MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

A Kalman Filter for Online Calibration of Optimal Controllers

Menner, Marcel; Berntorp, Karl; Di Cairano, Stefano

TR2021-091 August 10, 2021

Abstract

This paper proposes an approach for the calibration of the cost function of optimization-based controllers. The approach uses a Kalman filter that estimates the cost function parameters using data of closed-loop system operation. It adapts the parameters online and robustly, provides safety guarantees, is computationally efficient, has low data storage requirements, and is easy to implement making it appealing for many real-time applications. The approach provides a dataefficient alternative to Bayesian optimization and an automated alternative to learning from demonstrations. Simulation results show that the approach is able to learn cost function parameters quickly (approximately 95% faster than Bayesian optimization), is able to adapt the parameters to compensate for disturbances (approximately 25% improvement on tracking precision), and is robust to noise.

IEEE Conference on Control Technology and Applications (CCTA)

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

A Kalman Filter for Online Calibration of Optimal Controllers

Marcel Menner, Karl Berntorp, and Stefano Di Cairano

***Abstract*— This paper proposes an approach for the calibration of the cost function of optimization-based controllers. The approach uses a Kalman filter that estimates the cost function parameters using data of closed-loop system operation. It adapts the parameters online and robustly, provides safety guarantees, is computationally efficient, has low data storage requirements, and is easy to implement making it appealing for many real-time applications. The approach provides a data-efficient alternative to Bayesian optimization and an automated alternative to learning from demonstrations. Simulation results show that the approach is able to learn cost function parameters quickly (approximately 95% faster than Bayesian optimization), is able to adapt the parameters to compensate for disturbances (approximately 25% improvement on tracking precision), and is robust to noise.**

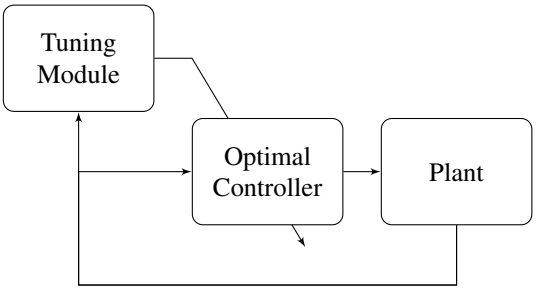

Fig. 1. Closed-loop adaptation of MPC objective function. The closed-loop of the optimal controller with the plant is augmented by a tuning module that takes data to adapt the parameters of the controller online.

I. Introduction

The control of many dynamical systems such as autonomous vehicles or robots include various specifications that are often conflicting, and thus require considerable manual calibration efforts. Furthermore, calibration is usually done at the production stage and it is often difficult to adjust the controller later, while the operating conditions of the dynamical system change over its lifetime. Hence, automating controller calibration and enabling to adapt the controller during operation is relevant in many applications [1]–[4].

In this paper, we propose an algorithm for the adaptation of cost function parameters for optimization-based control, such as model predictive control (MPC). The algorithm is implemented recursively using a Kalman filter that estimates the parameters of the cost function (rather than the state of the dynamical system). Fig. 1 shows the block-diagram of the proposed adaptation scheme, where the Kalman filter acts as a tuning module that uses data to calibrate the optimal controller. The main advantages of using a Kalman filter are that it (i) adapts the parameters online during system operation, (ii) is robust to noise due to the filter-based design, (iii) maintains safety guarantees of the closed-loop operation, (iv) is computationally efficient, (v) requires reduced data storage due to the recursive implementation, and (vi) is easy to implement, hence making it appealing for industrial applications. In this setting, the Kalman filter uses a training objective that evaluates the performance of the closed-loop system operation online, which is then used to adapt the parameters to improve upon the closed-loop system operation measured with respect to the training objective. The training objective has a highly flexible structure, while the optimal control cost function has a structure that is restricted due to its real-time application, e.g., the cost function often needs to be differentiable and convex such that it is suited for numerical optimization. Simulation results of a simple autonomous driving example show that the method achieves fast convergence of the parameters (approximately 95% faster than Bayesian optimization), improved closed-loop system performance (approximately 25% improvement on trajectory tracking precision), and is robust to noise.

All authors are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, 02139, USA (e-mail: menner@ieee.org; karl.o.berntorp@ieee.org; dicairano@ieee.org).

Related Work

The three research directions that are most closely related to the algorithm in this paper are inverse reinforcement learning (e.g., [5]–[10]), Bayesian optimization-based cost function learning (e.g., [11]–[13]), and retrospective cost optimization (e.g., [14]–[16]). For a more extensive discussion on cost function learning, the reader is referred to [17].

