

Localised Kinky Inference

A. Blaas¹, J.M. Manzano², D. Limon² and J. Calliess¹

Abstract—Their flexibility to learn general function classes renders nonparametric regression algorithms particularly attractive in system identification and data-based control settings, where little a priori knowledge about a dynamical system is to be presumed. Building on approaches known as NSM- or Lipschitz regression, we propose a new nonparametric machine learning approach. While it inherits theoretical learning guarantees from the methods it is built upon, it is designed to limit the computational effort both for learning and for generating predictions. This renders our method applicable to online system identification and control settings where the desired sample frequency precludes previous nonparametric approaches from being deployed. Apart from deriving a guarantee on the ability of our method to learn any continuous function, we illustrate some of its practical merits on a number of benchmark comparison problems.

I. INTRODUCTION

In system identification and data-based control, machine learning algorithms have gained increasing popularity. They promise increased flexibility to learn and predict in rich classes of dynamical systems. If learning and prediction can be performed with sufficient frequency, this would allow a controller to achieve its objective even if the plant dynamics either change, or substantially deviate from an a priori model. Among the manifold learning methods available, nonparametric regression approaches offer the greatest flexibility and typically are guaranteed to be able to learn the widest classes of functions in the infinite data limit [14].

Unfortunately, their computational complexity for prediction tends to increase with the observed data. Furthermore, most practically viable techniques for nonparametric regression, such as the Nadaraya-Watson estimator [19], [26], the LOESS method [11], Lipschitz interpolation [24] or Gaussian processes (GPs) [20], suffer from a practical limitation: their regression performance critically depends on an a priori choice of hyperparameters.

To make these approaches perform well in practice, training methods typically involve hyperparameter tuning which is often performed either manually or by employing optimisers which then effectively become part of the learning algorithm.

While such approaches can often yield impressive practical performance in control-related reinforcement learning applications [12], [13], the optimisers will typically have to be allowed to run for a long time each time they are presented with new data to obtain sensible results. This substantially

exacerbates the computational training effort, denying the application of such methods to system identification and online control in all but slowly evolving dynamical systems.

Recent developments that have sought to address these limitations were built on a class of nonparametric learning methods that have been referred to as *Kinky Inference* [5]. This class, which encompasses Lipschitz interpolation [24] and NSM-methods [17] allows for an easier theoretical analysis in control [5], [7]. Their computations are simple enough to be run on an embedded system, and their hyperparameter estimation is computationally tractable enough to allow for theoretical guarantees on the outcomes, which in turn can be extended to guarantees of the data-based controllers [3], [16], [15].

In the context of data-based model-predictive control, [16] designed a scheme to reduce prediction effort by partitioning the data space into hypercubes. A prediction of a query input would then be based on training examples contained in the same or adjacent hypercubes. By managing the size of the cubes appropriately, the authors could provide a probabilistic bound on the computational prediction error under distributional assumptions on the data. They have also demonstrated how this approach can render the approach tractable in MPC. However, their work assumed offline learning only. That is, the learner was trained on a fixed sample of data collected before the controller was applied. This was necessary, since their hyperparameter estimator was based on a modified Strongin estimate of a global Lipschitz constant which allowed them to link to existing theoretical convergence analysis of the resulting regressor as presented in [3]. Unfortunately, the computational effort for estimating the global Lipschitz constant hyperparameter still grows quadratically with the number of data points.

In this work, we will address this deficiency. Instead of computing one global Lipschitz hyperparameter, we will maintain a set of local Lipschitz constant hyperparameter estimates predictions will be based on. This promises to have two benefits: Firstly, it allows us to bound computational effort for training (provided we can bound the number of training examples to be considered for the computation of each constant and project onto this set in constant time). Secondly, it can improve prediction performance on locally Lipschitz target functions that have local Lipschitz constants that vary substantially in different parts of input space. Since our method is a Kinky Inference regression approach that relies on fully localised information only, we will refer to it as *Localised Kinky Inference (LoKI)*.

In the remainder of this paper, we introduce our LoKI method and show how it can be set up to have bounded

J.M. Manzano and D. Limon gratefully acknowledge support from MINECO-Spain and FEDER Funds under project DPI2016-76493-C3-1-R and the University of Seville under contract VI-PPITUS.

¹OMI, Dept. of Engineering Science, Oxford University, UK

²Dept. of Systems Engineering and Automation, Seville University, Spain

expected computational complexity, both for training and prediction. This property renders our approach amenable to online learning-based control settings with higher sampling frequency than could have been achieved with previous nonparametric learning methods. Furthermore, we provide accompanying learning-theoretic worst-case guarantees, that allow us to be confident that we can learn any continuous function and thereby differentiate our work from earlier attempts to reduce the computational effort of estimating the Lipschitz hyperparameter [2]. Finally, we illustrate the merits of our approach in a simple online learning-based flight manoeuvre control task previously considered as a test bed in the literature [8], [3], [4], and additionally compare the performance of our regression method to competing learning methods on a number of standard benchmark data sets.

II. KINKY INFERENCE AND LIPSCHITZ INTERPOLATION

The term *Kinky Inference (KI)* was introduced in [5] to describe a class of nonparametric regression rules which we will briefly rehearse in this section. The regression task under consideration is to learn a continuous *target function* $f : \mathcal{W} \rightarrow \mathcal{Z}$.

