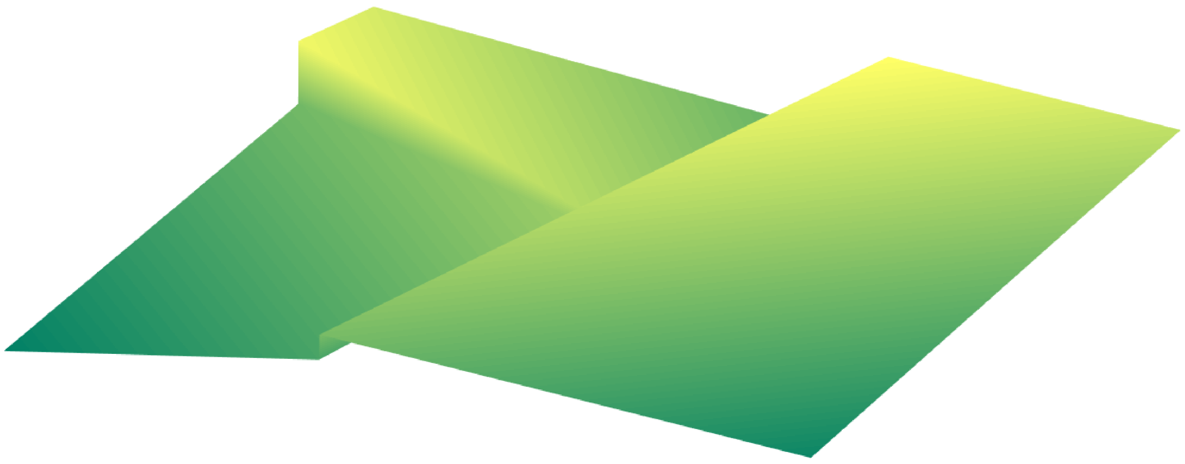
Sparsity based hybrid system identification using a SAT solver

J.H.M. Zwart

Master of Science Thesis



Delft University of Technology

Delft Center for Systems and Control

Sparsity based hybrid system identification using a SAT solver

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J.H.M. Zwart

October 23, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

System identification for switched linear systems from input output data has received substantial attention in recent years. There is a growing interest for techniques that pose the identification problem as a sparse optimisation problem. At the same time a vast amount of research is dedicated to improving SAT solvers which as a result become faster every year. In this work a novel identification method for Switched AutoRegressive eXogenous (SARX) systems and PieceWise AutoRegressive eXogenous (PWARX) systems is proposed that combines sparse optimisation with a SAT solver. The presented method aims to minimise the number of submodels needed to fit the data, while facilitating a prescribed minimum dwell time between switches. The procedure for the identification of switched ARX models is composed of two steps. The First phase determines the switching times in an iterative process aided by a SAT solver. Second, the model parameters and the switching sequence are estimated by optimising the sparsity of a sequence.

The identification procedure for PWARX models operates similarly although it incorporates the knowledge that switches depend on the regressor. An extension to these methods that makes the identification of large datasets tractable is also put forward. The proposed algorithm is evaluated on synthetic systems from the literature and shows promising results. Finally, the proposed algorithm is applied to an experimental benchmark dataset for a non-linear system. All the algorithms proposed in this thesis project are implemented in the form of a toolbox that is made publicly available.

Table of Contents

Preface and Acknowledgements **xi**

1 Introduction **1**

1-1 Notation . . . 3

2 Preliminaries and problem statement **5**

2-1 Switched ARX models . . . 5

2-2 Piecewise affine ARX models . . . 6

2-3 The identification problem . . . 7

2-3-1 Identification of SARX models . . . 8

2-3-2 Identification of PWARX models . . . 8

2-3-3 Research goals . . . 9

2-4 Boolean Satisfiability Problem . . . 10

2-4-1 Cardinality constraints . . . 10

2-4-2 SMT . . . 11

2-4-3 Lazy SMT . . . 11

3 SARX Identification Algorithm **13**

3-1 First Step - Identifying switching time sequence . . . 14

3-1-1 SAT solver . . . 15

3-1-2 Theory solver . . . 16

3-1-3 Algorithm switching time sequence . . . 17

3-1-4 Decrease computation time . . . 17

3-2 Second Step - Identifying model parameters . . . 19

3-2-1 Sparsity based approach for minimising models [1]: An overview . . . 19

3-2-2 Modification to fit intervals . . . 20

3-2-3 Resolving progress obstruction . . . 20

3-2-4 MIMO systems . . . 21

4 PWARX identification Algorithm **23**
4-1 Parameter estimation and classification 23
4-1-1 Testing for linear separability 23
4-2 Partitioning the regressor space 25

5 Dealing with large datasets **29**
5-1 Step 1 - Identification in parts 29
5-2 Step 2 - Minimising number of submodels 30
5-2-1 Solving the minimum hitting set problem using a SAT solver 31
5-3 Step 3 - Final parameter estimation 31

6 Experiments **33**
6-1 Synthetic systems 34
6-1-1 Switched ARX models 34
6-1-2 PWARX examples 36
6-1-3 Frequent switching 37
6-2 Random systems 38
6-3 Experimental data 39
6-3-1 Modeling room temperature 39
6-3-2 Case study 40

7 Discussion and conclusion **43**
7-1 Discussion 43
7-1-1 Properties of HySI-SAT 44
7-2 Conclusion 44
7-3 Future work 45

A Randomly generated systems **47**
A-1 Experiment: Size of dataset 47
A-2 Experiment: Large datasets 47

B Simulation with frequent switching **49**

C HySI-SAT toolbox **51**
C-1 Overview 51
C-2 Instruction 51

D HySI-SAT modified **53**

Bibliography **55**

Glossary **59**
List of Acronyms 59
List of Symbols 59

List of Figures

2-1 Visualisation of a SARX model with 3 modes 6
2-2 2-dimensional example of a regressor space divided into 4 regions 6
2-3 Effect of minimising switches versus minimising submodels 8
2-4 Schematic representation of the lazy SMT framework 12

3-1 Schematic overview of determining a valid switching time sequence 18

4-1 The average and standard deviation of critical parameters 24
4-2 Combining decision boundaries of one-vs-one classifiers can result in a hole. . . . 26
4-3 Fused decision boundary 27

6-1 Mean value of Δ_n for example system 2 with different values of ω_{max}. 34
6-2 Average number of submodels for example system 2 with different values of ω_{max}. 35
6-3 Identified PWA map using HySI-SAT 37
6-4 Identified switching sequence in comparison to true switching sequence 38
6-5 Comparison switching sequence 40
6-6 Schematic overview of the coupled electric drives [2] 41
6-7 The simulated output of the model on the validation datasets 41
6-8 Input coupled electric drives dataset 1 41
6-9 Dataset Section 6.3.1 42

C-1 Graphical user interface of the HySI-SAT program 52

List of Tables

6-1 Δ_n for identification of system 6-7 35
6-2 Estimated parameter vectors for system 6-7 36
6-3 FIT for validation of system 6-10 37
6-4 Identified model parameters system 6-10 38
6-5 Computation time and Δ_n for increasingly large datasets 39
6-6 Trade-off computation time versus number of blocks 39
6-7 Comparison RMSE on the coupled electric drive dataset 42

List of Algorithms

1 Lazy SMT procedure 11
2 HySI-SAT step 1 17
3 Find $\tilde{\boldsymbol{\theta}}$ from [1] 20
5 HySI-SAT (SARX) 20
4 HySI-SAT step 2 21
6 IdentLarge-step1 30
7 IdentLarge-step2 31
8 HySI-SAT modified 53

Preface and Acknowledgements

First, I would like to thank my supervisor dr.ir. M. Mazo for his enthusiastic guidance and for coming up with a thesis topic which was fun and challenging to work on. I am also grateful to everyone that visited the weekly group meetings and provided suggestions and feedback. Their advice was of great help. Thanks is also extended to the members on the examination committee for offering their time. Finally, I wish to thank my family and friends for their support during my whole study.

Delft, University of Technology J.H.M. Zwart
October 23, 2019

Chapter 1

Introduction

Hybrid systems have received a lot of attention in recent years. Their ability to describe systems which exhibit continuous as well as discrete dynamics makes them very powerful. Hybrid systems can be used to model a wide range of processes and systems and various applications of hybrid systems exist in biology, computer vision, the process industry, communication systems and in many more fields. A great deal has already been published about the control, reachability and stability of hybrid systems, but similarly to conventional system, most analysis and applications require a model of the system. In most situations a model can not easily be derived from the laws of physics but instead data driven system identification is required.

There are many different classes of hybrid systems, used to model different processes. Various methods can be found that deal with the identification of a specific type of hybrid system. This thesis will be involved with the identification of two particular types of hybrid systems: Switched AutoRegressive eXogenous (SARX) systems and PieceWise AutoRegressive eXogenous (PWARX) systems. These systems both belong to the class of linear switch systems, and consist of a finite number of affine or linear submodels and a discrete state. This discrete state keeps track of which submodel is active. At all times only one submodel, also known as a mode, is active. SARX models and PWARX models are commonly used for two purposes. The first one is to describe physical systems which exhibit different behaviour for different operating modes. One can think of a car changing gears for instance. The second application is to approximate nonlinear systems. PWARX systems have the ability to approximate all nonlinear system with arbitrary precision [3].

The identification of hybrid systems is not a trivial task. The difficulty is strongly related to the prior information available. In case there is no information on when each submodel was active, the identification problem becomes complicated. Now the classification of the datapoints to a certain submodel is directly correlated with the estimation of the model paramaters. In the literature, different methods that address this problem exist. For a comprehensive overview see the survey paper by Garulli et al. [4].

Vidal et al. [5], [6] tackled this problem by fitting a polynomial of a high dimension to the dataset. This polynomial has a higher dimension than the submodels and captures the information about all the submodels at the same time. This method is often called the algebraic approach, and has sparked a fruitful line of research. One downside of the algebraic method is the sensitivity to noise. The identification problem can also be cast as a mixed-integer program [7]. One drawback this brings forward, is that the computation time scales exponentially with the length of the dataset. Therefore this is only tractable for small datasets. Juloski et al. [8] incorporate probability into the identification process. The parameters describing the PWARX model, the switching sequence and the observations are described by their probability density functions. Methods based on clustering are a popular choice for identifying PWARX models in particular [9], [10]. These methods group datapoints that have a short distance in between together to the same submodel. More recently methods that are based on the notion of sparsity received attention [11], [12], [1], [13], [14], [15].

Ozay et al. [1] proposed a method for identifying switched ARX systems that recasts the identification problem as a maximisation problem in which the sparsity of a sequence is maximised. In their work two different algorithms which attempt to maximise two different criteria are proposed. One tries to minimise the number of submodels needed to fit the data while placing a bound on the error. This method results in simple small SARX models which are computationally attractive for further use. The second method tries to minimise the number of switches between submodels, which is useful in, amongst other things, data segmentation. Observations show that their approach for minimising the number of submodels comes at the cost of a switching sequence with a high number of switches. This can for instance be seen in Figure 2-3. A high switching frequency in a model could be undesired when this does not properly reflect the system being identified. Most switched systems have the tendency to remain in a specific mode for a certain amount of time. Another issue with this methodology is that submodels are identified one by one. This favors datapoints to be attributed to submodels which are identified in the beginning of the algorithm as opposed to later identified submodels. These submodels therefore tend to be overfitted.

In this master thesis a method based on this methodology is proposed. The objective is still minimising the number of submodels in the identified model. However a lower bound on the time between switches is prescribed. In order to enforce this lower bound a SAT solver is used. Sat solvers are a structured and fast instrument to reason on propositional logic, and are becoming faster every year which makes them an attractive tool to include. The involvement of a SAT solver has the added benefit of being able to straightforwardly add certain other constraints on the switches in the identified model.

This thesis is organised as follows. Chapter 2 further introduces the PWARX model and SARX model, and the identification problem is more formally addressed. This chapter also introduces the Boolean satisfiability problem and SAT solvers which will be a core part of the proposed method. The proposed method for the identification of SARX systems and PWARX systems is illustrated in respectively chapter 3 and 4, and extended to the case of large datasets in Chapter 5. The performance of the discussed algorithms is evaluated by simulations and on experimental data in Chapter 6. Finally, in Chapter 7 the results are discussed and the strengths and weaknesses are highlighted. Here a conclusion is also provided and recommendations for future work are given.

1-1 Notation

In this work vectors will be denoted by bold lowercase letters, matrices by capital letters and sets by calligraphic letters. We will write $|\mathcal{A}|$ or $\#\mathcal{A}$ for the cardinality of set $\mathcal{A}$, and $min(\mathcal{A})$ for the smallest element in set $\mathcal{A}$. For scalars $|a|$ will denote the absolute value instead. Both sets and sequences appear in this work. To differentiate between them, sets are build using braces ($\{\cdot\}$), while sequences are denoted by angle brackets ($\langle\cdot\rangle$). $\boldsymbol{e}$ will denote a vector with every entry equal to one. Let $\boldsymbol{x}$ be a vector in $\mathbb{R}^n$. $||\boldsymbol{x}||_p$ denotes the ℓ_p-norm of a vector and is defined as $||\boldsymbol{x}||_p \triangleq \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$, where 0<p<$\infty$. Let $||\boldsymbol{x}||_\infty$ denote the ∞-norm, defined as $||\boldsymbol{x}||_\infty \triangleq \max(\{x_1, ..., x_n\})$, and $||\boldsymbol{x}||_0$ the ℓ_0-norm defined as $||\boldsymbol{x}||_0 \triangleq \#\{i|x_i \neq 0\}$. Likewise $||\langle\boldsymbol{x}(k)\rangle_{k=1}^{N}||_0 \triangleq \#\{\boldsymbol{z} \in \{\boldsymbol{x}(1), .., \boldsymbol{x}(N)\} \mid \boldsymbol{z} \neq \begin{bmatrix}0 & ... & 0\end{bmatrix}^T\}$, which is simply the number of vectors in the sequence $\langle\boldsymbol{x}(k)\rangle_{k=1}^{N}$ which are not equal to the zero vector.

Chapter 2

Preliminaries and problem statement

This chapter further introduces the classes of SARX and PWARX models. First the equations describing these models will be given and then the problem of their identification will be addressed.

2-1 Switched ARX models

A SARX model consists of a collection of ARX models that describe the output as a linear combination of past inputs and outputs together with a noise term:

$$y(k) = \sum_{i=1}^{n_y} a^i_{\sigma(k)} y(k-i) + \sum_{i=1}^{n_u} b^i_{\sigma(k)} u(k-i) + \omega(k) \tag{2-1}$$

where $y(k) \in \mathbb{R}$, $u(k) \in \mathbb{R}$ and $\omega(k) \in \mathbb{R}$ are respectively the output, input and a noise term at time k. A graphical example of a switched ARX model with three modes is represented in Figure 2-1. The discrete switching sequence that maps each point in time to one particular submodel is denoted by $\sigma(k)$, $\sigma : T \to \Sigma$, where Σ is the set of submodels and T is the length of the available data set. The cardinality of Σ, or $|\Sigma|$ denotes the total number of modes present in the SARX model. Lastly, n_y and n_u denote the order of the output and input respectively. We will initially focus on the SISO case. For the extension to MIMO systems see Section 3-2-4. For identification it is more practical to consider the following form of Equation (2-1)

$$y(k) = \boldsymbol{\theta}^T_{\sigma(k)} \boldsymbol{r}(k) + \omega(k) \tag{2-2}$$

where $\boldsymbol{r}(k)$ denotes the regressor which contains the past inputs and outputs.
$\boldsymbol{r}(k) = \begin{bmatrix} y(k-1) & y(k-2) & \dots & y(k-n_y) & u(k-1) & u(k-2) & \dots & u(k-n_u) \end{bmatrix}^T \in \mathbb{R}^{n_y+n_u}$,
and $\boldsymbol{\theta}_{\sigma(k)}$ is the parametervector that encodes the values of $a^i_{\sigma(k)}$ and $b^i_{\sigma(k)}$ at each time step:
$\boldsymbol{\theta}_{\sigma(k)} = \begin{bmatrix} a^1_{\sigma(k)} & \cdots & a^{n_y}_{\sigma(k)} & b^1_{\sigma(k)} & \cdots & b^{n_u}_{\sigma(k)} \end{bmatrix}^T \in \mathbb{R}^{n_y+n_u}$. For SARX models in this report

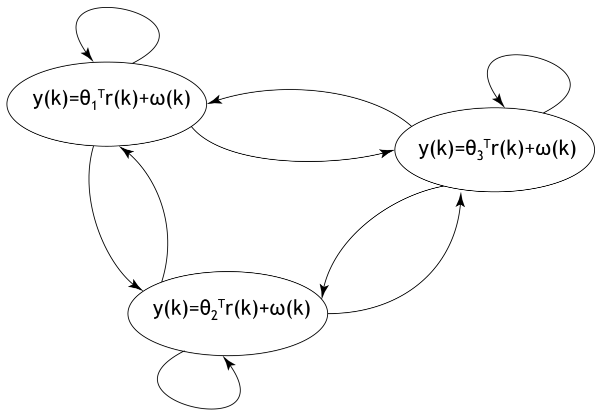


Figure 2-1: Visualisation of a SARX model with 3 modes

the submodels are assumed to be linear. When instead affine submodels are desired, the SARX model needs to be altered slightly by extending the regressor: $\bar{\boldsymbol{r}}\boldsymbol{(k)} = \begin{bmatrix}\boldsymbol{r}(k)^T & 1\end{bmatrix}^T \in \mathbb{R}^{n_y+n_u+1}$. The extended parametervector now also includes one extra term that needs to be identified: $\boldsymbol{\theta}_{\sigma(k)} = \begin{bmatrix}a^1_{\sigma(k)} & \dots & a^{n_y}_{\sigma(k)} & b^1_{\sigma(k)} & \dots & b^{n_u}_{\sigma(k)} & c^1_{\sigma(k)}\end{bmatrix}^T \in \mathbb{R}^{n_y+n_u+1}$

2-2 Piecewise affine ARX models

PWARX models can be considered a subclass of SARX models. The difference lies in the cause of a switch between submodels. In SARX models switches are assumed to be regulated externally, i.e. the cause of a switch is not part of the model. For PWARX models on the other hand, the switching sequence is based on the regressor. The regressor space is partitioned, and the systems current location in this space determines which model is active.

