Experience-driven Predictive Control

Vishnu R. Desaraju and Nathan Michael
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{rajeswar, nmichael}@cmu.edu

***Abstract*—This work presents Experience-driven Predictive Control (EPC) as a fast technique for solving nonlinear model predictive control problems with uncertain system dynamics. EPC leverages a linear dynamics model that is updated online via Locally Weighted Project Regression (LWPR) to capture nonlinearities, uncertainty, and changes in the system dynamics. This allows the NMPC problem to be re-cast as a quadratic program. The QP can then be solved via multi-parametric techniques to generate a mapping from state, reference, and dynamics model to a locally optimal, affine feedback control law. These mappings, in conjunction with the basis functions learned in LWPR, define a notion of experience for the controller as they capture the full input-output relationship for previous actions the controller has taken. The resulting experience database allows EPC to avoid solving redundant optimization problems, and as it is constructed online, enables the system to operate more efficiently over time. We demonstrate the performance of EPC through a set of simulation studies with a quadrotor micro aerial vehicle that is subjected to unmodeled exogenous perturbations.**

I. Introduction

As robots are deployed in complex and unknown real-world environments, the ability to track trajectories accurately becomes essential for safety. However, this can be particularly difficult if the robot's dynamics change online, e.g., due to environmental effects or hardware degradation. Furthermore, operation in these types of environments may preclude reliable, high-rate communication with a base station, and as a result, the robot must be able to operate safely and reliable with typically limited onboard computational resources. Therefore, in this work we aim to develop an intelligent, computationally-efficient, feedback control strategy that enables accurate and reliable operation in the presence of unmodeled system dynamics.

High-rate adaptive control is easily achieved via feedback control techniques, such as model-reference adaptive control [16] and L1 adaptive control [25]. However, this simplicity may be at the expense of safety, as such methods do not provide constraint satisfaction guarantees. Additionally, these purely reactive techniques seek to eliminate the effects of unmodeled dynamics, even when they may be beneficial. In contrast, model predictive control (MPC) techniques seek to balance the reactive nature of traditional feedback controllers and the anticipative nature of infinite-horizon optimal control techniques. Consequently, they can yield improved trajectory tracking via finite-horizon optimization while reducing the computational complexity relative to infinite-horizon formulations.

However, performance of these predictive approaches is largely dependent on the accuracy of the prediction model. When applied to a linear system, or a system that does not deviate significantly from a nominal operating point, the linear MPC problem can be formulated and solved efficiently as either a constrained linear or quadratic program [13]. However, if the operating range can deviate greatly from a nominal linearization point, the formulation must account for the nonlinear dynamics to ensure the optimization is performed with respect to an accurate prediction of system evolution. Moreover, even a fixed nonlinear model may be insufficient to accurately predict the system's motion due to modeling errors and unmodeled dynamics. The use of a nonlinear dynamics model also significantly increases the computational complexity of the resulting nonlinear MPC (NMPC) problem, which must be formulated as a constrained nonlinear program.

Therefore, there are two key challenges that must be addressed in order to apply NMPC to challenging control problems: maintaining an accurate model of uncertain, time-varying dynamics and reducing complexity to increase computational efficiency.

A. Model Accuracy

The issue of model accuracy for predictive control has been addressed through various adaptation and learning-based approaches. Most existing adaptive MPC approaches assume a structured system model with uncertain parameters that can be estimated online. These approaches then combine a standard MPC formulation with an online parameter estimator, e.g., a Luenberger observer or Kalman filter, to achieve more accurate, deliberative actions [2, 9, 11].

However, treating all model uncertainty as parameters to estimate can limit the overall model accuracy, especially when the system is subject to complex, exogenous perturbations, such as aerodynamic effects on an aerial vehicle. Learning-based function approximation techniques can be applied to address this issue. The resulting semi-structured approaches augment a structured system model with a non-parametric, online-learned component, e.g., via a Gaussian process [17]. The resulting model is then queried within the NMPC formulation while continuing to adapt to model changes. While techniques such as Gaussian process regression scale poorly with the amount of training data, another kernel-based approach, Locally Weighted Projection Regression (LWPR), summarizes training data using linear basis functions [24]. The resulting

incremental updates enable fast model learning that is suitable for finite-horizon control [15].

B. Computational Efficiency

Computational efficiency can be evaluated in terms of increased solution speed and decreased redundant computation. For linear MPC formulations, there are a variety of techniques aimed at increasing solution speed. Several of these approaches leverage efficient convex optimization techniques [7, 20] and exploit matrix structure in the LP or QP formulations [26] to compute solutions quickly. Alternatively, explicit MPC approaches precompute the optimal linear MPC solutions for a polytopic decomposition of the state space, reducing the complexity of online computation [1, 6]. Other approaches, such as partial enumeration (PE) [18], balance the strengths of the online and offline approaches and demonstrate fast solution times on very large problems.

While some fast, online NMPC solution techniques have been developed, they rely on iterative, approximate solution techniques built around fast convex optimization solvers [3, 5, 12]. Consequently, they inherently cannot achieve the solution speeds attained by linear MPC formulations. Explicit NMPC [10] moves the optimization offline to achieve high-speed online control, but it is known to scale poorly as the resulting lookup table grows exponentially with the horizon length and number of constraints. As a result, NMPC has not been amenable to high-rate, realtime operation, particularly on computationally constrained systems. The nonlinear partial enumeration (NPE) algorithm [4] combines linear and nonlinear formulations to achieve high-rate predictive control with a nonlinear model, while also improving performance over time to better approximate the NMPC solution. However, its dependence on nonlinear optimization for performance improvement limits scalability and the rate at which performance improves.

While some MPC algorithms seek to reduce the amount of redundant computation performed by reusing past solutions [7], they still must solve an optimization problem at every control iteration. PE-based techniques achieve greater efficiency through the online creation of a controller database, which dramatically reduces the number of optimization problems that must be solved. However, since they assume the dynamics model is fixed and accurate, the controllers produced are invalidated if the dynamics change.

