Nonparametric Adaptive Control using Gaussian Processes with Online Hyperparameter Estimation

Robert C. Grande, Girish Chowdhary, and Jonathan P. How

Abstract— **Many current model reference adaptive control methods employ parametric adaptive elements in which the number of parameters are fixed a-priori and the hyperparameters, such as the bandwidth, are pre-defined, often through expert judgment. Typical examples include the commonly used Radial Basis Function (RBF) Neural Networks (NNs) with pre-allocated centers. As an alternative to these methods, a nonparametric model using Gaussian Processes (GPs) was recently proposed. Using GPs, it was shown that it is possible to maintain constant coverage over the operating domain by adaptively selecting new kernel locations without any previous domain knowledge. However, even if kernel locations provide good coverage of the input domain, incorrect bandwidth selection can result in poor characterization of the model uncertainty, leading to poor performance. In this paper, we propose methods for learning hyperparameters online in GP-MRAC by optimizing a modified likelihood function. We prove the stability and convergence of our algorithm in closed loop. Finally, we evaluate our methods in simulation on an example of wing rock dynamics. Results show learning hyperparameters online robustly reduces the steady state modeling error and improves control smoothness over other MRAC schemes.**

I. INTRODUCTION

It is often infeasible to obtain an exact model of a system's dynamics, either due to cost or inherent model complexity. In order to represent the resulting modeling uncertainty online, a common approach is to use some set of basis functions and estimate the associated parameters. Following the pioneering work by Sanner and Slotine [22], Radial Basis Function Networks (RBFNs) are perhaps the most widely used adaptive elements when a basis for the modeling uncertainty is unknown. However, if the system operates outside of the domain of the RBF kernel's centers [18], or if the bandwidth of the kernel is not correctly specified [15, 24, 26], the modeling error will be high and stability cannot be guaranteed globally.

To overcome the aforementioned limitations of RBFNs and other fixed-parameter adaptive elements, we propose to use the Gaussian Process (GP) nonparametric adaptive elements [4]. A GP model is a data-driven model for Bayesian probabilistic inference [19] that has gained popularity in the control and robotics communities [12, 17]. The benefit of using a probabilistic framework in contrast to a deterministic framework is that stochasticity is included in the generative story of the data, and as such, noise and other non-deterministic effects can be explicitly accommodated.

In the author's past work, GP-MRAC has been used to address limitations of RBFNs with fixed, preallocated centers [4, 5]. In GP-MRAC, the kernel centers are allocated based on the data, thereby the controller automatically adjusts to the unknown operating domain. However, even if the kernel centers evolve to cover the input domain, using incorrect hyperparameters such as the kernel bandwidth and measurement noise variance can result in poor characterization of the model uncertainty and affect control performance [19, 26]. It can be shown theoretically using results from [19, Chapter 7], as well as empirically that moderate deviations in the hyperparameters from its optimal value can increase the expected modeling error significantly. Furthermore, selection of kernel centers using criteria in [4, 11] depends on the hyperparameters, and inaccurate values may result in poor domain coverage.

Learning hyperparameters in GPs in an online setting provides a unique set of challenges. Typically, it is assumed in the GP literature [13, 19, 25] that data is available in batch. In order to optimize the joint likelihood, as is common in the literature, multiple points must be used, so the gradient cannot be calculated at each time instant using only the current measurement as with RBFN [15, 24, 26]. Additionally, gradient calculations scale poorly with the number of data points, so one cannot use many points before losing online tractability. A significant amount of research has been pursued in sparsification to reduce computation time [13, 25]; however, these approaches only consider batch data. Therefore, we can only use a subset of the data to ensure online tractability. A random or windowed subset of data can be used, but this may lead to over fitting if points are spaced too far apart to capture correlations or too close such that any variation in measurements is due primarily to noise. Clearly, a more intelligent selection scheme must be used.

In this paper, we propose online hyperparameter optimization methods to enhance the robustness of GP-MRAC to wrong choice of hyperparamters. We also provide stability and convergence guarantees in presence of hyperparameter adaptation. Using a modified likelihood function, the hyperparameters are updated using stochastic gradient ascent. This modified reward function incorporates observations at locations which maximize information about the underlying GP as well as observations at each time instant to avoid overfitting. Alternative methods that have been proposed for online hyperparameter learning include the extended Kalman filter (EKF) [20] and particle filter (PF) [28]. However, the EKF is not guaranteed to converge, and the PF based estimator proposed in [28] uses batch sequential data. In both papers, the focus is on estimation only, and stability in closed loop control is not considered.

R. Grande, G. Chowdhary, and J. How are with the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology, Cambridge, MA.

II. Approximate Model Inversion based Model Reference Adaptive Control

AMI-MRAC is an approximate feedback-linearization based MRAC method that allows the design of adaptive controllers for a general class of nonlinear plants (see e.g. [2, 7]). The GP-MRAC approach is introduced in the framework of AMI-MRAC, although it should be noted that it is applicable to other MRAC architectures (see e.g. [16, 27]). Let $x(t) = [x_1^T(t), x_2^T(t)]^T \in D_x \subset \mathbb{R}^n$, such that $x_1(t) \in \mathbb{R}^{n_s}$, $x_2(t) \in \mathbb{R}^{n_s}$, and $n = 2n_s$. Let $\delta \in D_\delta \subset \mathbb{R}^l$, and consider the following multiple-input controllable control-affine nonlinear uncertain dynamical system

$$\begin{aligned} \dot{x}_1(t) &= x_2(t), \\ \dot{x}_2(t) &= f(x(t)) + b(x(t))\delta(t). \end{aligned} \tag{1}$$

