Blending MPC & Value Function Approximation for Efficient Reinforcement Learning

Mohak Bhardwaj[1] **Sanjiban Choudhury**[2] **Byron Boots**[1]

[1]University of Washington [2]Aurora Innovation Inc.

Abstract

We present a framework for improving on Model-Predictive Control (MPC) with model-free reinforcement learning (RL). The key insight is to view MPC as constructing a series of local Q-function approximations using an approximate dynamics model. We show that by using a parameter λ, similar to the trace decay parameter in TD(λ), we can systematically trade-off learned value estimates against the local Q-function approximations. We present a theoretical analysis that shows how errors from the two sources can be balanced and validate our approach on challenging high-dimensional manipulation tasks with biased models in simulation. Project website: https://bit.ly/33K2CK6

1 Introduction

Model-free RL is increasingly used in challenging sequential decision-making problems including high-dimensional robotics control tasks [4, 12] as well as games [9, 14, 15]. While these approaches can theoretically solve complex problems with little prior knowledge, they also typically require a large quantity of training data to succeed. In robotics and engineering domains, data may be collected from real-world interaction, a process that can be dangerous, time-consuming, and expensive.

MPC offers a more practical alternative. While RL typically uses data to learn a global model offline, which is then deployed at test time, MPC solves for a policy *online* by optimizing an approximate model for a finite horizon at a given state. This policy is then executed for a single timestep and the process repeats. MPC is one of the most popular approaches for control of complex, safety-critical systems [1, 16], owing to its ability to use approximate models to optimize complex cost functions and constraints [8, 7].

However, *model bias* in MPC may result in *persistent errors* that eventually compound and become catastrophic. For example, in non-prehensile manipulation, practitioners often use a simple quasi-static model that assumes an object does not roll or slide away when pushed. For more dynamic objects, this can lead to aggressive pushing policies that perpetually over-correct, eventually driving the object off the surface.

Recently, there have been several approaches to combine MPC with model-free RL by learning a terminal cost function via RL, thereby increasing the effective horizon of MPC [18, 6, 2]. However, the learned value function is only applied at the end and model errors would still persist in the horizon. In this paper, we focus on a broader question: can machine learning be used to both increase the effective horizon of MPC, while also correcting for model bias?

One straightforward approach is to try to learn (or correct) the model from real data. However, hand-constructed models are often crude-approximations of reality and lack the expressivity to represent encountered dynamics. Moreover, increasing the complexity of such models leads to computationally expensive updates that can harm MPC's online performance. Model-based RL approaches [3, 10, 13] aim to learn general neural network models directly from data. However, learning globally consistent models suffers from covariate shift issues [11].

We propose a framework, MPQ(λ), for weaving together MPC with learned value estimates to trade-off errors in the MPC model and approximation error in a learned value function. Our key insight is to view MPC as tracing out a series of local Q-function approximations. We can then blend each of these

Q-functions with value estimates from reinforcement learning. We show that by using a blending parameter λ, similar to the trace decay parameter in TD(λ), we can systematically trade-off errors between these two sources. Moreover, by smoothly decaying λ over time we can achieve the best of both worlds - a policy can depend on a prior model before it has encountered any data and then gradually become more reliant on learned value estimates with experience. We present theoretical analysis of finite-horizon planning with approximate models and value functions and empirically evaluate our approach on challenging manipulation problems with varying degrees of model-bias.

2 Preliminaries

2.1 Reinforcement Learning

We consider an infinite-horizon discounted Markov Decision Process (MDP) defined by a tuple $\mathcal{M}=(\mathcal{S},\mathcal{A},c,P,\gamma,\mu)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $c(s,a)$ is the per-step cost function, $s_{t+1}\sim P(\cdot|s_t,a_t)$ is the stochastic transition dynamics and γ is the discount factor. $\mu(s_0)$ is a distribution over initial states and policy $\pi(\cdot|s)$ is a distribution over actions given state. Let $\mu^{\pi}_{\mathcal{M}}$ be the distribution over state-action trajectories obtained by running policy π on $\mathcal{M}$. The value function for a given policy π, is defined as $V^{\pi}_{\mathcal{M}}(s)=\mathbb{E}_{\mu^{\pi}_{\mathcal{M}}}\left[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t)\,|\,s_0=s\right]$ and the action-value function as $Q^{\pi}_{\mathcal{M}}(s,a)=\mathbb{E}_{\mu^{\pi}_{\mathcal{M}}}\left[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t)\,|\,s_0=s,a_0=a\right]$. The objective is to find an optimal policy $\pi^*=\underset{\pi}{\operatorname{argmin}}\,\mathbb{E}_{s_0\sim\mu}[V^{\pi}_{\mathcal{M}}(s_0)]$. We can also define the (dis)-advantage function $A^{\pi}_{\mathcal{M}}(s,a)=Q^{\pi}_{\mathcal{M}}(s,a)-V^{\pi}_{\mathcal{M}}(s)$, which measures how good an action is compared to the action taken by the policy in expectation. It can be equivalently expressed in terms of the Bellman error as $A^{\pi}_{\mathcal{M}}(s,a)=c(s,a)+\gamma\mathbb{E}_{s'\sim P,a'\sim\pi}[Q^{\pi}_{\mathcal{M}}(s',a')]-\mathbb{E}_{a\sim\pi}[Q^{\pi}_{\mathcal{M}}(s,a)]$.

