Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Interval Based Parameter Identification for System Biology

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Mohsen Alami

LiTH-ISY-EX--12/4545--SE

Linköping 2012

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Interval Based Parameter Identification for System Biology

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av

Mohsen Alami

LiTH-ISY-EX--12/4545--SE

Handledare: **Sina Khoshfetrat Pakzad**
ISY, Linköpings universitet

Examinator: **Torkel Glad**
ISY, Linköpings universitet

Linköping, 25 January, 2012

Avdelning, Institution Division, Department Division of Automatic Control Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	**Datum** Date 2012-01-25

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ______________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ______________	**ISBN** — **ISRN** LiTH-ISY-EX--12/4545--SE **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://www.control.isy.liu.se
http://www.ep.liu.se

Titel
Title

Intervallbaserad parameteridentifiering för systembiologi

Interval Based Parameter Identification for System Biology

Författare
Author

Mohsen Alami

Sammanfattning
Abstract

This master thesis studies the problem of parameter identification for system biology. Two methods have been studied. The method of interval analysis uses subpaving as a class of objects to manipulate and store inner and outer approximations of compact sets. This method works well with the model given as a system of differential equations, but has its limitations, since the analytical expression for the solution to the ODE is not always obtainable, which is needed for constructing the inclusion function. The other method, studied, is SDP-relaxation of a nonlinear and non-convex feasibility problem. This method, implemented in the toolbox BIO.SDP, works with system of difference equations, obtained using the Euler discretization method. The discretization method is not exact, raising the need of bounding this discretization error. Several methods for bounding this error has been studied. The method of ∞-norm optimization, also called worst-case-∞-norm is applied on the one-step error estimation method.

The methods have been illustrated solving two system biological problems and the resulting $\mathbb{SCP}$s have been compared.

Nyckelord
Keywords

Interval analysis, SIVIA,IMAGESP, parameter identification, error estimation, SDP-relaxation, infeasibility certificate

Abstract

This master thesis studies the problem of parameter identification for system biology. Two methods have been studied. The method of interval analysis uses subpaving as a class of objects to manipulate and store inner and outer approximations of compact sets. This method works well with the model given as a system of differential equations, but has its limitations, since the analytical expression for the solution to the ODE is not always obtainable, which is needed for constructing the inclusion function. The other method, studied, is SDP-relaxation of a nonlinear and non-convex feasibility problem. This method, implemented in the toolbox BIO.SDP, works with system of difference equations, obtained using the Euler discretization method. The discretization method is not exact, raising the need of bounding this discretization error. Several methods for bounding this error has been studied. The method of ∞-norm optimization, also called worst-case-∞-norm is applied on the one-step error estimation method.

The methods have been illustrated solving two system biological problems and the resulting $\mathbb{SCP}$s have been compared.

Sammanfattning

Det här examensarbetet studerar problemet med parameteridentifiering för systembiologi. Två metoder har studerats. Metoden med intervallanalys använder union av intervallvektorer som klass av objekt för att manipulera och bilda inre och yttre approximationer av kompakta mängder. Denna metod fungerar väl för modeller givna som ett system av differentialekvationer, men har sina begränsningar, eftersom det analytiska uttrycket för lösningen till differentialekvationen som är nödvändigt att känna till för att kunna formulera inkluderande funktioner, inte alltid är tillgängliga. Den andra studerade metoden, använder SDP-relaxering, som ett sätt att komma runt problemet med olinjäritet och icke-konvexitet i systemet. Denna metod, implementerad i toolboxen BIO.SDP, utgår från system av differensekvationer, framtagna via Eulers diskretiserings metod. Diskretiseringsmetoden innehåller fel och osäkerhet, vilket gör det nödvändigt att estimera en gräns för felets storlek. Några felestimeringsmetoder har studerats. Metoden med ∞-norm optimering, också kallat worst-case-∞-norm är tillämpat på ett-stegs felestimerings metoder.

Metoderna har illustrerats genom att lösa två system biologiska problem och de accepterade parametermängderna, benämnt $\mathbb{SCP}$, har jämförts och diskuterats.

Acknowledgments

I would like to thank my examiner professor Torkel Glad for giving me the opportunity of writing this thesis. I'm specially thankful to Sina Khoshfetrat Pakzad, for supervising my thesis and specially for his guidance and patience throughout the whole work.

I also want to thank Johan Löfberg for his help with the toolbox Yalmip. Thanks also to Gunnar Cedersund at the ISB Group and Jan Hasenauer at the university of Stuttgart Germany, who all run similar projects in parallel. Thanks to Pelle Lundberg for providing the system biological models and data.

Many thanks go also to Sebastian Tornil-Sin for sending me his toolbox scs and always answering my questions. Last but not least, many thanks to Gustav Hendeby, the developer of the LaTeX-class, `liuthesis`. With his work he has made the life of many students much easier than it would have been.

Contents

Notation

In the following, the abbreviations and notations are listed and explained.

Notation	Meaning
$\mathbf{x}_C$	Vector of the time-continous state variables
$\mathbf{x}_D$	Vector of the time-discrete state variables
$\mathbf{y}_C$	Vector of the time-continuous model outputs
$\mathbf{y}_D$	Vector of the time-discrete model outputs
$\mathbf{u}_C$	Vector of the time-continuous model inputs
$\mathbf{u}_D$	Vector of the time-discretes model inputs
$\mathbf{p}$	Vector of the unknown parameters
$\mathbf{e}_D^{(k)}$	Vector of the discretization error at the time instant k.
$\mathtt{dom}f$	Domain of a function, f.
$\mathcal{X}$	Set of the state vectors
$\mathcal{Y}$	Set of the output vectors
$\mathcal{U}$	Set of the input vectors
$\mathcal{P}$	Set of the unknown parameters
P_0	Initial set or initial box
$\mathcal{P}^{CT}$	Set of parameters consistent with the time-continuous dynamical model
$\mathcal{P}^{DT}$	Set of parameters consistent with the time-discrete dynamical model
$I(0,N)$	Denoting the integer set $\{0,1,...,N\}$
$N+1$	Number of the measurements
$\mathbb{R}$	Set of all real numbers
$\mathbb{R}^n$	Set of all real vectors
$\mathbb{R}^{n_x}$	Set of all real vectors with the dimension equal to n_x=number of states. The same is applicable for n_y, n_u and etc.
$\mathbb{IR}$	Set of all interval real numbers
$\mathbb{IR}^n$	Set of all n-dimensional rea interval vectors
$\mathbb{SCP}$	Set of Consistent Parameters

Chapter 1

Introduction

Model validation and parameter estimation is a challenging task, specially because of the uncertainty in the parameters and measurement data. Classical methods based on statistical and regression methods require a huge amount of experimental data, in order to be efficient for the purpose of validation. The subject of this thesis is parameter identification for biological systems, where the amount of available measurement data is limited to some few samples. As a result it is not possible to use these classical methods. This is why set-based parameter identification approach is more efficient, because by set-based methods it is possible to analyze complete sets of data instead of finite number of distinct points [8, 17, 4]. Even if the restriction regarding the limitation on measurement data had not been a problem, it would have been difficult to validate the correctness of a model, based on measurements taken from the system. Also, deriving a mathematical proof that verifies a model is impossible in practice [17]. These are the reasons why it seems to be more realistic to treat the converse problem, *model falsification*. Related works in the field treat the problem in a similar way, namely by obtaining an infeasibility certificate using set-based approaches instead of trying to develop a mathematical proof that can be used for validation, see for instance [8, 17, 4].

For this purpose all the above mentioned works formulate a non-linear feasibility problem, which is then relaxed to an SDP (Semidefinite Programming) problem. In addition, the authors in [17] try to take the discretization error into consideration, when finding the solution to the relaxed problem.

The aim with this thesis is to test different methods on important class of biological models. The interval analysis approach is based mainly on [11]. The method of relaxing the feasibility problem is motivated by the work presented in [8], since we use the provided Matlab toolbox [10] devoloped by the same author(s). In addition, several methods for estimating the discretization error has been studied, one of which has been implemented in the toolbox.

1.1 Problem formulation

Let us start with some basic definitions of some entities, necessary for understanding the text in this chapter,

Definition 1.1 *Define*
$\mathcal{P}$ set of unknown parameters.
$\mathcal{X}$ set of the states.
$\mathcal{Y}$ set of the outputs.
$\mathcal{U}$ set of the inputs.
N+1 number of measurement points.
Consider the dynamical system below,

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; \mathbf{p}) \\ \mathbf{y} = h(\mathbf{x}; \mathbf{p}). \\ \end{cases} \tag{1.1}$$

This model can be expressed in its discrete counter part as given below,

$$\begin{aligned} 0 &= F(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}, \mathbf{u}^{(k)}, \mathbf{p}),\ \mathbf{x}^{(0)} = \mathbf{x}_0 \\ 0 &= H(\mathbf{y}^{(k)}, \mathbf{x}^{(k)}, \mathbf{p}),\ k \in I(0, N) \end{aligned} \tag{1.2}$$

where k denotes the kth time point. We also assume that we are given $N + 1$ samples of output measurements indexed by $I(0, N)$. In this thesis the dynamical system in (1.1) usually describes the rate of reactions in a chemical/biological system, in continuous time, where $\mathbf{x}$,$\mathbf{u}$, $\mathbf{y}$ and $\mathbf{p}$ are vectors of the states, inputs, outputs and the unknown parameters, respectively. The question now is whether there exists a parameter vector $\mathbf{p} \in \mathcal{P}$ consistent with the provided data, i.e., $\mathbf{y} \in \mathcal{Y}$ given $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{U}$, or equivalently with set theoretical notation,

$$\begin{aligned} \mathcal{P}^{CT} = \{\mathbf{p} \in \mathcal{P} | \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}, \mathbf{u} \in \mathcal{U} :& \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; \mathbf{p}) \\ & \mathbf{y} = h(\mathbf{x}; \mathbf{p})\} \end{aligned}$$

However, the model falsification is performed for the discretized system (1.2). As a result it is necessary to investigate how the introduced error by performing discretization will affect the consistency of the results, and whether the emptiness of $\mathcal{P}^{DT}$ as below would lead to model rejection.

$$\begin{aligned} \mathcal{P}^{DT} = \{\mathbf{p} \in \mathcal{P} | \mathbf{x}^{(k)} \in \mathcal{X}, \mathbf{y}^{(k)} \in \mathcal{Y}, \mathbf{u}^{(k)} \in \mathcal{U} :& \mathbf{x}^{(k+1)} = F(\mathbf{x}^{(k)}, \mathbf{u}^{(k)}; \mathbf{p}) \\ & \mathbf{y}^{(k)} = H(\mathbf{x}^{(k)}; \mathbf{p})\} \end{aligned} \tag{1.3}$$

In case the discretization does not introduce too much error, the emptiness of $\mathcal{P}^{DT}$ will result in the rejection of the proposed model. In this thesis, the method of Euler is used for discretization and we will try to address how the discretization error, introduced by this method will affect the $\mathbb{SCP}$(Set of Consistent Parameters).

Chapter 2

Mathematical Preliminaries

In this chapter, the mathematical background needed to understand the chapters to come are presented. In section §2.1 we review basic definitions in set theory and interval analysis. This includes methods of building inclusion functions as well as subpavings. Basics in mathematical optimization (used in this thesis) is also presented briefly in section §2.2. Finally in sections §2.3,§2.4,§2.5 we discuss the basic ideas in norms, inner products and singular Value Decomposition, respectively. For a more comprehensive theory, the reader is mainly referred to [5, 14, 11] and in some extension also to [16, 18].

2.1 Interval analysis

We begin with a brief introduction to the basics in set theory and interval analysis. Then we continue with a short introduction into the concept of interval vectors and interval matrices. Three methods of building inclusion functions are presented as well as the method of building unions of non-overlapping boxes, referred to as *subpaving*. The section ends with a short presentation of two methods where the notions of inclusion function and inclusion test are used for set computation, using subpaving as a class of objects to represent sets. In the examples illustrating the two set computational methods SIVIA and IMAGESP, the MATLAB toolboxes INTLAB and SCS are used, see [16, 18].

2.1.1 Set theory and interval analysis, basic definitions

Definition 2.1 *Consider two sets $\mathbb{X}$ and $\mathbb{Y}$. We define*

$$\begin{array}{lll} \texttt{Intersection} & \mathbb{X}\cap\mathbb{Y} & =\Big\{x \mid x\in\mathbb{X} \quad and \quad x\in\mathbb{Y}\Big\} \\ \texttt{Union} & \mathbb{X}\cup\mathbb{Y} & =\Big\{x \mid x\in\mathbb{X} \quad or \quad x\in\mathbb{Y}\Big\} \\ \texttt{Complement} & \mathbb{X}\backslash\mathbb{Y} & =\Big\{x \mid x\in\mathbb{X} \quad and \quad x\notin\mathbb{Y}\Big\} \\ \texttt{Cartesian Product} & \mathbb{X}\times\mathbb{Y} & =\Big\{(x,y) \mid x\in\mathbb{X} \quad and \quad y\in\mathbb{Y}\Big\} \end{array} \tag{2.1}$$

Definition 2.2 *Wrappers*
A set $\mathbb{IX}$ is called a set of wrappers for the set $\mathbb{X}$ if $\mathbb{X}$ and each singleton of it belong to $\mathbb{IX}$ and if $\mathbb{IX}$ is closed by intersection, i.e.,

$$\mathbb{X}_1 \in \mathbb{IX} \textit{ and } \mathbb{X}_2 \in \mathbb{IX} \Rightarrow \mathbb{X}_1 \cap \mathbb{X}_2 \in \mathbb{IX}.$$

A singleton set, denoted $\{a\}$, is the simplest example of a nonempty set [1].

We also present a simplified version of a distance, called *Hausdorf distance*. Assume that the two sets $\mathbb{A}$ and $\mathbb{B}$ are subsets of the compact set $\mathcal{C}(\mathbb{R}^n)$. The amount that is needed to inflate $\mathbb{B}$ so that it contains $\mathbb{A}$, is called the *proximity* of $\mathbb{A}$ to $\mathbb{B}$ and is defined as,

$$h^0_\infty(\mathbb{A},\mathbb{B}) = \inf\{r \in \mathbb{R}^+ | \mathbb{A} \subset \mathbb{B} + r\mathbb{U}\}$$

where $\mathbb{U}$ is the unit ball and $\mathbb{R}^+$ stands for the set of all positive real numbers , see Figure 2.1. In this figure, the set $\mathbb{A}$ is represented by the area inside the triangle and the set $\mathbb{B}$ by the area enclosed by the square. In a similar way, one can get $h^0_\infty(\mathbb{B},\mathbb{A})$ by inflating $\mathbb{A}$ until it contains $\mathbb{B}$.

Definition 2.3 *Hausdorff and complementary Hausdorff distance*
The distance defined as,

$$h_\infty(\mathbb{A},\mathbb{B}) = max\{h^0_\infty(\mathbb{A},\mathbb{B}), h^0_\infty(\mathbb{B},\mathbb{A})\} \tag{2.2}$$

is called Hausdorff distance. The distance

$$\overline{h}_\infty(\mathbb{A},\mathbb{B}) = max\{\overline{h}^0_\infty(\mathbb{A},\mathbb{B}), \overline{h}^0_\infty(\mathbb{B},\mathbb{A})\} \tag{2.3}$$

is called the complementary Hausdorff distance, *where*

$$\begin{aligned}\overline{h}^0_\infty(\mathbb{B},\mathbb{A}) &= h^0_\infty(\mathbb{R}^n\backslash\mathbb{B}, \mathbb{R}^n\backslash\mathbb{A})\\ \overline{h}^0_\infty(\mathbb{A},\mathbb{B}) &= h^0_\infty(\mathbb{R}^n\backslash\mathbb{A}, \mathbb{R}^n\backslash\mathbb{B})\end{aligned} \tag{2.4}$$

and where $\mathbb{R}^n\backslash\mathbb{A}$ and $\mathbb{R}^n\backslash\mathbb{B}$ denote the respective complementary sets of $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{R}^n$. The distances in (2.4) *are attained if the reverse of the operation shown by Figure 2.1 can be performed, i.e., the operation of deflating the set $\mathbb{A}$ until it is contained in the set $\mathbb{B}$ gives $\overline{h}^0_\infty(\mathbb{B},\mathbb{A})$.* The distance, based on (2.2) and (2.3), is the so called m_∞-distance, and is defined as,

$$m_\infty(\mathbb{A},\mathbb{B}) = max(h_\infty(\mathbb{A},\mathbb{B}), \overline{h}_\infty(\mathbb{A},\mathbb{B})) \tag{2.5}$$

More details and comprehensive theory with illustrative figures are available in [11].