The functions $f(0)$, $f(0) = 0$ and b are partially unknown functions assumed to be Lipschitz over a domain $\mathcal{D}$ and the control input δ is assumed to be bounded and piecewise continuous, so as to ensure the existence and uniqueness of the solution to (1) over $\mathcal{D}$. Also assume that $l \leq n_s$. Further note that while the development here is restricted to control-affine systems the AMI-MRAC framework can also be extended to a class of non-affine in control systems [8, 10]. The AMI-MRAC approach used here feedback linearizes the system by finding a pseudo-control input $\nu(t) \in \mathbb{R}^{n_s}$ that achieves a desired acceleration. If the exact plant model in (1) is known and invertible, the required control input to achieve the desired acceleration is computable by inverting the plant dynamics. Since this usually is not the case, an approximate inversion model $\hat{f}(x) + \hat{b}(x)\delta$, where $\hat{b}$ chosen to be nonsingular for all $x(t) \in D_x$, is employed.

Given a desired pseudo-control input $\nu \in \mathbb{R}^{n_s}$ a control command δ can be found by approximate dynamic inversion:

$$\delta = \hat{b}^{-1}(x)(\nu - \hat{f}(x)). \tag{2}$$

Let $z = (x^T, \delta^T)^T \in \mathbb{R}^{n+l}$ for brevity. The use of an approximate model leads to modeling error Δ for the system,

$$\dot{x}_2 = \nu(z) + \Delta(z), \tag{3}$$

with

$$\Delta(z) = f(x) - \hat{f}(x) + (b(x) - \hat{b}(x))\delta. \tag{4}$$

Were b known and invertible with respect to δ, then an inversion model exists such that the modeling error is not dependent on the control input δ. A designer chosen reference model is used to characterize the desired response of the system

$$\begin{aligned} \dot{x}_{1_{rm}} &= x_{2_{rm}}, \\ \dot{x}_{2_{rm}} &= f_{rm}(x_{rm}, r), \end{aligned} \tag{5}$$

where $f_{rm}(x_{rm}, r)$ denotes the reference model dynamics, assumed to be continuously differentiable in x_{rm} for all $x_{rm} \in D_x \subset \mathbb{R}^n$. The command $r(t)$ is assumed to be bounded and piecewise continuous. Furthermore, f_{rm} is assumed to be such that x_{rm} is bounded for a bounded command.

Define the tracking error to be $e(t) = x_{rm}(t) - x(t)$, and the pseudo-control input ν to be

$$\nu = \nu_{rm} + \nu_{pd} - \nu_{ad}, \tag{6}$$

consisting of a linear feed forward term $\nu_{rm} = \dot{x}_{2_{rm}}$, a linear feedback term $\nu_{pd} = [K_1, K_2]e$ with $K_1 \in \mathbb{R}^{n_s \times n_s}$ and $K_2 \in \mathbb{R}^{n_s \times n_s}$, and an adaptive term $\nu_{ad}(z)$. For ν_{ad} to be able to cancel Δ, the following assumption needs to be satisfied:

Assumption 1 There exists a unique fixed-point solution to $\nu_{ad} = \Delta(x, \nu_{ad})$, $\forall x \in D_x$.

Assumption 1 implicitly requires the sign of control effectiveness to be known [10]. Sufficient conditions for satisfying this assumption are available in [10].

Using (3) the tracking error dynamics can be written as

$$\dot{e} = \dot{x}_{rm} - \begin{bmatrix} x_2 \\ \nu + \Delta \end{bmatrix}. \tag{7}$$

Let $A = \begin{bmatrix} 0 & I \\ -K_1 & -K_2 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$, where $0 \in \mathbb{R}^{n_s \times n_s}$ and $I \in \mathbb{R}^{n_s \times n_s}$ are the zero and identity matrices, respectively. From (6), the tracking error dynamics are then,

$$\dot{e} = Ae + B[\nu_{ad}(z) - \Delta(z)]. \tag{8}$$

The baseline full state feedback controller ν_{pd} is chosen to make A Hurwitz. Hence, for any positive definite matrix $Q \in \mathbb{R}^{n \times n}$, a positive definite solution $P \in \mathbb{R}^{n \times n}$ exists for the Lyapunov equation

$$0 = A^T P + PA + Q. \tag{9}$$

III. Adaptive Control using Gaussian Process Regression

Consider the uncertainty $\Delta(z)$ in (4). Traditionally in MRAC, it has been assumed that the uncertainty is a (smooth) deterministic function. Here we offer an alternate view of modeling the uncertainty $\Delta(z)$ as a Gaussian Process (GP). A GP is defined as a collection of random variables, any finite subset of which has a joint Gaussian distribution [19] with mean $m(x)$ and covariance $k(x(t'), x(t))$. For the sake of clarity of exposition, we will assume that $\Delta(z) \in \mathbb{R}$; the extension to the multidimensional case is straightforward. Hence, the uncertainty $\Delta(z)$ here is specified by its mean function $m(z) = \mathbb{E}(\Delta(z))$ and its covariance function $k(z, z') = \mathbb{E}[(\Delta(z) - m(z))(\Delta(z') - m(z'))]$, where z' and z represent different points in the state-input domain. We will use the notation: $\Delta(z) \sim \mathcal{GP}(m(z), k(z, z'))$. See [19] for a complete analysis of the properties of GPs.

