

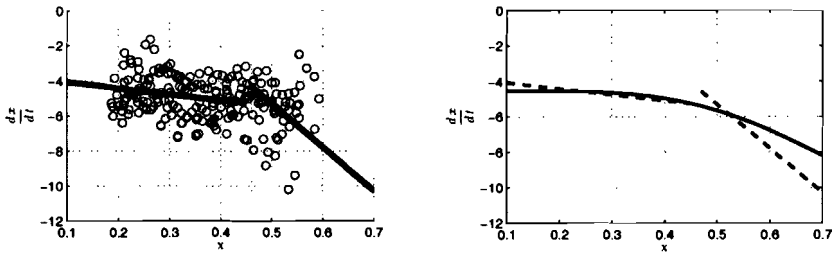
$$\vec{l} = \{ \; -3.7 \quad -3.4 \quad 10.8 \quad -21.4 \; \} \tag{2.22}$$

This resultant threshold location is not at the global minimum for $J(\vec{\theta}, \vec{l})$. This minimum lies at $\theta = 0.53$, but as can be seen from figure 2.4(b), there is a lot of noise in this region and the identification result just describes that noise. Therefore that threshold was discarded and the next minimum used. This flaw could be counteracted by either demanding a minimal number of data points per mode, or by adding a weighting to the data points, emphasizing precision in the middle of each interval and decreasing influence of offset near thresholds.

In figure 2.5(a), the resulting fit is shown.

Finding the original parameters $V_{max,\text{SERCA}}$ and $K_{m,\text{SERCA}}$ from the parameters of the PWA system is not trivial. Due to the strong interdependence of these two parameters, a nonlinear least squares optimization generally does not yield a plausible result. For this identification, the result is $V_{max,\text{SERCA}} = 2931$ and $K_{m,\text{SERCA}} = 1.1$, while known values are in range $[100, 800]$ for $V_{max,\text{SERCA}}$ and around 0.2 for $K_{m,\text{SERCA}}$. By fixing $K_{m,\text{SERCA}}$ to 0.2, a plausible value of $V_{max,\text{SERCA}} = 676$ is found.

These results are not very consistent, yet this could be attributed to a set of problems with the original nonlinear model. First of all, V_{SERCA} never



(a) Results of identification

(b) Fit of nonlinear SERCA kinetics. $K_{m,\text{SERCA}} = 0.2$, $V_{max,\text{SERCA}} = 676$

Figure 2.5: Identification results

approaches $V_{max,\text{SERCA}}$ with $K_{m,\text{SERCA}} = 0.2$ within the sampled range. Also, when estimating the nonlinear parameters based on the PWA approximation, the correctness of the assumed $B(x)$ has a large influence. This correctness is not guaranteed and several other approaches to this nonlinear system are being researched.

Chapter 3

Qualitative Simulation of PWA Systems

The qualitative abstraction method presented here is based on work by de Jong et.al. ([1, 2, 3]), but extends this research to allow for a larger class of hybrid models. The class of models used in their previous work is a subclass of the models used here (equation 2.8).

The model class chosen by de Jong allows for a vast number of simplifications in the algorithm. One of the major differences is the fact that a differential equation in their class can be equal to zero, only depending on the state variable corresponding to that differential equations. So $\dot{x_n} = 0 \rightarrow x_n = \alpha$. This results in simple planes representing the solutions to nullcline equalities. Whereas the work presented here has to consider slanted planes and multiple possible configurations with respect to intersection with the threshold planes. This also results in non-(hyper)rectangular flow domains, which complicates derivation of transition rules.

The goal of this qualitative simulation is to find a directed graph representing all possible trajectories through the state space of the model. The nodes of the graph will represent subspaces of the state space where identical behavior is observed at any point in that subspace. Each edge represents a trajectory crossing the boundary between two adjacent subspaces, represented by the connected nodes. The direction of the edge is dictated by the order in time at which the trajectory passes through each subspace.

Note that the graph will represent every possible trajectory. If a choice between multiple possible trajectories has to be made, all resulting edges will appear in the resulting graph.

3.1 Division of the State Space

The first step in the simulation algorithm is the dividing the state space into areas where the signs of the derivative of each state variable is constant. In

this context the sign of a function is defined as:

Definition 7

$$\text{sign}(x) = \left\{ \begin{array}{ll} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{array} \right\} \tag{3.1}$$

The state space of a PWA system can be seen as a multidimensional space, with the number of dimensions equal to the number of state variables in the system. This space will be limited in each dimension by a minimum and maximum possible value for the state variable corresponding to that dimension. If Ω_n is the range of state variable x_n, so $\Omega_n = [\min(x_n), \max(x_n)]$, then the state space of the complete system is $\Omega = \Omega_0 \times \Omega_2 \times \ldots \times \Omega_N$.

Each state variable has a set of thresholds (θ_n) due to step functions (definition 3) in the differential equations. The thresholds divide the state space into several subspaces, which are hypercubes. These subspaces, enclosed by, but not including the thresholds, are known as regular domains and the set of all regular domains is denoted by $\mathcal{D}_r$

Definition 8 *A regular domain is the largest contiguous subspace of Ω in which no state variable reaches a threshold.A regular domain has the same dimension as Ω. The set of all regular domains is called $\mathcal{D}_r$. Regular domains correspond to modes of the PWA system.*

In regular domains, the differential equations of the system are reduced to a system of linear differential equations.

Exactly on the thresholds, (hyper)planes divide the regular domains. These (hyper)planes are delimited by the nearest thresholds for all state variables which are not at a threshold. Each of these (hyper)planes is called a switch domain and the set of all switch domains is denoted by $\mathcal{D}_s$.

Definition 9 *A switch domain is the largest contiguous subspace of Ω in which at least one state variable is at a threshold and the others do not reach a threshold. A switch domain is a subspace of a lower dimension than Ω. Specifically, if D denotes a switch domain, $\dim(D) = \dim(\Omega) - \#s$, where $\#s$ is the number of state variables at a threshold. The set of switch domains is called $\mathcal{D}_s$*

In a switch domain, the differential equations of the PWA system are undefined due to the definition of the step-function (definition 3). This is a necessity which is explained later.

Definition 10 *A domain is a subspace of the state space Ω of a PWA system, which belongs either to the set of regular or switch domains. The set of all domains is $\mathcal{D} = \mathcal{D}_r \cup \mathcal{D}_s$. It also holds that $\bigcup_{D \in \mathcal{D}} D = \Omega$.*

Now some practical relations between domains will be defined as presented in [3].

Definition 11 *let D be a domain of dimension n. Let C be a hyperplane of dimension n containing D. The set $B(D)$ of all points $x \in C$, such that each ball $B_C(x, \epsilon)$ in C of center x and radius $\epsilon > 0$ intersects both D and $C \backslash D$*

This is a fairly complex definition, but it boils down to $B(D)$ containing any points, which lie in an expansion of D, where the distance from this point to the nearest point in D, δ, conforms to $\lim_{\epsilon \downarrow 0} 0 < \delta \leq \epsilon$. With the expansion of D, the subspace of the same dimension of D, expanded in the directions where D is nonsingular, is meant.