The construction of a database from past actions in order to facilitate choosing future actions is also the foundation of transfer learning and lifelong learning algorithms. These learning-based approaches consider executing tasks, which, by analogy to the PE approaches, can be viewed as a particular state-reference sequence. Transfer learning seeks to use knowledge about past tasks to bootstrap learning a new task [23], similar to efficient MPC strategies [7]. Lifelong learning shares similarities with the PE approaches in that it makes this knowledge transfer bidirectional to learn policies that maximize performance over all past and present tasks [21]. However, the PE approaches maintain a finite set of controllers that are updated through infrequent computation and do not permit interpolation. Whereas lifelong learning algorithms, such as ELLA [21] or OMTL [22], maintain a set of bases that aid in reconstructing task models whenever new data is received.

Therefore, we propose an Experience-driven Predictive Control (EPC) methodology that combines aspects of NPE with online model learning via LWPR. As in the PE techniques, EPC leverages an online-updated database of past experiences in order to achieve high-rate, locally-optimal feedback control with constraint satisfaction. However, we also parameterize the learned feedback control laws by the system dynamics, enabling online adaptation to model perturbations.

II. Approach

In this section, we present the Experience-driven Predictive Control (EPC) algorithm for fast, adaptive, nonlinear model predictive control. In the context of predictive control, we first define experience to be the relationship between previous states, references, and system dynamics models and the optimal control law applied at that time. Past dynamics models capture the effects of uncertainty on observed system evolution, while previous states capture the system's behavior under optimal control policies for a given dynamics model. Therefore, EPC constructs and leverages a two-part representation of past experiences to improve the accuracy of its finite-horizon lookahead. The first is the set of linear basis functions maintained by the Locally Weighted Projection Regression (LWPR) algorithm that capture observed variations in the system dynamics. The second is a mapping from states and references to locally optimal controllers that is updated online and is parameterized by the current estimate of the vehicle dynamics.

A. Online Model Adaptation via LWPR

Predictive control techniques for nonlinear systems employ either a nonlinear dynamics model, which incurs the complexity of solving nonlinear programs, or a more computationally efficient local approximation of the nonlinear dynamics. Therefore, given the nonlinear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, nominal state $\mathbf{x}^*$ and nominal control $\mathbf{u}^*$, we define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$ and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$ and derive an affine approximation of the dynamics via a first-order Taylor series expansion, $\bar{\mathbf{x}}_{k+1}^{\text{nom}} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c}$. We can then extend this model with an online-learned component via LWPR, which estimates perturbations to the nominal model, including nonlinearities, modeling errors, and unmodeled exogenous forces.

LWPR models a nonlinear function (from an input $\mathbf{z}$ to an output $\mathbf{y}$) by a Gaussian-weighted combination of local linear functions [24]. These basis functions encapsulate all past dynamics information, in contrast to Gaussian processes, which require storing all past training data. New linear functions are added as required when the existing set of bases are insufficient to represent new data with the desired accuracy. It also has a forgetting factor to control rate of adaptation to model changes by adjusting the effects of prediction error on the weight for each basis. As a result, LWPR is robust to uninformative

or redundant data, can retain information capturing all past experience, and can adapt its estimate to changing dynamics.

LWPR updates its estimate incrementally via partial least squares, which has $\mathcal{O}(|\mathbf{z}|)$ complexity, making it well-suited to real-time operation. Partial least squares projects the inputs onto a lower dimensional space defined by projection direction vectors $\boldsymbol{\nu}_r$ and $\boldsymbol{\rho}_r$, as detailed in [24]. It also computes slope coefficients β_r for each projection direction and an offset β_0 to generate a prediction of a scalar output. Therefore, following Mitrovic, et al. [15], we fit the dynamics model element-wise: for the i^{th} element in $\mathbf{y}$, local linear model j (with r_j projection directions) is given by

$$\begin{aligned}\Psi_j(\mathbf{z}) &= \beta_0 + \begin{bmatrix}\beta_1, \dots, \beta_{r_j}\end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1^{\mathrm{T}} \\ \boldsymbol{\nu}_2^{\mathrm{T}}\mathbf{P}_1 \\ \vdots \\ \boldsymbol{\nu}_{r_j}^{\mathrm{T}}(\mathbf{P}_1 \cdots \mathbf{P}_{r_j-1}) \end{bmatrix} (\mathbf{z} - \mathbf{m}_j) \\ &= \alpha_j + \boldsymbol{\beta}_j^T (\mathbf{z} - \mathbf{m}_j)\end{aligned}$$

where $\mathbf{P}_r = \mathbf{I} - \mathrm{diag}(\boldsymbol{\rho}_r)\begin{bmatrix}\boldsymbol{\nu}_r, \dots, \boldsymbol{\nu}_r\end{bmatrix}^{\mathrm{T}}$. The prediction model (consisting of N_i local models with weights w_j defined by a Gaussian kernel with mean $\mathbf{m}_j$ and covariance $\mathbf{D}_j$) is

$$\begin{aligned} p_i(\mathbf{z}) &= \frac{1}{W}\sum_{j=1}^{N_i} w_j(\mathbf{z})\Psi_j(\mathbf{z}) \\ w_j(\mathbf{z}) &= \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{m}_j)^T\mathbf{D}_j(\mathbf{z}-\mathbf{m}_j)\right) \\ W &= \sum_{j=1}^{N_i} w_j(\mathbf{z}) \end{aligned}$$

Taking $\mathbf{z} = \begin{bmatrix}\mathbf{x}_k^{\mathrm{T}} & \mathbf{u}_k^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ and $\mathbf{y} = \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}^{\text{nom}}$, the prediction output $\hat{\mathbf{p}} = \begin{bmatrix}p_0, p_1, \dots\end{bmatrix}^{\mathrm{T}}$ gives the estimated perturbation to the system dynamics at a query point $\mathbf{z}$. The total predictive dynamics model is then given by

$$\begin{aligned}\bar{\mathbf{x}}_{k+1} &= \mathbf{x}_{k+1}^{\text{nom}} + \mathbf{y} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c} + \hat{\mathbf{p}} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \end{aligned} \tag{1}$$

Since LWPR learns the perturbation model online, it may initially return high-variance estimates when the system enters a new region of the input space (i.e., values of $\mathbf{z}$ for which the system has minimal experience). Therefore, to limit the effects of the resulting transients in the estimate, we introduce a simple gate based on the model uncertainty maintained by LWPR. If model uncertainty is high at a given query point, we instead use a zero-order hold on the previous estimate. As the system continues to gain experience in its operating domain, this gate will cease to be applied.

