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; Abdufattohkov 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 controller 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 $S^{++}_n$, 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 $\text{diag}(A_1, \ldots, A_n)$. 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)**

\[
\begin{align*}
\text{minimize} & \quad \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} (x_{k+1} - r)^\top Q (x_{k+1} - r) + u_k^\top R u_k, \\
\text{subject to} & \quad x_{k+1} = Ax_k + Bu_k, \\
& \quad u_{\min} \leq u_k \leq u_{\max}, x_{\min} \leq x_{k+1} \leq x_{\max},
\end{align*}
\]

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 \( u_0^\star \) from the optimal solution is applied to the system in a receding horizon fashion.

**Problem 2 (Linear MPC)**

\[
\begin{align*}
\text{minimize} & \quad \sum_{k=0}^{N-1} (x_{k+1} - r)^\top Q (x_{k+1} - r) + u_k^\top R u_k, \\
\text{subject to} & \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N-1, \\
& \quad u_{\min} \leq u_k \leq u_{\max}, x_{\min} \leq x_{k+1} \leq x_{\max}, \quad k = 0, \ldots, N-1.
\end{align*}
\]

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

**Problem 3 (Standard-form QP)**

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} y^\top P y + q^\top y, \\
\text{subject to} & \quad H y + b \geq 0,
\end{align*}
\]

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 \( y = [u_0^\top \cdots u_{N-1}^\top]^\top \) 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:

\[
\begin{align*}
n_{qp} &= N m_{sys}, \quad m_{qp} = 2N (m_{sys} + n_{sys}), \\
P &= 2(B^\top Q B + R), \quad q = B^\top Q (A x_0 - r), \quad H = -C B - D, \quad b = e - C A x_0.
\end{align*}
\]

\(^1\) 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
\[
A = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad B = \begin{bmatrix} B \\ AB \\ B \\ \vdots \\ A^{N-1}B \end{bmatrix}, \quad C = I_N \otimes [I_{n_{xyz}} - I_{n_{xyz}}, 0_{n_{xyz}} \times 2n_{xyz}]^\top, \\
D = I_N \otimes [0_{m_{xyz} \times 2n_{xyz}}, I_{m_{xyz}} - I_{m_{xyz}}]^\top, \\
e = I_N \otimes [r_{\max} - r_{\min}, r_{\max} - r_{\min}]^\top, \\
r = I_N \otimes r, \quad Q = I_N \otimes Q, \quad R = I_N \otimes R.
\]

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

\[
z^{i+1} = R_{\mathbb{R}^{m_{QP}}}(I - 2\alpha F)z^i + \alpha(I - 2F)\lambda^i - 2\alpha\mu, \quad \lambda^{i+1} = F(z^i + \lambda^i) + \mu, \quad i = 0, 1, \ldots
\]

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:

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

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

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

**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:

\[
p^i - p^* \leq \|\lambda^i\|_2 r^i_{\text{prim}} + \|y^i - y^*\|_2 r^i_{\text{dual}}.
\]

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

\[
r^i_{\text{prim}} = Hy^i + b - z^i, \quad r^i_{\text{dual}} = H y^i + q - H^\top \lambda^i.
\]

**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 \((z, \lambda)\) can be implemented by interleaving an affine transformation whose parameters \((F, \mu)\) 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.
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 \(x_0\):

\[
q(x_0, r; W_q) = W_q [x_0^\top \: r^\top]^\top, \quad b(x_0, r; W_b, b_b) = W_b x_0 + b_b,
\]

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 \( n_{\text{iter}} \) 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 \( n_{\text{iter}} \) 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., \( n_{\text{iter}} = 10 \)), which mitigates the computational burden of the unrolling process. Intuitively, the sufficiency of a small \( n_{\text{iter}} \) 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 \( D \) of transition samples, it is defined as follows:

\[
\ell_{\text{res}}(\theta; D) = \frac{1}{|D|} \sum_{k=1}^{|D|} \left[ \| H y_{k}^{n_{\text{iter}}} + b_k - z_{k}^{n_{\text{iter}}} \|_2^2 + \| P y_{k}^{n_{\text{iter}}} + q_k - H^T \lambda_k^{n_{\text{iter}}} \|_2^2 \right],
\]

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 $\pi_\theta$ shown in Fig. 1, regularization coefficient $\rho_{res}$

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

for $\text{epoch} = 1, 2, \ldots$ do
  1. Interact with $Env$ using current policy $\pi_\theta$ to collect a dataset $\mathcal{D}$
  2. Compute RL loss, denoted by $\ell_{RL}(\theta; \mathcal{D})$
  3. Compute residual loss $\ell_{res}(\theta; \mathcal{D})$ using (13)
  4. Update $\theta$ 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 $x_0 \in X_0 = \{x | Gx \leq c\}$, 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 $x_0 \in X_0 = \{x | Gx \leq c\}$, if the optimal value of the following nonconvex QCQP is nonnegative:

$$\begin{align*}
\min_{x_0, u_0, \nu} & \quad \{f(x_0, u_0, \nu)|g(x_0, u_0, \nu) \leq 0, u_0 = \pi_\theta(x_0)\} \geq 0 \Rightarrow \mathcal{P} \text{ holds when } x_0 \in X_0,
\end{align*}$$

(14)

where $\pi_\theta$ denotes the $\theta$-parameterized control policy described in Section 3, $\nu$ 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 $\pi_\theta$ as:

$$\pi_\theta(x_0) = y^*_1, y^* \in \arg \min \left\{ (1/2)y^\top P y + q^\top y | Hy + b \geq 0 \right\},$$

where $q = W_q x_0, b = W_b x_0 + b_b$.