2.2 Model-Predictive Control

Instead of trying to solve for a single, globally optimal policy, MPC optimizes simple, local policies *online*. At every encountered state, MPC uses an approximate dynamics model to search for a parameterized policy that minimizes cost over a finite horizon. An action is sampled from the policy and executed on the system. The process is repeated from the next state, often by warm-starting the optimization from the previous solution.

We formalize this process as solving a simpler *surrogate* MDP $\hat{\mathcal{M}}=(\mathcal{S},\mathcal{A},\hat{c},\hat{P},\gamma,\hat{\mu},H)$ online, which differs from $\mathcal{M}$ by using an approximate cost function $\hat{c}$, transition dynamics $\hat{P}$ and limiting horizon to H. Due to finite H, a terminal state-action value function $\hat{Q}$ that estimates the cost-to-go is used. The start state distribution $\hat{\mu}$ is a dirac-delta function centered on the current state $s_0=s_t$. MPC can be viewed as iteratively constructing an estimate of the Q-function of the original MDP $\mathcal{M}$, given policy π_ϕ at state s:

$$Q^{\phi}_{H}(s,a)=\mathbb{E}_{\mu^{\pi_\phi}_{\mathcal{M}}}\left[\sum_{i=0}^{H-1}\gamma^i\hat{c}(s_i,a_i)+\gamma^H\hat{Q}(s_H,a_H)\,|\,s_0=s,a_0=a\right] \tag{1}$$

MPC then iteratively optimizes it (at system state s_t) to update the policy parameters $\phi^*_t=\underset{\phi}{\operatorname{argmin}}\,Q^{\phi}_{H}(s_t,\pi_\phi(s_t))$. Alternatively, we can view the above from the perspective of disadvantage minimization. Let $A(s_i,a_i)=c(s_i,a_i)+\gamma\hat{Q}(s_{i+1},a_{i+1})-\hat{Q}(s_i,a_i)$ be an estimator for the 1-step disadvantage with respect to $\hat{Q}$. We can equivalently write the above optimization as minimizing the discounted sum of disadvantages via the telescoping sum trick as $\underset{\pi\in\Pi}{\operatorname{argmin}}\,\mathbb{E}_{\mu^{\pi_\phi}_{\mathcal{M}}}\left[\hat{Q}(s_0,a_0)+\lambda\sum_{i=0}^{H-1}\gamma^i A(s_i,a_i)\,|\,s_0=s_t\right]$. Although this queries the $\hat{Q}$ at every timestep, it is still exactly equivalent to (1) and hence, does not mitigate the effects of model-bias. Next, we build a concrete method to address this by formulating a way to blend Q-estimates from MPC and a learned value function that can balance their respective errors.

3 Mitigating Bias in MPC via Reinforcement Learning

The performance of MPC algorithms critically depends on the quality of the estimator $Q^{\phi}_{H}(s,a)$. There are three major sources of approximation error - (1) Model-bias causes compounding errors in predicted state trajectories, (2) the error in the terminal value gets propagated back to the Q-estimate at the start, and (3) a small H makes the algorithm myopic.We provide a formal bound on the performance of the policy with approximate models and approximate learned value functions in Theorem A.2 in Appendix A.1, where we see that the model error *increases* with horizon H while the learned value error *decreases* with H indicating that there is an optimal planning horizon H^* that minimizes the bound (derivation in

Appendix A.1.3). In practice, model and value function errors are unknown making it impossible to set H to H^*. Instead, we propose a strategy to blend the Q-estimates from MPC and the learned value function at every timestep along the horizon, such that we can properly balance the different sources of error.