Definition 12

$$A(D) = \{D' \in \mathcal{D} | D' \subset B(D)\}$$

$A(D)$ is the set of domains which are of lower dimension than D and which are on the boundary of D. For a regular domain, the set $A(D)$ comprises all switch domains at the border of D and for a switch domain D, it comprises only those switch domains which are adjacent to D and have one more state variable at a threshold.

Definition 13

$$R(D) = \{D' \in \mathcal{D}_r | D \subset B(D')\}$$

$R(D)$ is the set of regular domains D' for which it holds that $D \in A(D')$.

The simulation algorithm is based on the idea that one can define subspaces of the state space Ω in which the sign pattern of the differential equations is the same at any point in that subspace. The sign pattern is a representation which corresponds to the direction of a trajectory passing through a single point. For regular domains, this sign pattern is a vector of signs of the differential equations. In switch domains, however, the sign pattern can be empty, corresponding to instantaneous crossing of the switch domain, or be one or more vectors, representing possible sliding mode behavior at that threshold. This representation is necessary due to the dependence of sliding mode behavior at thresholds on the numeric values in the actual parameter vector, which is not available during qualitative simulation.

Definition 14 *A sign pattern for some point $\vec{x}$ in the phase space Ω is $\mathcal{S}(\vec{x})$. $\mathcal{S} : \Omega \rightarrow 2^{\{-1,0,+1\}^N}$. For the sign pattern it holds that $\mathcal{S}(\vec{x}) = \{\mathcal{S}_1(\vec{x}) \times \mathcal{S}_2(\vec{x}) \times \ldots \times \mathcal{S}_N(\vec{x})\}$.*

For $x \in \bigcup_{D \in \mathcal{D}_r} D$, $\mathcal{S}_n(\vec{x}) = \text{sign}(\dot{x}_n)$. For $x \in \bigcup_{D \in \mathcal{D}_s} D$, not all $\dot{x}_n$ are defined. Therefore the sign pattern needs to be derived from sign patters of surrounding regular domains.

Definition 15 *A flow domain is a subspace of a domain in which the sign pattern is constant i.e. for any $\vec{x} \in F$ and any $\vec{y} \in F$ it holds that $\mathcal{S}(\vec{x}) = \mathcal{S}(\vec{y})$. A flow domain is, like a domain, the largest subspace in which previous propositions hold. The set of all flow domains will be $\mathcal{F}$.*

For further analysis the concept of the nullcline equation is needed. The nullcline equation is defined as $\dot{x}_n = 0$. The points of $\vec{x}$ which are a solution to the nullcline equation form a (hyper)plane for the class of systems used. These planes are known as nullcline planes.

For regular domains in which the differential equations do not change sign, or in other words in which none of the nullcline equations have a solution, the regular domain is also a flow domain which will be called a regular flow domain.

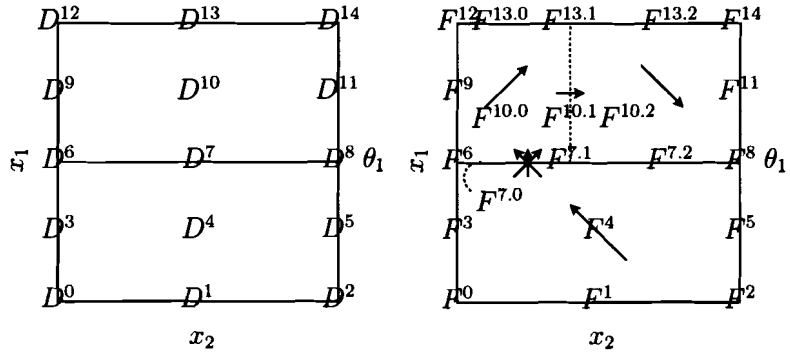
In case one or more nullcline equations do have a solution, the regular domain will be divided into subspaces in which no sign changes occur. This subdivision can be achieved by dividing the domain at (hyper)planes, each corresponding to the solution of one of the nullcline equalities as far as the solution lies within that regular domain. Due to the linearity of the differential equations in regular domains, there will always be a unique solution to the nullcline equality and that solution will form a (hyper)plane in the regular domain.

The subspaces of the regular domain enclosed by nullcline equation solutions and switch domains are regular flow domains. The (hyper)planes, being solutions to one or more nullcline equality, form a set of flow domains. Each flow domain will be a contiguous part of one of these (hyper)planes in which the same nullcline equalities have a solution. Therefore, an intersection of two (hyper)planes will be a separate flow domain and the parts of the hyperplanes on either side of the intersection will be different flow domains. These flow domains are called nullcline flow domains.

Definition 16 *A regular flow domain F is a flow domain and a subspace of a regular domain and for any $\vec{x} \in F$ it holds that $\mathcal{S}(\vec{x}) \in \{-1, +1\}^N$. Regular flow domains are N-dimensional.*

Definition 17 *A nullcline flow domain F is a flow domain and a subspace of a regular domain and for any $\vec{x} \in F$ it holds that some $\mathcal{S}_n(\vec{x}) = 0$. Nullcline flow domains are $(N-k)$-dimensional, where k is the number of state variables for which $\mathcal{S}_n(\vec{x}) = 0$*

The switch domains will be switch flow domains if none of the regular domains which have that switch domain at their border contain a solution to their respective nullcline equalities. If one of these regular domains does contain a solution to one of their nullcline equalities, the switch domain will be divided at the intersections of the (hyper)planes of the solutions with the domain. Due to this subdivision a number of flow domains are formed and these will all be switch flow domains.



(a) Division of Ω in Domains

(b) Division of Ω in flow domains. Arrows indicate signs of differential equations

Figure 3.1: Example system

Definition 18 *A switch flow domain F is a flow domain and a subspace of a switch domain D. Furthermore it holds that:*

1. *If for any $\vec{x} \in F$, $\vec{x}$ is a solution to one or more nullcline equalities of all domains in $R(D)$, then all $\vec{y} \in F$ should also be a solutions to those nullcline equalities.*
2. *If for any $\vec{x} \in F$, $\vec{x}$ is not a solution to any nullcline equality of all domains in $R(D)$, then no $\vec{y} \in F$ should be a solution to any of the nullcline equalities of all domains in $R(D)$.*

Switch flow domains are $(N-k)$-dimensional, where k is the number of state variables which are either at a solution of their nullcline inequality in some domain in $R(D)$ or at a threshold.

To clarify the procedure of finding the flow domains, a simple example will be shown.

$$\begin{array}{rcll} \dot{x_1} & = & l_{1,0} + l_{1,3} \cdot s^+(x_1, \theta_1) \cdot x_2 & l_{1,0} + l_{1,3} \cdot a = 0 \\ \dot{x_2} & = & l_{2,0} + l_{2,2} \cdot s^+(x_1, \theta_1) & l_{2,0} < -l_{2,2} \\ \min(x_2) < a < \max(x_2) & & & \end{array} \tag{3.2}$$

In figure 3.1(a) the division of the state space into domains can be seen. In this example,

$$\mathcal{D}_r = \{D^4, D^{10}\}$$

and

$$\mathcal{D}_s = \{D^0, D^1, D^2, D^3, D^5, D^6, D^7, D^8, D^9, D^{11}, D^{12}, D^{13}, D^{14}\}$$

There are no solutions for the nullcline equalities in D^4, as $x_1 = l_{1,0}$ and $x_2 = l_{2,0}$ in that domain. Therefore $F^4 = D^4$. Solutions for all nullcline

equalities in D^{10} are $x_2 = a$, and therefore D^{10} will be subdivided, as well as D^7 and D^{13}, as is shown in figure 3.1(b).

