Hardware-in-the-Loop Iterative Optimal Feedback Control Without Model-Based Future Prediction

Yuqing Chen and David J. Braun, *Member, IEEE*

***Abstract*—Optimal control provides a systematic approach to control robots. However, computing optimal controllers for hardware-in-the-loop control is sensitively affected by modeling assumptions, computationally expensive in online implementation, and time-consuming in practical application. This makes the theoretical appeal of optimization challenging to exploit in real-world implementation. In this paper, we present a novel online optimal control formulation that aims to address the above-mentioned limitations. The formulation combines a model with measured state information to efficiently find near-optimal feedback controllers. The idea to combine a model with measurements from the actual motion is similar to what is used in model predictive control formulations, with the difference that here the model is not used for future prediction, the optimization is performed along the measured trajectory of the system, and the online computation is reduced to a minimum; it requires a small-scale, one time step, static optimization, instead of a large-scale, finite time horizon, dynamic optimization. The formulation can be used to solve optimal control problems defined with nonlinear cost, nonlinear dynamics, and box-constrained control inputs. Numerical simulations and hardware-in-the-loop experiments demonstrate the effectiveness of the proposed hardware-in-the-loop optimal control approach.**

***Index Terms*—Control architectures and programming, feedback control, hardware-in-the-loop control, optimization and optimal control.**

I. Introduction

OPTIMAL control provides a systematic computational approach to find controllers for robots [1]. The theory of optimal control [2], [3] lies at the heart of these approaches. However, optimal control has two limitations when it comes to practical application: one is the *model bias*, which makes it nontrivial to use imperfect model information to speed up the optimization without introducing performance degradation, while the other is the *excessive computation* required within the time constraint in online optimization. A number of model predictive control (MPC) methods [4]–[14] and model-based reinforcement learning (m-RL) approaches [15]–[27] aim to address these limitations. Nevertheless, finding controllers for robots that self-optimize through their motion is time-consuming, computationally expensive, and sensitively affected by modeling assumptions. These make the theoretical appeal of model-based optimization challenging to exploit in practical application.

In this paper, we propose a hardware-in-the-loop optimal control (HIL-OC) approach to solve finite-time horizon nonlinear control constrained dynamic optimization problems. We assume that an imperfect model of the dynamics is provided and that it offers useful information, one that can substantially speed up the optimization. This idea is the underpinning of industry standard MPC methods. MPC combines a model with measured state information to find near-optimal controllers for hardware-in-the-loop control. It solves a finite time horizon optimization using model-based forward prediction at each time step, during the actual motion. The online computational cost of this formulation is tied to a finite time horizon constrained optimization, and the use of an imperfect model for future prediction often significantly affects optimality in practical application.

In order to circumvent these limitations, we propose a novel way to combine an imperfect model with measured state information. The resulting formulation significantly reduces performance degradation by *completely avoiding model-based future prediction*, and by optimizing the control inputs along the *actual motion*, instead of optimizing along trajectories obtain by model-based future prediction. The formulation also reduces the online computation to a small-scale static optimization, one that is equivalent to a one-time step dynamic optimization. This is the minimal amount of computation to implement dynamic optimization. These features are not within the reach of the state-of-the-art nonlinear MPC methods [9], [10], [28].

The proposed HIL-OC approach extends previously derived offline iterative optimization algorithms—including differential dynamic programming (DDP) [29], the method of successive approximation (MSA) [30], [31], the iterative linear quadratic regulator (iLQR) [32], the constrained differential dynamic programming approach (cDDP) [33], and the iterative constrained optimal control [34]—to a HIL-OC method. This extension is significant because it enables robots to iteratively self-optimize through repeated task execution, in a similar way as models of robots have been previously optimized in numerical simulation. The key novelty of the proposed formulation is that it uses

Manuscript received March 20, 2019; accepted June 12, 2019. Date of publication October 22, 2019; date of current version December 3, 2019. This paper was recommended for publication by Associate Editor S.-J. Chung and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported by the MOE2016-T2-2-051 Tier 2 Academic Research Funding provided by Singapore's Ministry of Education. *(Corresponding author: David J. Braun.)*

Y. Chen is with Engineering Product Development Pillar, Singapore University of Technology and Design, 487372 Singapore (e-mail: yuqing_chen@mymail.sutd.edu.sg).

D. J. Braun is with the Department of Mechanical Engineering, Vanderbilt University, Nashville, TN 37212 USA (e-mail: david.braun@vanderbilt.edu).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org. The video shows a hardware-in-the-loop implementation of the proposed optimal control algorithm. The size of the video is 43MB.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2019.2929014

measured trajectories instead of trajectories obtained by forward integration of an inexact model. We show that this approach significantly reduces model bias.

One way to reduce model bias is to learn a better model, because a better model provides a better model-based future prediction, and therefore, a better controller. This approach is taken by m-RL methods [17]–[26]. Another way to reduce model bias is to avoid forward predictions obtained by inexact models. This approach is taken by the proposed HIL-OC method. Therefore, the advantage of the HIL-OC method does not come from model learning; it comes from not using the model for future prediction, regardless of whether the model is derived from first principles [35], estimated offline [36], or learned from data online [37].

The question we address with the HIL-OC method is: 1) *How to reduce model bias given an inexact model that cannot be improved by learning?* Our key contribution is to show that the HIL-OC method can reduce model bias even if the model cannot be precisely learned. This contribution may be appreciated because even the best model is inexact, and unless the time horizon of the control problem is short, inexact models cannot provide accurate future prediction. An alternative question concerning a number of recently developed m-RL methods [19]–[22] is: 2) *How to learn a better model efficiently from data?* Learning a better model is essential in m-RL methods because the promise of m-RL methods is to provide better control by a better model. While the HIL-OC method can use any model, including a model that is learned from data, in this paper we do not use a learned model in order to unambiguously show that the benefit provided by the HIL-OC method is *complementary* to the benefit provided by model learning.

This paper has three main contributions.

1) We present a novel HIL-OC method which is general enough to iteratively solve *finite time horizon optimal control problems defined by nonlinear cost, nonlinear dynamics, and box-constrained control inputs*. The method is shown to significantly reduce model bias, partly because it does not rely on inexact model-based future prediction, but also because it takes the constraints into account during the actual motion, instead of taking them into account during an inexact model-based future prediction. The proposed use of inexact models and the treatment of constraints may be used to extend prior formulations that use model-based future prediction in a finite horizon setting [15], [16], [18]–[24], [38]–[40], and methods applicable to infinite horizon stabilization problems [26], [27], [41], [42].
2) We present a new hardware-in-the-loop algorithm which belongs to the class of successive approximation methods [29]–[34]. Our algorithm does not use model-based future prediction, and calculates the control inputs by *minimizing the Hamiltonian under constraints forward in time, hardware-in-the-loop, based on measured states*. This approach differs from an alternative implementation where the Hamiltonian is minimized between two hardware-in-the-loop iterations to calculate the feed-forward control sequence and the locally valid feedback controller, as promoted by previously developed successive approximation methods [32]–[34]. We provide a proof of convergence for the proposed algorithm and characterize the suboptimality of the controller. Our theoretical analysis extends to the case when the inputs are constrained and the model is inexact. These results represent the theoretical contribution of this paper.
3) For any given inexact model, we show that the proposed HIL-OC method outperforms optimal feedback control and MPC methods in terms of optimality and computational efficiency; HIL-OC provides lower cost and has lower real-time computational requirement than MPC. This result is significant not only on its own right, but also because it suggests a way to enhance the performance of previously developed m-RL methods when they cannot improve the model by learning.

This paper is organized as follows. In Section II, we present a finite time horizon nonlinear optimal control problem subject to control constraints. In Section III, we recall an iterative approach that can be used to solve the aforementioned optimal control problem using model-based prediction. In Section IV, we extend this formulation to a novel hardware-in-the-loop control approach which does not require model-based prediction. In Section V, we numerically show that the proposed formulation reduces model bias compared to optimal feedback control and MPC methods. In Section VI, we demonstrate hardware-in-the-loop finite-time horizon nonlinear dynamic optimization using a three-link torque-controlled robot subject to input constraints. The purpose of the experiment is to show the advantage of the HIL-OC approach. The supplementary video of the experiment shows the first practical implementation of the algorithm presented in this paper. Section VII concludes this paper.

II. Problem Formulation

We consider the following constrained nonlinear optimization problem:

$$\min_{\mathbf{u}(\cdot)\in\mathcal{U}} \quad \mathcal{I}[\mathbf{u}(\cdot)] = \min_{\mathbf{u}(\cdot)\in\mathcal{U}} \phi(\mathbf{x}(T),T) + \int_0^T \mathcal{L}(\mathbf{x}(t),\mathbf{u}(t),t)dt \tag{1}$$

$$\text{subject to: } \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) \text{ and } \mathbf{x}(0) = \mathbf{x}_0 \tag{2}$$

where $t \in \mathbb{T} = [0,T]$ is the time interval, $T \in \mathbb{R}_+$ is the terminal time, $\mathbf{x}(t) = [x_1(t),\ldots,x_n(t)]^\top \in \mathbb{R}^n$ is the state, $\mathbf{u}(t) = [u_1(t),\ldots,u_m(t)]^\top \in \mathbb{R}^m$ is the control, $\phi(\mathbf{x}(T),T) : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ is the terminal cost, $\mathcal{L}(\mathbf{x}(t),\mathbf{u}(t),t) : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R}$ is the running cost, and $\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system dynamics.

We assume that the dynamics is once continuously differentiable function of $\mathbf{x}$ and $\mathbf{u}$, the terminal and running costs are twice continuously differentiable functions of $\mathbf{x}$ and $\mathbf{u}$ and measurable in t, and that the running cost is convex with respect to $\mathbf{u}$ along the optimal solution. We also assume that the control function $\mathbf{u}(\cdot) : \mathbb{T} \to \mathbb{U}$ belongs to a set of measurable functions

Algorithm 1: Model-in-the-Loop Optimal Control [34].

Input: Control input $\mathbf{u}^0(\cdot) \in \mathbb{U}$, model
Output: $\mathbf{u}^*(\cdot) \in \mathbb{U}$
Initialize: $conv = false$, $i = 1$, $\epsilon_I > 0$, $\epsilon_u > 0$
$\mathbf{x}^0(\cdot) \leftarrow \big(\mathbf{u}^0(\cdot),\ \text{model}\big)$, $\mathcal{I}^0 \leftarrow (\mathbf{x}^0(\cdot), \mathbf{u}^0(\cdot))$, $\delta\mathbf{x}^0 = \mathbf{0}$
COMPUTATION
while $conv = false$ **do**
  for $j = n_T : -1 : 1$ **do**
    Evaluate partial derivatives along $(\mathbf{x}_j^{i-1}, \mathbf{u}_j^{i-1})$
    $\delta\mathbf{u}^i_{j\alpha^*} \leftarrow$ (15), $\mathbf{R}_j^{i-1}$, $\mathbf{r}_j^{i-1}, \delta\mathbf{x}_j^{i-1}$
    $\mathbf{R}_{j-1}^{i-1}$, $\mathbf{r}_{j-1}^{i-1} \leftarrow$ (12), (13), $\delta\mathbf{u}^i_{j\alpha^*}$
  end
  $\mathbf{u}^i(\cdot) \leftarrow \mathbf{u}^{i-1}(\cdot) + \delta\mathbf{u}^i_{\alpha^*}(\cdot)$
  $\mathbf{x}^i(\cdot) \leftarrow \big(\mathbf{u}^i(\cdot), \text{model}\big)$
  $\delta\mathbf{x}^i(\cdot) \leftarrow \mathbf{x}^i(\cdot) - \mathbf{x}^{i-1}(\cdot)$
  $\mathcal{I}^i \leftarrow (\mathbf{x}^i(\cdot), \mathbf{u}^i(\cdot))$
  if $|\mathcal{I}^i - \mathcal{I}^{i-1}| \le \epsilon_I$ **and** $\sum_{j=1}^{n_T} \|\delta\mathbf{u}^i_{j\alpha^*}\| \le \epsilon_u$ **then**
    $\mathbf{u}^*(\cdot) = \mathbf{u}^i(\cdot)$, $\mathbf{x}^*(\cdot) = \mathbf{x}^i(\cdot)$, $conv = true$
  end
  $i = i + 1$
end
IMPLEMENTATION
for $j = 1 : n_T$ **do**
  $\mathbf{x}_j \leftarrow$ robot
  $\delta\mathbf{x}_j \leftarrow \mathbf{x}_j - \mathbf{x}_j^*$
  $\delta\mathbf{u}_{j\alpha^*} \leftarrow$ (15), $\mathbf{R}_j^*$, $\mathbf{r}_j^*, \alpha^* = 1, \delta\mathbf{x}_j$
  $\mathbf{u}_j \leftarrow \mathbf{u}_j^* + \delta\mathbf{u}_{j\alpha^*}$ (feedforward and feedback)
  robot $\leftarrow \mathbf{u}_j$
end

$\mathcal{U}$, while the control input belongs to a nonempty closed and bounded set

$$\forall t \in \mathbb{T}:\ \mathbf{u}(t) \in \mathbb{U} = \{\mathbf{u}(t) \in \mathbb{R}^m : \mathbf{u}_{\min} \preceq \mathbf{u}(t) \preceq \mathbf{u}_{\max}\} \tag{3}$$

where $\mathbf{u}_{\min} \prec \mathbf{u}_{\max}$ are fixed lower and upper bounds. In the remainder of this paper, we use $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ to denote the control and state variables in relations that are valid at every time instant $t \in \mathbb{T}$. Also, we use $\mathbf{u}(\cdot)$ and $\mathbf{x}(\cdot)$ to denote the control function and the state trajectory, respectively.

