Algorithms for Identifying Flagged and Guarded Linear Systems

Guillaume O. Berger
UCLouvain
Belgium
guillaume.berger@colorado.edu

Monal Narasimhamurthy
University of Colorado Boulder
USA
monal.narasimhamurthy@colorado.edu

Sriram Sankaranarayanan
University of Colorado Boulder
USA
first.lastname@colorado.edu

ABSTRACT

We present an approach for identifying two subclasses of piecewise affine (PWA) systems that we call flagged and guarded linear systems. Flagged linear system dynamics are given by a sum of k linear dynamical modes, each activated based on a latent binary variable, called a flag. Additionally, guarded linear systems define each flag as the sign of an affine "guard" function. We term the discovery of the latent flag values and the corresponding linear dynamics as the "flagged regression" and "guarded regression" problems, respectively. We show that the system identification problem is NP-hard even for these models, making the identification problem computationally challenging. For both problems, we provide approximation algorithms that identify a model whose error is within some user-defined constant away from the optimum. The time complexity of these algorithms is linear in the number of data points but exponential in the state-space dimension and the number of flags. The linear complexity in data size allows our approach to potentially scale to large data sets. We evaluate our algorithms on benchmark problems in order to learn models for mechanical systems with contact forces and a nonlinear robotic arm benchmark. Our approach compares favorably against neural network learning and the PARC algorithm for identifying PWA models proposed by Bemporad.

ACM Reference Format:
Guillaume O. Berger, Monal Narasimhamurthy, and Sriram Sankaranarayanan. 2024. Algorithms for Identifying Flagged and Guarded Linear Systems. In *27th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '24), May 14–16, 2024, Hong Kong SAR, China.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3641513.3650140

1 INTRODUCTION

Piecewise affine systems (PWA) have been widely studied as a subclass of switched/hybrid systems models. Examples include mixed-logical dynamical (MLD) systems defined by Bemporad et al. [6] that are governed by a differential/difference equation whose right-hand sides involve discrete logic in the form of flags (Boolean variables) and constraints that relate the values of the flags to the system variables. These systems appear naturally in a wide range of applications including mechanical systems with impact, electrical circuits with switching and networked systems [6]. Identifying a system model from observed data enables applications ranging from control design to runtime monitoring. Furthermore, MLD systems are equivalent to piecewise affine (PWA) systems [4, 22].

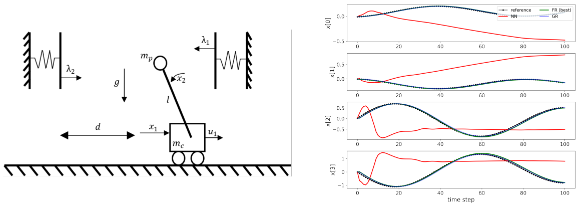


Figure 1: The cartpole with soft wall exhibits discontinuous PWA dynamics: the force applied on the pole depends discontinuously on its position. (Left) Cartpole schematics (from [2]). (Right) Our approach (green curve) yields more accurate predictions of future states when compared to a neural network model (red curve) trained on the same data (dots).

In this paper, we focus on two special classes of PWA/MLD systems, that we call *flagged linear systems* (FLS) and *guarded linear systems* (GLS). The dynamics are assumed to be a sum of k terms each involving linear dynamics that can be active or inactive, depending on the value of k latent flag variables: $\vec{x}(t+1) = A_0\vec{x}(t) + \sum_{j=1}^{k}[\text{flag}_j(t)]A_j\vec{x}(t)$. Note that $[\text{flag}_k(t)] \in \{-1, 1\}$. In FLS identification, our goal is to find the latent flags and the matrices $A_0, \ldots, A_k$. In GLS identification, the flag is not exogenous but is determined by an affine function called a guard, activating the mode or not depending on the sign of the guard at the state. A system with k flags has up to 2^k modes.

Example 1.1 (Cartpole with Soft Walls). Fig. 1 shows a cartpole system with two soft walls taken from Aydinoglu et al. [2]. Such a system is described as a PWA system with three modes in continous time that describe contacts with the two walls and the contactless conditions. The state $\vec{x}(t) = (x, \theta, v, \omega)$ represents the position, angle, linear and angular velocities, respectively. Using data obtained from simulating this continuous-time model, our approach identifies a discrete time GLS model with time step $h = 0.01$.

$$\begin{aligned}
x' &= x + 0.01v, \quad \theta' = \theta + 0.01\omega \\
v' &= (3.99x + 1.85\theta + 0.42v + 2.16\omega - 0.69) + \\
&\quad \text{sign}(g_1)\ (-4.72x - 2.13\theta + 0.61v - 2.22\omega + 0.64) + \\
&\quad \text{sign}(g_2)\ (0.49x - 0.3\theta + 0.01\omega + 0.05) \\
\omega' &= (1.82x + 0.44\theta - 0.23v + 1.9\omega - 0.31) + \\
&\quad \text{sign}(g_1)(-2.12x - 0.96\theta + 0.27v - \omega + 0.29) + \\
&\quad \text{sign}(g_2)(0.22x - 0.13\theta + 0.01\omega + 0.02) \\
g_1(\vec{x}) &= -x + \theta + 0.11v - 0.48\omega + 1 \\
g_2(\vec{x}) &= 0.51x - \theta + 0.02v + 0.12\omega + 0.11
\end{aligned}$$

HSCC '24, May 14–16, 2024, Hong Kong SAR, China

ACM ISBN 979-8-4007-0522-9/24/05
https://doi.org/10.1145/3641513.3650140

Here $\vec{x}(t+h) = (x', \theta', v', \omega')$. The identified model accurately predicts the test data 100 steps into the future; see Fig. 1.

Identifying hybrid linear models (including switched and piecewise linear models) is NP-hard [26]. In particular, it does not scale well with the number of data points. A straightforward encoding into MILP or SMT problem has a complexity 2^N in terms of the number of data points N. Lauer et al. have shown how to reduce this to $N^{O(nm)}$ wherein n is the dimension of the state space and m is the number of modes with a careful analysis of the number of data points needed to fix a model [25].

The main contribution of this paper is to provide an algorithm for FLS and GLS identification that scales linearly with the number of data points, but exponentially in the dimension and number of flags. This enables interesting applications to data sets of very large size but relatively low dimensions ~ 6 and number of flags ~ 3.

Our algorithm falls into the category of approximation algorithms because it does not solve the FLS/GLS identification problem exactly, but rather a relaxed version of it. More precisely, if the best fitting error achievable with a FLS or GLS model with k modes is ϵ^*, then the algorithm computes a model with error $\epsilon^* + \epsilon_{\text{gap}}$ in time linear in the number of data points and polynomial in $1/\epsilon_{\text{gap}}$.

Our algorithm is counterexample-guided and uses the principle of *cutting-planes* in convex optimization [11]. Namely, it uses subsets of the data to identify potential models and then uses the points that are not fitted by the potential models, called counterexamples, to create new subsets and identify new models, until a suitable one is found. The counterexamples have the effect of adding new constraints to the space of potential models, thereby enabling the cutting-plane principle. This bounds the size of the subsets to consider, from which we derive an upper bound on the number of these subsets, that is *independent of the number of data points.* The contributions of this work are as follows:

- Define the FLS and GLS models, compare against other PWA models, and the computational complexity of their identification problem and its MILP formulation (Sec. 2).
- Define a approximation version of these problems, and provide an algorithm with complexity linear in the number of data points and polynomial in the accuracy gap (Sec. 3).
- Demonstrate the applicability and performance of our algorithm over a set of synthetic and application benchmarks, and compare it with the MILP formulation and other approaches. The MILP formulation fails to handle more than a few hundred data points, while our algorithm scales up to ten thousand data points, for similar fitting and prediction accuracy. Other more scalable approaches, such as neural networks, fail to learn a precise model (Sec. 4).

One limitation of our algorithm is the exponential dependence of the complexity on the number of flags k and dimension n of the state space. From the NP-hardness of the FLS/GLS identification problems (proved in Sec. 2.4.3), this dependence is to be expected, unless P = NP. Showing that this dependence is unavoidable also for the relaxed version is a direction for future work.

1.1 Related Work

There are several techniques available in the literature for identifying switched linear and piecewise linear models. We refer the reader to [26, 31] for surveys. In particular, Vidal et al. [38] propose an algebraic approach that utilizes polynomial factorization to identify a model in the absence of noise. However, it is important to work with approximate fits in the presence of noise. Mixed-integer linear programming (MILP) is a popular approach for solving the identification problem. Roll et al. [34] and Sadraddini and Belta [35] propose MILP formulations for identifying piecewise linear models. The Hinging Hyperplane models introduced in [13] and studied in [34] are a special case of GLS models, but they are less expressive, since the guard is related to the dynamics of each mode. The models studied in [35] have pieces defined by k hyperplanes. Our GLS model is similar to that one, except that the linear dynamics on the 2^k pieces are not independent since they are the weighted superposition of k linear dynamics. An important limitation of MILP formulations is the exponential dependence on the number of data points. As mentioned earlier, Lauer et al. propose an approach that is polynomial time in the number of points [25]. However, the degree of the polynomial depends on the dimension of the state space. Our approach, by contrast, has a linear time complexity in the size of the data. Algorithms for time series segmentation [23, 30] also have polynomial time-dependence on the number of data points but they are restricted to one-dimensional inputs.

Our work is closely related to the recent work of Berger et al. [8] and can be regarded as an extension of that work to the problems of flagged and guarded regression. Therein, a counterexample-guided algorithm was proposed to identify switched linear models, with complexity linear in the number of data points. The present work differs in many ways: (a) Berger et al. focus on identifying the latent modes and dynamics of a switched linear system, we solve for a switched system whose structure is specified through flags; (b) Berger et al. obtain an exponential complexity in the number of *modes*, we obtain a complexity bound that is exponentially faster in terms of the number of modes[1]; (c) we identify GLS models, an extension that is not possible using Berger et al.'s algorithm. Our approach is loosely related to numerous counterexample-guided approaches that have been considered in formal methods and control theory to prove properties of programs, synthesize programs, compute Lyapunov functions and invariant sets [9, 10, 14–16, 33, 36].

Inexact algorithms for switched and piecewise linear regression include clustering [24, 29], continuous optimization [21, 27, 28], and others [3, 5, 18]. However, these algorithms offer no guarantees on the quality of the solution, for instance, they can be stuck in a local minimum with fitting error arbitrarily larger than the optimal one. By contrast, our approach provides guarantees on the gap between the optimal accuracy and that of the returned solution. We compare our approach with some of these methods on a set of interesting application examples in Sec. 4.

Approximation algorithms have a long history in theoretical computer science and applied mathematics. We refer the reader to [37] for a survey. These algorithms aim at solving computationally challenging problems in a reasonable time while providing a guarantee on the performance of the solution compared to the optimum. This paper investigates a unique approximation algorithm for FLS and GLS identification.

[1] m^3 vs. $m^{O(m)}$ where m is the total number of modes.