A. GP Regression

Given some set of noisy data points $y^T = [y_1, y_2, \ldots, y_n]$ at corresponding locations $Z = [z_1, z_2, \ldots, z_i]$, we would like to predict the expected value y_{i+1} at some new location z_{i+1}. As is common in the GP literature, we assume that the GP has a zero mean. In general this assumption is not limiting since the posterior estimate of the latent function

is not restricted to zero. Using Bayes law and properties of Gaussian distributions, the conditional probability can be calculated as a normal variable [19] with mean

$$m(y_{i+1}) = \alpha^T k(Z, z_{i+1}), \tag{10}$$

where $\alpha = [K(Z,Z) + \omega_n^2 I]^{-1} y$ are the kernel weights, and covariance

$$\begin{aligned} \mathbf{\Sigma}(z_{i+1}) \quad &= \quad k(z_{i+1}, z_{i+1}) \\ &\quad -\bar{k}^T(Z, z_{i+1})[K(Z,Z) + \omega_n^2 I]^{-1} \bar{k}(Z, z_{i+1}). \end{aligned} \tag{11}$$

where the elements of $K(Z,Z)$ are defined as $K_{l,m} = k(z_l, z_m)$, and $k(Z, z_{i+1}) \in \mathbb{R}^i$ denotes the kernel vector corresponding to the $i+1^{th}$ measurement. Due to Mercer's theorem and the addition of the positive definite matrix $\omega_n^2 I$, the matrix inversion in (10) and (11) is well defined. Furthermore, the weights α and covariance (11) can be recomputed recursively with a rank-1 update [4, 6].

B. Hyperparameter Optimization

Prior to this point it has been assumed that the kernel function $k(\cdot,\cdot)$ is fully specified by the user and is static with respect to time. This approach is guaranteed to be stable [4, 11], but may perform poorly if the design hyperparameters are far from the optimal values, or if the optimal hyperparameters are time varying.

In this section, we outline general results for gradient based optimization in GPs. In order to learn the most likely hyperparameters, we would ideally maximize the posterior estimate. In general, the relationship between the posterior estimate of the hyperparameters $\theta = (\theta_1, \ldots, \theta_q)$ and the data likelihood can be derived from Bayes' rule as $P(\theta \mid Z, y) \propto P(y \mid \theta, Z) P(\theta)$. The calculation of $P(\theta)$ at each iteration is intractable in general, so instead we maximize the data likelihood, as is popular in GP literature [19]. Equivalently, we can maximize the log likelihood,

$$\log P(y \mid \theta, Z) = -\frac{1}{2}\left(y^T \Sigma_Z^{-1} y + \log\det \Sigma_Z + N\log(2\pi)\right) \tag{12}$$

where $\Sigma_Z = (K(Z,Z) + \omega_n^2 I)$. Taking the derivative with respect to θ and rearranging yields

$$\frac{\partial}{\partial \theta_j} \log P(y \mid \theta, Z) = \frac{1}{2}\text{tr}\left((\bar{a}\bar{a}^T - \Sigma^{-1})\frac{\partial \Sigma}{\partial \theta_j}\right) \tag{13}$$

where $\bar{a} = \Sigma_Z^{-1} y$. Using the closed form of the gradient, one can choose any gradient based optimization method. In the next section, we introduce a modified joint likelihood function and perform optimization using a variable step size.

C. GP nonparametric model based MRAC

In this section, we describe GP-MRAC with online hyperparameter optimization. We let $\hat{(\cdot)}$ denote the use of a kernel with estimated hyperparameters. The adaptive element ν_{ad} is modeled as

$$\nu_{ad}(z) \sim \mathcal{N}(\hat{m}(z), \hat{\Sigma}_Z(z)), \tag{14}$$

where $\hat{m}(z)$ is the estimate of the mean of $\Delta(z)$ at the current time instant [4]. In this case, one can choose $\nu_{ad} = \hat{m}(z)$ or draw from the distribution in (14).

Traditional GP regression (see [19]) can become intractable in an online setting. Hence, instead of placing a kernel at every data point, we adopt the notion of placing kernels at a set of a budgeted *active basis vector set* $\mathcal{BV}$ which adequately cover the domain [4]. From an information theoretical standpoint, if the conditional entropy, which is given by the natural log of the conditional covariance (11), exceeds a certain threshold ϵ_{tol}, we add the point to the current budget. Equivalently, we can use the conditional variance,

$$\gamma_{t+1} = \hat{k}(z_{t+1}, z_{t+1}) - \hat{k}(Z, z_{t+1})^T \hat{\Sigma}_Z^{-1} \hat{k}(Z, z_{t+1}). \tag{15}$$

GPs have deep connections with Reproducing Kernel Hilbert spaces (RKHS) [19], and this scoring criteria is equivalent to the kernel linear independence test (see e.g. [6, 11]).

Once a preset budget has been reached, we delete basis vectors from $\mathcal{BV}$ to accommodate new data according to one of two possible methods. In the first, denoted as **OP**, one simply deletes the oldest vector: this is the simplest scheme and is intuitively appealing Chowdhary et al. have shown that this scheme works best in situations where the uncertainty in (4) is time varying [5]. In the second scheme, denoted as **KL**, we compute the KL divergence between the current GP and the $t+1$ GPs missing one vector each, and select the one with the greatest KL divergence. Intuitively, this approach deletes the point which contributes the least uncertainty reduction to the GP model. This can be efficiently computed based on approximations to the KL divergence [6].