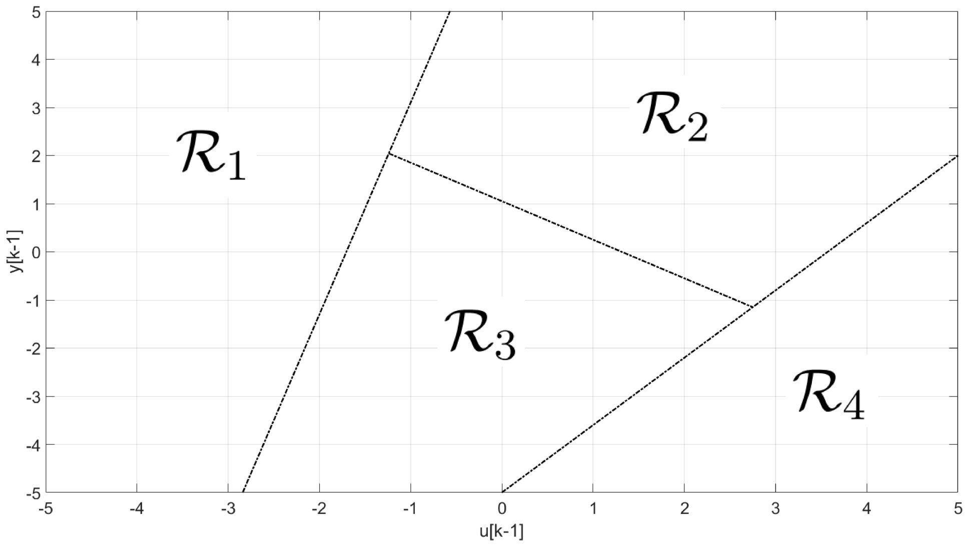


Figure 2-2: 2-dimensional example of a regressor space divided into 4 regions

Each region in this regressor space, denoted by $\mathcal{R}_i$, is a convex polyhedra. These regions are separated from each other by a hyper plane. Let f be a piecewise affine function defined as in Equation 2-4. The output of a PWARX system is now given by:

$$y(k) = f(\bar{\boldsymbol{r}}(k)) + \omega(k) \tag{2-3}$$

where

$$f(\bar{\boldsymbol{r}}(k) = \begin{cases} \boldsymbol{\theta}_1^T \bar{\boldsymbol{r}}(k) & \text{if} \quad \boldsymbol{r}(k) \in \mathcal{R}_1 \\ \boldsymbol{\theta}_2^T \bar{\boldsymbol{r}}(k) & \text{if} \quad \boldsymbol{r}(k) \in \mathcal{R}_2 \\ \vdots \\ \boldsymbol{\theta}_{|\Sigma|}^T \bar{\boldsymbol{r}}(k) & \text{if} \quad \boldsymbol{r}(k) \in \mathcal{R}_{|\Sigma|} \end{cases} \tag{2-4}$$

and $\bar{\boldsymbol{r}}(k) = \begin{bmatrix} \boldsymbol{r}(k)^T & 1 \end{bmatrix}^T \in \mathbb{R}^{n_y+n_u+1}$ is the extended regressor. The discrete state is related to the regressor in the following way:

$$\sigma(k) = i \Leftrightarrow \boldsymbol{r}(k) \in \mathcal{R}_i \tag{2-5}$$

Finally, the regions are given by

$$\mathcal{R}_i = \{\boldsymbol{r} \in \mathbb{R}^{n_y+n_u} \mid H_i \bar{\boldsymbol{r}} \leq 0\} \tag{2-6}$$

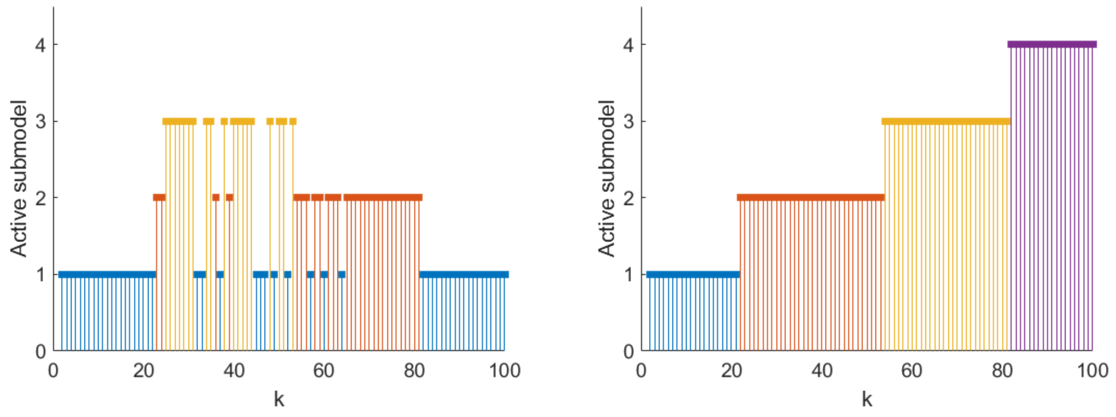
where H_i denotes the coefficientmatrix of the hyperplanes separating region i. An illustrative example of a possible separating of the regressorspace can be seen in Figure 2-2.

2-3 The identification problem

Often the number of submodels is assumed to be known a priori in hybrid system identification techniques. When this is the case, this results naturally in the following SARX/PWARX identification problem (when using a quadratic loss function)

$$\min_{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{|\Sigma|}} \sum_{k=1}^{T} \min_i (y(k) - \boldsymbol{\theta_i}^T \boldsymbol{r}(k))^2, \quad \forall i \in \Sigma, \ \forall k \in T \tag{2-7}$$

Note that this generally represents a non convex optimisation problem. When the number of submodels is not prescribed the problem becomes ill-defined, and care must be taken not to overfit. As reported in [16], the datafit generally goes up when choosing more submodels but this is often not desired as this results in overfitting on a specific dataset and in complex SARX models. This can be avoided by placing a penalty on the number of submodels [15] or by first estimating the number of submodels from data. A different approach to tackle this problem is trying to minimise the number of submodels while placing a bound on the error [1],[16]. This will also be the path followed in this thesis.



(a) Minimising number of submodels

(b) Minimising number of switches

Figure 2-3: Comparison of the switches sequence between two methods from [1]. The same dataset and hyperparameters were used.

2-3-1 Identification of SARX models

To order to make sure that minimising the number of submodels does not result in a turbulent switching sequence a minimum dwell time will be enforced in the form of a constraint added to the SARX identification problem.

Definition 2-3.1. *The minimum dwell time, denoted by τ_d, is the minimum time the system stays in one mode before a switch to another mode can occur. The minimum dwell time constraint enforces that if a mode becomes active, it remains active for at least τ_d timesteps:* ***if*** $\sigma(k) \neq \sigma(k-1)$ ***then*** $\sigma(k) = \sigma(k+1) = ... = \sigma(k+\tau_d-1), \quad \forall k \in T$

With the minimum dwell time in place this leads to the following identification problem for SARX models.

Problem 2-3.1. Given a dataset of input-output data $\langle y(k), u(k)\rangle_{k=1}^{T}$, the order of the model, n_y and n_u, and a specified minimum dwell time τ_d, identify the parameter vectors $\boldsymbol{\theta}_i, 1, ..., s$ and the switching sequence $\sigma(k)$ of the SARX model by minimising the number of submodels. This SARX model should fit every input-output pair within a bound δ, and obey the minimum dwell time τ_d.

2-3-2 Identification of PWARX models

The identification of PWARX models will be treated in a similar way to the identification of SARX models. For the identification of PWARX models the minimum dwell time requirement is swapped to a different constraint. Regressor dependant switching often involves the system being in some regions for a short time while longer in others. To still prevent a switching sequence with a high number of mode switches a less restrictive constraint will be put in place, which restricts the maximum number of switches that are allowed. A limit on the number of switches is equivalent to a certain minimum average dwell time.

Definition 2-3.2. *The switching limit, denoted by L_s, is the maximum number of switches that can appear in the identified switching sequence. The switching limit constraint is given by:*
$\sum_{k=1}^{T} g(\sigma(k)) \leq L_s$
where $g(\sigma(k)) = \begin{cases} 0, & \text{if } \sigma(k-1) - \sigma(k) = 0 \\ 1, & \text{if } \sigma(k-1) - \sigma(k) \neq 0 \end{cases}$

Remark 2-3.1. *The switching limit and minimum dwell time constraint are interchangeable in the proposed identification method and either one can be used for the identification of SARX models and PWARX models.*

The identification problem of PWARX systems is more elaborate. Apart from the parametervectors and the discrete state, the regions of the regressorspace also need to be identified. This leads to the following problem.

Problem 2-3.2. Given a dataset of input-output data $\langle y(k), u(k)\rangle_{k=1}^{T}$, the order of the model, n_y and n_u, and a specified switching limit L_s, identify the parametervectors $\boldsymbol{\theta}_i, i = 1, ..., s$, the switching sequence $\sigma(k)$ and the coefficients of the hyperplanes seperating the regions $H_i, i = 1, ..., s$ of a PWARX model that minimises the number of submodels. This PWARX model should fit every input-output pair within a bound δ, and obey the switching limit described in Definition 2-3.2.

2-3-3 Research goals

As described in the literature report written at the start of this thesis project [17], the goal of this thesis work is to design an identification method for hybrid systems that strives to minimise the number of submodels while placing a limit on the frequency of switching. The goal is therefore to develop two algorithms that offer a solution to Problem 2-3.1 for SARX models and to Problem 2-3.2 for the identification of PWARX models.

The proposed identification method in question is based on the work of Ozay et al. [1], and includes a SAT solver as a core part. The goal is to implement this method in the form of a toolbox written in Python, where the user can input a dataset, and specify certain parameters. The produced toolbox is publicly available [18]. The next section provides background information regarding the SAT problem and SAT solvers.

2-4 Boolean Satisfiability Problem

The Boolean Satisfiablity Problem (SAT) is a well-known decision problem in the field of computer science that has seen a lot of interest in recent years. The problem is to find out whether a Boolean formula is satisfiable, i.e. if there exist a combination of Boolean variables that makes the formula amount to *True*. In addition to this, a combination of variables that achieves this goal is also desired. The Boolean Satisfiablity Problem (SAT) was the first problem that was proven to be NP complete, by [19] and [20] independently.

Although finding a solution for the SAT problem has in the worst case exponential run time, algorithms were developed for it that can handle large problems in a reasonable time. Because the SAT problem is NP complete, all NP problems can be cast to a SAT problem in polynomial time, and afterwards solved by a SAT solver. This makes SAT solvers very powerful. They are widely used in software [21] and hardware verification [22] and in planning problems [23]. The basis of Most SAT solvers is the Davis - Putnam - Logemann - Loveland (DPLL) algorithm [24], [25]. This algorithm guesses variables to be *True* or *False*. When a conflict occurs, it backtracks to a point where it still had the possibility of being satisfiable. SAT solvers often require a special form of the Boolean formula called Conjunctive Normal Form (CNF). The CNF-SAT problem (commonly just referred to as SAT problem), consists of the following elements.

— *atoms*: can either be a constant or a Boolean variable. Booleans variables can only have two values: *True* (1) or *False* (0): b_i.

— *Literals*, are atoms or their negation: b_i, $\neg b_i$.

— *Clauses*, consist of a combination of Literals and the logical OR ($\vee$) operator: $\neg b_i \vee b_j$

— *CNF formula*, which consists of a combination of clauses and the logical AND operator ($\wedge$): $C_i \wedge C_j$

An assignment or valuation of a CNF formula, denoted by μ, is obtained when true or false are assigned to each variable. Example 2-4.1 gives an example of a SAT problem.

Example 2-4.1. $(b_1 \vee b_2 \vee \neg b_3) \quad \wedge \quad (b_2 \vee b_3) \quad \wedge \quad (\neg b_1 \vee b_2)$
A Satisfiable assignment is given by $b_1 = $ *False*, $b_2 = $ *True*, $b_3 = $ *False*, therefore this problem is *satisfiable*.

2-4-1 Cardinality constraints

Cardinality constraints on Boolean variables are encountered often when translating a problem into a SAT problem. These constraints of the form

$$\sum_{i=1}^{n} x_i \leq k \tag{2-8}$$

have been dealt with in multiple ways. For an overview see [26] and references therein. Most encodings introduce new auxiliary Boolean variables. The encodings differ in the quantity of required extra clauses and required extra variables. In this work the sequential encoding [27] is used. It requires $\mathcal{O}(n \cdot k)$ new variables and $\mathcal{O}(n \cdot k)$ new clauses.

2-4-2 SMT

One major drawback of SAT solvers is that many real world problems are not easily encoded to a SAT problem. Most real world problems involve variables that can have a wide range of values and are not limited to *True* or *False*. An extension to the SAT problem that overcomes this drawback is called SMT. The SMT problem is involved with deciding whether a first order formula with respect to a specific background theory is satisfiable. The Atoms in SMT are not limited anymore to Boolean variables but can be any expression that can amount to *True* or *False* for a specific theory. This is illustrated in Example 2-4.2, which depicts a SMT formula using linear arithmetic as a theory.

Example 2-4.2. $((x \leq y) \vee x > 0) \wedge (y < 2) \wedge (x < 1)$

Notice that each atom, e.g. x>0, still can either amount to *True* or *False* depending on the value of the variables. Example 2-4.2 can be abstracted to the SAT formula $(b_1 \vee b_2) \wedge (b_3) \wedge (b_4)$, using the map: $\{\ x \leq y \leftrightarrow b_1, x > 0 \leftrightarrow b_2, y < 2 \leftrightarrow b_3, x < 1 \leftrightarrow b_4\}$.

2-4-3 Lazy SMT

Two main approach to solving SMT formulas exist. The approach that has gained the most traction is referred to as lazy SMT. In lazy SMT two solvers are used, A SAT solver and a theory solver. the SAT solver will handle the Boolean reasoning and the theory solver handles the theory specific part. First the SMT formula φ_{SMT} is abstracted to a SAT formula φ_b using

Algorithm 1 Lazy SMT procedure

```
 1: procedure DETERMINE SATISFIABILITY SMT-FORMULA
 2:     $\varphi_b \leftarrow$ Boolean abstraction($\varphi_{SMT}$)
 3:     while True do
 4:         (satisfiability, $\mu$) $\leftarrow$ SAT solver($\varphi_b$)
 5:         if satisfiability==UNSAT then
 6:             return satisfiability
 7:         end if
 8:         $\mathcal{M} = Map(\mu)$
 9:         $(\mathcal{T}\text{-}satisfiability, \varphi_{cert}) \leftarrow$ Theory solver($\mathcal{M}$)
10:         if $\mathcal{T}$-satisfiabilility==$\mathcal{T}$-$SAT$ then
11:             return ($\mathcal{T}$-satisfiability, $\mathcal{M}$)
12:         end if
13:         $\varphi_b \leftarrow \varphi_b \wedge \varphi_{cert}$
14:     end while
15: end procedure
```

a dictionary. This propositional formula is checked for satisfiability by a SAT-solver which proposes a valuation of Booleans μ. This valuation is mapped according to the dictionary to a set of constraints $\mathcal{M}$. Now the theory solver is invoked to check if this set of constraints is feasible with respect to a certain theory. If there is a solution consistent with all constraints, the set of constraints is called $\mathcal{T}$-satisfiable. If this is not the case, the theory solver will generate a conflict clause called a certificate. This certificate captures information regarding the conflict and limits the search space for the SAT solver. This certificate clause is added to the SAT solver, and the cycle continues. The collaboration of a SAT solver with a theory specific solver has received much interest in recent years. Applications range from robotic motion planning by Shoukry et al. [28] to secure state estimation [29]. In the next chapter an identification method for switched linear systems is proposed along the lines of lazy SMT.