Inverse reinforcement learning (IRL): IRL uses human demonstrations to learn a cost function. Often, it aims at transferring human expertise to an autonomous system, e.g., for humanoid locomotion [5], identifying human movements [6], robot manipulation tasks [7], [8], or autonomous driving [9], [10]. Similarly to IRL, we also learn a cost function. Differently from IRL, we do not utilize human demonstrations and we do not require an episodical learning task (although our method can be implemented episodically). Instead, our approach estimates cost function parameters online during system operation.

Bayesian optimization (BO): BO-based approaches usually learn a mapping as a black-box function from the cost function parameters to a pre-specified performance metric, e.g., a reward function, using trial-and-error search [11]–[13]. Similarly to BO, we use a training objective to learn cost function parameters. Differently from BO, we do not require an episodical learning task and we do not have

a trial-and-error implementation. Instead, our algorithm is implemented recursively, which allows to compensate for systematic disturbances, while learning the cost function parameters. Furthermore, the recursive implementation is data efficient as the data history need not be stored.

Retrospective cost optimization (RCO): RCO uses transfer functions and optimizes the cost function retrospectively [14]–[16]. RCO adapts certain coefficients such that the new set of coefficents would have led to better performance over a previous window of operation. The idea is that the coefficients that would have performed well in the past will also perform better in the future, which is the case, e.g., in the presence of systematic disturbances. Similarly to retrospective cost optimization, we also optimize the optimal controller retrospectively, which allows us to compensate for systematic disturbances. Differently from retrospective cost optimization, we use a Kalman filter with a training objective to tune cost function parameters.

II. Preliminaries

A. Notation

Given two integer indices n, m with $m < n$ and a vector $x_i \in \mathbf{R}^{n_x}$, we define $X_{m|n} \in \mathbf{R}^{n_x(n-m+1)}$ as the vectorized sequence that comprises x_i from $i = m$ through $i = n$,

$$X_{m|n} := \begin{bmatrix} x_m \\ \vdots \\ x_n \end{bmatrix}. \tag{1}$$

Further, $\mathrm{diag}(\lambda) \in \mathbf{R}^{n_\lambda \times n_\lambda}$ is a matrix, whose diagonal entries are the entries of a vector $\lambda \in \mathbf{R}^{n_\lambda}$, I is an identity matrix of appropriate dimension, and 0 is an all-zero matrix of appropriate dimension. We define $\|x\|_\Sigma := x^T \Sigma x$ and the conditional probability density (PDF) function of a vector x_k at time steps $k = 0, ..., N$, conditioned on y, as $p(x_{0:N}|y) := p(x_0, x_1, ..., x_N|y)$. $\mathcal{N}(\mu, \Sigma)$ is the Gaussian distribution with mean vector μ and covariance matrix Σ.

B. KKT Conditions

Consider an optimization problem of the form

$$\min_z \ \theta^T \phi(z) \tag{2a}$$
$$\text{s.t. } D(z) = 0 \tag{2b}$$
$$C(z) \leq 0, \tag{2c}$$

where z is the optimization variable, $\theta^T \phi(z)$ is the cost function to be minimized, subject to equality constraints, $D(z) = 0$, and inequality constraints, $C(z) \leq 0$. The Lagrangian of (2) is

$$\mathcal{L}(z, \lambda, \nu) = \theta^T \phi(z) + \lambda^T C(z) + \nu^T D(z)$$

and the Karush-Kuhn-Tucker (KKT) conditions are

$$\nabla_z \mathcal{L}(z, \lambda, \nu) = 0 \tag{3a}$$
$$\lambda^T C(z) = 0 \tag{3b}$$
$$D(z) = 0 \tag{3c}$$
$$C(z) \leq 0 \tag{3d}$$
$$\lambda \geq 0, \tag{3e}$$

where λ and ν are called the dual variables or Lagrange multipliers (of the inequality and equality constraint, respectively), (3a) is the stationarity condition, (3b) is the complementary slackness condition, (3e) is called dual feasibility, and (3c) and (3d) are the primal feasibility conditions. The KKT conditions are first-order derivative tests for constrained optimization problems (necessary conditions for local minima). The reader is referred to [18] for more details.

