Proceedings of Machine Learning Research vol 242:399–413, 2024 6th Annual Conference on Learning for Dynamics and Control

MPC-Inspired Reinforcement Learning for Verifiable Model-Free Control

Yiwen Lu LUYW20@MAILS.TSINGHUA.EDU.CN
Zishuo Li LIZS19@MAILS.TSINGHUA.EDU.CN
Yihan Zhou ZHOUYH23@MAILS.TSINGHUA.EDU.CN
Na Li NALI@SEAS.HARVARD.EDU
Yilin Mo* YLMO@TSINGHUA.EDU.CN
Department of Automation and BNRist, Tsinghua University

Editors: A. Abate, M. Cannon, K. Margellos, A. Papachristodoulou

Abstract

In this paper, we introduce a new class of parameterized controllers, drawing inspiration from Model Predictive Control (MPC). The controller resembles a Quadratic Programming (QP) solver of a linear MPC problem, with the parameters of the controller being trained via Deep Reinforcement Learning (DRL) rather than derived from system models. This approach addresses the limitations of common controllers with Multi-Layer Perceptron (MLP) or other general neural network architecture used in DRL, in terms of verifiability and performance guarantees, and the learned controllers possess verifiable properties like persistent feasibility and asymptotic stability akin to MPC. On the other hand, numerical examples illustrate that the proposed controller empirically matches MPC and MLP controllers in terms of control performance and has superior robustness against modeling uncertainty and noises. Furthermore, the proposed controller is significantly more computationally efficient compared to MPC and requires fewer parameters to learn than MLP controllers. Real-world experiments on vehicle drift maneuvering task demonstrate the potential of these controllers for robotics and other demanding control tasks.

1. Introduction

Recent years have witnessed the development of Deep Reinforcement Learning (DRL) for control (Lillicrap et al., 2015; Duan et al., 2016; Haarnoja et al., 2018), with the locomotion of agile robots (Xie et al., 2018; Li et al., 2021; Margolis et al., 2022; Rudin et al., 2022) being a notable example. Many such applications use Multi-Layer Perceptron (MLP) as the entirety or a part of the control policy, which, despite their remarkable empirical performance, face limitations in terms of explainability (Agogino et al., 2019) and performance guarantees (Osinenko et al., 2022). Research efforts have been devoted to the stability verification of MLP controllers (Dai et al., 2021; Zhou et al., 2022), structured controller parameterizations (Srouji et al., 2018; Johannink et al., 2019; Sattar and Oymak, 2020; Ni et al., 2021), or a combination of both (Zinage and Bakolas, 2023), and the learning of explainable and verifiable controllers has remained an active topic.

On the other hand, Model Predictive Control (MPC) has been a prevalent choice for high performance controller, and its stability and safety has been well-studied (Morari and Lee, 1999; Qin and Badgwell, 2003; Schwenzer et al., 2021). Recently, there is a growing interest in augmenting MPC with learning, a large portion of which are focused on addressing the challenges in designing critical components of MPC like prediction models (Desaraju and Michael, 2016; Soloperto et al., 2018; Hewing et al., 2019), terminal costs and constraints (Brunner et al., 2015; Rosolia and Borrelli, 2017; Abdufattokhov et al., 2021), stage costs (Englert et al., 2017; Menner et al., 2019), or a combination of these components (Gros and Zanon, 2019); readers are referred to Hewing et al. (2020)

for a comprehensive overview. Most of the methods follow a model-based framework – using data to estimate a model and then perform optimal control in a receding horizon fashion. These methods still suffer various challenges, such as requiring intensive computation at each time horizon, and making myopic decisions that lead to infeasibility or inefficiency in the long run.

Motivated by the advantages of DRL and MPC, we propose an MPC-inspired yet model-free controller. Leveraging the fact that linear MPC solves a Quadratic Programming (QP) problem at each time step, we consider a parameterized class of controllers with QP structure similar to MPC. However, the key distinction lies in the approach to obtaining the QP problem parameters: instead of deriving them from a model, they are optimized via DRL. This approach ensures that the resulting controllers not only have theoretical guarantees akin to MPC, thanks to its QP structure, but also demonstrate competitive performance and computational efficiency when empirically compared to MPC and MLP controllers.

Contrasting with works from the learning community, such as Amos et al. (2018), which uses MPC as a module in a larger policy network, and Ha and Schmidhuber (2018); Hansen et al. (2023); LeCun (2022), which adapt MPC ideas by planning in latent spaces, our work retains the QP structure of linear MPC. While these learning-based approaches aim to be general and tackle more challenging tasks by integrating MPC concepts into comprehensive models, they often include black-box components without control-theoretic guarantees. In contrast, our approach specializes in control tasks, emphasizing performance guarantees and computational efficiency. However, empirical evidence on a real-world robotic system demonstrates that our controller may generalize beyond simple linear systems, as illustrated in an aggressive vehicle control setting.

Our Contribution. In this paper, we propose a new parameterized class of MPC-inspired controllers. Specifically, our controller resembles an unrolled QP solver, structured similarly to a Recurrent Neural Network (RNN), with its parameters learned rather than derived via a predictive model. To train the parameters of the controller, most of the existing DRL methods, such as PPO (Schulman et al., 2017), could be used. However, in contrast to most DRL-trained controllers, which often lack rigorous theoretical guarantees, our MPC-inspired controller is proven to enjoy verifiable properties like persistent feasibility and asymptotic stability. We also compare the proposed controller on benchmark tasks with other methods such as classical MPC and DRL-trained neural network controllers, showing that our proposed controller enjoys lighter computation and increased robustness. Lastly, though we only provide theoretical guarantees for controlling a linear system, the generalizability of the proposed constroller is empirically demonstrated via vehicle drift maneuvering, a challenging nonlinear robotics control task, indicating potential applications of our controller to real-world nonlinear robotic systems.

