2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

# Learning Adaptive Horizon Maps Based on Error Forecast for Model Predictive Control

Carlos Gonzalez[1], Seung Hyeon Bang[1], Po-han Li[2], Sandeep Chinchali[2], Luis Sentis[1]

***Abstract*— We present a model predictive control framework that uses varying prediction horizons according to the current forecasted uncertainties and estimated distance of the terminal state from its desired state. Our results suggest that the space of such optimal horizons, which we call horizon maps, is well structured for linear systems, meaning that it can be easily learned using tools from machine learning. Our approach is well suited for real-time control and can scale to higher dimensional systems. We also perform an analysis on the required quality of the datasets used to learn the horizon maps and conclude with results of this framework using an externally-driven, constrained linear quadratic regulator problem.**

## I. Introduction

Model Predictive Control (MPC) has become ubiquitous in the realm of robotics. It has been successfully employed in multiple real systems, from ground and aerial vehicles performing aggressive maneuvering tasks [1], [2], to legged robots performing agile locomotion [3], [4]. This is in part due to its ability to find actions based on a performance index that takes into account its predicted future states over a specified horizon. Using this knowledge and doing this iteratively, it is able to correct most deviations arising from model mismatches and uncertainties.

While its success has been prominent in real and complex systems, it is still bound to be efficient only on systems that meet certain conditions. For instance, when used in high-dimensional systems, it is often used under the assumption that the underlying system dynamics are approximately linear [5]. This assumption degrades rapidly as the system deviates from its linearized trajectory. In addition, errors in the modeled noise or in other external inputs to the system will cause the state prediction to worsen with prediction horizon. Since the horizon is often chosen based on heuristics, it can also limit the controller in being able to find a solution or might waste computational time by using excessively long horizons when they are not needed [6].

Finding a suitable horizon in real time can easily increase the computation time of the optimization problem as it can turn the problem into a mixed-integer program, or it can require solving the MPC several times. Instead, if the set of optimal horizons varies smoothly, we can learn this landscape offline and predict values online at no computational cost. In order to find such a suitable horizon, we define a performance metric that trades off proximity to the desired state at the end of the horizon and takes into account a forecast error model to penalize accruing costs with larger horizons.

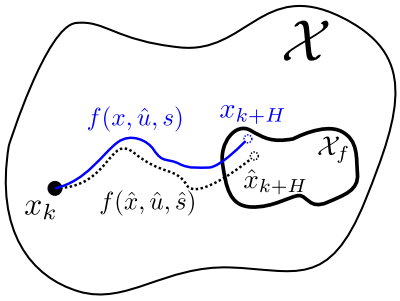


Fig. 1: **Forecast errors accrue with increasing horizons in MPC.** A controller plans its control actions $\hat{u}$ for the next $H$ steps based on a nominal model while using external forecasts $\hat{s}$. However, the real system will evolve according to the true model as it is perturbed by $s$, hence ending in a different state $x_{k+H}$. A proper horizon must be chosen to avoid accumulating errors while approaching its desired terminal state.

Several alternatives to the fixed-time MPC problem have been previously considered. A common alternative is to solve the standard MPC problem by keeping the number of time steps fixed but including the duration of the time step, e.g., $\Delta t$, in the optimization variable [7], [5], effectively changing the overall prediction horizon. However, this also changes the fidelity of the approximation of the continuous-time dynamics, especially as $\Delta t$ becomes large. Fast solutions to the free-time LQR problem have also been proposed using several methods, such as bilevel optimization [8], Differential Dynamic Programming [9], and move blocking [10]. However, these are derived explicitly using the classical cost function for variable-horizon MPC consisting of quadratic penalties in the state and control inputs, plus a linear penalty on the time to completion. A strategy similar in spirit to the previously mentioned ones is the Adaptive Horizon MPC (AHMPC) [11]. The approaches that resemble ours the most are [12], [13] and are indeed inspired by the idea of using AHMPC in combination with techniques from reinforcement learning and machine learning. These, however, differ from our formulation, in that we take into account a model of forecast errors, i.e., knowledge on how much the system is anticipated to diverge with planning time.

Given the current state of the art, our contributions are the following: (1) we derive an optimization formulation that finds an optimal horizon for MPC such that it leverages

[1]C. Gonzalez, S.H. Bang, and L. Sentis are with the Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, TX 78712, USA

[2]P. Li and S. Chinchali are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, TX 78712, USA

reaching a target state and deviating from the optimal cost due to forecast errors growing with the horizon, (2) we derive an algorithm to construct optimal horizon maps that can be used in a real-time manner with an MPC controller with a single MPC computation, and (3) we show that our MPC controller with learned horizons achieves comparable performance to a long, constant-horizon MPC controller while using an approach that scales to higher dimensional systems.

The organization of the remainder of this paper is described next. We formally define the problem we are interested in solving in Sec. II. In Sec. III, we detail our approach to solve this problem through the use of optimal horizon maps. Then, we present a set of numerical experiments and studies on the horizon maps. Lastly, we conclude with a short discussion and directions of future work in Sec. V.

## II. Problem Statement

Consider the receding horizon controller scenario depicted in Fig. 1. At the current state, $x_k$, the controller finds an optimal control input sequence such that it can reach a desired target set in the following $H$ steps while satisfying its state and control constraints. The predicted state evolution is based on a nominal system model, whose state at time $k+H$ is denoted by $\hat{x}_{k+H}$. However, due to external perturbations, $s$, the real dynamics of the system may differ from the modeled one, causing the real state to evolve differently from the anticipated state trajectory, ending at $x_{k+H}$, instead.

A discrete-time linear system that captures these modeling discrepancies is the input-driven linear dynamical system presented in [14],

$$x_{k+1} = Ax_k + Bu_k + Cs_k, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{n\times p}$ are time-invariant matrices that determine the next state of the system based on the current state $x_k \in \mathbb{R}^n$, control input $u_k \in \mathbb{R}^m$, and external timeseries $s_k \in \mathbb{R}^p$, at the current time step $k \in \mathbb{N}$. Note that the external input $s_k$ is unknown to the controller and is unaffected by the controller. In the best scenario, a forecast of this input, $\hat{s}_k$, can be modeled and provided to the controller to obtain better performance.