Definition 2.4 *Interval, some definitions*

$$\begin{array}{lll} \texttt{Interval} & [x] & = \{x \in \mathbb{R} \mid \underline{x} \le x \le \overline{x}\} \\ \texttt{Lower bound} & \underline{x} & = sup\{a \in \mathbb{R} \cup \{-\infty,\infty\} | \forall x \in [x], a \le x\} \\ \texttt{Upper bound} & \overline{x} & = inf\{b \in \mathbb{R} \cup \{-\infty,\infty\} | \forall x \in [x], x \le b\} \\ \texttt{Width} & w([x]) & = \overline{x} - \underline{x} \\ \texttt{Midpoint} & mid([x]) & = \frac{\underline{x}+\overline{x}}{2} \end{array} \tag{2.6}$$

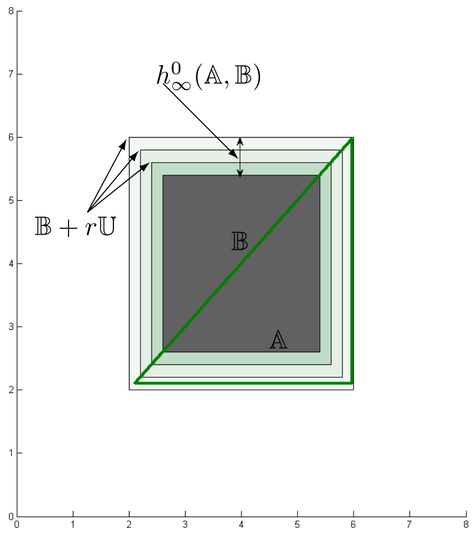


Figure 2.1. A graphical illustration of the proximity of $\mathbb{A}$ to $\mathbb{B}$. The set $\mathbb{A}$ is the area inside the triangle and the not yet inflated set $\mathbb{B}$ is represented by the area inside the smallest of the rectangles. The set $\mathbb{B}$ is inflated until the set $\mathbb{A}$ is contained in it, resulting in the proximity of $\mathbb{A}$ to $\mathbb{B}$. The inflated set $\mathbb{B}$ is now the area enclosed by the largest of the rectangles.

Arithmetic Operations

$$\begin{aligned}
[x]+[y] =& \quad [\underline{x}+\underline{y}, \overline{x}+\overline{y}] \\
[x]-[y] =& \quad [\underline{x}-\overline{y}, \overline{x}-\underline{y}] \\
[x]*[y] =& \quad [min\left\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right\}, \\
& \quad max\left\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right\}] \\
[x]/[y] =& \quad [x]*(1/[y])
\end{aligned} \tag{2.7}$$

$$[x]^2 = \begin{cases} [\underline{x}^2, \overline{x}^2], & 0 \leq \underline{x} \leq \overline{x}, \\ [\overline{x}^2, \underline{x}^2], & \underline{x} \leq \overline{x} \leq 0, \\ [0, max\left\{\underline{x}^2, \overline{x}^2\right\}], & \underline{x} \leq 0 \leq \overline{x} \\ \end{cases} \tag{2.8}$$

The division rules for defining $1/[y]$ depend mainly on the content of the interval $[y]$,

$$\begin{aligned}
1/[y] \quad &= \emptyset && \text{if } [y]=[0,0] \\
&= [1/\overline{y}, 1/\underline{y}] && \text{if } 0 \notin [y] \\
&= [1/\underline{y}, \infty[&& \text{if } \underline{y}=0 \text{ and } \overline{y}>0 \\
&=]-\infty, 1/\underline{y}] && \text{if } \underline{y}<0 \text{ and } \overline{y}=0 \\
&=]-\infty, \infty[&& \text{if } \underline{y}<0 \text{ and } \overline{y}>0
\end{aligned} \tag{2.9}$$

Example 2.1: An example on interval arithmetic

Consider an interval $[x] = [2, 3]$ and interval $[y] = [-1, 3]$. We use these two intervals to show the arithmetic operations presented in (2.7).

$$\begin{aligned}
[x]+[y] =& [2-1, 3+3] = [1, 6] \\
[x]-[y] =& [2-3, 3-(-1)] = [-1, 4] \\
[x]*[y] =& [min\left\{2*(-1), 2*3, 3*(-1), 3*3\right\} \\
& , max\left\{2*(-1), 2*3, 3*(-1), 3*3\right\}] \\
=& [-3, 9] \\
[x]/[y] =& [2, 3]*[1/y] \\
=& [2, 3]*]-\infty, \infty[\\
=&]-\infty, \infty[
\end{aligned} \tag{2.10}$$

2.1.2 Interval vectors and interval matrices

As we defined an interval $[x] \in \mathbb{R}$ to be a set of real numbers bounded by an upper and a lower bound, we similarly define the vector of intervals as the Cartesian product of intervals,

$$[\mathbf{x}] = [x_1] \times [x_2] \times ... \times [x_n] \subset \mathbb{IR}^n \tag{2.11}$$

where $\mathbb{IR}^n$ denote the set of all n-dimensional real interval vectors. For the n-dimensional box $[\mathbf{x}]$ we can define,

$$\begin{array}{lll} \texttt{Lower bound} & [\underline{\mathbf{x}}] & = (\underline{x}_1, ..., \underline{x}_n) \\ \texttt{Upper bound} & [\overline{\mathbf{x}}] & = (\overline{x}_1, ..., \overline{x}_n) \\ \texttt{Width} & w([\mathbf{x}]) & = max_{1 \le i \le n} w([x_i]) \\ \texttt{Midpoint} & mid([\mathbf{x}]) & = (mid([x_1]), ..., mid([x_n])). \end{array} \tag{2.12}$$

An interval matrix is a matrix whose elements are interval numbers,

$$[A] = \begin{pmatrix} [a_{11}]...[a_{1n}] \\ \ddots \\ [a_{m1}]...[a_{mn}] \end{pmatrix}$$

where $[a_{ij}] = [\underline{a}_{ij}, \overline{a}_{ij}]$. Similar to the vector intervals we define upper bounds and lower bounds of the interval matrix A,

$$\underline{A} = \begin{pmatrix} \underline{a}_{11}...\underline{a}_{1n} \\ \ddots \\ \underline{a}_{m1}...\underline{a}_{mn} \end{pmatrix}, \overline{A} = \begin{pmatrix} \overline{a}_{11}...\overline{a}_{1n} \\ \ddots \\ \overline{a}_{m1}...\overline{a}_{mn} \end{pmatrix}. \tag{2.13}$$

A good application for interval matrices is in solving a set of linear equations with unknowns in the coefficients matrix, which its exact solution can not be decided with methods like for instance Gaussian elimination. Then using interval analysis and the MATLAB toolbox INTLAB the solution can be found as an exact interval set.

Example 2.2: Solving linear equations using Intlab

Consider a linear system with two unknowns and the nonsingular interval coefficient matrix A,

$$\begin{aligned} &[A][\mathbf{x}] = [b] \\ &[A] = \begin{pmatrix} [4 \quad 4] & [0 \quad 2] \\ [0 \quad 2] & [4 \quad 4] \end{pmatrix} \text{ and } [b] = \begin{pmatrix} [-1 \quad 1] \\ [-1 \quad 1] \end{pmatrix} \end{aligned} \tag{2.14}$$

Now using INTLAB we can plot and show that the exact solution set is the shaded area and the entire solution set is enclosed by the narrowest possible interval vector (interval hull),

$$[\mathbf{x}] = \begin{pmatrix} [-0.5000 \quad 0.5000] \\ [-0.5001 \quad 0.5001] \end{pmatrix}$$

depicted in the Figure 2.2

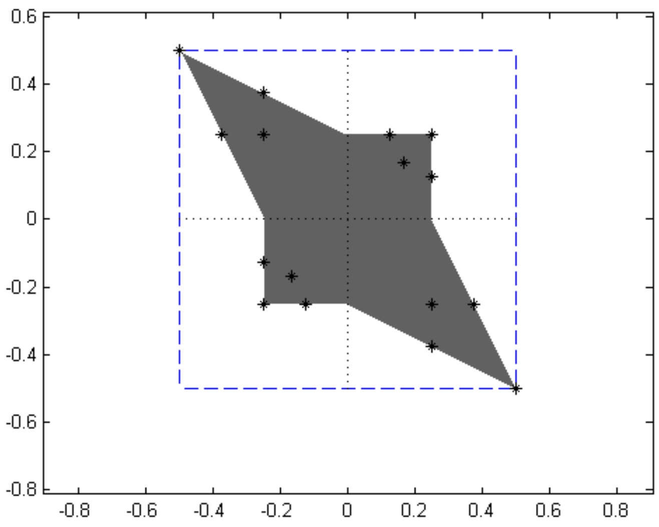


Figure 2.2. Exact solution set to $[A][\mathbf{x}] = [b]$, with $[A]$ and $[b]$ given in equation (2.14).

2.1.3 Inclusion functions

Previous subsections introduced interval numbers, arithmetic rules, interval vectors and matrices. In this subsection, the focus instead will be on using them in a function. The aim here will be to (i) introduce the concept of inclusion functions and (ii) to learn how to construct them, given a real (punctual) function. We also discuss how to make use of an inclusion function later on this section. Assume that we have a function from $\mathbb{R}^n$ to $\mathbb{R}^m$, e.g., $\mathbf{f}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^2$. If we substitute the real vector $\mathbf{x}$ by an interval $[\mathbf{x}] \subset \mathbb{IR}^2$, where, $\mathbb{IR}^2$ stands for real interval vector, we get the image function, $\mathbf{f}([\mathbf{x}])$. Inclusion function,$[\mathbf{f}]([\mathbf{x}])$, is then defined as the box which encloses the image function, i.e.

$$\mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]).$$

Building an inclusion function this way, by replacing the $\mathbf{x}$ with $[\mathbf{x}]$ and also the arithmetic operators with their interval counter part, gives the *natural inclusion function*. Figure 2.3 illustrates the concept and the fact that no matter what the shape or size of $\mathbf{f}([\mathbf{x}])$ are it is always possible to compute a box $[\mathbf{f}]$ that contains it. The smallest of the boxes is said to be minimal and is denoted by $[\mathbf{f}]^*$.

Definition 2.5 *An inclusion function* $[\mathbf{f}]$ *is* convergent *if for any sequence of boxes* $[\mathbf{x}](k)$,

$$\lim_{k\to\infty} w([\mathbf{x}](k)) = 0 \Rightarrow \lim_{k\to\infty} w([\mathbf{f}]([\mathbf{x}](k))) = 0. \tag{2.15}$$

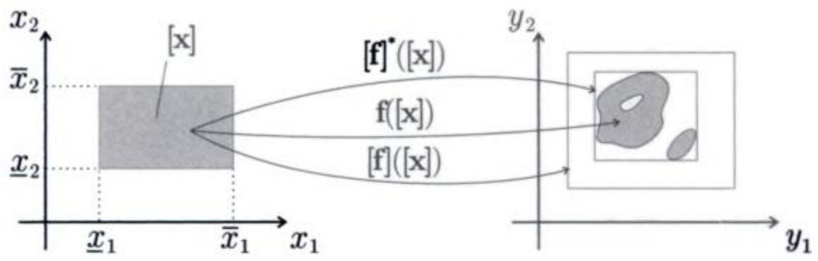


Figure 2.3. Images of a box by a vector function **f** and two of its inclusion functions [**f**] and $[\mathbf{f}]^*$; $[\mathbf{f}]^*$ is minimal [11]

It is though important remembering that when building an interval-valued function, its value may depend on how the originally real-valued function was expressed. As we will see in the example below, in the real case $x^2 = x * x$, whereas this is not the case with the interval-valued counterpart. The reason to this is that in the interval arithmetics we have (*i*) *dependency issue* and (*ii*) the *wrapping effect*. *Dependency issue* simply means that in interval functions occurrence of each interval variable is treated individually and independently. This can cause overestimation problem, which can be solved by rewriting the expression and trying to obtain an expression in which each variable occurs as rare as possible, see [11, 14]. *Wrapping effect* occurs when a result is not representable by an interval, as illustrated by the example bellow.

Example 2.3: An example on dependency issue and wrapping effect

Consider a real-valued function of a real variable,$f(\mathbf{x}) = x^2 = x * x$. For the purpose of illustrating the dependency issue, consider the following two natural inclusion functions,

$$\begin{aligned}[f_1]([\mathbf{x}]) &= [x]^2 \\ [f_2]([\mathbf{x}]) &= [x] * [x].\end{aligned} \tag{2.16}$$

In the case of punctual vector $\mathbf{x}$ and $f(\mathbf{x})$, there should be no doubt that both x^2 and $x * x$ return exactly the same value. However, in interval arithmetic the variable $[x]$ in $[x]^2$ is treated as one single variable, whereas the same variable in $[x] * [x]$ is treated as two different variables. Let [x]=[-1,1], then

$$[f_1]([\mathbf{x}]) = [x]^2 = [0, 1]$$
whereas
$$[f_2]([\mathbf{x}]) = [x] * [x] = [-1, 1],$$

see (2.7). The overestimation in $[f_2]$, as illustrated by the Figure 2.4, is because of the dependency issue. This problem can be solved by trying to express the problem in a way such that each variable occurs as rare as possible in a problem, as in the case $[f_1]$, see Figure 2.4. As the figure illustrates the inclusion function $[f_1]$,in which the variable $[x]$ appears only once, is minimum.

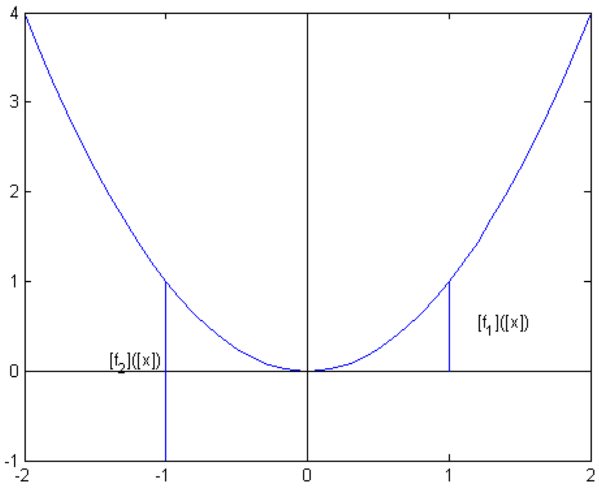


Figure 2.4. An illustration of the dependency effect. The interval variable $[x]$ is treated as one variable in the expression $[f_1]([x]) = [x]^2$. The same variable is treated as two separate variables in $[f_2]([x]) = [x] * [x]$ leading to a too large inclusion function $[f_2]$. $[f_1]$ is minimal.

For the wrapping effect, consider the function $f(x_1 + x_2, x_2)$, see [15], with the natural inclusion function $[f]([x_1] + [x_2], [x_2])$ where $[x_1] = [-2, 2]$ and $[x_2] = [-2, 2]$. The image set $f([x_1] + [x_2], [x_2])$ is a parallelogram with corners at $(-4, -2), (0, 2), (4, 2)$ and $(0, -2)$. This parallelogram is not representable as an interval-box, but can be wrapped in its inclusion function, a box, which in general also contains points or areas outside the actual image set. For this particular example, the inclusion function or the box is $[-4, 4] \times [-2, 2])$, see Figure 2.5.

In the following two other methods for building inclusion functions, namely centered and mixed centered inclusion functions, are presented briefly. For the method of *Taylor* and further reading about inclusion functions the reader is referred to [11, 14].

Centered inclusion functions

For the purpose of building the centered inclusion function, consider a scalar function f of a vector $\mathbf{x}$ from $\mathbb{R}^n$ to $\mathbb{R}$ and assume f to be differentiable over the

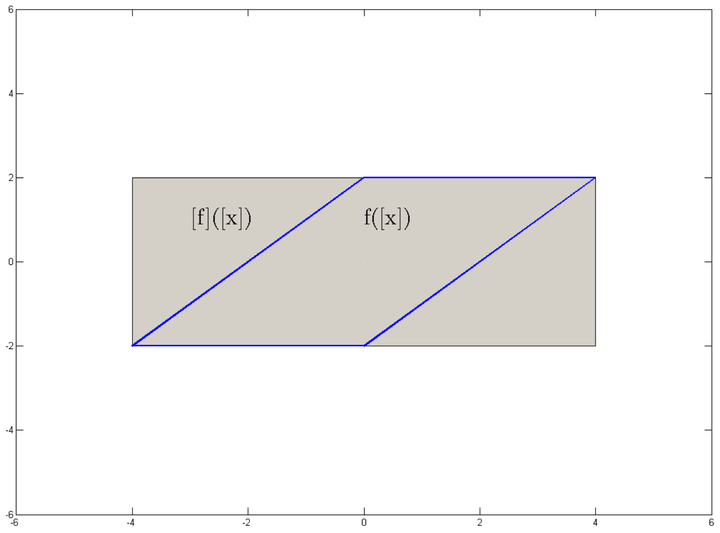


Figure 2.5. The true image of the function $f([x_1]+[x_2],[x_2])$, here denoted $f([x])$, is a parallelogram not representable as an interval. The inclusion function, $[f]([x])$ is a box, enclosing the parallelogram but, because of the wrapping effect, also areas outside the parallelogram.

interval vector $[\mathbf{x}]$ and let,

$$\mathbf{m} = \mathrm{mid}([\mathbf{x}])$$
$$\mathbf{g} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

The mean-value theorem implies that,

$$\forall \mathbf{x} \in [\mathbf{x}], \exists \mathbf{z} \in [\mathbf{x}] | f(\mathbf{x}) = f(\mathbf{m}) + \mathbf{g}^T(\mathbf{z})(\mathbf{x} - \mathbf{m}).$$

Letting $[\mathbf{g}^T]$ be an inclusion function for $\mathbf{g}^T$, we have,

$$\forall \mathbf{x} \in [\mathbf{x}], f(\mathbf{x}) \in f(\mathbf{m}) + [\mathbf{g}^T]([\mathbf{x}])([\mathbf{x}] - \mathbf{m}).$$

By this, the interval function

$$[f_c]([\mathbf{x}]) = f(\mathbf{m}) + [\mathbf{g}^T]([\mathbf{x}])([\mathbf{x}] - \mathbf{m}) \tag{2.17}$$

defines the *centered inclusion function* for $\mathbf{f}$.