2. Problem Formulation and Preliminaries

Notations. Subscripts denote the time index, e.g., x_k stands for the system state at step k, and $x_{0:k}$ means the sequence $x_0, x_1, \ldots, x_k$. Superscripts denote the iteration index in an iterative algorithm, e.g. y^i stands for the variable y at the i-th iteration. Bracketed subscripts denote slicing operation on a vector, e.g., $v_{[1]}$ denotes the first element of the vector v, and $v_{[1:n]}$ denotes its first n elements. The set of positive definite $n \times n$ matrices is denoted as $\mathbb{S}^n_{++}$, and the nonnegative orthant of $\mathbb{R}^n$ is denoted as $\mathbb{R}^n_+$. The Kronecker product of two matrices A and B is denoted as $A \otimes B$. The block diagonal matrix with diagonal blocks $A_1, \ldots, A_n$ is denoted as $\mathrm{diag}(A_1, \ldots, A_n)$. The projection operator to a convex set C is denoted as $\Pi_C(\cdot)$.

2.1. Problem Formulation

In this paper, we consider the discrete-time infinite-horizon constrained linear-quadratic optimal control problem, formulated as follows:

Problem 1 (Infinite-horizon constrained linear-quadratic optimal control)

$$\underset{u_{0:\infty}}{\textit{minimize}} \quad \limsup_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} (x_{k+1}-r)^\top Q (x_{k+1}-r) + u_k^\top R u_k, \tag{1a}$$

$$\textit{subject to} \quad x_{k+1} = Ax_k + Bu_k, \tag{1b}$$

$$u_{\min} \le u_k \le u_{\max}, x_{\min} \le x_{k+1} \le x_{\max}, \tag{1c}$$

where $x_k \in \mathbb{R}^{n_{sys}}$ are state vectors, $u_k \in \mathbb{R}^{m_{sys}}$ are control input vectors, $r \in \mathbb{R}^{n_{sys}}$ is the reference signal, $A \in \mathbb{R}^{n_{sys}\times n_{sys}}$ and $B \in \mathbb{R}^{n_{sys}\times m_{sys}}$ are the system and input matrices, $Q \in \mathbb{S}_{++}^{n_{sys}}$ and $R \in \mathbb{S}_{++}^{m_{sys}}$ are the stage cost matrices, and $u_{\min}, u_{\max} \in \mathbb{R}^{m_{sys}}$ and $x_{\min}, x_{\max} \in \mathbb{R}^{n_{sys}}$ are bounds on control input and state respectively. It is assumed without loss of generality that (A, B) is controllable.

2.2. Linear MPC and its QP Representation

Problem 1 is typically computationally intractable due to infinite planning horizons and constraints. A commonly adopted approximation is to truncate it to finite horizon N, and solve the problem formulated in Problem 2 at each time step, with x_0 being the current state. The first control input u_0^* from the optimal solution is applied to the system in a receding horizon fashion.

Problem 2 (Linear MPC)

$$\underset{x_{1:N}, u_{0:N-1}}{\textit{minimize}} \quad \sum_{k=0}^{N-1} (x_{k+1}-r)^\top Q (x_{k+1}-r) + u_k^\top R u_k, \tag{2a}$$

$$\textit{subject to} \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \cdots, N-1, \tag{2b}$$

$$u_{\min} \le u_k \le u_{\max}, x_{\min} \le x_{k+1} \le x_{\max}, \quad k = 0, \cdots, N-1. \tag{2c}$$

The above MPC problem can be cast into a Quadratic Programming (QP) problem in the following standard form[1]:

Problem 3 (Standard-form QP)

$$\underset{y}{\textit{minimize}} \quad \frac{1}{2} y^\top P y + q^\top y, \quad \textit{subject to} \quad Hy + b \ge 0, \tag{3}$$

where $y \in \mathbb{R}^{n_{qp}}$, $P \in \mathbb{S}_{++}^{n_{qp}}$, $q \in \mathbb{R}^{n_{qp}}$, $H \in \mathbb{R}^{m_{qp}\times n_{qp}}$, and $b \in \mathbb{R}^{m_{qp}}$.

The translation from Problem 2 to Problem 3 can be performed by using the control sequence $y = \begin{bmatrix} u_0^\top & \cdots & u_{N-1}^\top \end{bmatrix}^\top$ as the decision variable, and eliminating the equality constraints (2b) by representing the trajectory $x_{1:N}$ using y. The resulting QP problem size and parameters are:

$$n_{qp} = Nm_{sys}, \quad m_{qp} = 2N(m_{sys} + n_{sys}), \tag{4}$$

$$P = 2(\boldsymbol{B}^\top \boldsymbol{Q} \boldsymbol{B} + \boldsymbol{R}), \quad q = \boldsymbol{B}^\top \boldsymbol{Q}(\boldsymbol{A} x_0 - \boldsymbol{r}), \quad H = -\boldsymbol{C}\boldsymbol{B} - \boldsymbol{D}, \quad b = \boldsymbol{e} - \boldsymbol{C}\boldsymbol{A} x_0, \tag{5}$$

1. Although a linear MPC without terminal costs or constraints is presented here for simplicity, one can derive a similar QP formulation for linear MPC with quadratic terminal cost and affine terminal constraint.