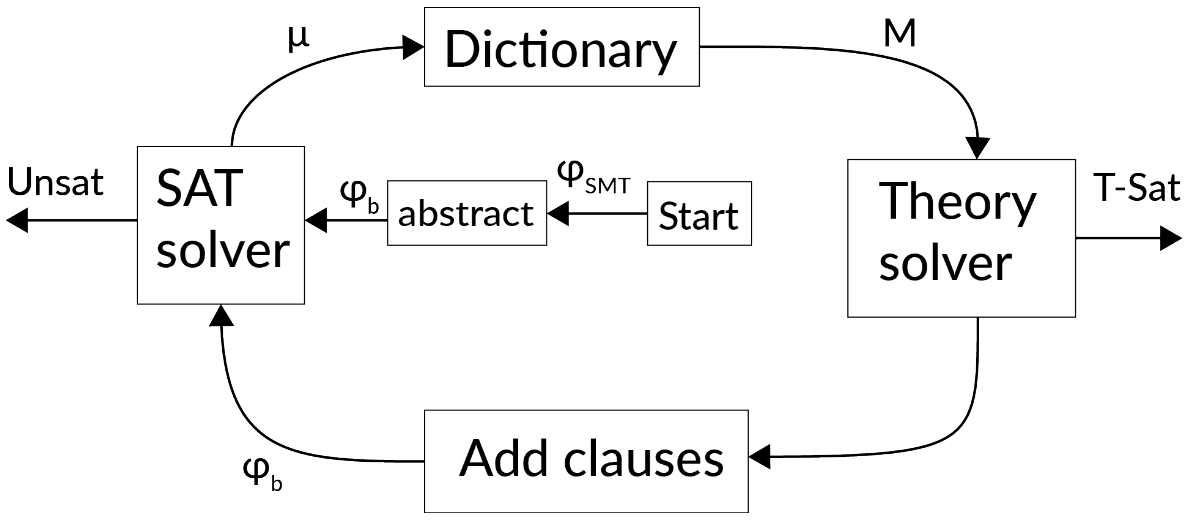


Figure 2-4: Schematic representation of the lazy SMT framework

Chapter 3

SARX Identification Algorithm

This chapter describes the proposed identification algorithm aimed at identifying SARX models. The identification method consists of two steps. In the first step a feasible sequence of switching times will be determined. To achieve this a SAT solver will be iteratively used in cooperation with linear programming.

In the second step the parameter vectors of the sub models will be determined and attributed to intervals between switches. The parameter vectors will be estimated by posing the identification problem as a problem of maximising the sparsity of a specific sequence. Together these two steps form the proposed algorithm called HySI-SAT (derived from "Hybrid System Identification" and "SAT solver").

3-1 First Step - Identifying switching time sequence

The goal of the first step is to determine a feasible switching time sequence.

Definition 3-1.1. *The switching time sequence S is defined as a sequence containing all the time instances in T where $\sigma(k) \neq \sigma(k-1)$, ordered from small to large.*

$$S = \langle t_s^1,\ t_s^2,\ ...,\ t_s^m \rangle \tag{3-1}$$

where t_s^i is the time instance of switch i and m is the total number of switches.

The first step along the lines of lazy SMT, lets a SAT solver interact with a theory solver. The SAT solver will propose a switching time sequence every iteration, which is satisfiable with respect to the Boolean constraints. The theory solver will decide if this switching time sequence is viable and if not, steer the SAT solver in the right direction.

This procedure starts at the SAT solver. The SAT solver proposes a switching time sequence that does not conflict with the minimum dwell time constraint (see Definition 2-3.1). Then the dataset is split up into parts according to the switching time sequence. These parts of the dataset are then individually checked by the theory solver for feasibility. The theory solver uses linear programming, to see if for all parts of the dataset there exist an ARX model that fits. If any of the parts of the dataset are deemed infeasible by the theory solver, we go back to the SAT solver to ask for a different switching time sequence. This is done in the form of adding a constraint clause to not allow this specific switching time sequence. This interplay between the SAT solver and linear programming is repeated and every iteration another constrained is added.

Termination

There are two ways the procedure can terminate. The first is that the Linear Programming approves all the parts of the dataset and thereby validates the last proposed switching time sequence. This is the desired result. In this case the method can continue with step 2. It is however also possible that after a while the SAT solver returns *UNSAT*, which indicates that there does not exist a switching time sequence that satisfies the constraints. This implies that the conditions are to strict and there does not exist a SARX model that fits the data within the prescribed bound while upholding the dwell time constraint. The program terminates and either the minimum dwell time needs to be decreased or the bound on the allowable error increased.

In the coming sections the components will be one by one addressed in more detail starting with a closer look at the SAT solver.

3-1-1 SAT solver

In the first step the SAT solver will be used to propose a sequence of switching times that is valid with respect to the constraints. To add the constraints to the SAT solver a set of Booleans is required. Therefore a set of Boolean variables $\{b(k)\}_{k=1}^{T}, b(k) \in \mathbb{B}$ is introduced which denote whether a switch occurs at time k, i.e. $b_i(k) \Rightarrow (\sigma(k) \neq \sigma(k-1))$ and $\neg b_i(k) \Rightarrow (\sigma(k) = \sigma(k-1))$.

Adding constraints

To add the dwell time constraint to the SAT solver it first must be posed as a Boolean formula. Constraints are added in the form of clauses. The dwell time constraint enforces a minimum time between switches. It decodes that if a switch is proposed at time k, then for the remainder of the dwell time no other switch can be proposed and therefore the corresponding Booleans should be set to *False*. This can be cast as a Boolean formula:

$$\varphi_{dwell} = b(k) \Rightarrow (\neg b(k+1) \wedge ... \wedge \neg b(k+\tau_d - 1)), \qquad \forall k \in T \tag{3-2}$$

and yields the following constraint in conjuntive normal form

$$\varphi_{dwell} = \big(\neg b(k) \vee \neg b(k+1)\big) \wedge ... \wedge \big(\neg b(k) \vee \neg b(k+\tau_d - 1)\big), \forall k \in T \tag{3-3}$$

The CNF formula that is fed to the SAT solver at the start, only consists of the dwell time constraint.

$$\varphi = \varphi_{dwell} \tag{3-4}$$

As mentioned previously, this constraint can be replaced by a switching limit constraint. In this case the Boolean formula is given by

$$\varphi_{switchinglimit} = \sum_{k=1}^{T} b(k) \leq L_s \tag{3-5}$$

3-1-2 Theory solver

After the SAT solver proposed a valuation μ, the switching time sequence is reconstructed.

$$S = \langle s_k | b(k) = 1 \rangle_{k=1}^{T} \tag{3-6}$$

According to the switching time sequence (S) the dataset will be split at each $t_k \in S$. Let interval i, denoted by $\mathcal{I}_i$, be the set of time instances between switch i and the preceding switch. The time instance associated with the preceding switch is also included in interval $\mathcal{I}_i$, see Equation 3-7. In case of the first interval the start is not marked by the preceding switch but rather by the beginning of the dataset. All the input-output pairs are now attributed to exactly one interval.

$$\begin{aligned} \mathcal{I}_1 &= \{k | k < t_s^1\}, \quad \forall k \in T \\ \mathcal{I}_2 &= \{k | t_s^1 \le k < t_s^2\}, \quad \forall k \in T \\ \vdots &= \quad \vdots \\ \mathcal{I}_{m+1} &= \{k | k \ge t_s^m\}, \quad \forall k \in T \end{aligned} \tag{3-7}$$

For each interval i, the following feasibility problem is solved by linear programming.

$$\begin{aligned} &\min_{\boldsymbol{\theta}} 1 \\ &s.t. \\ &|y(k) - \boldsymbol{\theta}^T \boldsymbol{r}(k)| \le \delta \\ &\forall k \in \mathcal{I}_i \end{aligned} \tag{3-8}$$

Certificate generation

If for any of the intervals, Problem (3-8) gives no solution then the switching time sequence does not meet our requirements. The following simple constraint certificate could therefore be created in the form of a Boolean formula to restrict this specific switching time sequence

$$\varphi_{basic} = \neg\mu \tag{3-9}$$

This yields the new formula fed to the SAT solver:

$$\varphi = \varphi \wedge \varphi_{basic} \tag{3-10}$$

By restricting just the proposed switching time sequence, the search space is only very slightly decreased every iteration. Therefore the algorithm generally takes a long time to find a valid switching time sequence. The process can be sped up significantly by using more of the information that is obtained by the linear programming. Any interval that is infeasible, signifies that the input-output pairs in that interval do not all belong to the same mode. By this we can infer that a switch is necessary somewhere in that interval. Lets $\mathcal{I}^*$ denote an interval where the smallest time instance k is removed. For every infeasible interval we can now create a conflict clause :

$$\varphi_{inf}^{j} = \bigvee_{i \in \mathcal{I}_j^*} b_i \tag{3-11}$$

This constraint clause enforces that one of the Booleans in that interval should be true and therefore enforces a switch. The conjunction of these conflict clauses leads to the following infeasible interval certificate

$$\varphi_{inf} = \varphi_1^{inf} \wedge ... \wedge \varphi_t^{inf} \tag{3-12}$$

where t is the number of infeasible intervals. Equation (3-10) now changes to

$$\varphi = \varphi \wedge \varphi_{inf} \tag{3-13}$$

As noted by Shoukry et al. [28], a shorter conflict certificate removes more of the search space each iteration. This generally leads to a faster solution when the computation time to generate the certificate is small in contrast to the rest of the algorithm.

Remark 3-1.1. *Both the infeasible interval certificate and the basic certificate only exclude false candidate solutions from the search space.*

3-1-3 Algorithm switching time sequence

The first step is summarised in Algorithm 2. A schematic overview is shown in Figure 3-1.

Algorithm 2 HySI-SAT step 1

```
1: procedure DETERMINE SWITCHING TIME SEQUENCE
2:     $\varphi \leftarrow \varphi_{dwell}$
3:     while True do
4:         (satisfiability, $\mu$) ← SAT solver($\varphi$)
5:         if satisfiability==$UNSAT$ then
6:             return satisfiability
7:         end if
8:         $S \leftarrow Equation_{3-6}(\mu)$
9:         ($\mathcal{T}$-satisfiability, $\varphi_{inf}$) ← Theory solver($S$)
10:        if $\mathcal{T}$-satisfiabilility==$\mathcal{T}$-$SAT$ then
11:            return ($\mathcal{T}$-satisfiability, S)
12:        end if
13:        $\varphi \leftarrow \varphi \wedge \varphi_{inf}$
14:    end while
15: end procedure
```

3-1-4 Decrease computation time

The following two features can be added to the algorithm to decrease the computation time. They are however not core to the method. These features are included in the HySI-SAT toolbox.

Remove redundant certificates

After a while some certificates may not provide extra information. A certificates φ_{inf} can be present that enforces a switch in a time frame that is completely covered by the second certificate. The second certificate has no added value in this situation and should be removed. More formally, a certificate i is redundant with respect to certificate j if the following holds:

$$\varphi_j \implies \varphi_i \tag{3-14}$$

In the first step of HySI-SAT, all infeasible interval certificates are checked for redundancy each cycle.

Dictionary of feasible intervals

It often occurs that the SAT solver will propose a switching time sequence that result in some intervals that were already checked for feasibility in a previous cycle. Checking the feasibility again is unnecessary. Therefore the theory solver will keep a list of all intervals that were feasible in previous cycles. Whenever the theory solver encounters an interval that is in this list it will assume it is feasible.

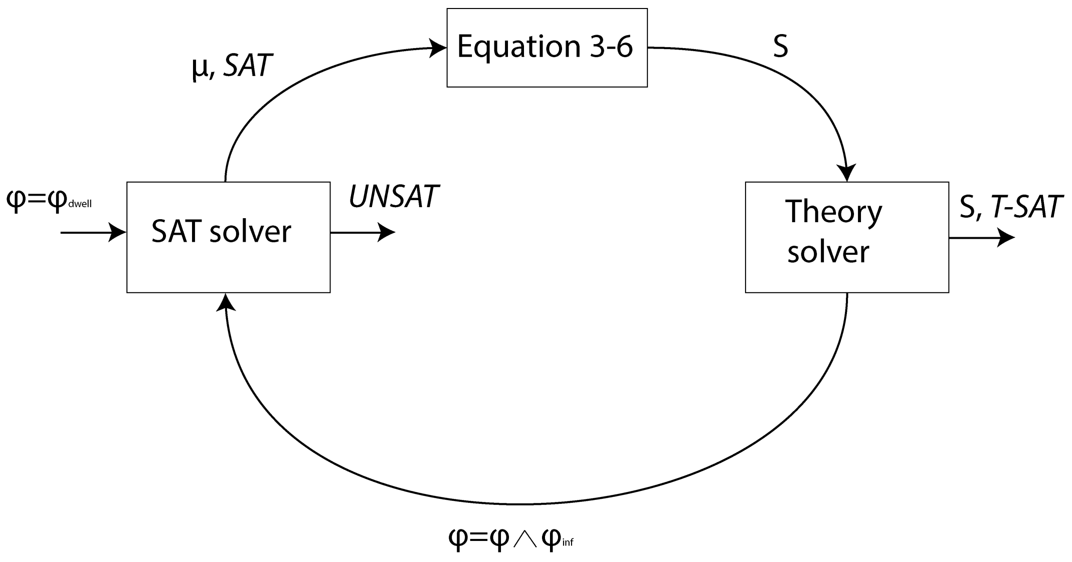


Figure 3-1: Schematic overview of determining a valid switching time sequence

3-2 Second Step - Identifying model parameters

This step can be seen as a modification to the sparsity based algorithm in [1] where the number of submodels needed to explain a dataset is minimised. This algorithm is first briefly summarised, after which the modification is highlighted. For the details of their approach or a more in depth explanation, the interested reader is direct to their work [1].

3-2-1 Sparsity based approach for minimising models [1]: An overview

The goal is to explain the data with a minimum number of submodels. [16] posed this problem as dividing an infeasible system of linear equations into a minimum number of feasible linear subsystems. This is known as the MIN PFS problem [30]. This problem is NP-hard. Therefore this is often relaxed to the MAX FS problem, which is the problem of finding the largest possible feasible subsystem. Solving this problem is also NP-hard [31]. In our case the MAX FS problem is related to finding a parametervector ($\tilde{\boldsymbol{\theta}}$) that fits as many datapoints as possible. $\tilde{\boldsymbol{\theta}}$ fits a time instance k if $||\tilde{\boldsymbol{\theta}}^T \boldsymbol{r}(k) - y(k)||_\infty \leq \delta$. Ozay et al. cast the problem of finding $\tilde{\boldsymbol{\theta}}$ as maximising the sparsity of the sequence $\langle \boldsymbol{\theta}(k) - \tilde{\boldsymbol{\theta}} \rangle_{k=1}^{T}$. Maximising the sparsity of a sequence is equivalent to minimising its ℓ_0-norm[1]. This leads to Equation (3-15) for finding a parameter vector that fits as many time instances as possible in a set $\mathcal{N}$.

$$\begin{aligned} \min_{\boldsymbol{\theta}(k),\tilde{\boldsymbol{\theta}}} \quad & ||\langle \boldsymbol{\theta}(k) - \tilde{\boldsymbol{\theta}} \rangle||_0 \\ s.t. \quad & ||\tilde{\boldsymbol{\theta}}^T \boldsymbol{r}(k) - y(k)||_\infty \leq \delta, \qquad \forall k \in \mathcal{N} \end{aligned} \tag{3-15}$$

Minimising the ℓ_0-norm is also generally NP-hard. Commonly, ℓ_0-optimisation problems are relaxed to convex ℓ_1-optimisation problems. Candès and Tao [32] introduced certain conditions which ensure that solving the ℓ_1-minimisation results in the same solution to the original ℓ_0-minimisation problem. Ozay et al. [1] follow the approach from [33], where instead of minimising the ℓ_0-norm, a sequence of weighted ℓ_1-minimisation problems is solved. In practise this approach shows good results even if the conditions from [32] do not hold.

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{\theta}(k),\tilde{\boldsymbol{\theta}}} \quad & \boldsymbol{w}^T \boldsymbol{z} \\ s.t. \quad & ||\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}(k)||_\infty \leq z_k \\ & ||y(k) - \boldsymbol{r}(k)^T \boldsymbol{\theta}(k)||_\infty \leq \delta \\ & \forall k \in \mathcal{N} \end{aligned} \tag{3-16}$$

This is summarised in Algorithm 3. ϵ denotes a regularisation constant and η is the number of iterations. The reweighted ℓ_1-minimisation is fairly robust to the choise of ϵ, according to Candès and Wakin [33].