2 PROBLEM STATEMENT

Notation. For $n \in \mathbb{N}$, let $[n] \coloneqq \{1, \ldots, n\}$. For $\vec{x} = (x_1, \ldots, x_n)$, let $\|\vec{x}\| \coloneqq \max_{i \in [n]} |x_i|$ be its $\mathcal{L}_\infty$ norm. For $A = [a_{ij}]_{i=1,j=1}^{m,n}$, let $\|A\| \coloneqq \max_{i \in [m]} \sum_{j=1}^{n} |a_{ij}|$ be its induced $\mathcal{L}_\infty$ norm.

Given a data set $\mathcal{D}$ consisting of N data points $(\vec{x}(t), \vec{y}(t))$ wherein $\vec{x}(t) \in \mathbb{R}^n$ and $\vec{y}(t) \in \mathbb{R}^m$ for $t \in [N]$, we seek a model $\vec{y}(t) \approx F(\vec{x}(t), \vec{z}(t))$ for latent variables $\vec{z}(t) \in Z$ that fits the data set with error tolerances (ϵ, τ) defined as follows.

Definition 2.1 (Data Fit). A model $F(\vec{x}, \vec{z})$ is said to *fit* the data set $\mathcal{D}$ with relative error tolerance $\epsilon \geq 0$ and absolute error tolerance $\tau \geq 0$ if there exist values of $\vec{z}(t) \in Z$ for $t \in [N]$ such that

$$\|\vec{y}(t) - F(\vec{x}(t), \vec{z}(t))\| \leq \epsilon \|\vec{x}(t)\| + \tau, \quad \forall\, t \in [N]. \tag{1}$$

Since $\|\cdot\|$ is the $\mathcal{L}_\infty$ norm, (1) is equivalent to mN scalar inequalities:

$$|y_i(t) - F_i(\vec{x}(t), \vec{z}(t))| \leq \epsilon \|\vec{x}(t)\| + \tau, \quad \forall\, i \in [m],\ \forall\, t \in [N].$$

We study two types of regression problems: (a) *flagged regression* for a linear model involving discrete latent *flags* and (b) *guarded regression* for mixed-logical dynamic systems in discrete time.

2.1 Flagged Regression Problem

For this problem, the latent variable $\vec{z}$ is a vector of "flags" with values in $\{-1, 1\}$, i.e., $\vec{z}(t) = (q_1(t), \ldots, q_k(t))$ for $q_i(t) \in \{-1, 1\}$.

Definition 2.2 (Flagged Regression Problem). Given a data set $\mathcal{D}$, norm-bound $\gamma > 0$ and error tolerances ϵ and τ, the flagged regression problem with k flags seeks a set of matrices $A_0, \ldots, A_k$ and a series of flags' values $\vec{z}(t) = (q_1(t), \ldots, q_k(t)) \in \{-1, 1\}^k$ for $t \in [N]$ such that the data points are fitted by the linear model:

$$\|\vec{y}(t) - A(\vec{z}(t))\vec{x}(t)\| \leq \epsilon \|\vec{x}(t)\| + \tau, \quad \forall\, t \in [N],$$

wherein $A(\vec{z}) = A_0 + \sum_{i=1}^{k} q_i A_i$ for $\vec{z} = (q_1, \ldots, q_k)$ and $\|A_i\| \leq \gamma$.

Remark 1. In general, the flags can take values in any set $\{\alpha, \beta\}$ for $\alpha \neq \beta$. A set of matrices $A_0, \ldots, A_k$ that fits the data for flags $q_1(t), \ldots, q_k(t) \in \{-1, 1\}^k$ can be transformed into another set of matrices $B_0, \ldots, B_k$ that fits the data with flags $r_1(t), \ldots, r_k(t) \in \{\alpha, \beta\}^k$ for $\alpha \neq \beta$: $r_i(t) = \frac{\beta - \alpha}{2} q_i(t) + \frac{\beta + \alpha}{2}$, $B_0 = A_0 - \frac{\beta + \alpha}{\beta - \alpha} \sum_{i=1}^{k} A_i$, and $B_i = \frac{2}{\beta - \alpha} A_i$ for $i = 1, \ldots, k$.

Remark 2. The bound on the norms of the matrices (γ) is needed to guarantee termination of our approach. However, setting γ in practice is problem dependent, based partly on the possible range of the outputs and those of each inputs to the model. To alleviate this, we may apply a "scaling" transformation to the "raw" data $(\hat{\vec{x}}(t), \hat{\vec{y}}(t))$ wherein we scale each $x_i(t) = \lambda_i \hat{x}_i(t)$, $y_j(t) = \pi_j \hat{y}_j(t)$ for some scaling factors $\lambda_i, \pi_j > 0$ to ensure that $||\vec{x}(t)||, ||\vec{y}(t)|| \leq 1$. Such a scaling is a common practice in machine learning. In general, a FLS model for the raw data can be transformed into one for the scaled data and vice-versa. We can set γ to a fixed but large value $\gamma_{\max} \sim 1000$. Our approach will have running time that is polynomial in γ in the worst case.

Remark 3. For technical reasons, we assume that *no* data points in $\mathcal{D}$ have $\vec{x}(t) = 0$ and $\|\vec{y}(t)\| > \tau$. Therefore, for a fixed τ, there exists a *minimal* $\epsilon^* \geq 0$ such that the flagged regression problem with k flags and error tolerances ϵ^* and τ has a solution.

2.2 Guarded Regression Problem

The guarded regression problem seeks a model of the form $\vec{y}(t) \approx A(\vec{x}(t))\vec{x}(t)$, wherein $A(\vec{x}) = A_0 + \sum_{i=1}^{k} \mathsf{sign}(\vec{c}_i^\top \vec{x}) A_i$. In contrast to flagged regression, the flags are the indicator variable of a linear inequality of the form $\vec{c}_i^\top \vec{x} \geq 0$, with value in $\{-1, 1\}$.

Definition 2.3 (Guarded Regression). Given a data set $\mathcal{D}$, norm-bound γ and error tolerances ϵ and τ, the guarded regression problem with k guards seeks a set of matrices $A_0, \ldots, A_k$ with $\|A_j\| \leq \gamma$ and nonzero coefficients $\vec{c}_1, \ldots, \vec{c}_k \in \mathbb{R}^n$ such that:

$$\|\vec{y}(t) - A(\vec{x}(t))\vec{x}(t)\| \leq \epsilon \|\vec{x}(t)\| + \tau, \quad \forall\, t \in [N],$$

wherein $A(\vec{x}) = A_0 + \sum_{i=1}^{k} \mathsf{sign}(\vec{c}_i^\top \vec{x}) A_i$.[2]

2.3 Expressivity of Flagged and Guarded Linear System Models

FLS are as expressive as linear switched systems. Any FLS with k flags can be converted into a linear switched system with 2^k modes. At the same time, given a linear switched system with modes defined by matrices $B_1, \ldots, B_k$, we define matrices $A_0 = \frac{1}{2} \sum_{i=1}^{k} B_i$ and $A_i = \frac{1}{2} B_i$ for $i \in [k]$. It holds that $B_i = A(\vec{z})$ where $\vec{z}$ is the vector of flags $(q_1, \ldots, q_k)$ with $q_j = 1$ if $j = i$ and -1 otherwise. Note that to model the switched system, we need to add the extraneous constraint $\sum_{j=1}^{k} q_j = 2 - k$ over the flags.

We will now turn to the issue of expressivity of Guarded Linear Systems (GLS). Hinging-Hyperplane systems [13], are models where the next state for each component x_j can be written as

$$x_j(t + h) = f_{j,0}(\vec{x}) + \sum_{i=1}^{k} \sigma_{j,i} \max(f_{j,i}(\vec{x}), g_{j,i}(\vec{x})),$$

wherein $f_{j,i}$ and $g_{j,i}$ are affine functions from $\mathbb{R}^n$ to $\mathbb{R}$, and $\sigma_{j,i} \in \{-1, 1\}$. They are known to be useful for approximating a wide variety of nonlinear functions.

LEMMA 2.4. *Any Hinging-Hyperplane system can be written as a GLS. However, there are GLS that cannot be expressed as Hinging-Hyperplane systems.*

PROOF. Proof is by observing that we can write each hinge function as $\max(f, g) = \frac{1}{2}(f + g) + \frac{1}{2}\mathsf{sign}(f - g)(f - g)$. The second part simply notes that all Hinging-Hyperplane systems are continuous PWA functions whereas GLS can be discontinuous. □

Similarly, it is easy to show that any one-dimensional PWA system can be expressed as a GLS. However, GLS are not as expressive as mixed-logical dynamical systems or general PWA systems.

LEMMA 2.5. *The function $f(x, y) = x + y$ if $x \geq 0$, $y \geq 0$ and 0 everywhere else, cannot be written as a GLS $f_0 + \sum_{i=1}^{k} \mathsf{sign}(g_i) f_i$ for any linear functions $f_0, \ldots, f_k$ and $g_1, \ldots, g_k$.*

Proof is provided in Appendix A.

Remark 4. Our approach extends to *affine* models (and *affine* guards) by augmenting the state vector $\vec{x}$ with a 1 [8, Remark 1]. Nonlinear models that are linear combinations of a fixed set of known basis functions can also be learned using our approach.

[2] For definiteness, we let $\mathsf{sign}(0) = 1$.

2.4 Computational Complexity and MILP Formulations

The flagged regression and the guarded regression problems can be formulated and solved as optimization problems. In this section, we first explain how to formulate these problems as mixed-integer linear programs (MILP), and discuss the computational complexity of solving them in this way. Then, we present theoretical bounds on the computational complexity of the problems.

2.4.1 MILP Formulation of Flagged Regression. Given an instance of the flagged regression problem (plus a bound γ on the matrices norm), the decision variables of the associated MILP are (1) $mn(k+1)$ continuous variables representing the entries of the $m \times n$ matrices $A_0, \ldots, A_k$; (2) kN binary variables encoding the decision between selecting -1 or 1 for each flag of each data point. Let $b_i(t) \in \{0, 1\}$ denote the binary variable corresponding to the flag $q_i(t)$ being 1 if $b_i(t) = 0$ and -1 if $b_i(t) = 1$; (3) kNm continuous variables, labeled $\vec{y}_1(t), \ldots, \vec{y}_k(t) \in \mathbb{R}^m$ for $t \in [N]$, serving as auxiliary variables.

The constraints are as follows: (1) For each $i \in [k]$, the norm of A_i is bounded by γ: $\|A_i\| \leq \gamma$. (2) For each $t \in [N]$ and $i \in [k]$, the value of $b_i(t)$ imposes constraints on the value of $q_i(t)$, and consequently on the value of the auxiliary variable $\vec{y}_i(t)$:

$$\|\vec{y}_i(t) - A_i\vec{x}(t)\| \leq b_i(t)M\|\vec{x}\|, \ \|\vec{y}_i(t) + A_i\vec{x}(t)\| \leq (1 - b_i(t))M\|\vec{x}\|.$$

Here, $M \gg \gamma$ is a sufficiently large constant, chosen so that $\|\vec{y}_i(t) \pm A_i\vec{x}(t)\| \leq M\|\vec{x}(t)\|$ holds trivially for all possible $A_i, \vec{x}(t), \vec{y}_i(t)$[3]. Thus, if $b_i(t) = 0$, $\vec{y}_i(t) = A_i\vec{x}(t)$ and if $b_i(t) = 1$, $\vec{y}_i(t) = -A_i\vec{x}(t)$. (3) For each $t \in [N]$, the fitting error is bounded:

$$\|\vec{y}(t) - A_0\vec{x}(t) - \sum_{i=1}^{k} \vec{y}_i(t)\| \leq \epsilon\|\vec{x}(t)\| + \tau.$$

The objective of the MILP can be set as a linear or convex piecewise linear function: e.g., minimize the overall residual norm $\sum_{t=1}^{N} \|\vec{y}(t) - A_0\vec{x}(t) - \sum_{i=1}^{k} \vec{y}_i(t)\|$. The complexity of MILP solvers (e.g., using branch-and-bound) is typically exponential in the number of binary variables (kN). As we will see in the numerical experiments (Sec. 4), this is a major limitation for applying the MILP approach to learn flagged regression models for real data sets.