To ensure robust performance using GP-MRAC, the hyperparameters must be optimized online. As stated previously, a subset of data must be intelligently selected to prevent over fitting as well as ensure convergence. Ideally, we would like to pick points which maximize the mutual information between the data points and with the underlying GP over a specified domain. A natural selection, then, is the set of observations at the basis vector locations. The basis vector locations are chosen such that conditional entropy is minimized and KL divergence is maximized, both of which increase mutual information. In order to prevent over fitting to a specific data set, we also will consider measurements available at each time instant.

Consider the modified joint probability function,

$$\log P(y|Z,\theta) \approx \sum_{i \notin |\mathcal{BV}|} \log P(y_Z, y_i | Z, Z_i, \theta) \tag{16}$$

where y_i is the observation at the current time instant. This modification assumes new measurements at each time instant are independent of each other, but facilitates convergence analysis. At each step, we calculate the derivative of the joint likelihood using (13). We propose the parameter update,

$$\theta_j[i+1] = \theta_j[i] + b[i]\frac{\partial}{\partial \theta_j} \log P(y_Z, y_{i+1} | Z, Z_{i+1}, \theta) \tag{17}$$

with the following two candidate functions for $b[i]$,

$$b[i+1] = \frac{i}{i+1} b[i] \tag{18}$$

$$b[i+1] = (1-\delta) b[i] + \eta \tag{19}$$

where b is a dynamic variable which controls the learning rate of θ_j, and $\eta \ll 1$ and $\delta \ll 1$ are parameters chosen by the user to control convergence.

Lemma 1 Given the update equations, (17) and (18), the parameter estimate θ will almost surely converge to a local maxima of the modified joint likelihood function (16).

Proof: Eq. 17 is the update step of a stochastic gradient ascent algorithm with respect to the modified joint likelihood (16). Assuming mild conditions about the smoothness of (12), θ converges almost surely to a local optima if $\sum_{i=1}^{\infty} b[i]^2 < \infty$, and $\sum_{i=1}^{\infty} b[i] = \infty$ [1]. It can be easily verified that (18) satisfy these conditions. ∎

If one believes the hyperparameters are static, then (18) should be used to guarantee convergence. However, if the model uncertainty is time varying, it's possible that the hyperparameters are time varying as well. In this case, it may be advantageous to use (19). Equation (19) can be is an EKF inspired estimator: $b[i]$ represents the fictitious certainty in θ and η represents fictitious process noise.

The budgeted GP regression algorithm in [4, 6] assumed static hyperparameters. As the hyperparameter values are changed by the update equations, the values computed from the linear independence test (15) no longer reflect the current model. Therefore, some basis vector points which were previously thought to be roughly independent are now sufficiently correlated in the new model, and should be removed. However, naïvely recomputing the linear independence test requires $O(|\mathcal{BV}|^3)$ operations, so as an alternative, we check to see if the conditional entropy between two points is below ϵ_{tol}, which requires $O(|\mathcal{BV}|^2)$ operations, and show that this never removes too many points.

Lemma 2 Given the set of points $Z \in \mathcal{BV}$ selected by the threshold test (15) using the kernel $k_0(\cdot,\cdot)$, and given some new kernel $k_1(\cdot,\cdot)$, it is possible $\exists$ some $z_i \in Z$ s.t. $\gamma_i < \epsilon_{tol}$ from (15). Furthermore, if $\exists j$ s.t. $\hat{\Sigma}^2_{i|j} < \epsilon_{tol}$, then $\gamma_i < \epsilon_{tol}$.

Proof: Proof by example. Let $k_0(z_i, z_j)_l = \exp\left(\frac{-\|z_i - z_j\|^2}{2}\right)$ and $\hat{\Sigma}^2_{i|j} > \epsilon_{tol}$. Given the new kernel $k_1(z_i, z_j)_l = \exp\left(-(0)\|z_i - z_j\|^2\right)$, then $\hat{\Sigma}^2_{i|j} = 0 < \epsilon_{tol}$. For the second part, conditional variance is strictly decreasing and monotonic as a function of number of conditioning variables. That is, conditional variance never increases given more conditioning variables. Mathematically, $\forall k$, $\hat{\Sigma}^2_{i|j,k} \leq \hat{\Sigma}^2_{i|j}$. Thus, if $\hat{\Sigma}^2_{i|j} < \epsilon_{tol}$, then $\gamma_i = \hat{\Sigma}^2_{i|Z} < \epsilon_{tol}$. ∎

In order to remove points, it is sufficient to iterate over all points once as described in Algorithm 2. After the budget is updated, the weights are reinitialized to

$$\alpha = \Sigma_Z^{-1} y \tag{20}$$

Algorithm 1 The Generic Gaussian Process - Model Reference Adaptive Control (GP-MRAC) algorithm

while new measurements are available **do**
  Call Algorithm 2
  Given z_{t+1}, compute γ_{t+1} by (15)
  Compute $y_{t+1} = \dot{x}_{2_{t+1}} - \nu_{t+1}$
  if $\gamma_{t+1} > \epsilon_{tol}$ **then**
    Store $Z(:, t+1) = z(t+1)$
    Store $y_{t+1} = \dot{x}_2(t+1) - \nu(t+1)$
    Calculate $K(Z, Z)$ and increment $t+1$
  end if
  if $|\mathcal{BV}| > p_{max}$ **then**
    Delete element in $\mathcal{BV}$ per methods in Section III-C
  end if
  Update α recursively per methods in [6]
  Calculate $\hat{k}(Z, z(t+1))$
  Calculate $\hat{m}(z(t+1))$ and $\hat{\Sigma}(z_{t+1})$ using (10) and (11)
  Set $\nu_{ad} = \hat{m}(z(t))$
  Calculate pseudo control ν using (6)
  Calculate control input using (2)
end while