[1]The ℓ_0-norm is not a real norm because the homogeneity property is violated. $||c\boldsymbol{x}||_0$ is generally not equal to $|c| \cdot ||\boldsymbol{x}||_0$

Algorithm 3 Find $\tilde{\boldsymbol{\theta}}$ from [1]

```
1: procedure Weighted $\ell_1$ relaxation from [33]
2:     $\boldsymbol{w} \leftarrow \boldsymbol{e}$
3:     for i=1:$\eta$ do
4:         $(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}) \leftarrow Equation_{3-16}(\mathcal{N}, \boldsymbol{w})$
5:         $w_k \leftarrow \frac{1}{z_k+\epsilon}, \quad \forall k \in \mathcal{N}$
6:     end for
7:     return $\tilde{\boldsymbol{\theta}}$
8: end procedure
```

After $\tilde{\boldsymbol{\theta}}$ is found, the time instances that fit this parameter vector are removed from the search space. For the remaining time instances Algorithm 3 is run again, removing fitting datapoints at each iteration. This continues until all time instances are attributed to a submodel.

3-2-2 Modification to fit intervals

In step 1 a valid switching time sequence S was determined. The intervals can easily be inferred from S. In contrast to the method proposed in [1], our goal is to find a parameter vector that fits as many intervals as possible opposed to as many time instances as possible. To achieve this Equation 3-16 is modified which results in Equation 3-17.

$$\begin{aligned}
\min_{\boldsymbol{z},\boldsymbol{\theta}(j),\tilde{\boldsymbol{\theta}}} \quad & \boldsymbol{w}^T \boldsymbol{z} \\
s.t. \quad & ||\boldsymbol{\theta}(j) - \tilde{\boldsymbol{\theta}}||_\infty \le z_j \\
& ||y(k) - \boldsymbol{r}(k)^T \boldsymbol{\theta}(j)||_\infty \le \delta, \qquad \forall k \in \mathcal{I}_j \\
& \forall j \in \{1, ..., |S|+1\}
\end{aligned} \tag{3-17}$$

Algorithm 4 describes this procedure. Algorithm 4 returns the parameters of the model together with a collection of sets of time instances belonging to each submodel. From this the switching sequence can straightforwardly be recovered, which gives the final SARX model.

These two steps form HYSI-SAT and are summarised in Algorithm 5.

Algorithm 5 HySI-SAT (SARX)

```
1: procedure SARX identification
2:     $(S, \mathcal{T}\text{-satisfiability}) \leftarrow HySI\text{-}SAT\ step\ 1(\langle y(k), u(k)\rangle_{k=1}^T)$    (see Algorithm 2)
3:     $(\mathcal{K}_i, ..., \mathcal{K}_s, \theta_1, ..., \theta_s) \leftarrow HySI\text{-}SAT\ step\ 2(S)$    (see Algorithm 4)
4:     $\sigma(k) \leftarrow$ Recover switching sequence$(\mathcal{K}_i, i = 1, ..., s)$
5:     $\Sigma \leftarrow \{\theta_1, ..., \theta_s\}$
6:     return $(\Sigma, \quad \sigma(k))$
7: end procedure
```

3-2-3 Resolving progress obstruction

Notice that in Equation 3-17 only one constraint is placed on $\tilde{\boldsymbol{\theta}}$: $||\boldsymbol{\theta}(j) - \tilde{\boldsymbol{\theta}}||_\infty \le z_j$. $\tilde{\boldsymbol{\theta}}$ is here only bounded by z_j, which is unbounded itself. Therefore when the weighted ℓ_1-relaxation

Algorithm 4 HySI-SAT step 2

```
1:  procedure FIND SUBMODELS
2:      $l \leftarrow 0$
3:      $\mathcal{N}_1 \leftarrow \{1, ..., |\mathcal{S}| + 1\}$        ▷ Initialise a set containing the indices of all intervals
4:      while $|\mathcal{N}_{l+1}| > 0$ do        ▷ Continue till all intervals are accounted for
5:          $l \leftarrow l + 1$
6:          $\boldsymbol{w} \leftarrow \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}^T$
7:          for i=1:$\eta$ do        ▷ Reweighted $\ell$1-minimisation
8:              $(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}) \leftarrow Equation_{3-17}(\mathcal{N}_l, \boldsymbol{w})$
9:              $w_j \leftarrow \frac{1}{z_j+\epsilon}, \quad \forall j \in \mathcal{N}_l$
10:         end for
11:         $\boldsymbol{\theta_l} \leftarrow \tilde{\boldsymbol{\theta}}$
12:         $K_l \leftarrow \{j \in \mathcal{N}_l| \quad ||y(k) - \boldsymbol{r}(k)^T \boldsymbol{\theta_l}||_\infty \leq \delta, \quad \forall k \in \mathcal{I}_j\}$
13:         $\mathcal{N}_{l+1} \leftarrow \mathcal{N}_l \setminus \mathcal{K}_l$        ▷ Create new set with indices from unacounted intervals
14:     end while
15:     return $(\mathcal{K}_i, i = 1, ..., l, \boldsymbol{\theta}_i, i = 1, ..., l)$
16: end procedure
```

does not converge to the same value as the ℓ_0-minimisation, it can occur that the found $\tilde{\boldsymbol{\theta}}$ does not fit any interval. When this is the case no intervals are removed from the set of unaccounted intervals this iteration. The next time the weighted ℓ_1-relaxation is solved the minimisation problem is identical to the problem last time which will again result in the same $\tilde{\boldsymbol{\theta}}$ that does not fit any interval. Progress is halted and the algorithm can not continue. This problem arises because the used reweighted ℓ_1-minimisation is completely deterministic. Experience shows that this problem does not occur all that often. Nonetheless we propose two solutions to this problem. The first solution removes the deterministic nature by initialising the problem with random weights. Line 6 of Algorithm 4 is replaced by

$$w_j \sim \mathcal{U}(0, 1), \forall j \in \mathcal{N} \tag{3-18}$$

The second course of action one can take for resolving the issue is by marking the q intervals that have the lowest error with the troublesome $\tilde{\boldsymbol{\theta}}$. The next loop, only the marked intervals are considered in the reweighted ℓ_1-minimisation. If now the obstruction still remains q can be decreased till a fitting $\tilde{\boldsymbol{\theta}}$ is found and the algorithm can proceed normally.

3-2-4 MIMO systems

The proposed identification method can also identify MIMO SARX models of the form

$$\boldsymbol{y}(k) = \sum_{i=1}^{n_y} A^i_{\sigma(k)} \boldsymbol{y}(k-i) + \sum_{j=1}^{n_u} B^j_{\sigma(k)} \boldsymbol{u}(k-j) + \boldsymbol{\omega}(k) \tag{3-19}$$

where the output, input and noise term are now given by $\boldsymbol{y}(k) \in \mathbb{R}^n$, $\boldsymbol{u}(k) \in \mathbb{R}^m$ and $\boldsymbol{\omega}(k) \in \mathbb{R}^n$ respectively. $A^i_{\sigma(k)}$ and $B^i_{\sigma(k)}$ are parameter matrices of appropriate dimension. Stacking the outputs together into one vector yields the following regressor used for identifying

MIMO models.

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{y}(k-1)^T & ... & \boldsymbol{y}(k-n_y)^T & \boldsymbol{u}(k-1)^T & ... & \boldsymbol{u}(k-n_u)^T \end{bmatrix}^T \tag{3-20}$$

Chapter 4

PWARX identification Algorithm

Mode switching for a PWARX system is based on the regressor. This gives us extra information that can aid us during identification. In this chapter HySY-SAT will be extended to PWARX models, where we will use this extra information in the process. Similarly to many existing PWARX identification methods, e.g. [8], [16], [34], first the parameter vectors are estimated and the data points classified, and afterwards the regressor space is partitioned based on this classification.

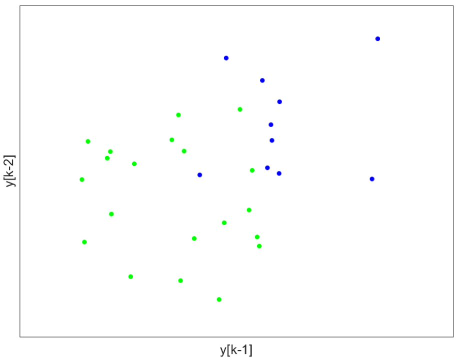
4-1 Parameter estimation and classification

The regressor space for a PWARX model is partitioned into disjoint convex regions. Therefore we can assume that in a noiseless setting, groups of datapoints belonging to different models are linearly separable. This is used as an extra check to determine if a switching time sequence is feasible. At this point the models are not known, but a switch signifies that the interval before the switch will belongs to a different model than the interval after the switch. This property of linear separability can be checked for all couples of adjacent intervals.

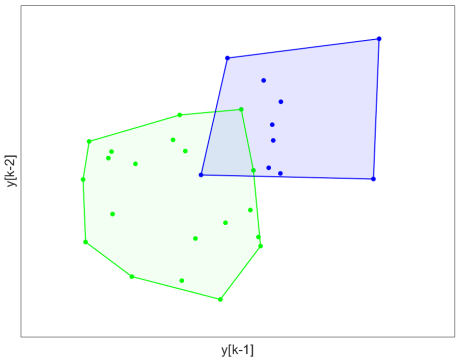
4-1-1 Testing for linear separability

In order to test the linear separability of two intervals, first the convex hull of both groups of datapoints is calculated. The convex hull of a collection of points is defined as the smallest convex set containing all those points. The convex hull of a group of points forms a polyhedron. An example is shown in Figure 4-1b. We take noise into consideration by assuming that datapoints fall at most a distance δ outside of their true region. To incorporate this, both convex hulls are eroded to a smaller set. This is done by taking their Minkowski difference. Various definitions of the Minkowski difference are in use. We will use the following definition [35]:

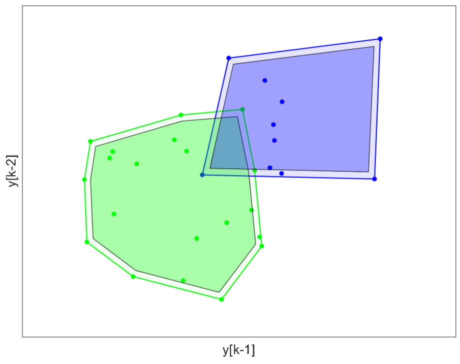
Definition 4-1.1. *The Minkowski difference of two sets of points in* $\mathbb{R}^n$, *denoted by* $\mathcal{A}_1$ *and* $\mathcal{A}_2$, *is defined as* $\mathcal{A}_1 \ominus \mathcal{A}_2 = \{\boldsymbol{x} \in \mathbb{R}^n | \mathcal{A}_2 + \boldsymbol{x} \subseteq \mathcal{A}_1\}$



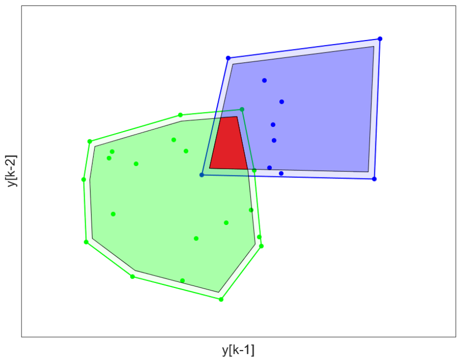
(a) Two groups of points



(b) convex hull



(c) eroded sets



(d) Checking for intersection

Figure 4-1: A 2-dimensional example of the test conducted on two adjacent intervals. In case the intersection is not empty (red area), a li-certificate is created and added to the SAT solver.

Let $\mathcal{B}$ be a closed ball centered around the origin with radius δ, and $\mathcal{P}$ the convex hull of the points in one interval. The eroded set is now given by

$$\tilde{\mathcal{P}} = \mathcal{P} \ominus \mathcal{B} \tag{4-1}$$

The two eroded sets of points are now checked for an intersection. When the intersection is not empty a *linearly inseparable* (li)-certificate is created that enforces one of the two intervals to include a switch, or remove the switch that separates them. Next, this certificate is added to the SAT solver. Let $\mathcal{I}_1$ denote the first of two adjacent intervals and $\mathcal{I}_2$ the interval which occurs later in time. The li-certificate is defined as

$$\varphi_{li} = \left(\bigvee_{i \in (\mathcal{I}_1^\star \cup \mathcal{I}_2^\star)} b_i \right) \vee \neg b_{min(\mathcal{I}_2)} \tag{4-2}$$

Recall that the asterisk sign denotes the exclusion of the smallest element in an interval. This test for linear separability is included in every iteration of the first step of the PWARX identification algorithm, and is run on every couple of adjacent intervals. Step 2 of HySI-SAT is identical for SARX models and PWARX models.

4-2 Partitioning the regressor space

Finally, the regressor space is partitioned. This resembles a common problem in the field of pattern recognition, and numerous linear classifiers exist that can be used. A popular solution is to use a Support Vector Machine (SVM) [36], for instance used for PWARX identification in [10], [9]. SVMs are very fast but are binary classifiers which only take two classes into consideration at the same time. One method of extending SVM to multi class problems is by classifying all possibles couples of 2 classes, resulting in $\frac{K \cdot (K-1)}{2}$ binary One-versus-one (OVO) [37] classifiers. One drawback however, as mentioned by [9], is that this can leave holes in the regressor space which are not classified to any class, see Figure 4-2. One other approach of extending SVM to a multiclass problem is by training K One-versus-rest (OVR) classifiers. A OVR classifier sets one class against a combination of all the other classes combined into one "rest" class. Straightforwardly combining these can still result in holes and some regions which are ambiguous as they are classified to more than one class, as is the case in Figure 4-3a. The problem of holes can be resolved by classifying a datapoint in this region to the class closest to this datapoint. In the same way, we classify datapoints that are attributed to more than one class, to the class where the datapoint is furthest from the decision boundary. This approach was first proposed in [38]. The decision boundaries are given by the equation $\boldsymbol{w}_i^T \boldsymbol{x} = 0$, where $\boldsymbol{w}_i$ denotes a vector of weights and $\boldsymbol{x}$ denotes a datapoint. $\boldsymbol{x}$ is now classified to a class according to the following equation

$$class = \underset{i}{\operatorname{argmax}}\, \boldsymbol{w}_i^T \boldsymbol{x} \tag{4-3}$$

The hyperplanes that form the new decision boundary between class i and class j are therefore given by the equation

$$(\boldsymbol{w}_i - \boldsymbol{w}_j)^T \boldsymbol{x} = 0 \tag{4-4}$$

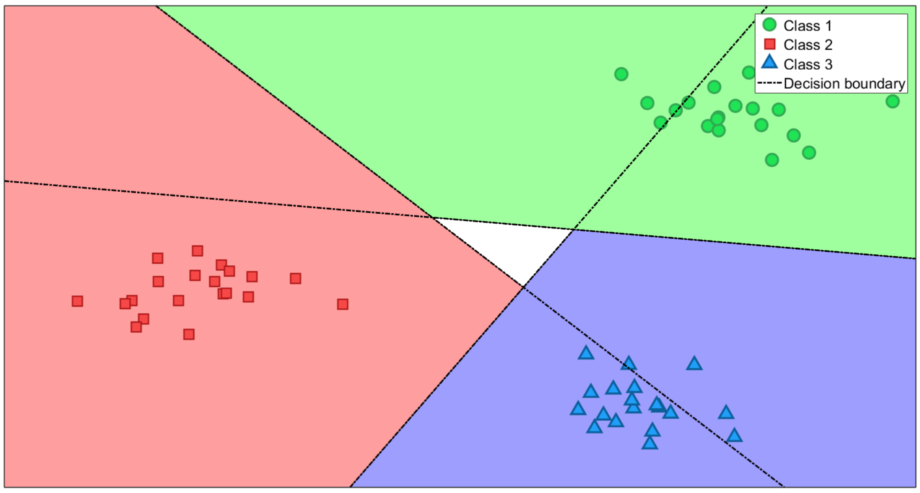
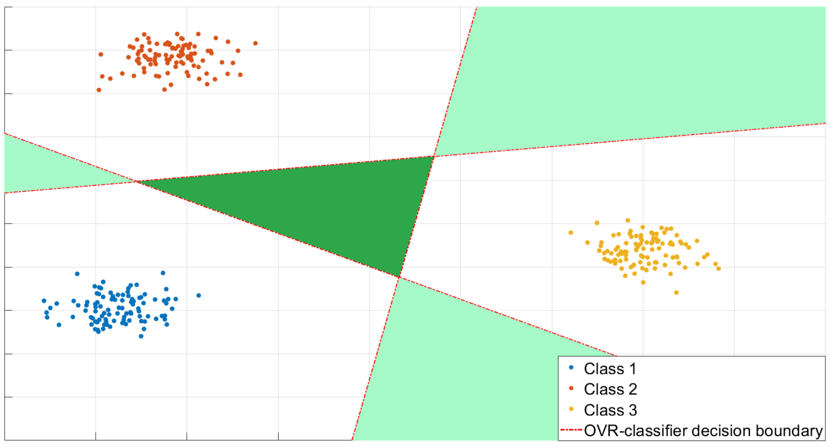


Figure 4-2: Combining decision boundaries of one-vs-one classifiers can result in a hole.