III. Iterative Optimal Control With Model-Based Forward Prediction

In this section, we recall an optimal control method applicable to finite time horizon problems defined by nonlinear cost, nonlinear dynamics, and systems controlled with box-constrained control inputs [34]. This method belongs to the family of successive approximation methods DDP [29], MSA [30], [31], iLQR [32], and cDDP [33].

Algorithm 1 shows both the feedforward and the feedback control implementations of the method in [34]. This implementation can be used for online robot control using model-based forward prediction, similar to iLQR and cDDP.

We will use this algorithm as a benchmark for the novel hardware-in-the-loop successive approximation method proposed in Section IV.

A. Constrained Linear Quadratic (cLQR) Subproblem

We assume that $\mathbf{u}^0(\cdot) \in \mathcal{U}$ is a feasible control trajectory and $\mathbf{x}^0(\cdot)$ is the corresponding state trajectory obtained from (2). We define the variations of the state and control trajectories by

$$\forall t \in \mathbb{T}:\ \delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}^0(t) \ \text{ and } \ \delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}^0(t).$$

A local approximation of the nonlinear optimization problem (1)–(3) around the trajectory $(\mathbf{x}^0, \mathbf{u}^0)$ is given by the following cLQR subproblem[1]:

$$\min_{\delta\mathbf{u}(\cdot)\in\delta\mathcal{U}^0} \ \Delta\mathcal{I}[\delta\mathbf{u}(\cdot)] = \min_{\delta\mathbf{u}(\cdot)\in\delta\mathcal{U}^0} \Delta\phi + \int_0^T \Delta\mathcal{L}dt \tag{4}$$

$$\text{subject to: } \delta\dot{\mathbf{x}} = \mathbf{f}_{\mathbf{x}}\delta\mathbf{x} + \mathbf{f}_{\mathbf{u}}\delta\mathbf{u}, \ \ \delta\mathbf{x}(0) = \mathbf{0} \tag{5}$$

where

$$\Delta\phi = \phi_{\mathbf{x}}\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^\top\phi_{\mathbf{xx}}\delta\mathbf{x} \ \text{ and}$$

$$\Delta\mathcal{L} = \begin{bmatrix}\mathcal{L}_{\mathbf{x}} & \mathcal{L}_{\mathbf{u}}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}^\top & \delta\mathbf{u}^\top\end{bmatrix}\begin{bmatrix}\mathcal{L}_{\mathbf{xx}} & \mathcal{L}_{\mathbf{xu}}\\ \mathcal{L}_{\mathbf{ux}} & \mathcal{L}_{\mathbf{uu}}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}. \tag{6}$$

The variation of the control is a bounded measurable function $\delta\mathbf{u}(\cdot) \in \delta\mathcal{U}^0$, and at every time instant, the variation of the control belongs to a nonempty, closed, and bounded set

$$\forall t \in \mathbb{T}:\ \delta\mathbf{u}(t) \in \delta\mathbb{U}^0 = \{\delta\mathbf{u} \in \mathbb{R}^m : \mathbf{u}_{\min} - \mathbf{u}^0(t) \preceq \delta\mathbf{u} \preceq \mathbf{u}_{\max} - \mathbf{u}^0(t)\}. \tag{7}$$

Necessary conditions of optimality: We proceed by deriving the necessary conditions of optimality for the subproblem (4)–(7). Following a typical approach [3], we define the Hamiltonian using the linear model (5) and the costate variable $\boldsymbol{\lambda}$,

$$\Delta\mathcal{H}(\delta\mathbf{x}, \boldsymbol{\lambda}, \delta\mathbf{u}) = \Delta\mathcal{L} + \boldsymbol{\lambda}^\top(\mathbf{f}_{\mathbf{x}}\delta\mathbf{x} + \mathbf{f}_{\mathbf{u}}\delta\mathbf{u}). \tag{8}$$

This Hamiltonian is used to derive the costate equation and the control input

$$\dot{\boldsymbol{\lambda}} = -\Delta\mathcal{H}_{\delta\mathbf{x}}(\delta\mathbf{x}, \boldsymbol{\lambda}, \delta\mathbf{u}^*), \quad \boldsymbol{\lambda}(T) = \phi_{\mathbf{x}} + \phi_{\mathbf{xx}}\delta\mathbf{x}(T) \tag{9}$$

$$\delta\mathbf{u}^* = \arg\min_{\delta\mathbf{u}\in\delta\mathbb{U}^0} \Delta\mathcal{H}(\delta\mathbf{x}, \boldsymbol{\lambda}, \delta\mathbf{u}). \tag{10}$$

Solution of the subproblem: In order to decouple the costate equation (9) from the state equation (5), and to derive a relation between the control and the state from (10), we use the following linear transformation:

$$\boldsymbol{\lambda} = \mathbf{R}(t)\delta\mathbf{x} + \mathbf{r}(t) \tag{11}$$

where $\mathbf{R}(t) \in \mathbb{R}^{n\times n}$ and $\mathbf{r}(t) \in \mathbb{R}^n$. Substituting (11) into (9) and (10), the costate equation is transformed into the following

[1]Subscripts abbreviate partial derivatives. All derivatives are evaluated along $(\mathbf{x}^0(\cdot), \mathbf{u}^0(\cdot))$.

two linear differential equations:

$$\dot{\mathbf{R}} + \mathbf{R}\mathbf{f}_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\top}\mathbf{R} + \mathcal{L}_{\mathbf{xx}} = \mathbf{0}, \quad \mathbf{R}(T) = \phi_{\mathbf{xx}} \tag{12}$$

$$\dot{\mathbf{r}} + (\mathcal{L}_{\mathbf{xu}} + \mathbf{R}\mathbf{f}_{\mathbf{u}})\delta\mathbf{u}^{*} + \mathcal{L}_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\top}\mathbf{r} = \mathbf{0}, \quad \mathbf{r}(T) = \phi_{\mathbf{x}}. \tag{13}$$

Given $\mathbf{R}$ and $\mathbf{r}$, the control input is obtained from a constrained quadratic program (10),

$$\delta\mathbf{u}^{*} = \underset{\delta\mathbf{u}\in\delta\mathbb{U}^{0}}{\arg\min}\ \frac{1}{2}\delta\mathbf{u}^{\top}\mathcal{L}_{\mathbf{uu}}\delta\mathbf{u} + \mathbf{b}(\delta\mathbf{x})^{\top}\delta\mathbf{u} \tag{14}$$

where $\mathbf{b}(\delta\mathbf{x}) = (\mathcal{L}_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\top}\mathbf{R})\delta\mathbf{x} + (\mathcal{L}_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\top}\mathbf{r})$.

B. Iterative Optimization

The control update calculated by pointwise optimization for the entire time horizon $\delta\mathbf{u}^{*}(\cdot)$ (14) may be used to calculate the new control function $\mathbf{u}^{i}(\cdot) = \mathbf{u}^{i-1}(\cdot) + \delta\mathbf{u}^{*}(\cdot) \in \mathcal{U}$ if such a new control function decreases the cost (1). This requires $\delta\mathbf{u}^{*}$ not only to be small enough to justify the linear-quadratic approximation in (4) and (5), but also to be along the descent direction of the cost.

To maintain these practical requirements, we use the update formula:

$$\delta\mathbf{u}_{\alpha^{*}}^{i} = \underset{\delta\mathbf{u}\in\delta\mathbb{U}^{i-1}}{\arg\min}\ \frac{1}{2}\delta\mathbf{u}^{\top}\mathcal{L}_{\mathbf{uu}}^{\text{PD}}\delta\mathbf{u} + \alpha^{*}\mathbf{b}(\delta\mathbf{x})^{\top}\delta\mathbf{u} \tag{15}$$

$$\mathbf{u}^{i} = \mathbf{u}^{i-1} + \delta\mathbf{u}_{\alpha^{*}}^{i} \in \mathbb{U} \tag{16}$$

where i and $i-1$ refer to the current and previous iterations, $\delta\mathbf{u}_{\alpha^{*}}^{i}$ is the control update, $\mathcal{L}_{\mathbf{uu}}^{\text{PD}}$ is a modified positive definite Hessian, and $\alpha^{*} \in [0, 1]$ is the convergence control parameter.

The modified Hessian is the same as the original Hessian $\mathcal{L}_{\mathbf{uu}}$ when the latter is positive definite, otherwise, it is obtained by a modified Cholesky factorization of $\mathcal{L}_{\mathbf{uu}}$ [43]. The convergence control parameter is calculated using an inexact minimization,

$$\alpha^{*} \approx \underset{\alpha\in[0,1]}{\arg\min}\, \mathcal{I}[\mathbf{u}^{i-1}(\cdot) + \delta\mathbf{u}_{\alpha}^{i}(\cdot)] \tag{17}$$

such that (16) decreases the cost (1). For the extreme values of the convergence control parameter, (15) implies $\delta\mathbf{u}_{\alpha^{*}=0}^{i} = \mathbf{0}$ and $\delta\mathbf{u}_{\alpha^{*}=1}^{i} \approx \delta\mathbf{u}^{*}$ (the equality holds when $\mathcal{L}_{\mathbf{uu}}$ is positive definite).

C. Algorithm

Algorithm 1[2] shows a minimalistic implementation of the aforementioned method. In this algorithm, the state required to compute the control input comes from the forward prediction in the previous iteration using the inexact model (18), the sequence of control inputs is computed backward in time (12)–(14), and the precomputed control sequence is used to control the model forward in time. Upon convergence, the algorithm provides the optimal control sequence and the optimal trajectory predicted by the model, which can be used to control the system. The optimal control sequence can be used for online robot control, in particular, it can be used for 1) open-loop control, 2) locally optimal feedback control, or 3) model predictive feedback control [34].

Algorithm 1 is a modified version of prior successive approximation methods (DDP [29], MSA [31], iLQR [32], cDDP [33]). The novelty of Algorithm 1 comes from 1) the calculation of the control update using (12)–(14), which takes the constraints rigorously into account as discussed in [34], and 2) the convergence control (15)–(17), which will become important in the convergence proof presented in Section IV-D. All these algorithms can be used for online control of robots; the solution they provide can be used for open-loop optimal control, locally optimal feedback control [32], [44], and MPC [45]. Common to all these algorithms is that they integrate the inexact model forward in time to obtain a future prediction of the system's trajectory, and then use the predicted trajectory to calculate the optimal control input, or the locally optimal feedback control law [32], [44], [45]. Using the inexact model to make a future prediction leads to model bias, and this model bias affects all aforementioned formulations.

In Section IV, we suggest a novel optimal control algorithm where the model is never used to make a future prediction of the state trajectory, and the control inputs are computed along the measured trajectory of the robot. The former idea will be shown to significantly reduce model bias when the model is inexact, while the latter idea enables us to take the control constraints rigorously into account, during the actual motion, instead of during a model-based simulation [29]–[34], [44], [45].

IV. HIL-OC Without Model-Based Forward Prediction

We propose a novel HIL-OC method applicable to nonlinear control constrained dynamical systems. Algorithm 2 provides a detailed implementation of this method. In the following, we outline the main differences between typical successive approximation algorithms (Algorithm 1) and the proposed Algorithm 2. The convergence and optimality of Algorithm 2 are proven under technical assumptions, and the results are summarized in Section IV-D. The proofs are provided in the Appendix.

A. Optimal Control Along a Measured Trajectory

We assume that the model is imperfect and is given by

$$\dot{\hat{\mathbf{x}}} = \hat{\mathbf{f}}(\hat{\mathbf{x}}, \mathbf{u}). \tag{18}$$

This model can be analytically derived [35], estimated from data offline [36], or learned online [37]. Regardless of which approach is used, it is challenging to derive or learn good models, and therefore, in model-based optimization only an imperfect model can be used for control computation. The imperfection in the model cannot be neglected because when this imperfect model is forward integrated, even a small model error can lead to significant error in the predicted trajectory, compared to the measured trajectory of the system.

The proposed HIL-OC approach does not use the imperfect model (18) to forward predict the state trajectory $\hat{\mathbf{x}}(\cdot)$, instead it uses the measured state trajectory $\mathbf{x}(\cdot)$ of the real system. Although the measured state does not allow us to eliminate the model parameters from the computation of the optimal control

[2]In Algorithm 1, the upper index denotes the iteration number while the lower index refers to the time instant: $\mathbf{x}_{j}^{i-1} = \mathbf{x}^{i-1}(t_j)$ and $\mathbf{u}_{j}^{i-1} = \mathbf{u}^{i-1}(t_j)$. Also, the time is discretized $t_j = \frac{j-1}{n_T-1}T \in [0, T]$, and n_T denotes the number of discrete time points.

Algorithm 2: Hardware-in-the-Loop Optimal Control.