As in the classical LQR problem, we consider a quadratic cost performance index. Due to the time-invariant nature of the cost and dynamics, we re-write our running cost to start from $k = 0$. Hence, for a given horizon $H \in \mathbb{Z}^+$, the next control action is obtained from the following MPC problem:

$$\begin{aligned}
\underset{\hat{\mathbf{x}},\hat{\mathbf{u}}}{\text{minimize}} \quad & \sum_{k=0}^{H-1} \left(\hat{x}_k^\top Q_k \hat{x}_k + \hat{u}_k^\top R_k \hat{u}_k\right) + \hat{x}_H^\top Q_H \hat{x}_H && \text{(2a)}\\
\text{subject to} \quad & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k + C\hat{s}_k && \text{(2b)}\\
& \hat{x}_0 = x_0 && \text{(2c)}\\
& \hat{x}_k \in \mathcal{X}, \hat{u}_k \in \mathcal{U} && \text{(2d)}\\
& \hat{x}_H \in \mathcal{X}_f, && \text{(2e)}
\end{aligned}$$

where constraints (2b) and (2d) are enforced for all time steps $k \in \{0, \cdots, H-1\}$. The set of admissible and terminal states are denoted by $\mathcal{X}$ and $\mathcal{X}_f$, respectively, with $\mathcal{X}_f \subseteq \mathcal{X}$. Similarly, the set of admissible inputs is denoted by $\mathcal{U}$. Lastly, $Q \in \mathbb{S}_{\geq 0}$ and $R \in \mathbb{S}_{>0}$ are positive semi-definite and positive definite matrices penalizing state regulation and control effort, respectively. The forecasted state and control trajectories are contained in $\hat{\mathbf{x}} = [\hat{x}_0^\top, \cdots, \hat{x}_H^\top]^\top \in \mathbb{R}^{n(H+1)}$ and $\hat{\mathbf{u}} = [\hat{u}_0^\top, \cdots, \hat{u}_{H-1}^\top]^\top \in \mathbb{R}^{mH}$. We will use bold fonts to denote concatenated vectors or matrices, accordingly.

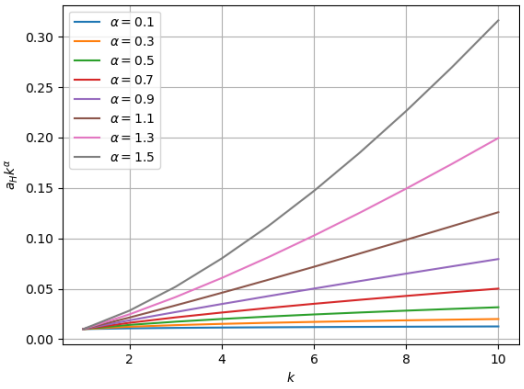


Fig. 2: **Model of exponential growth of errors with horizon.** Model (4) considers errors that can be bounded, linear, and also exponentially growing with horizon.

We wish to solve the MPC problem (2) using the horizon $H$ such that its terminal state is closest to our desired equilibrium point while penalizing the marginal difference in control cost to forecast errors. Thus, a suitable time horizon for the MPC problem is obtained by solving the following bilevel optimization problem:

$$\begin{aligned}
\underset{H}{\text{minimize}} \quad & \hat{x}_H^\top P \hat{x}_H + V_e(H; \hat{\mathbf{s}} - \mathbf{s})\\
\text{subject to} \quad & (\hat{\mathbf{x}}^*, \hat{\mathbf{u}}^*) \in S(H; x_0),
\end{aligned} \tag{3}$$

where $S(H; x_0)$ is the set of solutions to the horizon-parameterized MPC problem (2). $P \in \mathbb{S}_{>0}$ is a positive definite matrix penalizing deviations of the terminal state from the origin, and $V_e(H, \mathbf{s})$ is an error function that penalizes deviations from our true dynamics based on the horizon length $H$ and errors in the external inputs $\hat{\mathbf{s}}$ and $\mathbf{s}$.

The optimal horizon returned by problem (3) is tailored to a problem that uses additional information about anticipated forecast errors. This generalization encapsulates the special case of penalizing time to completion by using, e.g., a linear or exponential penalty on the horizon, but also allows for inclusion of other error models. Since this is a bilevel mixed integer program, it is hard to solve in real time simply using brute-force methods.

## III. Approach

In its general form, the bilevel optimization problem (3) can be difficult to solve analytically. In particular, the upper-level problem is an optimization problem over a positive

integer variable. The first cost term, $\hat{x}_H^\top P \hat{x}_H^\top$, depends on the horizon implicitly through the solution of the lower-level (MPC) problem, meaning that for any horizon $H$, a new terminal state, $\hat{x}_H$ will be obtained. Consequently, a new terminal cost will be accrued. On the other hand, the second cost term can be an explicit function of the horizon.

*Assumption 1:* The forecast error of the externally-driven input grows with time according to

$$\|\hat{s}_{0:k} - s_{0:k}\| \leq a_H k^\alpha \tag{4}$$

with error magnitude $a_H \in \mathbb{R}^+$ and exponential rate $\alpha \in \mathbb{R}^+$.

The forecast error assumption (4) is shown in Fig. 2 for multiple values of $\alpha$. Intuitively, this means that the external series forecast error is either bounded (if $\alpha < 1$), or grows unbounded with time (if $\alpha \geq 1$). This assumption is further validated using a real data set in Sec. IV. In addition, assumption (4) also equips the upper-level cost function with the property that, as the horizon increases, the first term will tend to decay as the system approaches the origin, and the second term will increase as our long-term state forecast becomes less reliable. We show in the Appendix that although this problem can be further simplified, it remains a computationally complex integer program. Our proposed algorithm results in a computationally faster approach since it results in a function evaluation, rather than solving the mixed integer problem. Hence, we opt to continue using the bilevel problem description.