2.4.2 MILP Formulation for Guarded Regression. The MILP formulation for the guarded regression problem includes further decision variables and constraints in addition to the ones in the MILP formulation for flagged regression (Sec. 2.4.1). The decision variables include kn additional continuous variables to represent the guard coefficients: $\vec{c}_1, \ldots, \vec{c}_k \in \mathbb{R}^n$. We will constrain $\|\vec{c}_i\| \leq 1$. Additionally, we add the constraints:

$$(\delta - Mb_i(t))\|\vec{x}(t)\| \ \leq \ \vec{c}_i^{\top}\vec{x}(t) \ \leq \ (M(1 - b_i(t)) - \delta)\|\vec{x}(t)\|,$$

wherein $0 < \delta \ll 1$ is an additional input parameter that represents a lower bound on the margin of separation of the guards, and M is a big enough such that $(\delta - M)\|\vec{x}(t)\| \leq \vec{c}_i^{\top}\vec{x}(t) \leq (M - \delta)\|\vec{x}(t)\|$ holds for all $t \in [N]$ (when $\|\vec{c}_i\| \leq 1$). Note that when $b_i(t) = 0$, $\vec{c}_i^{\top}\vec{x}(t) \geq \delta\|\vec{x}(t)\|$ and when $b_i(t) = 1$, $\vec{c}_i^{\top}\vec{x}(t) \leq -\delta\|\vec{x}(t)\|$. This implies that for all $t \in [N]$, $\vec{x}(t)$ satisfies $|\vec{c}_i^{\top}\vec{x}(t)| \geq \delta\|\vec{c}_i^{\top}\|\|\vec{x}(t)\|$. Taking δ ensures that the condition is not restrictive.

Here again, the complexity of solving the MILP is typically exponential in the number of binary variables (kN), which is a major limitation for applying the MILP approach to learn guarded regression models for real data sets (see Sec. 4).

2.4.3 Complexity Bounds. We now present theoretical bounds on the computational complexity of the flagged regression and guarded regression problems using the NP-hardness of switched linear regression as a starting point [26].

Theorem 2.6. *The flagged regression problem and the guarded regression problem are both NP-hard.*

The proof is by reduction from the switched regression problem that is previously known to be NP-hard [26] and is provided in Appendix B. Despite being NP-hard, the problem of piecewise linear regression is known to have a complexity polynomial in the number of data points [26, Sec. 5.3.1]. The complexity is bounded by $O(N^{ns(s-1)/2})$ wherein s is the number of modes [26, Theorem 5.3]. However, this approach is impractical for large N. Note that a similar reasoning holds for the switched linear regression problem [26, Sec. 5.3.2].

2.5 Relaxed Problem Formulation

We introduce the following relaxed formulation of the flagged regression and guarded regression problems, using the idea of a "tolerance gap". The resulting problem formulation is called a *promise problem* or a *gap problem* [17, 19].

Definition 2.7 (Regression with Tolerance Gap). Given a data set $\mathcal{D}$, an absolute error tolerance τ, and two relative error tolerances $0 \leq \epsilon_1 < \epsilon_2$, the "gap" version of the flagged (guarded) regression problem seeks to decide between two alternatives:

(1) yes: There exists matrices (and guard coefficients) that fits the data with relative error $\epsilon \leq \epsilon_2$ and absolute error τ. Additionally, for this case the algorithm in this paper will find matrices that fit the data with relative error at most ϵ_2.
(2) no: All matrices (and guard coefficients) that fit the data with absolute error τ has relative tolerance $\epsilon > \epsilon_1$.

However, if the minimal relative error ϵ (for fixed absolute error tolerance τ) satisfies $\epsilon \in (\epsilon_1, \epsilon_2]$, then our algorithm can provide either yes or no answer, since they are both correct.

An equivalent way of expressing the guarantee of our algorithm is that for fixed absolute error tolerance τ, *if* there is a model that fits the data with relative error $\leq \epsilon_1$, our approach is guaranteed to find a model with relative error $\leq \epsilon_2$. In practice, our algorithm can be used in many ways by setting various values for ϵ_1, ϵ_2. (1) Setting $\epsilon_1 = 0, \epsilon_2 = \epsilon$ implies that our algorithm, upon a yes answer, yields a model with relative error bounded by ϵ. However, if the algorithm yields a no, we conclude that there is no model with relative error $\epsilon = 0$ and absolute error τ. This is analogous to techniques that find a local minimum, but do not establish a lower bound on the global optimum. (2) On the other hand, given a user-input bound ϵ_{gap}, our algorithm can be used repeatedly to find a model that fits the data to a relative error $\epsilon \leq \epsilon^* + 2\epsilon_{\text{gap}}$, wherein ϵ^* is the least relative error possible amongst all models whose absolute error tolerance is τ. First, we place an upper bound B on ϵ^* using linear

[3] This is the commonly known "big-M" trick in linear programming [40].

regression and then repeatedly call our approach at most $\log\left(\frac{B}{\epsilon_{\text{gap}}}\right)$ times. Appendix C presents this algorithm with a detailed analysis.

3 ALGORITHM

We will first start with the algorithm for flagged regression. The extension for the guarded regression problem will be outlined in Sec. 3.6. At the high level, the algorithm maintains a search tree wherein each node of the tree guesses the assignment of flags for a *subset* of the data points. Each iteration of the algorithm works by expanding a leaf node in the tree as follows:

(1) Solve least norm linear regression problem to identify a *candidate model* consisting of matrices $A_0, \ldots, A_k$ that satisfy the error tolerance bounds for just the subset of the data that has been assigned values for the latent flags.
(2) Test the candidate model against the remaining data points not yet assigned flag values. If the model fits these points, we have found our desired solution. Otherwise, we have a *counterexample* over which the candidate fails.
(3) For each possible assignment of latent flag values to the counterexample, we create a new child node that retains all the flag assignments from the parent node and additionally assigns flag values to the newly discovered counterexample.

Through an appropriate choice of the candidate model in step (1), we provide an upper bound on the depth of the tree constructed by our algorithm that is independent of the number of data points N.

We describe the tree structure, the computation of the candidate, and the validation of the candidate in Secs. 3.1–3.4 below. Then, we analyze the algorithm in Sec. 3.5.

3.1 Tree Structure

The algorithm constructs a tree data structure wherein each node in the tree stores a *flagged core*: a subset of the data with each data point in the subset being assigned values of the latent flags.

Definition 3.1 (Flagged Core Subset). A *flagged core* is a tuple (S, ϕ) wherein $S \subseteq [N]$ is a subset of time indices that selects a subset $\mathcal{D}' \subseteq \mathcal{D}$ of the input data and $\phi : S \to \{-1, 1\}^k$ is an assignment of latent flag values to each element of S.

Algo. 1 shows the overall algorithm. The initial tree is a root node containing the empty flagged core, i.e., with $S = \emptyset$ and $\phi = \emptyset$ (Line 1). The algorithm then picks an unexplored leaf node v from the tree (Line 3) and computes a candidate model for this node using the method FindCandidate (see Sec. 3.2) (Line 4). If no candidate can be found, the algorithm moves to another unexplored leaf node (Line 5). However, if a candidate has been found, then the algorithm tries to find a counterexample for the candidate using the method FindCounterexample (see Sec. 3.3) (Line 6). If no counterexample is found, then the algorithm stops and outputs the candidate (Line 7). Otherwise, the algorithm expands the node with children nodes using the counterexample (Line 8) and the method ExpandNode (see Sec. 3.4) . If there are no further unexplored leaf nodes in the tree, the algorithm returns infeasible (Line 9).

Algorithm 1: Flagged Regression Tree Search Algo.

```
Data: Data set $\mathcal{D}$, relative error tolerances $0 \leq \epsilon_1 < \epsilon_2$,
      absolute error tolerance $\tau$, bound $\gamma$.
Result: Either feasible and a set of matrices $A_0, \ldots, A_k$
        that fits $\mathcal{D}$ with error tolerances $\epsilon_2$ and $\tau$, or
        infeasible.
1 $T \leftarrow [\langle \emptyset, \emptyset \rangle]$        /* Initialize tree with root */
2 while $T$ is not empty do
3     $v \leftarrow$ Any unexplored node from $T$
4     $(A_0, \ldots, A_k) \leftarrow$ FindCandidate($v$)
5     if not exists($A_0, \ldots, A_k$) then continue
6     $c \leftarrow$ FindCounterexample($A_0, \ldots, A_k$)
7     if not exists($c$) then return $\langle$feasible$, A_0, \ldots, A_k\rangle$
8     $T \leftarrow$ ExpandNode($T, v, c$)
9 return infeasible
```

3.2 Finding a Candidate

Given a flagged core subset $\mathcal{D}'$ of the data, indexed by $S \subseteq [N]$, and an assignment map $\phi : S \to \{-1, 1\}^k$, we define a *candidate* as a model that fits $\mathcal{D}'$ with relative error tolerance ϵ_2, using the flag values given by ϕ, i.e., a set $A_0, \ldots, A_k$ satisfying

$$\left\|\vec{y}(t) - \left(A_0 + \sum_{i=1}^{k} q_i(t) A_i\right)\vec{x}(t)\right\| \leq \epsilon_2 \|\vec{x}(t)\| + \tau,\ \forall\, t \in S,$$

wherein $\phi(t) = (q_1(t), \ldots, q_k(t))$, plus the norm constraints $\|A_i\| \leq \gamma, \forall\, i \in [k] \cup \{0\}$. However, our algorithm requires us to select a candidate that is robust to bounded perturbations. To achieve this, we fix the notion of a θ-candidate.

Definition 3.2 (Set of θ-Candidates). Given $\theta > 0$ and node $v = (S, \phi)$, the *set of θ-candidates* at node v, denoted by $C(v, \theta)$, is defined as the set of all tuples of matrices $(A_0, \ldots, A_k)$ satisfying

$$\left\|\vec{y}(t) - \left(A_0 + \sum_{i=1}^{k} q_i(t) A_i\right)\vec{x}(t)\right\| \leq (\epsilon_2 - \theta)\|\vec{x}(t)\| + \tau,\ \ \forall\, t \in S,$$

and $\|A_i\| \leq \gamma - \frac{\theta}{k+1}, \quad \forall\, i \in [k] \cup \{0\}$.