Mixed centered inclusion function

Let us now consider a function f which maps $\mathbb{R}$ to $\mathbb{R}$,

$$\begin{aligned} & f(x) \in f(m) + f'([x])([x]-m) \\ & m = \text{mid}([x]). \end{aligned} \tag{2.18}$$

If now f is a function of n variables, we can build the *mixed centered inclusion function* by applying (2.18) n times, for each component of $\mathbf{x}$. To illustrate the method consider the case when $n = 2$ and $m_i = \text{mid}([x_i]), i = 1, 2$. If we apply (2.18) and consider $f(x_1, x_2)$ to be a function of x_2 only, we get

$$f(x_1, x_2) \in f(x_1, m_2) + g_2(x_1, [x_2])([x_2] - m_2). \tag{2.19}$$

Following the same procedure once more while considering $f(x_1, m_2)$ as a function of x_1 only results in

$$f(x_1, m_2) \in f(m_1, m_2) + g_1([x_1], m_2)([x_1] - m_1). \tag{2.20}$$

Now combining the two equations (2.19) and (2.20), we get

$$f(x_1, x_2) \in f(m_1, m_2) + g_1([x_1], m_2)([x_1] - m_1) + g_2(x_1, [x_2])([x_2] - m_2). \tag{2.21}$$

Thus

$$f([x_1], [x_2]) \subset \underbrace{f(m_1, m_2) + g_1([x_1], m_2)([x_1] - m_1) + g_2([x_1], [x_2])([x_2] - m_2)}_{[f_{mc}]([x_1],[x_2])}. \tag{2.22}$$

As a result for an arbitrary number of variables, the general expression for the *mixed centered inclusion function* is given by

$$[f_{mc}]([\mathbf{x}]) = f(\mathbf{m}) + \sum_{i=1}^{n} [g_i]([x_1], ..., [x_i], m_{i+1}, ..., m_n)([x_i] - m_i) \tag{2.23}$$

where $\mathbf{m}$ is the vector containing the median of each interval variable. Hence, $[f_{mc}]([\mathbf{x}])$ will be a smaller box than $[f_c]([\mathbf{x}])$, because of the mix of both punctual and interval arguments in (2.23), which results in a decreased pessimism in the inclusion function, as

$$[\mathbf{g}](\text{mid}([\mathbf{x}]), [\mathbf{x}]) \subset [\mathbf{g}]([\mathbf{x}]).$$

2.1.4 Inclusion Tests

Having derived an inclusion function using some of the method presented, one may for instance want to know whether the computed box or inclusion function contain data with certain properties. Data with certain properties could for instance be a small parameter box, in which we are looking for values consistent with a model and its outputs. In this small subsection, the methods of driving test functions is presented in brief. Let us start by defining the Boolean set

$$\mathbb{B} = \{false, true\}$$

and let

$$\mathbb{IB} = \{\emptyset, 0, 1, [0,1]\}$$

denote the set of all Boolean intervals, where $\emptyset = impossible, 0 = false, 1 = true$ and $[0,1] = indeterminate$. Similar to inclusion functions one can define a *Boolean inclusion.*If $\mathcal{B}$ is to be a Boolean function from $\mathbb{B}^n \to \mathbb{B}$, then there exists an *inclusion Boolean* $[\mathcal{B}] : \mathbb{IB}^n \to \mathbb{IB}$ if,

$$\forall([b_1], ..., [b_n]) \in \mathbb{IB}^n, \mathcal{B}([b_1], ..., [b_n]) \subset [\mathcal{B}]([b_1], ..., [b_n]). \tag{2.24}$$

The natural *Boolean inclusion* is obtained the same way as in the case of inclusion functions, namely by replacing all the arguments and operators of $\mathcal{B}$ by its interval counterpart.A natural Boolean inclusion is also minimal, if

$$\forall([b_1], ..., [b_n]) \in \mathbb{IB}^n, \mathcal{B}([b_1], ..., [b_n]) = [\mathcal{B}([b_1], ..., [b_n]).$$

Definition 2.6 *Inclusion Test*
A test *t is a function from the real vector, $\mathbb{R}^n$ to the* Boolean *set $\mathbb{B}$. An* inclusion test *$[t]$ for t is a function from real interval vector, $\mathbb{IR}^n$, to the Boolean interval, $\mathbb{IB}$ such that for any interval vector $[\mathbf{x}] \in \mathbb{IR}^n$,*

$$\begin{aligned} [t]([\mathbf{x}]) = 1 &\implies \forall \mathbf{x} \in [\mathbf{x}],\ t(\mathbf{x}) = 1 \\ [t]([\mathbf{x}]) = 0 &\implies \forall \mathbf{x} \in [\mathbf{x}],\ t(\mathbf{x}) = 0, \end{aligned} \tag{2.25}$$

similarly if $\mathbb{S} \subseteq \mathbb{R}^n$ is a set, then

$$\begin{aligned} [t_{\mathbb{S}}]([\mathbf{x}]) = 1 &\implies \forall \mathbf{x} \in [\mathbf{x}],\ t_{\mathbb{S}}(\mathbf{x}) = 1 \iff ([\mathbf{x}] \subset \mathbb{S}) \\ [t_{\mathbb{S}}]([\mathbf{x}]) = 0 &\implies \forall \mathbf{x} \in [\mathbf{x}],\ t_{\mathbb{S}}(\mathbf{x}) = 0 \iff ([\mathbf{x}] \cap \mathbb{S} = \emptyset) \end{aligned} \tag{2.26}$$

Furthermore, an inclusion test $[t]$ is said to be thin *if $[t](\mathbf{x}) = t(\mathbf{x})$ and it is* minimal *if,*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n,\ [t]([\mathbf{x}]) = \{t(\mathbf{x}) | \mathbf{x} \in [\mathbf{x}]\} \tag{2.27}$$

In the following some useful relations for intervals and sets are presented,

$$\begin{aligned} &([a,b] \leq [c,d]) = 1 && \text{if } b \leq c \\ &([a,b] \leq [c,d]) = 0 && \text{if } a > d \\ &([a,b] \leq [c,d]) = [0,1] && \text{if neither } b \leq c \text{ nor } a > d. \end{aligned} \tag{2.28}$$

If $\mathbb{S} \subset \mathbb{R}^n$, then

$$\begin{aligned}
&[t_{\mathbb{S}}]([\mathbf{x}]) \text{ is } \textit{a monotonic inclusion} \text{ iff}\\
&([\mathbf{x}] \subset [\mathbf{y}]) \implies ([t_{\mathbb{S}}]([\mathbf{x}]) \subset [t_{\mathbb{S}}]([\mathbf{y}]))\\
&[t_{\mathbb{S}}] \text{ is } \textit{minimal} \text{ iff}\\
&\forall [\mathbf{x}] \in \mathbb{IR}^n, [t_{\mathbb{S}}]([\mathbf{x}]) = t_{\mathbb{S}}([\mathbf{x}])\\
&[t_{\mathbb{S}}] \text{ is } \textit{thin} \text{ iff}\\
&\forall [\mathbf{x}] \in \mathbb{R}^n, [t_{\mathbb{S}}](\mathbf{x}) \neq [0,1]
\end{aligned} \tag{2.29}$$

2.1.5 Subpaving

This subsection presents a brief introduction to the concept of subpaving. Subpaving is defined as the union of non-overlapping sets, meaning that intersection of the member sets in a subpaving is empty. The subpaving obtained by a succession of bisections and selections of an initial box is called *regular subpaving* and the set of all such subpavings is denoted $\mathcal{RSP}([\mathbf{x}])$. Consider a box $[\mathbf{x}] = [x_1] \times ... \times [x_n]$, and define its *left* and *right children* as,

$$\begin{aligned}
L([\mathbf{x}]) &= [\underline{x}_1, \overline{x}_1] \times .... \times [\underline{x}_j, (\underline{x}_j + \overline{x}_j)/2] \times ... \times [\underline{x}_n, \overline{x}_n]\\
R([\mathbf{x}]) &= [\underline{x}_1, \overline{x}_1] \times .... \times [(\underline{x}_j + \overline{x}_j)/2, \overline{x}_j,] \times ... \times [\underline{x}_n, \overline{x}_n]
\end{aligned} \tag{2.30}$$

respectively, where

$$\begin{aligned}
&j = \min\{i | w([x_i]) = w([\mathbf{x}])\}\\
&L([\mathbf{x}]) \cap R([\mathbf{x}]) = \emptyset
\end{aligned} \tag{2.31}$$

where $w([.])$ stands for the width of an interval, see (2.6) and (2.12).

A subpaving can also be represented as a binary tree, where the initial box serves as a root to the tree, see Figure 2.8. The process of generating the two children above is called *bisection*. Let us look at an example where we let the initial box be $[\mathbf{x}_0] = [0,4] \times [0,6]$. From (2.31) we can compute $j = 2$. This value on j says that the bisection is done along $[x_1]$, resulting in left and right children, member sets of the subpaving, whose intersection is empty

$$\begin{aligned}
L([\mathbf{x}_0]) &= [0,4] \times [0,3]\\
R([\mathbf{x}_0]) &= [0,4] \times [3,6]
\end{aligned}$$

see Figure 2.6. Furthur bisection of the member sets above results on their own left and right children. For both, $L([\mathbf{x}_0])$ and $R([\mathbf{x}_0])$ we got $j = 1$ thus,

$$\begin{aligned}
LL([\mathbf{x}_0]) &= [0,2] \times [0,3]\\
RL([\mathbf{x}_0]) &= [2,4] \times [0,3]\\
LR([\mathbf{x}_0]) &= [0,2] \times [3,6]\\
RR([\mathbf{x}_0]) &= [2,4] \times [3,6].
\end{aligned}$$

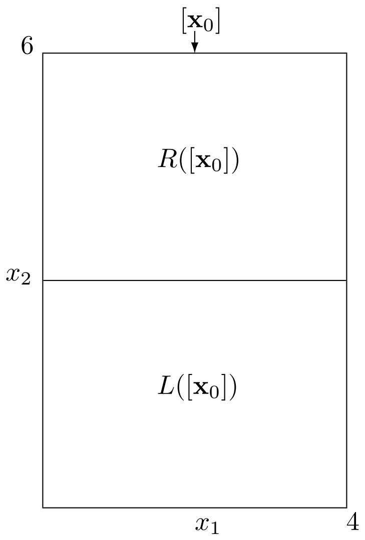


Figure 2.6. Subpaving of the initial box $[\mathbf{x}_0]$ and generating its children.

Child	$[x_1]$	$[x_2]$
$L([\mathbf{x}_0])$	[0,4]	[0,3]
$R([\mathbf{x}_0])$	[0,4]	[3,6]
$LL([\mathbf{x}_0])$	[0,2]	[0,3]
$RL([\mathbf{x}_0])$	[2,4]	[0,3]
$LR([\mathbf{x}_0])$	[0,2]	[3,6]
$RR([\mathbf{x}_0])$	[2,4]	[3,6]
$LRL([\mathbf{x}_0])$	[2,4]	[0,1.5]
$RRL([\mathbf{x}_0])$	[2,4]	[1.5,3]
$LLR([\mathbf{x}_0])$	[0,2]	[3,4.5]
$RLR([\mathbf{x}_0])$	[0,2]	[4.5,6]
$LLLR([\mathbf{x}_0])$	[0,1]	[3,4.5]
$RLLR([\mathbf{x}_0])$	[1,2]	[3,4.5]
$LRRL([\mathbf{x}_0])$	[2,3]	[1.5,3]
$RRRL([\mathbf{x}_0])$	[3,4]	[1.5,3]
$LLRL([\mathbf{x}_0])$	[2,3]	[0,1.5]
$RLRL([\mathbf{x}_0])$	[3,4]	[0,1.5]
$LRLR([\mathbf{x}_0])$	[0,1]	[4.5,6]
$RRLR([\mathbf{x}_0])$	[1,2]	[4.5,6]

Table 2.1. Resulting subboxes of initial box $[\mathbf{x}_0]$

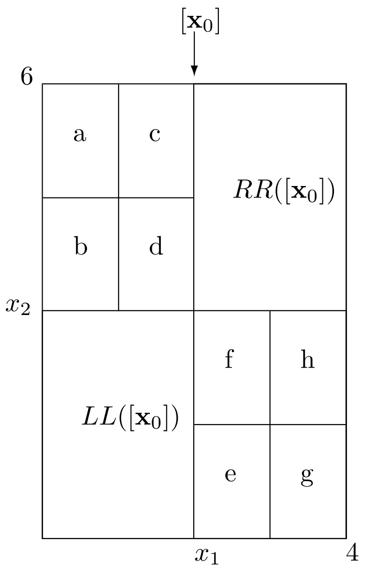


Figure 2.7. Further subpaving of the Figure 2.6

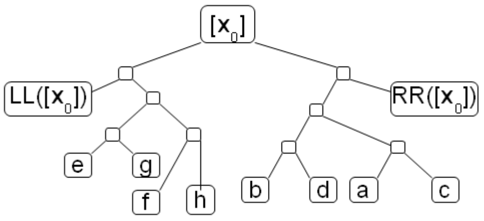


Figure 2.8. Binary tree representation of the subpaving given by Figure 2.7

The same procedure, recursively applied on the above generated children gives new subboxes, whose union shapes the final subpaving for the initial box, see Figure 2.7 and Table 2.1.The binary tree representaion of the same subpaving is shown by Figure 2.8.

In the subsections to come, we make use of the unions of non-overlapping sets(subpavings), inclusion functions and inclusion tests, presented this far.

2.1.6 Set Inversion

As stated before, the notions of inclusion function and inclusion test together with subpvaing, as a class of objects to represent sets, are used in this and next subsection to show how set computations can be performed.This subsection, presents the method SIVIA (Set inverter via interval analysis), by which the two regular subpavings $\underline{\mathbb{X}}$ and $\overline{\mathbb{X}}$ of $\mathbb{X}$ such that $\underline{\mathbb{X}} \subset \mathbb{X} \subset \overline{\mathbb{X}}$ can be obtained. Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, be a non linear function and let $\mathbb{Y}$ be a subpaving or subset of $\mathbb{R}^m$. The charachtarization,

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}) \tag{2.32}$$

is called Set inversion. Given a very large search box $[\mathbf{x}_0]$, guaranteed to contain $\overline{\mathbb{X}}$, four different situations are considered,

(a) If $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} \neq \emptyset$ (Figure 2.9 a) but $[\mathbf{f}]([\mathbf{x}])$ is not included in $\mathbb{Y}$, then nothing can be said other than that $[\mathbf{x}]$ is undetermined and has to be bisected to smaller boxes if $w([x]) \geq \epsilon$, where ϵ is a prespecified precision parameter. Recall how the children $L([\mathbf{x}])$ and $R([\mathbf{x}])$ were obtained.

(b) If $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$ then $\mathbf{f}([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$ as well, meaning that $[\mathbf{x}]$ does not belong to the solution set $\mathbb{X}$ and can be cut off from the solution tree, see Figure 2.9 b and recall that $\mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}])$, section 2.1.3.

(c) If, as depicted on Figure 2.9 c, $[\mathbf{f}]([\mathbf{x}])$ is included in $\mathbb{Y}$ then $\mathbf{f}([\mathbf{x}])$ is also included, meaning that $[\mathbf{x}]$ belongs to the solution subpaving $\mathbb{X}$ and can be stored in $\underline{\mathbb{X}}$ and $\overline{\mathbb{X}}$.

(d) If the box is undetermined and has the width lower than ϵ, then it is considered small enough and can be stored in $\overline{\mathbb{X}}$, Figure 2.9 d.

The SIVIA algorithm is summerized in the algorithm 1, where both the method of inclusion function and the method of inclusion test is illustrated.