To further motivate the choice for the cost landscape of problem (3), consider again the lower-level MPC problem (2). Following the derivations from [14], the analytical solution to the input-driven LQR problem is given by

$$\mathbf{u}^* = -\boldsymbol{K}^{-1}\boldsymbol{\kappa}(x_0, \mathbf{s}) \tag{5}$$

with

$$\boldsymbol{K} := \text{blockDiag}(R, H) + \sum_{k=0}^{H-1} \boldsymbol{M_k}^\top Q \boldsymbol{M_k},$$

$$\boldsymbol{\kappa}(x_0, \mathbf{s}) := \sum_{k=0}^{H-1} \boldsymbol{M_k}^\top Q (A^{k+1} x_0 + \boldsymbol{N_k}\mathbf{s}),$$

where $\text{blockDiag}(R, H)$ corresponds to a block diagonal matrix containing the matrix $R$, $H$ times along its diagonal. Furthermore, $\boldsymbol{M_k}$ and $\boldsymbol{N_k}$ correspond to the block matrices of the dynamics written as a function of the control and external input vectors, parameterized by the initial state, $x_0$:

$$x_{k+1} = A^{k+1} x_0 + \boldsymbol{M_k}\mathbf{u} + \boldsymbol{N_k}\mathbf{s}, \tag{6}$$

where $\boldsymbol{M_k} = [A^k B \;\; A^{k-1}B \;\; \cdots \;\; B \;\; \mathbf{0}] \in \mathbb{R}^{n \times mH}$ and $\boldsymbol{N_k} = [A^k C \;\; A^{k-1}C \;\; \cdots \;\; C \;\; \mathbf{0}] \in \mathbb{R}^{n \times pH}$. Let $J(\hat{\mathbf{u}}; x_0, \mathbf{s})$ be the total cost of the MPC problem, as given in (2a). Then, as shown in [14], the cost difference accrued from using a nominal model over the real model is quadratic in the forecast error according to

$$J(\hat{\mathbf{u}}; x_0, \hat{\mathbf{s}}) - J(\mathbf{u}^*; x_0, \mathbf{s}) = (\hat{\mathbf{s}} - \mathbf{s})^\top \boldsymbol{\Psi} (\hat{\mathbf{s}} - \mathbf{s}), \tag{7}$$

where $\boldsymbol{\Psi} := \boldsymbol{L}^\top \boldsymbol{K} \boldsymbol{L}$ and $\boldsymbol{L} := \sum_{k=0}^{H-1} \boldsymbol{M_k}^\top Q \boldsymbol{N_k}$.

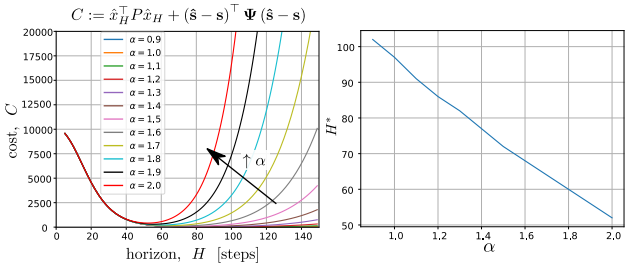


Fig. 3: **Optimal horizon for varying $\alpha$'s.** (Left) Optimal cost to problem (13) for multiple $\alpha$'s in the forecast prediction error (4). (Right) Optimal horizon for the corresponding $\alpha$, found numerically from the plot in the left.

Using assumption (4) and control cost (7), we can solve the bilevel optimization problem (3) by solving

$$\begin{aligned} \underset{H}{\text{minimize}} \quad & \|\hat{x}_H\|_P^2 + \|a_H(\text{range}(H, p))^\alpha\|_{\boldsymbol{\Psi}}^2 \\ \text{subject to} \quad & (\hat{\mathbf{x}}, \hat{\mathbf{u}}) \in S(H; x_0), \end{aligned} \tag{8}$$

where $\|\cdot\|_P$ denotes the $P$-weighted norm, e.g., $\|x\|_P^2 := x^\top P x$, and $\text{range}(H, p) := [0 \cdot \mathbf{1}^\top \;\; 1 \cdot \mathbf{1}^\top \;\; \cdots \;\; (H-1) \cdot \mathbf{1}^\top]^\top \in \mathbb{R}^{pH}$ and $(\cdot)^\alpha$ denotes the component-wise $\alpha$-exponent operator. The weighting matrix, $P$, of the norm in the first cost term of (8) can be obtained from the solution to the Discrete-time Algebraic Riccati Equation, i.e.,

$$P = \text{DARE}(A, B, Q, R). \tag{9}$$

To sum up, (8) is an integer programming problem with convex objective function. Hence, we later use a neural network to approximate its optimal solution, reducing the bilevel optimization problem (3) to problem (2) using learned optimal horizons. For derivation details, see the Appendix.

Similarly, the weight in the second cost term, $\boldsymbol{\Psi}$, is also determined from system matrices $(A, B, C)$ and performance matrices $(Q, R)$, as well as the current horizon $H$. Thus, the cost of (8) is completely determined from the problem at hand, given knowledge of the forecast errors.

For illustration purposes, consider the classic double integrator dynamics with forecast errors growing at a rate of $\|\hat{s}_{0:k} - s_{0:k}\| = 0.01k^\alpha$ for $0.9 \leq \alpha \leq 2.0$. The corresponding upper-level costs for multiple horizons, $H$, are shown in Fig. 3 for multiple values of $\alpha$. It can be seen that for small horizons, the cost is high even though the prediction errors are small. This is because the nominal state at the end of the horizon is expected to be far from the origin. As the horizon increases, the state is expected to approach the origin. However, prediction errors also start growing at different rates, according to the modeled error exponent $\alpha$. In particular, forecasted errors diverging faster (i.e., higher values of $\alpha$), result in higher costs accruing faster. This is also seen from Fig. 3, where the corresponding minimum horizons are shown for multiple $\alpha$'s. This shows explicitly that when the forecast error is expected to grow faster with time, the optimal horizon to choose is a smaller one, meaning we want to be more conservative as uncertainty about the future grows.

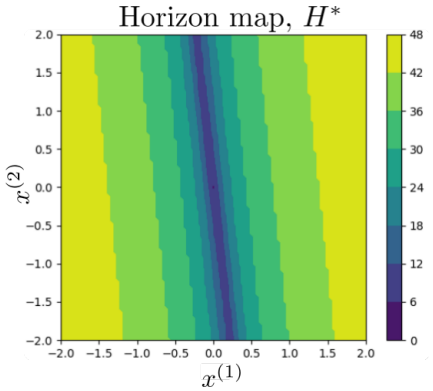


Fig. 4: **Optimal horizon map.** Color bar indicates $H^*$.