Input: Feasible $\mathbf{u}^0(\cdot) \in \mathbb{U}$, robot
Output: $\mathbf{u}^*(\cdot) \in \mathbb{U}$
Initialize: $conv = false$, $i = 1$, $\epsilon_I > 0$, $\epsilon_u > 0$
$\mathbf{x}^0(\cdot) \leftarrow \left(\mathbf{u}^0(\cdot),\ \text{robot}\right)$, $\mathcal{I}^0 \leftarrow (\mathbf{x}^0(\cdot), \mathbf{u}^0(\cdot))$,
$\delta\mathbf{u}^0_{\alpha^*=1}(\cdot) = \mathbf{0}$
COMPUTATION and IMPLEMENTATION
while $conv = false$ **do**
  for $j = n_T : -1 : 1$ **do**
    Evaluate partial derivatives along $(\mathbf{x}^{i-1}_j, \mathbf{u}^{i-1}_j)$
    $\hat{\mathbf{R}}^{i-1}_{j-1}$, $\hat{\mathbf{r}}^{i-1}_{j-1} \leftarrow$ (19), (20), $\delta\mathbf{u}^{i-1}_{j\alpha^*}$
  end
  for $j = 1 : n_T$ **do**
    $\mathbf{x}^i_j \leftarrow$ robot
    $\delta\mathbf{x}^i_j \leftarrow \mathbf{x}^i_j - \mathbf{x}^{i-1}_j$
    $\delta\mathbf{u}^i_{j\alpha^*} \leftarrow$ (21), $\hat{\mathbf{R}}^{i-1}_j$, $\hat{\mathbf{r}}^{i-1}_j, \delta\mathbf{x}^i_j$,
    $\mathbf{u}^i_j \leftarrow$ (22)
    robot $\leftarrow \mathbf{u}^i_j$
  end
  $\mathcal{I}^i \leftarrow (\mathbf{x}^i(\cdot), \mathbf{u}^i(\cdot))$
  if $|\mathcal{I}^i - \mathcal{I}^{i-1}| \le \epsilon_I$ **and** $\sum_{j=1}^{n_T} \|\delta\mathbf{u}^i_{j\alpha^*}\| \le \epsilon_u$ **then**
    $\mathbf{u}^*(\cdot) = \mathbf{u}^i(\cdot)$, $conv = true$
  end
  $i = i + 1$
end

input, it allows us to evaluate the model parameters along the measured trajectory of the system

$$\hat{\mathbf{f}}_{\mathbf{x}} = \hat{\mathbf{f}}_{\mathbf{x}}(\mathbf{x}, \mathbf{u}) \quad \text{and} \quad \hat{\mathbf{f}}_{\mathbf{u}} = \hat{\mathbf{f}}_{\mathbf{u}}(\mathbf{x}, \mathbf{u})$$

instead of the trajectory $\hat{\mathbf{x}}$ predicted by the inexact model (18). This is the first modification proposed to reduce model bias in HIL-OC.

B. Decoupled Backward Integration

In the previous formulation, the control update $\delta\mathbf{u}^*$ is coupled with (13) that defines $\mathbf{r}$. Consequently, both $\delta\mathbf{u}^*$ and $\mathbf{r}$ have to be calculated backward in time. If $\delta\mathbf{u}^*$ could be decoupled from (13), then $\mathbf{r}$ could be computed backward in time offline while the control update $\delta\mathbf{u}^*$ could be computed, by minimizing the Hamiltonian (15) along the measured trajectory, forward in time online.

In order to decouple (13) from the unknown control update, we replace (12) and (13) with the following equations:

$$\dot{\hat{\mathbf{R}}}^i + \hat{\mathbf{R}}^i\hat{\mathbf{f}}_{\mathbf{x}} + \hat{\mathbf{f}}_{\mathbf{x}}^\top\hat{\mathbf{R}}^i + \mathcal{L}_{\mathbf{xx}} = \mathbf{0},\ \hat{\mathbf{R}}^i(T) = \phi_{\mathbf{xx}} \tag{19}$$

$$\dot{\hat{\mathbf{r}}}^i + (\mathbf{R}\hat{\mathbf{f}}_{\mathbf{u}} + \mathcal{L}_{\mathbf{xu}})\delta\mathbf{u}^{i-1} + \mathcal{L}_{\mathbf{x}} + \hat{\mathbf{f}}_{\mathbf{x}}^\top\hat{\mathbf{r}}^i = \mathbf{0},\ \hat{\mathbf{r}}^i(T) = \phi_{\mathbf{x}} \tag{20}$$

where the control update from the previous iteration $\delta\mathbf{u}^{i-1}$ is used to calculate $\hat{\mathbf{r}}^i$ in the next iteration. This enables us to compute both $\hat{\mathbf{R}}^i$ (19) and $\hat{\mathbf{r}}^i$ (20) offline. This is the second modification proposed in HIL-OC.

C. Control Update in Online Optimization

Since the parameters $\hat{\mathbf{R}}^i$ and $\hat{\mathbf{r}}^i$ are computed offline between two subsequent online iterations, the quadratic program (15) provides an implicit constrained feedback control law. This control law allows us to compute the control input based on the state measured forward in time online.

To compute the control input, we measure the state at $t = t_j$, calculate its variation with respect to the state taken from the previous iteration

$$\delta\mathbf{x}^i(t_j) = \mathbf{x}^i(t_j) - \mathbf{x}^{i-1}(t_j)$$

and compute the control update

$$\delta\mathbf{u}^i_{\alpha^*}(t_j) = \underset{\delta\mathbf{u}\in\delta\mathbb{U}^i(t_j)}{\arg\min}\ \frac{1}{2}\delta\mathbf{u}^\top\mathcal{L}_{\mathbf{uu}}\delta\mathbf{u} + \alpha^*\delta\mathbf{u}^\top\hat{\mathbf{b}}(\delta\mathbf{x}^i(t_j)) \tag{21}$$

$$\mathbf{u}^i(t_j) = \mathbf{u}^{i-1}(t_j) + \delta\mathbf{u}^i_{\alpha^*}(t_j) \in \mathbb{U} \tag{22}$$

where $\hat{\mathbf{b}}(\delta\mathbf{x}^i(t_j)) = (\mathcal{L}_{\mathbf{ux}} + \hat{\mathbf{f}}_{\mathbf{u}}^\top\hat{\mathbf{R}}^i)\delta\mathbf{x}^i(t_j) + (\mathcal{L}_{\mathbf{u}} + \hat{\mathbf{f}}_{\mathbf{u}}^\top\hat{\mathbf{r}}^i)$.

Subsequently, we apply the input (22) to the system (2), the robot which has the exact dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}^i(t_j))$. As a response, the robot moves to a new state $\mathbf{x}^i(t_{j+1})$. Although the exact dynamics is not known, the new state $\mathbf{x}^i(t_{j+1})$ can be measured, and the computation of the control input, (21) and (22), can be repeated at the next time step $t = t_{j+1}$.

D. Convergence and Optimality

The HIL-OC Algorithm 2 may be seen as a heuristic modification of prior successive approximation methods, for example, a hardware-in-the-loop version of iLQR or cDDP with no convergence and optimality guarantees. In this section, we present two theorems stating the convergence and optimality properties of Algorithm 2. These results extend to the cases when the optimal control problem is subject to control constraints, and when the model is inexact.

Theorem 1 (HIL-OC with exact model): When the model is exact $\hat{\mathbf{f}} = \mathbf{f}$, Algorithm 2 converges to a locally optimal feasible control, if such control exists; there exists a sequence of convergence control parameters $i \in \mathbb{N} : \{\alpha^i\} \in (0, 1]$ such that:

1) the cost sequence $\{\mathcal{I}^i\}$ monotonically decreases and converges
$$i \in \mathbb{N} : \mathcal{I}^i > \mathcal{I}^{i+1} \ge \lim_{i\to\infty}\mathcal{I}^i = \mathcal{I}^* > -\infty;$$
2) the sequence of control functions $\{\mathbf{u}^i(\cdot)\}$ converge to a feasible control function:
$$\lim_{i\to\infty}\mathbf{u}^i(\cdot) \to \mathbf{u}^*(\cdot), \quad \forall t \in \mathbb{T} : \mathbf{u}^*(t) \in \mathbb{U};$$
3) the control function $\mathbf{u}^*(\cdot)$ and the corresponding state trajectory $\mathbf{x}^*(\cdot)$ satisfy the first-order necessary conditions of local optimality for the constrained nonlinear optimal control problem (1), (2).

Proof: The proof is given in Appendix C. ■

Theorem 2 (HIL-OC with inexact model): When the model is inexact $\hat{\mathbf{f}} \neq \mathbf{f}$, Algorithm 2 terminates after finitely many iterations with a feasible suboptimal control; there exists a finite

sequence of convergence control parameters $i \in \{1, i^*\} \subset \mathbb{N}$: $\{\alpha^i\} \in [0,1]$ where $\alpha^{i^*} = 0$, such that:

1) the cost sequence $\{\mathcal{I}^i\}$ monotonically decreases and terminates with a finite value

$$\forall i \leq i^* : \mathcal{I}^i > \mathcal{I}^{i+1} \text{ and } i > i^* : \mathcal{I}^i = \mathcal{I}^* > -\infty;$$

2) the sequence of control functions $\{\mathbf{u}^i(\cdot)\}$ terminates at a feasible control function

$$i > i^* : \mathbf{u}^i(\cdot) = \mathbf{u}^*(\cdot) \quad \forall t \in \mathbb{T} : \mathbf{u}^*(t) \in \mathbb{U};$$

3) the terminal control function $\mathbf{u}^*(\cdot)$ and the corresponding state trajectory $\mathbf{x}^*(\cdot)$ satisfy the first-order necessary conditions of local optimality of the nonlinear optimal control problem (1), (2), defined using the inexact model $\hat{\mathbf{f}}$ but evaluated along the measured state trajectory of the system, which has the exact dynamics $\mathbf{f}$.

Proof: The proof is given in Appendix D. ■

Theorem 2 shows that when the model is inexact, Algorithm 2 converges and provides a feasible control, but the control is suboptimal compared to the control that could be obtained by knowing the exact model, see Theorem 1(3) and Theorem 2(3). In practice, suboptimality is unavoidable because the model is by definition inexact. One way to mitigate suboptimality is to learn a better model. This approach is taken by m-RL methods. Another way to mitigate suboptimality is to avoid making forward predictions with an inexact model. This approach is taken by the proposed HIL-OC method.

E. Summary

The three features presented in Sections IV-A–IV-C ensure that Algorithm 2 is not biased by error accumulation due to model-based forward prediction. It is also noteworthy that Algorithm 2 strictly enforces the control constraints along the actual motion, instead of the trajectory predicted by an imprecise model in forward simulation. The former point is an important contribution of the HIL-OC method, but the latter point is also notable because control constraints come from physical limitations which cannot be avoided to guarantee the feasibility of the iterations [1] and cannot be relaxed without performance degradation.

We note that the advantage of the HIL-OC method does not come from online optimization using model-based forward prediction, as promoted by MPC methods, or a better control due to better model information obtained through learning, as promoted by m-RL methods. The relation between MPC, m-RL, and the proposed HIL-OC approach is discussed in Section VII.

V. Performance Evaluation

In this section, we compare the HIL-OC approach with alternative model-based optimal control approaches, assuming that the exact dynamics of the system is unknown and only an inexact model is available. This is the case in practical application regardless of whether the model is analytically derived, identified offline, or learned online from data. We use the same model in all comparisons.

We consider a finite horizon linear quadratic optimization problem (cLQR). Using this problem, we numerically show significant reduction in the cost when using the HIL-OC method (see Algorithm 2) compared to a more typical model-based approaches (see Algorithm 1),[3] and the alternative online MPC method [46].

Consider the following finite-time horizon cLQR problem:

$$\min_{\mathbf{u}(\cdot)\in\mathcal{U}} \mathcal{I}[\mathbf{u}(\cdot)] = \min_{\mathbf{u}(\cdot)\in\mathcal{U}} \frac{1}{2}\int_0^T \left[\mathbf{x}^\top(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^\top(t)\mathbf{R}\mathbf{u}(t)\right]dt$$
$$\text{subject to: } \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \text{ and } \mathbf{x}(0) = \mathbf{x}_0 \tag{23}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{Q} \in \mathbb{P}^n_+$, $\mathbf{R} \in \mathbb{P}^m_{++}$, $\mathbf{A} = \mathbf{V}^\top \mathbf{\Lambda}\mathbf{V}, \mathbf{B} \in \mathbb{R}^{n\times m}, \mathbf{V} \in \mathbb{R}^{n\times n}$ is a randomly generated unitary orthogonal matrix $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$, and $\mathbf{\Lambda} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with uniformly distributed eigenvalues $\lambda(\mathbf{\Lambda}) \in [\lambda_{\min}, \lambda_{\max}]^n$.

As in (3), we assume that the inputs are box constrained

$$\forall t \in [0,T] : \mathbf{u}(t) \in \mathbb{U} = \{\mathbf{u} \in \mathbb{R}^m : \mathbf{u}_{\min} \preceq \mathbf{u} \preceq \mathbf{u}_{\max}\}$$

and that only the imperfect model (18) is known

$$\dot{\hat{\mathbf{x}}}(t) = \hat{\mathbf{A}}\hat{\mathbf{x}}(t) + \hat{\mathbf{B}}\mathbf{u}(t) = (\mathbf{A} + \Delta\mathbf{A})\hat{\mathbf{x}}(t) + (\mathbf{B} + \Delta\mathbf{B})\mathbf{u}(t) \tag{24}$$

where $\Delta\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\Delta\mathbf{B} \in \mathbb{R}^{n\times m}$ represent parametric uncertainties in the state and input matrices. We consider the following parametric uncertainties:

$$\Delta\mathbf{A} = s\mathbf{I}_n \quad \text{and} \quad \Delta\mathbf{B} = -s\mathbf{B} \tag{25}$$

where $s \in [0, s_{\max}] \subset [0,1)$; $s = 0$ means no parametric uncertainty while $s = s_{\max}$ denotes the largest parametric uncertainty. Increasing s makes the open-loop system more unstable, and decreases the magnitude of control input if $\mathbf{B}$ is an identity matrix. Under these conditions, (25) represents the worst-case parametric uncertainty.

In the following example, we consider the worst-case scenario in which the open-loop system (23) is unstable; the state matrix $\mathbf{A}$ has positive eigenvalues while $\mathbf{B}$ is an identity matrix.[4] Under these conditions, we compare:

1) The offline feed-forward optimal inputs compute using Algorithm 1 [see Fig. 1, red stars];
2) The locally optimal feedback controller where the feedback is computed online using Algorithm 1 [34] [see Fig. 1, blue triangles];
3) The MPC in [46] with the shortest time horizon ($n_t = 1 \ll n_T = 50$)[5];
4) The MPC in [46] with a reasonably long time horizon ($n_t = 20 < n_T = 50$) [see Fig. 1, green squares];
5) The proposed HIL-OC method computed using Algorithm 2 [see Fig. 1, black circles].