It is assumed that f is Hölder continuous with constant L^* and exponent $0 < \alpha \leq 1$ relative to the input and output space pseudo-metrics \mathfrak{d} and $\mathfrak{d}_{\mathcal{Z}}$. That is, for all $w_1, w_2 \in \mathcal{W}$ we have $\mathfrak{d}_{\mathcal{Z}}(f(w_1), f(w_2)) \leq L^* \mathfrak{d}(w_1, w_2)^\alpha$. For the special case $\alpha = 1$, f is called Lipschitz continuous, and while our theoretical analysis readily extends to the general Hölder case, we shall assume $\alpha = 1$ for the remainder of this work (note, any continuous function is Lipschitz up to an arbitrarily small error, so our framework addresses learning of any continuous function). We henceforth assume $\mathcal{W} \subset \mathbb{R}^d$ and also, for ease of notation, but without loss of generality, we will restrict our exposition to the case where f is real-valued, i.e. $\mathcal{Z} = \mathbb{R}$ and $\mathfrak{d}_{\mathcal{Z}}(z, z') = \|z - z'\|$. Regression is the task of predicting the target function at unobserved query inputs on the basis of a (possibly noisy) sample or data set

$$\mathcal{D}_n := \{(w_i, \tilde{f}(w_i)) | i = 1, \dots, n\},$$

where \tilde{f} denotes a noisy version of f (note, we will use \tilde{f}_i and $\tilde{f}(w_i)$ interchangeably). The set containing only the input data points is denoted as $\mathcal{W}_{\mathcal{D}_n} = \text{Proj}_{\mathcal{W}}(\mathcal{D}_n)$. The regression task then is to utilise \mathcal{D}_n to compute a prediction $\hat{f}_n(q)$ of $f(q)$ for any *query* input $q \in \mathcal{W}$.

While there are a great many methods to compute such a prediction we will build upon the following class of regressors:

Definition 1 (Kinky Inference rule [5] - simplified): Let $q \in \mathcal{W}$ and \mathcal{D}_n defined as before. We define

$$u_n(q; \theta_n, \mathcal{D}_n) := \min_{w_i \in \mathcal{W}_{\mathcal{D}_n}} \tilde{f}_i + \mathfrak{d}(q, w_i; \theta_n), \quad (1)$$

$$l_n(q; \theta_n, \mathcal{D}_n) := \max_{w_i \in \mathcal{W}_{\mathcal{D}_n}} \tilde{f}_i - \mathfrak{d}(q, w_i; \theta_n) \quad (2)$$

as upper and lower bound functions (depending on some hyperparameter θ_n) from which we construct the Kinky

Inference predictor $\hat{f}_n(\cdot; \theta_n, \mathcal{D}_n) : \mathcal{W} \rightarrow \mathcal{Z}$ as per:

$$\hat{f}_n(q; \theta_n, \mathcal{D}_n) := \frac{1}{2} u_n(q; \theta_n, \mathcal{D}_n) + \frac{1}{2} l_n(q; \theta_n, \mathcal{D}_n). \quad (3)$$

As stated before, we restrict our analysis to the special case of KI where $\mathfrak{d}(x, y; \theta_n) = L_n \|x - y\|$, which is often referred to as *Lipschitz Interpolation* or as *Nonlinear Set Interpolation* with the hyperparameter $\theta_n = L_n$ being the supposed Lipschitz constant of the target. If L^* is known, it can be used as L_n . However when it is unknown, L_n it has to be estimated from the data, for which different methods have been proposed, starting with Strongin's estimator [23]. More recently, methods like LACKI [3] and POKI [6] have been developed that address the case of corrupted or noisy observations [5], [4].

Unfortunately, these approaches share certain drawbacks. Firstly, the computational effort for estimating L_n is quadratic in the number of sample points n , which for large data sets is prohibitive. Similarly, the prediction cost for computing predictor $\hat{f}_n(x)$ is linear in the number of sample points. This can become prohibitive as the data grows, especially in control settings which require frequent predictions during planning updates. Secondly, the resulting estimator does not take into account local differences in smoothness of the target function since L_n is estimated globally.

III. LOCALISED KINKY INFERENCE

Going back to a more general, non-simplified definition of the KI rule [5], we can define a version of the KI predictor which addresses the two drawbacks described above. The key element is based in the idea of only considering a certain subset $\mathcal{I}_n(q) \subset \{1, \dots, n\}$ of indices to construct the predictor in Eq. (3) [5]. Our main contribution is to extend this idea and also estimate the Lipschitz constant based on such subsets, instead of globally. Defining $\mathcal{I}_n(q)$ to only contain those indices whose sample points are in a same local region as the query input q , we call the resulting KI method *Localised Kinky Inference (LoKI)*.

Formally, the LoKI predictor is defined as

$$\hat{f}(q; L_n(q), \mathcal{D}_n) = \frac{1}{2} \min_{i \in \mathcal{I}_n(q)} (\tilde{f}_i + L_n(q) \|q - w_i\|) + \frac{1}{2} \max_{i \in \mathcal{I}_n(q)} (\tilde{f}_i - L_n(q) \|q - w_i\|), \quad (4)$$

for which we shorthand write $\hat{f}_n(q)$. As for the global KI predictor, different methods can be used to determine $L_n(q)$. However, to facilitate the theoretical analysis, in this paper we restrict our implementation of the LoKI method to the application of the LACKI estimator [3]:

$$L_n(q) := \max_{(w_i, w_j) | i, j \in \mathcal{I}_n(q), w_i \neq w_j} \frac{\|\tilde{f}(w_i) - \tilde{f}(w_j)\| - \gamma}{\|w_i - w_j\|} \quad (5)$$

where $\gamma := 2\bar{\epsilon}$ is a hyperparameter that should be set to twice the level of the maximal observational error $\bar{\epsilon}$. That is, for our theoretical analysis (but not necessarily our practical

applications), we will have to assume observational errors (stochastic noise or systematic errors) to be bounded by some $\bar{\epsilon} \geq 0$: $\forall w \in \mathcal{W} : \|\tilde{f}(w) - f(w)\| \leq \bar{\epsilon}$. In cases where $\bar{\epsilon}$ is unbounded or unknown, we recommend to utilise POKI instead.

While LACKI and all other global methods use all available data points to learn the hyperparameter and to predict at a new query point, our method only uses the subset $\mathcal{I}_n(q)$ to do so, which significantly reduces the computation time while simultaneously preventing the model from being overly conservative in particularly smooth regions. Fig. 1 illustrates this in comparison to the global (LACKI) estimation. As a trade-off, the LoKI predictor might exhibit discontinuities, but as we show in subsection C., any such discontinuities can be bounded and one can still recover (Lipschitz) continuity of the predictor almost everywhere.