C. Differentiating the KKT Conditions

Let z_0, λ_0, ν_0 be solutions to (3). Further, define $\phi_z(z_0) := \nabla_z \phi(z)|_{z=z_0}$, $D_z(z_0) := \nabla_z D(z)|_{z=z_0}$, $C_z(z_0) := \nabla_z C(z)|_{z=z_0}$, and $\mathcal{L}_{zz}(z_0, \lambda_0, \nu_0) := \nabla_{zz} \mathcal{L}(z, \lambda_0, \nu_0)|_{z=z_0}$. The sensitivity of the minimizer of (2), z_0, with respect to the parameters θ, i.e., $\frac{\Delta z}{\Delta \theta}$, can be obtained using the total derivative of (3),

$$W_0 \begin{bmatrix} \Delta z \\ \Delta \lambda \\ \Delta \nu \end{bmatrix} + V_0 \Delta \theta = 0 \tag{4}$$

with

$$V_0 = \begin{bmatrix} \phi_z(z_0)^T \\ 0 \\ 0 \end{bmatrix}$$

$$W_0 = \begin{bmatrix} \mathcal{L}_{zz}(z_0, \lambda_0, \nu_0) & C_z(z_0)^T & D_z(z_0)^T \\ \mathrm{diag}(\lambda_0) C_z(z_0) & \mathrm{diag}(C(z_0)) & 0 \\ D_z(z_0) & 0 & 0 \end{bmatrix}.$$

Remark 1: This strategy to compute the sensitivity is based on applying the Implicit Function Theorem to the KKT conditions [19], [20].

III. Problem Statement

A. System Dynamics and Constraints

We consider discrete-time systems of the form

$$x_{k+1} = f(x_k, u_k) + w_k, \tag{5}$$

where $x_k \in \mathbf{R}^{n_x}$ is the state at time step k, $u_k \in \mathbf{R}^{n_u}$ is the input, w_k is the process noise, and f is a general nonlinear function. Further, we consider constraints for the dynamical system in (5), given by

$$c(x_k, u_k) \leq 0.$$

We assume that both the state, x_k, and the input, u_k, are measurable.

B. Optimal Controller

We consider optimal predictive controllers of the form

$$\min_{x_k, u_k} \ \theta^T \phi(x_{0,...,N+1}, u_{0,...,N}) \tag{6a}$$
$$\text{s.t. } x_{k+1} = f(x_k, u_k) \quad \forall k = 0, ..., N \tag{6b}$$
$$c(x_k, u_k) \leq 0 \quad \forall k = 0, ..., N \tag{6c}$$
$$x_k = x(k) \tag{6d}$$

implemented in receding horizon fashion, e.g., MPC [21]. In (6a), $\theta^T \phi(x_{0,...,N+1}, u_{0,...,N})$ is the cost function with parameters θ, and $x(k)$ in (6d) is the current system state. Hence, (6) computes a plan (a control sequence) for N time steps, starting from the current state, $x(k)$.

C. Goal of the Learning Algorithm

The approach in this paper adapts the parameters of the cost function, θ, in (6). The adaptation algorithm calibrates θ based on sensor measurements, x_k and u_k, and a training objective or oracle, which evaluates the performance of the controller. The parameters are adapted online using a recursive algorithm such that the closed-loop system operation maintains its stability guarantees. As the algorithm uses time-varying parameters, i.e., the parameters are adjusted at each time step, we use time-indices (similar to the state and input variables) with

$$\theta_{k+1} = \theta_k + \Delta\theta_k, \tag{7}$$

where θ_k are the parameters at time step k. The goal is thus to find an adaptation law for the parameters, $\Delta\theta_k$.

Procedure and Training Objective: The learning algorithm takes sensor measurements, evaluates the performance of the controller, and outputs a new set of parameters for the controller, θ_{k+1}. It uses an evaluation function, $r(X_{k-N|k}, U_{k-N|k-1}) \in \mathbf{R}^{n_r}$, which takes the past state and input sequences (of length N), $X_{k-N|k}$ and $U_{k-N|k-1}$, and proposes a nominal value $y_k \in \mathbf{R}^{n_r}$ that the controller should ideally achieve. The nominal value, y_k, is a part of the training objective and is chosen by the system designer. It is closely related to a reference value to be tracked by a controller. One difference between the training objective and the cost function in (6a) is that the training objective is very generic in its structure and flexible in its specifications, while the cost function often requires a certain structure that enables real-time implementation, e.g., suited for convex optimization.

Remark 2: This approach is related to Bayesian optimization to train a cost function, where a reward/loss is obtained after a trial. The main conceptual differences are the online capable, recursive, and real-time feasible implementation, which uses a Kalman filter-based design (outlined in the following section) rather than a trial-and-error search.

Remark 3: The method proposed in this paper is similarly applicable if only a subset of the parameters are to be adjusted with $\theta_{k+1} = \theta_k + G\Delta\theta_k$, e.g., if certain parameters are known or should be fixed. For ease of exposition, however, we use (7) throughout.