Now the definitions for A and R will be extended for flow domains as follows

Definition 19

$$A(F) = \{F' \in \mathcal{F} | F' \subset B(F)\}$$

$A(F)$ is the set of flow domains which are of lower dimension than F and which are on the boundary of F. For a regular domain, the set $A(F)$ comprises all switch flow domains on the border of F and for a switch domain F, it comprises only those switch domains which are adjacent to F and have one additional state variable at a threshold or at a solution of its nullcline equality.

Definition 20

$$R(F) = \{F' \in \mathcal{F}_r | F \subset B(F')\}$$

$R(D)$ is the set of regular domains D' for which it holds that $D \in A(D')$.

Note that due to the definition of $A(F)$ that with $F \in \mathcal{F}_n$, only the switch flow domains in the same plane as F is contained in $A(F)$. This excludes any other flow domains contained in the switch domain in which any flow domain in $A(F)$ is contained.

3.2 Finding All Flow Domains Using Qualitative Information

To qualitatively determine all possible flow domains in the state space, a set of rules is needed. The first step, the division into regular and switch domains is easily done, by giving the number of thresholds for each state variable and the ordering of those thresholds. In other words, defining $\vec{\theta}_n$ for all n. The next step is to find out in which regular domains any nullcline equations have a solution. And, if so, how these solutions are positioned in that regular domain, determining the division of the regular domain into several regular flow domains and nullcline flow domains.

The differential equation for the PWA system reduces to a simple set of ODE's in each regular domain.

$$\dot{x}_n = k_{n,0} + k_{n,1} \cdot x_1 + \ldots + k_{n,n} \cdot x_n + \ldots + k_{n,N} \cdot x_N \tag{3.3}$$

Now, let the set of thresholds defining the upper boundary for the regular domain be $\vec{\Theta}^{max}$ and the set of lower boundaries be $\vec{\Theta}^{min}$, such that for each state variable x_n of $\vec{x} \in D$ it holds that $\Theta_n^{min} < x_n < \Theta_n^{max}$.

To find whether a nullcline equation has a solution in a certain domain, one has to determine whether that state variable changes sign, somewhere

in that domain. To find this change in sign, two points $\vec{p}$ and $\vec{q}$ are chosen, with

$$p_m = \begin{cases} \Theta_m^{min} & \text{if} \quad k_{n,m} > 0 \\ \Theta_m^{max} & \text{if} \quad k_{n,m} \leq 0 \end{cases} \tag{3.4}$$

$$q_m = \begin{cases} \Theta_m^{max} & \text{if} \quad k_{n,m} > 0 \\ \Theta_m^{min} & \text{if} \quad k_{n,m} \leq 0 \end{cases} \tag{3.5}$$

If $\dot{x}_n$ changes sign in this domain, it must hold that $\dot{x}_n(\vec{p}) < 0$ and $\dot{x}_n(\vec{q}) > 0$ or $\dot{x}_n(\vec{p}) > 0$ and $\dot{x}_n(\vec{q}) < 0$.

Now the next piece of information needed is the layout of the nullcline planes within the regular domain. Apart from the signs of the $k_{n,m}$, which determine the direction of the slope of the plane, knowledge is needed about which switch domains are intersected by the plane and whether multiple nullcline planes intersect within a single regular domain. To provide a general approach to this problem, a recursive algorithm is needed. The algorithm is based on the idea that if the signs of the differential equations on each corner are known, it can be easily derived whether the nullcline plane intersects any of the 1-dimensional switch domains.

By definition, a switch domain D^1 with $dim(D^1) = 1$ and $D^r \in R(D^1)$, has an $A(D^1)$ with two elements. Take the sign of the differential equation of a certain state variable as defined in D^r at the two points corresponding to each of the switch domains in $A(D^1)$. If these signs differ, the nullcline plane of that state variable in D^r intersects D^1.

Now the question remains on how the intersections with D^1 are ordered. An ordering needs to be defined to include each nullcline domain in all domains in $R(D^1)$ which intersect D^1. This might be provided as a-priori information, however, this information is not always available. Therefore, all possibilities not yet fixed due to a-priori information need to be investigated.

From the ordering information for all switch domains of dimension $m-1$ in $A(D^m)$ for some domain D^m, it can be derived if any of the nullcline flow domains in that domain intersect with one another. By definition, a switch domain D^m with $dim(D^m) = m$ has a set $A(D^m)$ of 2^m elements of dimension $m-1$. It is possible that more than two nullcline domains intersect with each other. This introduces a complexity as the order in which one of these nullcline domains is intersected by the others each lead to a different subdivision of D^m. Because, again, this ordering is usually not available a-priori, the algorithm should iterate over all possible orderings to obtain the full set of possible graphs.

By repeating the above iteration until $\dim(D^m) = \dim(\Omega)$, a unique subdivision is found. From this subdivision, the final graph can be calculated.

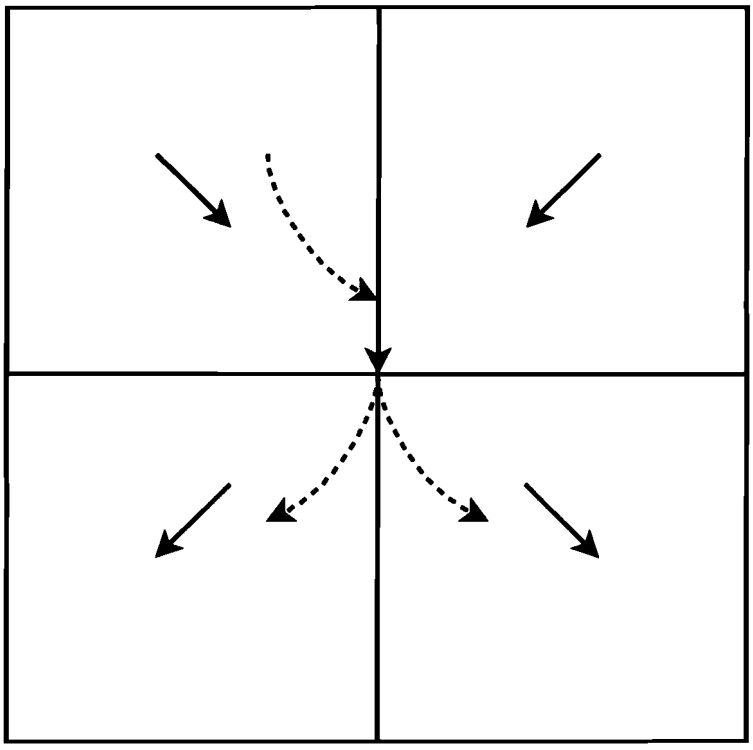


Figure 3.2: Example of ambiguity of behavior in switch domains

3.3 Deriving Possible Trajectories

As stated before, the sign pattern of switch flow domains needs to be derived from the surrounding regular flow domains. In quantitative simulation, this could be achieved by taking any of the differential equations of the surrounding regular mode domain. Another approach would be to take some averaged value from all differential equations in the surrounding regular domains. Nevertheless, due to the unavailability of quantitative information, this will usually not yield a single solution. For example, switch flow domain $F^{7.0}$ has:

$$S_2(F^{7.0}) = \{-1, 0, 1\} \tag{3.6}$$

In qualitative simulation another approach is necessary. As stated before, the goal of this simulation is to derive a graph representing all possible trajectories through the state space. Due to the fact that no quantitative information is available, any choice made on the behavior on thresholds will possibly exclude valid trajectories from the resulting graph.