A. Defining Hyperrectangles

So far, we have not specified $\mathcal{I}_n(q)$. In order to do so, we partition the input space $\mathcal{W} \subset \mathbb{R}^d$ into several disjoint subsets $\mathcal{W}_k \subset \mathcal{W}$ covering the domain on which we desire to predict target function values.

Correspondingly, \mathcal{D}_n is partitioned into a collection of pertaining nonintersecting subsets of sample points defined by $D_k := \{(w_i, \tilde{f}(w_i)) \in \mathcal{D}_n | w_i \in \mathcal{W}_k\}$. This partition is calculated offline and then a classification algorithm is built to locate the partition \mathcal{W}_k to which a query point q belongs to. $\mathcal{I}_n(q)$ is then composed of the indices of all data points which lie in D_k .

The partition can be calculated taking into account different objectives, as for instance, to ensure a regular distribution of data points. A survey of partitioning methods for system identification can be found in [21]. Let $w_{(l)}$ denote the l^{th} component of input $w \in \mathcal{W}$. Assuming that all data lies in some original hyperrectangle in input space of the form $H = \{w : w_{(l)} \in [\mathbf{a}_{(l)}, \mathbf{b}_{(l)}], \forall l = 1, \dots, d\}$ with $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$

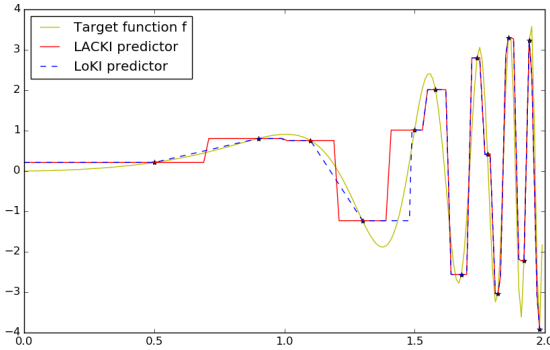


Fig. 1. Illustration of the advantage of learning the Lipschitz constant locally. The target function $f(x) = x^2 \sin(2x^2)$ (yellow) as predicted by LACKI (orange) and LoKI with $m = 3$ (dotted blue). While both predictors coincide in the rightmost hyperrectangle ($x > 1.5$), they diverge in the other hyperrectangles as LACKI overestimates the (local) Lipschitz constant and predicts further off the true target function whose local smoothness is better estimated by LoKI.

(once the samples are drawn, we can always ensure this by appropriate rescaling of the workspace), we partition H into m^d (sub-)hyperrectangles H_k indexed by the tuple $k := (k_1, \dots, k_d)$ where $k_l \in \{1, \dots, m\}$ for $(l = 1, \dots, d)$ and define

$$H_k = \left\{ w : k_l = \min\{m, \max\{1, \left\lfloor \frac{m(w_{(l)} - \mathbf{a}_{(l)})}{\mathbf{b}_{(l)} - \mathbf{a}_{(l)}} \right\rfloor + 1\}\} \right\}, \quad \forall l = 1, \dots, d.$$

These hyperrectangles then define our partition, i.e.

$$\mathcal{W}_k = H_k \quad (6)$$

B. Computational Complexity

We will now evaluate the computational complexity of the proposed model under the assumption that data points are i.i.d. uniformly drawn in H . This will be done separately for training and prediction.

1) Training.

Let n_k be the number of sample inputs contained in H_k . For a given sample set \mathcal{D}_n let $\mathcal{K} = \{k = 1, \dots, m^d | n_k > 0\}$ denote the indices of hyperrectangles with at least one sample input. For any given point $w \in H_k$, computing the index $k = (k_1, \dots, k_d)$ can be performed in $\mathcal{O}(d)$ basic computational steps. Once a hyperrectangle index of a new sample is computed, in the context of online learning, an incremental update of the Lipschitz constant estimate can be computed in $\mathcal{O}(n_k)$ computational steps.

On the other hand, in batch learning, where the entire data set is available at training time, we can train in two steps: in the first step, we assign all training points to their corresponding hyperrectangles, amounting to $\mathcal{O}(nd)$ computational effort. The local LACKI estimator then requires $\mathcal{O}(n_k^2)$ computational steps for computing local Lipschitz constant estimate L_k . Thus, the overall effort for batch training amounts to

$$T_{n,m,d}^{\text{train}} := c_1 \sum_{k \in \mathcal{K}} (c_2 + n_k^2) + c_3 n d + c_4$$

computational steps for some (algorithm-dependent) parameters c_1, c_2, c_3, c_4 . Given the uniform i.i.d. distribution of the sample points in \mathcal{W} , n_k is a random variable following a binomial distribution with success probability $p = m^{-d}$ (the probability for any single sample point to be in H_k) and number of trials n . The expected number of samples per hyperrectangle is $\mathbb{E}[n_k^2] = \frac{n^2}{m^{2d}} + \frac{n}{m^d} - \frac{n}{m^{2d}}$ and thus,

$$\begin{aligned} \mathbb{E}[T_{n,m,d}^{\text{train}}] &= \mathbb{E}\left[c_1 \sum_{k \in \mathcal{K}} (c_2 + n_k^2)\right] + c_3 n d + c_4 \\ &\leq c_1 \left(n c_2 + \frac{n^2 - n}{m^d} + n\right) + c_3 n d + c_4 \\ &\leq c_1 n (c_2 + 1 + (n-1)m^{-d}) + c_3 n d + c_4 \end{aligned}$$

Thus, $\mathbb{E}[T_{n,m,d}^{\text{train}}] \in \mathcal{O}(n^2 m^{-d} + nd)$, as opposed to the $\mathcal{O}(n^2)$ computational steps required for batch training with LACKI.

If training is done online, we can derive $\mathbb{E}[T_{n,m,d}^{\text{update}}]$, the

expected number of computational steps necessary per update, using

$$\mathbb{E}[n_k] = \frac{n}{m^d},$$

which together with similar arguments as above gives $\mathbb{E}[T_{n,m,d}^{\text{update}}] \in \mathcal{O}(m^{-d}n+d)$, compared to $\mathcal{O}(n)$ required for vanilla KI methods such as LACKI.