Finally, following the insight from L1 adaptive control [25], we introduce a low-pass filter on the disturbance estimate before it is incorporated into the predictive model (1). This enables LWPR to learn the perturbation model quickly while limiting changes to system dynamics to be within the bandwidth of the system.

B. Receding-Horizon Control Formulation

The use of an affine model (1) that automatically adapts to capture the effects of nonlinearities and unmodeled dynamics permits a simplified optimal control formulation for EPC relative to techniques such as nonlinear partial enumeration (NPE), which requires solving a nonlinear program due to the general nonlinear dynamics model. Taking the current state as the nominal state, $\mathbf{x}^* = \mathbf{x}_0$, and given N reference states $\mathbf{r}_1, \dots, \mathbf{r}_N$, let $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$. We can then formulate the receding-horizon control problem as a quadratic program:

$$\begin{aligned} \underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} &\frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}}\mathbf{Q}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) \\ &+ \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})^{\mathrm{T}}\mathbf{R}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\ & \mathbf{G}_{\mathbf{x}}\bar{\mathbf{x}}_{k+1} \le \mathbf{g}_{\mathbf{x}} \\ & \mathbf{G}_{\mathbf{u}}\bar{\mathbf{u}}_k \le \mathbf{g}_{\mathbf{u}} \\ & \forall\ k = 0, \dots, N-1 \end{aligned} \tag{2}$$

where we subtract $\bar{\mathbf{u}}_{\hat{\mathbf{p}}}$ (the control input corresponding to $\hat{\mathbf{p}}$) to avoid penalizing disturbance compensation.

To simplify notation, define $\boldsymbol{x} = \begin{bmatrix}\bar{\mathbf{x}}_1^{\mathrm{T}}, \dots, \bar{\mathbf{x}}_N^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{r} = \begin{bmatrix}\bar{\mathbf{r}}_1^{\mathrm{T}}, \dots, \bar{\mathbf{r}}_N^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{u} = \begin{bmatrix}\bar{\mathbf{u}}_0^{\mathrm{T}}, \dots, \bar{\mathbf{u}}_{N-1}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{u}_{\hat{\mathbf{p}}} = \begin{bmatrix}\bar{\mathbf{u}}_{\hat{\mathbf{p}}}^{\mathrm{T}}, \dots, \bar{\mathbf{u}}_{\hat{\mathbf{p}}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} \tilde{\mathbf{c}} \\ (\mathbf{A}+\mathbf{I})\,\tilde{\mathbf{c}} \\ \vdots \\ \sum_{i=0}^{N-1}\mathbf{A}^i\tilde{\mathbf{c}} \end{bmatrix},$$

$\boldsymbol{\mathcal{Q}} = \mathrm{diag}(\mathbf{Q}, \dots, \mathbf{Q})$, $\boldsymbol{\mathcal{R}} = \mathrm{diag}(\mathbf{R}, \dots, \mathbf{R})$, $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \mathrm{diag}(\mathbf{G}_{\mathbf{x}}, \dots, \mathbf{G}_{\mathbf{x}})$, $\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \mathrm{diag}(\mathbf{G}_{\mathbf{u}}, \dots, \mathbf{G}_{\mathbf{u}})$, $\boldsymbol{g}_{\mathbf{x}} = \begin{bmatrix}\mathbf{g}_{\mathbf{x}}^{\mathrm{T}}, \dots, \mathbf{g}_{\mathbf{x}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, and $\boldsymbol{g}_{\mathbf{u}} = \begin{bmatrix}\mathbf{g}_{\mathbf{u}}^{\mathrm{T}}, \dots, \mathbf{g}_{\mathbf{u}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$. Also, noting that $\bar{\mathbf{x}}_0 = \mathbf{0}$, we can rewrite (2) as

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} &\frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{x}-\boldsymbol{r}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}})^{\mathrm{T}}\boldsymbol{\mathcal{R}}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{\mathcal{B}}\boldsymbol{u} + \boldsymbol{c} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{x} \le \boldsymbol{g}_{\mathbf{x}} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{u}}\boldsymbol{u} \le \boldsymbol{g}_{\mathbf{u}} \end{aligned}$$

We can construct an equivalent QP entirely in terms of $\boldsymbol{u}$ by substituting the dynamics constraints and dropping constant terms in the cost function

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}}\ & \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{\Gamma}\boldsymbol{u} \le \boldsymbol{\gamma} \end{aligned} \tag{3}$$

where $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{c}-\boldsymbol{r}) - \boldsymbol{\mathcal{R}}\boldsymbol{u}_{\hat{\mathbf{p}}}$,

$$\boldsymbol{\Gamma} = \begin{bmatrix}\boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}}\end{bmatrix}, \text{ and } \boldsymbol{\gamma} = \begin{bmatrix}\boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}\end{bmatrix}$$

Defining $\boldsymbol{\lambda}$ as the vector of Lagrange multipliers and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$, the first two Karush-Kuhn-Tucker (KKT) conditions

for optimality (stationarity and complementary slackness) for the QP can then be written as

$$\begin{aligned} \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \mathit{\Gamma}^{\mathrm{T}}\boldsymbol{\lambda} &= \mathbf{0} \\ \boldsymbol{\Lambda}(\boldsymbol{\mathit{\Gamma}}\boldsymbol{u} - \boldsymbol{\gamma}) &= \mathbf{0} \end{aligned} \tag{4}$$