IV. Observer-based Parameter Adaptation

In this section, we propose the recursive algorithm to adjust the parameters of the cost function, θ. Given the parameters at the current time step, θ_k, we aim to perform an update, $\Delta\theta_k$, based on data of closed-loop system operation as in (7). In particular, the learning algorithm aims to make a certain function $r(X_{k-N|k}, U_{k-N|k-1})$ follow a nominal value y_k "as closely as possible", i.e., the dynamical system is controlled perfectly if $y_k = r(X_{k-N|k}, U_{k-N|k-1})$. In order to ease exposition, using the notation in (1), we define a, possibly nonlinear, function F with

$$X_{k|k+N+1} = F(U_{k|k+N}, x_k, W_{k|k+N}), \tag{8}$$

which is equivalent to applying (5) iteratively, from k to $k+N$. Let $y_k = r(X_{k-N|k}, U_{k-N|k-1}) + v_k$, where v_k denotes a slack variable. As the input sequence $U_{k-N|k-1}$ results from an optimal control problem as in (6), it is a function of the parameters $\kappa(\theta_k, x_{k-N-1}) := U_{k-N|k-1}$. Therefore,

$$\begin{aligned} y_k &= r(F(U_{k-N-1|k-1}, x_{k-N-1}, W_{k-N-1|k-1})) + v_k \\ &= r(F(\kappa(\theta_k, x_{k-N-1}), x_{k-N-1}, W_{k-N-1|k-1})) + v_k. \end{aligned}$$

To further ease notation, we define

$$h(\theta_k) := r(F(\kappa(\theta_k, x_{k-N-1}), x_{k-N-1}, W_{k-N-1|k-1})).$$

Finally, we obtain the following two main equations for the adaptation scheme proposed in this paper,

$$\theta_{k+1} = \theta_k + \Delta\theta_k \tag{9a}$$

$$y_k = h(\theta_k) + v_k. \tag{9b}$$

In order to drive the adaptation law, we model the parameters, $\Delta\theta_k$, as well as the slack variable, v_k, as random variables with Gaussian zero-mean prior distributions $\Delta\theta \sim \mathcal{N}(0, \mathrm{C}_\theta)$ and $v_k \sim \mathcal{N}(0, \mathrm{C}_v)$. We obtain the parameter adaptation law from the corresponding posterior distribution $p(\theta_{k+1}|\theta_{0:k}, y_{0:k}) = \prod_{i=0}^{k} p(\theta_{i+1}|\theta_i, y_i)p(\theta_0)$. In particular, we use a Kalman filter whose goal is to estimate θ_{k+1} (rather than the state), and y_k is a desired value for the system operation (rather than sensor measurements). One benefit of using the Kalman filter is that it allows for a recursive implementation, which means that we do not need to store the entire data history but only the data used for the Kalman filter recursion. Note that $h(\theta)$ in (9b) considers process noise, $W_{k|k+N}$, explicitly. Hence, we seek to find the parameters θ that optimize the training objective for the process noise distribution/disturbances at hand. This allows for accommodating systematic disturbances as we will show in Section V. In the following, we present two implementations of the Kalman filter-based adaptation method, as well as their stability properties.

Remark 4 (Interpretation of $\mathrm{C}_v, \mathrm{C}_\theta$): The covariance matrix C_θ determines the aggressiveness of the adaptation. The covariance matrix C_v defines the relative importance of the vector-valued objective vector y_k. It is also possible to tune $\mathrm{C}_v, \mathrm{C}_\theta$ during application, cf. [22]. Throughout this paper, $\mathrm{C}_\theta = I$ implying that there are no preferences for tuning specific parameters faster than others, and $\mathrm{C}_v = I$.

A. Extended Kalman Filter

If $h(\theta)$ is differentiable, the following implementation of the extended Kalman filter (EKF) can be used to compute the parameter update in (7),

$$\Delta\theta_k = K_k \left(y_k - h(\theta_k)\right) \tag{10}$$

with the Kalman gain, K_k, computed as

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{11a}$$

$$S_k = H_k P_{k|k-1} H_k^T + \mathrm{C}_v \tag{11b}$$

$$P_{k|k-1} = P_{k-1|k-1} + \mathrm{C}_\theta \tag{11c}$$

$$P_{k|k} = \left(I - K_k H_k\right) P_{k|k-1}, \tag{11d}$$

where $H_k = \frac{\partial}{\partial\theta} h(\theta) \mid_{\theta=\theta_k}$, S_k is the innovation covariance, and $P_{k|k}$ is the estimate covariance matrix. In order to compute the linearization H_k, we use the chain rule

$$H_k = \frac{\partial h(\theta)}{\partial \theta} = \frac{\partial h(\theta)}{\partial z}\frac{\partial z}{\partial \theta}, \quad z = \begin{bmatrix} X_{k-N|k} \\ U_{k-N|k-1} \end{bmatrix},$$

where we use $\frac{\partial z}{\partial \theta} \approx \frac{\Delta z}{\Delta \theta}$ and $\frac{\Delta z}{\Delta \theta}$ is obtained by differentiating the KKT conditions, see Section II-C.