An example of such a situation is shown in figure 3.2. Due to the signs of differential equations in the top two regular domains, a sliding mode

along the threshold between them is possible. As the threshold is reached leading towards the bottom two regular domains, it is not clear whether the trajectory will enter the left or the right domain. In the real system, one of the two trajectories will be followed, which depends on the actual numeric parameters. For qualitative simulation, however, both should be considered.

To come to a general rule to find the sign pattern for switch flow domains a definition of relative position is needed:

Definition 21 *The relative position from F^a to F^b, with either $F^a \in A(F^b)$ or $F^b \in A(F^a)$, is $W(F^a, F^b) = \vec{w}$, $w_n \in \{-1, 0, +1\}$. If there is some $x \in F^a$, $y \in F^b$ and $x_n = y_n$ for $n \neq m$ then $w_m = \text{sign}(y_m - x_m)$. if no such x and y can be found, $w_m = 0$. $W(F^b, F^a) = -W(F^a, F^b)$*

Theorem 1 *For some $F \in \mathcal{F}_s$:*

$$\mathcal{S}_n(F) = \left\{ r \in \{-1,0,1\} \;\middle|\; \begin{array}{lll} \min\limits_{F^t \in R(F)} \mathcal{S}_n(F^t) \leq r \leq \max\limits_{F^t \in R(F)} \mathcal{S}_n(F^t) & \text{if} & x_n \in \vec{\theta_n} \\ \min\limits_{F^t \in R(F)} \mathcal{S}_n(F^t) < r < \max\limits_{F^t \in R(F)} \mathcal{S}_n(F^t) & \text{if} & x_n \notin \vec{\theta_n} \end{array} \right\} \tag{3.7}$$

Proof 1 *see [3]*

The qualitative behavior of the hybrid system can be visualized as a directed graph. In this graph, each node is a flow domain. If any trajectory corresponding to a solution of the hybrid system exits a flow domain to enter another, an edge runs from the node corresponding to that flow domain towards the node corresponding to the target flow domain. Now conclusions drawn from the properties of this graph can be reasoned back to possible solution trajectories through the state space of the hybrid system.

To find the edges which connect the graph, one has to find out if a flow domain can be reached from another.

Theorem 2 *If for some solution ξ to the PWA system, $x(t) \in F^a$ for all $t \in [\tau_1, \tau_2)$ and $x(\tau_2) \in F^b$, the node corresponding to F^a is connected to node corresponding to F^b with an edge directed from node F^a to node F^b. F^b is reachable from F^a in one step in the graph.*

Intermezzo 1 *The qualitative abstraction is complete in the sense that a path in the graph exists for every possible trajectory in the real system. The converse is not true, however. Paths could exist in the graph which represent trajectories not feasible in the real system. As an example, consider a domain in which the differential equations are reduced to the set shown in equation 3.8. In figure 3.3 a quiver plot in this domain is shown. It can be seen that from F0, F1 can be reached and from F1, F2 can be reached.*

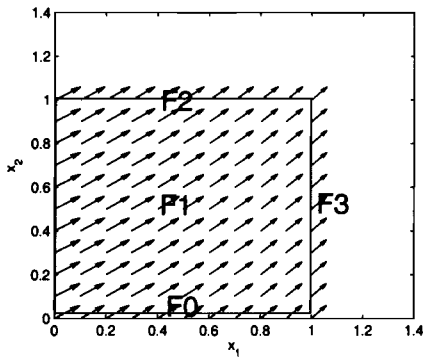


Figure 3.3: example where the qualitative transition graph is over-complete

However, it is impossible for a trajectory to pass from F0 to F1 to F2 in that order. All trajectories entering F1 from F0 will go to F3.

$$\begin{aligned} \dot{x_1} &= -2 \cdot x_1 + 5 \\ \dot{x_2} &= 3 \end{aligned} \tag{3.8}$$

Finding all of the edges in the graph by trying all possible ξ is impractical. Therefore transition rules need to be devised which simplify this procedure to checking if certain rules hold for adjacent flow domains and if so, a transition is possible. These rules are a combination of the rules for transitions as proposed in previous work by de Jong et.al. and rules from classical state space analysis.

Definition 22 *Transition rules:*

Transition from F to F', where $F' \in A(F)$

Transition to a lower dimensional flow domain. $\vec{x} \in F$, $\vec{y} \in F'$.

1. $\mathcal{S}(F) \neq \emptyset$,
2. *If* $F \in \mathcal{F}_n$, $\langle \forall n : S_n(\vec{x}) = 0 : \langle \forall m : m \neq n \wedge m \neq 0 : k_{n,m} = 0 \rangle \rangle$
3. $\langle \forall n : W_n(F, F') \neq 0 \wedge S_n(\vec{y}) \neq 0 : S_n(\vec{x}) \cdot W_n(F, F') = 1 \rangle$
4. *If* $\langle \nexists n :: W_n(F, F') \neq 0 \wedge S_n(\vec{y}) \neq 0 \rangle$, *then* $\langle \forall n :: (W_n(F, F') = 0 \wedge S_n(\vec{y}) = 0) \vee (W_n(F, F') \cdot S_n(\vec{x}) = 1) \rangle$

The first rule says that a lower dimensional flow domain can only be entered from an $F \in \mathcal{F}_s$ if there is a sliding mode in that flow domain. The second rule says the same for an $F \in \mathcal{F}_n$. The third rule says that to enter a lower dimensional flow domain, the trajectories in F should tend towards F' for all state variables. Also, when $F' \in \mathcal{F}_n$, the state variables which have a solution to the nullcline equalities have no influence on whether the

nullcline flow domain can be reached. The fourth rule says that a nullcline flow domain can be entered too, if all state variables that are parallel to the target domain do not change and all others tend toward that domain.

Transition from F to F', where $F \in A(F')$

Transition to a higher dimensional flow domain. $\vec{x} \in F$, $\vec{y} \in F'$.

1. $\mathcal{S}(F') \neq \emptyset$
2. *If* $F' \in \mathcal{F}_n$, $\langle \forall n : S_n(\vec{x}) = 0 : \langle \forall m : m \neq n \wedge m \neq 0 : k_{n,m} = 0 \rangle \rangle$,
3. $\langle \forall n : W_n(F, F') \neq 0 : S_n(\vec{y}) \cdot W_n(F, F') \neq 1 \rangle$

The first and second rule say that to enter a higher dimensional flow domain, that flow domain must have a sliding (or regular) mode. The third rule says that in order to exit domain F, there must be no trajectories which tend towards F.