where

$$\boldsymbol{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} B & & & \\ AB & B & & \\ \vdots & & \ddots & \\ A^{N-1}B & \cdots & & B \end{bmatrix}, \quad \begin{array}{l} \boldsymbol{C} = I_N \otimes [I_{n_{sys}} \ \ -I_{n_{sys}} \ \ \mathbf{0}_{n_{sys} \times 2m_{sys}}]^\top, \\ \boldsymbol{D} = I_N \otimes [\mathbf{0}_{m_{sys} \times 2n_{sys}} \ I_{m_{sys}} \ \ -I_{m_{sys}}]^\top, \\ \boldsymbol{e} = I_N \otimes [x_{\max}^\top \ \ -x_{\min}^\top \ u_{\max}^\top \ \ -u_{\min}^\top]^\top, \\ \boldsymbol{r} = I_N \otimes r, \boldsymbol{Q} = I_N \otimes Q, \boldsymbol{R} = I_N \otimes R. \end{array} \tag{6}$$

2.3. Algorithm for Solving QPs

A family of efficient methods for solving QPs is operator splitting algorithms (Ryu and Yin, 2022), which are adopted by existing solvers such as OSQP (Stellato et al., 2020). An iteration of an operator splitting algorithm for solving QPs can generally be represented as a combination of affine transformations and projections on the variable. For example, we derive a variant of the Primal-Dual Hybrid Gradient (PDHG) (Chambolle and Pock, 2011) algorithm, whose iteration can be expressed in the succinct form shown as follows: [2]

$$z^{i+1} = \Pi_{\mathbb{R}_+^{m_{qp}}} \left((I - 2\alpha F) z^i + \alpha (I - 2F) \lambda^i) - 2\alpha\mu \right), \quad \lambda^{i+1} = F(z^i + \lambda^i) + \mu, \tag{7}$$

where $z = Hy + b \in \mathbb{R}^{m_{qp}}$ is the primal variable of an equivalent form of the original problem (3), $\lambda \in \mathbb{R}^{m_{qp}}$ is a dual variable introduced by the same equivalent form, $\alpha > 0$ is the step size, and the parameters in the iteration are:

$$F = (I + HP^{-1}H^\top)^{-1}, \quad \mu = F(HP^{-1}q - b). \tag{8}$$

Once one obtains an approximate solution z^i, the original variable can be recovered from the equality-constrained QP problem $y^i \in \arg\min\{\frac{1}{2}y^\top Py + q^\top y \mid Hy + b = z^i\}$, where y^i can be explicitly represented as:

$$y^i = -P^{-1}q + P^{-1}H^\top (HP^{-1}H^\top)^\dagger (z^i - b + HP^{-1}q). \tag{9}$$

Theorem 1 *If $0 < \alpha < 1$ and the problem (3) is feasible, then the iterations (7) yields $y^i \to y^*$, where y^i is in (9) and y^* is the optimal solution of the original problem. Furthermore, the suboptimality gap satisfies:*

$$p^i - p^\star \leq \|\lambda^i\|_2 \|r^i_{prim}\|_2 + \|y^i - y^\star\|_2 \|r^i_{dual}\|_2, \tag{10}$$

where p^i, p^* are the primal value at iteration i and the optimal primal value respectively, and r^i_{prim}, r^i_{dual} are the primal and dual residuals defined as follows:

$$r^i_{prim} = Hy^i + b - z^i, r^i_{dual} = Py^i + q - H^\top \lambda^i. \tag{11}$$

Proof Results similar to Theorem 1 has been derived in (Chambolle and Pock, 2011)(Boyd et al., 2011). Therefore, the full proof is presented in the extended version of the paper (Lu et al., 2023a, Appendix A) due to space limit. ■

The iteration (7) on the primal-dual variable pair (z, λ) can be implemented by interleaving an affine transformation whose parameters (F, μ) depend on the problem parameters (P, q, H, b), and a projection of the z-part onto the positive orthant, which is equivalent to ReLU activation in

2. The form (7) is slightly different from the original PDHG iteration (Ryu and Yin, 2022, p. 75) in that the primal and dual variables are updated synchronously, making it conceptually more straightforward to draw similarity between solver iterations and neural networks; see (Lu et al., 2023a, Appendix A) for the details.

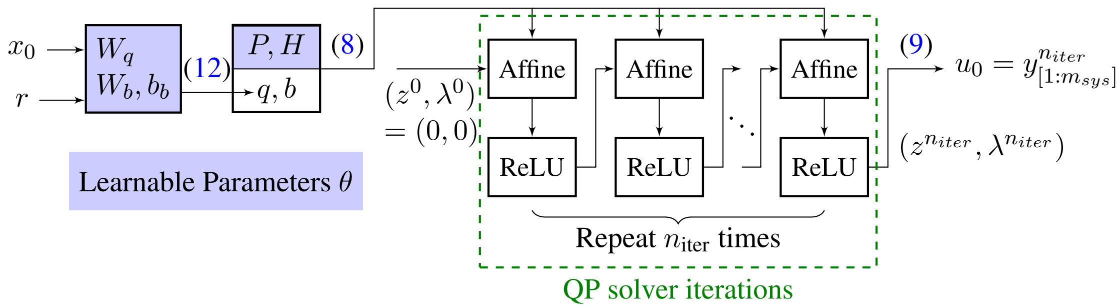


Figure 1: Proposed control policy architecture. The controller solves a QP problem in form (3), whose parameters P, H are shared across all initial state and reference (x_0, r), while q, b depend affinely on (x_0, r) with weights W_q, W_b and bias b_b (see (12)). An approximate solution to the QP problem, $y^{n_{iter}}$, is obtained by running a n_{iter} QP solver iterations (7) followed by a transform (9), whose first m_{sys} dimensions are used as the current control input u_0.

neural networks. Therefore, the sequence of iterations for solving a QP problem resembles a single-layer Recurrent Neural Network (RNN) with weights dependent on the QP parameters (P, q, H, b), followed by ReLU activation. This resemblance facilitates the end-to-end policy gradient-based reinforcement learning of the QP parameters, as is discussed in Section 3.