If we only consider the active constraints (i.e., with $\lambda > 0$) for a given solution, we can reconstruct $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ by solving a linear system derived from (4), where the subscript a indicates rows corresponding to the active constraints

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\mathit{\Gamma}}_a^{\mathrm{T}} \\ \boldsymbol{\mathit{\Gamma}}_a & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix}$$

Assuming the active constraints are linearly independent (Bemporad, et al. [1] suggest alternatives if this assumption fails), the resulting QP control law $\boldsymbol{u}$ is affine in the predicted state error $\boldsymbol{r}$ and parameterized by the system dynamics

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5\boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5\boldsymbol{c} - \boldsymbol{\mathcal{E}}_4\boldsymbol{\mathcal{R}}\boldsymbol{u}_{\tilde{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_3 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{5}$$

where $\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\mathit{\Gamma}}_a\boldsymbol{\mathcal{H}}^{-1}$, $\boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1\boldsymbol{\mathit{\Gamma}}_a^{\mathrm{T}})^{-1}$, $\boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^{\mathrm{T}}\boldsymbol{\mathcal{E}}_2$, $\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3\boldsymbol{\mathcal{E}}_1$, and $\boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4\boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}$. Moreover, since the coefficients in (5) are all functions of $\mathbf{A}$, $\mathbf{B}$, and $\tilde{\mathbf{c}}$, the overall control law $\boldsymbol{\kappa}(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N)$ can be written in terms of a parameterized feedback gain matrix $\mathbf{K}$ and feedforward vector $\mathbf{k}_{\text{ff}}$

$$\boldsymbol{\kappa}(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N) = \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})\boldsymbol{r} + \mathbf{k}_{\text{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \tag{6}$$

This parameterization also extends to the KKT condition checks to determine whether a previously computed controller is locally optimal. The active Lagrange multipliers $\boldsymbol{\lambda}_a$ follow a similar form to the control law

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6\boldsymbol{r} + \left(\boldsymbol{\mathcal{E}}_6\boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^{\mathrm{T}}\boldsymbol{\mathcal{R}}\boldsymbol{u}_{\tilde{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{7}$$

where $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^{\mathrm{T}}\boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}$.

Therefore, instead of storing the affine controller gains and Lagrange multipliers required to evaluate the KKT conditions, it is sufficient to store only the set of active constraints. The controller and KKT matrices can then be reconstructed online using (5), (7), and the current $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}$. Consequently, this parameterized formulation enables us to adapt and apply any previously computed controller, when appropriate according to the KKT conditions, even as the system dynamics evolve. The complete algorithm is detailed below.

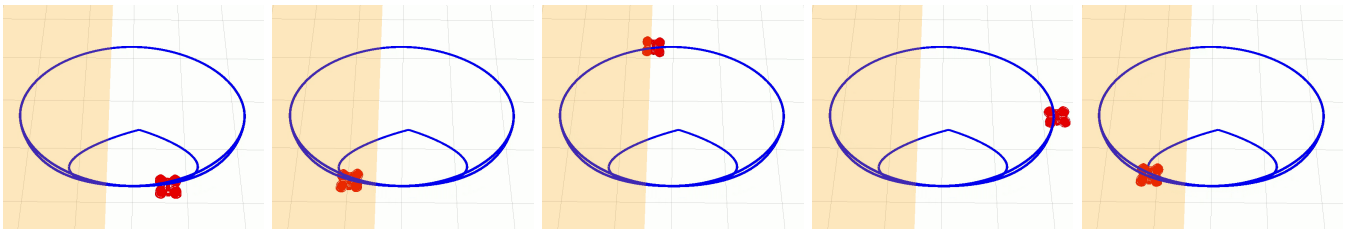
C. EPC Algorithm

As described in Alg. 1, EPC constructs a database defined as a mapping $\mathcal{M}$ from experiences to controllers. At the beginning of each control iteration, EPC queries the current state and reference, as well as the current linear model from LWPR, $(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})$. It then queries the parameterized mapping (line 6), and if the linear and nonlinear KKT conditions are met for an element, applies the corresponding controller. If no controller from prior experience is applicable (line 14), it solves the QP (3) to add a new parameterized element to the mapping, updating the stored experiences with the current scenario. In parallel, EPC applies commands from a short-horizon intermediate QP with slack on state constraints (line 15), in order to maintain a desired control update rate. As new controllers are added to the database, less valuable controllers (indicated by a lower `importance` score) can be removed (line 20) to bound the number of elements that may be queried in one control iteration.

Algorithm 1 Experience-driven Predictive Control

1: $\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
2: **while** control is enabled **do**
3: $\quad\boldsymbol{x} \leftarrow$ current system state
4: $\quad\boldsymbol{r} \leftarrow$ current reference sequence
5: $\quad\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}} \leftarrow$ current dynamics model from LWPR
6: $\quad$**for** each element $m_i \in \mathcal{M}$ **do**
7: $\quad\quad$Compute $\boldsymbol{u}, \boldsymbol{\lambda}$ via (5),(7)
8: $\quad\quad$**if** $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
9: $\quad\quad\quad$`importance`$_i \leftarrow$ current time, sort $\mathcal{M}$
10: $\quad\quad\quad$`solution_found` $\leftarrow$ true
11: $\quad\quad\quad$Apply affine control law (6) from m_i
12: $\quad\quad$**end if**
13: $\quad$**end for**
14: $\quad$**if** `solution_found` is false **then**
15: $\quad\quad$Apply interm. control via (3) with slack variables
16: $\quad\quad$Update QP formulation with current model
17: $\quad\quad$Solve QP (3) to generate new controller
18: $\quad\quad$**if** $|\mathcal{M}| >$ max table size **then**
19: $\quad\quad\quad$Remove element from $\mathcal{M}$ with minimum
20: $\quad\quad\quad$`importance`
21: $\quad\quad$**end if**
22: $\quad\quad$Add new element
23: $\quad\quad m_{\text{new}} = (\mathbf{x}_0, \mathbf{K}_1, \mathbf{K}_2, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\mathcal{M}$
24: $\quad$**end if**
25: **end while**