Algorithm 2 Update Hyperparameters

Calculate the derivative of the log likelihood per (13)
$\forall j$, Update $\theta_j[i+1]$ and $b[i+1]$ per (17)
if $\|\theta_j[i+1] - \theta_j[i]\| / \|\theta_j[i]\| >$threshold% **then**
  Update θ_j
  for all $z_i, z_j \in \mathcal{BV}$ **do**
    Calculate $\hat{\Sigma}^2_{i|j}$
    if $\hat{\Sigma}^2_{i|j} < \epsilon_{tol}$ **then**
      Delete z_i,y_i
    end if
  end for
  Recalculate covariance $\hat{\Sigma}_Z$ and weights $\alpha = \hat{\Sigma}_Z^{-1} y$
end if

to account for the new hyperparameter and kernel function values. Lastly, it is undesirable to recalculate the weights and kernel matrices if the hyperparameters change only slightly. We therefore add the constraint in practice that the model hyperparameters are only updated if the change exceeds some designated percentage. For example, in the simulations presented in Section IV, a threshold of 1% was used. We conclude with the generalized GP-MRAC algorithm with hyperparameter update in Algorithm 1.

D. Analysis of Stability

In this section we establish results relating to the stability of Gaussian process based MRAC using the online GP sparsification algorithm described in the companion paper [4]. In particular, the boundedness of the tracking error dynamics of (8) is shown, which is represented using the GP adaptive element:

$$de = Ae\ dt + B(\epsilon_m(z) - G\ d\xi), \tag{21}$$

where $\nu_{ad}(z) \sim \mathcal{GP}(\hat{m}(z), k(z,z'))$, $G = \mathbb{V}(\Delta(z))$ is the variance of $\Delta(z)$, and $\epsilon_m(z) = \hat{m}(z) - m(z)$. Note that we have used a Wiener process representation of the GP, that is we assume that a draw from the GP $\Delta(z)$ can be modeled as $m(z) + G\xi$. While other GP representations can be used, this approach facilitates our analysis. In the first lemma, we show that if the hyperparameters of our model $\hat{k}(z_i, z_j)$ are incorrect, the modeling error is bounded when reinitializing the weights. Once the weights have been initialized, and the hyperparameters remain constant, the system is guaranteed to be stable [4]. Thus, GP-MRAC is stable for all time while adapting hyperparameters.

Lemma 3 Let $\Delta(z)$ be represented by a Gaussian process, and $\hat{m}(z)$ be defined as in (10). Then $c_1 := \|\Delta(z) - \hat{m}(z)\|$ is almost surely (a.s.) bounded for any choice of hyperparameters θ, if α is reinitialized as in (20)

Proof: Let Z represent the set of basis vectors selected by either the OP or the KL variants of Algorithm 1 at the instant σ. Let $|\mathcal{BV}|$ denote the cardinality of $\mathcal{BV}$. The following proof holds true for any valid choice of Mercer kernel with $\sup k(\cdot,\cdot) = k_{max}$. Using the definition of GP regression, $\hat{m}(z) = \hat{k}(Z,z)^T \hat{\Sigma}_Z^{-1} y$, where y is the set of observations at the basis vectors, the error can be bounded as,

$$\begin{aligned}\|\Delta(z) - \hat{m}(z)\| &\leq \|\Delta(z)\| + \left\|\hat{k}(Z,z)^T \hat{\Sigma}_Z^{-1} y\right\| \\ &\leq \|\Delta(z)\| + \sigma_{max}\left(\hat{k}(Z,z)^T\right)\sigma_{max}\left(\hat{\Sigma}_Z^{-1}\right)\|y\|\end{aligned}$$

where $\sigma_{max}(\cdot)$ is the maximum singular value. Since the kernel function is bounded from above, it follows that $\sigma_{max}\left(\hat{k}(Z,z)^T\right) = |\mathcal{BV}| k_{max}$. Additionally, $\hat{\Sigma}_Z^{-1}$ a symmetric matrix, so the singular values are given by the eigenvalues. Due to the addition of the positive definite term $\omega_n^2 \mathbb{I}$, we have that $\lambda_{min}(\hat{\Sigma}_Z) \geq \omega_n^2$, so $\lambda_{max}(\hat{\Sigma}_Z^{-1}) \leq \omega_n^{-2}$. Plugging in,

$$\|\Delta(z) - \hat{m}(z)\| \leq \|\Delta(z)\| + |\mathcal{BV}| k_{max}(\omega_n^{-2})\,\|y\| \quad (22)$$

We assume that the function $\Delta(z)$ is finite at any given point and that y is a finite number of observations of finite values. Therefore, the right hand is bounded from above by a constant $\|\Delta(z) - \hat{m}(z)\| \leq \bar{c}$. ∎

The next Lemma shows that if Algorithm 1 adds or removes kernels from the basis set $\mathcal{BV}$ to keep a metric of the representation error bounded or if the hyperparameters change, $\epsilon_m(z)$ is also bounded.