[3]This offline optimization method belongs to the class of successive approximation methods, similar to DDP, MSA, iLQR, and cDDP.

[4]For the existence of an unconstrained stabilizing LQR controller, we assume that $(\mathbf{A}, \mathbf{B})$ and $(\hat{\mathbf{A}}, \hat{\mathbf{B}})$ are stabilizable and $(\mathbf{A}, \sqrt{\mathbf{Q}})$ and $(\hat{\mathbf{A}}, \sqrt{\mathbf{Q}})$ are detectable.

[5]n_t denotes the number of time steps in the prediction horizon used to implement the MPC method while n_T denotes the number of time steps in the total time horizon.

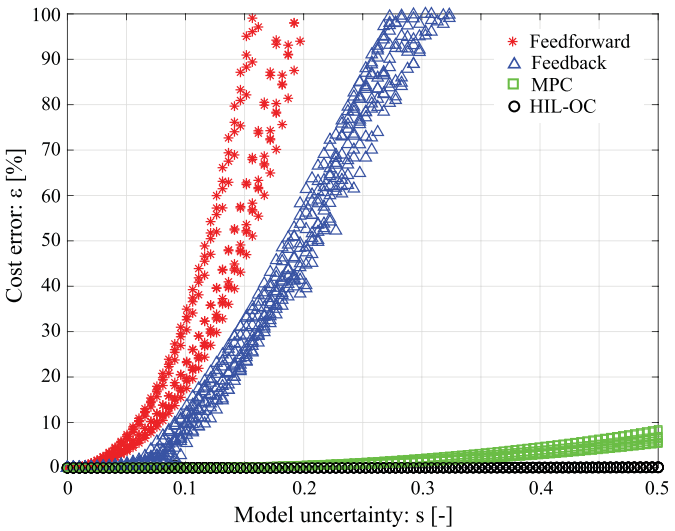


Fig. 1. Performance comparison. Cost error $\varepsilon = (\mathcal{I} - \mathcal{I}_{\text{opt}})/\mathcal{I}_{\text{opt}} \times 100\%$ for the HIL-OC method (black circles), feedforward (red stars), feedforward and online feedback (blue triangles), and the MPC method (green squares). In this example $n = 20$ is the number of states, $m = 20$ is the number of controls, $[\lambda_{\min}, \lambda_{\max}] = [-\frac{1}{4}, \frac{1}{4}]$ is the range of eigenvalues of the open-loop dynamics (the eigenvalues are defined by twenty uniformly selected grid-points in this range), $\mathbf{B} = \mathbf{I}_{20}$ is the control matrix, $\mathbb{U} = [-1, 1]^{20}$ is the admissible set of the control input, $\Delta t = 0.1$ is the time step, $T = 5$ is the time horizon, $[x_{0\min}, x_{0\max}] \in [-1, 1]$ is the range of initial conditions (the initial condition $\mathbf{x}_0$ is defined by 20 uniformly selected grid points in this range) while $\mathbf{Q} = \mathbf{I}_{20}$ and $\mathbf{R} = \mathbf{I}_{20}$ are the weights in the cost. We select hundred uniformly distributed grid points in $s \in [0, \frac{1}{2}]$, and for each of these points, we solve the optimization problem. Finally, we calculate the relative error in the cost ε with respect to the exact optimal cost $\mathcal{I}_{\text{opt}}$.

Figure 1 shows the cost obtained using the aforementioned optimization approaches. All methods recovered the numerically exact solution with the exact model ($s = 0$), except the short time horizon MPC method ($n_t = 1$, not shown). When the model error is small ($0 < s \leq 0.05$), all four stable methods are close to the numerically exact optimal solution; the error in the cost is less than $\varepsilon = (\mathcal{I} - \mathcal{I}_{\text{opt}})/\mathcal{I}_{\text{opt}} \times 100 < 5\%$. However, as the uncertainty increases ($s > 0.05$), the cost also increases. In particular:

1) The feed-forward optimal controller [Fig. 1, red stars] has large cost error ($\varepsilon \gg 100\%$) because it does not factor in the real system (23) into the computation of the control input.
2) The locally optimal feedback controller [Fig. 1, blue triangles] has smaller cost ($\varepsilon > 100\%$) because it uses feedback to compensate for the differences between the imperfect model (24), used to calculate the inputs, and the real system (23), used to evaluate the inputs. This controller shows the typical implementation of successive approximation methods [32]–[34], [44] where the feed-forward control sequence and the locally valid feedback is calculated using trajectories predicted by the inexact model.
3) The MPC method ($n_t = 1 \ll n_T = 50$) affords fast computation but is unstable due to the short prediction horizon, even if the model used to calculate the control inputs is exact ($s = 0$). This solution is not shown in Fig. 1.
4) The MPC method ($n_t = 20 < n_T = 50$) [Fig. 1, green squares] has an order of magnitude smaller cost error ($5\% < \varepsilon < 10\%$) because it uses the measured state trajectory generated by the exact dynamics (23).
5) Finally, the proposed HIL-OC method (Fig. 1, black circles) has the smallest cost error ($\varepsilon < 0.5\%$). This error is more than two orders of magnitude smaller than the one obtained using the offline feed-forward method 1) and the online feedback methods 2), and an order of magnitude smaller than the one obtained using the MPC ($n_t = 20$) method 4). In the HIL-OC method, the imperfect model is not used for forward prediction, and the constrained optimal feedback control inputs are calculated along the measured trajectory, the one generated by the system which has the exact dynamics.[6] These led to significant reduction of the cost, as shown by the black circles in Fig. 1.

We note that the feedback controller may be learned or computed by RL methods using a learned value function or Q-function [20] without using model based future prediction [26], [41], [42]. However, these methods require significant amount of data [47]. For example, even the simplest infinite horizon ($T \to \infty$) time-invariant approximation of the finite horizon ($T < \infty$) time-varying optimal controller in (23) requires the learning of a quadratic value function which has $\frac{1}{2}n \times (n - 1) = \frac{1}{2}20 \times 19 = 190$ unknown parameters. In finite horizon optimal control, the feedback controller is time-varying, even for the unconstrained LQR problem with time-invariant dynamics. Learning a time-varying feedback controller without model-based future prediction requires significant amount of data.

VI. Experiment

In this section, we use the proposed HIL-OC algorithm (see Algorithm 2) for hardware-in-the-loop control. We show how a three-link robot iteratively self-optimizes to perform a point to point reaching task. In this experiment, the optimal control problem is finite horizon, the cost and the dynamics are nonlinear, and the control inputs are subject to box constraints. This is a typical formulation in robot control application [1]. The purpose of the experiment is to demonstrate the advantage of the novel HIL-OC approach proposed in this paper.

A. Robot

The robot is shown in Fig. 2; it has three degrees of freedom and is driven by three motors. The configuration of the robot is defined using three generalized coordinates $\mathbf{q} = [q_1, q_2, q_3]^\top$. The robot is direct drive; the control inputs are the motor torques $\mathbf{u} = [u_1, u_2, u_3]^\top$. Due to motor torque limitations, these inputs are box constrained.

[6]When the MPC method was used with the exact dynamics $\mathbf{A}$ and $\mathbf{B}$ (this dynamics is not available in practice for the computation of the control inputs), the cost error was negligible. This can be seen in Fig. 1 where $\varepsilon \approx 0\%$ when $s \approx 0$. Consequently, the increased cost in Fig. 1 is not due to the finite horizon approximations used in MPC ($n_t = 20 < n_T = 50$), but due to the imperfection in the model ($s > 0$).

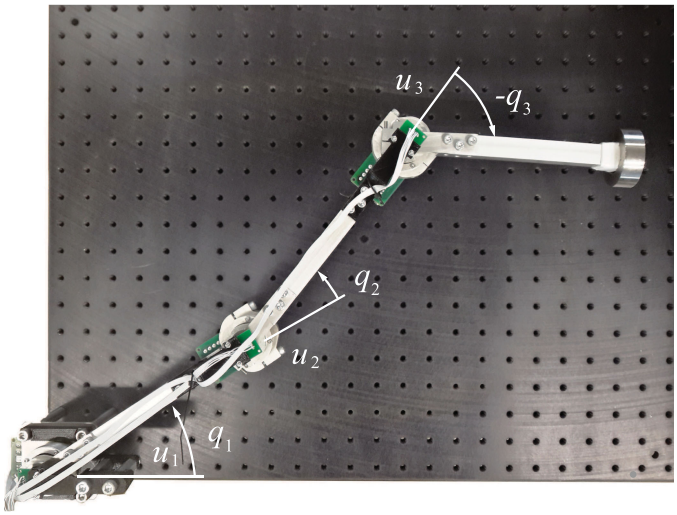


Fig. 2. Three-link robot. The length of the links is: $l_1 = l_2 = 0.2$ m and $l_3 = 0.15$ m. The masses of the links, joints, and the end-effector are: $m_1 = m_2 = m_3 = 0.21$ kg, $m_j = 0.14$ kg, and $m_e = 0.1$ kg. The joints are directly driven by three Maxon motors (EC45 flat). The joint angles are measured by magnetic encoders (AMS, AS5045).

B. Task

The task is to move from an initial point A to a final target point B, and then back to the initial point

$$A(x_A, y_A) \rightarrow B(x_B, y_B) \rightarrow A(x_A, y_A) \tag{26}$$

while minimizing the control effort and the error between the end-effector position $\mathbf{r}_E = [x_E, y_E]^\top$ and the desired end-effector position

$$\mathbf{e}(t, \mathbf{x}) = \mathbf{r}_E(\mathbf{x}) - \mathbf{r}_{Ed}(t).$$

The desired end-effector trajectory between AB is defined by

$$A \rightarrow B : \mathbf{r}_{Ed} = \begin{bmatrix} x_A \\ y_A + (y_A - y_B)(15\tau^4 - 6\tau^5 - 10\tau^3) \end{bmatrix}$$

where $\tau = 2t/T \in [0, 1]$ is the normalized time; it is defined similarly for the motion between BA. This desired trajectory is a minimum jerk trajectory previously used to explain human reaching [48].

C. Optimal Control Formulation

The task (26) is encoded by the following optimal control problem:

$$\min_{\mathbf{u}(\cdot)\in\mathcal{U}} \quad \mathcal{I}[\mathbf{u}(\cdot)] = \min_{\mathbf{u}(\cdot)\in\mathcal{U}} \int_0^T \big[\mathbf{e}(t, \mathbf{x}(t))^\top \mathbf{Q} \mathbf{e}(t, \mathbf{x}(t)) + \mathbf{u}^\top(t) \mathbf{R} \mathbf{u}(t)\big] dt$$
$$\text{subject to } \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \text{ and } \mathbf{x}(0) = \mathbf{x}_0 \tag{27}$$

where $\mathbf{x}(t) = [\mathbf{q}(t), \dot{\mathbf{q}}(t)]^\top \in \mathbb{R}^6$ is the state, $\forall t \in [0, T] : \mathbf{u}(t) \in \mathbb{U} = [\mathbf{u}_{\min}, \mathbf{u}_{\max}] \subset \mathbb{R}^3$ is the control input, $\mathbf{Q} \in \mathbb{P}^2_{++}$ and $\mathbf{R} \in \mathbb{P}^3_{++}$ are the weights in the cost, while $\mathbf{e}(t, \mathbf{x})$ is the error between the end-effector position and the desired end-effector position.

D. Results

Next, we show the result provided by the HIL-OC approach, where the robot is controlled to move from point A to B and back to point A while minimizing the position error $\mathbf{e}$ and the static motor power proportional to the squared control inputs. The exact dynamics of the robot $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is unknown, and we use Algorithm 2 with an imperfect analytical model $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ to implement the online optimization. The model we use is imperfect to the extent that it makes the offline optimal solution inadequate to control the robot, even if we use locally optimal feedback, and even if we use MPC to reoptimize the control inputs at every time step during the actual motion. We stress that all the methods in our comparisons use the same inexact model.

Figures 3 and 4 summarize the experimental results. Fig. 3 shows that (a) the HIL-OC method (see Algorithm 2) converges while the cost is reduced more than 90% within six iterations; (b), (c) the robot performs a typical minimum jerk motion; and (d)–(f) the control inputs are iteratively optimized within the constraints. Fig. 4(a)–(e) shows the frame sequence of the motion. Fig. 4(a) shows that the robot was started from rest. Fig. 4(e) shows the optimized motion. The video of the experiment is provided in the supplementary material. A detailed discussion of the results is given below.

Figure 3(a) shows three separate trials to verify the repeatability of the obtained result. For each trial, the cost converged within six iterations. The hardware-in-the-loop optimization was not sensitive to uncertainties including noise, feedback delay, unmodeled dynamics (e.g. friction and motor cables), and the algorithm effectively minimized the cost.

Figures 3(a) and 4(a)–(e) show that the optimization converged rapidly, within 1 minute. This result is notable, given that the nonlinear optimization problem solved here was started with completely uninformed initialization; the control input is set to zero at initialization [see Fig. 4(a)]. The hardware-in-the-loop optimization did not require judiciously chosen initialization, for example, one provided by human demonstration.

The online computation required to calculate the control inputs was less than 0.03 ms per time-step[7] to solve the small-scale constrained quadratic program (21) (this quadratic program had three decision variables and three box constraints). The offline computation was less than 60 ms to find the parameters (19), (20) between two subsequent iterations (this offline cost comes from time integration of two linear differential equations).

Due to the small number of inputs, this example did not challenge the real-time capability of the proposed algorithm. Nevertheless, in Section V, we optimized 20 control inputs subject to 20 box constraints. This optimization required 0.2 ms per time-step online computation, and 90 ms offline computation between subsequent iterations. Despite the large number of

[7]All calculations were performed using an Intel E3 2.8 GHz 32 GB RAM laptop computer. The constrained quadratic programs were solved using the online active set method [49] with warm start.