When $\theta = 0$, we denote $C(v) = C(v, 0)$, and we simply refer to this as the set of candidates. The key observation is that θ candidates are robust to perturbations whose norm depends on θ.

Proposition 3.3. *Let $(A_0, \ldots, A_k) \in C(v, \theta)$ and let $\Delta_0, \ldots, \Delta_k$ be perturbation matrices with norm $\|\Delta_i\| \leq \frac{\theta}{k+1}, \forall\, i \in [k] \cup \{0\}$. It holds that $(A'_0, \ldots, A'_k)$ defined by $A'_i = A_i + \Delta_i$ belongs to $C(v)$.*

Proof. For each $t \in [S]$, we have

$$\|A'_i \vec{x}(t) - A_i \vec{x}(t)\| \leq \|A'_i - A_i\| \|\vec{x}(t)\| \leq \frac{\theta}{k+1} \|\vec{x}(t)\|\,.$$

Thus, $\|A'\vec{x}(t) - A\vec{x}(t)\| \leq \theta \|\vec{x}(t)\|$, wherein $A = A_0 + \sum_{i=1}^{k} q_i(t) A_i$ and $A' = A'_0 + \sum_{i=1}^{k} q_i(t) A'_i$. Since, $\|\vec{y}(t) - A\vec{x}(t)\| \leq (\epsilon_2 - \theta)\|\vec{x}(t)\| + \tau$, we obtain $\|\vec{y}(t) - A'\vec{x}(t)\| \leq \epsilon_2 \|\vec{x}(t)\| + \tau$. □

Furthermore, the assignment of the flag values makes the constraints on $A_0, \ldots, A_k$ linear and convex:

Proposition 3.4. *$C(v, \theta)$ is a a bounded convex polyhedron.*

Proof is simply by examining the constraints in Def. 3.2. The boundedness of $C(v, \theta)$ comes directly from the bound on the norms of the matrices A_i.

Algorithm 2: FindCandidate

```
Data: Data set $\mathcal{D}$, relative error tolerances $0 \leq \epsilon_1 < \epsilon_2$,
      absolute error tolerance $\tau$, matrix element bound $\gamma$,
      node $v = \langle S, \phi \rangle$.
if $C(v, \epsilon_2 - \epsilon_1) = \emptyset$ then return FAIL
return a central point in $C(v)$
```

Algorithm 3: FindCounterexample

```
Data: Data $\mathcal{D}$, errors $\epsilon_2$ and $\tau$, matrices $A_0, \ldots, A_k$.
for $t \in [N]$ do
    if no $(q_1, \ldots, q_k) \in \{-1, 1\}^k$ satisfies (2) then return $t$
return FAIL
```

The procedure FindCandidate is implemented in Algo. 2. The method first checks whether there is a set of matrices in $C(v, \theta)$ for $\theta = \epsilon_2 - \epsilon_1$. If this is not the case, then the method returns FAIL. Otherwise, the method selects a *central point* in $C(v)$. Different notions of central points can be considered, such as center of gravity, analytic center, center of Maximum Volume Inscribed Ellipsoid, Chebyshev center. The consequences of this choice are explained further in Sec. 3.5. The following corollary of Prop. 3.3 shows that the Lesbegue volume $\mathrm{vol}(C(v))$ of the set of candidates in a node will not be become too small while running Algo. 1. Let $\mathcal{V}(r) \subseteq \mathbb{R}^{m \times n}$ denote the volume of a unit ball of radius r of $m \times n$ matrices with respect to the induced $\mathcal{L}_\infty$ norm. Recall that $\mathcal{V}(r) = \frac{(2r)^{mn}}{n!^m}$; see, e.g., [8, Lemma 4].

Corollary 3.5. *Let $V_{\min} = \left(\mathcal{V}(\epsilon_{\text{gap}})\right)^{k+1}$, wherein $\epsilon_{\text{gap}} = \epsilon_2 - \epsilon_1$. If $\mathrm{vol}(C(v)) < V_{\min}$, then Algo. 2 will return* FAIL.

Proof. If Algo. 2 does not return FAIL, it means that $C(v, \epsilon_{\text{gap}}) \neq \emptyset$. Prop. 3.3, $C(v)$ contains the Cartesian product of $k + 1$ balls of radius at least $r = \frac{\epsilon_{\text{gap}}}{k+1}$, thereby providing the desired result. □

We clarify that our algorithm will not attempt to compute volume of polyhedra at any point since that can be quite expensive. Instead, Corr. 3.5 will be used to place a bound on the depth of the tree. The choice and the computation of the central point and the impact on the complexity of the algorithm will be discussed in Sec. 3.5.

3.3 Finding a Counterexample

Given a candidate model $A_0, \ldots, A_k$, we define a *counterexample* as any data point $(\vec{x}(t), \vec{y}(t))$ in $\mathcal{D}$ for which there are no flag values $q_1, \ldots, q_k$ satisfying

$$\left\| \vec{y}(t) - \left(A_0 + \sum_{i=1}^{k} q_i A_i \right) \vec{x}(t) \right\| \leq \epsilon_2 \|\vec{x}(t)\| + \tau. \tag{2}$$

An implementation of FindCounterexample is given in Algo. 3. The method returns FAIL if no counterexample exists.

Proposition 3.6. *For node $v = \langle S, \phi \rangle$ and $(A_0, \ldots, A_k) \in C(v)$, the result of* FindCounterexample *does not belong to S.*

Note that each element $t \in S$ has an assignment $\phi(t)$ that satisfies (2), and thus cannot be the result from FindCounterexample.

Algorithm 4: ExpandNode

```
Data: Tree $T$, node $v = \langle S, \phi \rangle \in T$, counterexample $c \notin S$.
for $(q_1, \ldots, q_k) \in \{-1, 1\}^k$ do
    Let $v' = \langle S \cup \{c\}, \phi \cup \{c \mapsto (q_1, \ldots, q_k)\} \rangle$
    Add $v'$ to $T$ as a child of $v$
return $T$
```

3.4 Expanding a Node with a Counterexample

Given a node $v = \langle S, \phi \rangle$ and an associated candidate $A_0, \ldots, A_k$, the existence of a counterexample c to the candidate reveals that the flagged core at v needs to be expanded with new data points. Our strategy is to use the counterexample c as the new data point. For that, we also need to choose the flag values $q_1(c), \ldots, q_k(c)$ associated with the new data point. In order to be exhaustive, and not miss any flagged core, we need to consider all possible flag values $q_1(c), \ldots, q_k(c)$ for c. This requires to create $K = 2^k$ new nodes, each of them with the same index set $S' = S \cup \{c\}$ but with a different map $\phi' : S' \to \{-1, 1\}^k$, accounting for the 2^k different ways of assigning the values of the k flags $q_1(c), \ldots, q_k(c)$ to c.

The implementation of ExpandNode is provided in Algo. 4. The following proposition shows that the set of candidates gets strictly smaller from a node to any of its children (Sec. 3.5 shows how to decrease the volume of the set of candidates).

Proposition 3.7. *For any node $v = \langle S, \phi \rangle$, let $(A_0, \ldots, A_k) \in C(v)$ be the result of* FindCandidate *on node v and $v' = \langle S', \phi' \rangle$ be any child of v through Algo. 4. $C(v') \subseteq C(v) \setminus \{(A_0, \ldots, A_k)\}$ holds.*

Proof. Since $S \subseteq S'$, $C(v')$ contains the same constraints on $A_0, \ldots, A_k$ as $C(v)$ plus additional constraints given by the counterexample c. Hence, $C(v') \subseteq C(v)$. Furthermore, by the choice of c (Algo. 3), it holds that $(A_0, \ldots, A_k) \notin C(v')$. □

The definition of the expansion makes that the tree exploration *exhaustive* (meaning that no feasible node is excluded) and *non-looping* (meaning that a node is never revisited or produced twice).

Proposition 3.8 (Exhaustive). *Let $v_* = \langle [N], \phi_* \rangle$ be a node such that $C(v_*, \epsilon_{\text{gap}}) \neq \emptyset$. Then, at the beginning of each iteration of Algo. 1 (before Line 3 is executed), there is an unexplored node $v = \langle S, \phi \rangle$ so that $\phi(t) = \phi_*(t)$ for all $t \in S$.*

Proof. The proof is by induction on the iterations of the algorithm. This is obviously true at the first iteration since the only unexplored node is the root. Now, for the induction step, assume that the hypothesis is satisfied at the beginning of some non-terminal iteration ITER. We show that it is still the case at the iteration ITER+1. Let $v = \langle S, \phi \rangle$ be an explored node at the beginning of the iteration ITER such that $\phi(t) = \phi_*(t)$ for all $t \in S$, and let v_{ITER} be the node picked during the iteration. If $v_{\text{ITER}} \neq v$, then v is still unexplored at the iteration ITER + 1 so that the property holds trivially at that iteration as well. Now, assume that $v_{\text{ITER}} = v$. Then, since $\phi(t) = \phi_*(t)$ for all $t \in S$, it holds that $C(v^*, \epsilon_{\text{gap}}) \subseteq C(v, \epsilon_{\text{gap}})$, so that $C(v, \epsilon_{\text{gap}}) \neq \emptyset$. Thus, FindCandidate does not fail and v gets expanded (because the iteration ITER is non-terminal). Let c be the counterexample used for the expansion. By definition of ExpandNode, there is a children node $v' = \langle S', \phi' \rangle$ of v such that

$S' = S \cup \{c\}$ and $\phi'(c) = \phi_*(c)$. This node is marked as unexplored, so that at the iteration ITER + 1, the property holds with v'. This concludes the proof by induction. □

Proposition 3.9 (Non-Looping). *For nodes $v_1 = \langle S_1, \phi_1 \rangle$ and $v_2 = \langle S_2, \phi_2 \rangle$ produced at different iterations of Algo. 1, $S_1 \neq S_2$.*

Proof. Let $v = \langle S, \phi \rangle$ be the least common ancestor of v_1 and v_2. If $v_1 = v$ (or $v_2 = v$), we note that $S_1 \subset S_2$ ($S_2 \subset S_1$). Otherwise, v_1 and v_2 descend from two distinct children $v'_1 = \langle S'_1, \phi'_1 \rangle$ and $v'_2 = \langle S'_2, \phi'_2 \rangle$ of v. Let c be the counterexample associated to v. By definition of the expansion, it then holds that $\phi'_1(x) \neq \phi'_2(c)$. Thus, since $\phi_i(c) = \phi'_i(c)$ for $i \in [2]$, it follows that $\phi_1(c) \neq \phi_2(c)$, so that $v_1 \neq v_2$, concluding the proof. □

3.5 Analysis of the Algorithm

We start by showing the soundness of the algorithm, then discuss the termination and complexity.