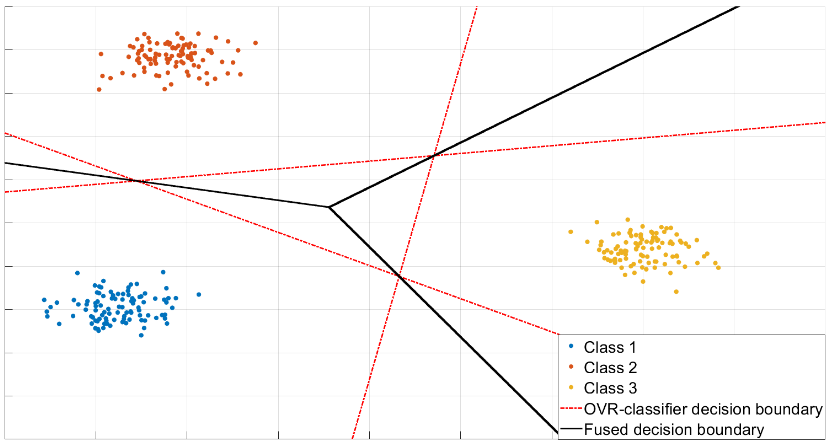
The coefficients H_i can finally be recovered by stacking the corresponding coefficients from equation 4-4 which yields

$$H_i = \begin{bmatrix} \boldsymbol{w}_i^T - \boldsymbol{w}_1^T \\ \boldsymbol{w}_i^T - \boldsymbol{w}_2^T \\ \vdots \\ \boldsymbol{w}_i^T - \boldsymbol{w}_K^T \end{bmatrix} \tag{4-5}$$

These coefficientmatrices describe the final partitioning according to Equation 2-6.



(a)



(b)

Figure 4-3: (a) Linear decision boundaries generated by One-versus-rest classifiers. Light green marks regions that belong in two classes, while dark green marks a region not attributed to any class. **(b)** Decision boundaries merged with the fusion rule.

Chapter 5

Dealing with large datasets

The proposed identification method can become slow for large datasets. This section proposes a way to deal with large datasets. The idea is to split the dataset into smaller parts to make the problem more tractable. This comes at the cost of obtaining a SARX/PWARX model with potentially more submodels. The method consists of three steps.

5-1 Step 1 - Identification in parts

First the dataset is split into m (almost) equal parts. Let the set containing all time instances for one specific part be called a block denoted by $\mathcal{B}_i, i = 1, ..., m$:

$$\mathcal{B}_i = \{k \in T | (i-1) \cdot \tfrac{T}{m} < k \leq i \cdot \tfrac{T}{m}\} \tag{5-1}$$

For the first block ($\mathcal{B}_1$) Algorithm 5 is run which results in a set of submodels and a switching sequence. Let Σ_{b1} denote this set of submodels: $\Sigma_{b_1} = \{\theta_1, ..., \theta_s\}$, where s is the number of submodels returned by Algorithm 5. Furthermore let $\sigma_{b1}(k)$ denote the identified switching sequence for the first block. The idea for the subsequent blocks is to use, if possible, the previously identified models. For these blocks a modified version of Algorithm 5 is run. Starting with the second block, The times of switching are determined according to Section 3-1. Then all the intervals between switches in block 2 are checked for feasibility with the previously identified models in set Σ_{b1}. Every interval that fits one of those submodels is removed from the set of intervals that still need to be attributed to a model. Then for the intervals that are left in this set, new models are estimated following the methodology of Section 3-2. These new models in addition to the models from Σ_{b1} together form set Σ. For the remaining blocks the process is repeated by first checking if the models from set Σ fit before identifying new submodels which are then again added to Σ. This modified version of Algorithm 5 is summarised in Algorithm 8 in Appendix D. Step 1 in its entirety is shown in Algorithm 6.

It is possible though not encouraged to stop after this step as the next two steps can be considered refinement steps. The final switching sequence can in this case be obtained by merging the switching sequences of the blocks in the following way: $\sigma(k) = \langle \sigma_{b1}(k), ..., \sigma_{bm}(k) \rangle$.

Algorithm 6 IdentLarge-step1

```
1: procedure FIRST STEP FOR IDENTIFYING LARGE DATASETS
2:     $\mathcal{N} \leftarrow \{1, ..., T\}$
3:     for i=1:m do
4:         $\mathcal{B}_i \leftarrow \{k \in \mathcal{N} | (i-1) \cdot \frac{T}{m} \leq k < i \cdot \frac{T}{m}\}$
5:         if i==1 then
6:             $(\sigma_{b1}(k), \Sigma_{b1}) \leftarrow$ HySI-SAT $(\mathcal{B}_i)$    (see Algorithm 5)
7:             $\Sigma = \Sigma_{b1}$
8:         else
9:             $(\sigma_{bi}(k), \Sigma_{bi}) \leftarrow$ HySI-SAT modified $(\mathcal{B}_i, \Sigma)$    (see Algorithm 8)
10:            $\Sigma = \Sigma \cup \Sigma_{bi}$
11:        end if
12:    end for
13:    $\sigma(k) \leftarrow \langle \sigma_{b1}(k), ..., \sigma_{bm}(k) \rangle$
14:    return $(\Sigma, \sigma(k))$
15: end procedure
```

5-2 Step 2 - Minimising number of submodels

Since not all data was taken into account simultaneously during the estimation of the parameter vectors, the set of submodels Σ generally overapproximates the required number of submodels. The cardinality of Σ can therefore potentially be reduced. First we introduce two sets: $\mathcal{M}$=$\{1, ..., |\Sigma|\}$ and $\mathcal{Q} = \{\mathcal{Q}_1, ..., \mathcal{Q}_p\}$, where p represents the total number of intervals resulting from the first stage. $\mathcal{Q}_j$ describes which models fit interval j, more specifically, $\mathcal{Q}_j = \Big\{i \in \mathcal{M} \Big| \, ||y(k) - \boldsymbol{r}(k)^T \boldsymbol{\theta}_i||_\infty \leq \delta, \quad \forall k \in I_j \Big\}$, and p represents the number of intervals. Now we want to eliminate as many submodels from Σ as possible while ensuring that each interval is accounted for by at least one fitting submodel. This leads to the following problem:

Problem 5-2.1. Given a set $\mathcal{M}$=$\{1, ..., |\Sigma|\}$, and a set $\mathcal{Q}$=$\{\mathcal{Q}_1, ..., \mathcal{Q}_p\}$ where $\mathcal{Q}_i \subseteq \mathcal{M}$. Find the minimal subset of $\mathcal{M}$ so that this subset has at least one element in common with each set $\mathcal{Q}_i$. ($\mathcal{M}_{sub} \cap \mathcal{Q}_i \neq \emptyset, \quad \forall i = 1, ..., p$)

This is further illustrated in Example 5-2.1

Example 5-2.1. Given $\mathcal{M}$=$\{1, 2, 3, 4\}$ and $\mathcal{Q} = \{\{1, 4\}, \{2\}, \{1, 3, 4\}\}$, solve Problem 5-2.1. If we exclude 2 and thus choose $\mathcal{M}_{sub}$ as $\{1, 3, 4\}$, $\mathcal{Q}_2$ would have no element in common with $\mathcal{M}_{sub}$ which is not allowed. However the elements 3 and 4 can safely be excluded, resulting in $\mathcal{M}_{sub} = \{1, 2\}$. This hitting set is a minimum hitting set since a hitting set with fewer element does not exist.

Problem 5-2.1 is a form of the minimum hitting set problem, which is a known problem in mathematics. It is equivalent to the minimum set cover problem [39], which is NP-hard to solve [40]. Multiple approaches to solving the minimum set cover problem exist. The decision problem of hitting set/set cover is NP-complete [40] and can therefore be cast to a SAT problem in polynomial time. Solving a sequence of these SAT problems now solves the minimum hitting set problem. This approach is used in the HySI-SAT program, as it does not require new software since the SAT solver is already used in a different phases of the algorithm. The mapping from [41] is used to map the hitting set problem to a Boolean satisfiability problem.

5-2-1 Solving the minimum hitting set problem using a SAT solver

Let us introduce the Boolean variables $x_i, i = 1, ..., p$. Each Boolean is associated with its corresponding element from $\mathcal{M}$ in Problem 5-2.1. The Boolean is true iff the corresponding element is in subset $\mathcal{M}_{sub}$. For every $\mathcal{Q}_i$ it is required to have atleast one element in common with subset $\mathcal{M}_{sub}$. This can be encoded by the following clause:

$$C_{\mathcal{Q}i} = \bigvee_{j \in \mathcal{Q}_i} x_j \tag{5-2}$$

For $\mathcal{Q}_i = \{2, 5, 8\}$ this would result in the clause: $x_2 \vee x_5 \vee x_8$. The course of action consists of finding the minimum hitting set by solving a sequence of SAT problems with an increasing bound on the sum of the Booleans. The cardinality is constraint by the following clause.

$$C^k_{card} = (\sum_i x_i \leq k) \tag{5-3}$$

This yields the following SAT formula:

$$C^k = C_{\mathcal{Q}1} \wedge C_{\mathcal{Q}2} \wedge ... \wedge C_{\mathcal{Q}p} \wedge C^k_{card} \tag{5-4}$$

A minimum hitting set can now be obtained by feeding this CNF-formula to a SAT solver for k=1, and repeat this for increasing values of k till the SAT solver returns *SAT*. This is illustrated in Algorithm 7.

Algorithm 7 IdentLarge-step2

```
1: procedure SECOND STEP FOR IDENTIFYING LARGE DATASETS
2:     $k \leftarrow 1$
3:     while true do
4:         (satisfiability, $\mu$) $\leftarrow$ SAT solver($C^k$)
5:         if satisfiability == $UNSAT$ then
6:             $k \leftarrow k + 1$
7:         else
8:             return (satisfiability,$\mu$)
9:         end if
10:    end while
11: end procedure
```

Solving the minimum hitting set problem presumably provides a reduced set of models. All intervals are attributed to the model from this reduced set that gives the smallest error. This results in the final switching sequence.

5-3 Step 3 - Final parameter estimation

The final step consists of estimating the parameter vectors one last time. Some intervals were attributed to different submodels in the previous step. Since the final switching sequence is already known, this problem falls back to a sequence of linear identification problems. For every submodel Problem 5-3.1 is solved.

Problem 5-3.1. Given a dataset $\langle y(k), u(k)\rangle_{k=1}^{T}$, a switching sequence σ(k), and a set Σ containing s different submodels, solve the following minimisation problem for i=1:s

$$\begin{aligned}&\min_{\boldsymbol{\theta}_i} \sum_{k=1}^{T} \left(y(k) - \boldsymbol{\theta}_i^T \boldsymbol{r}(k)\right)^2 \\ &|y(k) - \boldsymbol{\theta}_i^T \boldsymbol{r}(k)| \leq \delta, \quad \forall k \in \{k \in T | \sigma(k) = i\}\end{aligned}$$

Note that instead of splitting a large dataset the procedure described in this chapter, is also capable of combining datasets from different experiments.

Chapter 6

Experiments

In this chapter the performance of the proposed method is evaluated. Numerical experiments are conducted in addition to a case study on a real system. The proposed method was tested by using the toolbox which was created supplementary to this thesis report. The toolbox uses Z3solve [42] as a SAT solver and CPLEX [43] as a convex optimisation tool. For more information on this toolbox see Appendix C. All experiments where the computation time is shown were conducted on a laptop running Microsoft Windows 10 with 8GB of RAM and a Intel(R) Core(TM) i7-3630 CPU at 2.4 GHz. Moreover, the toolbox is written in python and compiled to C code.

To quantify the performance of the proposed method and compare it to established identification techniques some quality metrics are introduced. The FIT [44] is used as a measure for the accuracy of the datafit.

$$FIT = 100\% \cdot \left(1 - \frac{||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2}{||\boldsymbol{y} - \bar{y} \cdot \boldsymbol{e}||_2}\right) \tag{6-1}$$

The normalised parameter identification error measure [1], is used to quantify the error between the identified and real parameters.

$$\Delta_n = \frac{1}{T}\sum_{1}^{T} \frac{||\boldsymbol{\theta}(k) - \hat{\boldsymbol{\theta}}(k)||_2}{||\boldsymbol{\theta}(k)||_2} \tag{6-2}$$

Finally the Root-mean-squared-error is used in Section 6-3 to compare the standard deviation of the residuals.

$$RMSE = \sqrt{\frac{1}{T}\sum_{k=1}^{T} \left(\hat{y}(k) - y(k)\right)^2} \tag{6-3}$$

6-1 Synthetic systems

6-1-1 Switched ARX models

To evaluate the performance of the presented method a SARX system is borrowed from the work of Ozay et al. [1]. In this SARX system the output is given by

$$y(k) = \begin{bmatrix} y(k-1) & y(k-2) & u(k-1) \end{bmatrix} \boldsymbol{\theta}_{\sigma(k)} + \omega(k) \tag{6-4}$$

with

$$\begin{aligned} \boldsymbol{\theta}_1 &= \begin{bmatrix} 0.2 & 0.24 & 1 \end{bmatrix}^T, \\ \boldsymbol{\theta}_2 &= \begin{bmatrix} -1.4 & -0.53 & 2 \end{bmatrix}^T \end{aligned} \tag{6-5}$$

and the switching is defined as

$$\sigma(k) = \begin{cases} 1 & \text{if } \ k \le 25 \text{ or } 50 < k \le 75 \\ 2 & \text{if } \ 25 < k \le 50 \text{ or } k > 75 \end{cases} \tag{6-6}$$

An uniformly distributed input, $u(k) \sim \mathcal{U}$(-1,1), is used to simulate the system for T=100. The noise is uniformly distributed, $\omega(k) \sim \mathcal{U}(-\omega_{max}, \omega_{max})$. The example system is simulated for 15 different values of ω_{max}, ranging from 0.1 to 1.5. For every value of ω_{max}, 10 different runs were simulated. The mean of Δ_n is displayed in Figure 6-1, and Figure 6-2 illustrates the estimated number of submodels. The algebraic approach [6], and the sparsification approach [1] are included for comparison purpose. Note that for the algebraic method the number of submodels is assumed to be known beforehand. For HySI-SAT, τ_d is set to 20 during identification.

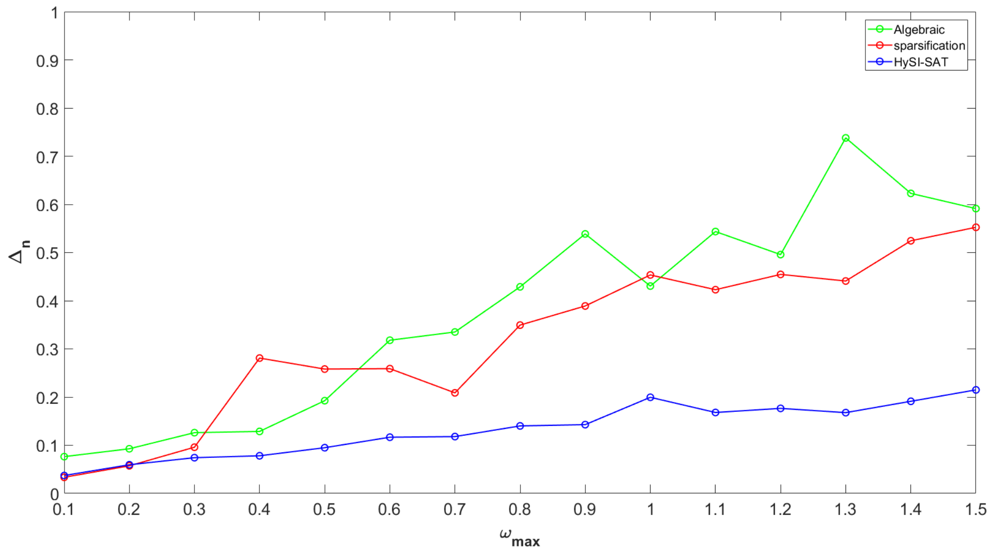


Figure 6-1: Mean value of Δ_n for example system 2 with different values of ω_{max}.

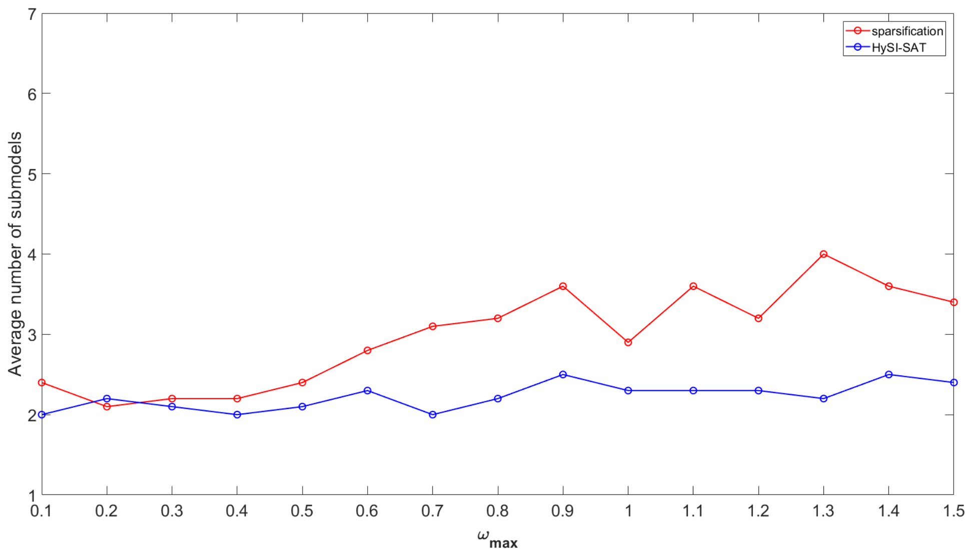


Figure 6-2: Average number of submodels for example system 2 with different values of ω_{max}.