3. Learning Model-Free QP Controllers

This section introduces a reinforcement learning framework to tune the parameters of QP problem (3), instead of deriving them using the predictive models via conventional MPC. This could be beneficial to combat the short-sightedness (Erez et al., 2012) and the lack of robustness (Forbes et al., 2015), which may occur in MPC.

The policy architecture facilitating this learning process is shown in Figure 1, where the control policy, i.e., the mapping from x_0 and r to the control action u_0, is represented as a fixed number of PDHG iteration for solving a QP problem, or equivalently, an unrolled RNN, the parameters of which can then be trained using most of the existing policy-gradient based or actor-critic DRL method.

Here we highlight several critical design components of the policy architecture. The first two are regarding the parameterization of the policy, i.e., *what* to learn:

• **State-independent matrices** P, H**:** Note from (5) that for the MPC controller, the matrices P, H holds the same across different initial and reference state (x_0, r)'s. Motivated by this fact, the matrices P, H are state-independent in the proposed policy architecture, i.e., only one matrix P and one matrix H need to be learned for a specific system. Additionally, to ensure the positive definiteness of P, we use the factor L_P in the Cholesky decomposition $P = L_P L_P^\top$ instead of the matrix P itself as the learnable parameter, and force its diagonal elements to be positive via a softplus activation (Zheng et al., 2015), a commonly applied trick for learning positive definite matrices (Haarnoja et al., 2016; Lutter et al., 2019).

• **Affine transformations yielding vectors** q, b**:** Continuing with the inspirations from MPC (5), we restrict the vectors q to depend linearly on the current state x_0 and the reference state r, and the vector b to depend affinely on x_0:

$$q(x_0, r; W_q) = W_q[x_0^\top \ r^\top]^\top, \quad b(x_0, r; W_b, b_b) = W_b x_0 + b_b, \tag{12}$$

where W_q, W_b (resp. b_b) are learnable matrices (resp. vector) of proper dimensions.

The above-described parameterization strategy ensures that when the chosen problem dimensions n_{qp}, m_{qp} match the dimensions of the QP translated from MPC, then the MPC policy is within the family of parameterized policies defined by the proposed architecture. In other words, the proposed controller can be viewed as a generalization of MPC. While the state-independence and state-affineness constraints in the parameterization limit the range of policies relative to those in MLP or MPC-like policies with generic function approximator components (Amos et al., 2018) , this reduced complexity is beneficial for deriving provable theoretical guarantees, as is discussed in Section 4. On the other hand, both numerical examples and real-world experiments show that the empirical performance of the proposed controller matches that of an MPC or MLP controller, and thus is not hindered by this restriction.

Another two design components determine *how* the parameters are learned:

• **Unrolling with a fixed number of iterations:** To solve the QP problem and differentiate the solution with respect to the problem parameters, we deploy a fixed number n_{iter} of QP solver iterations described in Section 2.3, and differentiate through the computational path of these iterations, a practice known as unrolling (Monga et al., 2021). Unlike implicit differentiation methods (Amos and Kolter, 2017; Amos et al., 2018; Agrawal et al., 2019), which differentiate through the optimality condition and hence requires the forward pass of the solver to reach the stationary point, our method directly differentiates the solution after n_{iter} iterations, and can obtain a correct gradient even if the stationary point is not reached within these iterations. According to our empirical results, a small number of iterations would suffice for good control performance (e.g., $n_{iter} = 10$), which mitigates the computational burden of the unrolling process. Intuitively, the sufficiency of a small n_{iter} can be accredited to the model-free nature of the proposed method, which, by discarding the restrictions imposed by model-based prediction (see (6)), gains the flexibility to learn a QP problem that not only optimizes the controller performance, but also is easy to solve.

• **Reinforcement learning with residual minimization:** The control policy described above, parameterized by $\theta = (L_P, H, W_q, W_b, b_b)$, can serve as a drop-in replacement for standard policy networks, and be optimized using various off-the-shelf policy-based or actor-critic RL algorithms, such as PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018) and DDPG (Lillicrap et al., 2015). However, apart from the standard RL loss, we also include a regularization term for minimizing the residuals given by the QP solver embedded in the policy. Given a dataset $\mathcal{D}$ of transition samples, it is defined as follows:

$$\ell_{res}(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \|Hy_k^{n_{iter}} + b_k - z_k^{n_{iter}}\|_2^2 + \|Py_k^{n_{iter}} + q_k - H^\top \lambda_k^{n_{iter}}\|_2^2, \tag{13}$$

which, motivated by the result stated in Theorem 1 that small residuals are indicative of near-optimality, encourages the learned QP problems to be easy to solve. From above, the procedure of policy learning using an RL algorithm is shown in Algorithm 1.

As an additional note, the learned QP problem parameterized by (P, H, q, b) can be enforced to be feasible, which facilitates the theoretical analysis of the learned controller. Readers are referred to (Lu et al., 2023a, Appendix B) for the details.