3.5.1 *Soundness.*

Soundness of the algorithm means that the output set of matrices must fit the data with relative error tolerance ϵ_2, and when the algorithm outputs INFEASIBLE, then no set of matrices (with bounded norm) can fit the data with relative error tolerance ϵ_1. This is shown in the following theorem:

Theorem 3.10. *Algo. 1 is sound: (1) If it outputs $\langle$FEASIBLE$, A_0, \ldots, A_k \rangle$, then $A_0, \ldots, A_k$ fits the data with error tolerances ϵ_2 and τ. (2) If it outputs INFEASIBLE, then no set of matrices $A_0, \ldots, A_k$ with bounded norm $\|A_i\| \leq \gamma - \frac{\epsilon_{\text{gap}}}{k+1}$ fits the data with error tolerances ϵ_1 and τ.*

Proof. If case 1) occurs, this means that a candidate $A_0, \ldots, A_k$ had no counterexample and the algorithm outputted $A_0, \ldots, A_k$. By definition of a counterexample, the non-existence of one means that $A_0, \ldots, A_k$ fits the data with error tolerances ϵ_2 and τ.

If case 2) occurs, this means that at the end of some iteration there was no unexplored node since this is the only way to reach Line 9. Assume, for a contradiction, that there is a set of matrices $A_0, \ldots, A_k$ satisfying the hypothesis in case 2). Then, there is $v_* = \langle [N], \phi_* \rangle$ such that $C(v_*, \epsilon_{\text{gap}}) \neq \emptyset$. This is a contradiction with Prop. 3.8, which guarantees that at the beginning of each iteration there is an unexplored node that either has no counterexample or is expanded (thereby creating more unexplored nodes). □

3.5.2 *Termination and Complexity.*

The termination of the algorithm is guaranteed by the non-looping property of the exploration process (Prop. 3.9), which guarantees that nodes never get visited twice. Since the set of possible nodes is finite (bounded by 2^{kN}), the algorithm terminates in finite time.

The upper bound 2^{kN} on the number of iterations is not satisfactory since it is exponential in the number of data points (it is in fact the same as for the MILP formulation). We will show that the number of iteration can in fact be upper bounded by $\kappa(n, m, k, \epsilon_{\text{gap}}, \gamma)$ for some κ depending on $n, m, k, \epsilon_{\text{gap}}, \gamma$ but not N. For that, we rely on an argument of volume contraction, inspired from *cutting-plane* or *localization* methods in convex optimization [11].

Volume Contraction. To explain the argument of volume contraction, let us start with a deeper analysis of the property in Prop. 3.7. This proposition implies that as we go deeper in the tree, the set of candidates gets smaller with respect to set inclusion. If we can show that the set of candidates actually gets a guaranteed *decrease in Lebesgue volume*, then we can combine this observation with the property in Corr. 3.5 (that a node is not expanded if the volume of the set of candidates is too small) to bound the depth of the tree, thereby obtaining a second bound on the number of iterations of the algorithm. This is formalized below:

Theorem 3.11. *Let $\alpha > 1$. Consider an execution of Algo. 1. Assume that whenever v and v' are two nodes in T such that v' is a child of v, it holds that $\text{vol}(C(v')) \leq \frac{1}{\alpha}\text{vol}(C(v))$. Then, the depth D of T satisfies $D \leq \left\lfloor (k+1)mn \log_\alpha \left(\frac{\gamma(k+1)}{\epsilon_{\text{gap}}} \right) \right\rfloor + 1$.*

Proof. Let $v_1, \ldots, v_p$ be a sequence of nodes in T such that v_{j+1} is a child of v_j. By definition of $C(v)$, it holds that $C(v_1) \subseteq \{(A_0, \ldots, A_k) : \|A_i\| \leq \gamma\}$. Hence,$\text{vol}(C(v_1)) \leq V_{\min} \left(\frac{\gamma(k+1)}{\epsilon_{\text{gap}}} \right)^{(k+1)mn}$.

Furthermore, by assumption, $\text{vol}(C(v_{p-1}) \leq \alpha^{1-p}\text{vol}(C(v_1))$. On the other side, since v_{p-1} got expanded, it holds by Corr. 3.5 that $\text{vol}(C(v_{p-1})) \geq V_{\min}$. Hence, $\alpha^{1-p} \geq \left(\frac{\epsilon_{\text{gap}}}{\gamma(k+1)}\right)^{(k+1)mn}$. Thus, $p - 1 \leq \left\lfloor (k+1)mn \log_\alpha \left(\frac{\gamma(k+1)}{\epsilon_{\text{gap}}} \right) \right\rfloor$. Since $v_1, \ldots, v_p$ was arbitrary, this gives the desired upper bound on the depth D of T. □

Remark 5. Note that it is the ratio $\gamma/\epsilon_{\text{gap}}$ that is relevant in the bound in Theorem 3.11, and not γ and ϵ_{gap} separately. This shows that the complexity depends on the desired *relative* precision on the matrix entries with respect to the assumed bound γ.

Theorem 3.12. *Under the assumption of Theorem 3.11, Algo. 1 explores at most $\frac{2^{kD}-1}{2^k-1}$ nodes and terminates in finite time.*

Proof. For a tree with max depth D and branching factor is $K = 2^k$, total number of nodes is $\sum_{j=0}^{D-1} K^j = \frac{K^D-1}{K-1}$. □

Guaranteed Volume Decrease. We now explain how to guarantee the relative volume decrease with factor $\alpha > 1$ required in Theorem 3.11. This can be guaranteed if we choose the candidate of each node carefully. For instance, if we select the candidate as the *center of gravity* of $C(v)$, then we can guarantee a volume decrease with factor $\frac{1}{\alpha} = 1 - \frac{1}{e} \approx 0.63$. Alternatively, if we select the candidate as the *center of the Maximum Volume Ellipsoid* inscribed in $C(v)$, then we can guarantee a volume decrease with factor $\frac{1}{\alpha} = 1 - \frac{1}{(k+1)mn}$.

Lemma 3.13. *[11, Sec. 4.2 and 4.3] Let $A, B \subseteq \mathbb{R}^d$ be two bounded convex sets such that $B \subseteq A$. If B does not contain the center of gravity of A, then $\text{vol}(B) \leq (1 - \frac{1}{e})\text{vol}(A)$. If B does not contain the center of the MVE inscribed in A, then $\text{vol}(B) \leq (1 - \frac{1}{d})\text{vol}(A)$.*

Note that in our case the sets $C(v)$ are bounded convex subsets of $(\mathbb{R}^{m \times n})^{k+1}$ by Prop. 3.7, and $C(v')$ is a convex subset of $C(v)$ that does not contain the candidate so that Lemma 3.13 applies.

We are now able to finish the complexity analysis of the algorithm. Under the above assumptions on the choice of the candidate, we first give an upper bound κ on the number of iterations of the algorithm, then we derive an upper bound on the computational complexity of the algorithm.

We start with the center of gravity as the choice of the candidate. By Theorem 3.12, the number of iterations is upper bounded by

$$\frac{2^{-k(k+1)mn\log_{0.63}\left(\frac{\gamma(k+1)}{\epsilon_{\mathrm{gap}}}\right)+1}-1}{2^k-1} \le 2^{O\left(k^2 mn\log\left(\frac{\gamma k}{\epsilon_{\mathrm{gap}}}\right)\right)}.$$

The complexity of computing the center of gravity of a polytope in $\mathbb{R}^d$ described by e linear constraints can be upper bounded by a function of d and e [32]. Since any data point in S for a node $v = \langle S, \phi \rangle$ adds a fixed number of linear constraints to $C(v)$, and since the size of S is bounded by the depth D, we obtain that the computation of the candidate has a complexity that depends only on k, n, m and D. In particular, the complexity of computing the candidate does not depend on N. Finally, finding a counterexample amounts to check all remaining data points and find those for which the candidate does not satisfy (2) for any flag values. This can be done in time linear in n, m, 2^k and N. Thus, the overall complexity of the algorithm is linear in N.

If the MVE center is used to choose the candidate, By Theorem 3.12, the number of iterations is upper bounded by[4]

$$\frac{2^{-k(k+1)mn\log_{1-\frac{1}{(k+1)mn}}\left(\frac{\gamma(k+1)}{\epsilon_{\mathrm{gap}}}\right)+1}-1}{2^k-1} \le 2^{O\left(k^3 m^2 n^2\log\left(\frac{\gamma k}{\epsilon_{\mathrm{gap}}}\right)\right)}.$$

The complexity of computing the MVE center of a polytope in $\mathbb{R}^d$ described by e linear constraints can be upper bounded by a polynomial function of d and e [7]. Thus, the complexity of computing the candidate is polynomial in k, n, m and D, and again does not depend on N, The computation of the counterexample is the same. In conclusion, the complexity of the algorithm is linear in N, and can be upper bounded by

$$2^{O\left(k^3 m^2 n^2\log\left(\frac{\gamma k}{\epsilon_{\mathrm{gap}}}\right)\right)}\mathrm{poly}\left(k, n, m, \log\left(\frac{\gamma}{\epsilon_{\mathrm{gap}}}\right)\right)N, \tag{3}$$

wherein poly is a polynomial function. Note that the factor m^2 in the exponent can be changed to m using an argument similar to the one in [8, Lemma 9]. However, for the sake of simplicity, this argument is not expanded here.

Note that (3) is an upper bound: to be tight, it would need that all nodes up to maximal depth D are explored. In practice, many nodes are deemed infeasible (and thus not expanded) way before reaching the maximal depth, so that the tree contains only a few deep branches. This implies that in practice the number of explored nodes and consequently the overall algorithmic complexity is way below the theoretical upper bound.

This concludes the presentation and analysis of the algorithm for the flagged regression problem. We now turn to guarded regression.

3.6 Algorithm for Guarded Regression

The modification of Algo. 1 to solve the guarded regression problem is rather straightforward. For each node $v = \langle S, \phi \rangle$, we try to find a candidate model consisting of a set of matrices $A_0, \ldots, A_k$ and guard coefficients $\vec{c}_1, \ldots, \vec{c}_k$ such that

$$\left\| \vec{y}(t) - \left(A_0 + \sum_{i=1}^{k} q_i(t) A_i \right) \vec{x}(t) \right\| \le (\epsilon_2 - \theta)\|\vec{x}(t)\| + \tau,\ \forall\, t \in S,$$

and $q_i(t)\, \vec{c}_i^\top \vec{x}(t) \ge \delta\|\vec{x}(t)\|$, $\forall\, i \in [k]$, $\forall\, t \in S$, wherein $\phi(t) = (q_1(t), \ldots, q_k(t))$, $\|A_i\| \le \gamma - \frac{\theta}{k+1}$, $\forall\, i \in [k] \cup \{0\}$, and $\|\vec{c}_i^\top\| \le 1$, $\forall\, i \in [k]$, for given parameters $\theta > 0$ and $\delta > 0$. Here, θ plays the same role as in Sec. 3.2, while δ bounds the minimal allowed angle between the points $\vec{x}(t)$ and any of the guard hyperplanes $H_i \doteq \{\vec{x} : \vec{c}_i^\top \vec{x} = 0\}$, since the conditions imply that $|\vec{c}_i^\top \vec{x}(t)| \ge \delta\|\vec{c}_i^\top\|\|\vec{x}(t)\|$. Analogously to the flagged regression case, we let $C(v, \theta, \delta)$ be the set of matrices $A_0, \ldots, A_k$ and guard coefficients $\vec{c}_1, \ldots, \vec{c}_k$ satisfying the above conditions. The computation of the candidate then amounts to check whether $C(v, \epsilon_{\mathrm{gap}}, \delta) \neq \emptyset$, and if this is the case, compute a central point in $C(v) = C(v, 0, 0)$.