**Algorithm 1** Optimal horizon map for MPC (2D)

1: **Input:** Grid steps $\Delta_1$, $\Delta_2$ over $[x^{(1)}_{\min}, x^{(1)}_{\max}] \times [x^{(2)}_{\min}, x^{(2)}_{\max}] \in \mathcal{X}$, system matrices $(A, B, C)$, performance matrices $(Q, R)$, forecast error model parameters $a_H$ and $\alpha$, and maximum horizon $H_{\max}$
2: **Initialize:** $x_0 \leftarrow (x^{(1)}_{\min}, x^{(2)}_{\min})$
3: **for** $i = 1, \ldots, N_1$ **do**
4:  **for** $j = 1, \ldots, N_2$ **do**
5:  $\{J_h\}_{h=1}^{H_{\max}} \leftarrow \{0\}^{H_{\max}}$
6:  **for** $H = 1, \ \ldots, \ H_{\max}$ **do**
7:  $(\hat{\mathbf{x}}^*, \hat{\mathbf{u}}^*) \leftarrow$ solve problem (2)
8:  $J_H \leftarrow$ cost of problem (8) with $\hat{x}^*_H$ and $H$
9:  **end for**
10:  $H^*_{(i,j)} \leftarrow h$ where $\min \{J_h\}_{h=1}^{h=H_{\max}}$ is attained
11:  $x^{(2)}_0 \leftarrow x^{(2)}_{\min} + j \cdot \Delta_2$
12:  **end for**
13:  $x^{(1)}_0 \leftarrow x^{(1)}_{\min} + i \cdot \Delta_1$
14: **end for**
15: **Output:** Optimal horizon map, $H^*$

### A. Algorithm

Given the nature of the formulation and cost terms, we expect this behavior to generalize to higher dimensional systems of the same linear form. We next propose an algorithm to reduce the burden of the computation time in solving bilevel problem (3) in a real-time and closed-loop manner.

Recall that problem (3) is a mixed-integer programming problem since the optimization variable of the upper-level problem is an integer, while the optimization variables of the lower-level problem are continuous. Thus, this poses the biggest challenge in allowing us to reliably solve problem (3) fast and accurately. However, solving this problem for multiple points in a subset of the feasible state space suggests that the set of optimal horizons might vary smoothly in the vicinity of a given point. For the double integrator example, the optimal horizon map over the subset $[-2, 2] \times [-2, 2]$ is shown in Fig. 4. It can be seen that the optimal horizons decrease monotonically as the state approaches the origin and, in fact, get smaller towards the origin.

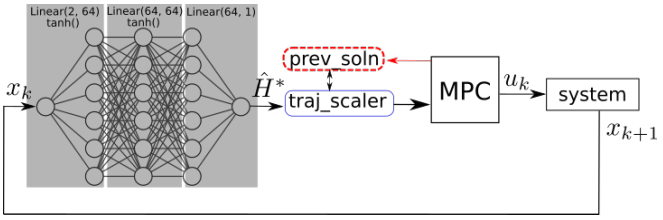


Fig. 5: **Combined neural network and MPC controller scheme.** At each time step, the current state $x_k$ is passed to the NN, which in turn outputs the estimated optimal horizon: $\hat{H}^*$. The module `traj_scaler` adjusts the previous solution of the MPC according to the size of the new horizon. This is then used to warm start the the solver at the current iteration, for which $\hat{H}^*$ is used.

To this end, we leverage offline MPC simulations to construct a discrete optimal horizon map. We then aim to learn a general form of this map using a Neural Network (NN), which also allows us to obtain a pseudo-optimal horizon in a real-time loop without qualitatively sacrificing optimality. We first explain the procedure to construct the optimal horizon map following Algorithm 1 assuming $n = 2$, however, this readily extends to systems with $n > 2$. Besides the system and performance matrices $(A, B, C)$ and $(Q, R)$, and the external input error forecast model parameters $a_H$ and $\alpha$ from (4), we need a grid map for which the map is constructed. For ease of clarity, assume a map covering $[x^{(1)}_{\min}, x^{(1)}_{\max}] \times [x^{(2)}_{\min}, x^{(2)}_{\max}]$, where the superscript indicates the element in the state variable, i.e., $(x^{(1)}, x^{(2)}) \in \mathbb{R}^2$. Denote the distance between two points along the first direction by $\Delta_1$ and along the second direction by $\Delta_2$. Then, we start our algorithm from one corner of the grid map, as indicated in line 2. We then start the iteration process along the $N_1$ grid points along the first direction and along the $N_2$ grid points along the second direction. In line 5, we reset the sequence of cost for the current initial state, $x_0$, to zero. Then, we populate this sequence of costs for multiple horizons by first solving the MPC problem (2), then recording the upper-level cost incurred by this optimal solution, as shown in lines 6-9. This results in costs such as the ones shown in Fig. 3, for a fixed $\alpha$. After iterating through all horizons, we store the horizon resulting in the minimum cost in the optimal horizon map, $H^*_{(i,j)}$, where the sub-indices indicate the corresponding coordinates in the grid. In line 11, we simply move "up" on the grid and use this new value as our next initial state. Similarly, in line 13, we move to the "right". This algorithm outputs the optimal horizon map shown in Fig. 4.

We wish to obtain a pseudo-optimal horizon, $\hat{H}^*$, even for states $x$ that do not belong to the grid created through Algorithm 1. Instead of interpolating in between optimal horizons around the nearest grid point, we treat the horizon map as a multiclass classification problem. Thus, we train an NN to learn a mapping from the current state to the