4. Performance Guarantees of Learned QP Controller

In this section, we propose a method for establishing performance guarantees of a learned QP controller with the architecture described in Section 3. We provide sufficient conditions for persistent feasibility and asymptotic stability of the closed-loop system under a QP controller, which parallel

Algorithm 1: Framework of Learning of QP Controllers

Input: Simulation environment Env with nominal dynamics (1b), RL algorithm RL, policy architecture π_θ shown in Fig. 1, regularization coefficient ρ_{res}

Output: Optimized policy parameters $\theta = (L_P, H, W_q, W_b, b_b)$

for *epoch* $= 1, 2, \ldots$ **do**
- Interact with Env using current policy π_θ to collect a dataset $\mathcal{D}$
- Compute RL loss, denoted by $\ell_{RL}(\theta; \mathcal{D})$
- Compute residual loss $\ell_{res}(\theta; \mathcal{D})$ using (13)
- Update θ according to the loss $\ell_{RL}(\theta; \mathcal{D}) + \rho_{res}\ell_{res}(\theta; \mathcal{D})$

the theoretical guarantees for linear MPC (Borrelli et al., 2017). For simplicity, we consider the stabilization around the origin, i.e., $r = 0$, but the method of analysis can be extended to the general case. Additionally, we assume throughout the section that the optimal solution of the learned QP problem is attained, which can be ensured by allowing the QP solver to run sufficient iterations until convergence when deploying.

Denote the property under consideration as $\mathcal{P}$. Suppose that a certificate to $\mathcal{P}$, given the initial state x_0 is in a polytopic set $\mathcal{X}_0$, can be written in the following form:

$$\min_{x_0 \in \mathcal{X}_0, u_0, \nu} \{f(x_0, u_0, \nu) | g(x_0, u_0, \nu) \le 0, u_0 = \pi_\theta(x_0)\} \ge 0 \Rightarrow \mathcal{P} \text{ holds when } x_0 \in \mathcal{X}_0, \tag{14}$$

where π_θ denotes the θ-parameterized control policy described in Section 3, ν is an auxiliary variable, and f, g are quadratic (possibly nonconvex) functions. The optimization problem in the LHS of (14) can be expressed as a bilevel problem by explicitly expanding the control policy π_θ as:

$$\begin{aligned} &\pi_\theta(x_0) = y^\star_{[1:m_{sys}]}, y^\star \in \arg\min \left\{(1/2)y^\top P y + q^\top y \mid Hy + b \ge 0\right\}, \\ &\text{where } q = W_q x_0, b = W_b x_0 + b_b. \end{aligned} \tag{15}$$

Replacing the inner-level problem in (15) by its KKT condition, the verification problem in (14) can be cast into a nonconvex Quadratically Constrained Quadratic Program (QCQP) with variables x_0, ν, y, μ. Various computationally tractable methods for lower bounding the optimal value of a QCQP are available, such as Lagrangian relaxation (d'Aspremont and Boyd, 2003) and the method of moments (Lasserre, 2001), and once a nonnegative lower bound is obtained, the property $\mathcal{P}$ is verified.

Verification of persistent feasibility and asymptotic stability both fall into the framework described above. The conclusions are stated as follows:

Theorem 2 (Certificate for Persistent Feasibility) *The control policy* (15) *if persistently feasible (i.e., gives a valid control input that keeps the next state inside the bounds at every step) for all initial states $x_0 \in \mathcal{X}_0 = \{x | Gx \le c\}$, if the optimal value of the following nonconvex QCQP is nonnegative:*

$$\begin{aligned} \underset{x_0, \nu, y, \mu}{\textit{minimize}} \quad & -\nu^\top (G(Ax_0 + By_{[1:m_{sys}]}) - c), \\ \textit{subject to} \quad & Gx_0 \le c, \nu \ge 0, \mathbf{1}^\top \nu = 1, \\ & Py + W_q x_0 - H^\top \mu = 0, Hy + W_b x_0 + b_b \ge 0, \mu \ge 0, \mu^\top (Hy + W_b x_0 + b_b) = 0. \end{aligned}$$

To certify asymptotic stability, we consider the Lyapunov function of a stabilizing baseline MPC, and attempt to show that the Lyapunov function decreases along all trajectories even if the learned QP controller is deployed instead of the baseline MPC. A similar technique has been applied to the stability analysis of approximate MPC (Schwan et al., 2023). To formalize this idea, we define the following notations: $l(x, u) = x^\top Qx + u^\top Ru$ is the stage cost; the baseline MPC policy has horizon N, terminal constraint $x_N \in \mathcal{X}_f$ and terminal cost $V_f(x_N)$; the function $J(x_0, u_{0:N-1}) = \sum_{k=0}^{N-1} l(x_k, u_k) + V_N(x_N)$, where $x_{k+1} = Ax_k + Bu_k$, is the objective function of the baseline MPC. To ensure that the baseline MPC is stabilizing as long as it is feasible, one can choose $\mathcal{X}_f$ to be an invariant set under a stabilizing linear feedback controller $u = Kx$, and $V_f(x)$ to be the cost-to-go under $u = Kx$. Based on these notations, a certificate for asymptotic stability can be stated as follows:

Theorem 3 (Certificate for Asymptotic Stability) *Let $\mathcal{X}_0 = \{x|Gx \le c\}$ be a set where the baseline MPC is well-defined and the policy (15) is persistently feasible. The closed-loop system under (15) is asymptotically stable on $\mathcal{X}_0$, if $b_b \ge 0$, and there exists $\epsilon > 0$ and $N \in \mathbb{N}^*$, such that the optimal value of the following problem is nonnegative:*

$$\begin{aligned}
\underset{x_0, \bar{u}_{0:N}, y, \mu}{\textit{minimize}} \quad & J(x_0, \bar{u}_{0:N}) + l(x_0, y_{[1:m_{sys}]}) - J(x_0, (y_{[1:m_{sys}]}, \bar{u}_{1:N})) - \epsilon \|x_0\|^2, \\
\textit{subject to} \quad & Gx_0 \le c, x_{N+1}(Ax_0 + By_{[1:m_{sys}]}, \bar{u}_{1:N}) \in \mathcal{X}_f, \\
& Py + W_q x_0 - H^\top \mu = 0, Hy + W_b x_0 + b_b \ge 0, \mu \ge 0, \mu^\top (Hy + W_b x_0 + b_b) = 0.
\end{aligned}$$

Proofs of Theorems 2 and 3, as well as numerical examples showcasing the verification of a learned controller on a double integrator system, are provided in (Lu et al., 2023a, Appendix D).

5. Benchmarking Results

In our empirical evaluations, we aim to answer the following questions:

- How does the learned QP controller compare with common baselines (MPC, RL-trained MLP) on nominal linear systems?
- Can the learned QP controller handle modeling inaccuracies and disturbances?
- Does the method generalize to real-world robot systems with modeling inaccuracy and non-linearity?

We only briefly describe the experimental setup and typical results in this section, with complete details on systems, setup, hyperparameters, baseline definitions, and additional results in (Lu et al., 2023a, Appendix E). Code is available at `https://github.com/yiwenlu66/learning-qp`.

5.1. Results on Nominal Systems

We compare the Learned QP (LQP) controller with MPC and MLP baselines on benchmark systems like the quadruple tank (Johansson, 2000) and cartpole (Geva and Sitte, 1993), generating random initial states and references across 10^4 trials. For MPC, we evaluate variants with and without manually tuned terminal costs over short (2 steps) and long (16 steps) horizons, all implemented using OSQP (Stellato et al., 2020), a solver known for its efficiency in MPC applications (Forgione et al., 2020), with default solver configurations. Both LQP and MLP are trained using PPO, maintaining consistent reward definitions and RL hyperparameters. We incrementally increase the MLP size until further increases yield negligible performance improvements, selecting this size for comparison. The LQP is assessed in both small ($n_{qp} = 4, m_{qp} = 24$) and large ($n_{qp} = 16, m_{qp} = 96$) configurations, approximately aligning with the QP problem sizes from short- and long-horizon MPC.

Table 1: Performance comparison on benchmark systems.

Method \ Metrics	Quadruple Tank					Cartpole Balancing				
	Fail%	Cost	P-Cost	FLOPs	#Params	Fail%	Cost	P-Cost	FLOPs	#Params
MPC(2)	16.59	236.1	275.7	$95\text{K}_{+1.2\text{M}}$	-	100.0	1.36	129	$67\text{K}_{+814\text{K}}$	-
MPC(16)	4.27	228.3	237.2	$22\text{M}_{+52\text{M}}$	-	46.86	<u>0.34</u>	8.39	$3.9\text{M}_{+47\text{M}}$	-
MPC-T(2)	4.23	239.6	248.5	$470\text{K}_{+779\text{K}}$	-	100.0	1.41	122	$89\text{K}_{+792\text{K}}$	-
MPC-T(16)	3.22	**224.8**	<u>231.5</u>	$26\text{M}_{+49\text{M}}$	-	4.74	**0.30**	**0.79**	$51\text{M}_{+50\text{M}}$	-
RL-MLP	**0.03**	266.7	266.7	<u>23K</u>	11K	**3.23**	0.57	0.91	<u>87K</u>	43K
LQP(4, 24)	0.18	272.5	272.8	**14K**	**0.3K**	<u>3.49</u>	0.76	1.12	**14K**	**0.2K**
LQP(16, 96)	<u>0.13</u>	<u>227.3</u>	**227.6**	208K	<u>2.6K</u>	4.11	0.44	<u>0.87</u>	208K	<u>2.2K</u>

Methods: MPC(N) = MPC (Problem 2) with horizon N without terminal cost; MPC-T(N) = MPC with horizon N and manually tuned terminal cost; RL-MLP = reinforcement learning controller with MLP policy; LQP(n_{qp}, m_{qp}) = proposed learned QP controller with problem dimensions (n_{qp}, m_{qp}). Metrics: Fail% = percentage of early-terminated trials due to constraint violation; Cost = average LQ cost until termination; P-Cost = average cost with penalty for constraint violation; FLOPs = floating point operations per control step (reported as $\text{median}_{+(\text{max}-\text{median})}$ for variable data); #Params = number of learnable policy parameters. Best is highlighted in **bold**, and second best is <u>underlined</u>.

All training (including simulation and policy update) are performed on a single NVIDIA RTX 4090 GPU, with the small and large configurations taking 1.2 hours and 2.7 hours respectively.

The results of the benchmarking experiments are summarized in Table 1. In terms of control performance, LQP demonstrates comparable effectiveness to both MPC and MLP baselines. A benefit of LQP is its independence from manual tuning of the terminal cost, which can be important for MPC methods. Regarding computational efficiency, LQP stands out for its minimal demand for achieving similar control performance. This efficiency stems from LQP's fixed number of unrolled QP solver iterations. While MPC's computation cost varies based on implementation, the light computation of LQP is still noteworthy, especially in scenarios with tight computational limits. For example, the LQP(4, 24) configuration, despite having lowest FLOPs among all methods, still manages acceptable control performance. Finally, in terms of the number of learnable policy parameters, LQP requires substantially fewer than the RL-MLP. This hints at LQP's suitability for memory-constrained embedded systems and applicability to online few-shot learning.