In addition to introducing adaptation to unmodeled dynamics, the parameterization by experience and the introduction of an online updated linear dynamics model eliminates the most computationally expensive component of NPE - the nonlinear program. Although the nonlinear program does not limit the control rate in NPE, it does limit how quickly new controllers can be computed, consequently limiting the practical horizon length and increasing the dependence on the intermediate controller. With its quadratic program formulation, EPC has the advantage of faster solution times in the parallel thread, which can be leveraged to reduce the dependence on the intermediate controller or increase the prediction horizon. Additionally, the nonlinear program solutions in NPE serve



(a) (b) (c) (d) (e)

Fig. 1: Snapshots of the quadrotor executing the elliptical trajectory that traverses the disturbance region (highlighted).

as fixed feedforward terms in the resulting affine control laws, precluding a completely adaptive control strategy. With EPC, the local controllers are fully parameterized, allowing controllers computed using past experience to be adapted to the present scenario.

III. Results

To validate the performance of the EPC algorithm, we conducted a series of simulations with a quadrotor micro-aerial vehicle tracking a trajectory that crosses a region where strong exogenous forces (e.g., wind) act on the vehicle.

The simulator and controller are built around ROS [19], and the controller uses the qpOASES library [8] to solve the quadratic programs. The simulation is run on a 2.9 GHz Intel mobile processor. We employ a hierarchical control setup [14], applying EPC separately to the translational and rotational dynamics. The quadrotor is commanded to fly ten laps at 0.7 m/s around an elliptical trajectory (Fig. 1) that intersects a region in which a constant disturbance torque is applied about the x and y axes. Since the disturbance acts on the rotational dynamics, we focus on the EPC used for attitude control in following results. Since attitude controllers are commonly run at rates exceeding 200 Hz [4], we note that a viable attitude controller should return a control input within 5 ms.

To demonstrate safety under limited control authority, we enforce constraints on the torque control inputs that are more restrictive than the nominal commands that would be applied to track the trajectory. As a result, these constraints are activated repeatedly as the vehicle tracks the trajectory. In order to satisfy these constraints, EPC learns 22 different parameterized feedback control laws, as shown in Fig. 2. Moreover, the intermediate controller (denoted controller 0) is only applied in the early laps, indicating that the majority of the controllers are learned quickly and then reused in subsequent laps. This intelligent controller switching also yields reliable constraint satisfaction, as shown in Fig. 3.

Over the course of this trial, the mean time required to query the controller database is 0.29 ms with a variance of 0.36 ms. This confirms that EPC is a computationally efficient approach for adaptive model predictive control suitable for high-rate applications, such as attitude control of a quadrotor.

In addition to constraint satisfaction, EPC substantially

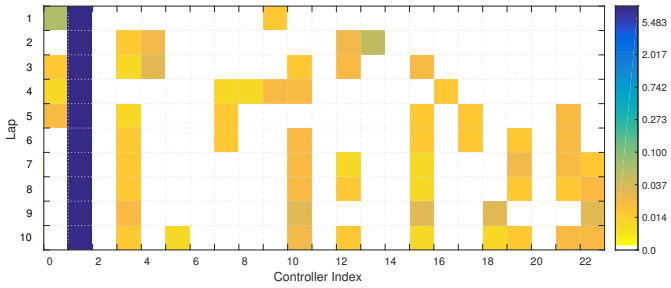


Fig. 2: Learned controllers are reused in subsequent laps, ultimately eliminating the dependence on the intermediate controller (column 0). Colors denote the total usage time (in seconds) for each controller.

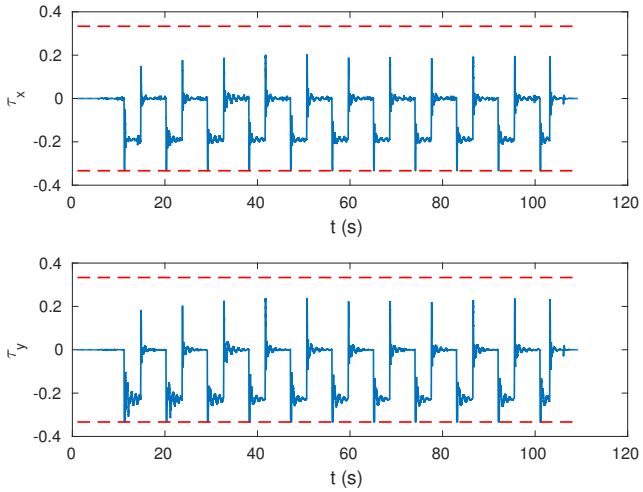


Fig. 3: EPC successfully satisfies roll and pitch control input constraints (dashed red lines) via controller switching

improves trajectory tracking accuracy in the presence of sudden changes to the system dynamics, as shown in Fig. 4. As expected, tracking performance improves over time as additional experience is gained. In addition to extending the controller database, this experience refines the LWPR model. Consequently, the model yields increasingly accurate estimates of the exogenous torques, as shown in Fig. 5.