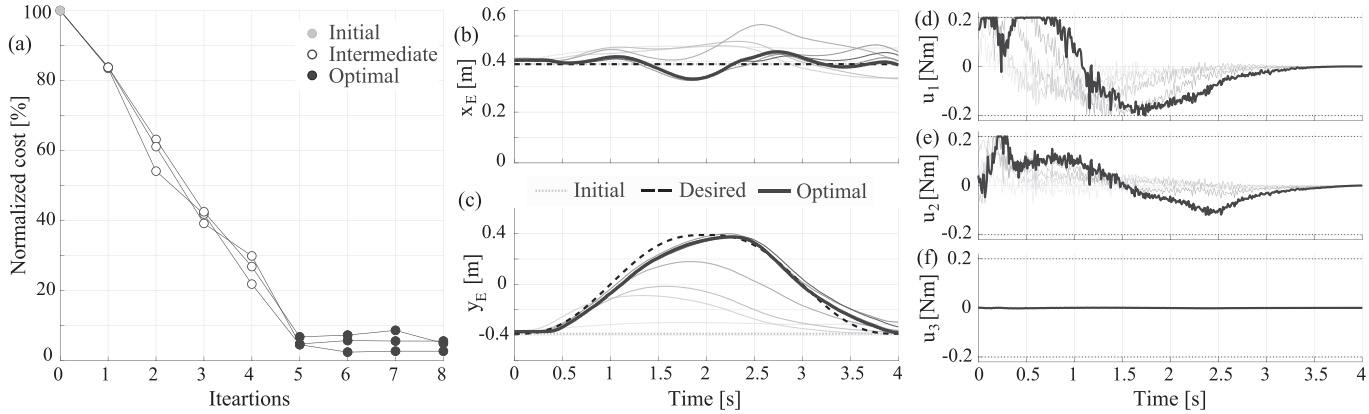


Fig. 3. Hardware-in-the-loop online optimization. (a) Total cost for each iteration of the three separate trials. (b), (c) End-effector trajectory x_E and y_E. (d)–(f) Motor torques $\mathbf{u}$. The final solution is denoted with thick black lines. The trajectories during iterations are denoted with gray lines. The initialization is denoted with gray dotted lines. The desired trajectory used to define the error in the cost is shown with dashed black lines. In this experiment, we set $T = 4$ s, $\mathbf{Q} = \text{diag}([500, 100])$, $\mathbf{R} = \text{diag}([1, 1, 50])$, $\mathbf{u}_{\min} = -0.2 \times \mathbf{I}_3$ Nm, $\mathbf{u}_{\max} = 0.2 \times \mathbf{I}_3$ N· m, and $\mathbf{x}_0 = [-\pi/4, 0, 0, 0, 0, 0]^\top$. The constraints on the inputs and the relatively large control cost on the third motor make the realization of this task challenging.

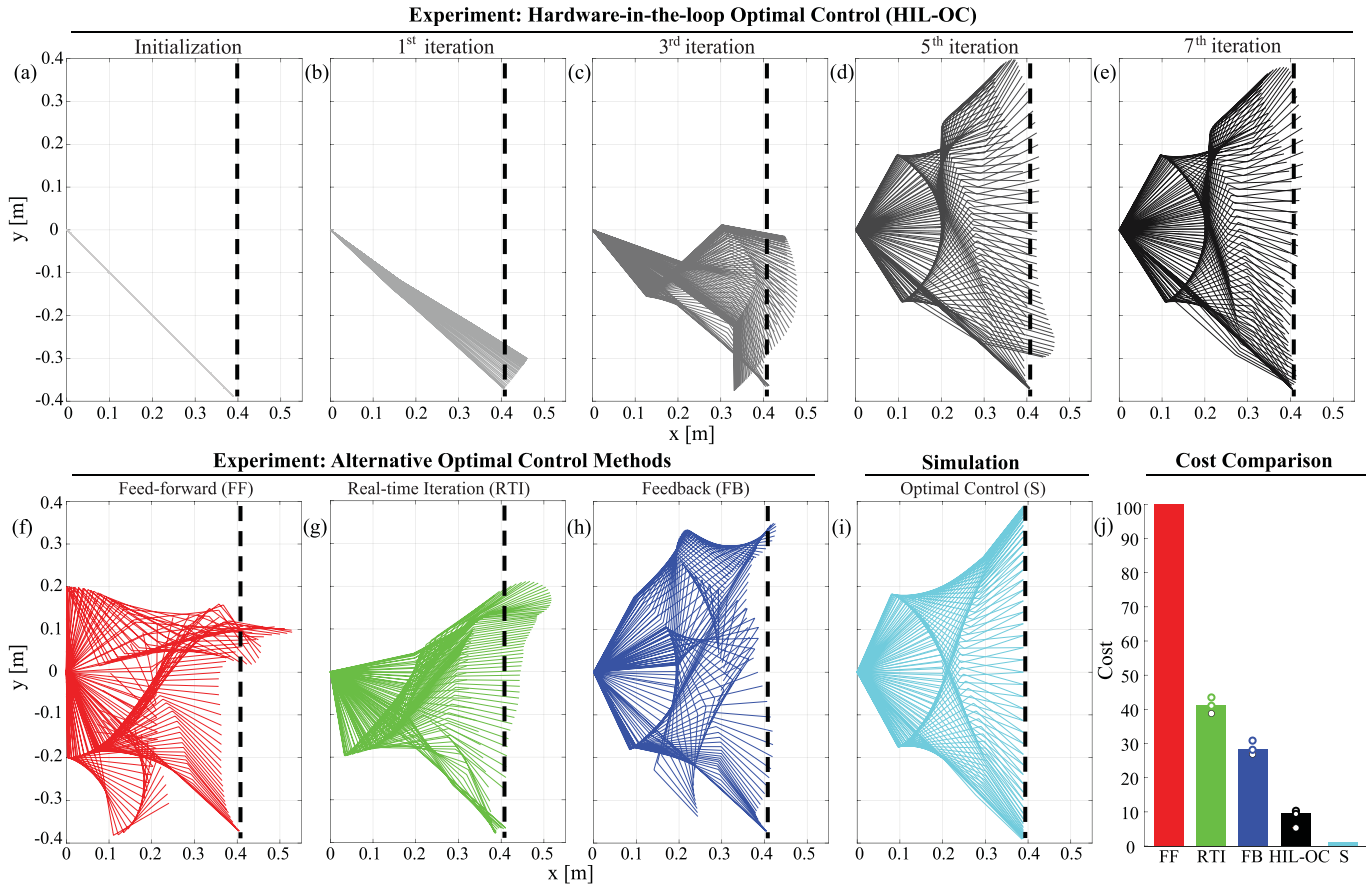


Fig. 4. Experimental frame sequence. (a)–(e) Online optimization. (a) Initialization $\mathbf{u} = \mathbf{0}$; (e) Optimized motion. The desired end-effector position used to define the error in the cost is shown with dashed black lines. (f)–(h) Comparison: Frame sequences of the robot controlled with (f) optimal feed-forward control inputs, (g) the reoptimized inputs at each step starting from the measured state, and (h) local optimal feedback controller. (i) Frame sequence of the numerically simulated robot using the inexact model. (j) Comparison of the cost with different controllers. The video of the experiments is provided in the supplemental material.

control inputs, this example has also not challenged the real-time capability of the proposed algorithm, even if we only allow 1 ms for the online computation. The computational time of 1 ms for 20 control inputs is appreciable for state-of-the-art operational space controllers [50], but is currently not within the reach of model predictive controllers [9], [34]. In summary, the amount of computation and the distribution between online and offline computation make the proposed method well suited to real-time implementation.

Fig. 3(d)–(f) show the control inputs (motor torques) which are subject to box constraints $\mathbf{u} \in \mathbb{U}$. During the iterations, the intermediate inputs are denoted with gray lines, while the final optimized inputs are shown with black lines. We note that the inputs of the first two motors u_1 and u_2 are constrained at the beginning of the motion $t \in [0, 1]$ s. The temporal pattern of these inputs suggests that the links are initially accelerated and then decelerated at the middle of the motion. This pattern originates from the target minimum jerk trajectory [see Fig. 3(b) and (c)]. We can also see that the third input u_3 is small along the entire motion. This is partly because moving the third motor is costly due to the high running cost $\mathbf{R}_{33} = 50$[8] compared to $\mathbf{R}_{11} = \mathbf{R}_{22} = 1$, but also because the third motor appears largely redundant to realize the two-dimensional line following task with the three link robot.

E. Comparison

A fair comparison between different optimal control methods necessitates the use of the same objective and the same system model. In this section, we compare four alternative optimal control approaches. We consider:

1) The proposed method (see Algorithm 2) [HIL-OC, see Fig. 4(e)];
2) The optimal feed-forward control method (see Algorithm 1) [FF, see Fig. 4(f)];
3) The locally optimal feedback control method (see Algorithm 1 where (14) is used to compute the feedback corrections online) [FB, see Fig. 4(h)];
4) The real-time iteration-based MPC method (Algorithm 1, rerun at every time step starting from the measured state of the system) [RTI, see Fig. 4(g)].

In all cases, we use the same objective and the same model.

When we use the exact model in simulation, all four methods 1)–4) recovered the same optimal solution. This is shown in Fig. 4(i), where the frame sequences of the robot obtained with the different methods are overlapped. However, in the experiment, the model is inexact, and the proposed HIL-OC method outperforms all three alternative control methods [see Fig. 4(j), black bar 10%]. This is despite the fact that all these methods use the same model.

The feed-forward optimal control is sensitive to model imperfections, and as such it is not surprising that the feed-forward control has the largest cost [FF, see Fig. 4(j) red bar 100%]. However, the online computed locally optimal feedback controller, which can adequately deal with small model imperfections, could also not reduce the cost effectively in the considered task [FB, see Fig. 4(j) blue bar 30%]. This is because in direct drive torque-controlled robots, model uncertainty can significantly affect the motion. This effect could not be mitigated even with real-time reoptimization of the control inputs at every time step [RTI, see Fig. 4(j) green bar 40%].

Our implementation of the real-time reoptimization presented in [34] (controller II) follows the spirit of the real-time iteration RTI method, originally proposed in [7]. We use Algorithm 1 to do the online iterations with 20 ms time step between each reoptimization. Our RTI implementation recovers the optimal solution under exact model information in simulation. However, the same real-time reoptimization did not perform adequately in online implementation [see Fig. 4(g)]; the imperfect model information, and the relatively long time horizon, made our RTI-base MPC implementation underperform the local feedback controller in the experiment [see Fig. 4(g), (h), (j)]. This happened when we used the theoretically ideal initialization of the control function and state trajectories given by the model-based optimal solution [see Fig. 4(i)], but also when we used the HIL-OC solution for initialization; the latter provided the lowest cost in hardware-in-the-loop control [see Fig. 4(e)]. Our RTI implementation uses a single iteration within each quadratic program, as proposed in [7], together with warm start. The lowest cost achieved using this method is shown in Fig. 4(j) [RTI, green bar 40%].

In the same task, the cost of the ideal simulation was 1%, as shown in Fig. 4(j) light blue bar. Therefore, the cost of the HIL-OC method [see Fig. 4(j), black bar 10%] could be further reduced by learning a better model. However, according to Fig. 4(j), the cost of the HIL-OC method cannot be reduced more than 10%, even if one can learn the exact model of the robot. The 10% reduction, achievable by model learning, appears limited compared to the 90% reduction provided by the HIL-OC method [see Fig. 4(j) black bar]. A similar conclusion can be drawn from the example in Section V (see Fig. 1).

VII. Conclusion

In this paper, we suggested an iterative algorithm that uses measured data and an imperfect model of the system to do HIL-OC. The idea is to replace the model in the forward prediction with measured data, and to use the model parameters evaluated along the measured trajectory for backward integration only. The model in HIL-OC can come from first principles [35], offline system identification [36] as in typical MPC methods, or online learning as in m-RL methods [37]. In any case, the model is imperfect, and the advantage of the HIL-OC method over MPC and m-RL methods comes from no model-based forward prediction. Consequently, in HIL-OC, the error due to the imperfect model is not accumulated; this was discussed in Section VII-A. The additional advantage of the HIL-OC method over MPC is the reduced online computational cost; this was discussed in Section VII-B. Finally, the advantage of the HIL-OC method over the bulk of m-RL approaches is the rigorous treatment of

[8]The high running cost associated with the third motor in this optimization problem makes the line following task challenging.

the constraints during the actual motion; this was discussed in Section VII-C.

A. Reduced Model Bias

In DDP, MSA, iLQR, and cDDP [29]–[33], the control input is calculated using trajectories predicted by the imperfect model and the feed-forward control update is calculated by backward integration in time (similar to Algorithm 1). Compared to these methods, Algorithm 2 uses the measured trajectory of the system instead of the trajectory predicted by the model, and the control update is calculated by minimizing the Hamiltonian along the actual motion, instead of minimizing it in a numerical simulation.

The resulting HIL-OC method can significantly reduce the cost, compared to model prediction based optimal control approaches, given the same imperfect model. In Section V, we carried out 4000 simulations to compare the cost of a cLQR optimization problem solved with: the HIL-OC approach (see Algorithm 2); an alternative optimal control approach that uses model-based future prediction (see Algorithm 1); and the MPC method presented in [46]. The results showed that the HIL-OC method can significantly reduce model bias [see Fig. 1].