Table 6-1: Δ_n for identification of system 6-7

	HySI-SAT	Segreg[11]	Son-em[15]
Δ_n	0.0134	0.0318	0.0174

Segmentation

The sparsification approach and the algebraic approach do not assume any continuity in the switching sequence. Now the presented method is compared against a methods that penalise changes in the parametervector. Consider the following SARX model consisting of 3 modes [14]:

$$y(k) = \begin{bmatrix} y(k-1) & y(k-2) & u(k) & u(k-1) & u(k-2) \end{bmatrix} \boldsymbol{\theta}_{\sigma(k)} + \omega(k) \tag{6-7}$$

with

$$\begin{aligned} \boldsymbol{\theta}_1 &= \begin{bmatrix} 0.6 & 0.3 & 1 & 0.5 & -0.3 \end{bmatrix}^T, \\ \boldsymbol{\theta}_2 &= \begin{bmatrix} 0.7 & -0.6 & 1.4 & -0.4 & -0.2 \end{bmatrix}^T, \\ \boldsymbol{\theta}_3 &= \begin{bmatrix} -0.5 & -0.4 & -0.2 & 1.3 & 0.9 \end{bmatrix}^T \end{aligned} \tag{6-8}$$

and the switching is defined as

$$\sigma(k) = \begin{cases} 1 & \text{if } k \leq 60 \\ 2 & \text{if } 60 < k \leq 130 \\ 3 & \text{if } k > 130 \end{cases} \tag{6-9}$$

The input values are chosen from an uniform distribution $\mathcal{U}$(-1,1), and the noise ω(k) is Gaussian with normal distribution, $\omega(k) \sim \mathcal{N}(0, 0.05^2)$. This system is simulated for T=200, and identified with HySI-SAT (SARX), Segreg [11] and SON-EM [15]. The identification with Segreg and SON-EM was conducted using Matlab [45] code retrieved from [46] and [47] respectively. The estimated parameter vectors are compared to their true values in Table 6-2. The normalised parameter identification error is shown in Table 6-1.

Table 6-2: Estimated parameter vectors for system 6-7

	HySI-SAT	Segreg [11]	SON-EM [15]	True value
$\boldsymbol{\theta}_1$	$\begin{bmatrix}0.6143\\0.2847\\0.9924\\0.5029\\-0.2950\end{bmatrix}$	$\begin{bmatrix}0.5557\\0.3452\\0.9942\\0.5621\\-0.3119\end{bmatrix}$	$\begin{bmatrix}0.5895\\0.3099\\1.0150\\0.5026\\-0.2834\end{bmatrix}$	$\begin{bmatrix}0.6\\0.3\\1\\0.5\\-0.3\end{bmatrix}$
$\boldsymbol{\theta}_2$	$\begin{bmatrix}0.7008\\-0.5937\\1.392\\-0.3891\\-0.2111\end{bmatrix}$	$\begin{bmatrix}0.6983\\-0.6079\\1.404\\-0.3931\\-0.1778\end{bmatrix}$	$\begin{bmatrix}0.7011\\-0.5975\\1.4189\\-0.4084\\-0.2031\end{bmatrix}$	$\begin{bmatrix}0.7\\-0.6\\1.4\\-0.4\\-0.2\end{bmatrix}$
$\boldsymbol{\theta}_3$	$\begin{bmatrix}-0.4863\\-0.3975\\-0.1913\\1.309\\0.8917\end{bmatrix}$	$\begin{bmatrix}-0.5262\\-0.3969\\-0.2037\\1.307\\0.9170\end{bmatrix}$	$\begin{bmatrix}-0.4829\\-0.4087\\-0.2162\\1.3145\\0.8802\end{bmatrix}$	$\begin{bmatrix}-0.5\\-0.4\\-0.2\\1.3\\0.9\end{bmatrix}$

6-1-2 PWARX examples

The following synthetic system is used to demonstrate the ability of HySI-SAT to identify PWARX systems. Let the output of a system as described by Equation 2-3 be given as:

$$y(k) = \begin{cases} \begin{bmatrix}0.6 & -0.4 & 0.4\end{bmatrix}\begin{bmatrix}\boldsymbol{r}(k)\\1\end{bmatrix} + \omega(k), & \text{if } \boldsymbol{r}(k) \in \mathcal{R}_1 \\ \begin{bmatrix}0.5 & 0.7 & 0.2\end{bmatrix}\begin{bmatrix}\boldsymbol{r}(k)\\1\end{bmatrix} + \omega(k), & \text{if } \boldsymbol{r}(k) \in \mathcal{R}_2 \\ \begin{bmatrix}0.4 & 0.4 & -1.3\end{bmatrix}\begin{bmatrix}\boldsymbol{r}(k)\\1\end{bmatrix} + \omega(k), & \text{if } \boldsymbol{r}(k) \in \mathcal{R}_3 \end{cases} \tag{6-10}$$

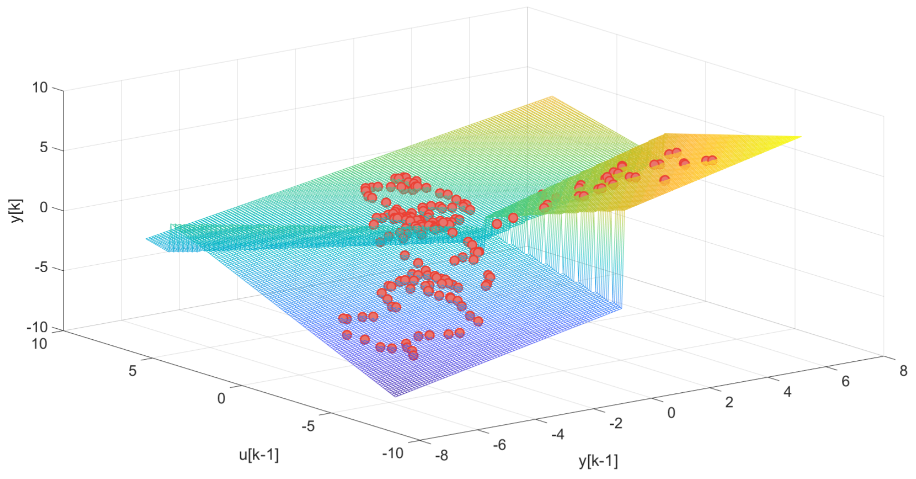


Figure 6-3: Identified PWA map using HySI-SAT. Red dots represent datapoints.

where $\boldsymbol{r}(k) = \begin{bmatrix} y(k-1) & u(k-1) \end{bmatrix}^T$ and the regions are defined as

$$\mathcal{R}_1 = \{\boldsymbol{r}(k) \in \mathbb{R}^2 | \begin{bmatrix} -5 & 1 & -2 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} \leq 0\} \tag{6-11}$$

$$\mathcal{R}_2 = \{\boldsymbol{r}(k) \in \mathbb{R}^2 | \begin{bmatrix} 5 & -1 & 2 \\ 1 & 3 & -3 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} \leq \mathbf{0}\} \tag{6-12}$$

$$\mathcal{R}_3 = \{\boldsymbol{r}(k) \in \mathbb{R}^2 | \begin{bmatrix} -1 & -3 & 3 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} \leq 0\} \tag{6-13}$$

The dataset used for identification is obtained by applying the following input for T=200

$$u(k) = 1.1 \cdot sin(k) + 4 \cdot sin(0.2k) + 0.8 \cdot sin(3k) \tag{6-14}$$

$\omega(k)$ is simulated as a uniformly distributed sequence $\omega(k) \sim \mathcal{U}(-0.1, 0.1)$. For validation a sawtooth signal of amplitude 10 and period $2 \cdot \pi$ was used.

Table 6-3: FIT for validation of system 6-10

	HySI-SAT	PWASON[48]
FIT [%]	69.0	60.3

6-1-3 Frequent switching

This section evaluates the performance of the proposed method on a system where during simulation the submodels were activate for only a short period before switching to the next submodel. The system in question is borrowed from [16] (also portrayed in Appendix B) and simulated for T=50. The identified switching sequence and true switching sequence are shown

Table 6-4: Identified model parameters system 6-10. 2 datapoints are attributed to extra mode 4

	Identified parameter vectors
$\boldsymbol{\theta}_1$	$\begin{bmatrix}0.570 & -0.426 & 0.429\end{bmatrix}^T$
$\boldsymbol{\theta}_2$	$\begin{bmatrix}0.507 & 0.691 & 0.194\end{bmatrix}^T$
$\boldsymbol{\theta}_3$	$\begin{bmatrix}0.411 & 0.394 & -1.28\end{bmatrix}^T$
$\boldsymbol{\theta}_4$	$\begin{bmatrix}-3.00 & -0.0214 & -2.10\end{bmatrix}^T$

in Figure 6-4. It can be seen that the true switching sequence is erratic. In fact, only for 7 samples the regressor was located in the same region as previous time instance. From this experiment it becomes apparent that the presented methodology does not function properly on datasets with a very low dwell time.

Remark 6-1.1. *Note that for a SARX systems that switch every time instance, the switching time sequence would be given by $\langle n \rangle_{n=1}^{T}$. In the optimal case that this would be the switching time sequence determined in the first step, the second step of our method becomes equivalent to the approach from [1]. Therefore in these situations their algorithm is preferred.*

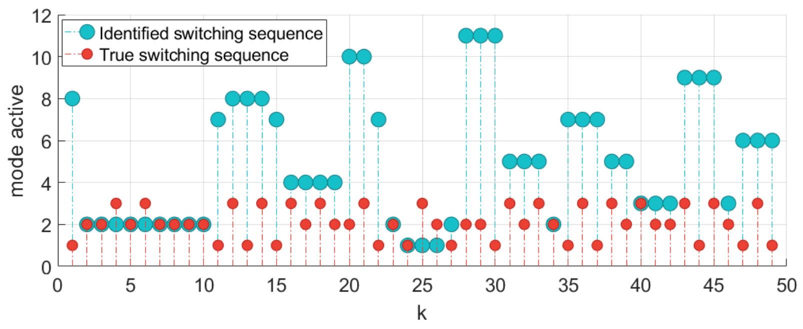


Figure 6-4: Identified switching sequence in comparison to true switching sequence

6-2 Random systems

The following synthetic experiment is involved with the effect of the number of data points in comparison to the computation time and accuracy of the parameter estimation. To evaluate this random systems were created and simulated. The specification of these systems can be found in Appendix A-1. For each case 100 random systems were simulated. The results are shown in Table 6-5. The results show that as the number of datapoints increases the accuracy of the estimated parameters and the computation time also generally increases, which is to be expected.

Table 6-5: Computation time and Δ_n for increasingly large datasets

	T	60	120	180	240	300	360
Δ_n	median	$5.5 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$	$3.4 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$	$3.3 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$
	(IQR)	(0.0114)	(0.0102)	(0.0056)	(0.0151)	(0.0119)	(0.0095)
computation time [s]	median	3.48	19.6	52.7	117	101	269
	(IQR)	(3.56)	(13.3)	(41.1)	(78.5)	(109)	(191)

Table 6-6: Trade-off computation time versus number of blocks

	Number of blocks	1	2	3
computation time [s]	mean	$2.82 \cdot 10^3$	$1.41 \cdot 10^3$	$1.14 \cdot 10^3$
	(std)	(453)	(346)	(504)
estimated # of submodels	mean	3.50	5.70	7.10
	(std)	(1.65)	(2.21)	(2.64)

Large datasets

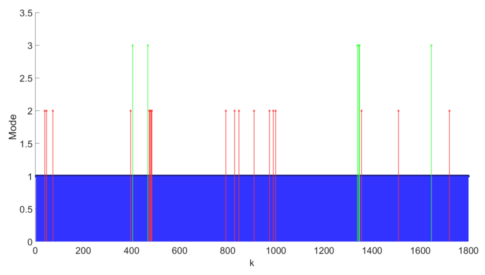
The next experiment demonstrates the procedure described in Chapter 5 for dealing with large datasets. It is applied to datasets of 2000 input-output pairs. These datasets are generated by random SARX systems described in Appendix A-2. For each case 10 random datasets were generated. The average computation time and number of submodels is displayed in Table 6-6. The trade-off between computation time and the number of blocks the dataset is separated in, can clearly be seen.

6-3 Experimental data

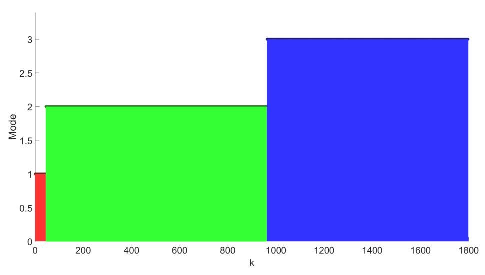
6-3-1 Modeling room temperature

As stated in the introduction, the objective of the proposed method is to minimise the number of submodels while maintaining a minimum dwell time. This will now be demonstrated by modeling the temperature of a room using a dataset containing 1800 input-output data samples (Figure 6-9). The inputs consist of the temperature of adjacent rooms (T_1-T_3), the outside air temperature (T_o), the supply air mass flow (W_A) and lastly the Temperature of the supply air (T_A). The Temperatures were measured in degrees Celsius ($°C$), and the mass flow in kilograms per second ($kg \cdot s^{-1}$). The dataset was identified by HySI-SAT and the sparsification based approach [1] which also aims at minimising the number of submodels.

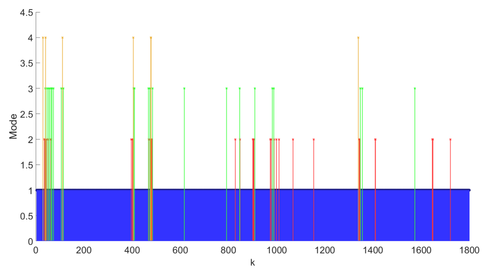
The resulting switching sequence is shown in Figure 6-5. It can be seen that for the bound on the error given by $\delta = 0.36$, the same number of submodels was obtained for both methods while a significant difference in the switching frequency can be observed. For $\delta = 0.26$ the sparsification approach is able to explain the data with less models. However the switching frequency corresponding to HySI-SAT is significantly lower, which fulfils our objective of identifying models with less switches.



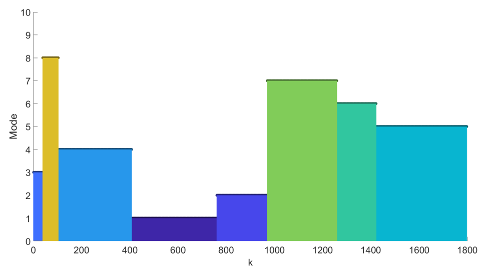
(a) Sparsification $\delta = 0.36$



(b) HySI-SAT $\delta = 0.36$



(c) Sparsification $\delta = 0.26$



(d) HySI-SAT $\delta = 0.26$

Figure 6-5: Switching sequence obtained for 2 different bounds on the error.

6-3-2 Case study

This section is concerned with the identification of a real system in order to demonstrate the ability of HySI-SAT to approximate nonlinear systems by a piecewise affine model. The system in question is the coupled electric drives [49]. The dataset was provided by Wigren and Schoukens [50], and is part of a collection of nonlinear benchmarking datasets which are made publicly available.

System description

The coupled electric drives consists of two electric motors connected to a pulley by a flexible belt. The pulley is held in place by a spring. The setup is shown in Figure 6-6. The input and output of the system are respectively the voltage applied to the motors and the measured angular speed of the pulley. The data is collected by applying a piecewise constant input signal that switches every 5 samples to a random value in the range of -1.5 V to +3.0 V. The input and output were obtained with a sampling period of 20 milliseconds. Two datasets were provided. The first one was split into a training set and a validation set. 200 samples were used for training and 300 for validation. The second dataset was only used for validation. During the validation phase the input data and the first three values of the measured output were used. The following hyperparameters were used during identification: $L_s = 20, \delta = 0.05$.

The quality measure to evaluate the performance on the benchmark dataset is the Root mean squared error, given by Equation 6-3.