Figure 6 illustrates the performance of EPC relative to

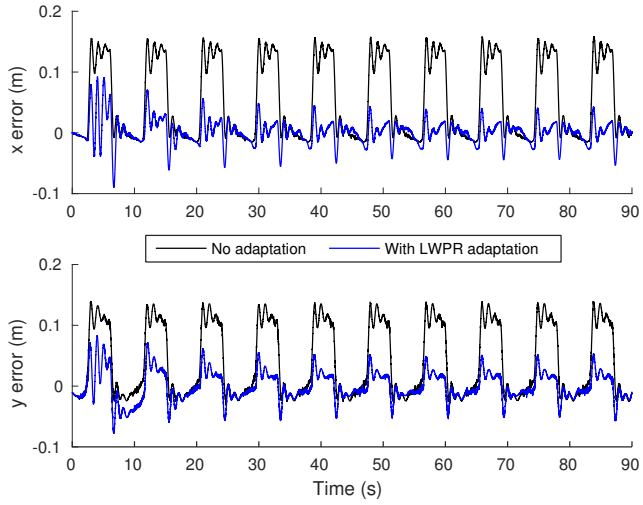


Fig. 4: Comparison of EPC with and without LWPR-based adaptation

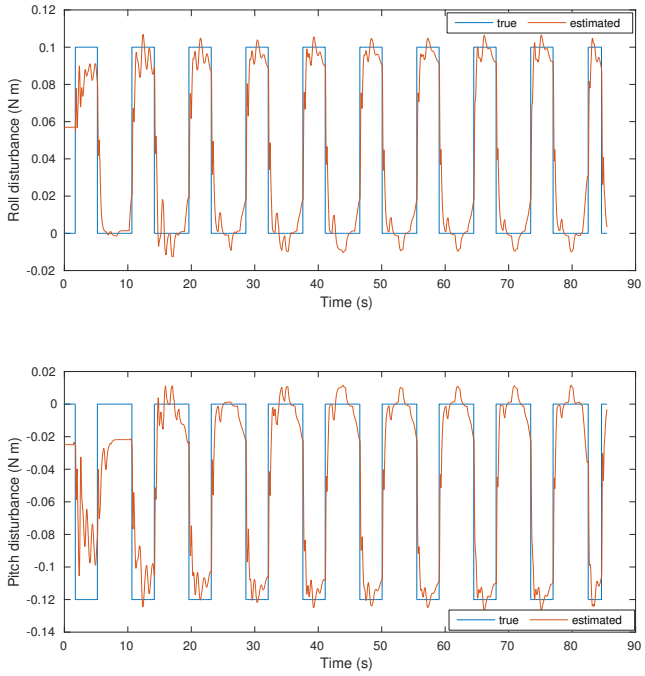


Fig. 5: LWPR accurately estimates the torque disturbances about the x- and y axes as it tracks the elliptical trajectory

two baseline approaches: L1 adaptive control (L1AC)[25] and an adaptive MPC formulation based on a state predictor (Luenberger observer). The gains for the L1AC were selected to match the nominal gains computed by EPC. The low-pass filter bandwidth was also set equal for both controllers to ensure a fair comparison of the adaptation laws. Since the core EPC formulation is equivalent to a quadratic program based MPC, we used EPC with the Luenberger observer as the second baseline. Additionally, we loosen the constraints on the control inputs applied in EPC for these simulations. EPC embeds the disturbance estimate in the prediction model

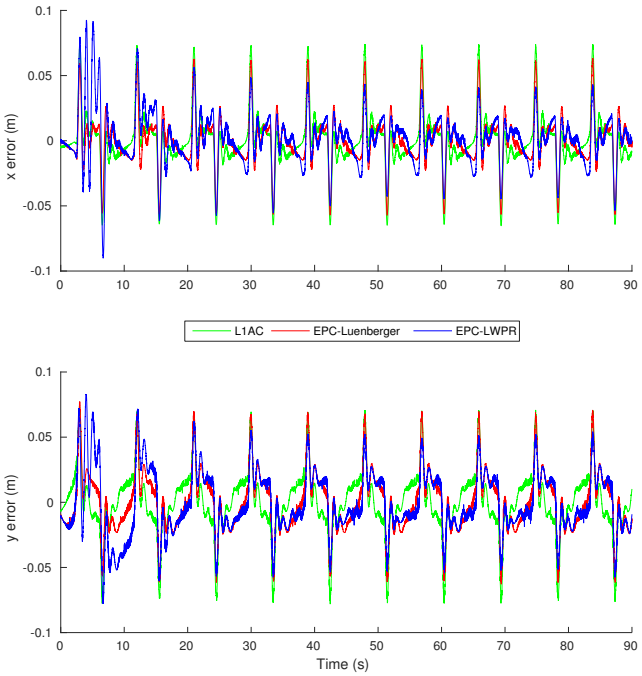


Fig. 6: EPC with LWPR yields improved position tracking error compared to L1 adaptive control (L1AC) and EPC with a simple state predictor (EPC-Luenberger)

to enable constraint satisfaction, whereas L1AC adds it as a compensation term to the resulting command. Therefore, it lacks any safe means of constraint satisfaction, precluding a comparison of constrained control performance.

As Fig. 6 shows, EPC (after obtaining sufficient experience) reduces peak tracking error by an average of 26.8% relative to L1 adaptive control. EPC (with LWPR) also reduces peak tracking error by an average of 17.2% relative to the variant with a Luenberger observer, confirming that the improvement relative to L1AC is not simply due to integrating the estimate into the prediction model. Moreover, these results show that the combination of a predictive controller driven by an online learned, reusable model can yield significantly improved tracking performance.

Finally, to evaluate the generalizability of experience, we consider a more complex scenario. Over the course of this 1000 s trial, the quadrotor is commanded to track a series of smooth but random trajectories through the same environment as before. Figures 7 and 8 show these trajectories, which achieve maximum commanded velocities of 1.7 m/s and accelerations of 5.1 m/s^2. The vehicle dynamics are also perturbed by a stochastic process emulating turbulent air flow, introducing noise into the LWPR training data.