2) Prediction.

The number $T_{n,m,d}^{\text{predict}}$ of computational steps required for prediction is the same as for the projected KI method introduced in [16]. We have

$$\mathbb{E}[T_{n,m,d}^{\text{predict}}] = c_5 m^{-d} n + c_6 d + c_7 \in \mathcal{O}(m^{-d}n + d).$$

These results highlight the reduction of computational complexity that can be achieved with LoKI and they can be utilised to answer the question of how to choose parameters m to control the average and maximal computational effort per query point.

C. Consistency Guarantees

We will now show that the LoKI predictor based on LACKI inherits important desirable theoretical properties such as sample consistency and worst-case consistency (if observational errors are bounded). Improving computational effort by localisation might introduce discontinuities, which however, we will bound as well. Our proofs rest on properties of the global LACKI estimator, which from now on, we assume the LoKI predictor to be based on (refer Eqn. (5)). This renders most proofs to be straight-forward extensions of the ones given in [3], which is why for conciseness, we restrict our exposition to proof sketches only.

Lemma 1 (Sample consistency): For any $w_i \in \mathcal{W}_{\mathcal{D}_n}$, we have $\|\hat{f}_n(w_i) - f(w_i)\| \leq \frac{\gamma}{2} + \bar{\epsilon}$

Proof. The proof is completely analogous to that of Lemma 2.7 in [3], where one only needs to replace \mathcal{D}_n with $D_k(w_i)$ to apply their proof to our case.

Lemma 2 (Worst case error of LoKI): Under the assumption that f is Lipschitz continuous with best Lipschitz constant L^* , the following guarantee holds for the worst case prediction error of LoKI with partition defined as in (6): For any $q \in \mathcal{W}$, we have

$$\|\hat{f}_n(q) - f(q)\| \leq (L_n(q) + L^*)\|q - \xi_n^q\| + \frac{\gamma}{2} + \bar{\epsilon} \quad (7)$$

$$\leq 2L^*\|q - \xi_n^q\| + \frac{\gamma}{2} + \bar{\epsilon} \quad (8)$$

with $\xi_n^q \in \arg \min_{w_i \in H_k(q)} \|w_i - q\|$ being the nearest neighbour of q in the same hyperrectangle (among the sample data points).

Proof. Again, the proof is very similar to the equivalent statement for the LACKI method provided in [3]. Let $\Delta :=$

$\|\hat{f}_n(q) - f(q)\|$. Applying the triangle inequality entails:

$$\begin{aligned} \Delta &\leq \|\hat{f}_n(q) - \hat{f}_n(\xi_n^q)\| + \|\hat{f}_n(\xi_n^q) - f(\xi_n^q)\| \\ &\quad + \|f(\xi_n^q) - f(q)\| \\ &\stackrel{(i)}{\leq} \|\hat{f}_n(q) - \hat{f}_n(\xi_n^q)\| + \|\hat{f}_n(\xi_n^q) - f(\xi_n^q)\| \\ &\quad + L^*\|\xi_n^q - q\| \\ &\stackrel{(ii)}{\leq} \|\hat{f}_n(q) - \hat{f}_n(\xi_n^q)\| + \frac{\gamma}{2} + \bar{\epsilon} \\ &\quad + L^*\|\xi_n^q - q\| \\ &\stackrel{(iii)}{\leq} L_n(q)\|q - \xi_n^q\| + \frac{\gamma}{2} + \bar{\epsilon} \\ &\quad + L^*\|\xi_n^q - q\| \\ &\stackrel{(iv)}{\leq} 2L^*\|\xi_n^q - q\| + \frac{\gamma}{2} + \bar{\epsilon} \end{aligned}$$

Here, (i) follows from the Lipschitz continuity of f , (ii) follows from Lemma 1 and (iii) follows from Lemma 2.6 in [3] because by definition of ξ_n^q we have $\mathcal{I}_n(q) = \mathcal{I}_n(\xi_n^q)$. Finally it is easy to see from Eq. 5 that $L_n(q) \leq \max_{(w_i, w_j) | i, j \leq n, i \neq j} \frac{\|\hat{f}(w_i) - \hat{f}(w_j)\| - \gamma}{\|w_i - w_j\|}$. This expression has been shown in Remark 2.2 in [3] to be smaller or equal to L^* , from which (iv) follows and the claim is proven.

On the basis of Lemma 2, it is straight-forward to establish that our method is capable of learning any Lipschitz continuous function in the limit of increasing data density:

Corollary 1 (Consistency of the LoKI predictor): If the grid of sample points becomes pointwise dense in \mathcal{W} as $n \rightarrow \infty$, the pointwise prediction error vanishes up to $\frac{\gamma}{2} + \bar{\epsilon}$, i.e. $\|\hat{f}_n(q) - f(q)\| \rightarrow [\frac{\gamma}{2} + \bar{\epsilon}]$ for any $q \in \mathcal{W}$. The same holds true for the worst case prediction error $\sup_{q \in \mathcal{W}} \|\hat{f}_n(q) - f(q)\|$ if the grid of sample points becomes uniformly dense in \mathcal{W} .

Lemma 3 (Boundedness of discontinuities): Given \mathcal{D}_n and the partition defined in (6), any discontinuity of the predictor is bounded by $c = 4L^*z(m, \|\cdot\|) + \gamma + 2\bar{\epsilon} < \infty$, where z is a bounded function that maps the number of hyperrectangles per dimension, m , and the norm used, $\|\cdot\|$, onto the maximum possible metric distance between two points in the same hyperrectangle under that norm (and is therefore decreasing in m).