The verification of a candidate $A_0, \ldots, A_k$ and $\vec{c}_1, \ldots, \vec{c}_k$ simply computes for all $t \in [N]$, $q_i = \vec{c}_i^\top \vec{x}(t)$, then checks whether (2) is satisfied. Any $t \in [N]$ for which this fails can be returned as a counterexample. The node expansion is the same as in Algo. 1.

Soundness. The algorithm is sound in the sense that if it returns a set of matrices and guard coefficients, those provide a valid solution to the guarded regression problem with error tolerances ϵ_2 and τ, while if the algorithm returns INFEASIBLE, this means that no set of matrices and guard coefficients solves the guarded regression problem with error tolerances ϵ_1 and τ, plus the additional constraints on the norm of the matrices (bounded by $\gamma - \frac{\theta}{k+1}$) and the minimal angle between the input points and the guard hyperplanes (bounded by δ) as discussed above.

Remark 6. The minimal angle condition can be alleviated by introducing a "gray region" in the determination of the flags from the guards, meaning that if $|\vec{c}_i^\top \vec{x}(t)| < \delta\|\vec{c}_i^\top\|\|\vec{x}(t)\|$, then $q_i(t)$ can be either -1 or 1. This formulation makes sense for instance if the points $\vec{x}(t)$ are corrupted by noise, and is solved by simply changing $C(v, \epsilon_{\mathrm{gap}}, \delta)$ to $C(v, \epsilon_{\mathrm{gap}}, 0^+)$ and $C(v)$ to $C(v, 0, -\delta)$ in the generation of the candidate.

Complexity. The complexity analysis follows the same reasoning as for the flagged regression algorithm. The only difference is that this time the unknown variables are $A_0, \ldots, A_k$ and $\vec{c}_1, \ldots, \vec{c}_k$ so that the volume decrease argument of $C(v)$ must be adapted accordingly. If we use the MVE center as the candidate, we obtain the following upper boun on the running time (poly is a polynomial function):

$$2^{O\left(k^3 mn^2\log\left(\frac{\gamma k}{\epsilon_{\mathrm{gap}}}\right)+k^2n^2\log\left(\frac{1}{\delta}\right)\right)}\mathrm{poly}\left(k, n, m, \log\left(\frac{\gamma}{\epsilon_{\mathrm{gap}}}\right), \log\ \delta\right)N\,.$$

Approximation of the MVE Center: The MVE center can be computed in polynomial time using for instance semidefinite programming [7]. Nevertheless, in practice, the computation can be cumbersome and subject to numerical instability. Therefore, in our numerical experiments, we used the *Chebyshev center* (center of a Maximum Volume Inscribed Ball), which can be computed efficiently using Linear Programming [12].

4 EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed approach (flagged and guarded regression), along with that of the reference MILP approach, on a set of mixed logical dynamical (MLD) system benchmarks. We compare our approach with two local optimization techniques: 1) Feedforward neural networks (NN), and 2) Piecewise Affine Regression and Classification tool (PARC) [3].

[4] We used that the facts that $\log_a(x) = \frac{\log(x)}{\log(a)}$ and $\log(1 - \frac{1}{r}) \le -\frac{1}{r}$.

Implementation. We assume that the number of flags k is an input to the algorithm. One can also systematically search for k to identify a regression model with the desired level of trade-off between the model complexity and data fit (refer to [31]). In all our experiments, we used $\epsilon_1 = 0$ and ϵ_2 as the desired bound on the model's relative error. See discussion in Section 2.5.

All experiments were conducted on a Linux server running Ubuntu 22.04 OS with 24 cores and 64 GB RAM. Both the MILP and the proposed approach were implemented in Python 3 as single-threaded programs. The LPs for estimating the Chebyshev centers of P in the proposed approach and the MILPs in the reference approach were encoded and solved using the Python interface of the Gurobi optimizer (version 10.0.3) [20].

4.0.1 *Micro-Benchmarks.*

We synthesized several micro-benchmarks with varying number of flags k, number of inputs n, and number of outputs m by randomly generating $k + 1$ matrices $A_0, \ldots, A_k$ from $\mathbb{R}^{n\times m}$. The matrices were generated by uniformly sampling each matrix entry from the interval $[-1, 1]$. For guarded regression, we additionally generated the guard coefficients $\vec{c}_1, \ldots, \vec{c}_k$ by uniformly sampling points on the unit sphere.

For each micro-benchmark, we generated N data points by uniformly sampling a random input point $\vec{x}(t) \in [-1, 1]^n$ and computing the output $\vec{y}(t)$ according to the synthesized matrices and the latent flag values (or guard coefficients). In the case of flagged regression, we picked the flag values $\vec{z}(t)$ uniformly at random from $\{-1, 1\}^k$. In guarded regression, the switching signal was directly determined by the synthesized guard coefficients. We also added uniform additive noise with amplitude τ to each $\vec{y}(t)$.

Comparison against MILP. We evaluated the timing performance of the MILP, flagged regression and guarded regression approaches with parameter values $\gamma = 2$, $\epsilon_2 = 0.1$, and $\tau = 0.05$. For guarded regression, we set $\delta = 0.02$. All the experiments were repeated 10 times and we computed the mean and standard deviation of the computation time of the approaches. Fig. 2 shows how the proposed approaches and the MILP approaches scale with the number of data points N, for a micro-benchmark with $n = m = 2$ and $k = 3$. We observed that the MILP approaches time out at $N \geq 20$ (for timeout values $\Delta_{\text{flagged}} = 90$ sec, $\Delta_{\text{guarded}} = 30$ sec). The proposed approaches, in contrast, exhibit a linear trend, consistent with the results presented in Sec. 3.5, as we scale N from 10 to 10000.

Fig. 3a shows how the proposed approaches and the MILP approaches scale as the dimension n is varied for a micro-benchmark with $m = n$ outputs, $k = 2$ flags, and $N = 100$ data points (parameter values same as above). Similarly, Fig. 3b shows how the approaches scale as the number of flags k is varied for a micro-benchmark with $n = m = 1$ and $N = 100$. These results show that i) the theoretical complexity guarantees bear out in practice and ii) our prototype outperforms a highly-optimized commercial MILP solver.

We now present an evaluation of the proposed approach on a set of mixed logical dynamical system identification benchmarks.

Cartpole with Soft Walls. The benchmark from Aydinoglu et al. [2] consists of a cartpole system moving on a frictionless track between two walls modeled as spring contacts. A controller balances the pole in the inverted position on the cart. The benchmark has four state variables, representing the position and velocity of both the cart and the pole. Hence, we set $n = 5$ (we augment the input $\vec{x}(t)$ with 1 for affine regression; see Rem. 4) and $m = 4$.

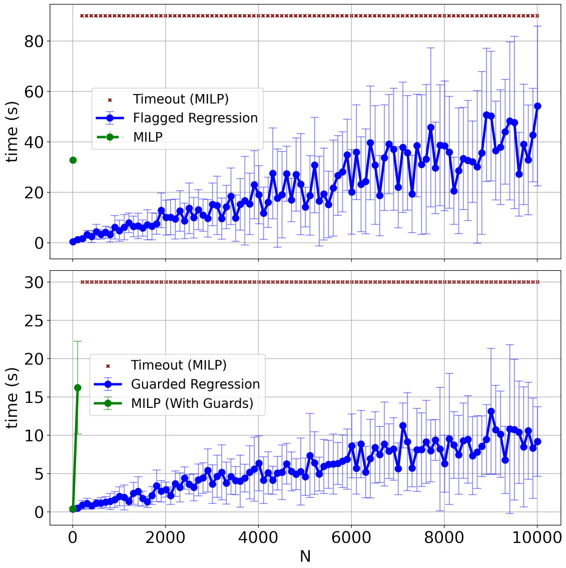
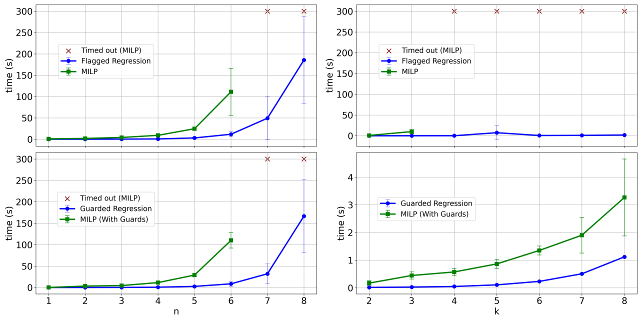


Figure 2: Timing comparison of MILP (green) with flagged regression (top) and guarded regression (bottom) approach on micro-benchmark with $n = m = 2$ and $k = 3$ as the number of data points N scales from 10 to 10000. The error bars report the average and standard deviation of the time taken across 10 experiments. The red crosses indicate timeouts.



(a) Scaling n **(b) Scaling k**

Figure 3: Timing comparison of MILP (green) and the proposed flagged regression (top) and guarded regression (bottom) approach as input/output size n, m and the number of flags k scale up. The red crosses indicate timeouts (300 sec).

Table 1 presents the performance of the proposed approaches: flagged regression (FR) and guarded regression (GR), along with that of a feedforward neural network (NN) and PARC on a held-out test data set. We sampled trajectories of length $T = 100$ time steps and created a data set with N data points. The relevant parameter values for FR/GR are $k = 4$, $\epsilon = 0.1$, $\tau = 0.0$, $\delta = 0.05$, and $\gamma = 100$. We ran the PARC algorithm with parameter values $K = 10$, $\alpha = 10^{-4}$, and maxiter = 15. We trained a feedforward neural network with 2 layers, each containing 32 nodes with ReLU activation using the Adam optimizer in Tensorflow [1] with a batch size of 32 for 100 training epochs. The MILP approach timed out after 100 sec.

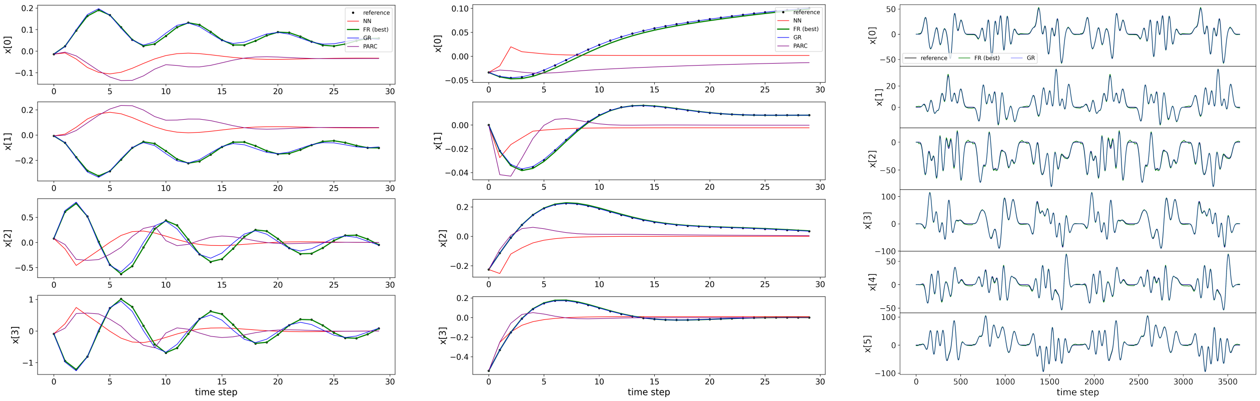


Figure 4: (Left) Simulation of the Acrobot with Soft Joint Limits (Left) and the Cartpole with Soft Walls (Middle), using flagged regression (blue), guarded regression (green), a feedforward neural network (red), and PARC (purple) on a test trajectory with a prediction horizon of 30 steps. (Right) Performance on the robotic arm benchmark.