The performance is compared to different techniques from the literature that were as well tested on this benchmark dataset. The RMSE of the following methodologies is considered:

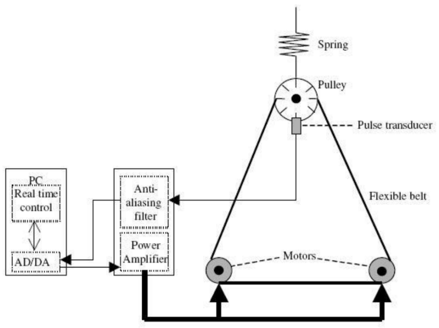
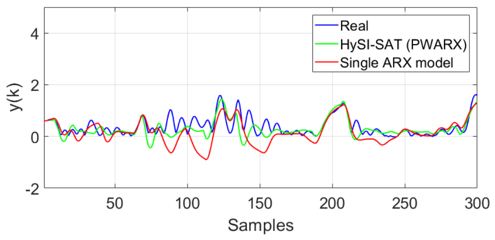
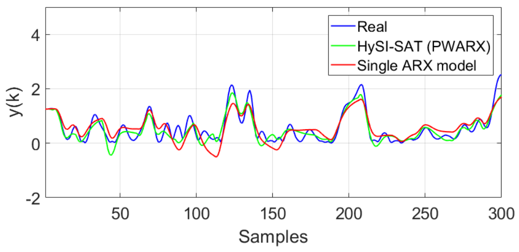


Figure 6-6: Schematic overview of the coupled electric drives [2]



(a) Validation on dataset 1

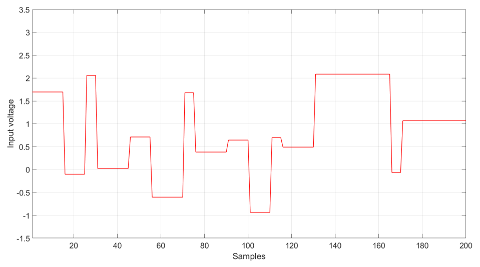


(b) Validation on dataset 2

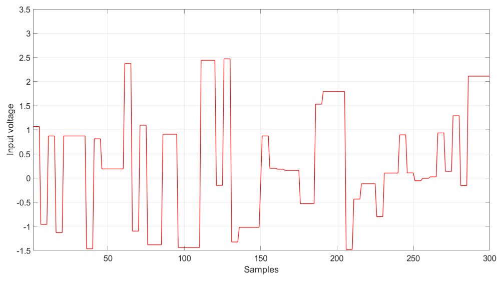
Figure 6-7: The simulated output of the model on the validation datasets

an approach which uses a neural network with radial basis functions [51], a fuzzy system and an extended fuzzy logic system (FLe), both reported by [52], and lastly a LOLIMOT [54] fuzzy model reported in [53]. In addition to results from the literature, a different technique for identifying PWARX model was considered, which is the sum of norm regularisation method proposed by Ohlsson and Ljung [48]. Using their code [55], a large range of values for the different tuning parameters was tried out. The best result was obtained using N=12, and λ=0.0001. Lastly, to put the results into perspective, a single 3th order ARX model was fitted to the data.

LOLIMUT was trained in [53] using the first 374 samples of the second dataset and validated using the next 126 samples. In [51], the first dataset was used for training the neural network



(a) Input for training the model



(b) Input for validation

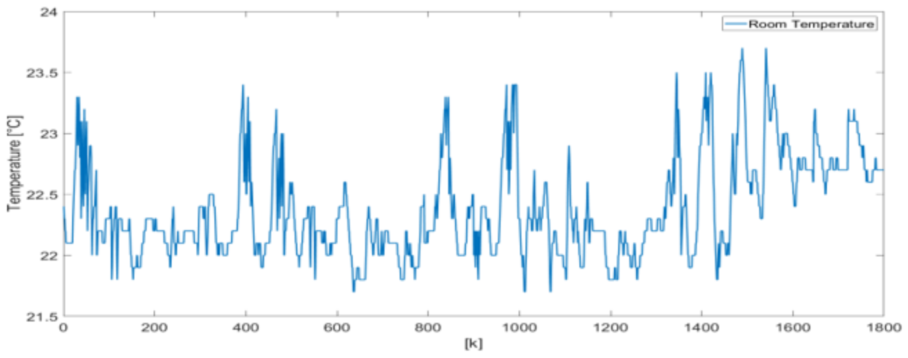
Figure 6-8: Input coupled electric drives dataset 1

Table 6-7: Comparison RMSE on the coupled electric drive dataset

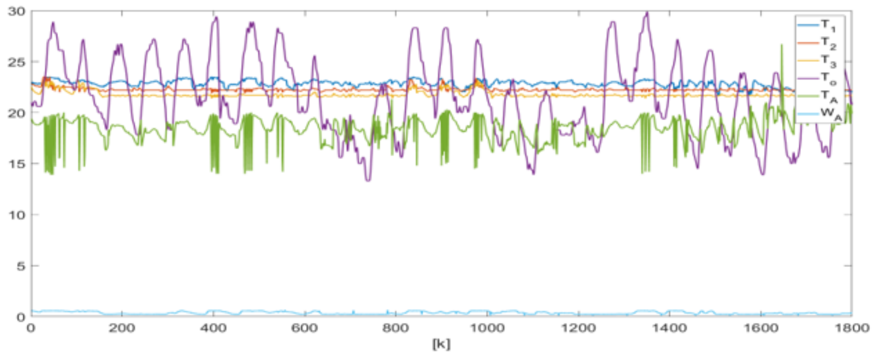
	Single ARX model	HySi-SAT (PWARX)	PWASON [48]	RBFNN[1] [51]	Fuzzy system [52]	FLe [52]	LOLIMOT [1] [53], [54]
Dataset 1	0.463	0.304	0.397	-			-
					0.323	0.0921	
Dataset 2	0.299	0.231	0.255	0.185			0.0699

and the second dataset for validating it. In [52] no details were provided on which of the two datasets was used and neither on the quantity of samples used for training and validation.

The Root-mean-squared-error is portrayed in Table 6-7.



(a)



(b)

Figure 6-9: Output **(a)** and inputs **(b)** of dataset Section 6.3.1

[1]A different error measure was reported which was converted to the RMSE for easier comparison

Chapter 7

Discussion and conclusion

In this chapter the proposed algorithms and the experimental results from Section 6 are discussed, and some interesting properties of the algorithm are highlighted. Finally, the thesis is concluded and recommendations for future research are provided.

7-1 Discussion

One goal that was set out for this thesis project was to develop an identification method for SARX models that minimises the number of submodels while respecting the minimum dwell time. This was motivated by the observation that the approach proposed in [1] worked very well for minimising the number of submodels, however resulted in a model with many switches, and overfitting of some submodels. To validate if the presented approach deals with these issues both methods are applied to experimental data in Section 6-3-1 The results show that performance of HySI-SAT on the front of minimising the number of submodels comes close or even matches that of the method described in [1]. Nevertheless the model identified by HySI-SAT has considerably less switches due to the minimum dwell time constraint. Noteworthy is the fact that the identified switching sequence seem to exhibit a far longer average dwell time than is enforced by the minimum dwell time constraint. This behaviour is observed more often than not. We suspect this to be a consequence of the specific SAT solver used. The SAT solver tend to prefer valuations with relatively many Booleans assigned to *False*, resulting in sporadic switching. This effect would be interesting to look into in the future.
Two synthetic SARX systems from the literature were identified in Section 6-1-1. HySI-SAT was able to accurately estimate the model parameters, and recover the switching sequence. The proposed algorithm for identifying PWARX models is also validated on two example systems (Section 6-1-2). These experiments show that the suggested algorithm is proficient in identifying PWARX models when the system transitions slowly between modes. When the system switches rapidly the performance deteriorates, which results in a complex model with many submodels.

In Section 6-3 the proposed method is validated on a benchmark dataset. Experimental data from the coupled electric drives is used to estimate a model and afterwards validate it. The error for the validation data, displayed in Table 6-7, shows that HySI-SAT has a lower error in contrast to a single ARX model and PWASON [48]. The error of the proposed method is considerably higher than the reported errors of the neural network, fuzzy extended logic system and the fuzzy system reported in [53]. However one benefit the PWARX model identified via HySI-SAT has in respect to the last three is it's simplicity. The identified PWARX model consists of 5 third order affine submodels while the methods with a considerably lower error make either use of a more complex regressor or significantly more local models.

7-1-1 Properties of HySI-SAT

HySI-SAT is deterministic, given that the SAT solver is deterministic and the method for resolving progress obstruction (Section 3-2-3) is deterministic. If these conditions are fulfilled, running the algorithm twice, will result in the same identified model. Another interesting property arises by the incorporation of a SAT solver in the identification process.

Different constraints

The inclusion of a SAT solver presents the opportunity to add extra constraints. Many constraints on the desired times of switching between models can straightforwardly be added to the SAT problem. A few examples will be given, but this list is by no means exhaustive.

- Particular number of switches:
  $\sum b(k) = d_s$
  where d_s denotes the desired number of switches.
- No switch in a particular time frame:
  $\neg b(k_{start}) \wedge ... \wedge \neg b(k_{end})$
- Required switch in a particular time frame:
  $b(k_{start}) \vee ... \vee b(k_{end})$

The last two constraints can be added to include extra information. For instance, when it is observed that a physical process visibly changes around a certain time but the exact time is not known, the third example can be added to enforce a switch in a specific time window.

7-2 Conclusion

In this thesis a novel method for the identification of Piecewise affine ARX and Switched ARX systems from input-output data is presented and discussed. This approach seeks to find a SARX/PWARX model with a minimum number of switches that is consistent with the data and respects a minimum dwell time. This method first segments the dataset into feasible subsets. The segmentation is performed by an iterative procedure between a SAT solver and a convex optimisation tool. Then the identification problem is posed as a problem of fitting the minimum number of submodels to these subsets. Based on the method of Ozay et al. [1], this problem is reformulated to a ℓ_0-minimisation problem of a sequence, and subsequently relaxed to a convex optimisation problem.

A procedure for dealing with large datasets is also presented. Here the dataset is divided into parts that are successively identified. Next, the identified models are fitted to the data. Finally, by means of solving a form of the minimum hitting set problem the number of submodels is further minimised.

The proposed methods are demonstrated by numerical experiments. These experiments suggest that it holds up well against established methods from the literature. However the proposed method does not seem appropriate for datasets with frequent switching.

The suggested method for PWARX system identification is applied to experimental data from the coupled electric drives. This highly nonlinear system is used as a benchmark system in the literature. The results show a good combination of datafit and model simplicity.

Supplementary to this thesis a toolbox is developed that incorporates the presented algorithms. This toolbox is publicly available and free to use/build upon.

7-3 Future work

Datapoints close to a switch

The SAT solver does not look for the optimal switching sequence but for a feasible one. Datapoints close to a switch can sometimes be included in both intervals adjacent to that switch, and result in a feasible switching sequence. Assigning points to the wrong interval can cause datapoints that were simulated by different subsystems to be grouped together. In this case the interval is commonly classified as a new separated mode which results in extra modes in the final model. It would be interesting to look into a way of refining the switching time sequence after the first step, by shifting particular switching times marginally backward or forward in time.

Spacial seperation

During the first phase a feasible switching sequence is established in which time instances between switches are classified to belong to the same submodel. This can be seen as a preclustering procedure. It inherently favors datapoints in close temporal proximity to belong to the same submodel. For PWARX systems that switch frequently it would be interesting to partition the dataset based on spacial features. The Booleans could encode a partitioning based on location of the datapoints. The SAT solver proposes a separating in groups, which will each be checked for feasibility. This iterative procedure could work in a similar manner to the presented approach in this work.

Outliers

Momentary sensor failure or unmodeled dynamics can result in observations that deviate considerably from the expected value. These outliers in the data can degrade the performance of identification. This motivates the search for a method of dealing with outliers. In [34], [56] methods which are robust to outliers are presented. This raises the question how the proposed method can be shaped to remove (the effect of) outliers.

ℓ_1-relaxation

Recently alternatives to the reweighted ℓ_1-relaxation have emerged that choose the weights differently. Le [57] proposed a different method of choosing the weights that results in better convergence to the ℓ_0-norm. Choosing the weights differently might be one road worth pursuing to improve HySI-SAT.

Appendix A

Randomly generated systems

A-1 Experiment: Size of dataset

The system is described by the following equation.

$$
\begin{aligned}
&y(k) = \begin{bmatrix} y(k-1) & y(k-2) & u(k-1) & u(k-2) \end{bmatrix} \boldsymbol{\theta}_{\sigma(k)} + \epsilon(k) \qquad \text{(A-1)}\\
&\text{with}\\
&\boldsymbol{\theta}_1 = \begin{bmatrix} a_1^1 & a_2^1 & b_1^1 & b_2^1 \end{bmatrix}^T,\\
&\boldsymbol{\theta}_2 = \begin{bmatrix} a_1^2 & a_2^2 & b_1^2 & b_2^2 \end{bmatrix}^T,\\
&a_1^1, a_2^1, a_1^2, a_2^2 \sim \mathcal{U}(-0.5, 0.5),\\
&b_1^1, b_2^1, b_1^2, b_2^2 \sim \mathcal{U}(-2, 2)\\
&\sigma(k) \in \{1, 2, 3\}
\end{aligned}
$$

with $\epsilon(k) \sim \mathcal{U}(-0.1, 0.1)$ and $u(k) \sim \mathcal{U}(-1, 1)$. The discrete state stays constant for 16 samples at a time. Then every sample it has a probability of one in three to change to a randomly picked different mode until it switches. The first two values of y(k) are samples from the $\mathcal{U}(-1, 1)$ distribution.

A-2 Experiment: Large datasets

The system is identical to system A-1, except on two fronts. First the noise description is given as $\epsilon(k) \sim \mathcal{U}(-0.03, 0.03)$, and secondly the discrete state stays constant for atleast 20 samples at a time.

Appendix B

Simulation with frequent switching

The following system system from [16] was identified in Section 6-1-3.

$$y(k) = \begin{cases} \begin{bmatrix} -0.4 & 1 & 1.5 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} + \omega(k), & \text{if } \boldsymbol{r}(k) \in \mathcal{R}_1 \\ \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} + \omega(k), & \text{if } \boldsymbol{r}(k) \in \mathcal{R}_2 \\ \begin{bmatrix} -0.3 & 0.5 & -1.7 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} + \omega(k), & \text{if } \boldsymbol{r}(k) \in \mathcal{R}_3 \end{cases} \tag{B-1}$$

where $\boldsymbol{r}(k) = \begin{bmatrix} y(k-1) & u(k-1) \end{bmatrix}^T$ and the regions are defined as

$$\mathcal{R}_1 = \{\boldsymbol{r}(k) \in \mathbb{R}^2 | \begin{bmatrix} 4 & -1 & 10 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} \leq 0\} \tag{B-2}$$

$$\mathcal{R}_2 = \{\boldsymbol{r}(k) \in \mathbb{R}^2 | \begin{bmatrix} -4 & 1 & -10 \\ 5 & 1 & -6 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} \leq \mathbf{0}\} \tag{B-3}$$

$$\mathcal{R}_3 = \{\boldsymbol{r}(k) \in \mathbb{R}^2 | \begin{bmatrix} -5 & -1 & 6 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(k) \\ 1 \end{bmatrix} \leq 0\} \tag{B-4}$$

$$\tag{B-5}$$

where u(k) and $\omega(k)$ are sampled from the uniformly distribution $\mathcal{U}$(-4,4) and $\mathcal{U}$(-0.2,0.2) respectively.

Appendix C

HySI-SAT toolbox

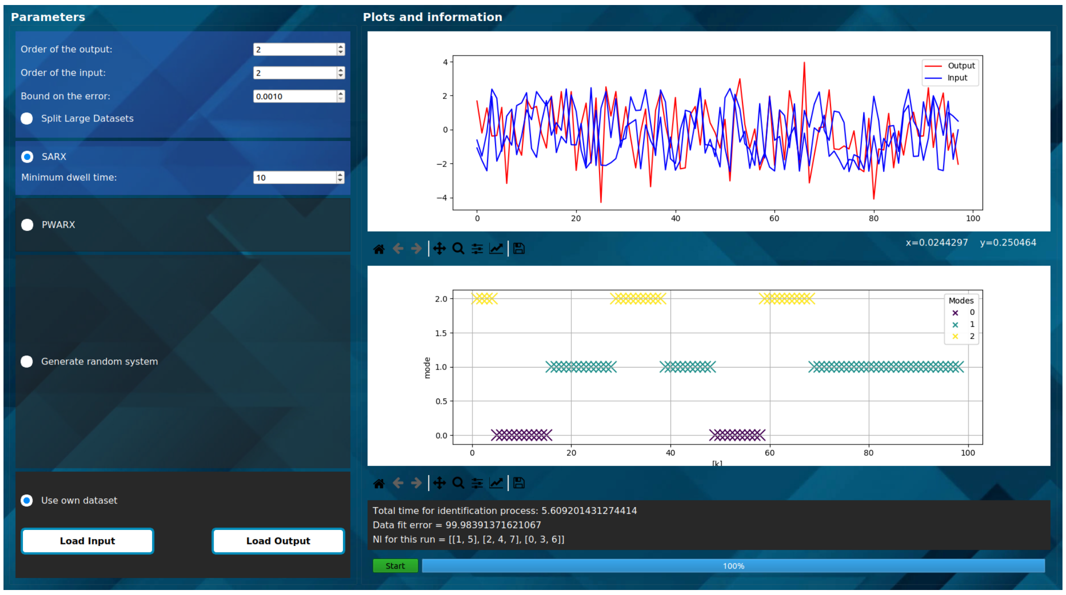
This section contains information about the toolbox supplementary to this thesis work. First a short overview of the toolbox will be given, followed by a brief instruction on how one could use the tool.