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 $x_0, \nu, y, \mu$. 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 $x_0 \in X_0 = \{x | Gx \leq c\}$, if the optimal value of the following nonconvex QCQP is nonnegative:

$$\begin{align*}
\min_{x_0, \nu, y, \mu} & \quad -\nu^\top (G(Ax_0 + By_1) - c),
\end{align*}$$

subject to $Gx_0 \leq c, \nu \geq 0, 1^\top \nu = 1, P y + W_q x_0 - H^\top \mu = 0, H y + W_b x_0 + b_b \geq 0, \mu \geq 0, \mu^\top (H y + W_b x_0 + b_b) = 0.$

7
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: \( l(x, u) = x^\top Q x + u^\top R u \) is the stage cost; the baseline MPC policy has horizon \( N \), terminal constraint \( x_N \in X_f \) and terminal cost \( V_f(x_N) \); the function \( J(x_0, u_0, u_{0:N-1}) = \sum_{k=0}^{N-1} l(x_k, u_k) + V_N(x_N) \), 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 \( X_f \) to be an invariant set under a stabilizing linear feedback controller. To formalize this idea, we define a certificate for asymptotic stability can be stated as follows:

**Theorem 3 (Certificate for Asymptotic Stability)** Let \( X_0 = \{ x | Gx \leq c \} \) 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 \( X_0 \), if \( b_0 \geq 0 \), and there exists \( \epsilon > 0 \) and \( N \in \mathbb{N}^* \), such that the optimal value of the following problem is nonnegative:

\[
\min_{x_0, \bar{u}_0, N, b, \mu} J(x_0, \bar{u}_0, N) + l(x_0, y_{[1:m_{sys}]}(\bar{u}_1:N)) - J(x_0, (y_{[1:m_{sys}]}), \bar{u}_1:N)) - \epsilon \|x_0\|^2,
\]

subject to

\[
Gx_0 \leq c, x_{N+1} = Ax_0 + By_{[1:m_{sys}]}(\bar{u}_1:N) \in X_f,
\]

\[
P y + W_q x_0 - H^\top \mu = 0, H y + W_b x_0 + b_b \geq 0, \mu \geq 0, \mu^\top (H y + W_b x_0 + b_b) = 0.
\]

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](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.
MPC-Inspired Reinforcement Learning for Verifiable Model-Free Control

Table 1: Performance comparison on benchmark systems.

<table>
<thead>
<tr>
<th>Method</th>
<th>Metrics</th>
<th>Quadruple Tank</th>
<th>Cartpole Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fail%</td>
<td>Cost</td>
<td>P-Cost</td>
</tr>
<tr>
<td>MPC(2)</td>
<td>16.59</td>
<td>236.1</td>
<td>275.7</td>
</tr>
<tr>
<td>MPC(16)</td>
<td>4.27</td>
<td>228.3</td>
<td>237.2</td>
</tr>
<tr>
<td>MPC-T(2)</td>
<td>4.23</td>
<td>239.6</td>
<td>248.5</td>
</tr>
<tr>
<td>MPC-T(16)</td>
<td>3.22</td>
<td>224.8</td>
<td>231.5</td>
</tr>
<tr>
<td>RL-MLP</td>
<td>0.03</td>
<td>266.7</td>
<td>266.7</td>
</tr>
<tr>
<td>LQP(4, 24)</td>
<td>0.18</td>
<td>272.5</td>
<td>272.8</td>
</tr>
<tr>
<td>LQP(16, 96)</td>
<td>0.13</td>
<td>227.3</td>
<td>227.6</td>
</tr>
</tbody>
</table>

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 median + (max − median) for variable data); #Params = number of learnable policy parameters. Best is highlighted in **bold**, and second best is underlined.

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Fail%</th>
<th>Cost</th>
<th>P-Cost</th>
<th>Time(s)</th>
<th>#Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC-T(16)</td>
<td>82.6</td>
<td><strong>216.8</strong></td>
<td>713.4</td>
<td>0.25, 0.56</td>
<td>-</td>
</tr>
<tr>
<td>Tube</td>
<td>81.9</td>
<td>233.3</td>
<td>597.9</td>
<td>2.22, 2.44</td>
<td>-</td>
</tr>
<tr>
<td>Scenario</td>
<td>16.4</td>
<td>236.9</td>
<td>273.2</td>
<td>5.21, 1.18</td>
<td>-</td>
</tr>
<tr>
<td>RL-MLP</td>
<td><strong>1.3</strong></td>
<td>238.9</td>
<td><strong>241.5</strong></td>
<td>1 × 10⁻³</td>
<td>43K</td>
</tr>
<tr>
<td>LQP(4, 24)</td>
<td>1.5</td>
<td>256.7</td>
<td>261.8</td>
<td>2 × 10⁻³</td>
<td><strong>0.3K</strong></td>
</tr>
<tr>
<td>LQP(16, 96)</td>
<td>1.4</td>
<td>240.6</td>
<td>243.4</td>
<td>2 × 10⁻³</td>
<td>2.7K</td>
</tr>
</tbody>
</table>

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).

![Figure 2: Result of deploying learned QP controller to the vehicle drift maneuvering task. Video available at: https://youtu.be/-XYt12b4OVc.](https://example.com/figure2.png)

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.
MPC-INSPIRED REINFORCEMENT LEARNING FOR VERIFIABLE MODEL-FREE CONTROL

References