Remark 5 (Computational requirements): The computationally most demanding calculation step is the gradient computation H_k. Thus, using the approach outlined in Section II-C, this implementation requires solving one optimal control problem similar to (6) to obtain λ_0, ν_0, z_0 and one linear least squares problem using (4).

B. Unscented Kalman Filter

An alternative to EKF is to use an unscented Kalman filter (UKF) to compute the Kalman gain K_k. UKF uses deterministic samples (called sigma points) around the mean, which are propagated and used to update the mean and covariance estimates [23]. In the following, we use superscripts, i, to index sigma points, as opposed to the subscripts indicating the time step, k. Using a UKF, $h(\theta)$ need not be differentiable, which is one major advantage of this implementation. Here, too, the adaptation law for the parameters is given in (10) but the Kalman gain, K_k, is computed as

$$K_k = C_{sz} S_k^{-1} \tag{12a}$$
$$S_k = \mathrm{C}_v + \textstyle\sum_{i=0}^{2L} w^{c,i}(y^i - \hat{y})(y^i - \hat{y})^T \tag{12b}$$
$$C_{sz} = \textstyle\sum_{i=0}^{2L} w^{c,i}(\theta^i - \hat{\theta})(y^i - \hat{y})^T \tag{12c}$$
$$\hat{y} = \textstyle\sum_{i=0}^{2L} w^{a,i} y^i \tag{12d}$$
$$y^i = h(\theta^i) \tag{12e}$$
$$P_{k|k-1} = \mathrm{C}_\theta + \textstyle\sum_{i=0}^{2L} w^{c,i}(\theta^i - \hat{\theta})(\theta^i - \hat{\theta})^T \tag{12f}$$
$$\hat{\theta} = \textstyle\sum_{i=0}^{2L} w^{a,i} \theta^i \tag{12g}$$
$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T, \tag{12h}$$

where θ^i with $i = 0, ..., 2L$ are the sigma points, $w^{c,i}$ and $w^{a,i}$ are the weights of the sigma points, C_{sz} is the cross-covariance matrix, and S_k and $P_{k|k}$ are similar to the EKF.

Remark 6: In this paper, we choose the following weights and sigma points

$$\begin{aligned} \theta^0 &= \theta_k \\ \theta^i &= \theta_k + \sqrt{L/(1-w^0)}A^i \quad i = 1, ..., L \\ \theta^i &= \theta_k - \sqrt{L/(1-w^0)}A^i \quad i = L+1, ..., 2L \end{aligned}$$

with weights $-1 < w_0^a = w_0^c < 1$, $w_i^a = w_i^c = (1 - w_0^a)/(2L)$ and A^i being the ith column of A with $P_{k-1|k-1} = AA^T$, i.e., A is calculated using the Cholesky decomposition. This choice is motivated by its simplicity. Other choices of sigma points and weights are also possible.

Remark 7 (Computational requirements): The main advantage of this implementation is the ability to embed non-differentiable objectives. The main disadvantage compared to the EKF implementation is that since computing $h(\theta^i)$ in (12e) means solving an optimization problem, the UKF requires solving $2L+1$ optimization problems, where L is the number of parameters. However, due to independence they can be solved in parallel, thereby limiting the overall increase of computation time.

Remark 8: We use the EKF implementation to refer to the gradient-based method and UKF to the gradient-free method. However, as outlined in [24], there are implementations of the EKF that avoid computing the gradient explicitly.

Remark 9: The model-based nature of the adaptation algorithm allows for verifying the parameters using control-theoretic properties. For example, recursive feasibility follows from standard arguments (on constraints and terminal set/cost) and Lyapunov stability follows from cost-decrease arguments. In this paper, we omit details to conserve space.

V. Simulation Results

We consider the kinematic single track vehicle model [25]

$$x_{k+1} = (I + T_s A_c)\, x_k + T_s B_c u_k + w_k$$

with

$$\begin{aligned} x_k &= \begin{bmatrix} p_{X,k} & p_{Y,k} & \psi_k & V_k & \delta_k \end{bmatrix}^T \\ u_k &= \begin{bmatrix} \dot{V}_k & \dot{\delta}_k \end{bmatrix}^T \\ A_c &= \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x_{\mathrm{lin}}, u=u_{\mathrm{lin}}} \\ B_c &= \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=x_{\mathrm{lin}}, u=u_{\mathrm{lin}}} \end{aligned}$$

linearized at $x_{\mathrm{lin}} = [0,\ 0,\ 0,\ 10\ \mathrm{m/s},\ 0]^T$, $u_{\mathrm{lin}} = [0,\ 0]^T$ and

$$f(x_k, u_k) = \begin{bmatrix} v_{x,k}\cos(\psi_k + \beta_k)/\cos(\beta_k) \\ v_{x,k}\sin(\psi_k + \beta_k)/\cos(\beta_k) \\ v_{x,k}\tan(\delta_k)/L \\ \dot{V}_k \\ \dot{\delta}_k \end{bmatrix}$$

where p_X and p_Y mark the vehicle's position in the world frame, ψ is the heading (yaw) angle, v_x is the longitudinal velocity, δ is the steering angle of the front wheel, $L = l_f + l_r$ is the wheel base, and $\beta = \arctan(l_r \tan(\delta)/L)$ is the kinematic body-slip angle. The inputs u_1 and u_2 are the longitudinal acceleration and the steering rate.