B. Computational Cost

In order to compute the control update, Algorithm 2 involves two main calculations: 1) it calculates the parameters in the controller (19) and (20) along the measured trajectory backward in time between the iterations, and 2) it solved the constrained quadratic program (21) at each time step. The first calculation requires the integration of two linear differential equations with $\frac{1}{2}n \times (n+1) + n$ unknowns (where $\mathbf{x} \in \mathbb{R}^n$). The second calculation solves a static optimization problem which has m decision variables subject to m box constraints ($\mathbf{u} \in \mathbb{R}^m$). Notably, this was the smallest size static optimization problem to find the control input $\mathbf{u} \in \mathbb{R}^m$ for the dynamic optimization problem (1) and (2). This feature enables computationally efficient HIL-OC of systems driven by a large number of inputs.

In MPC methods, the optimization problem to be solved online for the nonlinear problem (1), (2), or the linear-quadratic time-varying problem (4), (5), had $n_t \cdot m$ decision variables subject to $n_t \cdot m$ box constraints, where n_t denotes the number of sampling points in the reduced prediction horizon optimization problem solved at each time step. The number of time points n_t cannot be small due to stability constraints (see Section V), and because shorter time horizon leads to greedy optimization and significant performance degradation [9]. Consequently, MPC formulations that can handle the nonlinear optimal control problem (1), (2) lead to a comparatively large-scale constrained optimization to be solved in each time step; their computational complexity scales at least linearly with the time horizon n_t [34] (the typical scaling is cubic n_t^3 [7], [10]). In contrast, the online computational complexity of HIL-OC does not depend on the time horizon. This means that even for a relatively short time horizon, for example, $n_t = 10$, an alternative MPC formulation requires at least an order of magnitude larger time step.

Reducing the amount of time required for online optimization leads to a one time step horizon MPC ($n_t = 1$) method [28] applicable to a subclass of linear time-invariant systems. Extension of this result to constrained nonlinear optimal control remains challenging[9] [9]. For hardware-in-the-loop control, the online computational cost is critical because it is limited by the real-time step used in sample time control implementation. In case of robot control, a 1 ms time step for a system driven with 20 inputs is desirable, and achievable with the HIL-OC method, but not within the reach of state-of-the-art MPC methods, including methods that promote efficient real-time implementation [7], [10], [11], [34], [51].

The online computational cost of a feedback controller is less than the online computational cost of the proposed controller because feedback gains were typically precomputed along the model predicted trajectories offline, and then applied to control the real system online [32], [33], [44]. However, feedback controllers that use gains precomputed along model predicted trajectories introduced model bias that leads to increased cost [see Fig. 1 blue triangles versus black circles and Fig. 4(j) blue versus black bars].

C. Control Constraints

Incorporating constraints into continuous-time RL formulations is possible [26], [38], [41], [42] but challenging [27]; it requires complex function approximation to adequately capture the relation between the state and the control input in presence of constraints because constraints lead to nonsmooth value function [52] violating the basic smoothness assumption of continuous-time RL formulations.

Many of the prior m-RL formulations used to solve finite time horizon optimal control problems [15], [16], [20], [23] do not take into account control constraints. This allows for a simple linear feedback control (LQR) calculation. Such linear relation extends to constrained time-invariant linear quadratic optimization [6], but does not extend to constrained nonlinear optimization (1), (2).

One way to incorporate control constraints into RL formulations is to use a nonquadratic control cost that penalizes constraint violation. This approach was taken by a number of RL formulations [26], [27], [41], [42]. Another way to incorporate control constraints into m-RL formulations is to apply constrained optimal control along the trajectory predicted by an inexact model. For example, one could use iLQR [32], cDDP [33], [44], or the method to embed the control constraints into the dynamic model [1], to find the constrained feed-forward control sequence, and the feedback gains, which are optimal for the trajectory predicted by the inexact model. Alternatively, one could use the m-RL approach [17], [18] to estimate the optimal search direction for the constrained control inputs using the model offline, and then perform line search until the cost decreases online.

[9]The one time step horizon MPC ($n_t = 1$) method was unstable for the cLQR problem considered in Section V.

Regardless of which method is used, controllers that use trajectories predicted by inexact models to find the search direction do not factor in the constraints, according to the principle of optimality during the actual motion. Similar to analytical models derived from first principles, models learned from data are imperfect, and a search direction that is calculated using a model predicted trajectory can significantly differ from the search direction that is calculated along a measured trajectory. This is especially true in presence of constraints (3), unstable dynamics (see Section V), and longer time horizon optimization (see Section VI).

D. Future Work

The proposed HIL-OC can be used with a model derived from first principles or a learned model. Although the benefit provided by the HIL-OC does not come from model learning, using a better model can further reduce the cost, as shown in Section V and Fig. 1. In iterative m-RL [16]–[18], [23], the type of RL that builds on the idea of iterative learning [53], [54], the model is updated offline between subsequent iterations. This type of offline system identification can be incorporated into the HIL-OC formulation, without giving up the fast online computation. The combination of the HIL-OC method with model learning, and the convergence analysis of such combined approach, is a part of our future work.

APPENDIX

In the following, we discuss the convergence and optimality of the proposed iterative algorithm.

A. Assumptions

A1) For any admissible control $\mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$ and $t \in [0, T]$, the state is bounded $\mathbf{x}(t) \in \mathbb{X} \subset \mathbb{R}^n$.

A2) The dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u})$, cost $\mathcal{L}(\mathbf{x}, \mathbf{u}, t)$, and their first- and second-order partial derivatives are continuous on $(\mathbf{x}, \mathbf{u}, t) \in \mathbb{R}^n \times \mathbb{U} \times [0, T]$.

A3) The parameters of the exact linear dynamics $\mathbf{f_x}$ and $\mathbf{f_u}$ are bounded with respect to the parameters of the linear model $\hat{\mathbf{f}}_\mathbf{x}$ and $\hat{\mathbf{f}}_\mathbf{u}$.

A4) The cost $\mathcal{L}(\mathbf{x}, \mathbf{u}, t)$ is a convex function for all $t \in [0, T]$ with respect to $\mathbf{u}$ along the optimal solution $(\mathbf{x}^*, \mathbf{u}^*)$.

The above assumptions imply the following conditions:

$$\|\mathbf{x}\| \leq c_\mathbf{x} < \infty, \|\mathbf{u}\| \leq c_\mathbf{u} < \infty \tag{28}$$

$$\|\mathbf{f_x}\|, \|\mathbf{f_u}\| \leq c_\mathbf{f} < \infty \tag{29}$$

$$\|\mathcal{L}_\mathbf{x}\|, \|\mathcal{L}_\mathbf{u}\|, \|\mathcal{L}_\mathbf{xx}\|, \|\mathcal{L}_\mathbf{uu}\|, \|\mathcal{L}_\mathbf{xu}\|, \|\mathcal{L}_\mathbf{xu}\| \leq c_\mathcal{L} < \infty \tag{30}$$

$$\|\hat{\mathbf{f}}_\mathbf{x} - \mathbf{f_x}\| \leq \epsilon_\mathbf{x} < \infty, \|\hat{\mathbf{f}}_\mathbf{u} - \mathbf{f_u}\| \leq \epsilon_\mathbf{u} < \infty \tag{31}$$

$$\mathcal{L}_\mathbf{uu}(\mathbf{x}^*, \mathbf{u}^*, t) \succ \mathbf{0}. \tag{32}$$

B. Estimate of the Cost Variation

The following proposition provides an upper bound for the cost variation as a function of the control update. This proposition is used in Appendices C and D to prove Theorems 1 and 2 presented in Section IV-D.

Proposition 1: When solving (1), (2) using Algorithm 2, at iteration $i \in \mathbb{N}_+$, there exist $c_u \in (0, \infty)$ and $c_\epsilon \in [0, \infty)$, such that the following inequality holds:

$$\Delta\mathcal{I}^i \leq \int_0^T \left[-\left(\frac{p_{\min}}{2\alpha} - c_u \right) \|\delta\mathbf{u}^i\|^2 + c_\epsilon \|\delta\mathbf{u}^i\| \right] dt.$$

Proof: For the sake of simplicity, we consider the Lagrange problem of optimal control[10] where

$$\Delta\mathcal{I}^i = \int_0^T \Delta\mathcal{L}(\delta\mathbf{x}^i, \delta\mathbf{u}^i, t) dt \tag{33}$$

is the change of the cost in the i-th iteration. In the derivation, we use the following approximate Hamiltonian function:

$$\Delta\hat{\mathcal{H}} = \Delta\mathcal{L}(\delta\mathbf{x}^i, \delta\mathbf{u}^i) + \hat{\boldsymbol{\lambda}}^\top \underbrace{(\mathbf{f_x}\delta\mathbf{x}^i + \mathbf{f_u}\delta\mathbf{u}^i)}_{\delta\dot{\mathbf{x}}^i} \tag{34}$$

where $\delta\dot{\mathbf{x}}^i$ refers to the exact linear dynamics, and $\hat{\boldsymbol{\lambda}}$ is the approximate costate

$$\hat{\boldsymbol{\lambda}} = \hat{\mathbf{R}}\delta\mathbf{x}^i + \hat{\mathbf{r}} \tag{35}$$

where the parameters $\hat{\mathbf{R}}$ and $\hat{\mathbf{r}}$ are calculated using (19) and (20).

The cost in each subproblem (33) is given by

$$\begin{aligned}
\Delta\mathcal{I}^i &= \int_0^T (\Delta\hat{\mathcal{H}} - \hat{\boldsymbol{\lambda}}^\top \delta\dot{\mathbf{x}}^i) dt \\
&= \int_0^T \left(\frac{\partial\Delta\hat{\mathcal{H}}}{\partial\delta\mathbf{u}}\delta\mathbf{u}^i - \frac{1}{2}(\delta\mathbf{u}^i)^\top \mathcal{L}_\mathbf{uu}\delta\mathbf{u}^i \right) dt \\
&\quad + \int_0^T \left(\frac{\partial\Delta\hat{\mathcal{H}}}{\partial\delta\mathbf{x}}\delta\mathbf{x}^i - \hat{\boldsymbol{\lambda}}^\top \delta\dot{\mathbf{x}}^i \right) dt \\
&\quad - \int_0^T \left(\frac{1}{2}(\delta\mathbf{x}^i)^\top \mathcal{L}_\mathbf{xx}\delta\mathbf{x}^i + (\delta\mathbf{u}^i)^\top \mathcal{L}_\mathbf{ux}\delta\mathbf{x}^i \right) dt.
\end{aligned} \tag{36}$$

1) Let us now consider the first integral in (36). The first term in this integral is

$$\begin{aligned}
\frac{\partial\Delta\hat{\mathcal{H}}}{\partial\delta\mathbf{u}}\delta\mathbf{u}^i = (\delta\mathbf{u}^i)^\top \mathcal{L}_\mathbf{uu}\delta\mathbf{u}^i + (\delta\mathbf{u}^i)^\top \mathcal{L}_\mathbf{ux}\delta\mathbf{x}^i + \mathcal{L}_\mathbf{u}\delta\mathbf{u}^i \\
+ \hat{\boldsymbol{\lambda}}^\top \mathbf{f_u}\delta\mathbf{u}^i.
\end{aligned} \tag{37}$$

The cQP (21) used to calculate the control update implies

$$(\delta\mathbf{u}^i)^\top \mathcal{L}_\mathbf{uu}\delta\mathbf{u}^i + \alpha[(\delta\mathbf{u}^i)^\top \mathcal{L}_\mathbf{ux}\delta\mathbf{x}^i + \mathcal{L}_\mathbf{u}\delta\mathbf{u}^i + \hat{\boldsymbol{\lambda}}^\top \hat{\mathbf{f}}_\mathbf{u}\delta\mathbf{u}^i] \leq 0$$

where $\alpha \in [0, 1]$.

If the Hessian $\mathcal{L}_\mathbf{uu}$ is not positive definite, then we replace it with a positive definite matrix[11]

$$\mathcal{L}^{\text{PD}}_\mathbf{uu} \in \begin{cases} \mathcal{L}_\mathbf{uu}, & \text{if } \mathcal{L}_\mathbf{uu} \succ \mathbf{0} \\ \mathcal{L}^{\text{PD}}_\mathbf{uu}, & \text{if } \mathcal{L}_\mathbf{uu} \preceq \mathbf{0} \end{cases}.$$

[10]The Bolza problem (1) can be transformed to the Lagrange problem. The reader may refer to [55] for the details of the transformation.

[11]We compute $\mathcal{L}^{\text{PD}}_\mathbf{uu}$ using the modified Cholesky factorization of $\mathcal{L}_\mathbf{uu}$ [43].

Using this modified Hessian[12], we transform the above inequality relation into

$$(\delta\mathbf{u}^i)^\top \frac{1}{\alpha}\mathcal{L}_{\mathbf{uu}}^{\text{PD}}\delta\mathbf{u}^i + (\delta\mathbf{u}^i)^\top \mathcal{L}_{\mathbf{ux}}\delta\mathbf{x}^i + \mathcal{L}_{\mathbf{u}}\delta\mathbf{u}^i + \hat{\boldsymbol{\lambda}}^\top \hat{\mathbf{f}}_{\mathbf{u}}\delta\mathbf{u}^i \le 0 \tag{38}$$

where $\alpha \neq 0$.