Results on additional systems, including a numerical example of verifying the stability of the learned controllers, are deferred to the supplementary materials due to space limit (Lu et al., 2023a).

5.2. Validation of Robustness

This subsection is concerned with the robustness of the learned QP controller against modeling inaccuracies and disturbances. Instead of the nominal dynamics (1b), we now consider the following perturbed dynamics:

$$x_{k+1} = (A + \Delta A)x_k + (B + \Delta B)u_k + w_k,$$

where $\Delta A, \Delta B$ are parametric uncertainties, and w_k is a disturbance. LQP and MLP are trained using domain randomization (Tobin et al., 2017; Mehta et al., 2020), where the simulator randomly sample these uncertain components during training. Robust MPC baselines,

Table 2: Performance comparison on quadruple tank system with process noise and parametric uncertainties.

Method	Fail%	Cost	P-Cost	Time(s)	#Params
MPC-T(16)	82.6	**216.8**	713.4	$0.25_{+0.56}$	-
Tube	81.9	<u>233.3</u>	597.9	2.22_{+44}	-
Scenario	16.4	236.9	273.2	5.21_{+18}	-
RL-MLP	**1.3**	238.9	**241.5**	<u>1×10^{-3}</u>	43K
LQP(4, 24)	1.5	256.7	261.8	$\mathbf{2\times10^{-4}}$	**0.3K**
LQP(16, 96)	<u>1.4</u>	240.6	<u>243.4</u>	2×10^{-3}	<u>2.7K</u>

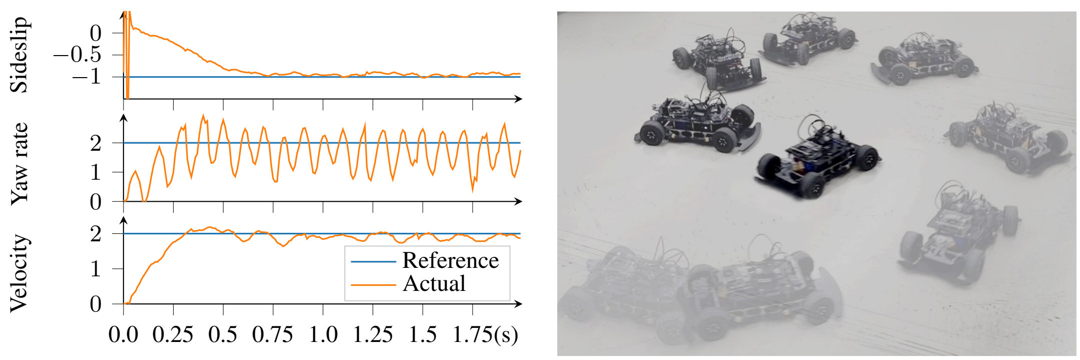
Notations are similar to those in the caption of Table 1. Computation time instead of FLOPs per control step is used as the metric for computational efficiency since it is difficult to obtain the exact FLOPs from the robust MPC baselines.

including tube MPC (Mayne et al., 2005) and scenario MPC (Bernardini and Bemporad, 2009) implemented by the do-mpc toolbox (Fiedler et al., 2023), are included for comparison.

The results in Table 2 highlight LQP's robustness, as it achieves the success rate and constraint-violation-penalized cost comparable to MLP. Also, it requires significantly less online computation compared to robust MPC methods, benefitting from domain randomization known for its effectiveness in empirical RL and robotics (Loquercio et al., 2019; Margolis et al., 2022).

5.3. Application Example on a Real-World System: Vehicle Drift Maneuvering

LQP is also evaluated on a challenging robotics control task, namely, the drift maneuvering of a 1/10 scale RC car, similar to the problem studied in Yang et al. (2022); Domberg et al. (2022); Lu et al. (2023b). The objective is to track the yaw rate, side slip angle, and velocity references, such that the car enters and maintains a drifting state. Despite the high nonlinearity of the system, the proposed controller formally introduced on linear systems successfully generalizes to this task. As shown in Fig. 2, the learned QP controller can track the references and maintain the drifting state, performing similarly to previous RL-trained MLP methods on this task (Domberg et al., 2022).



(a) Tracking performance (b) Blended frames of experiment video

Figure 2: Result of deploying learned QP controller to the vehicle drift maneuvering task. Video available at: `https://youtu.be/-XYtl2b4OVc`.

6. Conclusion

This work presents a novel class of QP controllers inspired by MPC. The proposed controllers not only retain the theoretical guarantees akin to MPC, but also exhibit desirable empirical performance and computational efficiency. Benchmarks including applications in real-world scenarios like vehicle drift maneuvering, further validate the effectiveness and robustness of our approach.

Despite the promising results, several challenges remain: (i) the empirical results are not exhaustive nor conclusive, and further evidence is required to understand LQP's benefits and limitations in practice; (ii) the stability verification (Theorem 3) still relies on the Lyapunov function of MPC, urging the simultaneous learning of policy and certificate similar to Chang et al. (2019); (iii) the performance guarantees assume that the optimal QP solution is attained, a restriction that can potentially be lifted using techniques from Wu et al. (2022).

Acknowledgments

This work is supported by National Natural Science Foundation of China under grant no. 62192752.