Lemma 4 Let $\Delta(z)$ be represented by a Gaussian process, and $\hat{m}(z)$ be defined as in (10) inferred based on sparsified data, and let $m(z)$ be the mean of the GP without any sparsification, then $\epsilon_m(z) := m(z) - \hat{m}(z)$ is bounded almost surely.

Proof: From Equation (22) in [6] and from the non-parametric Representer theorem (see Theorem 4 of [23])

$$\|\epsilon_m(z)\| = \frac{\|\Delta(z) - \hat{m}(z)\|}{\omega^2}\|k_{t+1}^* - k_{z_{t+1}}^T K_t^{-1} k_{z_{t+1}}\|, \quad (23)$$

where $K_t := K(Z_t, Z_t)$ over the basis set $\mathcal{BV}$. The second term $\|k_{t+1}^* - k_{z_{t+1}}^T K_t^{-1} k_{z_{t+1}}\|$ is bounded above by ϵ_{tol} due to Algorithm 1. Using Lemma 3 it follows that

$$\|\epsilon_m(z)\| \leq \frac{\bar{c}}{\omega^2}\epsilon_{tol}, \quad (24)$$

∎

The boundedness of the tracking error can now be proven.

Theorem 1 Consider the system in (1), the reference model in (5), the control law of (6) and (2). Let the uncertainty $\Delta(z)$ be represented by a Gaussian process, then Algorithm 1 in and the adaptive signal $\nu_{ad}(z) = \hat{m}(z)$ guarantees that the system is mean square ultimately bounded a.s. inside a compact set.

Proof: Let $V(e(t)) = \frac{1}{2}e^T(t)Pe(t)$ be the Stochastic Lyapunov candidate, where $P > 0$ satisfies the Lyapunov equation (9). It follows that $\frac{1}{2}\lambda_{\min}P\|e\|^2 \leq 2V(e) \leq \frac{1}{2}\lambda_{\max}P\|e\|^2$. The Itô differential of the Lyapunov candidate along the solution of (21) for the σ^{th} system is

$$\begin{aligned}\mathcal{L}V(e) &= \sum_i \frac{\partial V(e)}{\partial e_i} A e_i + \frac{1}{2}\mathrm{tr}\sum_{i,j} BG(BG)^T \frac{\partial^2 V(e)}{\partial e_j e_i} \\ &= -\frac{1}{2}e^T Q e + e^T P B \epsilon_m(z) + \frac{1}{2}\mathrm{tr} BG(BG)^T P.\end{aligned}$$

Letting $c_6 = \frac{1}{2}\|P\|\|BG\|$, $c_7 = \|PB\|$, it follows from Lemma 4 that

$$\mathcal{L}V(e) \leq -\frac{1}{2}\lambda_{\min}(Q)\|e\|^2 + c_7\|e\|c_{8_\sigma} + c_6 \quad (25)$$

a.s. and where $c_{8_\sigma} = \frac{\bar{c}}{\omega^2}\epsilon_{tol}$. Therefore outside of the compact set $\|e\| \geq \frac{-c_7 c_{8_\sigma} + \sqrt{c_7^2 c_{8_\sigma}^2 + 2\lambda_{\min}(Q)c_6}}{\lambda_{\min}(Q)}$, $\mathcal{L}V(e) \leq 0$ a.s. Since this is true for all σ, (21) is mean square uniformly ultimately bounded in probability inside of this compact set a.s. (see [9]).

∎

Lemma 3 strengthens the theoretical results of past GP-MRAC results in [3, 5, 11]. Past work [3, 4, 11] assume that the model uncertainty $\Delta(z)$ can be described by a static, known, kernel function. In this work, it was shown that selecting new kernel locations and updating the kernel weights α would not cause the tracking error to become unbounded, even with suboptimal hyperparameters. We have also shown that if we reinitialize the kernel function to any Mercer kernel function, the modeling and tracking error is bounded. Combining these results, it follows that GP-MRAC with hyperparameter optimization maintains previous stability guarantees.

IV. Application to Trajectory Tracking in Presence of Wingrock Dynamics

Modern highly swept-back or delta wing fighter aircraft are susceptible to lightly damped oscillations in roll known as "Wing rock". Wing rock often occurs at conditions commonly encountered at landing ([21]), making precision control in presence of wing rock critical. In this section, we compare the performance between GP-MRAC with hyperparameter learning (GP-MRAC-HP), GP-MRAC with static

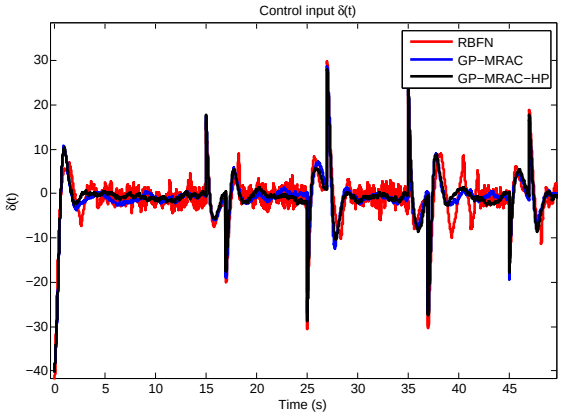

Fig. 1. Sample output of the control signal $\delta(t)$ for the different algorithms. GP-MRAC has much smoother control signals in comparison to RBFN.