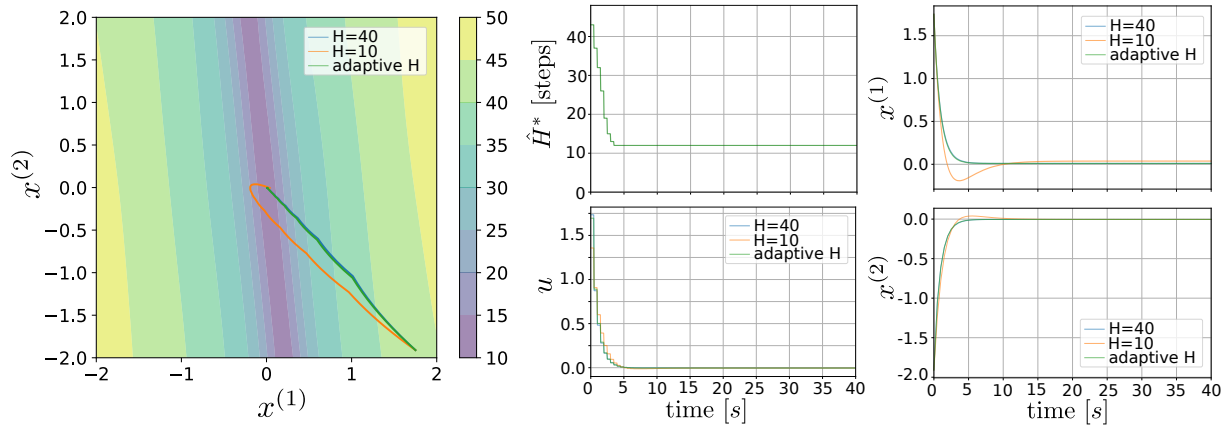


Fig. 6: **Adapting the horizon allows the controller to use long horizons only when needed without degrading performance.** The MPC with the learned horizons (green) will have performance closer to a longer constant horizon (blue) one while being able to smoothly transition to lower horizons when needed. Choosing a short horizon (orange) from the beginning will result in poor performance.

corresponding estimated optimal horizon,

$$\hat{H}^* = \phi(x_0; \boldsymbol{\theta}) \tag{10}$$

where $\boldsymbol{\theta}$ denotes the NN model parameters and $\phi : \mathbb{R}^n \longmapsto \mathbb{N}$ represents a regressor function. This approach allows us to scale this same algorithm to higher dimensional systems.

Lastly, since the horizon of the MPC can vary from one control loop to the next, the size of the state and control optimization variables will also change. We wish to reuse our solution from time $k$ to warm-start the solver at time $k+1$. To address this, we create a module (`traj_scaler`) that "stretches" or "compresses" the MPC solution from the previous time step (`prev_soln`) by scaling the trajectories and linearly interpolating in between points. The proposed combined architecture is shown in Fig. 5.

## IV. Numerical Experiments

We now present our varying-horizon MPC controller and compare its performance against the constant horizon case. We then compare the optimal horizon map in three scenarios, namely when $\alpha < 1$, when $\alpha = 1$, and when $\alpha > 1$, and motivate scenarios where these modeling parameters are practical assumptions. We then perform a comparative analysis on the learned horizon map to determine the impact of the grid refinement at the time of training. In these studies, we consider the externally driven double integrator with external input matrix $C = [0.1, \ -0.3]^\top$, cost matrices $Q_k = \text{diag}(100, 100)$ and $R_k = 50 \ \forall \ k \in [0, H-1]$, terminal cost matrix $Q_H = P$ as given by (9), and $a_H = 0.01$.

### A. Performance Against Constant-Horizon MPC

Simulations of the input-driven double integrator are shown in Fig. 6. The initial state is selected randomly within the trained space of $[-2, 2] \times [-2, 2]$. An external series perturbs the state at a rate of $\alpha = 1.0$. The controller starts off with a horizon of $H = 42$ and as it traverses the state space while approaching the origin, it chooses a shorter and shorter horizon, until it settles at $H = 12$ steps. The gain in computation time is clearly shown in Fig. 9, where we show the computation times for the constant horizons $H = 40$ and $H = 10$, along with the adaptive horizon case. The performance of the controller is not compromised, and evidently, the optimization problem reduces drastically in size even in this simple example as the optimization variable $\hat{\mathbf{u}}$ reduces with the horizon, $H$, from 42 to 12 within the first few seconds. In Fig. 6, the short constant horizon of $H = 10$ is chosen for comparison as the simulation steps in between MPC updates for all cases is chosen to be of 10 time steps.

### B. Optimal Horizons for Different Forecast Error Models

We first look at the effect of different error forecast models in the optimal horizon map to reason how these models impact the controller's horizon choice. To this end, we let the control space be $\mathcal{U} = \mathbb{R}$ and perform our offline construction of the horizon map according to Algorithm 1.

The resulting optimal horizon maps for the forecast error model (4) with $\alpha \in \{0.8, 1.0, 1.2\}$ are shown in Fig. 7. The area where the horizon is large, e.g., where $H > 35$ gets smaller with higher $\alpha$-exponents, meaning that due to higher uncertainties at higher horizons, the controller chooses to plan over shorter horizons.

Next we discuss some practical examples representative of the model (4) for varying $\alpha$-exponents. The case $\alpha < 1$ corresponds to a bounded error with increasing horizon where, in addition, small errors result over small horizons. This is common in applications such as networked control, where predictions of future events need to be made in advance and, thus rely on timeseries models. Consider the Uber data set containing pick up locations in NYC [15]. Using a state-of-the-art timeseries forecaster, TSAI [16], we train a forecast model $\hat{\mathbf{s}}$ for the aforementioned system using multiple modeled horizons, similar to how it is done in [17]. It can be shown that the errors in such model have small deviations over short horizons but increase with the horizon, yet become bounded, as seen in Fig. 10. The case of $\alpha = 1$ corresponds to a linear penalty with horizon. This is practical

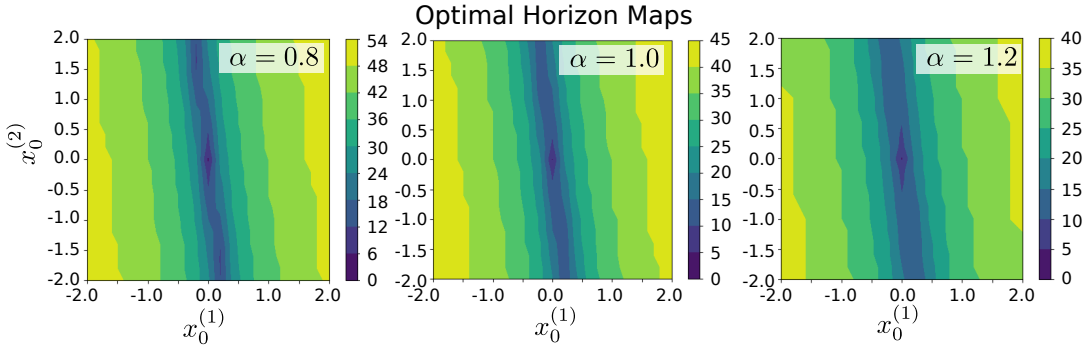


Fig. 7: **Optimal horizon decreases with increasing forecast error.** The horizon maps quantitatively capture that if the real dynamics are expected to deviate faster from the estimated dynamics (higher value of $\alpha$ in (4)), shorter plans are better.