Table 1: Performance of proposed approach (FR, GR) in comparison to the MILP, NN, PARC approaches on a test dataset (of size N) from the Acrobot and Cart-Pole benchmarks.

	N=200		N=400		N=800		N=1000	
Acrobot	R^2 score	t(s)	R^2 score	t(s)	R^2 score	t(s)	R^2 score	t(s)
NN	-0.75	1.90	0.74	2.84	0.87	5.17	0.89	6.9
PARC	-0.95	1.34	0.94	4.23	0.99	6.9	0.96	7.04
FR	0.99	2.25	0.99	7.6	0.99	8.41	0.99	11.5
GR	0.99	13.16	0.99	12.0	0.92	21.8	0.99	19.1
Cart-Pole	R^2 score	t(s)	R^2 score	t(s)	R^2 score	t(s)	R^2 score	t(s)
NN	0.72	1.61	0.79	2.95	0.89	3.81	0.90	6.0
PARC	-0.01	1.62	0.68	5.37	0.89	6.09	0.92	9.8
FR	0.93	4.28	0.92	3.68	0.99	11.62	0.97	18.4
GR	0.91	4.51	0.89	13.69	0.92	48.23	0.97	44.4

Fig. 4 shows the performance of the proposed algorithm on an (unseen) test trajectory. The identified model tracks the reference trajectory for a prediction horizon of 30 time steps. We also observe that the NN and PARC approaches rapidly diverge, underscoring the challenges associated with these approaches.

Acrobot with Soft Joint Limits. This benchmark from Aydinoglu et al. [2] features a double pendulum with an elbow actuator and soft joint limits. It has four state variables to represent the angles and velocities of the two links in the pendulum. Hence, for the proposed approaches, we set $n = 5$ and $m = 4$.

Table 1 presents the performance of the proposed approaches: flagged regression (FR) and guarded regression (GR), along with that of a feedforward neural network (NN) and PARC on a held-out test data set. We sampled trajectories of length $T = 100$ time steps and created a data set with N data points. The parameter values for FR/GR, PARC algorithm and the neural network training are the same as for the cartpole system. The MILP approach timed out after 100 sec with the same parameters. Fig. 4 shows the performance of the flagged regression, guarded regression, and a feedforward NN on an (unseen) test trajectory. The NN and PARC approaches rapidly diverge, whereas FR and GR track the reference trajectory.

Robotic Arm Benchmark. This nonlinear system identification benchmark from Weigand et al. [39] contains measurement data from a real-world industrial robotic arm. It includes six state variables to represent the positions of the six joints on the robot, along with the six motor torque inputs that control and maneuver them. We applied the proposed approaches (with $n = 13$, $m = 6$, $k = 4$, $\gamma = 10$, $\epsilon = 0.1$, $\tau = 1$, $\delta = 0.01$) to solve the forward model identification task as specified in the benchmark. We also applied the NN and PARC approaches with similar parameters as specified in the previous benchmarks. Fig. 4 shows the performance of the flagged regression and the guarded regression algorithm on the test data in simulation mode. The normalized-root-mean-squared-error (NMRSE) and R^2 scores, averaged over all joints for the test data set are reported in Table 2. We see that the proposed approaches perform better (on the test data set) than the other approaches.

Table 2: Comparison using robotic arm benchmark data.

Approach	Test NMRSE	R^2 score	Time (s)
Linear [39]	0.83	0.31	unspecified
NN	0.30	0.88	3.02
PARC	1.78	-7.63	27.71
FR	0.14	0.98	82.32
GR	0.19	0.93	115.84

5 CONCLUSION

We introduced the flagged regression and guarded regression problems as interesting cases of switched linear and piecewise linear regression. We provided an approximation algorithm for this problem, whose complexity scales very well with the number of data points, as demonstrated in theory and in experiments. In future work, we plan to extend this approach to other classes of hybrid systems, such as switched/piecewise nonlinear systems, and hybrid automata, and to take physics-informed constraints into account.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their detailed comments and suggestions. This research was funded in part by the Belgian-American Education Foundation (BAEF) and the US National Science Foundation (NSF) under award numbers 1836900 and 1932189.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Alp Aydinoglu, Philip Sieg, Victor M Preciado, and Michael Posa. 2021. Stabilization of complementarity systems via contact-aware controllers. *IEEE Transactions on Robotics* 38, 3 (2021), 1735–1754.

[3] Alberto Bemporad. 2022. A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems. *IEEE Trans. Automat. Control* (2022).

[4] Alberto Bemporad, Giancarlo Ferrari-Trecate, and Manfred Morari. 2000. Observability and controllability of piecewise affine and hybrid systems. *IEEE transactions on automatic control* 45, 10 (2000), 1864–1876.

[5] Alberto Bemporad, Andrea Garulli, Simone Paoletti, and Antonio Vicino. 2005. A bounded-error approach to piecewise affine system identification. *IEEE Trans. Automat. Control* 50, 10 (2005), 1567–1580.

[6] Alberto Bemporad and Manfred Morari. 1999. Control of systems integrating logic, dynamics, and constraints. *Automatica* 35, 3 (1999), 407–427.

[7] Aharon Ben-Tal and Arkadi Nemirovski. 2001. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications.* SIAM.

[8] Guillaume Berger, Monal Narasimhamurthy, Kandai Watanabe, Morteza Lahijanian, and Sriram Sankaranarayanan. 2022. An Algorithm for Learning Switched Linear Dynamics from Data. *Advances in Neural Information Processing Systems* 35 (2022), 30419–30431.

[9] Guillaume O Berger and Sriram Sankaranarayanan. 2022. Learning fixed-complexity polyhedral Lyapunov functions from counterexamples. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 3250–3255.

[10] Guillaume O Berger and Sriram Sankaranarayanan. 2023. Counterexample-guided computation of polyhedral Lyapunov functions for piecewise linear systems. *Automatica* 155 (2023), 111165.

[11] Stephen Boyd and Lieven Vandenberghe. 2007. Localization and cutting-plane methods. *From Stanford EE 364b lecture notes* 386 (2007).

[12] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization.* Cambridge university press.

[13] Leo Breiman. 1993. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory* 39, 3 (1993), 999–1013.

[14] Ya-Chien Chang, Nima Roohi, and Sicun Gao. 2019. Neural lyapunov control. *Advances in neural information processing systems* 32 (2019).

[15] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-Guided Abstraction Refinement. In *Computer Aided Verification.* Springer, Berlin, Germany, 154–169. https://doi.org/10.1007/10722167_15

[16] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. 2021. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152* (2021).

[17] Shimon Even, Alan L. Selman, and Yacov Yacobi. 1984. The complexity of promise problems with applications to public-key cryptography. *Information and Control* 61, 2 (May 1984), 159–173. https://doi.org/10.1016/S0019-9958(84)80056-X

[18] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. 2003. A clustering technique for the identification of piecewise affine systems. *Automatica* 39, 2 (2003), 205–217.

[19] Oded Goldreich. 2006. On Promise Problems: A Survey. In *Theoretical Computer Science: Essays in Memory of Shimon Even.* Springer, 254–290.

[20] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[21] András Hartmann, João M Lemos, Rafael S Costa, João Xavier, and Susana Vinga. 2015. Identification of switched ARX models via convex optimization and expectation maximization. *Journal of Process Control* 28 (2015), 9–16.

[22] Wilhemus PMH Heemels, Bart De Schutter, and Alberto Bemporad. 2001. Equivalence of hybrid dynamical models. *Automatica* 37, 7 (2001), 1085–1091.

[23] Ath Kehagias, Ev Nidelkou, and V Petridis. 2006. A dynamic programming segmentation procedure for hydrological and environmental time series. *Stochastic Environmental Research and Risk Assessment* 20 (2006), 77–94.

[24] Fabien Lauer. 2013. Estimating the probability of success of a simple algorithm for switched linear regression. *Nonlinear Analysis: Hybrid Systems* 8 (2013), 31–47.

[25] Fabien Lauer. 2015. On the complexity of piecewise affine system identification. *Automatica* 62 (2015), 148–153.

[26] Fabien Lauer, Gérard Bloch, Fabien Lauer, and Gérard Bloch. 2019. *Hybrid system identification.* Springer.

[27] Fabien Lauer, Gérard Bloch, and René Vidal. 2011. A continuous optimization framework for hybrid system identification. *Automatica* 47, 3 (2011), 608–613.

[28] Eberhard Münz and Volker Krebs. 2005. Continuous optimization approaches to the identification of piecewise affine systems. *IFAC Proceedings Volumes* 38, 1 (2005), 349–354.

[29] Hayato Nakada, Kiyotsugu Takaba, and Tohru Katayama. 2005. Identification of piecewise affine systems based on statistical clustering technique. *Automatica* 41, 5 (2005), 905–913.

[30] Necmiye Ozay, Mario Sznaier, Constantino M Lagoa, and Octavia I Camps. 2011. A sparsification approach to set membership identification of switched affine systems. *IEEE Trans. Automat. Control* 57, 3 (2011), 634–648.

[31] Simone Paoletti, Aleksandar Lj Juloski, Giancarlo Ferrari-Trecate, and René Vidal. 2007. Identification of hybrid systems a tutorial. *European journal of control* 13, 2-3 (2007), 242–260.

[32] Luis A Rademacher. 2007. Approximating the centroid is hard. In *Proceedings of the twenty-third annual symposium on Computational geometry.* 302–305.

[33] Hadi Ravanbakhsh and Sriram Sankaranarayanan. 2019. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots* 43 (2019), 275–307.

[34] Jacob Roll, Alberto Bemporad, and Lennart Ljung. 2004. Identification of piecewise affine systems via mixed-integer programming. *Automatica* 40, 1 (2004), 37–50.

[35] Sadra Sadraddini and Calin Belta. 2018. Formal guarantees in data-driven model identification and control synthesis. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week).* 147–156.

[36] Armando Solar-Lezama. 2013. Program sketching. *International Journal on Software Tools for Technology Transfer* 15, 5 (Oct. 2013), 475–495. https://doi.org/10.1007/s10009-012-0249-7

[37] Vijay V Vazirani. 2001. *Approximation algorithms.* Vol. 1. Springer.

[38] René Vidal, Stefano Soatto, Yi Ma, and Sankar Sastry. 2003. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *42nd IEEE International Conference on Decision and Control*, Vol. 1. 167–172 Vol.1. https://doi.org/10.1109/CDC.2003.1272554

[39] Jonas Weigand, Julian Götz, Jonas Ulmen, and Martin Ruskowski. 2023. Dataset and Baseline for an Industrial Robot Identification Benchmark. (2023).