hyperparameters (GP-MRAC)[4], and traditional RBFN with a fixed kernel bandwidth. Let θ denote the roll attitude of an aircraft, p denote the roll rate and δ_a denote the aileron control input. Then a model for wing rock dynamics is [14]

$$\dot{\theta} = p \tag{26}$$

$$\dot{p} = L_{\delta_a}\delta_a + \Delta(x), \tag{27}$$

$$\Delta(x) \quad = W_0^* + W_1^*\theta + W_2^* p + W_3^*|\theta|p + W_4^*|p|p + W_5^*\theta^3 \tag{28}$$

where $\Delta(x)$ is the unknown model error and $L_{\delta_a} = 3$. The parameters for wing rock motion are $W_1^* = 0.2314, W_2^* = 0.6918, W_3^* = -0.6245, W_4^* = 0.0095, W_5^* = 0.0214$, with trim error $W_0^* = 0.8$. The chosen inversion model has the form $\nu = \frac{1}{L_{\delta_a}}\delta_a$. This choice results in the mean of the modeling uncertainty of (4) being given by $\Delta(x)$. The inverting controller of (6) is used in the framework of the GP-MRAC algorithm (Algorithm 1). The gain for the linear part of control law (ν_{pd}) is given by $[1.2, 1.2]$. A second order reference model with natural frequency of $1\frac{\text{rad}}{\text{sec}}$ and damping ratio of 0.5 is chosen. The simulation uses a time-step of 0.05 sec. The maximum budget (p_{max}) was set to 100 with threshold $\epsilon_{tol} = 1\text{e}{-4}$ for (15). State measurements were corrupted with Gaussian white noise with standard deviation $\omega_n = 0.03$. For GP-MRAC-HP, update law (18) was used with $b[1] = 0.1$. Additionally, hyperparameter adaptation was not started until a quarter of the budget was added to the basis vector set.

For GP-MRAC, we adopted the kernel $\hat{k}(z_i, z_j) = \exp\left(\frac{-\|z_i - z_j\|^2}{2\mu^2}\right)$. For our analysis, we fixed the noise of the model to $\omega_n = 0.03$ and ran trials with bandwidth μ initialized at 11 different values spaced logarithmically between 0.12 and 12, with $\mu \approx 1.2$ yielding optimal performance for GP-MRAC. Due to numerical instability reasons, the RBFN was not evaluated at the last three values of μ. Each trial was run 10 times and averaged to receive statistics. Fig. IV shows an example of the control signal for the different algorithms.

As expected, learning hyperparameters online creates a more robust adaptive control algorithm that requires very little domain knowledge. Regardless of initialization value, GP-MRAC-HP performs significantly better in characterizing the model uncertainty. As seen in the first figure of Fig. 2, using hyperparameter learning, the error in $\|\Delta(z) - \hat{m}(z)\|$ is nearly independent of the bandwidth initialization values, and is a full order of magnitude less than the other two methods at certain points. As μ becomes very large, it takes longer for the algorithm to choose basis vector points and hyperparameter optimization starts later, resulting in higher mean error over the length of the trial. However, this error is transient and if given sufficient time, the mean error will approach the baseline seen for lower initial μ. The error for the RBFN approach is the largest.

Tracking error is similar between all approaches; GP-MRAC-HP differs from RBFN by less than 10% on average. While the tracking error is comparable, GP-MRAC and GP-MRAC-HP use significantly less control effort and have smoother control signals than RBFN. With a more accurate characterization of $\Delta(z)$, ν_{ad} can better predict $\Delta(z)$, resulting in improved control efficiency. On average RBFN uses about 35-40% more effort $\|\delta(t)\|$ than GP-MRAC. In terms of the control smoothness, or the mean control derivative $\|\dot{\delta}(t)\|$, GP-MRAC-HP offers the best performance with regards to smoother control inputs as well as robustness to incorrect hyperparameter initialization. The right figure of Fig. 2 shows that GP-MRAC-HP outperforms RBFN by a factor of 5 at times and outperforms GP-MRAC by 70% when the hyperparameters are far from optimal. Furthermore, in all graphs, the objective function remains nearly constant for GP-MRAC-HP, demonstrating its general robustness to suboptimal initialization values. In summary, these results indicate that GP-MRAC-HP enhances the robustness of GP-MRAC to hyperparameter initialization and outperforms RBFN-MRAC in many metrics.

V. Conclusion

Gaussian Process Model Reference Adaptive Control (GP-MRAC) architecture builds upon the idea of using Bayesian nonparameteric probabilistic framework, such as GPs, as an alternative to deterministic adaptive elements to enable adaptation with little or no prior domain knowledge. In this work we extended GP-MRAC by enabling online adaptation of the GP kernel hyperparameters. The hyperparameters were updated online using stochastic gradient ascent with a modified joint likelihood function. We proved the convergence of the hyperparameter values to a local optima. The budgeted online GP regression scheme in our companion paper [4] was modified to account for hyperparameter adaptation by removing kernels from the basis vector set that are no longer informative. Lastly, we proved that the tracking error remains bounded when optimizing hyperparameters. Numerical simulations show that GP-MRAC with hyperparameter learning outperforms fixed hyperparameter RBFN and GP-MRAC in terms of characterizing model uncertainty and generating a smoother, more efficient control signal.

Acknowledgment

This research was supported in parts by ONR MURI Grant N000141110688 and by AeroJet.