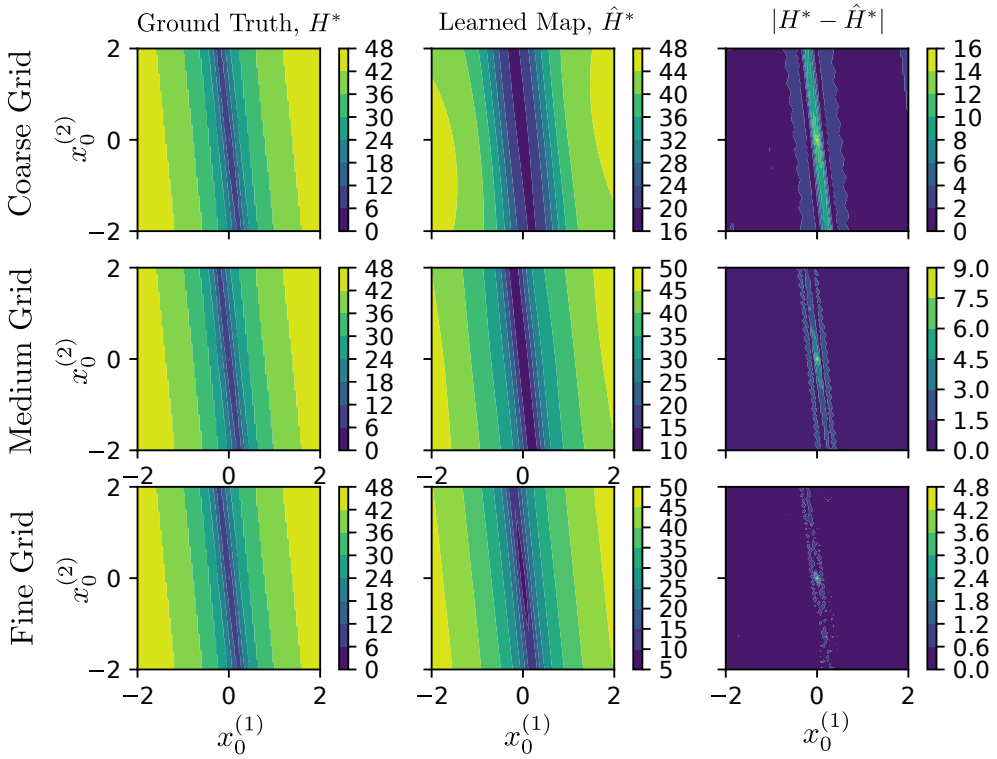


Fig. 8: **Errors in learned horizon maps mostly present in small horizons.** Effect of training dataset with different grid discretizations for the double integrator problem. The left column represents the ground-truth dataset $\mathcal{D}$ with $101 \times 101$ samples, while the middle column shows the output of the NNs with (top row) coarse dataset ($21 \times 21$ samples), (middle row) the medium size dataset ($51 \times 51$ samples), and (bottom row) the fine dataset ($101 \times 101$ samples). The right column shows the prediction errors for the corresponding NNs.

in scenarios where it is desired to penalize long computation times and has been similarly used in previous works [9]. The last case, corresponding to $\alpha > 1$ can be reminiscent of situations where a linear model is assumed to control a nonlinear system.

### C. Horizon Map Grid Size Used in Learned Map

To analyze the effect of different grid discretizations of $\mathcal{X}$ when pre-training the network that predicts the optimal horizon of MPC, we created three different datasets obtained by using: a coarse grid, a medium-sized grid, and a fine grid. These are equidistantly sampled to have $21 \times 21$, $51 \times 51$, and $101 \times 101$ data points in $\mathcal{X} = [-2, 2] \times [-2, 2]$. These are clustered along with their corresponding optimal MPC horizon $H^*$ into a collection of training datasets, $\mathcal{D} = \{\mathcal{D}^i\}_{i=1}^{n_t}$, where

$$\mathcal{D}^i \triangleq \{x_0^i, (H^*)^i\} \tag{11}$$

and $n_t$ denotes the number of data points, e.g., $441$ for the coarse, $2,601$ for the medium, and $10,201$ for the fine grids.

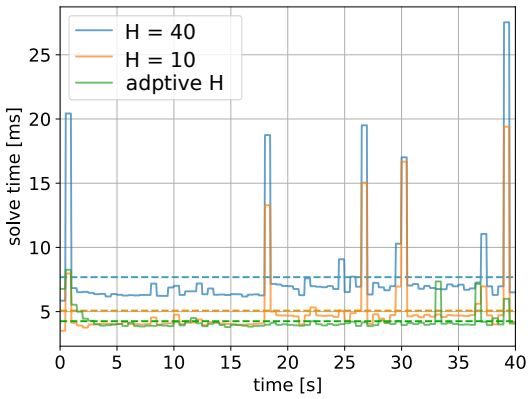


Fig. 9: **Adapting the horizon speeds up the computation time.** The computation time of the adaptive horizon controller decreases with the horizon $H$, resulting in a safer real-time execution. The dashed lines indicate average values.

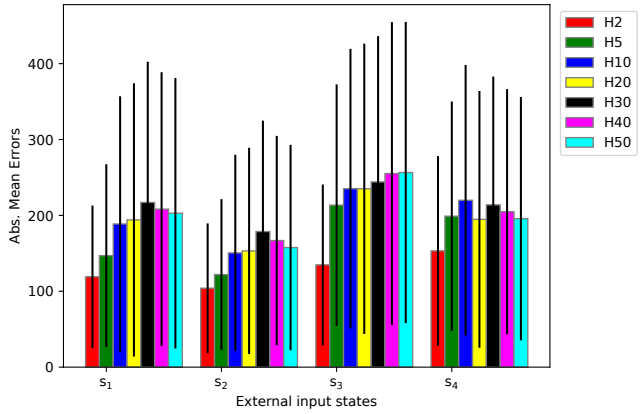


Fig. 10: **Example of real data motivating applicability of our forecast error model assumption.** Errors in forecasted taxi demand using Uber dataset [15] trained using [16] showing that both the mean forecast errors follow the proposed exponential model bounded by (4).