Due to the randomization, the quadrotor enters and exits the disturbance region following a variety of trajectories. The resulting disturbance estimate (Fig. 9) shows transient behavior during the initial traversals of the disturbance region (e.g. during the first 200 s of the trial), with disturbance

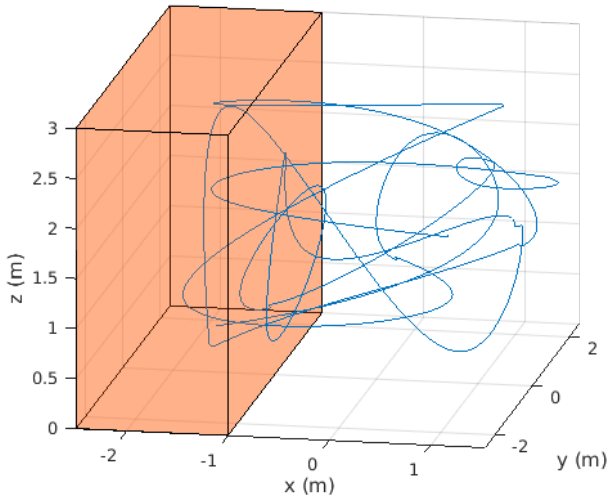


Fig. 7: Representative trajectories entering and exiting the disturbance regions, taken from a 100 s window of the randomized trial

estimate rise times greater than 1.5 s. However, these transients do not reappear, even as the vehicle traverses the region in previously unseen ways due to the variety of trajectories executed. Moreover, the disturbance estimate has a consistent rise time of approximately 0.5 s for the remainder of the trial. This indicates that the experience gained through the initial traversals is applicable to the numerous novel scenarios encountered in the future and yields a consistent improvement in disturbance estimation performance.

The controller also performs as expected. Even for this long trial with diverse trajectories, EPC only computes 52 controllers to maintain constraint satisfaction (see Fig. 10). Additionally, the time to query this database has a mean of 0.30 ms with a variance of 0.29 ms. This again illustrates the computational efficiency of this Experience-driven Predictive Control approach.

IV. Conclusions and Future Work

In this work, we have presented the Experience-driven Predictive Control (EPC) algorithm for fast, adaptive, nonlinear model predictive control. EPC constructs a database of reusable feedback controllers that are parameterized by the system dynamics. When combined with an online-learned model of the system dynamics based on Locally-Weighted Projection Regression (LWPR), this enables online adaption to perturbations to the dynamics model. As the system gains experience through operation, both the controller database and the dynamics model are improved to yield increased tracking accuracy, even in the presence of sudden changes in the dynamics model. This also implies that if the system were to start with some experience (e.g., from past operation), it could further reduce the transient effects of learning.

The simulation trials presented in this work provide a preliminary assessment of the EPC algorithm. We will therefore continue to evaluate the algorithm in simulation as well as pursuing experimental validation of the approach running onboard a small-scale quadrotor platform.

Acknowledgments

We gratefully acknowledge the support of ARL Grant W911NF-08-2-0004.

References

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

[2] P. Bouffard, A. Aswani, and C. Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, St. Paul, MN, May 2012.

[3] F. Debrouwere, M. Vukov, R. Quirynen, M. Diehl, and J. Swevers. Experimental validation of combined nonlinear optimal control and estimation of an overhead crane. In *Proc. of the Intl. Fed. of Autom. Control*, pages 9617–9622, Cape Town, South Africa, August 2014.

[4] V. Desaraju and N. Michael. Fast Nonlinear Model Predictive Control via Partial Enumeration. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016.

[5] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. G. Bock, E.-D. Gilles, and J. P. Schlöder. An Efficient Algorithm for Nonlinear Model Predictive Control of Large-Scale Systems Part I: Description of the Method. *Automatisierungstechnik*, 50(12):557–567, 2012.

[6] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones. Learning a Feasible and Stabilizing Explicit Model Predictive Control Law by Robust Optimization. In *Proc. of the IEEE Conf. on Decision and Control*, pages 513–519. IEEE, December 2011.

[7] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, May 2008.

[8] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[9] H. Fukushima, T.H. Kim, and T. Sugie. Adaptive model predictive control for a class of constrained linear systems based on the comparison model. *Automatica*, 43 (2):301–308, 2007.

[10] A. Grancharova and T.A. Johansen. *Explicit Nonlinear Model Predictive Control*, volume 429. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

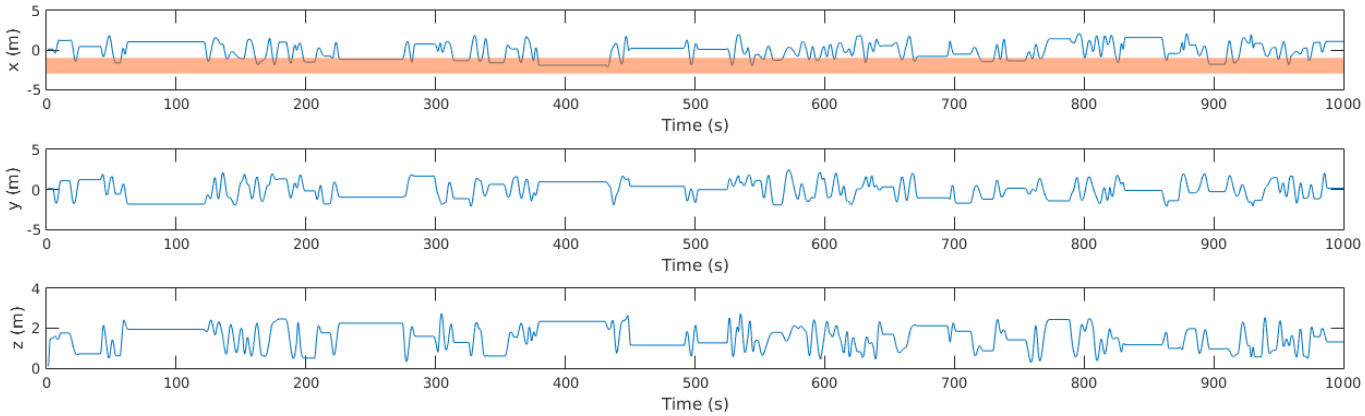


Fig. 8: Reference trajectory components for the randomized trial, with the disturbance region highlighted relative to the x-axis