Proof (sketch). For any $q_1, q_2 \in \mathcal{W}$ we have for $\Lambda = \|\hat{f}_n(q_1) - \hat{f}_n(q_2)\|$ (using the triangle inequality and Lemma 2)

$$\begin{aligned} \Lambda &\leq \|f(q_1) - \hat{f}_n(q_2)\| + \|\hat{f}_n(q_1) - f(q_1)\| \\ &\leq \|f(q_1) - f(q_2)\| + \|f(q_2) - \hat{f}_n(q_2)\| \\ &\quad + 2L^*\|q_1 - \xi_n^{q_1}\| + \frac{\gamma}{2} + \bar{\epsilon} \\ &\leq 2L^*(\|q_1 - \xi_n^{q_1}\| + \|q_2 - \xi_n^{q_2}\|) \\ &\quad + \gamma + 2\bar{\epsilon} + \|f(q_1) - f(q_2)\| \\ &\leq 4L^*z(m, \|\cdot\|) + \gamma + 2\bar{\epsilon} \\ &\quad + \|f(q_1) - f(q_2)\| \end{aligned}$$

By Lipschitz continuity of f , the last term $\|f(q_1) - f(q_2)\|$

vanishes as $\|q_1 - q_2\| \rightarrow 0$ and the claim follows.

Lemma 4 (Occurrence of discontinuities): Given \mathcal{D}_n and the partition defined in (6), the Lebesgue measure of the set of possible discontinuities in the predictor \hat{f}_n is 0, i.e. for $S_n := \{q \in \mathcal{W} | \exists \epsilon > 0 \forall \delta > 0 \exists q' \in \mathcal{W} : \|q - q'\| < \delta, \|\hat{f}_n(q) - \hat{f}_n(q')\| \geq \epsilon\}$, we have $\lambda(S_n) = 0$.

Proof. In the interior of any hyperrectangle H_k , the LoKI predictor corresponds to the global KI predictor which is known to be Lipschitz continuous and thereby continuous [24]. Discontinuities can therefore only occur at the boundaries of the hyperrectangles, which are subsets of the union $d - 1$ -dimensional hyperplanes. It is a standard result in measure theory that $d - 1$ -dimensional hyperplanes have 0 d -dimensional Lebesgue measure.

From this last result it follows that under any probability measure built on top of the Lebesgue measure, there is 0 probability of sampling from the boundaries between the partitioning hyperrectangles, and therefore any query point is surrounded with probability 1 by an epsilon ball where \hat{f}_n is continuous.

While these last results show that the discontinuities that can occur in theory using LoKI are not to be expected to cause problems in practice, removing them altogether could be attempted by using soft windows around prediction points instead of fixed hyperrectangles. The drawback of this method would however be that it would increase prediction effort time substantially and would undermine our wish to bound worst-case computational effort.

IV. EXPERIMENTS

In this section, we compare our approach to a number of well-established machine learning methods on problems in both control (A.) and standard pattern recognition (B.).

A. Model-Reference Adaptive Flight Control under Wingrock

We conceived our LoKI approach, bearing in mind the requirements in adaptive control settings. Here, we want to deploy learning methods that can learn and predict online with sufficient rapidity, to inform control actions with the desired sample frequency. To test our method against these requirements in comparison to alternative algorithms, we will apply it to a simple simulated benchmark tracking control problem that has previously been considered in the literature ([4], [8], [10]) as a test bed for learning-based model reference adaptive controllers.

We will commence with 1) briefly reviewing model reference adaptive control (MRAC) [1] as considered in [9] and explaining the deployment of Kinky Inference to this framework before proceeding with 2) the description of the test bed and our benchmark results.

1) *Model reference adaptive control and KI:* We will now rehearse the description of MRAC for second-order systems following [9].

Assume $s \in \mathbb{N}$ to be the dimensionality of a configuration of the system in question and define $h = 2s$ to be the dimensionality of the pertaining state space \mathcal{X} .

Let $x = [x_1; x_2] \in \mathcal{X}$ denote the state of the plant to be controlled. Given the control-affine system

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = a(x) + b(x)u(x) \quad (9)$$

it is desired to find a control law $u(x)$ such that the closed-loop dynamics exhibit a desired reference behaviour:

$\dot{\xi}_1 = \xi_2, \dot{\xi}_2 = f_r(\xi, r)$ where r is a reference command, f_r some desired response and $t \mapsto \xi(t)$ is the reference trajectory.

If a priori a and b are believed to coincide with \hat{a}_0, \hat{b}_0 respectively, the inversion control $u = \hat{b}_0^{-1}(-\hat{a}_0 + u')$ is applied. This reduces the closed-loop dynamics to $\dot{x}_1 = x_2, \dot{x}_2 = u' + \tilde{a}(x, u)$ where $\tilde{a}(x, u)$ captures the modelling error of the dynamics:

$$\tilde{a}(x, u) = a(x) - \hat{a}_0(x) + (b(x) - \hat{b}_0(x))u. \quad (10)$$

Let $I_h \in \mathbb{R}^{h \times h}$ denote the identity matrix. If b is perfectly known, then $b - \hat{b}_0^{-1} = 0$ and the model error can be written as $\tilde{a}(x) = a(x) - \hat{a}_0(x)$. In particular, \tilde{a} has lost its dependence on the control input.

In this situation [9], [8] propose to set the pseudo control as follows: $u'(x) := \nu_r + \nu_{pd} - \nu_{ad}$ where $\nu_r = f_r(\xi, r)$ is a feed-forward reference term, ν_{ad} is a yet to be defined output of a learning module *adaptive element* and $\nu_{pd} = [K_1 K_2]e$ is a feedback error term designed to decrease the *tracking error* $e(t) = \xi(t) - x(t)$ by defining $K_1, K_2 \in \mathbb{R}^{s \times s}$ as described in what is to follow.

Inserting these components, we see that the resulting *error dynamics* are:

$$\dot{e} = \dot{\xi} - [x_2; \nu_r + \nu_{pd} + \tilde{a}(x)] = Me + B(\nu_{ad}(x) - \tilde{a}(x)) \quad (11)$$

where $M = \begin{pmatrix} O_s & I_s \\ -K_1 & -K_2 \end{pmatrix}$ and $B = \begin{pmatrix} O_s \\ I_s \end{pmatrix}$. If the feedback gain matrices K_1, K_2 parameterising ν_{pd} are chosen such that M is stable then the error dynamics converge to zero as desired, provided the learning error E_λ vanishes, i.e. if $E_\lambda(x(t)) = \|\nu_{ad}(x(t)) - a(x(t))\| \xrightarrow{t \rightarrow \infty} 0$.