We then trained three NN models (one per grid size) to approximate $H^*$ and denote the learned horizon $\hat{H}^*$. Each NN is composed of a 3-layer neural network where the hidden layers consist of 64 neurons with a $\tanh$ activation function. The outputs of the trained NNs were then compared with the ground truth dataset $\mathcal{D}$ by numerically analyzing their errors of $H^*$, as shown in Fig. 8. As expected, the NN trained with more data points (finer grid discretization) shows the best approximation to the optimal horizon $H^*$, but it is noteworthy that the NN trained with smaller data points (medium and coarse grid discretization) also predicts fairly accurately in the state space $\mathcal{X}$.

## V. Conclusions and Discussion

In this work, we have introduced a framework to find and use optimal horizons for an MPC setting by considering a model of the forecast errors and the modeled linear system dynamics and LQR performance matrices $Q$ and $R$. We provided an algorithm that generalizes the sampled space of optimal horizons to higher dimensions by training an NN to approximate and generalize this map. We have shown the performance of our approach used online in a linear system with externally driven inputs, showing how performance is preserved while relying on varying (and often shorter) horizons. We present a study on how these horizon maps change with varying forecast error models and the effect of the dimension of the training dataset in the learned horizon map.

In this study, we focused on accuracy rather than speed while generating the training data set, hence have left as future work exploring data efficient methods in learning the optimal horizon map. Similarly, we plan to additional use cases in applications related to locomotion and navigation of complex systems.

## Appendix

We consider here a simplified formulation of the bilevel optimization problem (3) and show that even this case remains a challenging problem to solve analytically. Consider timeseries prediction errors satisfying inequality (4). We seek to find the final horizon $H$ that penalizes both the marginal difference in control cost to forecast errors and deviations of the predicted terminal state from its desired terminal state. The former one can be expressed as a penalty on the terminal state, while the latter can be expressed as a penalty on the control cost sensitivity due to forecast errors. In other words, this can be expressed by

$$J = \hat{x}_H^\top P \hat{x}_H + (\mathbf{s} - \hat{\mathbf{s}})^\top \mathbf{\Psi} (\mathbf{s} - \hat{\mathbf{s}}) \tag{12}$$

with $\mathbf{\Psi}$ as defined in [18] and $\mathbf{s} := [s_1^\top, s_2^\top, \cdots, s_H^\top]^\top \in \mathbb{R}^{pH}$. The bilevel optimization problem is:

$$\begin{aligned} \underset{H}{\text{minimize}} \quad & \hat{x}_H^\top P \hat{x}_H + (\mathbf{s} - \hat{\mathbf{s}})^\top \mathbf{\Psi} (\mathbf{s} - \hat{\mathbf{s}}) \\ \text{subject to} \quad & (\hat{x}_k, \hat{u}_k) \in \mathcal{P}_1, \end{aligned} \tag{13}$$

where $\mathcal{P}_1$ is the solution to the low-level MPC problem

$$\begin{aligned} \underset{\hat{\mathbf{x}}, \hat{\mathbf{u}}}{\text{minimize}} \quad & \sum_{k=0}^{H-1} \left( \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k \right) + \hat{x}_H^\top Q \hat{x}_H \\ \text{subject to} \quad & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k + C\hat{s}_k, \end{aligned} \tag{14}$$

with the constraint being satisfied for all $k \in \{0, \cdots, H-1\}$. Problem $\mathcal{P}_1$ constitutes a simplification of problem (2), which can be formulated, for instance, by implementing the hard constraints as weighted quadratic soft constraints. Following [14], the forecasted linear dynamics can be written as a function of the initial state, the control input vector, and the forecasted external series, i.e.,

$$\hat{x}_{k+1} = A^{k+1} x_0 + \boldsymbol{M_k} \hat{\mathbf{u}} + \boldsymbol{N_k} \hat{\mathbf{s}} \tag{15}$$

with $\boldsymbol{M_k} = [A^k B \;\; A^{k-1} B \;\; \cdots \;\; B]$, $\boldsymbol{N_k} = [A^k C \;\; A^{k-1} C \;\; \cdots \;\; C]$, $\hat{\mathbf{u}} = [\hat{u}_0 \; \hat{u}_1 \; \cdots \; \hat{u}_{k-1}]$, and $\hat{\mathbf{s}} = [\hat{s}_0 \; \hat{s}_1 \; \cdots \; \hat{s}_{k-1}]$. With this, we can write the solution

to (2) as $\hat{\mathbf{u}}^\star = -\boldsymbol{K}^{-1}\boldsymbol{\kappa}(x_0, \hat{\mathbf{s}})$, with $\boldsymbol{K}$ and $\boldsymbol{\kappa}(x_0, \hat{s})$ as defined in [14]. Substituting $\hat{\mathbf{u}}^\star$ into the dynamics, we get

$$\hat{x}_H = A^H x_0 + \boldsymbol{M_k}\hat{\mathbf{u}}^\star + \boldsymbol{N_k}\hat{\mathbf{s}} \tag{16}$$

$$= A^H x_0 - \boldsymbol{M_{H-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa}(x_0, \hat{\mathbf{s}}) + \boldsymbol{N_{H-1}} + \hat{\mathbf{s}} \tag{17}$$

$$= \left(A^{H-1} - \boldsymbol{M_{H-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa_1}\right) x_0 + \left(\boldsymbol{N_{H-1}} - \boldsymbol{M_{H-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa_2}\right) \hat{\mathbf{s}}, \tag{18}$$

where we have re-written

$$\boldsymbol{\kappa}(x_0, \mathbf{s}) = \underbrace{\sum_{k=0}^{H-1} \boldsymbol{M_k}^\top Q A^{k+1}}_{\boldsymbol{\kappa_1}} x_0 + \underbrace{\sum_{k=0}^{H-1} \boldsymbol{M_k}^\top Q \boldsymbol{N_k}}_{\boldsymbol{\kappa_2}} \hat{\mathbf{s}}. \tag{19}$$