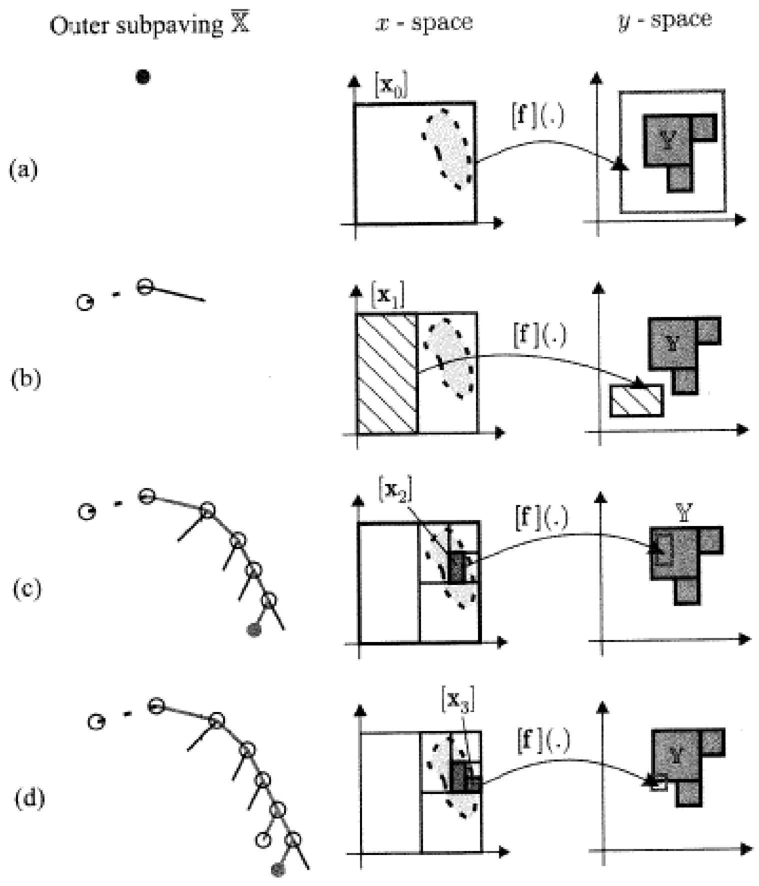


Figure 2.9. Graphical illustration of SIVIA [11]

Algorithm 1: Set Inverter Via Interval Analysis

```
1. IF INCLUSION FUNCTION $\Longrightarrow$ SIVIA(in: $[\mathbf{x}]$,$\mathbf{f}$,$\epsilon$, $\mathbb{Y}$,inout: $\underline{\mathbb{X}}$,$\overline{\mathbb{X}}$)
    a. Set $\underline{\mathbb{X}} := \emptyset$ and $\overline{\mathbb{X}} := \emptyset$
    b. if $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$ return;
    c. If $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$ then
       $\{\overline{\mathbb{X}} := \overline{\mathbb{X}} \cup [\mathbf{x}]; \underline{\mathbb{X}} := \underline{\mathbb{X}} \cup [\mathbf{x}];$ return$\}$;
    d. If $w([\mathbf{x}]) < \epsilon$ then $\{\overline{\mathbb{X}} := \overline{\mathbb{X}} \cup [\mathbf{x}];$ return$\}$;
    e. Recursively call SIVIA ont he children of $[\mathbf{x}]$
       i.e. SIVIA$(\mathbf{f}, \mathbb{Y}, L([\mathbf{x}]), \epsilon, \underline{\mathbb{X}}, \overline{\mathbb{X}})$; SIVIA$(\mathbf{f}, \mathbb{Y}, R([\mathbf{x}]), \epsilon, \underline{\mathbb{X}}, \overline{\mathbb{X}})$
2. IF TEST FUNCTION $\Longrightarrow$ SIVIA(in: $t$,$[\mathbf{x}]$,$\epsilon$, inout: $\underline{\mathbb{X}}$,$\overline{\mathbb{X}}$ )
    a. Set $\underline{\mathbb{X}} := \emptyset$ and $\overline{\mathbb{X}} := \emptyset$;
    b. If $[t]([\mathbf{x}]) = 0$ return;
    b. If $[t]([\mathbf{x}]) = 1$ then
       $\{\overline{\mathbb{X}} := \overline{\mathbb{X}} \cup [\mathbf{x}]; \underline{\mathbb{X}} := \underline{\mathbb{X}} \cup [\mathbf{x}];$ return$\}$;
    c. If $w([\mathbf{x}]) < \epsilon$ then $\{\overline{\mathbb{X}} := \overline{\mathbb{X}} \cup [\mathbf{x}];$ return$\}$;
    e. Recursively call SIVIA on the children of $[\mathbf{x}]$
       i.e. SIVIA$(t, L([\mathbf{x}]), \epsilon, \underline{\mathbb{X}}, \overline{\mathbb{X}})$; SIVIA$(t, R([\mathbf{x}]), \epsilon, \underline{\mathbb{X}}, \overline{\mathbb{X}})$
```

2.1.7 Image evaluation

The previous subsection presented computation of solution set via SIVIA, the focus in this subsection is to compute a direct image $\mathbb{Y} \subset \mathbb{R}^m$ of a regular subpaving $\mathbb{X} \subset \mathbb{R}^n$ under a continues function $\mathbf{f}$. The characterization

$$\mathbb{Y} = f(\mathbb{X}) = \{\mathbf{y} \in \mathbb{R}^m | \exists \mathbf{x} \in \mathbb{X}, \mathbf{f}(\mathbf{x}) = \mathbf{y}\} \tag{2.33}$$

is the image evaluation problem.Assume that a convergent inclusion function $[\mathbf{f}]$ for $\mathbf{f}$ is available, see Definition 2.5 on page 10.The set $\mathbb{Y}$, a regular subpaving, is the image of a regular subpaving, contained in the box $[\mathbf{f}](\mathbb{X})$. The algorithm proceeds in three steps,

(a) Given an initial regular subpaving $\mathbb{X} \subset \mathbb{R}^n$, Figure 2.10 a.

(b) *Mincing*:split the boxes in the initial subpaving in non-minimal subpaving $\mathbb{X}_\epsilon$, such that $w([\mathbf{x}_{\epsilon,i}]) < \epsilon$, Figure 2.10 b.

(c) *Evaluation*:the inclusion function is evaluated over each boxes in $\mathbb{X}_\epsilon$, provided by the previous step, resulting in $[\mathbf{f}_{\epsilon,i}]([\mathbf{x}_{\epsilon,i}])$. These boxes are then stored into a list $\mathcal{U}$, Figure 2.10 c.

(d) *Regularization*: a regular subpaving $\overline{\mathbb{Y}}$, containing the union $\mathbb{U}$ of all the boxes of the list $\mathcal{U}$ is computed, Figure 2.10 d.

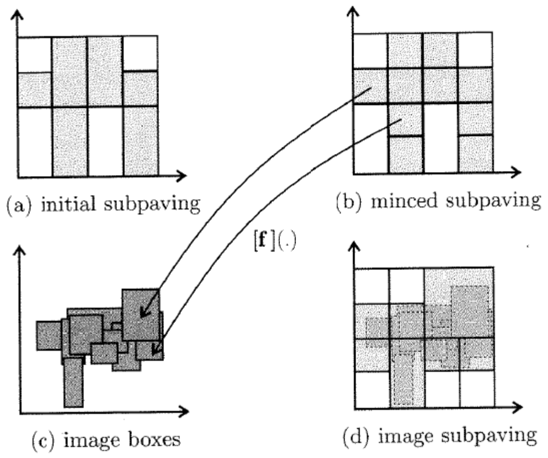


Figure 2.10. The steps of IMAGESP [11]

Since $\mathbf{f}(\mathbb{X}) \subset \mathbb{U}$, the final regularization step can be seen as an evaluation of inverse image to $\mathbb{U}$ by the identity function, using SIVIA.

The image evaluation algorithm is summarized in Algorithm 2

Algorithm 2: IMAGESP(in: $\mathbf{f}, \mathbb{X}, \epsilon$; out: $\overline{\mathbb{Y}}$)

```
1. $\mathbb{X}_\epsilon$:=mince($\mathbb{X}, \epsilon$);
2. $\mathcal{U} := \emptyset$;
3. for each $[\mathbf{x}] \in \mathbb{X}_\epsilon$ add $[\mathbf{f}]([\mathbf{x}])$ to the list $\mathcal{U}$;
4. SIVIA($\mathbf{y} \in \mathbb{U}, [\mathbf{f}]([\mathbb{X}]), \epsilon, \underline{\mathbb{Y}}, \overline{\mathbb{Y}}$).
```

2.2 Convex optimization problems

An optimization problem,

$$\begin{cases} \texttt{minimize} & f_0(x) \\ \texttt{subject to} & f(x) \leq 0 \\ & a^T x = b \end{cases} \tag{2.34}$$

is convex if both the objective function and the inequality constraint function in the problem are convex. Moreover, the equality constraint must be affine, meaning that it has to be a sum of a linear function and a constant. In the example above,

$$g(x) = a^T x - b$$

is affine [chap.4 [5]]. One basic requirement for a function to be convex is that its domain is a convex set. A convex set is defined as a set for which a line segment between tow points in the set still lies inside the set. More precisely $\mathtt{dom}f$ is a convex set if for any two points, $x_1, x_2 \in \mathtt{dom}f$ and any θ with $0 \leq \theta \leq 1$

$$\theta x_1 + (1-\theta)x_2 \in \mathtt{dom}f$$

holds.

A simple and straight forward way to check the convexity of a function f is to drive its *gradient* or first order derivative. A function f is convex if the domain of f, $\mathtt{dom}f$, is convex and the condition

$$f(x_2) \geq f(x_1) + \bigtriangledown f(x_1)^T (x_2 - x_1)$$

holds for all $x_1, x_2 \in \mathtt{dom}f$.

Another way of checking the convexity of a function is through its second derivative (if it exists). In that case, if , as above, the $\mathtt{dom}f$ is convex and if the *Hessian* of f is positive semidefinite,

$$\bigtriangledown^2 f(x) \succeq 0$$

then the function is said to be convex [chap.3 [5]]. Note that when the optimization problem is a maximization problem, then the problem is not convex unless the objective function is concave. This property is important to keep in mind when reading the subsection dealing with dual-primal problem.

2.2.1 SDP (Semidefinite Programming)

In this subsection a brief introduction to the optimization family of SDP (Semidefinite Programming) is presented. For more in dept theories and discussions, the reader is referred to [5, 19, 21]. Consider the problem of minimization of a *linear* function of variable $x \in \mathcal{R}^m$,

$$a^T x = a_1 x_1 + a_2 x_2 + ... + a_m x_m$$

subject to the matrix inequality

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \geq 0 \tag{2.35}$$

where $F_0, ..., F_m \in \mathcal{R}^{n\times n}$ and $a \in \mathcal{R}^m$ are data for the optimization problem below

$$\begin{cases} \texttt{minimize} & a^T x \\ \texttt{subject to} & F(x) \geq 0 \\ & \end{cases} \tag{2.36}$$

The inequality sign in $F(x) \geq 0$ (in some literature denoted $\succeq$) means that $F(x)$ is positive semidefinite, i.e.

$$z^T F(x) z \geq 0$$

for all $z \in R^n$. Since the function $F(x)$ is a linear matrix and the constraint an inequality one, we call $F(x) \geq 0$ a *linear matrix inequality* and the problem (2.36) an SDP. It is obvious that an SDP problem is a convex optimization problem, since,

- the objective function $a^T x$ is linear and thus convex.
- The inequality constraint $F(x) \geq 0$ is convex. Let, as in the previous subsection, $x_1, x_2 \in \texttt{dom}F$ and if $F(x_1) \geq 0$ and if $F(x_2) \geq 0$ then,

  $$F(\theta x_1 + (1-\theta)x_2) = \theta F(x_1) + (1-\theta)F(x_2) \geq 0$$

  for all $0 \leq \theta \leq 1$.
- The affinity condition is not relevant in this particular problem. But if we have had an equality constraint, $A(x) = b \leftrightarrow A(x) - b = 0$ it had been shown in a similar way as in the previous subsection that it is affine.

2.2.2 Dual and primal problems

In mathematical optimization it is sometimes routine to take advantage of the duality relationship and solve the dual problem instead of the originally given primal one. Consider a general non-linear optimization problem,

$$\begin{cases} \texttt{minimize} & f_0(x) \\ \texttt{subject to} & f_i(x) \leq 0, \ i = 1, ..., m \\ & h_i(x) = 0, \ i = 1, ..., p \\ & \end{cases} \tag{2.37}$$

where $x \in \mathcal{R}^n$ is the optimization variable and $f_0(x)$ the objective function. The domain of this problem is then defined as all permissible $x's$,i.e. for which all the constraints hold,

$$\mathcal{D} = \bigcap_{i=0}^{m} \texttt{dom} f_i \ \cap \ \bigcap_{i=0}^{p} \texttt{dom} h_i$$

Considering the optimization problem (2.37), the associated *Lagrangian* $\mathcal{L}$ is constructed by adding a wighted sum of the constraints to the objective function,

$$\begin{aligned} \mathcal{L}(x_i, \lambda_i, v_i) =& f_0(x) + \sum_{i=1}^{m} \lambda_i^T f_i(x) + \sum_{i=1}^{p} v_i^T h_i(x) \\ \lambda_i \geq& \, 0, i = 1, ..., m \end{aligned} \tag{2.38}$$

where λ_i, and v_i, are referred to as Lagrange multipliers or dual variables, associated with the inequality and equality constraints, respectively. The *Lagrangian dual function* g is formulated as the minimum of (2.38) over all $x \in \mathcal{D}$,

$$g(\lambda_i, v_i) = inf_{x \in \mathcal{D}} \mathcal{L}(x_i, \lambda_i, v_i) \tag{2.39}$$

The *Lagrangian* $\mathcal{L}$ in (2.38) is affine and unbounded below in x and therefore the *dual function* $g(\lambda, v)$ takes on the value $-\infty$. This means that, for any feasible point x_{feas} and any $\lambda \geq 0$ and v, the following holds,

$$g(\lambda, v) \leq f_0(x_{feas})$$

because

$$\sum_{i=1}^{m} \lambda_i^T f_i(x_{feas}) + \sum_{i=1}^{p} v_i^T h_i(x_{feas}) \leq 0$$

This last inequality follows from $f_i(x)$ being non positive and $h_i(x)$ being zero, see (2.37).

In order to estimate the best lower bound for the optimal value of the primal problem (2.37) we are to maximize the *dual function*, (2.39), for all nonnegative λ,

$$\begin{cases} \texttt{maximize} & g(\lambda, v) \\ \texttt{subject to} & \lambda_i \geq 0 \end{cases} \tag{2.40}$$

This problem is referred to as the *dual optimization problem* if (2.37) is to be the *primal*. Another important consequence of working with the dual problem is that, the dual optimization problem is always a convex problem, no matter whether the original primal problem was convex or not. It is because it turns to a maximization of a concave function and the constraints, as can be seen, are convex. For more reading, see [chap. 5 [5]]

2.3 Norms

In linear algebra, functional analysis and related areas in mathematics, a norm is a function $f : \mathbb{R}^n \to \mathbb{R}$ with the following properties,

- Domain of $f = \mathbb{R}^n$
- $f(x) \geq 0$ for all $x \in \mathbb{R}^n$
- f is definite: $f(x) = 0$ if and only if x=0
- f is homogeneous: $f(tx) = |t|\, x$ for all $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$
- f satisfies the triangle inequality: $f(x + y) \leq f(x) + f(y)$

Norm function is defined by $f(x) = \|x\|_{symb}$. The *symb* is usually used if the norm is other than the well known two norm. The index *symb* takes different values and for $x \in \mathbb{R}^n$,

- 1-norm, sum-absolute-value

$$\|x\|_1 = |x_1| + ... + |x_n| \tag{2.41}$$

- Euclidean norm or 2-norm for a vector x $\in \mathbb{R}^n$,

$$\|x\|_2 = (x^T x)^{1/2} = (x_1^2 + ... + x_n^2)^{1/2} \tag{2.42}$$

- Chebyshev or ∞-norm

$$\|x\|_\infty = \max\{|x_1|, ..., |x_n|\} \tag{2.43}$$

- The operator norm or the induced norm of X $\in \mathbb{R}^{m\times n}$ induced by the norms $\|.\|_a$ and $\|.\|_b$ on $\mathbb{R}^m$ and on$\mathbb{R}^n$, respectively is defined as,

$$\|X\|_{a,b} = \sup\{\|Xu\|_a \, | \|u\|_b \leq 1\} \tag{2.44}$$

  When both $\|.\|_a$ and $\|.\|_b$ are similar, interesting results are achieved. For instance if a=b=2, the operator norm is its maximum-singular-value,

$$\|X\|_2 = \sigma_{max}(X) = (\lambda_{max}(X^T X))^{1/2} \tag{2.45}$$

Similar results can also be achieved for other values of a and b. If $\|.\|$ is any norm on $\mathbb{R}^n$, then there exists a quadratic norm $\|.\|_P$ for which

$$\|x\|_P \leq \|x\| \leq \sqrt{n}\|x\|_P \tag{2.46}$$

2.4 Inner product

Standard inner product for vectors x and y $\in \mathbb{R}^n$ is defined as,

$$\langle x, y\rangle = x^T y = \sum_{i=1}^{n} x_i y_i \tag{2.47}$$

and for matrices X,Y $\in \mathbb{R}^{m\times n}$,

$$< X, Y >= tr(x^T y) = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} y_{ij} \tag{2.48}$$

with trace of a matrix being sum of its diagonal elements.

2.5 Singular Value Decomposition

Singular value theorem is used in coming chapters for computing an upper bound for the discretization error. Let $A \in R^{m\times n}$ be a real valued matrix with m rows and n columns. If Rank(A)=r then there exists $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ such that $U^T U = I$ and $V^T V = I$, and nonzero scalars $\sigma_i, i = 1, 2, ...r$ so that,

$$\begin{aligned} &A = U\Sigma V^T \\ &\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_r) \\ &\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \end{aligned} \tag{2.49}$$

The largest of the singular values, i.e. σ_1, can also be written as,

$$\sigma_{max}(A) = \sup_{x,y\neq 0}\frac{x^T Ay}{\|x\|_2\|y\|_2} = \sup_{y\neq 0}\frac{\|Ay\|_2}{\|y\|_2} \tag{2.50}$$

Chapter 3

Infeasibility Certificate

The proposed model for describing the desired biological system or chemical reaction is usually given as a continuous-time system,

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; \mathbf{p}) \\ \mathbf{y} = h(\mathbf{x}; \mathbf{p}). \end{cases} \tag{3.1}$$

However, the model rejection is performed for discrete-time systems using discrete data points. As a result we need to discretize the given time continuous system using the algorithm,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + h^{(k)}\Phi(\mathbf{x}^{(k)}, \mathbf{p}) \tag{3.2}$$

where for the special case of Euler method $\Phi = f(\mathbf{x}^{(k)}, \mathbf{p})$, [7]. In this thesis only the Euler method is considered. In (3.2) the sampling rate h usually is a constant scalar, but this can vary depending on the problem under consideration. Now using (3.2) one can rewrite (3.1) in its discrete form,

$$\begin{aligned} 0 &= F(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}, \mathbf{u}^{(k)}, \mathbf{p}),\ \mathbf{x}^{(0)} = \mathbf{x}_0 \\ 0 &= H(\mathbf{y}^{(k)}, \mathbf{x}^{(k)}, \mathbf{p}) \end{aligned} \tag{3.3}$$

where $\mathbf{x}^{(k)} \in \mathbb{R}^{n_x}, \mathbf{y}^{(k)} \in \mathbb{R}^{n_y}, \mathbf{u}^{(k)} \in \mathbb{R}^{n_u}, \mathbf{p} \in \mathbb{R}^{n_p}$ represents the vectors of the states, outputs, inputs and the unknown constant parameters to be estimated, respectively. The functions $F(.)$ and $H(.)$ must be polynomials in all their arguments. In case any of these two is rational then one can just multiply both sides of the equations with its least common denominator.