It is assumed that the adaptive element is the output of a learning algorithm that is tasked to learn \tilde{a} online. This is done by continuously feeding it training examples of the form $(x(t_i), \tilde{a}(x(t_i)) + \varepsilon_i)$. Here ε_i is observational noise, we assume the states $x(t_i)$ are observable and serve as inputs to our learned prediction model about a (thus, to link to our previous notation $\mathcal{X} = \mathcal{W}, x_i = w_i, a = f$ and $\tilde{a}(x(t_i)) + \varepsilon_i = \tilde{f}_i$).

Intuitively, assuming the learning algorithm is suitable to learn target \tilde{a} and that the controller manages to keep the visited state space bounded, the learning error (as a function of time t) should vanish.

Substituting different learning algorithms yields different adaptive controllers, e.g. *GP-MRAC* employs Gaussian process learning [20] to learn \tilde{a} ([9], [8]) and *LACKI-MRAC* employs LACKI [5].

In what is to follow, we utilise our LoKI method as the adaptive element and compare it against these established alternative learning methods. Following the nomenclature of the previous methods, we name the resulting adaptive controller *LoKI-MRAC*.

2) *Control of an F-4 fighter jet under wing rock*: As pointed out in [10], modern fighter aircraft designs are susceptible to lightly damped oscillations in roll known as “wing rock”. Commonly occurring during landing [22], removing wing rock from the dynamics is crucial for precision control of such aircraft. Precision tracking control in the presence of wing rock is a nonlinear problem of practical importance and has served as a test bed for a number nonlinear adaptive control methods [9], [18], [10].

We choose to replicate the experiments of Chowdhary et al. [9], [8]¹ for comparison. Using a realistic model of the roll dynamics of an F-4 fighter jet, the authors examined the task of using MRAC to perform a roll manoeuvre under uncertain wing rock. The task was to control the aircraft’s ailerons between times t_0 and t_f in order to cause the aircraft’s state trajectory $x : [t_0, t_f] \rightarrow \mathbb{R}^2$ to closely follow a roll manoeuvre prescribed by the reference trajectory $\xi(\cdot)$, with the first component of the state and reference being the roll angle and the second being the angular velocity.

Since wing rock can destabilise the dynamics, the authors proposed using GP-MRAC as a controller which allows to learn a model of the wing rock dynamics online. Choosing the feedback gains of the linear pseudo controller to be $K_1 = K_2 = 1$ (see [9] for more explanations), they demonstrated this could significantly improve tracking performance over competing methods.

In our experiments, we replicate their experimental setup with the only exception of setting the time increments to 0.01 s instead of 0.005 s to reduce overall experimental time and compare the performances of LoKI-MRAC, LACKI-MRAC and GP-MRAC. As a baseline comparison and to put our results into perspective, we also test the performance of a simple *PD*– controller with just the feedback gains (i.e. with $\nu_{ad} = 0$).

We create 30 randomised test runs of the wing rock dynamics and tracking problem and test each control algorithm on each one of them. For each of these test runs, the initial state $x(t_0)$ is drawn uniformly from $[0, 7] \times [0, 7]$, and the length scale of the squared exponential covariance function of each GP model is drawn uniformly from $[0.05, 2]$. We follow Chowdhary et al. [8], [10] and keep the hyperparameters of the GP fixed instead of optimising it, in order to cope with the computational real time constraints of online adaptation during runtime. In contrast to their work, we did not implement the GPs as sparse GPs. Instead, we utilised the standard Python implementation which is part of the scikit-learn package. For LoKI, we set $m = 10$. For both LACKI and LoKI, the Lipschitz constant estimate gets updated with each new sample point received. In addition, the partitioning

of the input space gets updated every 100 timesteps for LoKI by reevaluating the minimal and maximal values per dimension of all observed sample points. Fig. 2 illustrates this repartitioning.

The performance of all controllers across these randomised trials is depicted in Fig. 3. Each data point of each boxplot represent a performance measurement for one particular trial.

For each method, the figures show the boxplots of the following recorded quantities:

- *log-XERR*: average angular position error (log-deg), i.e. $\log(\frac{1}{t_f} \int_{t_0}^{t_f} \|\xi_1(t) - x_1(t)\| dt)$.
- *log-XDOTERR*: average roll rate error (log-deg/sec.), i.e. $\log(\frac{1}{t_f} \int_{t_0}^{t_f} \|\xi_2(t) - x_2(t)\| dt)$.
- *log-PREDERR*: average log-prediction error, i.e. $\log(\frac{1}{t_f} \int_{t_0}^{t_f} \|\hat{f}_n(x(t)) - f(x(t))\| dt)$ where f is a vector field affected by the wing rock.
- *log-CMD*: average cumulative control magnitude (log-scale), i.e. $\log(\frac{1}{t_f} \int_{t_0}^{t_f} \|u(t)\| dt)$.
- *log-max. RT (predictions)*: the log of the maximal run time (within time span $[t_0, t_f]$) each method took to generate a prediction ν_{ad} within the time span.
- *log-max. RT (learning)*: the log of the maximal run time (within time span $[t_0, t_f]$) it took each method to incorporate a new training example of the drift \tilde{a} .

Discussion: All three adaptive methods outperformed the simple *PD*– controller by magnitudes in terms of tracking error. Among the adaptive controllers, the tracking errors of LoKI-MRAC seem to have been slightly superior, as their outliers were lower than those of LACKI-MRAC and GP-MRAC, while the bulk of tracking errors were at similar levels.