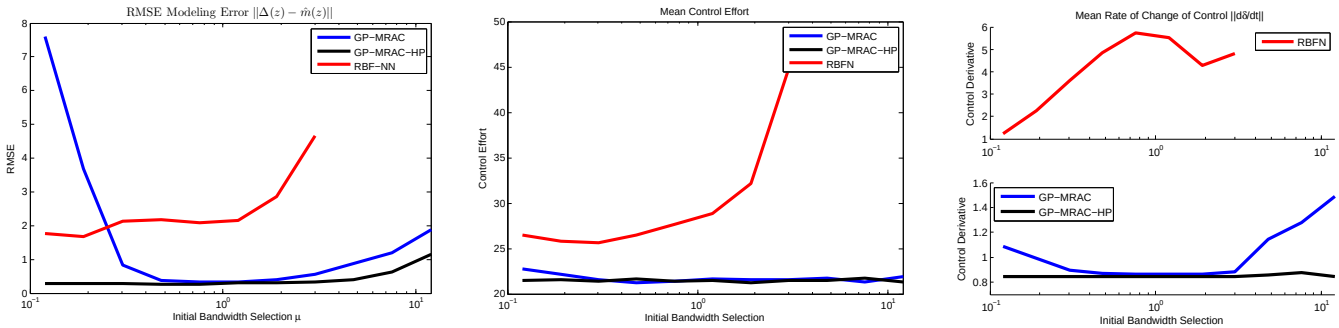

Fig. 2. Comparison of RMSE for $\|\Delta(z) - \hat{m}(z)\|$ (on left), control effort $\|\delta(t)\|$(center), and rate of change of the controller $\|\dot{\delta}(t)\|$ (on right), as a function of bandwidth initialization value. We compare performance for 11 different bandwidth values spaced logarythmically from 0.12 to 12, the optimal bandwidth being about 1.2. 10 simulations were averaged for each bandwidth value. GP-MRAC-HP significantly outperforms other methods in terms of model learning and smoothness of control signal regardless of initialization values.

REFERENCES

[1] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17:9, 1998.

[2] A. Calise, N. Hovakimyan, and M. Idan. Adaptive output feedback control of nonlinear systems using neural networks. *Automatica*, 37(8):1201–1211, 2001. Special Issue on Neural Networks for Feedback Control.

[3] G. Chowdhary, J. How, and H. Kingravi. Model reference adaptive control using nonparametric adaptive elements. In *Conference on Guidance Navigation and Control*, Minneapolis, MN, August 2012. AIAA. Invited.

[4] G. Chowdhary, H. Kingravi, J. How, and P. Vela. A bayesian nonparameteric approach to adaptive control using gaussian processes. In *IEEE Conference on Decision and Control (CDC)*. IEEE, 2013.

[5] G. Chowdhary, H. Kingravi, J. How, and P. Vela. Nonparameteric adaptive control of time varying systems using gaussian processes. In *American Control Conference*. IEEE, 2013.

[6] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[7] E. Johnson and S. Kannan. Adaptive trajectory control for autonomous helicopters. *Journal of Guidance Control and Dynamics*, 28(3):524–538, May 2005.

[8] S. Kannan. *Adaptive Control of Systems in Cascade with Saturation*. PhD thesis, Georgia Institute of Technology, Atlanta Ga, 2005.

[9] R. Khasminksii. *Stochastic Stability of Differential Equations*. Springer, Berlin, Germany, 2 edition, 2012.

[10] N. Kim. *Improved Methods in Neural Network Based Adaptive Output Feedback Control, with Applications to Flight Control*. PhD thesis, Georgia Institute of Technology, Atlanta Ga, 2003.

[11] H. A. Kingravi, G. Chowdhary, P. A. Vela, and E. N. Johnson. Reproducing kernel hilbert space approach for the online update of radial bases in neuro-adaptive control. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(7):1130 –1141, july 2012.

[12] J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[13] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[14] M. M. Monahemi and M. Krstic. Control of wingrock motion using adaptive feedback linearization. *Journal of Guidance Control and Dynamics*, 19(4):905–912, August 1996.

[15] F. Nardi. *Neural Network based Adaptive Algorithms for Nonlinear Control*. PhD thesis, Georgia Institute of Technology, Atlanta, GA 30332, Nov 2000.

[16] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, 1989.

[17] D. Nguyen-Tuong, M. Seeger, and J. Peters. Local gaussian process regression for real time online model learning and control. In *Advances in Neural Information Processing Systems 22 (NIPS 2008), Cambridge, MA: MIT Press*, 2009.

[18] J. Park and I. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computatations*, 3:246–257, 1991.

[19] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[20] S. Reece and S. Roberts. An introduction to gaussian processes for the kalman filter expert. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–9. IEEE, 2010.

[21] 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, Wright-Patterson Air Force Base, Dayton, Ohio, 2000.

[22] R. Sanner and J.-J. Slotine. Gaussian networks for direct adaptive control. *Neural Networks, IEEE Transactions on*, 3(6):837 –863, nov 1992.

[23] B. Scholkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In D. Helmbold and B. Williamson, editors, *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 416–426. Springer Berlin / Heidelberg, 2001.

[24] P. Shankar. *Self-Organizing radial basis function networks for adaptive flight control and aircraft engine state estimation*. Ph.d., The Ohio State University, Ohio, 2007.

[25] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT press, 2006.

[26] N. Sundararajan, P. Saratchandran, and L. Yan. *Fully Tuned Radial Basis Function Neural Networks for Flight Control*. Springer, 2002.

[27] G. Tao. *Adaptive Control Design and Analysis*. Wiley, New York, 2003.

[28] Y. Wang and B. Chaib-draa. A marginalized particle gaussian process regression. In *Advances in Neural Information Processing Systems 25*, pages 1196–1204, 2012.