Based on (37) and (38), we have the following relation:

$$\frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{u}}\delta\mathbf{u}^i \le -(\delta\mathbf{u}^i)^\top\left(\frac{1}{\alpha}\mathcal{L}_{\mathbf{uu}}^{\text{PD}} - \mathcal{L}_{\mathbf{uu}}\right)\delta\mathbf{u} + \hat{\boldsymbol{\lambda}}^\top(\mathbf{f}_{\mathbf{u}} - \hat{\mathbf{f}}_{\mathbf{u}})\delta\mathbf{u}^i.$$

Also, an upper bound of the first integral in (36) can be derived

$$\begin{aligned}
&\int_0^T \left(\frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{u}}\delta\mathbf{u}^i - \frac{1}{2}(\delta\mathbf{u}^i)^\top \mathcal{L}_{\mathbf{uu}}\delta\mathbf{u}^i\right) dt \\
&\le -\int_0^T \frac{1}{2\alpha}(2-\alpha)(\delta\mathbf{u}^i)^\top \mathcal{L}_{\mathbf{uu}}^{\text{PD}}\delta\mathbf{u}^i dt \\
&\quad + \int_0^T \hat{\boldsymbol{\lambda}}^\top(\mathbf{f}_{\mathbf{u}} - \hat{\mathbf{f}}_{\mathbf{u}})\delta\mathbf{u}^i dt \\
&\le \int_0^T \left[-\frac{1}{2\alpha}(\delta\mathbf{u}^i)^\top \mathcal{L}_{\mathbf{uu}}^{\text{PD}}\delta\mathbf{u}^i + \hat{\boldsymbol{\lambda}}^\top(\mathbf{f}_{\mathbf{u}} - \hat{\mathbf{f}}_{\mathbf{u}})\delta\mathbf{u}^i\right] dt.
\end{aligned} \tag{39}$$

2) The second integral in (36) is

$$\begin{aligned}
&\int_0^T \left(\frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\delta\mathbf{x}^i - \hat{\boldsymbol{\lambda}}^\top \delta\dot{\mathbf{x}}^i\right) dt \\
&= \int_0^T \left(\frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\delta\mathbf{x}^i - \frac{d}{dt}(\hat{\boldsymbol{\lambda}}^\top \delta\mathbf{x}^i) + \dot{\hat{\boldsymbol{\lambda}}}^\top \delta\mathbf{x}^i\right) dt \\
&= \int_0^T \left(\dot{\hat{\boldsymbol{\lambda}}} + \frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\right)\delta\mathbf{x}^i dt - \hat{\boldsymbol{\lambda}}^\top \delta\mathbf{x}^i\Big|_0^T \\
&= \int_0^T \left(\dot{\hat{\boldsymbol{\lambda}}} + \frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\right)\delta\mathbf{x}^i dt.
\end{aligned} \tag{40}$$

Substituting (39) and (40) into (36), we obtain the following inequality relation:

$$\begin{aligned}
\Delta\mathcal{I}^i \le \int_0^T \Bigg[&-\frac{1}{2\alpha}(\delta\mathbf{u}^i)^\top \mathcal{L}_{\mathbf{uu}}^{\text{PD}}\delta\mathbf{u}^i + \hat{\boldsymbol{\lambda}}^\top(\mathbf{f}_{\mathbf{u}} - \hat{\mathbf{f}}_{\mathbf{u}})\delta\mathbf{u}^i \\
&+ \left(\dot{\hat{\boldsymbol{\lambda}}} + \frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\right)\delta\mathbf{x}^i \\
&- \frac{1}{2}(\delta\mathbf{x}^i)^\top \mathcal{L}_{\mathbf{xx}}\delta\mathbf{x}^i - (\delta\mathbf{u}^i)^\top \mathcal{L}_{\mathbf{ux}}\delta\mathbf{x}^i\Bigg] dt,
\end{aligned} \tag{41}$$

where (19), (20), (34), and (35) can be used to define

$$\begin{aligned}
\dot{\hat{\boldsymbol{\lambda}}} + \frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}} = &(\mathbf{f}_{\mathbf{x}} - \hat{\mathbf{f}}_{\mathbf{x}})^\top \hat{\mathbf{R}}\delta\mathbf{x}^i + \hat{\mathbf{R}}(\mathbf{f}_{\mathbf{x}} - \hat{\mathbf{f}}_{\mathbf{x}})\delta\mathbf{x}^i \\
&+ \hat{\mathbf{R}}(\mathbf{f}_{\mathbf{u}}\delta\mathbf{u}^i - \hat{\mathbf{f}}_{\mathbf{u}}\delta\mathbf{u}^{i-1}) + \mathcal{L}_{\mathbf{xu}}(\delta\mathbf{u}^i - \delta\mathbf{u}^{i-1}) \\
&+ (\mathbf{f}_{\mathbf{x}} - \hat{\mathbf{f}}_{\mathbf{x}})^\top \hat{\mathbf{r}}.
\end{aligned} \tag{42}$$

[12]This modification does not affect the optimal solution of the original problem as long as $\mathcal{L}_{\mathbf{uu}}$ is positive definite along the optimal solution (32).

If the model is exact $\mathbf{f}_{\mathbf{x}} - \hat{\mathbf{f}}_{\mathbf{x}} = \mathbf{0}$, $\mathbf{f}_{\mathbf{u}} - \hat{\mathbf{f}}_{\mathbf{u}} = \mathbf{0}$ and the control update is the same in two subsequent iterations $\delta\mathbf{u}^i = \delta\mathbf{u}^{i-1}$, then $\dot{\hat{\boldsymbol{\lambda}}} + \partial\Delta\hat{\mathcal{H}}/\partial\delta\mathbf{x} = \mathbf{0}$ is the costate equation (one of the necessary conditions of optimality) according to the Maximum Principle.

Next, we convert (41) into the following inequality relation:

$$\begin{aligned}
\Delta\mathcal{I}^i \le \int_0^T \Bigg[&-\frac{1}{2\alpha}p_{\min}\|\delta\mathbf{u}^i\|^2 + \|\hat{\boldsymbol{\lambda}}\|\|\mathbf{f}_{\mathbf{u}} - \hat{\mathbf{f}}_{\mathbf{u}}\|\|\delta\mathbf{u}^i\| \\
&+ \left\|\dot{\hat{\boldsymbol{\lambda}}} + \frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\right\|\|\delta\mathbf{x}^i\| \\
&+ \frac{1}{2}\|\mathcal{L}_{\mathbf{xx}}\|\|\delta\mathbf{x}^i\|^2 + \|\mathcal{L}_{\mathbf{ux}}\|\|\delta\mathbf{u}^i\|\|\delta\mathbf{x}^i\|\Bigg] dt
\end{aligned} \tag{43}$$

where $p_{\min} > 0$ is the smallest eigenvalue of $\mathcal{L}_{\mathbf{uu}}^{\text{PD}}$,

$$\|\hat{\boldsymbol{\lambda}}\| \le \|\hat{\mathbf{R}}\|\|\delta\mathbf{x}^i\| + \|\hat{\mathbf{r}}\| \tag{44}$$

is the upper bound of the costate (35), and

$$\begin{aligned}
\|\dot{\hat{\boldsymbol{\lambda}}} + \frac{\partial \Delta\hat{\mathcal{H}}}{\partial \delta\mathbf{x}}\| \le &2\|\mathbf{f}_{\mathbf{x}} - \hat{\mathbf{f}}_{\mathbf{x}}\|\|\hat{\mathbf{R}}\|\|\delta\mathbf{x}^i\| \\
&+ \|\hat{\mathbf{R}}\|(\|\mathbf{f}_{\mathbf{u}}\|\|\delta\mathbf{u}^i\| + \|\hat{\mathbf{f}}_{\mathbf{u}}\|\|\delta\mathbf{u}^{i-1}\|) \\
&+ \|\mathcal{L}_{\mathbf{xu}}\|(\|\delta\mathbf{u}^i\| + \|\delta\mathbf{u}^{i-1}\|) \\
&+ \|\mathbf{f}_{\mathbf{x}} - \hat{\mathbf{f}}_{\mathbf{x}}\|\|\hat{\mathbf{r}}\|
\end{aligned} \tag{45}$$

is the upper bound of the error in the costate equation (42).

Using Schwarz's inequality, Gronwall's inequality, and the linear dynamics (5), one can derive the following relations:

$$\int_0^T \|\delta\mathbf{x}^i\| dt \le e^{c_{\mathbf{f}}T}\int_0^T \|\delta\mathbf{u}^i\| dt \tag{46}$$

$$\int_0^T \|\delta\mathbf{x}^i\|^2 dt \le \frac{c_{\mathbf{f}}^2 T^2 e^{2c_{\mathbf{f}}T}}{2}\int_0^T \|\delta\mathbf{u}^i\|^2 dt \tag{47}$$

$$\int_0^T \|\delta\mathbf{x}^i\|\cdot\|\delta\mathbf{u}^i\| dt \le \frac{c_{\mathbf{f}}Te^{c_{\mathbf{f}}T}}{\sqrt{2}}\int_0^T \|\delta\mathbf{u}^i\|^2 dt. \tag{48}$$

Using Schwarz's inequality, one can also show that there exists $c_{\delta\mathbf{u}}^2 \in (0,\infty)$ such that the following relation holds for any $\delta\mathbf{u}^i$ and $\delta\mathbf{u}^{i-1}$:

$$\int_0^T \|\delta\mathbf{x}^i\|\|\delta\mathbf{u}^{i-1}\| dt \le \frac{c_{\mathbf{f}}Te^{c_{\mathbf{f}}T}c_{\delta\mathbf{u}}^{-1}}{\sqrt{2}}\int_0^T \|\delta\mathbf{u}^i\|^2 dt. \tag{49}$$

When $\int_0^T \|\delta\mathbf{u}^{i-1}\|^2 dt = 0$, the above relation is valid for any $\int_0^T \|\delta\mathbf{u}^i\|^2 dt$. When $\int_0^T \|\delta\mathbf{u}^i\|^2 dt = 0$, the above relation follows from (47) for any $\int_0^T \|\delta\mathbf{u}^{i-1}\|^2 dt$. If $\int_0^T \|\delta\mathbf{u}^{i-1}\|^2 dt \neq 0$ and $\int_0^T \|\delta\mathbf{u}^i\|^2 dt \neq 0$, then the above relation is valid for $\forall c_{\delta\mathbf{u}}^2 \in (0, \int_0^T \|\delta\mathbf{u}^i\|^2 dt/\int_0^T \|\delta\mathbf{u}^{i-1}\|^2 dt]$.

Finally, using Gronwall's inequality, one can show that the solution of the two linear equations (19) and (20) is bounded

$$\|\hat{\mathbf{R}}\| \le c_{\mathbf{R}}, \quad \|\hat{\mathbf{r}}\| \le c_{\mathbf{r}} \tag{50}$$

where

$$c_{\mathbf{R}} = \frac{c_{\mathcal{L}}}{2(c_{\mathbf{f}}+\epsilon_{\mathbf{x}})}(e^{2(c_{\mathbf{f}}+\epsilon_{\mathbf{x}})T}-1)$$

$$c_{\mathbf{r}} = \frac{2c_{\mathbf{u}}[c_{\mathbf{R}}(c_{\mathbf{f}}+\epsilon_{\mathbf{u}})+c_{\mathcal{L}}]+c_{\mathcal{L}}}{c_{\mathbf{f}}+\epsilon_{\mathbf{x}}}(e^{(c_{\mathbf{f}}+\epsilon_{\mathbf{x}})T}-1)$$

because all partial derivatives of the dynamics and the cost are bounded (29) and (30), and because (28) implies that the control update in (20) is bounded, $\|\delta\mathbf{u}^{i-1}\| = \|\mathbf{u}^{i-1}-\mathbf{u}^{i-2}\| \leq \|\mathbf{u}^{i-1}\| + \|\mathbf{u}^{i-2}\| \leq 2c_{\mathbf{u}}$.

Based on (44)–(50), relation (43) reduces to

$$\Delta\mathcal{I}^i \leq \int_0^T \left[-\left(\frac{p_{\min}}{2\alpha}-c_u\right)\|\delta\mathbf{u}^i\|^2 + c_\epsilon\|\delta\mathbf{u}^i\|\right]dt \quad (51)$$

where

$$c_\epsilon = c_{\mathbf{r}}(\epsilon_{\mathbf{x}}e^{c_{\mathbf{f}}T}+\epsilon_{\mathbf{u}})$$

$$c_u = \left(\frac{1}{2}c_{\mathcal{L}}+2c_{\mathbf{R}}\epsilon_{\mathbf{x}}\right)\frac{c_{\mathbf{f}}^2T^2e^{2c_{\mathbf{f}}T}}{2} + c_{\mathbf{R}}(3c_{\mathbf{f}}+\epsilon_{\mathbf{u}})\frac{c_{\mathbf{f}}Te^{c_{\mathbf{f}}T}}{\sqrt{2}} + [c_{\mathcal{L}}+c_{\mathbf{R}}(c_{\mathbf{f}}+\epsilon_{\mathbf{u}})]\frac{c_{\mathbf{f}}Te^{c_{\mathbf{f}}T}c_{\delta\mathbf{u}}^{-1}}{\sqrt{2}}.$$

■

C. Proof of Theorem 1

Proof: 1) Assuming that the model is exact $c_\epsilon = 0$, the cost (51),

$$\Delta\mathcal{I}^i \leq -\left(\frac{p_{\min}}{2\alpha}-c_{u0}\right)\int_0^T\|\delta\mathbf{u}^i\|^2dt \quad (52)$$

will decrease for

$$\forall\alpha\in(0,\alpha_{\max}) = (0,1]\cap\left(0,\frac{p_{\min}}{2c_{u0}}\right)$$

where $c_{u0} = c_u|_{\epsilon_{\mathbf{x}}=\epsilon_{\mathbf{u}}=0}$. Because $p_{\min}/2c_{u0} > 0$, there exist $\alpha\in(0,\alpha_{\max})$ for which the cost decreases. Consequently, a backtracking algorithm, started with $\alpha = 1$, is guaranteed to find $\alpha^*\in(0,1]$ for which the cost (52) decreases.