More importantly, as expected by the theory provided in the previous section, LoKI-MRAC outperformed LACKI-MRAC both with regard to training time and prediction time. It also outperformed GP-MRAC in terms of training time, while being roughly comparable in terms of prediction time. If the controller was given a finite time budget for training and prediction at each time step, as one would expect in a real world control setting, this would make LoKI-MRAC the

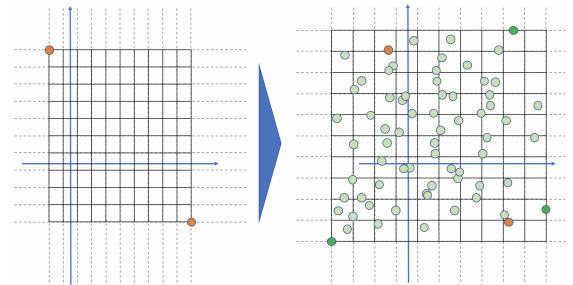


Fig. 2. Illustration of repartitioning with LoKI in online setting. Left side: initially, the first two points (orange) span the hyperrectangle H which is partitioned into m^d hyperrectangles ($t = t_0$). Right side: after 100 time steps, with 100 new data points (green) the partitioning is redone, this time 3 of the new points (darker green) span H ($t = t_{100}$). Dotted lines indicate the assignment of points to hyperrectangles beyond H as derived from previously observed sample points.

¹We are grateful to the authors for kindly providing their code.

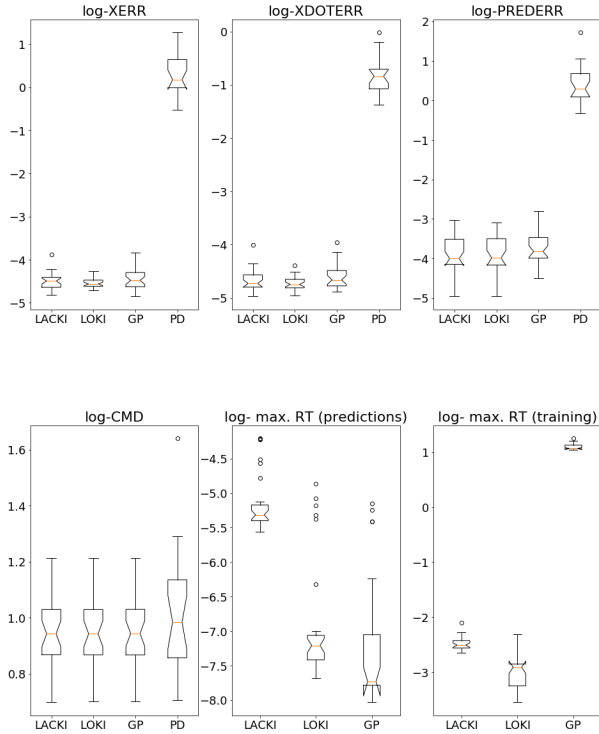


Fig. 3. Performance of the different online controllers over a range of 30 trials with randomised parameter settings and initial conditions. We see that LoKI exhibited similar or better performance than the other methods while markedly outperforming the other learning methods in terms of time required for learning.

controller of choice among the controllers tested. Overall, it can therefore be concluded that LoKI-MRAC exhibited superior performance compared to the other tested methods in our wing rock experiments and that LoKI therefore seems well-suited to be employed in control problems.

B. UCI data sets

While the wing rock dynamics control experiment indicates the advantages and suitability of LoKI for control problems, its dimensionality is rather low ($d = 2$). Also, the learning models are trained in an online setting, rather than from scratch. In order to demonstrate the computational speed-up that can be achieved with LoKI when training is done from scratch even in high-dimensional environments, we also test it on a series of machine learning benchmark regression data sets that are publicly available in the UCI data base.

Table I shows their characteristics. Each data set gets randomly split into 80% training and 20% test data before we center and normalise both using mean and variance of the training data. Subsequently, we train LoKI, LACKI, and a range of variations of GPs with different covariance functions (namely the automatic relevance determination version of the squared exponential, the Matern32 and the Matern52 kernels) on each of the resulting training sets. As further baseline, we include ordinary least squares (OLS) regression in all experiments.

For LoKI, we adjusted m^d , the number of hyperrectangle,

according to dimensionality and size of data set in order to get both enough reduction in terms of computational complexity and still enough (expected) number of samples per hyperrectangle to get a reliable estimate of the local Lipschitz constant. The chosen m is also included in Table I. GP hyperparameter training was done by marginal likelihood maximisation as implemented in the GPflow package with 5 randomised restarts. For data sets with more than 2,000 training points, we resorted to sparse approximations through variational inference as proposed in [25] using 250 inducing points, as on these sets the cubic computational cost of full GP training was prohibitive.

The results of our experiments are exhibited in Table II.

Discussion: As expected, LoKI was consistently (magnitudes) faster to train than (even sparse) GPs and LACKI, and its training time was almost as short as the OLS training time for many data sets. However, its prediction accuracy was often outperformed by the best prediction results obtained by GPs, which we suspect to be able to infer more structure in data (e.g. Yacht Dynamics, Energy Efficiency). Nonetheless, there are other data sets in which LoKI's prediction performance was comparable or better to the best performance obtained with GPs (e.g. Power Plant, Protein Structure) and other data sets where it drastically outperformed the worst GP results (Boston Housing, Kin8nm). We also note that the predictive performance of LoKI on the UCI data sets was always at least on par with the (global) LACKI method, and in some cases (Energy Efficiency), where smoothness in the data seems to vary locally, even superior. In general, LoKI seems to offer a sound compromise between the predictive accuracy that can be achieved with a GP, and the speed of training of OLS, on the data sets tested. Lastly, we should point out that GPflow implements GPs in highly efficient tensorflow code, while our LACKI and LoKI implementations are in native Python. Therefore, our runtime measurements were skewed in favor of the GPs. Hence, we would expect the runtime advantages of our LoKI method to be even more pronounced than they already were if it was implemented in a comparably efficient manner.

V. CONCLUSIONS

We have presented LoKI, an extension of recent work on nonparametric regression related to Lipschitz interpolation and NSM methods. In contrast to previous work,

TABLE I
CHARACTERISTICS OF THE UCI DATA SETS.