Chapter 4

Parameter Estimation for PWA Approximation of Nonlinear Systems

In this chapter, the procedure for parameter identification will be shown as well as how to use the information from the qualitative simulation algorithm described in chapter 3.

The first step to this identification procedure is to transform the nonlinear system to a PWA system. This can be done by using a fitting procedure to find a PWA approximation for a nonlinear part, or by visual inspection of measurement data to identify (nearly)linear working areas. This results in a description of the system as a PWA system as defined in equation 2.8.

Now the identification problem is transformed into identification of a PWA system. This identification procedure consists of two parts. First, each data point needs to be classified into a mode of the PWA system. Then, for each mode, a linear identification needs to be carried out, which results in an estimate for the parameters of the differential equations.

For identification a least squares optimization algorithm is used. The cost function to be minimized is:

$$J(\vec{\theta},\vec{l}) = \sum_{D^i \in \mathcal{D}_r} \sum_{\vec{x}(k) \in D^i} \left\| \frac{\vec{x}(k+1) - \vec{x}(k)}{\delta t} - f(\vec{x}(k), \vec{\theta}, \vec{l}) \right\|^2 \tag{4.1}$$

Where $f(\vec{x})$ is the PWA function found, like equation 2.8, and $\vec{x}(k)$ are measured values with a sample interval of δt. Now, the thresholds and the parameters are found by respectively:

$$\vec{\theta} = \arg\min_{\vec{\theta}_m} J(\vec{\theta}_m) \tag{4.2}$$

$$\vec{l} = \arg\min_{\vec{l}} J(\vec{\theta},\vec{l}) \tag{4.3}$$

In equation 4.3, the $\vec{\theta}$ found by minimizing equation 4.2 is used.

These two minimizations need to be separated because $J(\vec{\theta}, \vec{l})$ is a piecewise constant function in $\vec{\theta}$. Because of the fact that this value of J only changes if a data point switches domain due to a change in $\vec{\theta}$. This also means that the minimization with respect to $\vec{\theta}$ is not possible using standard least squares optimization techniques.

4.1 Clustering of Data points and Parameter Estimation

The first step in parameter estimation of a PWA system is the assignment of data points to modes of the system. Due to the structure of the class of PWA systems used, this clustering is relatively easy. As the mode of the PWA system is determined by the relative position of each state variable's value with respect to the thresholds defined for that state variable, clustering can be done by analyzing each state variable's value individually. The cost function equation 4.1 is nonlinear in $\vec{\theta}$ which causes equation 4.2 to be a nonlinear optimization problem.

There is no guaranteed way to solve this problem due to the shape of the cost function. Therefore an exhaustive search algorithm should be used to ensure a global minimum is found. Exhaustive search tends not to scale very well with the number of dimensions of the system and the number of modes of the system. For the class of PWA systems used, however, the number of possibilities can be reduced significantly.

Due to the fact that the mode is determined by the values of each state variable independently, the number of possibilities to be checked during an exhaustive search is limited. To calculate the number of threshold positions to be tested, a derivation of all possibilities is given. Assume there are L measurements of N variables and each state variable has M intervals, caused by $M-1$ thresholds. If a minimum of k data points between each two thresholds is desired, the number of possibilities that need to be inspected is:

$$N \cdot \begin{pmatrix} L - k \cdot M + M - 1 \\ M \end{pmatrix} \tag{4.4}$$

This follows from the fact that both the data points and the thresholds are already ordered. With this knowledge, finding a possible division of a single state variable reduces to finding any possible multiset containing $L - k \cdot M$ items, each is any of M possible symbols. A multiset is set-like, but ordering is ignored and therefore only the number of occurences of symbols is important. When the contents of the multiset is ordered, each item corresponds to a data points and it's symbol corresponds to which threshold this data point belongs. This is a standard combinatorial counting

problem which is solved by the multichoose operator. This operator takes two arguments (k,n) and the result is the number of possible multisets of length k on n symbols. By substituting k with $L - k \cdot M$ and n by M, equation 4.4 is found.

For example, if there are 150 data points and 2 thresholds in each of 3 dimensions and at least 3 data points per interval, the number of possible threshold positions is 1431573. This approach is feasible for a limited number of data points and thresholds, which is fairly common in biochemical measurements. Different approaches are possible, such as e.g. a Bayesian approach [5], but these are beyond the scope of this paper.

4.2 Identification of the Parameters

The second step of identification of the system consists in doing a parameter identification on the ODE's for each mode of the system. Usually many modes share differential equations for some state variable, as not all differential equations change on each threshold crossing. Therefore, a simultaneous optimization is used to estimate all parameters in one go. When the positions of all thresholds are given, equation 4.3 is a linear least squares optimization problem.

In MATLAB[1] the **lsqlin** function is provided for linear least squares optimization. This function is able to use certain forms of a-priori information to bound the search space for parameters.

$$\min_{\vec{k}} \frac{1}{2} \left\| \mathbf{C}\vec{k} - \vec{d} \right\|_2^2 \tag{4.5}$$

$$\text{such that :} \quad \mathbf{A}\vec{k} \leq \vec{b} \tag{4.6a}$$

$$\mathbf{A_{eq}}\vec{k} = \vec{b_{eq}} \tag{4.6b}$$

$$\vec{lb} \leq \vec{k} \leq \vec{ub} \tag{4.6c}$$

The cost function which is minimized by **lsqlin** is shown in equation 4.5, the equation which is solved by **lsqlin**. In equation 4.6a to equation 4.6c, the available equations to limit the parameter search space are shown. As these equations are the only way to introduce a-priori knowledge about the system into the parameter estimation procedure, the goal is to convert as much available a-priori information as possible into these (in)equalities. In table 4.1 a number of possible restrictions which can be applied using the constraint equations.

In biological systems, most a-priori information comes in the form of some knowledge about the behavior of the system. As this information in

[1] ©1984-2005 The MathWorks, Inc.

equation 4.6a: $k_n \leq k_m$

equation 4.6a: $k_n + k_m + \ldots \leq k_z$

equation 4.6a: $\frac{k_n}{k_m} \leq a$

equation 4.6b: $k_n = a \cdot k_m$

equation 4.6b: $k_n = k_m$

equation 4.6c: $lb_n \leq k_n \leq ub_n$

Table 4.1: Usable a-priori information for the Matlab[1] **lsqlin** function

itself is not usable to constrain the parameter search space, the qualitative simulation algorithm introduced in chapter 3 will be used to derive this information. The simulation algorithm is able to derive a graph containing all possible trajectories in the state space of a certain PWA system, based on the structure of the PWA system and a set of restrictions on the parameters.

From the set of information on the behavior of the system, a set of features of the graph can be derived. For example, it is known that some oscillation occurs in which certain state variables take part and some other state variables remain at a constant level. This implies that the graph contains a cycle through a set of nodes in which the oscillating state variables change and the non-oscillating state variables remain constant. As another example, a system could have some unreachable subspaces when some initial condition is satisfied, therefore, the graph should not contain a trajectory starting from any node containing (part of) the initial condition towards any node contained in the forbidden subspaces.

There might be many possible graphs which can be devised to conform to the available set of a-priori information. As each of these graphs is devised based on the a-priori information, each of the graphs conforms to this information.