[40] H Paul Williams. 2013. *Model building in mathematical programming.* John Wiley & Sons.

A PROOF OF LEMMA 2.5

We will prove that the function

$$f(x,y) = \begin{cases} x+y & \text{if } x \geq 0,\ y \geq 0, \\ 0 & \text{otherwise} \end{cases}$$

cannot be expressed as a GLS $f_0 + \sum_{i=1}^{k} \operatorname{sign}(g_i)\, f_i$ where $f_0,\ldots,f_k$ and $g_1,\ldots,g_k$ are linear functions of x and y.

Let us assume without loss of generality that no guard g_i is identically zero, i.e., $g_i \not\equiv 0$ for all $i \in [k]$. It is then a trivial fact that the set S where no guard vanishes, i.e., $S = \{(x,y) \in \mathbb{R}^2 : g_i(x,y) \neq 0\, \forall\, i \in [k]\}$, is open, dense in $\mathbb{R}^2$ and symmetric with respect to the origin. Let S_1 be an open subset of $S \cap \mathbb{R}^2_{\geq 0}$. Then, for all $(x,y) \in S_1$, $f(x,y) = x+y$, and $f(-x,-y) = 0$ since $(-x,-y) \in \mathbb{R}^2_{\leq 0}$. Plugging such (x,y) and $(-x,-y)$ into the GLS expression, we get:

$$f_0(x,y) + \sum_{i=1}^{k} \operatorname{sign}(g_1(x,y))\, f_i(x,y) = x+y, \tag{4}$$

$$f_0(-x,-y) + \sum_{i=1}^{k} \operatorname{sign}(g_1(-x,-y))\, f_i(-x,-y) = 0. \tag{5}$$

Using the linearity of f_i and g_i, (5) is equivalent to

$$-f_0(x,y) + \sum_{i=1}^{k} \operatorname{sign}(g_1(x,y))\, f_i(x,y) = 0. \tag{6}$$

By subtracting (6) from (4), we get that $f_0(x,y) = \frac{x+y}{2}$ for all $(x,y) \in S_1$. Since S_1 is an open set and since f_0 is linear, we deduce that $f_0(x,y) = \frac{x+y}{2}$ for all $(x,y) \in \mathbb{R}^2$. We can do the same reasoning with an open subset $S_2 \subseteq S \cap (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\leq 0})$. This provides the conclusion that $f_0(x, y) = 0$ for all $(x, y) \in \mathbb{R}^2$. This is a contradiction with $f_0(x, y) = \frac{x+y}{2}$, concluding the proof. □

B PROOF OF THEOREM 2.6

We will start with the proof that the flagged regression problem is NP-hard. The "bounded-error" switched linear regression problem with two modes can be reduced in polynomial time to the flagged regression problem with one flag. Indeed, the first problem amounts to find two matrices $B_1, B_2 \subseteq \mathbb{R}^{m \times n}$ such that for all $t \in [N]$, there is $\sigma(t) \in [2]$ satisfying that $\|\vec{y}(t) - B_{\sigma(t)}\vec{x}(t)\| \leq \eta$, for some given error tolerance η. This can be formulated as a flagged regression problem with one flag and error tolerances $\epsilon = 0$ and $\tau = \eta$. Indeed, B_1, B_2 and $\sigma(t)$ for $t \in [N]$ is a solution of the switched linear regression problem iff $A_0 = \frac{1}{2}(B_1 + B_2)$ and $A_1 = \frac{1}{2}(B_1 - B_2)$ and $q(t) = 3 - 2\sigma(t)$ for $t \in [N]$ is a solution of the flagged regression problem. Since the switched linear regression problem is NP-hard [26, Sec. 5.2.4], the flagged regression problem is too.

Secondly, the "exact" piecewise affine regression problem with two modes can be reduced in polynomial time to the guarded regression problem with one guard. Indeed, the first problem amounts to find two matrices $B_1, B_2 \subseteq \mathbb{R}^{m \times n}$ and a vector $\vec{g} \in \mathbb{R}^n \setminus \{0\}$ such that for all $t \in [N]$, $\vec{y}(t) = B_{\sigma(t)}\vec{x}(t)$, wherein $\sigma(t) = \frac{1}{2} + \frac{1}{2}\mathrm{sign}(\vec{g}^\top \vec{x}(t))$. This can be formulated as a guarded regression problem with one guard and error tolerances $\epsilon = \tau = 0$. Indeed, B_1, B_2 and $\vec{g}$ is a solution of the piecewise linear regression problem iff $A_0 = \frac{1}{2}(B_1 + B_2)$ and $A_1 = \frac{1}{2}(B_1 - B_2)$ and $\vec{c} = \vec{g}$ is a solution of the flagged regression problem. Since the piecewise linear regression problem is NP-hard [26, Sec. 5.2.3], the guarded regression problem is too. □

C APPROXIMATION OF OPTIMAL SOLUTION USING REPEATED CALLS TO ALGO. 1

In this appendix, we show how repeated calls to Algo. 1 can be used to construct a model that approximates the optimal solution to within $2\epsilon_{\text{gap}}$.

As inputs, we assume a fixed data set $\mathcal{D}$, absolute error tolerance τ, bound γ on the coefficients and $\epsilon_{\text{gap}} > 0$. For technical reasons, we require the data set $\mathcal{D}$ to be fit with a linear regression model (no flags) with absolute error tolerance τ and bound γ on the coefficients (Cf. Remark 3). However, the relative error of such a model is not required to be within bounds. Let B be the relative error achieved by such a linear regression model.

Let ϵ^* be the optimal relative error tolerance such that (a) there exits a model with relative error ϵ^* and (b) no model fits the data with absolute error tolerance τ and bound γ and relative error $< \epsilon^*$.

Lemma C.1. *If a linear regression model with coefficients bound γ, absolute error τ and relative error B exists, then $0 \leq \epsilon^* \leq B$.*

Note that we can find a linear regression model and the corresponding relative error bound B in polynomial time using linear programming.

Algo. 5 presents the repeated-call algorithm to approximate the optimal solution. We provide below a detailed analysis of the algorithm.

Algorithm 5: Approximation of Optimal Solution.

```
Data: Data set $\mathcal{D}$, absolute error $\tau$, bound $\gamma$, gap $\epsilon_{\text{gap}}$.
Result: Bounds $(\ell, u)$ such that there is a model that fits the
        data with error $\tau$, bound $\gamma$ and relative error $u$,
        $u - l \leq 2\epsilon_{\text{gap}}$ and $\ell \leq \epsilon^* \leq u$.
1  $B \leftarrow$ findLinearRegressorWithBounds($\mathcal{D}, \tau, \gamma$)
    /* Assume: linear regression succeeded and $B$ is
    an upper bound on $\epsilon^*$. */
2  $(\ell, u) \leftarrow (0, B)$
3  while $(u - l) > 2\epsilon_{\text{gap}}$ do
4      $m \leftarrow \frac{u+l}{2}$
5      Run Algo. 1 with $\mathcal{D}, \tau, \gamma, \epsilon_1 = m - \epsilon_{\text{gap}}/2$,
        $\epsilon_2 = m + \epsilon_{\text{gap}}/2$
6      if Feasible then
7          $u \leftarrow$ relative error of model returned by Algo. 1
8      else
9          $l \leftarrow m - \frac{\epsilon_{\text{gap}}}{2}$
10 return $(\ell, u)$
```

Lemma C.2. *Whenever control is in Line 3 of Algo. 5, the following facts hold:*

(1) *There exists a flagged linear model with relative error u, absolute error τ, and bound γ.*
(2) $\ell \leq \epsilon^* \leq u$

Proof. Proof is by induction on the number of times, the body of the while loop runs. The base case is when the loop has run 0 times. We have $\ell = 0, u = B$ and the statements hold trivially.

Suppose it were true after i iterations of the loop. If the loop ran once more. Suppose the call to Algo. 1 was feasible, then the new value of u corresponds to the relative error of a model. The two statements hold at the beginning of the next iteration. Otherwise, Algo. 1 guarantees that $\epsilon^* > \epsilon_1 = \frac{l+u-\epsilon_{\text{gap}}}{2}$. Therefore, the statement holds at the start of the next iteration in this case as well. □

Let ℓ_i, u_i be the values of the program variables ℓ, u after $i \geq 0$ iterations of the while loop. We can prove by induction that

$$u_i - \ell_i \leq \frac{B - \epsilon_{\text{gap}}}{2^i} + \epsilon_{\text{gap}} .$$

Theorem C.3. *If there exists a linear regression model satisfying with absolute error τ, gap γ and relative error B then Algo. 5 yields bounds ℓ, u such that (a) there exists a flagged linear model with relative error u, absolute error τ, and bound γ; (b) $\ell \leq \epsilon^* \leq u$; (c) $u - \ell \leq 2\epsilon_{\text{gap}}$. Furthermore, its running time is in $O\left(\log_2\left(\frac{B-\epsilon_{\text{gap}}}{\epsilon_{\text{gap}}}\right)\right)$.*

Proof. Note that at any iteration of the algorithm, we know that there are no models with relative error $< \ell_i$ and there is a model with relative error $\leq u_i$. This is true at the very beginning and note that after each iteration of the while loop, the Algo 1 guarantees either a model with relative error at most ϵ_2 or no models with relative error $< \epsilon_1$. Thus, when the algorithm exits after k iterations, we automatically note that there are no models with relative error $< \ell_k$, there is a model with relative error $\leq u_k$ and $u_k - \ell_k \leq 2\epsilon_{\text{gap}}$. This proves (a), (b).

The bound on the running time is a direct consequence of Lemma C.2. Note that at the very beginning, $l_0 = 0, u_0 = B$ and furthermore, after i steps of the loop iteration, we have

$$u_i - \ell_i \leq \frac{B - \epsilon_{\text{gap}}}{2^i} + \epsilon_{\text{gap}} \,.$$

The algorithm exits when $u_i - \ell_i \leq 2\epsilon_{\text{gap}}$. Combining, these observations, we obtain that the running time is upper bounded by $O\left(\log_2\left(\frac{B-\epsilon_{\text{gap}}}{\epsilon_{\text{gap}}}\right)\right)$. □

D PARAMETERS FOR NEURAL NETWORK AND PARC APPROACHES

The neural networks used in Sec. 4 consist of 2 layers of 32 ReLU-activated nodes. We ensured that the neural networks used in Sec. 4 for the experimental evaluation were sufficiently large to capture the various modes in the system. Training of these networks was performed using the Adam optimization algorithm in TensorFlow. Our training approach is “standard”: adapted from the scripts provided for training NNs for regression as part of the TensorFlow package user manual. We used a batch size of 32 for 100 training epochs. We also ensured that in each case, the reported training error was quite small ($\sim\ 10^{-4}$). We used a (single fold) cross-validation approach wherein 80% of the data was used for training while 20% of the data was used for testing.

The PARC tool [3] fits a piecewise affine model over a polyhedral partitioning of the feature region. The parameters of the tool include K which represents the maximum number of regions in the partitioning. We set this value to 10, in accordance with the examples provided in the tool. In the experimental evaluation in Sec. 4, all of the identified models used fewer partitions than K. We also set $\alpha = 10^{-4}$, and maxiter = 15, as recommended by the tool.