Data Set	Data Points	Dimensions	m for LoKI
Boston Housing	504	13	3
Concrete Strength	1,030	8	4
Energy Efficiency	768	8	2
Kin8nm	8,192	8	4
Power Plant	9,568	4	9
Protein Structure	45,730	9	6
Yacht Dynamics	308	6	2

TABLE II

TRAINING TIMES AND TEST SET RMSE FOR DIFFERENT UCI DATASETS. FOR GPS, BEST (WORST) TEST SET RMSE RESULTS OF ALL TRAINED GPS ARE GIVEN; RESULTS OBTAINED WITH SPARSE GPS ARE INDICATED BY *.

Data Set	Training time in s				RMSE			
	GP	LACKI	LoKI	OLS	GP	LACKI	LoKI	OLS
Boston Housing	48	4	<1	<1	0.30 (1.04)	0.44	0.46	0.57
Concrete Strength	50	20	1	<1	0.33 (0.34)	0.57	0.58	0.64
Energy Efficiency	40	9	<1	<1	0.04 (0.22)	0.25	0.13	0.28
Kin8nm	131*	943	1	<1	0.29 (1.00)*	0.47	0.49	0.76
Power Plant	228*	1,074	3	<1	0.22 (0.22)*	0.25	0.25	0.27
Protein Structure	1,233*	24,643	286	<1	0.68 (0.70)*	0.69	0.67	0.86
Yacht Dynamics	2	1	<1	<1	0.02 (0.03)	0.39	0.39	0.63

we proposed to estimate Lipschitz hyperparameters of the prediction model locally and in a manner that allows us to provide a bound to the expected computational effort for training as well as for prediction. We presented simulations of a learning-based model-reference adaptive control application illustrating that our approach can effectively learn a predictive dynamical system online, resulting in reliable control performance. Furthermore, our results demonstrate that, owing to its computational advantages, LoKI could operate at a much higher sample frequency as part of an online learning-based controller than competing nonparametric learning methods could. In a range of additional experiments on standard machine learning benchmark data sets, we have confirmed the computational advantages of LoKI even in higher dimensional settings and demonstrated that its predictive performance is at least on par with the global LACKI method also on more complex inference problems. In addition to its practical benefits, we provided some first worst-case consistency guarantees. Future work could readily translate online learning and control convergence guarantees, which exist for the related LACKI method, to our LoKI approach. In addition, we intend to convert our bounds on the expected computational effort into worst-case as well as into high-probability bounds and consider deployment in the context of robust data-based model-predictive control. Note, for estimation and prediction, our LoKI method only took training examples into account that fell into the same hyperrectangle as the query point. Future work could compare this approach to softer approaches that also take into account information from neighbouring subregions of input space.

REFERENCES

- [1] K. J. Aström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, 2nd edition, 2013.
- [2] Arno Blaas, Adam D Cobb, Jan-Peter Calliess, and Stephen J Roberts. Scalable bounding of predictive uncertainty in regression problems with SLAC. In *International Conference on Scalable Uncertainty Management*, pages 373–379. Springer, 2018.
- [3] J. Calliess. Lazily Adapted Constant Kinky Inference for Nonparametric Regression and Model-Reference Adaptive Control. *arXiv preprint arXiv:1701.00178*, 2016.
- [4] J. Calliess, S. J. Roberts, C. E. Rasmussen, and J. Maciejowski. Nonlinear set membership regression with adaptive hyper-parameter estimation for online learning and control. In *ECC*, 2018.
- [5] Jan-Peter Calliess. *Conservative decision-making and inference in uncertain dynamical systems*. PhD thesis, University of Oxford, 2014.
- [6] Jan-Peter Calliess. Lipschitz optimisation for Lipschitz interpolation. In *2017 American Control Conference (ACC)*, pages 3141–3146. IEEE, 2017.
- [7] M. Canale, L. Fagiano, and M. C. Signorile. Nonlinear model predictive control from data: a set membership approach. *Int. J. Robust Nonlinear Control*, 2014.
- [8] G. Cho, G. Chowdhary, A. Kingravi, J. P. . How, and A. Vela. A Bayesian nonparametric approach to adaptive control using Gaussian processes. In *CDC*, 2013.
- [9] Girish Chowdhary, H.A. Kingravi, J.P. How, and P.A. Vela. Bayesian nonparametric adaptive control using Gaussian processes. Technical report, MIT, 2013.
- [10] Girish Chowdhary, Hassan A. Kingravi, Jonathan How, and Patricio A. Vela. Nonparametric adaptive control of time-varying systems using Gaussian processes. In *American Control Conference (ACC)*, 2013.
- [11] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 1979.
- [12] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.
- [13] M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2013.
- [14] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, 2002.
- [15] D. Limon, J. Calliess, and Jan Marian Maciejowski. Learning-based nonlinear model predictive control. *IFAC-PapersOnLine*, 2017.
- [16] Jose Maria Manzano, Daniel Limon, David Muñoz de la Peña, and Jan Peter Calliess. Output feedback MPC based on smoothed projected kinky inference. *IET Control Theory & Applications*, 2019.
- [17] M. Milanese and C. Novara. Set membership identification of nonlinear systems. *Automatica*, 2004.
- [18] M.M. Monahemi and M. Krstic. Control of wingrock motion using adaptive feedback linearization. *J. of Guidance Control and Dynamics*, 1996.
- [19] E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 1964.
- [20] C.E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [21] Jacob Roll. *Local and piecewise affine approaches to system identification*. Univ., 2003.
- [22] A. A. Saad. *Simulation and Analysis of wing rock physics for a generic fighter model with three degrees of freedom*. PhD thesis, Air Force Institute of Technology, Air University, 2000.
- [23] R. G. Strongin. On the convergence of an algorithm for finding a global extremum. *Engineering in Cybernetics*, 1973.
- [24] A.G. Sukharev. Optimal method of constructing best uniform approximation for functions of a certain class. *Comput. Math. and Math. Phys.*, 1978.
- [25] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, 2009.
- [26] G. S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics*, 1964.