The following simulations represent a reference tracking task, in which a vehicle tracks a certain lateral deviation from the center-line and a certain velocity. We use the quadratic cost function $\theta^T\phi(\cdot) = \|M(x_N - x_{\mathrm{ref}})\|_P + \sum_{k=0}^{N-1} \|M(x_k - x_{\mathrm{ref}})\|_Q + \|u_k\|_R$ with $M = [0_{4\times 1}\ I_4]$, i.e., all states but the longitudinal progress are penalized (road oriented in p_X direction). Hence, we learn the cost function parameters $P, Q \in \mathbf{R}^{4\times 4}$ and $R \in \mathbf{R}^{2\times 2}$, where $P, Q, R \succeq \epsilon \cdot I$, with a small ϵ. We include constraints on the available acceleration with $|\dot{V}_k| \leq 1\ \mathrm{m/s^2}$. Throughout this section, we use the sampling time $T_s = 0.25$ s and a planning horizon of the predictive controller $N = 20$. For all simulations and methods in this section, we initialized the parameters, Q, R, using randomly sampled positive definite matrices, where P is the result of the DARE.

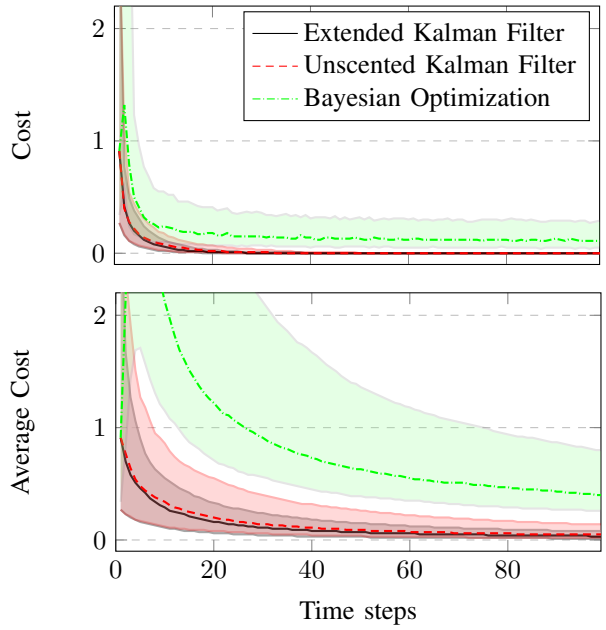

Fig. 2. Evaluation of Controller Performance. Both recursive algorithms, EKF and UKF, as well as BO are displayed. The plot shows the median (solid or dashed lines), as well as 75th and 25th percentiles (shaded areas) of 500 trials. Top: Cost of operation at each time step. Bottom: Average cost of controllers over time. The bottom plot is particularly interesting as it shows the increased cost that BO incurs due to the trial-and-error implementation.

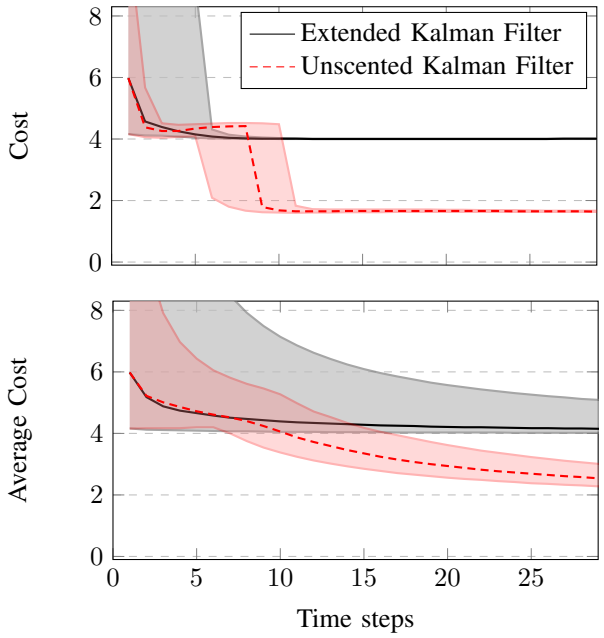

Fig. 3. Evaluation of Controller Performance for Partially Non-Differentiable Objective. The plot shows the median (solid or dashed lines), as well as 75th and 25th percentiles (shaded areas) of 500 trials. Top: Cost of operation at each time step. Bottom: Average cost of controllers over time. The UKF implementation is able to identify the cost-beneficial tuning of the parameters. It can be seen that the UKF moves toward the beneficial region, even if it means to encounter slightly higher cost temporarily (top plot, around time steps 4–8).