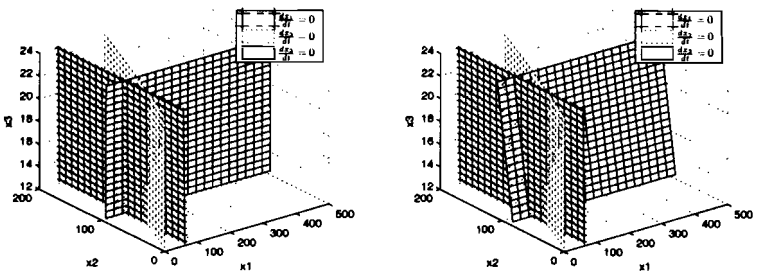
The qualitative simulation algorithm creates a graph based on both the structure of the PWA system and a set of restrictions. These restrictions define the signs of the differential equations at 0-dimensional switch domains and the ordering of intersecting flow domains. The combination of the PWA system description and the restrictions will generate an unique graph. Unique in the sense that this graph is the single possible graph, given the PWA system and restrictions. As this graph contains all possible trajectories a solution of the PWA system can follow, the fact that a certain path exists in the graph implies that, with the right parameters, a solution to the PWA system follows a trajectory corresponding to the path in the graph.

Demands on the absence of a trajectory is more difficult, as due to intermezzo 1, there might be paths in the graph which correspond to impossible trajectories, depending on the actual value of the parameters. This is impossible to check without knowing the parameters of the PWA system. Therefore, it should be detected during qualitative simulation if certain paths suffer from this condition.

If there are paths in the graph which should not be possible from a-priori information, then if these paths are introduced due to the over-completeness of the graph, the graph is still a good solution. This combination of PWA system and restrictions should then be checked too.

The set of restrictions is usually very large. As not all of this information might be available, the only way to allow the use of this simulation method is to try any possible combination of restrictions for all unavailable information. If the resultant graph is valid with respect to the known a-priori information, the set of restrictions apply for the system to be identified. To keep this process feasible, automation is necessary. By defining the a-priori information as a set of propositions on either the set of information for defining the simulation or the properties of the resultant graph, the process can be carried out by computer algorithms.

After this procedure, the set of all graphs which can be generated from the known a-priori information is available and for each graph the set of information needed to generate this graph. As all of these graphs are valid with respect to the available a-priori information, the set of parameters to be found should conform to one or a union of some of the sets of information used to generate the graphs and this information can be used to constrain the parameters during identification.



(a) $\theta_1 = \theta_2 = 12$ (b) $\theta_1 = 12$ and $\theta_2 = 24$

Figure 4.1: Nullcline planes for one and two thresholds

4.3 Example: Biochemical Oscillator

This second example is a more complex system, where qualitative simulation can be used to optimize identification. The system analyzed is a biochemical oscillator which describes a genetically regulated enzymatic conversion of a substrate into a product, which is also treated in [6]. The process can be described by nonlinear differential equations:

$$\begin{aligned} \dot{x}_1 &= \frac{V_1}{k_{m1}+x_3} - k_{d1} \cdot x_1 \\ \dot{x}_2 &= k_2 \cdot x_1 - k_{d2} \cdot x_2 \\ \dot{x}_3 &= k_3 \cdot x_2 - \frac{V_2 \cdot x_3}{k_{m2}+x_3} \end{aligned} \tag{4.7}$$

Both nonlinear functions in these differential equations are converted to PWA functions as proposed in table 2.1. This conversion to a hybrid system introduces two steps with thresholds dependant on x_3. The differential equations of this hybrid system become:

$$\begin{aligned} \dot{x}_1 &= l_{1,0} + l_{1,1} \cdot x_3 + s^+(x_3, \theta_1) \cdot (l_{1,2} + l_{1,3} \cdot x_3) - k_{d1} \cdot x_1 \\ \dot{x}_2 &= k_2 \cdot x_1 - k_{d2} \cdot x_2 \\ \dot{x}_3 &= k_3 \cdot x_2 - (l_{3,0} + l_{3,1} \cdot x_3 + s^+(x_3, \theta_2) \cdot (l_{3,2} + l_{3,3} \cdot x_3)) \end{aligned} \tag{4.8}$$

As these two thresholds are on x_3, it is possible that they might be joined into a single threshold. To find out whether this is possible, a qualitative simulation of the system with two thresholds and with a single threshold must be carried out. Because of the very large amount of flow domains involved in this simulation, only the results are presented.

By using available values for V_1, k_{m1}, V_2 and k_{m2} found in literature a fit can be made using the algorithm presented in section 2.3. From the results of this fit the conclusion can be drawn that it always holds that $\theta_1 < \theta_2$. To visualize the qualitative simulation, a part of the phase space of the hybrid system is shown in figure 4.1. For the space shown it holds that $\theta_1 \leq x_3 \leq \theta_2$, $x_1 \in \Omega_1$ and $x_2 \in \Omega_2$. The planes shown are the nullcline planes for each

Table 4.2: Parameter identification

Parameter	$\sigma = 0$	$\sigma = 1$	$\sigma = 2.5$	$\sigma = 5$
θ	12.37	12.12	12.51	14.34
$l_{1,0}$	55.08	55.24	55.94	52.04
$l_{1,1}$	-2.91	-2.92	-3.10	-2.37
$l_{1,2}$	-47.58	-47.63	-48.30	-44.30
$l_{1,3}$	2.86	2.87	3.05	2.32
$l_{3,0}$	5.57	5.57	5.61	6.24
$l_{3,1}$	0.052	0.053	0.053	0.052
$l_{3,2}$	3.71	3.74	3.67	3.00
$l_{3,3}$	-0.047	-0.048	-0.048	-0.048
k_{d1}	0.11	0.11	0.11	0.11
k_2	0.10	0.10	0.10	0.10
k_{d2}	0.10	0.10	0.10	0.10
k_3	0.10	0.10	0.10	0.10

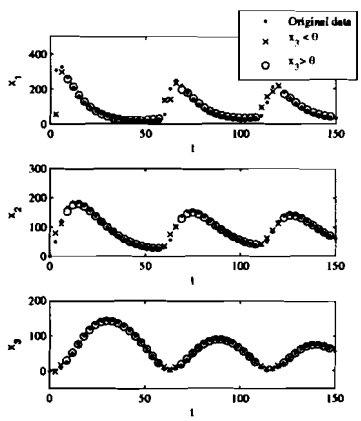


Figure 4.2: One step ahead prediction

variable. It can be seen that the qualitative behavior in figure 4.1(a) will be the same as in figure 4.1(b). Therefore the conclusion can be drawn that in identification the thresholds can be joined together, without losing the characteristic limit cycle.

To provide experimental data, the nonlinear system of equation 4.7 is simulated using the ODE45 solver of MATLAB. Several levels of white noise are added to simulate various amounts of measurement noise. Parameters of the hybrid model were estimated using the two-step procedure as presented in chapter 4. The results of parameter estimation are shown in table 4.2.

Chapter 5

Discussion and Further Research

This paper shows that it is feasible to use a piecewise linearization method for identification of nonlinear biochemical systems. A useable class of PWA systems is presented as well as a method to convert any known nonlinear system into a PWA system. A method to integrate most a-priori information into the identification procedure is provided using a qualitative simulation system.