2) When both the control input and the state are bounded (28), the cost is lower bounded

$$-\infty < \inf_{\mathbf{u}\in\mathbb{U}}\mathcal{I}(\mathbf{u}) \leq \mathcal{I}(\mathbf{u}).$$

Based on (52), the sum of the control updates is bounded

$$\sum_{i=0}^{\infty}\int_0^T\|\delta\mathbf{u}^i\|^2dt \leq c^{-1}\sum_{i=0}^{\infty}-\Delta\mathcal{I}^i \leq c^{-1}\left(\mathcal{I}^0-\inf_{\mathbf{u}\in\mathbb{U}}\mathcal{I}(\mathbf{u})\right)<\infty$$

where $c = \min_{i\in\mathbb{N}}[p_{\min}/2\alpha^i - c_{u0}^i] < \infty$. Consequently

$$\lim_{i\to\infty}\int_0^T\|\delta\mathbf{u}^i\|^2dt = 0. \quad (53)$$

According to (53) and (47), the control and state variations are zero at convergence,

$$\delta\mathbf{u}^*(t) = \mathbf{0} \text{ and } \delta\mathbf{x}^*(t) = \mathbf{0}$$

for almost all $t\in[0,T]$. Therefore, the sequence of control and state trajectories $\{\mathbf{u}^i\}$ and $\{\mathbf{x}^i\}$ converge to $\mathbf{u}^*$ and $\mathbf{x}^*$ in the space of square integrable functions.

3) When the model is exact, (20) and (21) provide

$$\dot{\mathbf{r}}^* + \mathcal{L}_{\mathbf{x}}(\mathbf{x}^*,\mathbf{u}^*,t) + \mathbf{f}_{\mathbf{x}}^\top(\mathbf{x}^*,\mathbf{u}^*)\mathbf{r}^* = \mathbf{0},\quad \mathbf{r}^*(T) = \mathbf{0} \quad (54)$$

$$[\mathcal{L}_{\mathbf{u}}(\mathbf{x}^*,\mathbf{u}^*,t) + \mathbf{f}_{\mathbf{u}}^\top(\mathbf{x}^*,\mathbf{u}^*)\mathbf{r}^*](\mathbf{u}^*-\mathbf{u}) \leq 0 \quad \forall\mathbf{u}\in\mathbb{U} \quad (55)$$

where $\exists\epsilon > 0 : \|\mathbf{u}^*-\mathbf{u}\| \leq \epsilon$, for almost all $t\in[0,T]$.

These conditions represent the local necessary conditions of optimality according to PMP [3]. As an implication, $\mathbf{x}^*$ and $\mathbf{u}^*$ satisfy the first-order necessary conditions of optimality (54) and (55) for almost all $t\in[0,T]$. ■

D. Proof of Theorem 2

Proof: 1) The worst-case analysis we used to obtain the upper bound of the cost (51) suggests that if the model is inexact, then the cost can decrease or remain the same under the following condition:

$$\exists\alpha\in[0,\alpha_{\max}) = [0,1]\cap\left[0,\frac{p_{\min}}{2c_u}\right) : \frac{2\alpha c_\epsilon}{p_{\min}-2\alpha c_u} \leq \int_0^T\|\delta\mathbf{u}^i\|dt \leq c_{\max}\alpha \quad (56)$$

where $c_{\max} = c_{\mathcal{L}}c_{\mathbf{x}} + c_{\mathcal{L}} + (c_{\mathbf{R}}c_{\mathbf{x}}+c_{\mathbf{r}})(c_{\mathbf{f}}+\epsilon_{\mathbf{u}}) < \infty$.

The inequality on the right side is derived from (38); it provides an upper bound on the integrated control update. The inequality on the left side is derived from $\Delta\mathcal{I}^i \leq 0$; it shows that the model error introduces a lower bound on the integrated control update.

If there exist $\alpha^*\in(0,\alpha_{\max})$ such that (56) holds, then α^* can be found by bracketing the minimum of $\Delta\mathcal{I}^i$ in [0,1]. This can be done using the golden-section search and parabolic interpolation method [56]. A necessary, but not sufficient, condition for $\alpha^*\in(0,\alpha_{\max})$ to exist is given by $0\leq c_\epsilon < c_{\max}p_{\min}/2$. It is easier to satisfy this condition for smaller model error. For no model error $c_\epsilon = 0$, this condition is always satisfied [see (56)], and it is also sufficient[13] [see (52)]. If there is no $\alpha^*\in(0,\alpha_{\max})$ for (56) to hold, then the cost cannot be further decreased, and the algorithm terminates with $\alpha^* = 0$.

2) The proposed algorithm cannot increase the cost because in the worst case $\alpha^* = 0$, (56) leads to $\int_0^T\|\delta\mathbf{u}^i\|dt = 0$, and therefore, the cost in the next iteration remains the same as the cost in the previous iteration.

When the model error $c_\epsilon > 0$ prevents the control update to converge to zero, the algorithm can terminate after finitely many iterations $i = i^* < \infty$ under the following condition:

$$\alpha^{i^*} = \alpha^{i^*+1} = 0 : \int_0^T\|\delta\mathbf{u}^{i^*}\|dt = \int_0^T\|\delta\mathbf{u}^{i^*+1}\|dt = 0.$$

[13]The worst-case analysis we use here to obtain the upper bound of the cost (51) indicates that decreasing the convergence control parameter α to zero does not guarantee descent direction for the control update when the model is inexact. This result does not show that given a small to moderate model error, the control update can still be in the descent direction of the cost as α is reduced; something we observe in our numerical calculations in Section V.

The solution $\mathbf{u}^* = \mathbf{u}^i$ and $\mathbf{x}^* = \mathbf{x}^i$ will be suboptimal; it will not satisfy the necessary conditions of optimality derived using the exact model (54) and (55).

3) When the model is inexact $c_\epsilon > 0$, the control and state trajectories $\mathbf{u}^*, \mathbf{x}^*$ satisfy the first-order necessary conditions of optimality defined by the inexact model, but evaluated along the measured trajectory of the system

$$\dot{\mathbf{r}}^* + \mathcal{L}_{\mathbf{x}}(\mathbf{x}^*, \mathbf{u}^*, t) + \hat{\mathbf{f}}_{\mathbf{x}}^\top(\mathbf{x}^*, \mathbf{u}^*)\mathbf{r}^* = \mathbf{0}, \quad \mathbf{r}^*(T) = \mathbf{0}$$

$$[\mathcal{L}_{\mathbf{u}}(\mathbf{x}^*, \mathbf{u}^*, t) + \hat{\mathbf{f}}_{\mathbf{u}}^\top(\mathbf{x}^*, \mathbf{u}^*)\mathbf{r}^*](\mathbf{u}^* - \mathbf{u}) \leq 0 \quad \forall \mathbf{u} \in \mathbb{U}$$

where $\exists \epsilon > 0 : \|\mathbf{u}^* - \mathbf{u}\| \leq \epsilon$, for almost all $t \in [0, T]$. ■

In the aforementioned conditions, all parameters are evaluated along the measured trajectory instead of the trajectory predicted by the inexact model $\hat{\mathbf{x}}^*$. Consequently, the suboptimality of the solution is not due to the error contaminated trajectory predicted by the inexact model (18).

REFERENCES

[1] D. J. Braun *et al.*, "Robots driven by compliant actuators: Optimal control under actuation constraints," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1085–1101, Oct. 2013.

[2] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[3] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko, *The Mathematical Theory of Optimal Processes*. Hoboken, NJ, USA: Wiley, 1962.

[4] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1218, 1998.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[6] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[7] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[8] M. Abdolhosseini, Y. Zhang, and C. A. Rabbath, "An efficient model predictive control scheme for an unmanned quadrotor helicopter," *J. Intell. Robot. Syst.*, vol. 70, no. 1–4, pp. 27–38, 2013.

[9] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, pp. 2967–2986, 2014.

[10] M. Vukov *et al.*, "Real-time nonlinear MPC and MHE for a large-scale mechatronic application," *Control Eng. Practice*, vol. 45, pp. 64–78, 2015.

[11] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1261–1285, 2016.

[12] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 10–17, Jan. 2017.

[13] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. Rehg, and E. A. Theodorou, "Robust sampling based model predictive control with sparse objective information," in *Proc. Robot. Sci. Syst.*, Pittsburgh, PA, USA, pp. 1–8, 2018.

[14] C. Liu, H. Li, J. Gao, and D. Xu, "Robust self-triggered min–max model predictive control for discrete-time nonlinear systems," *Automatica*, vol. 89, pp. 333–339, 2018.

[15] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proc. Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 1997, pp. 12–20.

[16] J. van den Berg *et al.*, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, 2010, pp. 2074–2081.

[17] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, 2006, pp. 1–8.

[18] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *Int. J. Robot. Res.*, vol. 29, no. 13, pp. 1608–1639, 2010.

[19] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. Int. Conf. Mach. Learn.*, Bellevue, Washington, USA, 2011, pp. 465–472.

[20] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[21] S. Levine and V. Koltun, "Guided policy search," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 1–9.

[22] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Proc. Advances Neural Inf. Process. Syst.*, Montreal, QC, Cananda, 2014, pp. 1907–1915.

[23] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Stockholm, Sweden, 2016, pp. 378–383.

[24] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "PLATO: Policy learning using adaptive trajectory optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, Singapore, 2017, pp. 3342–3349.

[25] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, QLD, Australia, 2018, pp. 7559–7566.

[26] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[27] F. A. Yaghmaie and D. J. Braun, "Reinforcement learning for a class of continuous-time input constrained optimal control problems," *Automatica*, vol. 99, pp. 221–227, 2019.

[28] C. Müller, D. E. Quevedo, and G. C. Goodwin, "How good is quantized model predictive control with horizon one?" *IEEE Trans. Autom. Control*, vol. 56, no. 11, pp. 2623–2638, Nov. 2011.

[29] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York, NY, USA: Elsevier, 1970.

[30] S. K. Mitter, "Successive approximation methods for the solution of optimal control problems," *Automatica*, vol. 3, pp. 135–149, 1966.

[31] F. L. Chernousko and A. A. Lyubushin, "Method of successive approximations for solution of optimal control problems," *Optimal Control Appl. Methods*, vol. 3, pp. 101–114, 1982.

[32] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system," *Int. J. Control*, vol. 80, no. 9, pp. 1439–1453, 2007.

[33] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, China, 2014, pp. 1168–1175.

[34] Y. Chen, L. Roveda, and D. J. Braun, "Efficiently computable constrained optimal feedback controllers," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 121–128, Jan. 2019.

[35] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2006.

[36] L. Ljung, *System Identification: Theory for the User*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.

[37] S. Ross and J. A. Bagnell, "Agnostic system identification for model-based reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Edinburgh, U.K., 2012, pp. 1703–1710.

[38] R. Munos, "A study of reinforcement learning in the continuous case by the means of viscosity solutions," *Mach. Learn.*, vol. 40, no. 3, pp. 265–299, 2000.

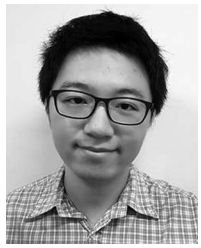
[39] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, "Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1725–1737, Nov. 2007.

[40] A. Heydari and S. N. Balakrishnan, "Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 1, pp. 145–157, Jan. 2013.

[41] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[42] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1513–1525, Oct. 2013.

[43] J. Nocedal and S. Wright, *Numerical Optimization*. New York, NY, USA: Springer-Verlag, 2006.
[44] D. Mitrović, S. Nagashima, S. Klanke, T. Matsubara, and S. Vijayakumar, "Optimal feedback control for anthropomorphic manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, 2010, pp. 4143–4150.
[45] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Vilamoura, Portugal, 2012, pp. 4906–4913.
[46] P. O. M. Scokaert and J. B. Rawlings, "Constrained linear quadratic regulation," *IEEE Trans. Autom. Control*, vol. 43, no. 8, pp. 1163–1169, Aug. 1998.
[47] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, no. 1, pp. 253–279, 2019.
[48] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, 1985.
[49] M. J. Best, "An algorithm for the solution of the parametric quadratic programming problem," *Applied Mathematics and Parallel Computing*. Heidelberg, Germany: Springer, 1996, pp. 57–76.
[50] A. Del Prete and N. Mansard, "Robustness to joint-torque-tracking errors in task-space inverse dynamics," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1091–1105, Oct. 2016.
[51] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *J. Guid., Control, Dyn.*, vol. 37, no. 6, pp. 1725–1740, 2014.
[52] F. H. Clarke and R. B. Vinter, "The relation between the maximum principle and dynamic programming," *SIAM J. Control Optim.*, vol. 25, no. 5, pp. 1291–1311, 1987.
[53] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *J. Robot. Syst.*, vol. 1, pp. 123–140, 1984.
[54] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "Survey of iterative learning control: A learning-based method for high-performance tracking control," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 96–114, Jun. 2006.
[55] L. Cesari, *Optimization Theory and Applications: Lagrange and Bolza Problems of Optimal Control and Other Problems*. New York, NY, USA: Springer-Verlag, 1983.
[56] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007.



Yuqing Chen received the B.Eng. and M.S. degrees in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2013 and 2015, respectively. Since 2015, he has been working toward the Ph.D. degree with the Singapore University of Technology and Design.

His research is focused on optimal control of dynamical systems.

David J. Braun (M'09) received the Ph.D. degree in mechanical engineering from Vanderbilt University (VU), Nashville, TN, USA, in 2009.

He is an Assistant Professor of Mechanical Engineering at VU. Prior to joining VU, he was a Visiting Researcher with the Institute for Robotics and Mechatronics, German Aerospace Center, a Postdoctoral Research Fellow with the Statistical Machine Learning and Motor Control Group, University of Edinburgh, and an Assistant Professor with the Singapore University of Technology and Design. His research interests include dynamical systems, optimal control, and robotics.

Dr. Braun received the 2013 IEEE TRANSACTIONS ON ROBOTICS Best Paper Award. He was the Scientific Program Cochair of the 2015 IEEE International Conference on Rehabilitation Robotics, Area Chair for the 2018 Robotics Science and Systems Conference, and Associate Editor of the 2020 IEEE International Conference on Robotics and Automation.