C-1 Overview

HySI-SAT is an application written in python and compiled to C using cython. The toolbox requires Z3 solve [42] as a SAT solver and Cplex [43] as a convex optimisation program. The toolbox can be obtained from [18]. A more elaborate manual and installation instructions can also be retrieved from there.

C-2 Instruction

The toolbox features a Graphical User Interface (GUI) where the user can interact with the program. An overview of the GUI is given in Figure C-1. On the left side parameters can be specified in addition to the choice of identifying either a PWARX model or a SARX model. The user can either generate a random model for identification which can be used to compare this method with other methods, or input a dataset of input output data. The input and output should be supplied as a comma seperated text file. When the dataset is loaded the input and output are displayed on the right. The user can start identification by pressing the start button. A bar on the bottom of the screen shows the progress. When identification is completed, the resulting switching sequence is displayed on the right. An information panel will display information during and after identification. When the identified model meets the demands, the gui can be closed. A folder will be created, named after the time and date of identification, that contains information about the identified model. The switching sequence, model parameters, regressor, input and output are stored in the form of Matlab formatted data. Moreover, in case a PWARX model is identified, one extra file is added.



Order of the output: 2
Order of the input: 2
Bound on the error: 0.0010
Split Large Datasets
[✓ SARX]
Minimum dwell time: 10
PWARX
Generate random system
[✓ Use own dataset]

x=0.0244297 y=0.250464

```
Total time for identification process: 5.609201431274414
Data fit error = 99.98391371621067
NI for this run = [[1, 5], [2, 4, 7], [0, 3, 6]]
```

Figure C-1: Graphical user interface of the HySI-SAT program

This file stores the coefficient matrices that determine the partitioning of the regressorspace. Every identification procedure is saved to a database that allows easy storage and retrieval of previously identified models.

Appendix D

HySI-SAT modified

The following algorithm appears in Section 5 to identify models for every block except the first one.

Algorithm 8 HySI-SAT modified

```
1: procedure SARX IDENTIFICATION BLOCK $\mathcal{B}_i$
2:     $(S, \mathcal{T}\text{-satisfiability}) \leftarrow \textit{HySI-SAT step } 1(\langle y(k), u(k)\rangle | k \in \mathcal{B}_i)$
3:     $l \leftarrow 0$
4:     $\mathcal{N}_1 \leftarrow \{1, ..., |\mathcal{S}| + 1\}$
5:     Fit previously identified submodels to the data:
6:     while $l < |\Sigma|$ do
7:         $l \leftarrow l + 1$
8:         $K_l \leftarrow \{j \in \mathcal{N}_l| \quad ||y(k) - \boldsymbol{r}(k)^T \boldsymbol{\theta_l}||_\infty \leq \delta, \quad \forall k \in \mathcal{I}_j\}$
9:         $\mathcal{N}_{l+1} \leftarrow \mathcal{N}_l \setminus \mathcal{K}_l$
10:    end while
11:    Identify submodels for the remaining intervals:
12:    while $|\mathcal{N}_{l+1}| > 0$ do
13:        $l \leftarrow l + 1$
14:        $\boldsymbol{w} \leftarrow \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}^T$
15:        for p=1:$\eta$ do
16:            $(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}) \leftarrow Equation_{3-17}(\mathcal{N}_l, \boldsymbol{w})$
17:            $w_j \leftarrow \frac{1}{z_j + \epsilon}, \quad \forall j \in \mathcal{N}_l$
18:        end for
19:        $\boldsymbol{\theta_l} \leftarrow \tilde{\boldsymbol{\theta}}$
20:        $K_l \leftarrow \{j \in \mathcal{N}_l| \quad ||y(k) - \boldsymbol{r}(k)^T \boldsymbol{\theta_l}||_\infty \leq \delta, \quad \forall k \in \mathcal{I}_j\}$
21:        $\mathcal{N}_{l+1} \leftarrow \mathcal{N}_l \setminus \mathcal{K}_l$
22:    end while
23:    $\sigma_{bi}(k) \leftarrow$ Recover switching sequence$(\mathcal{K}_i, i = 1, ..., l)$
24:    $\Sigma_{bi} \leftarrow \{\theta_{|\Sigma|}, ..., \theta_l\}$
25:    return $(\sigma_{bi}(k), \Sigma_{bi})$
26: end procedure
```

Bibliography

[1] N. Ozay, M. Sznaier, C. M. Lagoa, and O. I. Camps, “A sparsification approach to set membership identification of switched affine systems,” *IEEE Transactions on Automatic Control*, vol. 57, pp. 634–648, March 2012.

[2] T. Wigren and M. Schoukens, *Coupled electric drives data set and reference models.* No. 024 in Technical Report Uppsala Universitet, Uppsala University Sweden, 11 2017.

[3] J. . Lin and R. Unbehauen, “Canonical piecewise-linear approximations,” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 39, pp. 697–699, Aug 1992.

[4] A. Garulli, S. Paoletti, and A. Vicino, “A survey on switched and piecewise affine system identification,” *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 344 – 355, 2012. 16th IFAC Symposium on System Identification.

[5] R. Vidal, S. Soatto, Yi Ma, and S. Sastry, “An algebraic geometric approach to the identification of a class of linear hybrid systems,” in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 1, pp. 167–172 Vol.1, Dec 2003.

[6] Y. Ma and R. Vidal, “Identification of deterministic switched arx systems via identification of algebraic varieties,” in *Hybrid Systems: Computation and Control* (M. Morari and L. Thiele, eds.), (Berlin, Heidelberg), pp. 449–465, Springer Berlin Heidelberg, 2005.

[7] J. Roll, A. Bemporad, and L. Ljung, “Identification of piecewise affine systems via mixed-integer programming,” *Automatica*, vol. 40, pp. 37–50, Jan. 2004.

[8] A. Juloski, S. Weiland, and W. M. Heemels, “A bayesian approach to identification of hybrid systems,” vol. 50, pp. 13 – 19 Vol.1, 01 2005.

[9] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, “A clustering technique for the identification of piecewise affine systems,” *Automatica*, vol. 39, no. 2, pp. 205 – 217, 2003.

[10] Z. Lassoued and K. Abderrahim, “New approaches to identification of pwarx systems,” *Mathematical Problems in Engineering*, vol. 2013, 11 2013.

[11] H. Ohlsson, L. Ljung, and S. Boyd, “Segmentation of arx-models using sum-of-norms regularization,” *Automatica*, vol. 46, no. 6, pp. 1107 – 1111, 2010.

[12] L. Bako, “Identification of switched linear systems via sparse optimization,” *Automatica*, vol. 47, no. 4, pp. 668 – 677, 2011.

[13] H. Ohlsson and L. Ljung, “Identification of switched linear regression models using sum-of-norms regularization,” *Automatica*, vol. 49, no. 4, pp. 1045 – 1050, 2013.

[14] D. Piga and R. Tóth, “An sdp approach for l0-minimization: Application to arx model segmentation,” *Automatica*, vol. 49, no. 12, pp. 3646 – 3653, 2013.

[15] A. Hartmann, J. Lemos, R. S. Costa, J. Xavier, and S. Vinga, “Identification of switched arx models via convex optimization and expectation maximization,” *Journal of Process Control*, vol. 28, pp. 9–16, 04 2015.

[16] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino, “A bounded-error approach to piecewise affine system identification,” *IEEE Transactions on Automatic Control*, vol. 50, pp. 1567–1580, Oct 2005.

[17] J. Zwart, “Literature survey: Identification of hybrid systems via sat solving techniques,” Master’s thesis, Delft University of Technology, 2019.

[18] J. Zwart, “Hysi-sat toolbox.” `https://github.com/joostzwart/ThesisSARXSAT`, 2019.

[19] S. A. Cook, “The complexity of theorem-proving procedures,” *STOC ’71 Proceedings of the third annual ACM symposium on Theory of computing*, pp. 151–158.

[20] L. A. Levin, “Universal sequential search problems,” *Problems of Information Transmission*, vol. 9, no. 3, pp. 265–266, 1973.

[21] D. Déharbe and S. Ranise, “Satisfiability solving for software verification,” *International Journal on Software Tools for Technology Transfer*, vol. 11, pp. 255–260, Jul 2009.

[22] A. Gupta, M. K. Ganai, and C. Wang, “Sat-based verification methods and applications in hardware verification,” in *Formal Methods for Hardware Verification* (M. Bernardo and A. Cimatti, eds.), (Berlin, Heidelberg), pp. 108–143, Springer Berlin Heidelberg, 2006.

[23] H. Kautz and B. Selman, “Planning as satisfiability,” in *Proceedings of the 10th European Conference on Artificial Intelligence*, ECAI ’92, (New York, NY, USA), pp. 359–363, John Wiley & Sons, Inc., 1992.

[24] M. Davis and H. Putnam, “A computing procedure for quantification theory,” *J. ACM*, vol. 7, pp. 201–215, July 1960.

[25] M. Davis, G. Logemann, and D. Loveland, “A machine program for theorem-proving,” *Commun. ACM*, vol. 5, pp. 394–397, July 1962.

[26] A. M. Frisch and P. A. Giannaros, “Sat encodings of the at-most-k constraint some old , some new , some fast , some slow,” 2010.

[27] C. Sinz, “Towards an optimal cnf encoding of boolean cardinality constraints,” in *Principles and Practice of Constraint Programming - CP 2005* (P. van Beek, ed.), (Berlin, Heidelberg), pp. 827–831, Springer Berlin Heidelberg, 2005.

[28] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, “Smc: Satisfiability modulo convex programming,” *Proceedings of the IEEE*, vol. 106, pp. 1655–1679, Sep. 2018.

[29] Y. Shoukry, M. Chong, M. Wakaiki, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, J. P. Hespanha, and P. Tabuada, “Smt-based observer design for cyber-physical systems under sensor attacks,” in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 1–10, April 2016.

[30] E. Amaldi and M. Mattavelli, “The min pfs problem and piecewise linear model estimation,” *Discrete Appl. Math.*, vol. 118, pp. 115–143, Apr. 2002.

[31] N. Chakravarti, “Some results concerning post-infeasibility analysis,” *European Journal of Operational Research*, vol. 73, pp. 139–143, February 1994.

[32] E. J. Candes and T. Tao, “Decoding by linear programming,” *IEEE Transactions on Information Theory*, vol. 51, pp. 4203–4215, Dec 2005.

[33] E. J. Candès, M. B. Wakin, and S. P. Boyd, “Enhancing sparsity by reweighted âĎŞ1 minimization,” *Journal of Fourier Analysis and Applications*, vol. 14, pp. 877–905, Dec 2008.

[34] Z. Lassoued and K. Abderrahim, “New results on pwarx model identification based on clustering approach,” *International Journal of Automation and Computing*, vol. 11, pp. 180–188, Apr 2014.

[35] R. Schneider, *Convex Bodies: The BrunnâĂŞMinkowski Theory*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2 ed., 2013.

[36] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, Sep 1995.

[37] S. Knerr, L. Personnaz, and G. Dreyfus, “Single-layer learning revisited: a stepwise procedure for building and training a neural network,” in *Neurocomputing* (F. F. Soulié and J. Hérault, eds.), (Berlin, Heidelberg), pp. 41–50, Springer Berlin Heidelberg, 1990.

[38] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[39] G. Ausiello, A. D’Atri, and M. Protasi, “Structure preserving reductions among convex optimization problems.,” *J. Comput. Syst. Sci.*, vol. 21, pp. 136–153, 08 1980.

[40] R. M. Karp, *Reducibility among Combinatorial Problems*, pp. 85–103. Boston, MA: Springer US, 1972.

[41] A. Fijany and F. Vatan, “New approaches for efficient solution of hitting set problem,” 2004.

[42] Microsoft Research, *Z3*. 4.8.0.0.

[43] IBM, *IBM ILOG CPLEX Optimization Studio Python API*. 12.8.0.

[44] L. Ljung, *System Identficiation Toolbox User's Guide*. Boston, MA: The MathWorks Inc., 9th ed., 2014.

[45] The Mathworks, Inc., Natick, Massachusetts, *MATLAB version 9.3.0.867777 (R2017b) Update 7*, 2017.

[46] H. Ohlsson, "Code segreg." http://users.isy.liu.se/en/rt/ohlsson/code.html. Accessed: 2019-08-07.

[47] A. Hartmann, "Son-em." https://github.com/AndrasHartmann/son-em, 2015. Accessed: 2019-06-06.

[48] H. Ohlsson and L. Ljung, "Identification of piecewise affine systems using sum-of-norms regularization," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6640 – 6645, 2011. 18th IFAC World Congress.

[49] P. E. Wellstead, "Introduction to physical system modelling," 1979.

[50] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *2013 European Control Conference (ECC)*, pp. 2933–2938, July 2013.

[51] H. V. H. Ayala, L. F. da Cruz, R. Z. Freire, and L. dos Santos Coelho, "Cascaded free search differential evolution applied to nonlinear system identification based on correlation functions and neural networks," in *2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, pp. 1–7, Dec 2014.

[52] F. Sabahi and M.-R. Akbarzadeh-T, "Extended fuzzy logic: Sets and systems," *IEEE Transactions on Fuzzy Systems*, vol. 24, pp. 1–1, 01 2015.

[53] D. Aleksovski, D. Dovzan, S. Dzeroski, and J. Kocijan, "A comparison of fuzzy identification methods on benchmark datasets," *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 31 – 36, 2016. 4th IFAC Conference on Intelligent Control and Automation SciencesICONS 2016.

[54] O. Nelles, A. Fink, and R. Isermann, "Local linear model trees (lolimot) toolbox for nonlinear system identification," *IFAC Proceedings Volumes*, vol. 33, no. 15, pp. 845 – 850, 2000. 12th IFAC Symposium on System Identification (SYSID 2000), Santa Barbara, CA, USA, 21-23 June 2000.

[55] H. Ohlsson, "Code pwason." http://users.isy.liu.se/en/rt/ohlsson/code.html. Accessed: 2019-02-10.

[56] A. Nakabayashi, S. Ukai, H. Wada, and T. Ohtani, "A bayesian robust identification method for piecewise affine autoregressive exogenous model from outlier-contaminated data," *Proceedings of the SICE Annual Conference*, pp. 511–518, 01 2013.

[57] V. Le, *Hybrid dynamical system identification : geometry, sparsity and nonlinearities*. PhD thesis, 10 2013.

Glossary

List of Acronyms

SARX	Switched AutoRegressive eXogenous
PWARX	PieceWise AutoRegressive eXogenous
SAT	Boolean Satisfiablity Problem
SMT	Satisfiability Modulo Theories
CNF	Conjunctive Normal Form
DPLL	Davis - Putnam - Logemann - Loveland
MIMO	Multiple Input Multiple Output
SVM	Support Vector Machine
OVR	One-versus-rest
OVO	One-versus-one
GUI	Graphical User Interface

List of Symbols

μ	Valuation of Booleans
τ_d	Dwell time
δ	Bound on the error
ω	Noise
Σ	Set of submodels
σ	Switching sequence
θ	Parameter vector

φ_{dwell}	Boolean dwell time constraint
$\varphi_{switchinglimit}$	Boolean switching limit constraint
$\boldsymbol{w}_i$	Weights of the decision function
$\mathcal{B}_i$	Block
$\mathcal{I}_i$	Interval
$\mathcal{N}$	Normal distribution
$\mathcal{R}_i$	Region of the regressorspace
$\mathcal{U}$	Uniform distribution
$\mathbb{B}$	Set of Boolean numbers
$\mathbb{R}$	Set of real numbers
$\ominus$	Minkowski difference
φ_b	Propositional formula
φ_{SMT}	SMT formula
$\vee$	Or operator
$\wedge$	And operator
b_i	Boolean
H_i	Coefficient matrix of a hyperplane
L_s	Switching limit
t_s^i	Time instance of a switch
$\mathcal{I}^*$	Interval without smallest element
C_i	Clause in a CNF formula
k	Time instance
n_u	Order of the input
n_y	Order of the output
r	Regressor
S	Switching time sequence
T	Length of the dataset
u	Input
V	Volts
y	Output

Index

B
block, 29

C
cardinality constraint, 10
certificate, 16
classifiers, 27
clause , 14

L
l_0-minimisation, 19
l_1-relaxation, 19

M
minimum dwell time, 8
minimum set cover problem, 30

N
notation, 3

P
PWARX models, 6

R
research goals, 9

S
SARX model, 5
SAT problem, 10
SAT solver, 10
SMT, 11
sparsity, 19
switching limit, 8
switching time sequence, 14