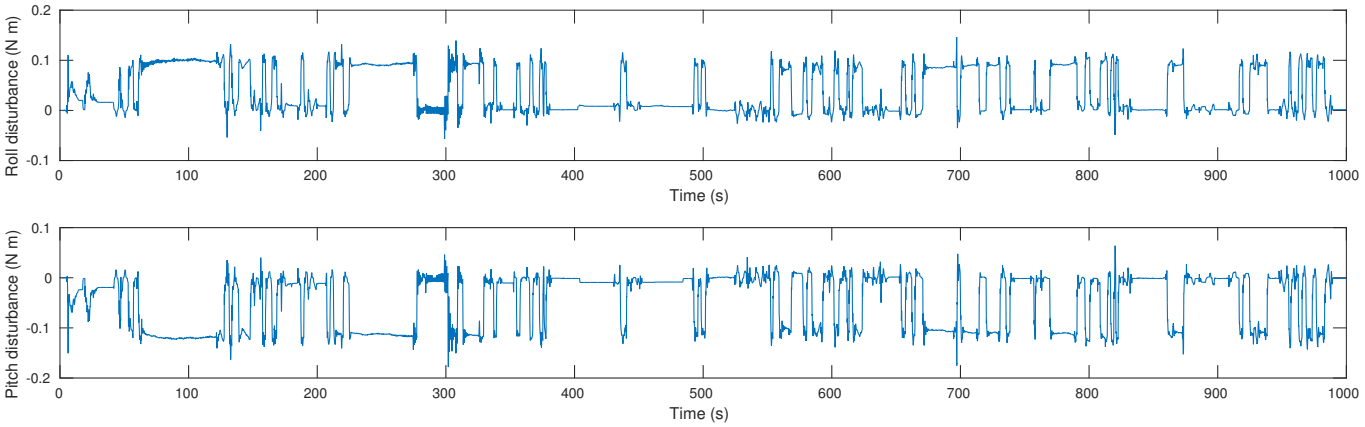


Fig. 9: Roll and pitch disturbance estimates for the randomized trial show an initial transient but have consistent performance for the remainder of the trial

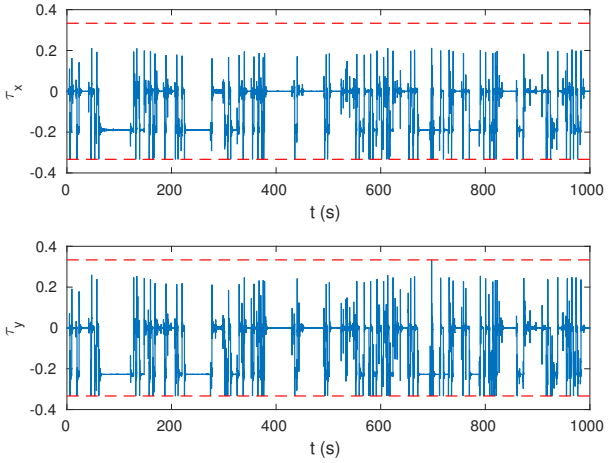


Fig. 10: EPC satisfies control input constraints for the entire duration of the randomized trial while tracking a diverse set of trajectories

[11] Y. K. Ho, H. K. Yeoh, and F. S. Mjalli. Generalized Predictive Control Algorithm with Real-Time Simultaneous Modeling and Tuning. *Industrial & Eng. Chem. Research*, 53(22):9411–9426, 2014.

[12] B. Houska, H. J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10): 2279–2285, 2011.

[13] Johan Löfberg. *Minimax approaches to robust model predictive control*. PhD thesis, Linköping University, 2003.

[14] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. Experimental evaluation of multirobot aerial control algorithms. *IEEE Robotics & Automation Magazine*, September 2010.

[15] D. Mitrovic, S. Klanke, and S. Vijayakumar. Adaptive optimal feedback control with learned internal dynamics models. *From Motor Learn. to Inter. Learn. in Rob.*, 264: 65–84, 2010.

[16] K. S. Narendra and A. M. Annaswamy. Prentice Hall.

[17] C. Ostafew, A. Schoellig, and T. Barfoot. Learning-Based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-Tracking in Challenging Outdoor Environments. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 4029–4036. IEEE, May 2014.

[18] G. Pannocchia, J. B. Rawlings, and S. J. Wright. Fast,

large-scale model predictive control by partial enumeration. *Automatica*, 43(5):852–860, 2007.

[19] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[20] S. Richter, C. N. Jones, and M. Morari. Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method. *IEEE Trans. Autom. Control*, 57(6):1391–1403, 2012.

[21] P. Ruvolo and E. Eaton. ELLA: An efficient lifelong learning algorithm. In *Intl. Conf. on Machine Learning*, pages 507–515, 2013.

[22] A. Saha, P. Rai, H. Daum e, and S. Venkatasubramanian. Online learning of multiple tasks and their relationships. In *Intl. Conf. on Artif. Intell. and Stat.*, pages 643–651, 2011.

[23] M. E. Taylor and P. Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *J. of Machine Learning Research*, 10:1633–1685, 2009.

[24] S. Vijayakumar, A. DSouza, and S. Schaal. Incremental Online Learning in High Dimensions. *Neural Comp.*, 17 (12):2602–2634, 2005.

[25] J. Wang, F. Holzapfel, E. Xargay, and N. Hovakimyan. Non-Cascaded Dynamic Inversion Design for Quadrotor Position Control with L1 Augmentation. In *Proc. of the CEAS Specialist Conf. on Guidance, Navigation & Control*, Delft, Netherlands, April 2013.

[26] Y. Wang and S. Boyd. Fast Model Predictive Control Using Online Optimization. *IEEE Trans. Control Syst. Technol.*, 18(2):267–278, March 2010.