The noisy measurements of the outputs of the system are usually represented as below [8]

$$\bar{\mathbf{y}}^{(k)} = \mathbf{y}^{(k)} + \mathbf{e}^{(k)}, \qquad k = 0, 1, ..., N \tag{3.4}$$

where N+1 is the number of measurement points and the measurement noise $\mathbf{e}^{(k)} \in \mathbb{R}^{n_y}$ belongs to a compact set $\varepsilon^{n_y} \subset \mathbb{R}^{n_y}$. Also in (3.4), $\mathbf{y}^{(k)}$ is the actual output of the system, which by construction is contained in the set $\mathcal{Y}^{(k)}$ at each time point k,

$$\mathcal{Y}^{(k)} = \{\mathbf{y} \in \mathbb{R}^{(n_y)} \mid \exists e \in \varepsilon^{(k)} : \bar{\mathbf{y}}^{(k)} = \mathbf{y} + \mathbf{e}\} \tag{3.5}$$

A graphical illustration of (3.5) is shown by Figure 3.1.

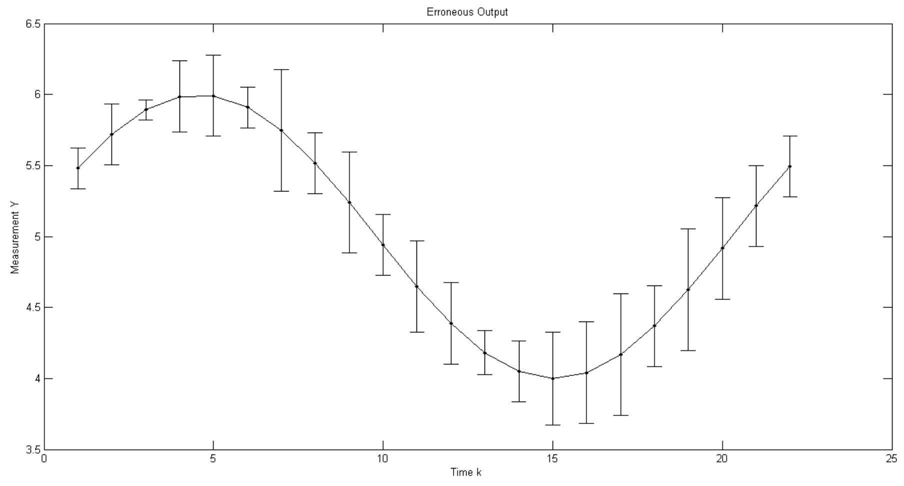


Figure 3.1. Graphical illustration of the output

Having discretized the system, we express the considered problem as a feasibility problem. This feasibility problem is described in (3.6) which in words means that we simply want to see if there exists a sequence of output **y**, state **x**, input **u** and unknown parameters **p** in the corresponding sets, for which (3.3) has a solution. This problem however is in general non-convex and not tractable as it is formulated below. The non-convexity comes from the nonlinearity in F and H in (3.3). In order to make this feasibility problem solvable, it needs to be relaxed to an SDP (SemiDefinite Programming) problem.

3.1 SDP Relaxation

The relaxation procedure here is motivated by the work presented in [8], because the Matlab toolbox developed by the same author(s) will be used for simulations in the implementation part. This procedure of relaxation is almost the same even in other works done in the area, see [17, 4]. Figure 3.2 illustrates the steps of this relaxation method. The start point is the following feasibility problem,

$$\begin{cases} \texttt{find} & \mathbf{y}^{(0,N)} \in \mathcal{Y}^{(0,N)}, \mathbf{x}^{(0,N)} \in \mathcal{X}^{(0,N)} \\ & \mathbf{u}^{(0,N-1)} \in \mathcal{U}^{(0,N-1)}, \mathbf{p} \in \mathcal{P} \\ \texttt{s.t.} & F(\mathbf{x}^{(k+1)}, \mathbf{x}^k, \mathbf{u}^k, \mathbf{p}) = 0, \quad k = 0,..,N-1 \\ & H(\mathbf{y}^k, \mathbf{x}^k, \mathbf{p}) = 0, \quad k = 0,..,N \\ & \end{cases} \tag{3.6}$$

where $\mathbf{y}^{(0,N)} = (y^{(0)}, ..., y^{(N)}), \mathbf{u}^{(0,N)} = (u^{(0)}, ..., u^{(N)})$ and $\mathbf{x}^{(0,N)} = (x^{(0)}, ..., x^{(N)})$ represent the output, input and state sequences, respectively, whereas

$$\begin{aligned} \mathcal{Y}^{(0,N)} &= \{\mathbf{y}^{(0,N)} | \mathbf{y}^{(k)} \in \mathcal{Y}^{(k)}\} \\ \mathcal{X}^{(0,N)} &= \{\mathbf{x}^{(0,N)} | \mathbf{x}^{(k)} \in \mathcal{X}^{(k)}\} \\ \mathcal{U}^{(0,N)} &= \{\mathbf{u}^{(0,N)} | \mathbf{u}^{(k)} \in \mathcal{U}^{(k)}\},\ k = 0, ..., N \end{aligned}$$

denote the corresponding admissible sets for $k = 0, ..., N$.

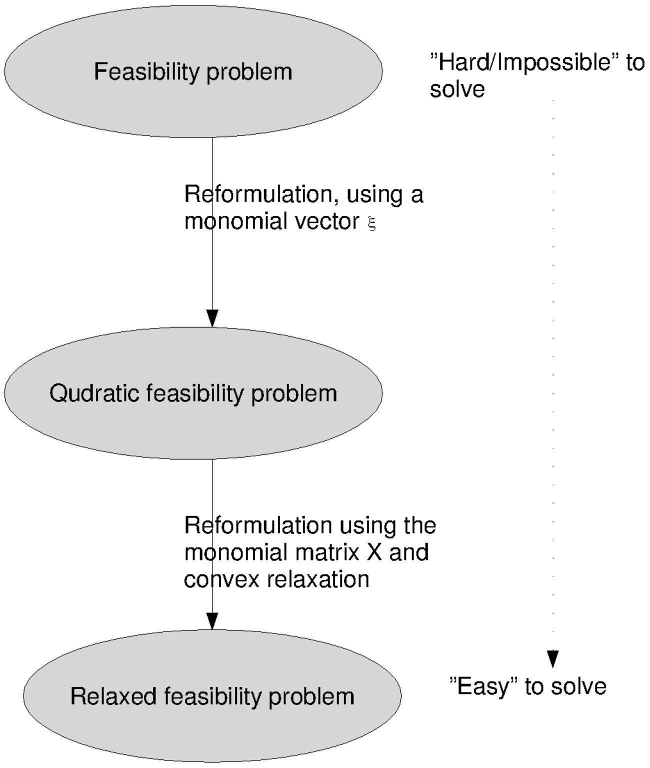

Feasibility problem
"Hard/Impossible" to solve
Reformulation, using a monomial vector ξ
Qudratic feasibility problem
Reformulation using the monomial matrix X and convex relaxation
Relaxed feasibility problem
"Easy" to solve

Figure 3.2. Relaxation procedure [9]

For the relaxation to a SDP , (3.6) is first rewritten as a quadratic feasibility problem, which also introduces the vector $\xi \in \mathbb{R}^{n_\xi}$ consisting of the monomials appearing in F and H in (3.3),

$$\begin{aligned} \xi =& (1, y_{i_y}^{(k)}, x_{i_x}^{(k)}, u_{i_u}^{(k)}, p_{i_p}, y_{i_y}^{(k)} x_{i_x}^{(k)}, x_{i_x}^{(k)} p_{i_p}, ...)^T \\ & i_y \in \mathcal{I}(1, n_y) \\ & i_x \in \mathcal{I}(1, n_x) \\ & i_u \in \mathcal{I}(1, n_u) \\ & i_p \in \mathcal{I}(1, n_p) \\ & k \in \mathcal{I}(0, N) \end{aligned} \tag{3.7}$$

Let $\mathcal{S}^{n_\xi}$ denote a square and symmetric matrix and let $\mathbf{Q}_i \in \mathcal{S}^{n_\xi}$ be a sparse matrix, then(3.3) can be transformed to,

$$\xi^T \mathbf{Q}_i \xi = 0, \; i \in \mathcal{I}(1, n_x N + n_y(N+1)). \tag{3.8}$$

When there are higher order terms in the vector ξ, additional constraints must be introduced leading to additional equations of the form

$$\xi^T \mathbf{Q}_i \xi = 0, \;\; i \in \mathcal{I}(n_x N + n_y(N+1) + 1, c)$$

where c denotes the total number of the equality constraints.

Similarly the constraints $\mathbf{y}^{(0,N)} \in \mathcal{Y}^{(0,N)}, \mathbf{x}^{(0,N)} \in \mathcal{X}^{(0,N)}, \mathbf{u}^{(0,N-1)} \in \mathcal{U}^{(0,N-1)}$ and $\mathbf{p} \in \mathcal{P}$ can also be expressed as

$$\mathbf{B}\xi \geq 0 \tag{3.9}$$

with $\mathbf{B} \in \mathbb{R}^{n_b \times n_\xi}, n_b$ being the number of constraints.

This leads us to the quadratic feasibility problem,

$$\begin{cases} \texttt{find} & \xi \in \mathcal{R}^{(n_\xi)} \\ \texttt{s.t.} & \xi^T \mathbf{Q}_i \xi = 0, \;\; i \in \mathcal{I}(1, c) \\ & \mathbf{B}\xi \geq 0 \\ & \xi_1 = 1 \end{cases} \tag{3.10}$$

Now introducing the matrix $\mathbf{X} = \xi\xi^T$ and dropping the non-convex constraint rank($\mathbf{X}$) = 1, leads us to a relaxed SDP feasibility problem,

$$\begin{cases} \texttt{find} & \mathbf{X} \in \mathcal{S}^{(n_\xi)} \\ \texttt{s.t.} & tr(\mathbf{Q}_i \mathbf{X}) = 0, \;\; i \in \mathcal{I}(1, c) \\ & tr(\mathbf{e}_1 \mathbf{e}_1^T \mathbf{X}) = 1 \\ & \mathbf{B}\mathbf{X}\mathbf{e_1} \geq 0 \\ & \mathbf{B}\mathbf{X}\mathbf{B}^T \geq 0 \\ & \mathbf{X} \succeq 0 \end{cases} \tag{3.11}$$

with $\mathbf{e}_1 = (1, 0, ..., 0)^T \in \mathbb{R}^{n_\xi}$.

Note that these steps do not necessarily need to be performed in the exact same way as described here, e.g., in [17, 4], these are done in a slightly different way.

Since the relaxation process is conservative, each feasible solution for (3.6) corresponds to a feasible solution for the relaxed problem in (3.11). However, additional solutions may be introduced that are only feasible for (3.11), which might lead to considering erroneously an invalid model to be valid. [Shona Laila and colleagues] in [17], reduce this error by introducing additional constraints $\mathbf{BXB}^T \geq 0$.

The problem in (3.11) together with additional constraints $\mathbf{BXB}^T \geq 0$ is large and is computationally heavy to solve. In order to reduce the computational effort, one can split up the measurement, state and input sequences into subsequences. For instance, for the measurement sequence $\mathcal{Y}$ this is shown in Figure 3.3 and results in the following subproblems,

$$P_j : \begin{cases} \texttt{find} & \mathbf{y}^{(j,j+m)} \in \mathcal{Y}^{(j,j+m)}, \mathbf{x}^{(j,j+m)} \in \mathcal{X}^{(j,j+m)} \\ & \mathbf{u}^{(j,j+m-1)} \in \mathcal{U}^{(j,j+m-1)}, \mathbf{p} \in \mathcal{P} \\ \texttt{s.t.} & F(\mathbf{x}^{(k+1)}, \mathbf{x}^k, \mathbf{u}^k, \mathbf{p}) = 0, \quad k \in \mathcal{I}(j, j+m-1) \\ & H(\mathbf{y}^k, \mathbf{x}^k, \mathbf{p}) = 0, \quad k \in \mathcal{I}(j, j+m) \end{cases} \tag{3.12}$$

where $1 \leq m \leq N$ and m+1 is the number of sequential measurement points in each such sub-feasibility problem. If the original feasibility problem, with all the measurements taken into account is feasible then also $P_j, j = 0, ..., N - m$, is feasible, but the converse is in general not true.

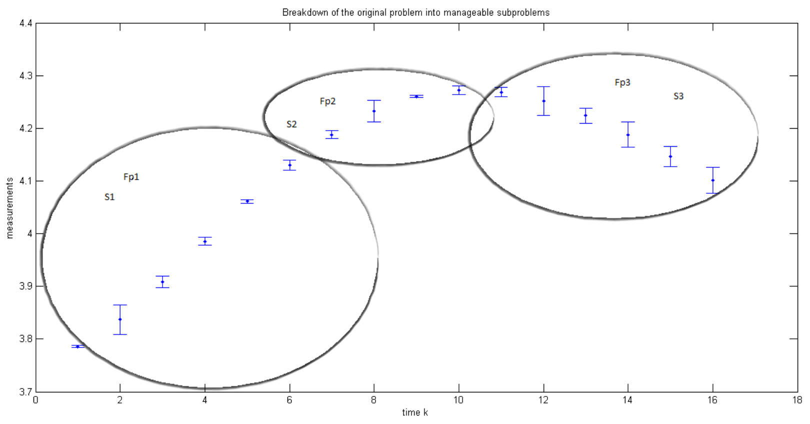


Figure 3.3. Breaking down the original problem into subproblems

3.2 Infeasibility Certificate

First, we construct the Lagrangian function $\mathcal{L}$ for the primal problem (3.11) and use it to formulate the dual problem, which can be used to certify the infeasibility of the primal problem. Generally the Lagrangian $\mathcal{L}$ for the optimization problem,

$$P : \begin{cases} \texttt{minimize} & f_0(x) \\ \texttt{s.t.} & g(x) \leq a \\ & h(x) = b \\ & x \geq 0 \end{cases} \tag{3.13}$$

is defined as [5, chapter 5],

$$\mathcal{L}(x, \lambda) = f_0(x) + \lambda_1^T(g(x) - a) + \lambda_2^T(h(x) - b) - vx \tag{3.14}$$

with the λ_i and v being the lagrange multipliers for the inequality and equality constraints, respectively. Based on (3.14) we can formulate the Lagrangian for the primal problem (3.11), where $f_0(x) = 0$, as

$$\begin{aligned}\mathcal{L}(\mathbf{X}, \lambda_1, \lambda_2, \lambda_3, v) = &- \lambda_1^T\mathbf{B}\mathbf{X}\mathbf{e_1} - tr(\lambda_2^T\mathbf{B}\mathbf{X}\mathbf{B}^T) - tr(\lambda_3\mathbf{X}) \\ &+ \sum_{i=1}^{c} v_{2,i}tr(\mathbf{Q}_i\mathbf{X}) + v_1 tr(e_1e_1^T\mathbf{X}) - v_1,\end{aligned} \tag{3.15}$$

with the Lagrange multipliers $\lambda_1 \in \mathcal{R}^{n_b}$,$\lambda_2 \in \mathcal{S}^{n_b}$,$\lambda_3 \in \mathcal{S}^{n_\xi}$,$v_1 \in \mathcal{R}$ and $v_2 \in \mathcal{R}^c$. As it can be seen in (3.15) the signs in front of each term corresponding to the inequalities in (3.11) are negative. The reason to this is that inequality signs in the minimization problem in (3.13) is chosen to have the opposite direction, i.e., $\leq$. Because of the cyclic property of the trace operator saying that $tr(ABC) = tr(BCA) = tr(CAB)$ we rewrite the first and second terms in (3.15),

$$tr(\lambda_2^T \mathbf{B}\mathbf{X}\mathbf{B}^T) = tr(\lambda_2^T \mathbf{B}^T \mathbf{X}\mathbf{B})$$

and

$$\begin{aligned}\lambda_1^T \mathbf{B}\mathbf{X}\mathbf{e_1} &= tr(\mathbf{e}_1 \frac{\lambda_1^T}{2}\mathbf{B}\mathbf{X} + tr(\mathbf{e}_1^T \frac{\lambda_1}{2}\mathbf{B}^T\mathbf{X}) \\ & tr((\mathbf{e}_1 \frac{\lambda_1^T}{2}\mathbf{B} + \mathbf{e}_1^T \frac{\lambda_1}{2}\mathbf{B}^T)\mathbf{X})\end{aligned}$$

Since the definition of the dual function is the minimum of the Lagrangian (3.15) over $x \in \mathbf{X}$, we are to solve the following problem first,

$$\texttt{Inf}_{x\in\mathbf{X}}\mathcal{L}(X, \lambda_1, \lambda_2, \lambda_3, v).$$

We can see that, all but the last term in (3.15) are linear in $\mathbf{X}$. This linearity can be described by the slope matrix,

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{X}} = \mathbf{e}_1\lambda_1^T\mathbf{B} + \mathbf{B}^T\lambda_1\mathbf{e}_1^T + \mathbf{B}^T\lambda_2\mathbf{B}\lambda_3 + v_1\mathbf{e}_1\mathbf{e}_1^T + \sum_{i=1}^{c} v_{2,i}\mathbf{Q}_i$$

For the derivative roles of the trace functions see [2]. This means that the Lagrangian minimum with respect to $\mathbf{X}$ is -∞, which is not acceptable. In order to avoid this, the expression in $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{X}}$ is set to zero, by including the constraint $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{X}} = 0$ in the dual problem. As a result we have,

$$\texttt{Inf}_{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{X}}=0}\mathcal{L}(X, \lambda_1, \lambda_2, \lambda_3, v) = v_1 \tag{3.16}$$

This leads to the following dual problem for the relaxed problem (3.11),

$$D(P): \begin{cases} \texttt{maximize} & v_1 \\ \texttt{s.t.} & \mathbf{e}_1\lambda_1^T\mathbf{B} + \mathbf{B}^T\lambda_1\mathbf{e}_1^T + \mathbf{B}^T\lambda_2\mathbf{B} \\ & \lambda_3 + v_1\mathbf{e}_1\mathbf{e}_1^T + \sum_{i=1}^{c} v_{2,i}\mathbf{Q}_i = 0 \\ & \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \succeq 0 \end{cases} \tag{3.17}$$

where λ_i are Lagrange multipliers, $\lambda_1 \in \mathbb{R}^{n_b}, \lambda_2 \in \mathcal{S}^{n_b}, \lambda_3 \in \mathcal{S}^{n_\xi}, v_1 \in \mathbb{R}$ and $v_2 \in \mathbb{R}^c$, see [8, 20]

Theorem 3.1 *Model invalidation*
Given the collections of measurements from the real process. Model (3.3) *is inconsistent with the measurements if* $D(P) \to \infty$, *[17].*
Theorem 3.1 is equivalent with the proposition in [8] and the theorem in [20], which its proof follows from weak duality and is available in [20].