References

Shokhjakon Abdufattokhov, Mario Zanon, and Alberto Bemporad. Learning convex terminal costs for complexity reduction in mpc. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2163–2168. IEEE, 2021.

Adrian Agogino, Ritchie Lee, and Dimitra Giannakopoulou. Challenges of explaining control. In *2nd ICAPS Workshop on Explainable Planning (XAIP'19)*, 2019.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018.

Daniele Bernardini and Alberto Bemporad. Scenario-based model predictive control of stochastic constrained linear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6333–6338. IEEE, 2009.

Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Florian D Brunner, Mircea Lazar, and Frank Allgöwer. Stabilizing model predictive control: On the enlargement of the terminal set. *International Journal of Robust and Nonlinear Control*, 25(15): 2646–2670, 2015.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

Alexandre d'Aspremont and Stephen Boyd. Relaxations and randomized methods for nonconvex qcqps. *EE392o Class Notes, Stanford University*, 1:1–16, 2003.

Vishnu R Desaraju and Nathan Michael. Experience-driven predictive control. *Robot Learning and Planning (RLP 2016)*, page 29, 2016.

Fabian Domberg, Carlos Castelar Wembers, Hiren Patel, and Georg Schildbach. Deep drifting: Autonomous drifting of arbitrary trajectories using deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7753–7759. IEEE, 2022.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

Peter Englert, Ngo Anh Vien, and Marc Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *The International Journal of Robotics Research*, 36(13-14): 1474–1488, 2017.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Infinite-horizon model predictive control for periodic tasks with contacts. *Robotics: Science and systems VII*, page 73, 2012.

Felix Fiedler, Benjamin Karg, Lukas Lüken, Dean Brandner, Moritz Heinlein, Felix Brabender, and Sergio Lucia. do-mpc: Towards fair nonlinear and robust model predictive control. *Control Engineering Practice*, 140:105676, 2023.

Michael G Forbes, Rohit S Patwardhan, Hamza Hamadah, and R Bhushan Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48(8):531–538, 2015.

Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded MPC. In *Proc. of the 21st IFAC World Congress 2020, Berlin, Germany, July 12-17 2020*, 2020.

Shlomo Geva and Joaquin Sitte. A cartpole experiment benchmark for trainable controllers. *IEEE Control Systems Magazine*, 13(5):40–51, 1993.

Sébastien Gros and Mario Zanon. Data-driven economic nmpc using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636–648, 2019.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. *Advances in neural information processing systems*, 29, 2016.

Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Lukas Hewing, Juraj Kabzan, and Melanie N Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2019.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

Karl Henrik Johansson. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. *IEEE Transactions on control systems technology*, 8(3):456–465, 2000.

Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62, 2022.

Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14, 2019.

Yiwen Lu, Zishuo Li, Yihan Zhou, Na Li, and Yilin Mo. Mpc-inspired reinforcement learning for verifiable model-free control. *arXiv preprint arXiv:2312.05332*, 2023a.

Yiwen Lu, Bo Yang, Jiayun Li, Yihan Zhou, Hongshuai Chen, and Yilin Mo. Consecutive inertia drift of autonomous rc car via primitive-based planning and data-driven control. In *2023 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2023b.

Michael Lutter, Kim Listmann, and Jan Peters. Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7718–7725. IEEE, 2019.

Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

Marcel Menner, Peter Worsnop, and Melanie N Zeilinger. Constrained inverse optimal control with application to a human manipulation task. *IEEE Transactions on Control Systems Technology*, 29(2):826–834, 2019.

Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & chemical engineering*, 23(4-5):667–682, 1999.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Pavel Osinenko, Dmitrii Dobriborsci, and Wolfgang Aumer. Reinforcement learning with guarantees: a review. *IFAC-PapersOnLine*, 55(15):123–128, 2022.

S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

Ernest K Ryu and Wotao Yin. *Large-scale convex optimization: algorithms & analyses via monotone operators*. Cambridge University Press, 2022.

Yahya Sattar and Samet Oymak. Quickly finding the best linear model in high dimensions via projected gradient descent. *IEEE Transactions on Signal Processing*, 68:818–829, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Roland Schwan, Colin N Jones, and Daniel Kuhn. Stability verification of neural network controllers using mixed-integer programming. *IEEE Transactions on Automatic Control*, 2023.

Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5-6):1327–1349, 2021.

Raffaele Soloperto, Matthias A Müller, Sebastian Trimpe, and Frank Allgöwer. Learning-based robust model predictive control with state-dependent uncertainty. *IFAC-PapersOnLine*, 51(20): 442–447, 2018.

Mario Srouji, Jian Zhang, and Ruslan Salakhutdinov. Structured control nets for deep reinforcement learning. In *International Conference on Machine Learning*, pages 4742–4751. PMLR, 2018.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12 (4):637–672, 2020.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Wuwei Wu, Jianqi Chen, and Jie Chen. Stability analysis of systems with recurrent neural network controllers. *IFAC-PapersOnLine*, 55(12):170–175, 2022.

Zhaoming Xie, Glen Berseth, Patrick Clary, Jonathan Hurst, and Michiel van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246. IEEE, 2018.

Bo Yang, Yiwen Lu, Xu Yang, and Yilin Mo. A hierarchical control framework for drift maneuvering of autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1387–1393. IEEE, 2022.

Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–4. IEEE, 2015.

Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. *Advances in Neural Information Processing Systems*, 35:29113–29125, 2022.

Vrushabh Zinage and Efstathios Bakolas. Neural koopman lyapunov control. *Neurocomputing*, 527:174–183, 2023.