Remark 10: The algorithm's performance using the linearized vehicle model was observed to be similar to the nonlinear vehicle model case. This is to be expected as the parameters are linear in the cost function for both cases (the linearized vehicle model and nonlinear vehicle model), cf. also related work [26].

A. Comparison with Bayesian Optimization

First, we consider a noise-free test case, $w_k = 0$, and a reference tracking task with $x_{\text{ref}} = x_{\text{lin}} = [0,\ 0,\ 0,\ 10\ \text{m/s},\ 0]^T$, where the initial state is $x_0 = [0,\ 2\ \text{m},\ 0,\ 12\ \text{m/s},\ 0]^T$. The learning algorithm uses the training objective $h(\theta) = [X_{0|N}^T,\ U_{0|N}^T]^T$ and $y_k = 0$. Fig. 2 reports the cost $\|y_k - h(\theta)\|_2^2$ over time, by showing the results of 500 trials, where the cost parameters are initialized randomly at the beginning of each trial. It shows the convergence behaviors of the two recursive algorithms proposed in this paper, the EKF and the UKF, along with a Bayesian optimization (BO) approach, which is added as comparison. Here we use an episodical learning task to obtain a meaningful comparison with BO. The BO approach uses a squared exponential kernel and the Upper Confidence Bound (UCB) acquisition function, both commonly used in the literature, e.g., [13]. Further, we restrict the BO approach also to the search space of positive definite matrices. It can be seen that both the EKF and UKF variants outperform BO in terms of convergence speed and steady-state cost at the end of the simulation at time step 100. Note that a potential drawback of the proposed methods for some applications might be that the lack of global exploration, compared to BO.

B. EKF and UKF for Non-Differentiable Objective

In addition to the tracking task described in Section V-A, the training objective in this simulation includes a penalty for each sign change in $\dot{\delta}$ ($y_k = 0$ and $h(\theta) = [X_{0|N}^T,\ U_{0|N}^T,\ \#\dot{\delta}\,\text{sign changes}]^T$). One rational behind this penalty is to reduce oscillatory or jittering motions of the vehicle. This additional penalty is a discontinuous function of the control input sequence. Fig. 3 shows the learning performance of the EKF and UKF implementation in the presence of the partially non-differentiable training objective. It shows that the UKF adjusts the parameters to capture the discontinuity, whereas the EKF converges to a set of parameters that incur higher cost.

C. Continuous Control Task with Disturbances

In order to test the robustness of the adaptation algorithm, next we present a simulation study of a continuous control task, in the presence of process noise and disturbances. Here, we present the EKF implementation but the UKF performs similarly. The results show that the algorithm is able to adjust to systematic disturbances and is able to reject random process noise, both of which are desirable properties. We simulate system (5) from time step $k = 0$ through $k = 1000$ with initial state $x_0 = x_{\text{ref}}$ and choose process noise, w_k, as

$$w_k = \begin{bmatrix} 0 & \Delta p_{Y,k} & 0 & 0 & 0 \end{bmatrix}^T,$$

where $\Delta p_{Y,k}$ is varied as displayed in Table I. In other words, we consider additive process noise/disturbances acting on the

TABLE I

AVERAGE COST FOR DIFFERENT PROCESS NOISE AND DISTURBANCES

Test case	Adaptive (diag. Q, R)	Adaptive	Not Adaptive True param.
$\Delta p_{Y,k} = 1$	9.133	7.916	10.748
$\Delta p_{Y,k} = \cos(10^{-3}k)$	3.443	3.323	4.355
$\Delta p_{Y,k} = \cos(10^{-2}k)$	5.031	4.014	5.472
$\Delta p_{Y,k} = \cos(10^{-1}k)$	4.195	4.4674	5.331
$\Delta p_{Y,k} = \cos(k)$	0.609	1.1599	1.664
$\Delta p_{Y,k} = \cos(10k)$	0.1912	0.1975	0.2621
$\Delta p_{Y,k} \sim \mathcal{N}(0,1)$	1.160	1.122	1.120