When the aim of parameter identification is estimation of the $\mathbb{SCP}$ (Set of Consistent Parameters), it is important to keep in mind that we are not looking for the $\mathbb{SCP}$ itself. Instead we try to drive an outer approximation for the $\mathbb{SCP}$, in which the actual $\mathbb{SCP}$ is contained. In other words, if we let $\mathcal{P}^*$ denote the actual $\mathbb{SCP}$ and $\overline{\mathcal{P}}^*$ its outer approximation, then

$$\mathcal{P}^* \subseteq \overline{\mathcal{P}}^*$$

Now, no matter if the aim of parameter identification work is the estimation of an outer approximation for the $\mathbb{SCP}$ or model falsification, we start with determining an initial parameter set $\mathcal{P}_0$, $\overline{\mathcal{P}}^* \subseteq \mathcal{P}_0$. For biochemical networks, determining the lower bounds of the initial set is not difficult, since they are all in general positive. Taking other facts into account it is not too difficult to determine the upper bound either. Having decided $\mathcal{P}_0$, if Theorem 3.1 holds, then the model can be rejected as invalid. If ,on the other hand, the purpose of the work is to find a consistency certificate, then $\mathcal{P}_0$ is bisected in subsets $\mathcal{P}_i$ and the dual problem (3.17) is analyzed for each of the subsets. This bisection algorithm is called repeatedly until the weighted volume

$$V(\mathcal{P}) = \int\limits_{\mathcal{P}} w(\mathtt{p})\mathrm{d}\mathtt{p}$$

is smaller than a threshold ϵ, see algorithm (3). The algorithm of bisection and also the mentioned volume is implemented in the MATLAB TOOLBOX, [10]. An outer approximation of the $\mathbb{SCP}$ is then the union of all the subsets containing parameter values consistent with the feasibility problem (3.6), i.e

$$\overline{\mathcal{P}}^* = \mathcal{P}_0 \backslash \bigcup_{\mathcal{I}} \mathcal{P}_{\mathcal{I}}$$

where $\bigcup_{\mathcal{I}} \mathcal{P}_{\mathcal{I}}$ marks the union of the rejected parameter sets, i.e. for which the problem proved infeasible and $\mathcal{I}$ denotes the set of integers, containing the indexes of the rejected subsets .

As stated before, a model can be rejected as invalid, whenever the algorithm returns empty, meaning that there does not exit a parameter value $\mathbf{p} \in \mathcal{P}_0$ consistent with the measurements, states and model input.

Algorithm 3: Analyze $\mathcal{P}(\mathcal{U},\mathcal{X},\mathcal{Y},\mathcal{P})$

```
Given a parameter set $\mathcal{P}$
  1. If $V(\mathcal{P}) < \epsilon$, return $\mathcal{P} = \mathcal{P}$
  2. Check feasibility of $D_j(\mathcal{U},\mathcal{X},\mathcal{Y},\mathcal{P}), \forall j \in I(0, N-m)$
  3. If $sup\{v_1^*, j | j \in I(0, N-m)\} = \infty$, return $\mathcal{P} = \emptyset$
  4. If $sup\{v_1^*, j | j \in I(0, N-m)\} \neq \infty$:
     (a) Bisection of $\mathcal{P}$ in $\mathcal{P}_1$ and $\mathcal{P}_2$
     (b) $\mathcal{P}_1 =$ Analyze $\mathcal{P}(\mathcal{U},\mathcal{X},\mathcal{Y},\mathcal{P}_1)$
     (c) $\mathcal{P}_2 =$ Analyze $\mathcal{P}(\mathcal{U},\mathcal{X},\mathcal{Y},\mathcal{P}_2)$
     (d) Return $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$
```

Chapter 4

Numerical Analysis, Error bounding

This chapter will cover the estimation of an upper bound for the error caused by the time discretization of a time-continuous system. The error estimation is done only for the linear systems. Computing this error bound is very difficult for a nonlinear system, without being overly conservative, see [17]. Different methods for estimating the error are studied and their advantages and disadvantages are discussed. In section §4.1 the $\|.\|_2$-norm is studied, where the error is expressed as a difference between the analytical expression for the solution of the ODE-model and its time-discrete counterpart. Section §4.2 explores the usage of $\|.\|_\infty$-norm. In this section we will also apply $\|.\|_\infty$-optimization on a method called one-step error estimation.

In this chapter, states representing both time-continuous and time-discrete will appear together and therefore we will introduce indexed states, like x_C for time-continuous and x_D for time-discrete to avoid confusion.

When the system is linear, given as

$$\begin{aligned} \dot{x}_C =& f(x_C;p) \\ =& P(p)x_C,\ x(0) = x_0 \\ y =& g(x_C;p), \end{aligned} \tag{4.1}$$

we can easily find its analytical solution, which is usually a matrix exponential function of the unknown parameters p, i.e.

$$x_C(p,k,h) = e^{P(p)hk}x_0 \tag{4.2}$$

for the kth time instant and sampling rate h. The analytical expression for the discrete system $x_D(.)$ is obtained using Euler's method for discretization, i.e.,

$$x_D^{(k)} = x_D^{(k-1)} + hf(x_D^{(k-1)},p) \tag{4.3}$$

where the given initial value $x(0)$ is used for approximation of $x_D^{(k)} = x_D^{(1)}$, i.e. when $k = 1$. An analytical expression for an arbitrary time instant k and sampling rate h, given that the time-continuous system is given by (4.1), is obtained in the following manner

$$\begin{aligned} x_D^{(1)} =& x_0 + hP(p)x_0 \\ =&(I + hP(p))x_0 \\ x_D^{(2)} =& x_D^{(1)} + hP(p)x_D^{(1)} \\ =&(I + hP(p))x^{(1)} \\ =&(I + hP(p))^{(2)}x_0 \\ &\vdots \\ x_D^{(k)} =& x_D^{(k-1)} + hP(p)x_D^{(k-1)} \\ =&(I + hP(p))^{(k)}x_0. \end{aligned} \tag{4.4}$$

In the subsections that follows, we try to compute an upper bound for the discretization error $e_D^{(k)}$.

4.1 The method of $\|.\|_2$

The basic idea for the method of singular value analysis and $\|.\|_2$-optimization is to bound the whole error expression by a constant upper bound, as mentioned before. This is done by using the relationship between the singular values of a matrix and its induced $\|.\|_2$, and by finding the maximum of the singular values for $P(p_i, k, h)$, for all possible values of p_i.

Consider an unconstrained and convex problem

$$\texttt{minimize } \left\|P(p)\right\|_2 \tag{4.5}$$

where $P(p) = P_0 + p_1P_1 + ... + p_nP_n, P_i \in \mathbb{R}^{r\times q}$ and $p_i \in \mathbb{R}^n$. The convexity follows from the fact that $\left\|P(p)\right\|_2$ is a convex function, see section §2.2. Assume a $\lambda \geq 0$ we can then state that $\left\|P(p)\right\|_2 \leq \lambda$, if and only if

$$P(p)^TP(p) \preceq \lambda^2 I$$

where I and $\preceq$ denotes the unit matrix and *positive semi-definiteness* respectively. We can then express the optimization problem (4.5) as

$$\begin{aligned} &\texttt{minimize } \lambda \\ &\texttt{subject to } P(p)^TP(p) \preceq \lambda I \end{aligned} \tag{4.6}$$

which also is a convex optimization problem, since $P(p)^TP(p) - \lambda^2 I$ is matrix convex in (p, λ). Given that we can express $P(p)^TP(p) \preceq \lambda^2 I$ as a linear matrix inequality, we have the following equivalence

$$P(p)^T P(p) \preceq \lambda^2 I \iff \begin{bmatrix} \lambda I & P(p)^T \\ P(p) & \lambda I \end{bmatrix} \succeq 0 \tag{4.7}$$

for $\lambda \geq 0$, resulting in the SDP problem bellow,

$$\begin{aligned} &\texttt{minimize } \lambda \\ &\texttt{subject to } \begin{bmatrix} \lambda I & P(p)^T \\ P(p) & \lambda I \end{bmatrix} \succeq 0 \end{aligned} \tag{4.8}$$

Going back to the error estimation problem, the result above is used to compute an upper bound for the discretization error, expressed as a difference between the time-continuous system, x_C and its time-discrete counterpart, x_D

$$e_D^{(k)} = x_C(p, kh) - x_D^{(k)}(p, h) \tag{4.9}$$

where $x_C(p, kh)$ is a matrix exponential function, see (4.2). In the expression above the time t has been replaced by kh, to avoid inconvenience in the notation. The time under which the time-continuous system is simulated and the sample rate h, at which the the samples has been collected gives the total number of the samples, k. The analytical expression for $x_D^{(k)}$ is given by (4.4).

To motivate the use of $\|.\|_2$ we take the $\|.\|_2$ of the error expression (4.9)

$$\begin{aligned} \left\|e_D^{(k)}\right\|_2 =& \left\|x_C(p, kh) - x_D^{(k)}(p, h)\right\|_2 \\ =& \left\|e^{P(p)kh} x_0 - (I + hP(p))^{(k)} x_0\right\|_2 \end{aligned} \tag{4.10}$$

and make use of Cauchy Schwartz and triangle inequality resulting in

$$\begin{aligned} \left\|e_D^{(k)}\right\|_2 &\leq \left(\left\|e^{P(p)kh}\right\|_2 + \left\|(I + hP(p_1))^k\right\|_2\right) \|x_0\|_2 \\ &\leq \left(\sigma_{max}(e^{\lambda_1 I}) + \sigma_{max}(\lambda_2 I)^k\right) \|x_0\|_2 \end{aligned} \tag{4.11}$$

where σ_{max} represents the maximum singular value of the respective matrices which can be calculated using the results given by equations (4.5)-(4.8) and λ_1 and λ_2 are given as follows

$$\begin{aligned} &\texttt{minimize}_{\lambda_1} \lambda_1 \\ &\texttt{subject to } \begin{bmatrix} \lambda_1 I & (P(p,k,h))^T \\ P(p,k,h) & \lambda_1 I \end{bmatrix} \succeq 0 \\ &\forall \qquad \alpha \leq p \leq \beta \end{aligned} \tag{4.12}$$

and similarly

$$\begin{array}{ll} \texttt{minimize}_{\lambda_2} \lambda_2 & \\ \texttt{subject to} \begin{bmatrix} \lambda_2 I & ((I+hP(p))^k)^T \\ (I+hP(p))^k & \lambda_2 I \end{bmatrix} \succeq 0 & \\ \forall \qquad \alpha \leq p \leq \beta & \end{array} \tag{4.13}$$

where α and β marks the given range for the unknown parameters.

The upper bound for the error obtained this way can be reduced via interval analysis, i.e., examining smaller regions of the parameter space, which would possibly lead to model rejection. However this requires the calculation of the largest maximum singular value of the matrix $P(p)$ and $(I+hP(p))$ with respect to all the possible values of parameters, which is in general not possible to solve. However, since in the problems above $P(p,k,h)$ and $(I+hP(p))$ are both linear functions of the parameter space,p, it would still be possible to solve.

Since the optimization problems above are NP-hard, meaning that they are in general not easy or possible to solve, we look at alternative approaches, one of which is presented in next subsection.

4.2 The method of $\|.\|_\infty$

We can make use of the relation $\left\|e_D^{(k)}\right\|_\infty \leq \sqrt{n}\left\|e_D^{(k)}\right\|_2$, see section §2.3 equation (2.46), and compute a tighter upper approximation for the discretization error. In this subsection we will look at two ways of expressing the discretization error. The first one, still the difference between the true and approximative solutions given by (4.9) and the second is the so called one-step method, see [6, 7]. For the one-step method two different scenarios are considered,

(i) a set of measurements, i.e., outputs of the time-continuous model are given,

(ii) no measurements given, but initial values are given.

For the scenario (i) consider a linear state space equation

$$\begin{aligned} \dot{x}_C &= f(x;p) \\ &= P(p)x_C,\ x(0) = x_0 \\ y &= g(x;p) \end{aligned} \tag{4.14}$$

where y denotes the output vector from the model. A Taylor series expansion of x_C around zero and step size h gives.

$$x_C(0+h) = \underbrace{x_C(0) + h\dot{x}_C(0)}_{x_D \text{ given by Euler}} + \frac{h^2}{2!}\ddot{x}_C(\epsilon) \tag{4.15}$$

where $0 \leq \epsilon \leq h$ for the first time instant and where, as indicated, the first two terms make the formula for Euler's discretization method. Taylor expansion of x_C around an arbitrary point $k-1$ is done in the same manner, resulting in $x_C(kh) = x_D^{(k)} + e_D^{(k)}$. Having said that, we formulate the one-step error for time instant k as

$$e_D^{(k)}(h) = \frac{h^2}{2!}(f_x f)(x_C(\epsilon_k)) \tag{4.16}$$

where $h_{k-1} \leq \epsilon_k \leq h_k$. In this thesis, the system (4.14), is always linear, thus the derivative $f_x f(x_C(\epsilon_k)) = P(p)^2 x_C(\epsilon_k)$, resulting in

$$e_D^{(k)}(h) = \frac{h^2}{2}(P(p)^2 x_C(\epsilon_k)). \tag{4.17}$$

Based on the result of equation (4.2), we can express $x_C(\epsilon_k)$ as

$$x_C(\epsilon_k) = e^{P(p)\epsilon_k} x_C(\epsilon_{k-1}),\ h_{k-1} \leq \epsilon_k \leq h_k \tag{4.18}$$

where the $(k-1)$th given measurement is used as initial value for obtaining $x_C(\epsilon_k)$. Now applying $\|.\|_\infty$ on $e_D^{(k)}(h)$, equation (4.17) and making use of (4.18) results in

$$\begin{aligned} \left\| e_D^{(k)}(h) \right\|_\infty &= \left\| \frac{h^2}{2} P(p)^2 x_C(\epsilon_k) \right\|_\infty \\ &\leq \frac{h^2}{2} \left\| P(p)^2 \right\|_\infty \left\| x_C(\epsilon_k) \right\|_\infty \\ &\leq \frac{h^2}{2} \|\lambda\|_\infty^2 \left\| e^{\lambda I} \right\|_\infty^{\epsilon_k} \left\| x_C(\epsilon_{k-1}) \right\|_\infty \end{aligned} \tag{4.19}$$

where I stands for the identity matrix. The discretization error, for the one-step method is thus

$$e_D^{(k)}(h) \leq \frac{h^2}{2} \|\lambda\|_\infty^2 \left\| e^{\lambda I} \right\|_\infty^{\epsilon_k} \left\| x_C(\epsilon_{k-1}) \right\|_\infty,\ 0 \leq \epsilon_k \leq h_k \tag{4.20}$$

where λ is the optimization variable, explained later in this section. The relation $0 \leq \epsilon_k \leq h_k$ comes from the fact that we are now using the $(k-1)$th measurement as initial value for computing $x_C(\epsilon_k)$, as stated before, and the distance between the $(k-1)$th and kth measurements is just h_k.