Similarly, the marginal difference in control cost can be written using (4) as

$$(\mathbf{s} - \hat{\mathbf{s}})^\top \Psi (\mathbf{s} - \hat{\mathbf{s}}) = a_H^2 \| \left(\mathrm{range}(H, p)\right)^\alpha \|_\Psi^2, \tag{20}$$

where $\mathrm{range}(H, p) \in \mathbb{R}^{pH}$ is a vector composed of vectors of size $p$ of integers starting from 0 and increasing up to $H$. Thus, the bilevel problem (13) reduces to

$$\begin{aligned} \underset{H}{\text{minimize}} \quad & \|\hat{x}_H\|_P^2 + a_H^2 \| \left(\mathrm{range}(H, p)\right)^\alpha \|_\Psi^2 \\ \text{subject to} \quad & \hat{x}_H = f(H; x_0, \hat{\mathbf{s}}), \end{aligned} \tag{21}$$

with

$$f(H; x_0, \hat{\mathbf{s}}) = \left(A^{H-1} - \boldsymbol{M_{N-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa_1}\right) x_0 + \left(\boldsymbol{N_{H-1}} - \boldsymbol{M_{H-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa_2}\right) \hat{\mathbf{s}}$$

which, in turn, reduces to the unconstrained optimization problem

$$\begin{aligned} \underset{H}{\text{minimize}} \quad & \| \left(A^{H-1} - \boldsymbol{M_{H-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa_1}\right) x_0 \\ & + \left(\boldsymbol{N_{H-1}} - \boldsymbol{M_{H-1}}\boldsymbol{K}^{-1}\boldsymbol{\kappa_2}\right) \hat{\mathbf{s}} \|_P^2 \\ & + a_H^2 \| \left(\mathrm{range}(H, p)\right)^\alpha \|_\Psi^2. \end{aligned} \tag{22}$$

Let us analyze the convexity of (22). The first term is weighted by $\boldsymbol{\Psi}$, which is positive semi-definite. The terms inside the $P$-weighted norm contain the optimization variable $H$ in either the matrix exponent (e.g., $A^{H-1}$, or the number of times $A$ is multiplied) or in the matrix dimensions (e.g., $\boldsymbol{M_{H-1}} \in \mathbb{R}^{n \times mH}$). The second term is weighted by $P$, which is positive definite (since we chose $P$ according to (9)), and the function in the norm is concave for $0 \leq \alpha < 1$ and convex for $\alpha \geq 1$ as seen from Fig. 2. Thus, the unconstrained cost is the sum of two convex functions for $\alpha \geq 1$. When $0 < \alpha < 1$, the $\Psi$-weighted norm monotonically approaches a constant, thus resulting in a function that decays to zero and a function that is always positive and approaches a constant, leading to a quasi-convex function.

## REFERENCES


[1] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear MPC for quadrotors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 690–697, 2022.

[2] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear MPC in real-time," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 2015-Febru, no. February. IEEE, 2014, pp. 2505–2510.

[3] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *IEEE International Conference on Intelligent Robots and Systems*, 2018, pp. 7440–7447.

[4] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining Learning-Based Locomotion Policy With Model-Based Manipulation for Legged Mobile Manipulators," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2377–2384, 2022.

[5] J. Li and Q. Nguyen, "Dynamic Walking of Bipedal Robots on Uneven Stepping Stones via Adaptive-Frequency MPC," *IEEE Control Systems Letters*, vol. 7, pp. 1279–1284, 2023.

[6] J. Lee, M. Seo, A. Bylard, R. Sun, and L. Sentis, "Real-Time Model Predictive Control for Industrial Manipulators with Singularity-Tolerant Hierarchical Task Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[7] J. v. d. Berg, "Extended LQR: Locally-Optimal Feedback Control for Systems with Non-Linear Dynamics and Non-Quadratic Cost," in *Springer Tracts in Advanced Robotics*, vol. 114, 2016, pp. 39–56.

[8] A. De Marchi and M. Gerdts, "Free finite horizon LQR: A bilevel perspective and its application to model predictive control," *Automatica*, vol. 100, pp. 299–311, 2019. [Online]. Available: https://doi.org/10.1016/j.automatica.2018.11.032

[9] K. Stachowicz and E. A. Theodorou, "Optimal-Horizon Model-Predictive Control with Differential Dynamic Programming," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 1440–1446.

[10] R. C. Shekhar and J. M. Maciejowski, "Robust variable horizon MPC with move blocking," *Systems and Control Letters*, vol. 61, no. 4, pp. 587–594, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.sysconle.2012.02.004

[11] A. J. Krener, "Adaptive Horizon Model Predictive Control," in *IFAC-PapersOnLine*, vol. 51, no. 13. Elsevier B.V., 2018, pp. 31–36. [Online]. Available: https://doi.org/10.1016/j.ifacol.2018.07.250

[12] E. Bøhn, S. Gros, S. Moe, and T. A. Johansen, "Reinforcement learning of the prediction horizon in model predictive control," in *IFAC-PapersOnLine*, vol. 54, no. 6. Elsevier Ltd, 2021, pp. 314–320. [Online]. Available: https://doi.org/10.1016/j.ifacol.2021.08.563

[13] M. S. M. Gardezi and A. Hasan, "Machine Learning Based Adaptive Prediction Horizon in Finite Control Set Model Predictive Control," *IEEE Access*, vol. 6, pp. 32 392–32 400, 2018.

[14] J. Cheng, M. Pavone, S. Katti, S. Chinchali, and A. Tang, "Data Sharing and Compression for Cooperative Networked Control," *Advances in Neural Information Processing Systems*, vol. 8, no. NeurIPS, pp. 5947–5958, 2021.

[15] "Uber pick ups in New York City." [Online]. Available: https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city

[16] "tsai: state-of-the-art deep learning for time series and sequences." [Online]. Available: https://github.com/timeseriesAI/tsai

[17] P.-h. Li, S. P. Chinchali, and U. Topcu, "Differentially Private Timeseries Forecasts for Networked Control," in *2023 American Control Conference (ACC)*. American Automatic Control Council, 2023, pp. 3595–3601. [Online]. Available: http://arxiv.org/abs/2210.00358

[18] P.-h. Li, U. Topcu, and S. P. Chinchali, "Adversarial Examples for Model-Based Control: A Sensitivity Analysis," in *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2022, pp. 1–7.