lateral position of the vehicle, which could, e.g., result from winds/gusts. The training objective is $h(\theta) = [X_{0|N}^T, U_{0|N}^T]^T$ and $y_k = 0$, as in Section V-A. As baseline, Table I shows the performance of the MPC controller without adaptation that uses the true parameters of the training objective (last column). Table I shows that the algorithm is able to improve upon the system operation in the presence of systematic disturbances (e.g., the closed-loop cost is reduced by roughly 25% for $\Delta p_{Y,k} = 1$), but is also not over-fitting to noise (e.g., the closed-loop cost is comparable for $\Delta p_{Y,k} \sim \mathcal{N}(0,1)$). As an example for how the algorithm operates, for the test case where $\Delta p_{Y,k} = 1$, the algorithm detects that there is a constant disturbance pushing the vehicle to the right side. As a consequence, the algorithm reacts by increasing the penalty for lateral deviations, p_Y. Table I shows the performance of the algorithm for learning nondiagonal matrices $Q, R \succ 0$, as well as diagonal matrices, which might be of interest for some readers as it is a common choice for control system designers. Note that we did not observe oscillations or other undesirable behaviors during our simulation studies.

VI. CONCLUSION

This paper proposed a method to estimate cost function parameters for optimal controllers. The method is implemented in a recursive fashion using a Kalman filter, which estimates the parameters (rather than the dynamical system's state). The main benefits of the proposed method are the low computational requirements, low data storage requirements, and a relatively high flexibility of embedding, e.g, non-differentiable objectives in the control system. Simulation results show that the method is able to learn the cost function parameters (i) quickly and robustly and (ii) adjust the parameters to systematic disturbances, thereby effectively reducing closed-loop cost of the system operation.

REFERENCES

[1] S. Di Cairano and A. Bemporad, "Model predictive control tuning by controller matching," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 185–190, 2009.

[2] A. Lucchini, S. Formentin, M. Corno, D. Piga, and S. M. Savaresi, "Torque vectoring for high-performance electric vehicles: an efficient MPC calibration," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 725–730, 2020.

[3] M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," *arXiv preprint arXiv:2003.11294*, 2020.

[4] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach," *Autonomous robots*, vol. 28, no. 3, pp. 369–383, 2010.

[5] D. Clever, R. M. Schemschat, M. L. Felis, and K. Mombaur, "Inverse optimal control based identification of optimality criteria in whole-body human walking on level ground," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 1192–1199, 2016.

[6] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained inverse optimal control with application to a human manipulation task," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 826–834, 2021.

[7] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[8] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3682–3690, 2020.

[9] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, and S. Roth, "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2658–2665, 2019.

[10] M. Menner, K. Berntorp, M. N. Zeilinger, and S. D. Cairano, "Inverse learning for data-driven calibration of model-based statistical path planning," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 131–145, 2021.

[11] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on gaussian process global optimization," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 270–277, 2016.

[12] M. Neumann-Brosig, A. Marco, D. Schwarzmann, and S. Trimpe, "Data-efficient autotuning with bayesian optimization: An industrial control study," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 730–740, 2019.

[13] Q. Lu, R. Kumar, and V. M. Zavala, "MPC controller tuning using Bayesian optimization techniques," *arXiv preprint arXiv:2009.14175*, 2020.

[14] M. A. Santillo and D. S. Bernstein, "Adaptive control based on retrospective cost optimization," *Journal of guidance, control, and dynamics*, vol. 33, no. 2, pp. 289–304, 2010.

[15] Y. Rahman, A. Xie, J. B. Hoagg, and D. S. Bernstein, "A tutorial and overview of retrospective cost adaptive control," in *2016 American Control Conference (ACC)*, pp. 3386–3409, 2016.

[16] Y. Rahman, A. Xie, and D. S. Bernstein, "Retrospective cost adaptive control: Pole placement, frequency response, and connections with lqg control," *IEEE Control Systems Magazine*, vol. 37, no. 5, pp. 28–69, 2017.

[17] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[18] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[19] A. V. Fiacco, *Introduction to sensitivity and stability analysis in nonlinear programming*. Mathematics in Science and Engineering, Burlington, MA: Elsevier, 1983.

[20] H. Pirnay, R. López-Negrete, and L. T. Biegler, "Optimal sensitivity based on IPOPT," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 307–331, 2012.

[21] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[22] T. Ardeshiri, E. Özkan, U. Orguner, and F. Gustafsson, "Approximate Bayesian smoothing with unknown process and measurement noise covariances," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2450–2454, 2015.

[23] G. A. Terejanu, "Unscented Kalman filter tutorial," *University at Buffalo, Buffalo*, 2011.

[24] F. Gustafsson and G. Hendeby, "Some relations between extended and unscented Kalman filters," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, 2012.

[25] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty - a control perspective," *European Journal of Control*, vol. 24, pp. 14–32, 2015.

[26] M. Menner and M. N. Zeilinger, "Maximum likelihood methods for inverse learning of optimal controllers," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5266–5272, 2020.