Using a PWA system to represent a nonlinear system for identification has several advantages to direct identification of a nonlinear model on a nonlinear system. First of all, as the actual identification uses a linear least squares optimization, no initial estimate is necessary to find the global minimum in optimization. This also leads to better identification results in case little measurement data is available. On the other hand, the division of the state space into modes diminishes this advantage, as the data needs to be divided over all modes. With little data available, the PWA approach should still perform better than direct nonlinear identification, as long as the number of modes is reasonably small.

A major advantage of using a PWA systems approach comes from the qualitative simulation algorithm presented in this paper. This algorithm allows most a-priori available information to be used during the identification process. This is especially useful when identifying biochemical systems as extensive studies on behavior are usually available. In nonlinear identification no method exists to use all of this a-priori information in identification. The algorithm has been partially implemented in C and estimations of completion times for simulation show that it is usable during identification procedures.

There are also a few downsides to this approach. The parameters in nonlinear models for biochemical processes usually have a physiological meaning. This meaning is lost in the conversion to a PWA system. Another problem

may lie in the introduction of many new parameters, which could lead to ambiguity as the system may be described using less parameters.

The qualitative simulation system can be useful in various different situations. A use of the system is to assess whether some thresholds might be eliminated without changing the behavior of the system. If one finds that the behavior of the system does not change when combining two adjacent thresholds, these thresholds can be combined in the identification procedure. By eliminating unnecessary regular domains, more data points will be available per domain, which results in a better chance of a good identification.

Further research should focus on several subjects related to the results presented in this paper. The qualitative simulation algorithm can be implemented in software, on which work has started already. When this implementation is finished, results will become available on the performance of the algorithm. Then results on the usability in practical situations will be found.

On the area of applicability of a piecewise approximation for nonlinear systems and techniques to diminish the downsides of these approximations more research is necessary. Most notably, the dependence of identification on the choice of linearization intervals and the deviation between linearized functions and the nonlinear functions near thresholds. The choice of linearization intervals is especially interesting with respect to comparing results of several identification rounds of the same (or slightly altered) systems. Diminishing influence of deviation of linearized functions near thresholds should improve identification results in general.

Appendix A

Software Implementation for Qualitative Simulation

The qualitative simulation algorithm as presented in chapter 3 can be implemented in software. A start has been made on the implementation of this algorithm in the C programming language. Uses some of the concepts of the implementation presented in [4]. The choice of reimplementation in C has been made primarily on the fact that C produces comparatively fast software, which is required for this method to be of practical relevance for system identification.

A.1 Data Types

To ease the implementation of data types containing all information needed for simulation, as well as parsing of files, a library called GLib [10] is used. GLib is an open-source library which was created as an utility library for GTK+ and GNOME. It provides a large set of utility functions ranging from memory allocation, string handling and some advanced type handling for linked lists, trees and more. It also provides support for threaded applications, signals and file, socket and pipe IO. The library compiles and runs on various operating systems including UNIX, Linux, Win32 and several others. This is important, as it will allow the simulation system to run on almost any operating system in use.

There are several types of information to be stored in memory. First of all, the PWA model is stored. This model is read from a file containing all a-priori information necessary. The syntax of this file is shown in table A.1. In this syntax **bold** text is to be included literally, [...] means any of ..., a + after an item means it must occur at least once and a ? means zero or one occurrence. This format is read using the lexical scanner (GScanner) from GLib, and stored in several structures in memory. The main model is stored in a type BPWAAModel. BPWAA stands for Biological PWA Analyzer. In

state_variables: (*variable*)+

(variable *variable* **{**
parameters: (*parameter*)+
thresholds: (*threshold*)+
equation: *equation*;
equilibrium {(*equilibrium_inequality*)+}
})+

variable:=*token*
parameter:=*token*
threshold:=*token*
token:=[**a-zA-Z0-9_**]+
equation:=*equation_term* (+ *equation_term*)+
equation_term:=[+-]*parameter*(* *step_function*)+ (* *variable*)?
step_fuction:=**s**[+-](*variable*,*threshold*)
equilibrium_inequality:=*equilibrium_term*[<>] **0**;
equilibrium_term:=[+-]*parameter** *threshold*

Table A.1: File syntax for qualitative simulation of a PWA model

```
struct StateVar {
  BPWAAModel* model;
  gchar * name;
  GPtrArray *parameters;
  GPtrArray *thresholds;
  Equation *equation;
  GPtrArray *equilIneq;
};
```

Table A.2: Statevar

current implementation, this is an array of state variables, which are stored in StateVar's (table A.2).

The StateVar structure consists of a backpointer to the BPWAAModel, the name of the state variable, the list of parameter names in the equation of that state variable, a list of threshold names for that variable, the equation and a list of equilibrium inequalities. The parameters and thresholds are both arrays of gchar*, which store the names of respectively the parameters and thresholds. Further reference to these parameters and thresholds are done by indexing in these arrays.

The equation is a more complex structure. It is stored as an array of

```
struct EquationTerm {
  gint  sign;
  gint  stateVar;
  GSList *  steps;
};
```

Table A.3: EquationTerm

```
struct EquationStep {
  gint sign;
  gint stateVar;
  gint threshold;
};
```

Table A.4: EquationStep

```
struct _EquilibriumInequality {
  gint8* parameterMask;
  gint* thresholds;
  gboolean lessThanZero;
};
```

Table A.5: EquilIneq

terms, which correspond to the math accompanying each parameter in the equation. From equation 2.8 it can be seen that each part consists of a sign, a s_m^n and an element of $\vec{x}$, i.e. a single state variable or a one. At this point, only AND'ing of several step functions is supported, but support for more complex combinations is easily implemented.

The second more complex part of the model description are the equilibrium inequalities. These inequalities are used during derivation of sign vectors in the regular domains and specifically, the locations of nullcline flow domains. The inequalities are stored as follows. Each inequality has a set of properties, namely, the direction of inequality ($<$ or $>$), a subset of the parameter set and the thresholds at which the inequality holds. This information is stored in the EquilIneq structure.

Now all structures needed to store the model are present, a few structures are needed to store the subdivision of the state space as required for simulation. This division is done in two levels. First the state space is divided into domains, which are then split into flow domains, as needed. All of the information is kept together using the BPWAASimulationResults (table A.6) structure. This structure holds, in order, a backpointer to the

```
struct BPWAASimulationResults {
  BPWAAModel* model;
  GMemChunk* vector_memory;
  GMemChunk* domain_memory;
  BPWAADomain** domains;
  gint* dimensionSizes;
  guint64 amount;
};
```

Table A.6: BPWAASimulationResults

```
struct BPWAADomain {
  gint* coordinates;
  gint* direction;
  gboolean direction_valid;
  GList* flowdomains;
};
```

Table A.7: BPWAADomain

associated model, two memory allocators, used to efficiently store the large amounts of equally sized blocks, an array containing all domains, a vector used to convert the coordinates of a domain into an index in the array of domains and finally the number of domains in the state space.

The domains are stored in a structure as shown in table A.7. Here, the coordinates of the domain, the calculated sign vector, a boolean indicating validity of the sign vector and a list of flow domains included in the domain, is stored. The state division of the state space can be seen as a N-dimensional matrix, where each element of the matrix corresponds to a domain. As it is very dificult and inefficient to directly store this matrix, a mapping is made to convert the coordinates of each element of the matrix into an index of the domain array.