For the scenario (ii) where only the initial values and no measurements are given, one can use the approximative data, given by the time-discrete model. This results in an error expression $e_D^{(k)}$, for the one-step method, where the discrete data $x_D^{(k-1)}$ is used as initial point, resulting in the following error expression

$$\begin{aligned}
\left\|e_D^{(k)}(h)\right\|_\infty &\leq \frac{h^2}{2}\|\lambda\|_\infty^2\left\|e^{\lambda I}\right\|_\infty^{\epsilon_k}\|x_0\|_\infty \text{, if } k=1\\
&\text{and}\\
\left\|e_D^{(k)}(h)\right\|_\infty &\leq \frac{h^2}{2}\|\lambda\|_\infty^2\left\|e^{\lambda I}\right\|_\infty^{\epsilon_k}\left\|x_D^{(k-1)}+e^{(k-1)}\right\|_\infty\\
&= \frac{h^2}{2}\|\lambda\|_\infty^2\left\|e^{\lambda I}\right\|_\infty^{\epsilon_k}\left\|(I+h\lambda I)^{(k-1)}x_0+\frac{h^2}{2}\|\lambda\|_\infty^2\left\|e^{\lambda I}\right\|_\infty^{\epsilon_{k-1}}(I+h\lambda I)^{(k-2)}x_0\right\|_\infty\\
&= \frac{h^2}{2}\|\lambda\|_\infty^2\left\|e^{\lambda I}\right\|_\infty^{\epsilon_k}\left(\left\|(I+h\lambda I)^{(k-1)}+\frac{h^2}{2}\|\lambda\|_\infty^2\left\|e^{\lambda I}\right\|_\infty^{\epsilon_{k-1}}(I+h\lambda I)^{(k-2)}\right\|_\infty\right)\|x_0\|\\
&\text{if } k\neq 1
\end{aligned}$$

$$(4.21)$$

where the first row on the right of the inequality sign is the discretization error for $k=1$ and thus the same as in (4.20) for $k=1$. For any other time instant $k\neq 1$, the discrete data $x_D^{(k-1)}$ is used as initial value for estimation of $e_D^{(k)}$. The term $e^{(k-1)}$ is the propagating error from discretization of $x_D^{(k-1)}$. In the third row, we have made use of the analytical expression for the time-discrete system given by (4.4) and finally, the last row makes the general expression of the discretization error for this method. We can by inspection see that this method must be more conservative than the one given in the previous case. One of the reasons to this might be that here we are using an already erroneous data $x_D^{(k-1)}$ for estimation of $e_D^{(k)}$. The other reason is that the errors from previous steps accumulate through the whole range of time horizon k.

The method of $\|.\|_\infty$ has also been used for the method where the error was expressed as the difference between the time-continuous and the time-discrete systems. The same procedure of applying $\|.\|_\infty$, Cauchy Schwartz and the triangle inequality results in

$$\begin{aligned}
\left\|e_D^{(k)}(h)\right\|_\infty &= \|x_C-x_D\|_\infty\\
&= \left\|(e^{P(p)kh}-(I+hP(p))^k)x_0\right\|_\infty\\
&\leq \left(\left\|e^{P(p)kh}\right\|_\infty+\left\|(I+hP(p))^k\right\|_\infty\right)\|x_0\|_\infty\\
&\leq \left(\left\|e^{\lambda Ikh}\right\|_\infty+\left\|(I+h\lambda I)^k)\right\|_\infty\right)\|x_0\|_\infty
\end{aligned} \qquad (4.22)$$

The parameter λ in (4.20),(4.21) and (4.22) is computed using the $\|.\|_\infty$, which for a matrix $P(p)$, where each of its component represent a variable, is defined as,

$$\|P(p)\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |p_{ij}| \,. \tag{4.23}$$

where p_{ij} represent the element in ith row and jth column. The relation in (4.23) is simply saying that the $\|.\|_\infty$ of a matrix is obtained as the maximum of sum of the absolute value of elements in its rows. Considering a matrix

$$A = \begin{pmatrix} -1 & 2 \\ 2 & -2 \end{pmatrix}$$

then

$$\begin{aligned} \|A\|_\infty &= \max_{i=1,2} \begin{pmatrix} |-1| + |2| \\ |2| + |-2| \end{pmatrix} \\ &= \max_{i=1,2} \begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ &= 4 \end{aligned}$$

see [5, 12]. For computation of the parameter λ, we solve the optimization problem bellow

$$\begin{aligned} &\texttt{minimize}_\lambda \lambda \\ &\texttt{subject to} \sum_{j=1}^{n} |p_{ij}| < \lambda I \qquad \forall i = 1...m, p \in \mathbb{P} \end{aligned} \tag{4.24}$$

where the optimization parameter λ is minimized, as the worst-case-$\|.\|_\infty$ is considered. A generalized yalmip-code implementing this optimization method is developed by Johan Löfberg, see [13], which then has been modified for the purpose of this thesis. This code is embedded in the `biosdp` TOOLBOX.

We can just by inspection of the two studied error expressions, (4.20) and (4.22), conclude certain things. Recalling that the aim of the error bounding task was to find an upper bound, which itself opens for conservativeness of the error bounds. But it is also desired that the computed upper bound is as tight as possible. The nature of the worst-case-matrix-norm, (4.24) , is to compute the smallest possible optimization problem λ, but keep it large enough so that the inequality

$$\sum_{j=1}^{n} |p_{ij}| < \lambda I$$

is still valid. This means that the optimization parameter λ is minimized, but the inequality constraint does not allow it to be too small. The error algorithms,(4.20) and (4.22), are directly dependent in this parameter, resulting in their conservativeness.

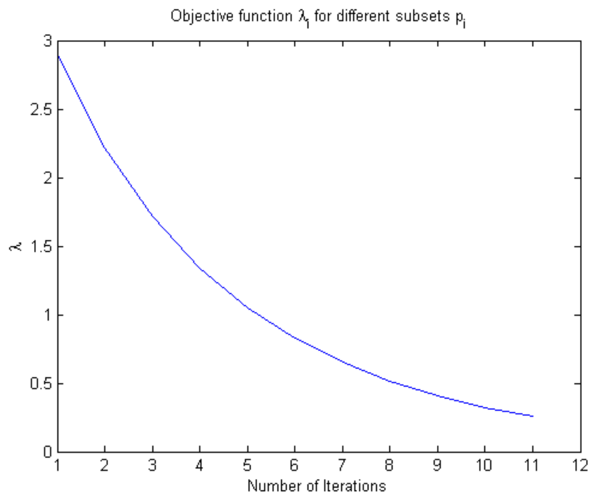


Figure 4.1. Objective function, λ

We can see from (4.22), that this method is too conservative. Using this as a method for bounding the discretization error will not allow rejection of almost any parameter sets. The reason to this is that the extended measurement intervals will be too wide, which makes it possible for much more parameter sets to qualify as consistent with the model.

In the next subsection a case study has been made, where the error bounds are computed, using both the methods given by (4.20) and (4.22).

4.2.1 A case study for $\|.\|_\infty$

In this chapter some simulation results using $\|.\|_\infty$-optimization is presented, with aim to make a comparison between the methods given by (4.20) and (4.22). The system under consideration is

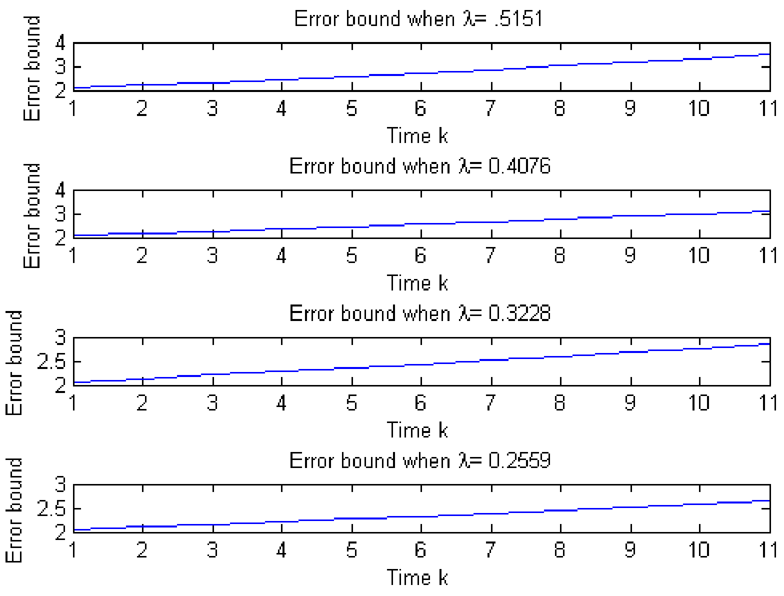
$$\begin{aligned}\dot{x} &= \begin{pmatrix} -p_1 & 0 \\ p_1 & -p_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_{01} \\ x_{02} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ y &= \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\end{aligned} \tag{4.25}$$

with the initial parameter set or box $\mathcal{P}_0$ given by

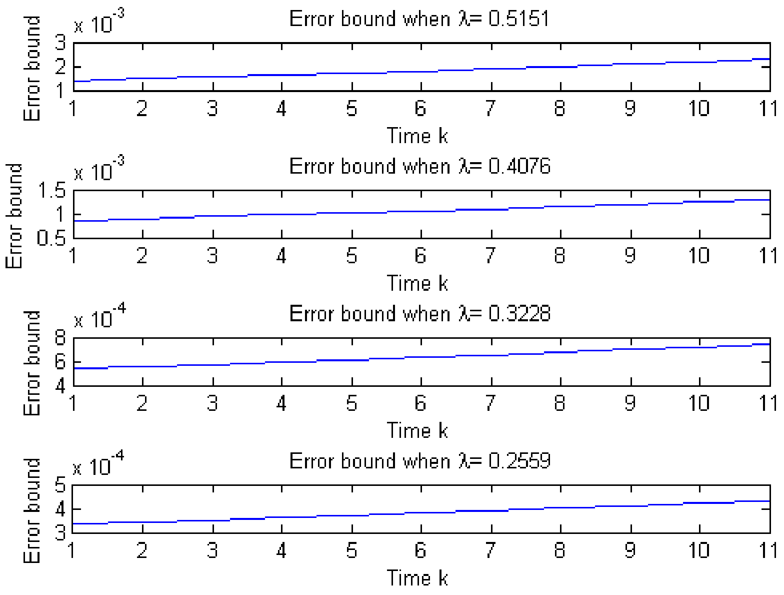
$$\mathcal{P}_0 = [\alpha_1 \quad \beta_1] \times [\alpha_2 \quad \beta_2] = [0.1 \quad 2] \times [0.1 \quad 2].$$

A simple box-shrinking code has been developed, which shrinks the intervals and after each iteration,i.e for each new parameter set, the optimization problem (4.24) is solved. Figure 4.1 illustrates the changes in the λ-value as function of p_i.

Figures 4.2(a) and 4.2(b) illustrate the resulting error bounds for the methods (4.22) and (4.20) respectively. As the comparison shows and also discussed before, the method given by (4.22) is too conservative, compared to the other one, given in (4.20).



(a) Error bounds using the method given by equation (4.22)



(b) Error bounds for the method given by equation (4.20)

Figure 4.2. Figures 4.2(a) and 4.2(b) illustrate the error bounds for the discretization error, as expressed in equations (4.22) and (4.20). As the bounds illustrate, the method (4.22) is too conservative compared to the one-step method, (4.20)

Chapter 5

Implementation

In this chapter, the theories presented this far is applied on some system biological problems. The method of SDP-relaxation and model falsification is implemented in the MATLAB TOOLBOX, [10]. What the implementation problems really represent in detail, from a biological point of view, is not discussed here. This will by no means limit the focus and aim of this thesis, since the problems solved in this chapter are treated as what ever mathematical formulation of some physical relations. How a problem should be set up in the mentioned TOOLBOX will not be discussed here, instead the reader is referred to the source, see [10].

All the systems studied, when using the TOOLBOX `bio.sdp`, are linear because of the limitations in the algorithm developed for estimating the discretization error. For the purpose of illustrating the effect of the discretization error, each problem is solved both with and without this error taken into consideration. When the discretization error is considered, the model output, as stated in (3.5) section §3, is extended with the computed error bounds

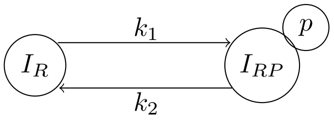
$$\begin{aligned} \bar{\mathbf{y}}^{(k)} &= \underbrace{\mathbf{y}^{(k)} \pm \mathbf{e}^{(k)}}_{\text{noisy data}} \pm \mathbf{e}_D^{(k)} \\ k &= 0, ..., N, \end{aligned} \tag{5.1}$$

where $N+1$ is the number of measurements and $\mathbf{e}_D^{(k)}$ represent the discretization error, given by equation (4.20), section §4.2. Furthermore, since the measurements contain some uncertainty and therefore each uncertain measurement is an interval, the measurement vector is extended with bounds for the discretization error.

The methods of interval analysis, described in section §2.1 are also illustrated, by solving some problems. For solving problems by these methods, the MATLAB TOOLBOX SCS is used, see [18].

5.1 Problem 1, Conversion reaction

The first problem to be simulated is a conversion reaction, where the states, I_R and I_{RP}, represent some biological substances, in which an amount of the substance I_{RP} is given, and the reaction in its whole describes a dephosphorylation of this substance to I_R. The parameters to be identified are k_1 and k_2 respectively.



The relationship described by the state diagram can be expressed as a system of differential equations,

$$\begin{aligned}\frac{dI_R}{dt} &= -k_1 I_R + k_2 I_{RP}\\ \frac{dI_{RP}}{dt} &= k_1 I_R - k_2 I_{RP}.\end{aligned} \tag{5.2}$$

To simplify the notations, we introduce new state and parameter variables, letting $I_R = x_1$, $I_{RP} = x_2$, $k_1 = p_1$ and $k_2 = p_2$. The task is now to identify an outer approximation for the parameter set, such that the output of the following model is inside the given uncertainty boundaries.

$$\begin{aligned}\dot{x}(t,p) &= \begin{pmatrix} -p_1 & p_2 \\ p_1 & -p_2 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}\\ y(t,p) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix},\ x(0) = \begin{pmatrix} 0 & 1 \end{pmatrix}^T\end{aligned} \tag{5.3}$$

The initial search box for the unknown parameters is assumed to be,

$$\mathbb{P}_0 = [\underline{p}_1, \overline{p}_1] \times [\underline{p}_2, \overline{p}_2] = [0, 10] \times [0, 10].$$

This problem is simulated and the resulting $\mathbb{SCP}$ is shown by Figures 5.1 and 5.2. Figure 5.1 shows the outer approximation of the $\mathbb{SCP}$, where the discretization error is not taken in consideration. As it has been pointed out before, the method used for discretization of the time-continuous system is not reliable and as a consequence the discrete model is erroneous. In order to compensate for the discretization error, each uncertain measurement interval is extended with the discretization error bounds, computed as discussed in section §4.2. The resulting outer approximation for the $\mathbb{SCP}$ is shown by Figure 5.2.

When it comes to the dependency of the discretization error on the sampling rate h, it is obvious that the smaller this parameter is the smaller the resulting error bound will be. This can be seen from a direct inspection of the algorithm used for estimation of the error bound, where $e_D^{(k)}$ is directly dependent on h,

$$e_D^{(k)}(h) \leq \frac{h^2}{2} \|\lambda\|_\infty^2 \left\| e^{\lambda I} \right\|_\infty^{\epsilon_k} \left\| x_C(\epsilon_{k-1}) \right\|_\infty,\ 0 \leq \epsilon_k \leq h_k \tag{5.4}$$

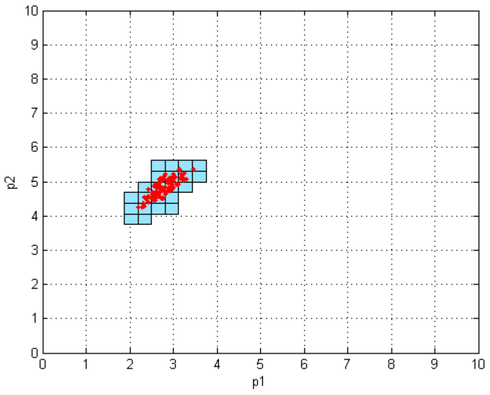


Figure 5.1. Outer approximation for the set of consistent parameters, where the discretization error is not taken in consideration, time step $h = 0.1$. The data in red represent the Monte-Carlo parameters for which the discretized counterpart of the problem (5.3) is feasible. see [10] and [3].

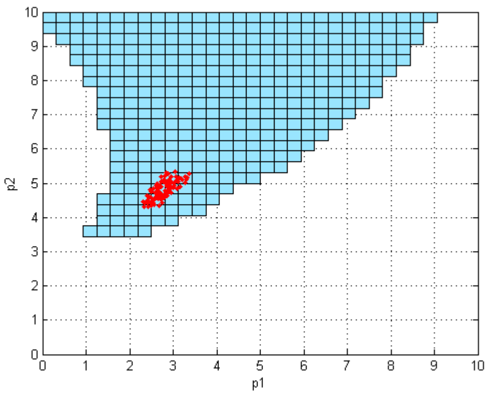


Figure 5.2. Outer approximation for the set of consistent parameters, where the discretization error is taken in consideration, time step $h = 0.1$. The algorithm for computing the upper bound for the discretization error introduces a pessimism in the set of consistent parameters. As a result the subsets of the initial parameter set that can be rejected decreases.

From (5.4) we can conclude that, the smaller the sample rate is, the faster $e_D^{(k)}(h)$ will decrease, i.e.,

$$\lim_{h\to 0} e_D^{(k)}(h) = 0. \tag{5.5}$$

5.1.1 Simulation of the time-continuous system, using sivia

Let us solve the coversion reaction problem,(5.3) with the method of SIVIA and the SCS toolbox. For this purpose we consider the analytical solution of (5.3)

$$\begin{aligned} x(t,p) &= e^{\begin{pmatrix} -p_1 & p_2 \\ p_1 & -p_2 \end{pmatrix} t} \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} \\ y(t,p) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1(t,p) \\ x_2(t,p) \end{pmatrix}, \quad x(0) = \begin{pmatrix} 0 & 1 \end{pmatrix}^T \end{aligned} \tag{5.6}$$

where the parameters to be identified is as before p_1 and p_2. For this problem we let the subpaving $\mathbb{Y}$ represent the union of sets in which the solution sets to (5.6) is included. The subpaving $\mathbb{Y}$ consist of the uncertain measurements given for the conversion reaction problem , see Table 5.1.