The flow domains are a more difficult to store, as these do not have such a strict ordering as the mode domains. These flow domains are stored in the FlowDomain structure. This structure contains a backpointer to the domain it is contained in, a vector containing the sign vector, a boolean indicating the direction has been calculated, the $A(F)$ set, the $R(F)$ set, a list of successors and predecessors. These successors and predecessors are other flow domains, indicating outgoing edges in the graph and incoming edges in the graph.

The graph being sought is actually stored in the flow domain structure. Each node of the graph corresponds to one of the FlowDomain structures and the successors and predecessors list all edges connected to that node.

```
struct FlowDomain {
  BPWAADomain *container;
  gint* direction;
  gboolean direction_valid;
  GList* adjacent;
  GList* regulars;
  GList* successors;
  GList* predecessors;
};
```

Table A.8: FlowDomain

A.2 Algorithms

Currently, the implemented software is able to find which domains contain nullcline flow domains and calculate directions within switch domains. The final steps of the simulation, division of the domains containing nullcline flow domains and finding the edges of the graph still needs to be implemented. The complete path from a datafile to the final graph can be divided into a number of steps. In the following list, each step is shown, along with it's status of implementation.

- Parsing of the datafile and storage of the model
  Status: Finished
  To do: More error checking

- Representation of the state space and flow domains
  Status: Finished
  To do: Nothing

- Calculation of signs in 0-dimensional switch domains
  Status: Finished
  To do: Possibly some optimization possible

- Inference of intersections of nullcline flow domains with switch domains
  Status: Finished, but only for 1-dimensional switch domains
  To do: Implementation for higher-dimensional switch domains

- Calculation of division of domains due to nullcline flow domains in all possible orderings
  Status: Not implemented
  To do: Implement

- Calculation of sign vectors for flow domains which envelop a whole domain

Status: Finished
To do: Nothing

- Calculation of sign vectors for other flow domains
  Status: Not implemented
  To do: Implement
- Derivation of first successor for each flow domain
  Status: Not implemented
  To do: Implement
- Output of suitable data for graph analysis
  Status: Not implemented
  To do: Implement

A.2.1 Finding Regular Domains Containing Nullcline Flow Domains

To find regular domains D containing nullcline flow domains, the sign of each of the differential equations, which govern the system in that domain, is needed at the extremities of that domain.

The first step is finding which of the parameters are active, i.e. $s_m^n = 1$, in that domain. This produces a mask, such as the mask stored in each of the equilibrium inequality structures. Then, at the coordinates of each of the 0-dimensional switch domains in $A(D)$, the corresponding equilibrium inequalities are checked to find the sign of the differential equation at that corner of the domain. If each corner has the same sign, there is no nullcline domain inside that regular domain, otherwise, there is.

A.2.2 Inference of Intersections of Nullcline Flow Domains with Switch Domains

This step uses the same ideas as the search for regular domains containing nullcline flow domains. For all 1-dimensional switch domains D, which have a regular domain D' containing nullcline flow domains in their set $R(D)$, we check the sign of the differential equations of all possible D' at the coordinates of the two switch domains in $A(D)$. If these differ, we know that the corresponding nullcline domain in D' intersects D. Repeating this procedure for all D' results in a list of intersections. Together with an ordering, all flow domains contained in D can be formed, together with their respective A and R sets.

A.2.3 Further Algorithms

The rest of the simulation algorithm either needs an actual implementation or is not suitable for textual explanation. In chapter 3 the unimplemented

algorithms are already described. For the rest of the algorithms used, please see to the annotated source code.

A.3 Performance

As the qualitative simulation will usually be carried out on systems with large numbers of flow domains, the performance of the algorithm is very important. For example, the single threshold system used in section 4.3 already consist of more than 150 flow domains. The system used for testing of the software implementation consists of more the 27000 mode domains, and the number flow domains are a multitude of this number. Note that this test-system is unusually large, due to the large amounts of thresholds. Therefore, it is likely to be a good worst case candidate.

At this point, the whole algorithm cannot be tested for speed, but preliminary results on the current implementation look promising. The finding of domains containing nullcline flow domains runs at about 20000 domains per second. The second step, finding the intersections of the flow domains, can be carried out simultaneously without much additional calculations. It is unclear how much calculations will be necessary for the final part of the algorithm, but these will probably take an amount of calculations of the same order as the first half of the algorithm. It is hard to estimate the total rate of calculation, yet based on some reasonable estimations, a rate of more than 5000 domains per second should be achievable.

The number of flow domains which will need to be analyzed depends heavily on the structure of the PWA model. Yet the current implementation has already been shown to be much faster than the implementation of the GNA algorithm as presented in [4]. To provide some perspective, the currently implemented steps are carried out in just over a second for the test-system, where the MATLAB[1] implementation would take several hours to complete. Also the memory usage is less by an order of magnitude. These properties give a good indication that the simulation algorithm will complete within a reasonable amount of time, which is very important for the practical use of the tool in identification.

[1] ©1984-2005 The MathWorks, Inc.

Bibliography

[1] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, M. Page, and D. Schneider. Qualitative analysis and verification of hybrid models of genetic regulatory networks: Nutritional stress response in escherichia coli. In 8th *International Workshop on Hybrid Systems: Computation and Control (HSCC 2005)*, pages 134–150, 2005.

[2] H. de Jong, J. L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Hybrid modeling and simulation of genetic regulatory networks: A qualitative approach. In *Proceedings HSCC 2003*, pages 267–282, 2003.

[3] H. de Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. In *Bull. Math. Biol.*, volume 66, pages 336–344, 2004.

[4] D. J. W. Lindenaar. Simulation of qualitative behavior of regulatory networks. Internship Report, Technische Universiteit Eindhoven, E-MBS-CS, December 2004.

[5] A. Lj. Juloski. *Observer Design and Identification Methods for Hybrid Systems.* Technische Universiteit Eindhoven, 2004.

[6] M. W. J. M. Musters, D. J. W. Lindenaar, A. Lj. Juloski, and N. A. W. van Riel. Hybrid identification of nonlinear biochemical processes. In *Proceedings* 16th *IFAC Symposium on System Identification*, 2005.

[7] M. W. J. M. Musters and N. A. W. van Riel. Analysis of the transforming growth factor-β_1 pathway and extracellular matrix formation as a hybrid system. In *Proceedings* 26th *IEEE Engineering in Medicine and Biology Society*, pages 2901–2094, 2004.

[8] N. A. W. van Riel, T. Ivanics, L. Ligeti, and G. J. van der Vusse. System identification to analyze changed kinetics of serca in intact rat heart. In *Proceedings* 5th *IFAC Symposium on Modelling and Control in Biomedical Systems*, pages 123–128, 2003.

[9] N. A. W. van Riel, A. Lj. Juloski, and J. op den Buis. One-step ahead prediction for parameter estimation in physiological hybrid models. Submitted to 16th IFAC Symposium on System Identification, 2005.

[10] *GLib Reference Manual.* http://developer.gnome.org/doc/API/2.0/glib/index.html.