The set of feasible parameters p_1 and p_2 consistent with the measurements is characterized as

$$\mathbb{SCP} = \{p \in \mathbb{P}_0 | x(t_1, p) \in [y(t_1)], ..., x(t_5, p) \in [y(t_5)]\} \tag{5.7}$$

t	$[y]$
0.1	$[0.00, 0.10]$
0.2	$[0.44, 0.54]$
0.3	$[0.57, 0.67]$
0.4	$[0.60, 0.70]$
0.5	$[0.56, 0.66]$

Table 5.1. The uncertain measurements for the problem conversion reaction, given as intervals

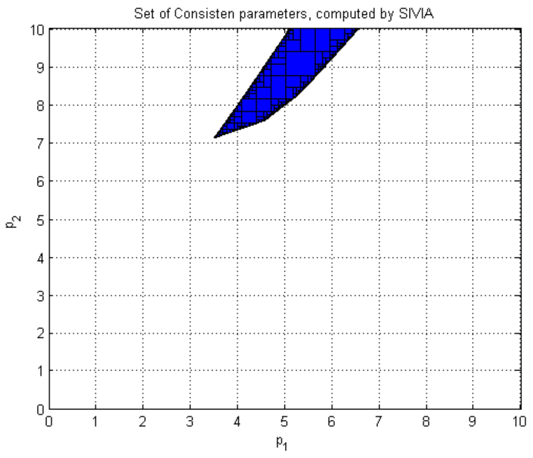


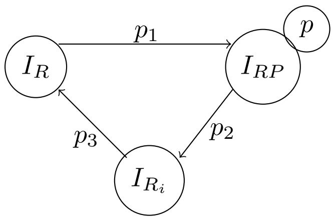
Figure 5.3. Set of consistent parameters $\mathbb{SCP}$ computed by the method SIVIA for the conversion reaction problem (5.6).

where $\mathbb{P}_0$ is the initial parameter box $[0, 10] \times [0, 10]$. The resulting outer approximation for the set of consistent parameters is shown by Figure 5.3

Figures 5.1 and 5.3 both illustrate the outer $\mathbb{SCP}$ for the same problem, but two models. For generating the $\mathbb{SCP}$ illustrated by Figure 5.1, an approximative model has been used, approximated using Euler's discretization method. A comparison between the resulting $\mathbb{SCP}$s shows that the discretized model is corrupted by the discretization error. Figure 5.2 shows the $\mathbb{SCP}$ when this discretization error is taken into consideration. This upper bound for the discretization error covers both the $\mathbb{SCP}$s resulting in a nonempty intersection between them, which is desired, despite it's conservativeness.

5.2 Problem 2, Internalization

This problem is a three dimensional phosphorization problem, where the substance I_R is phosphorized to I_{RP}. $I_{R_{tot}}$ denotes the total amount of the concentrations available, which in this specific case is assumed to be 100 units. The internal state, $I_{Ri} = I_{R_{tot}} - I_R - I_{RP}$, from which the title is inspired, has probably some chemical/biological significance, but since this is substituted and does not effect the ODE-description of the model, we accept it just as "some thing".



$$I_{Ri} = I_{R_{tot}} - I_R - I_{RP}$$

The relations described by the state diagram above can be expressed as a system of differential equations,

$$\begin{aligned}\frac{dI_R}{dt} &= -p_1 I_R + p_3 I_{Ri} \\ \frac{dI_{RP}}{dt} &= p_1 I_R - p_2 I_{RP}.\end{aligned} \tag{5.8}$$

Using the relation $I_{Ri} = I_{R_{tot}} - I_R - I_{RP}$ we can rewrite (5.8) in an equivalent form

$$\begin{aligned}\frac{dI_R}{dt} &= -(p_1 + p_3) I_R + p_3 (I_{R_{tot}} - I_{RP}) \\ \frac{dI_{RP}}{dt} &= p_1 I_R - p_2 I_{RP}.\end{aligned} \tag{5.9}$$

We introduce new state variables and let $x_1 = I_R, x_2 = I_{RP}$ and $x_0 = I_{R_{tot}}$ resulting in the following state space model

$$\begin{aligned}\dot{x}(t,p) &= \begin{pmatrix} -(p_1+p_3) & -p_3 \\ p_1 & -p_2 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} p_3 \\ 0 \end{pmatrix} x_0 \\ y &= \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}.\end{aligned} \tag{5.10}$$

One major difference from example 1, beside the intermediate state I_{Ri}, is that the sample rate h is a vector, with varying length for different time points. The initial parameter space for this problem is assumed to be

$$P_0 = [\underline{p_1}, \underline{p_2}, \underline{p_3}] \times [\overline{p_1}, \overline{p_2}, \overline{p_3}] = [10^{-6}, 10^{-6}, 10^{-6}] \times [10^6, 10^6, 10^6].$$

This problem is unfortunately not possible to solve, when the discretization error is taken into consideration, because of numerical problems. The error expression, as given in (4.20), contains matrix exponentials, which for parameter values of this size generate error bounds that are either undefined or infinitely large.

However, this problem is simulated in the `biosdp` TOOLBOX, and the resulting outer approximation for the $\mathbb{SCP}$ is shown by Figure 5.4. Since the discretization error is not possible to compute for this problem and neither any other information about this model is available, we can not draw any conclusions regarding the consistency of the $\mathbb{SCP}$ illustrated by Figure 5.4.

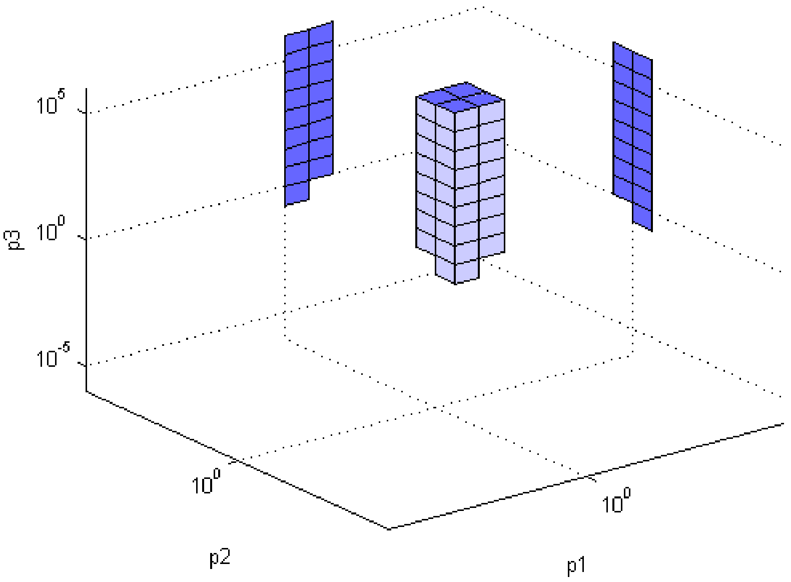


Figure 5.4. Outer approximation for the set of consistent parameters, where, because of numerical difficulties, it has not been possible to compute the discretization error. The gray area shows the 3-D set of consistent parameters and the blue areas its projection on the parameter planes. Since the discretization error is not possible to compute for this problem and neither any other information about this model is available, we can not draw any conclusions regarding the consistency of the $\mathbb{SCP}$.

5.3 Reachable States

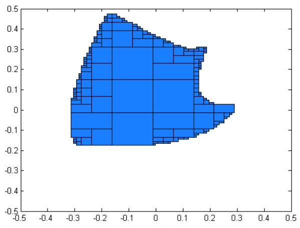
This example is taken directly from [18]. The example is used to illustrate the image evaluation algorithm, ImageSp, described in section §2.1. The principle of reachable states is simply saying that starting from a set of initial states $[\mathbf{x}_0]$, it is possible to compute a set $\mathbb{X}(k)$, which contains the state $\mathbf{x}(k)$. The system under consideration is the following quadratic system,

$$\begin{aligned}x_1(k+1) &= 0.5x_1^2(k) - 0.5x_2^2(k) + 0.4x_1(k)x_2(k) + 0.6x_2(k)\\ x_2(k+1) &= 0.6x_1^2(k) + 0.5x_2^2(k) + 0.6x_1(k)x_2(k) - 0.6x_2(k).\end{aligned} \tag{5.11}$$

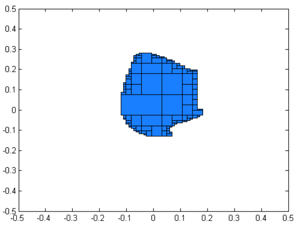
The states x_1 and x_2 are initially assumed to be contained in the box

$$[\mathbf{x}_0] = [-0.38, 0.38] \times [-0.38, 0.38].$$

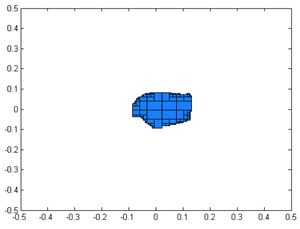
This problem is simulated with the image evaluation method, described by the algorithm (2), section §2.1, for k=30. Figures 5.5(a) - 5.5(f) illustrate the reachable states in the first six time instants. Figures 5.6(a) and 5.6(b) show the results in a longer time horizon, where the results are also compared with the results obtained using Monte Carlo simulation.



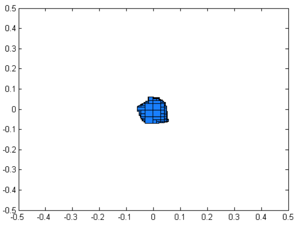
(a) k=1



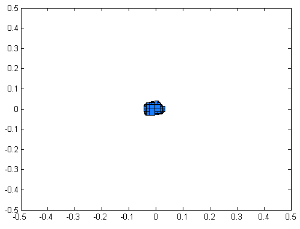
(b) k=2



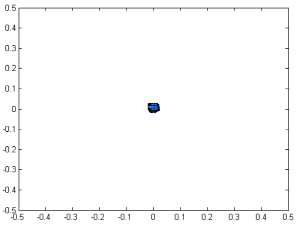
(c) k=3



(d) k=4

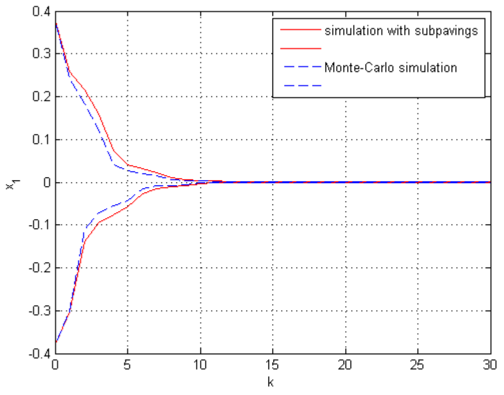


(e) k=5

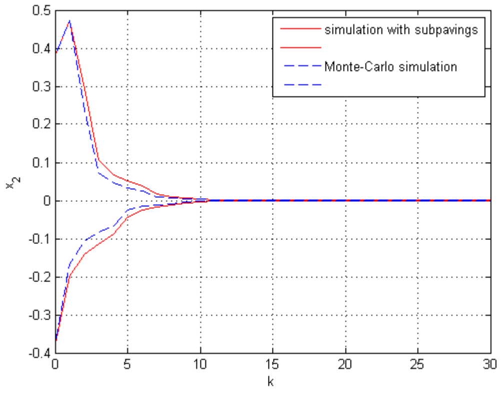


(f) k=6

Figure 5.5. Starting from the initial state box $[x_1(0)] \times [x_2(0)]$ it is possible to compute the box $\mathbb{X}(k)$, which containes $[x_1(k)] \times [x_2(k)]$ at an arbitrary time instant k. Figures 5.5(a) to 5.5(f) illustrate $\mathbb{X}(k)$ for k=1,...,6.



(a) The state x_1 in a longer time horizon



(b) The state x_2 in a longer time horizon

Figure 5.6. Figures 5.6(a) and 5.6(b) show the evolution of the limits for each states variable in a longer time horizon, where the simulation is also compared with a Monte-Carlo simulation of the states

Chapter 6

Discussion and conclusion

In this thesis the problem of parameter identification for system biology, using set based methods, has been studied. We mainly focus on the methods based on SDP-relaxation and interval analysis and we obtain an outer approximation for the $\mathbb{SCP}$. The SDP-relaxation method, based on the work presented in [8], defines an in general non-convex feasibility problem. This feasibility problem is then relaxed to a more computationally efficient feasibility problem, using the method of Semidefinite Programming approach. Such methods although aimed at continuous time model falsification, they use time-discrete reformulations, where the Euler's method is used for the discretization of the continuous ODE-model. Euler's method, however, is erroneous, meaning that the resulting discrete model does not in general behave the same way as the original ODE-model. In the case of parameter identification it means that the set of consistent parameters, obtained using the discrete model, can not be fully trusted, which rises the need for estimating the discretization error.

In this thesis several methods for bounding this discretization error has been studied and tested using different examples. For the numerical examples, it is assumed that the output of the models are given. In the cases where these measurements are available, one can use the method of one-step error estimation and achieve quite satisfactory results, although still conservative. This conservativeness is mainly due to the optimization parameter λ as it is computed as the worst-case-matrix-norm. The error bounds computed based on only the initial values, result in more conservative error bounds.

The other set-based method studied is based on interval analysis and regular subpaving of the parameter space for a time-continuous dynamical system. In this approach the outer approximation for the $\mathbb{SCP}$ is achieved using *Set inverter via interval analysis*, SIVIA. This method works well for low dimensional systems, but when it comes to more complicated and high dimensional problems, this method is quite slow.

Finally, a comparison between the Figures 5.1 and 5.3, both illustrating outer approximations for the $\mathbb{SCP}$s of Problem 1, shows that the discrete model is heavily corrupted by the discretization error. This can be seen by comparing the outer approximations in Figures 5.1 and 5.3, where, the conditions $\mathcal{P}^{CT} \subseteq \mathcal{P}^{DT}$, where $\mathcal{P}^{CT}$ is the set of parameters consistent with the time-continuous model and $\mathcal{P}^{DT}$ is the set of parameters consistent with the time-discrete model, is not satisfied. When the discretization error is taken into consideration, on the other hand, the true $\mathbb{SCP}$ is also covered,as can be seen in Figure 5.2.

Based on the results of problem 1 and the discussions above, we conclude that treating the discretization error is crucial when working with approximative models described by system of difference equations. Without this error estimation taken into consideration, no certain conclusion can be drawn about the time-discrete system biological models.

Bibliography

[1] Margherita Barile. Singleton set, from mathworld–a wolfram web resource, created by eric w. weisstein., November 2011. `http://mathworld.wolfram.com/SingletonSet.html`.

[2] Dennis S. Bernstein. *Matrix Mathematics Theory,facts and formulas with application to linear systems theory*. Princeton University Press, 2005.

[3] K. Binder. *Monte-Carlo Methods*. Wiley Online Library, 1979.

[4] S. Borchers, P. Rumschinski, S. Bosio, R. Weismantel, and R. Findeisen. A set-based framework for coherent model invalidation and parameter estimation of discrete time nonlinear systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6786 –6792, dec. 2009.

[5] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

[6] L. Elden, L. Wittmeyer-Koch, and H. B. Nielsen. *Introduction to Numerical Computation - analysis and* MATLAB *illustrations*. Studentlitteratur, Lund, jul 2004. `http://www2.imm.dtu.dk/pubdb/p.php?3202`.

[7] Torkel Glad. Differential equation models, interval methods and error analysis, june 2011. `http://www.isbgroup.eu/Workshops2011/GladOde_error.pdf`.

[8] J. Hasenauer, S. Waldherr, K. Wagner, and F. Allgöwer. Parameter identification, experimental design and model falsification for biological network models using semidefinite programming. *Systems Biology, IET*, 4(2):119 –130, march 2010.

[9] Jan Hasenauer. Set-based parameter estimation, model falsification and uncertainty analysis using convex optimization, june 2011.

[10] Jan Hasenauer and colleagues. The biosdp toolbox.

[11] L. Jaulin. *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*, volume 1. Springer Verlag, 2001.

[12] J. Lofberg. Yalmip : a toolbox for modeling and optimization in matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284 –289, sept. 2004.

[13] Johan Lofberg. Worst-case matrix norm, March 2011. `http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Blog.Worst-case-matrix-norm`.

[14] R.E. Moore, R.B. Kearfott, and M.J. Cloud. *Introduction to interval analysis*. Society for Industrial Mathematics, 2009.

[15] Francisco Fernandes Castro Rego. Determination of inner and outer bounds of reachable sets. pages 16 –17, may 2011.

[16] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. `http://www.ti3.tu-harburg.de/rump/`.

[17] P. Rumschinski, D. Shona-Laila, S. Borchers, and R. Findeisen. Influence of discretization errors on set-based parameter estimation. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 296 –301, dec. 2010.

[18] S. Tornil-Sin, V. Puig, and T. Escobet. Set computations with subpavings in matlab: The scs toolbox. In *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pages 1403 –1408, sept. 2010.

[19] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, pages 49–95, 1996.

[20] S. Waldherr, R. Findeisen, and F. Allgöwer. Global Sensitivity Analysis of Biochemical Reaction Networks via Semidefinite Programming. *ArXiv e-prints*, April 2009.

[21] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Netherlands, 2000.