3.1 Blending Model Predictive Control and Value Functions

A naive way to blend MPC Q-estimates with learned Q-estimates would be a convex combination of the two

$$(1-\lambda)\underbrace{\hat{Q}(s,a)}_{\text{model-free}}+\lambda\underbrace{Q^{\phi}_{H}(s,a)}_{\text{model-based}} \tag{2}$$

where $\lambda \in [0,1]$. Here, the value function contributes to a residual added to the MPC output [5]. However, this is solution is rather *ad hoc*. Why invoke the value function at only at the first and last timestep? As the value function gets better, it should be useful to invoke it *at all timesteps*. Instead, consider the following recursive formulation for the Q-estimate. Given (s_i,a_i), the state-action encountered at horizon i, the blended estimate $Q^{\lambda}(s_i,a_i)$ is expressed as

$$\underbrace{Q^{\lambda}(s_i,a_i)}_{\text{current blended estimate}}=(1-\lambda)\underbrace{\hat{Q}(s_i,a_i)}_{\text{model-free}}+\lambda(\underbrace{\hat{c}(s_i,a_i)}_{\text{model-based}}+\gamma\underbrace{Q^{\lambda}(s_{i+1},a_{i+1})}_{\text{future blended estimate}}) \tag{3}$$

The recursion ends at $Q^{\lambda}(s_H,a_H)=\hat{Q}(s_H,a_H)$. In other words, the current blended estimate is a convex combination of the model-free value function and the one-step model-based return, which in turn uses the future blended estimate. Note unlike (2), the model-free estimate is invoked at *every timestep*. We can unroll (3) in time to show $Q^{\lambda}_{H}(s,a)$, the blended $H-$horizon estimate, is simply an exponentially weighted average of *all horizon* estimates

$$Q^{\lambda}_{H}(s,a)=(1-\lambda)\sum_{i=0}^{H-1}\lambda^{i}Q^{\phi}_{i}(s,a)+\lambda^{H}Q^{\phi}_{H}(s,a) \tag{4}$$

where $Q^{\phi}_{k}(s,a)=\mathbb{E}_{\mu^{\pi_\phi}_{\mathcal{M}}}\left[\sum_{i=0}^{k-1}\gamma^{i}\hat{c}(s_i,a_i)+\gamma^{k}\hat{Q}(s_k,a_k)\,|\,s_0=s,a_0=a\right]$ is a k-horizon estimate. When $\lambda=0$, the estimator reduces to the just using $\hat{Q}$ and when $\lambda=1$ we recover the original MPC estimate Q_H in (1). For intermediate values of λ, we interpolate smoothly between the two by interpolating all H estimates. (4) can be implemented efficiently by switching to the disadvantage formulation by applying a similar telescoping trick

$$Q^{\lambda}_{H}(s,a)=\mathbb{E}_{\mu^{\pi_\phi}_{\mathcal{M}}}\left[\hat{Q}(s_0,a_0)+\sum_{i=0}^{H-1}(\gamma\lambda)^{i}A(s_i,a_i)\right] \tag{5}$$

This estimator has a similar form as the $TD(\lambda)$ estimator for the value function used for bias-variance trade-off. However, our blended estimator aims trade-off bias in dynamics model with bias in learned value function. We derive a critical λ^* that minimizes performance bound in Appendix A.1.4.

Why use blending λ when one can simply tune horizon H? First, H limits the *resolution* we can tune since it's an integer – as H gets smaller the resolution becomes worse. Second, the blended estimator $Q^{\lambda}_{H}(s,a)$ uses far more samples. Even if both Q^{λ}_{H} and $Q^{\phi}_{H^*}$ had the same bias, the latter uses a strict subset of samples as the former. Hence the variance of the blended estimator will be lower, with high probability.

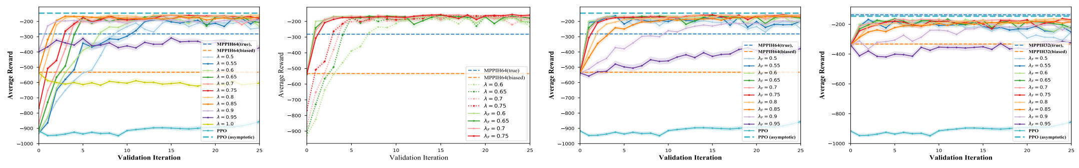
4 The MPQ(λ) Algorithm

We develop a simple variant of Q-Learning, called Model-Predictive Q-Learning with λ Weights (MPQ(λ)) that learns a parameterized Q-function estimate $\hat{Q}_\theta$. Our algorithm, presented in Alg. 1 in Appendix A.2, modifies Q-learning to use blended Q-estimates as described in (5), for both action selection and generating value targets. At timestep t, MPQ(λ) uses H-horizon MPC from the current state s_t to optimize parameters of a greedy policy π_ϕ w.r.t blended Q-estimator in (5)

$$\phi^*_t=\underset{\phi}{\operatorname{argmin}}\,\mathbb{E}_{\mu^{\pi_\phi}_{\mathcal{M}}}\left[\hat{Q}_\theta(s_0,a_0)+\sum_{i=0}^{H-1}(\gamma\lambda)^{i}A(s_i,a_i)\,|\,s_0=s_t\right] \tag{6}$$

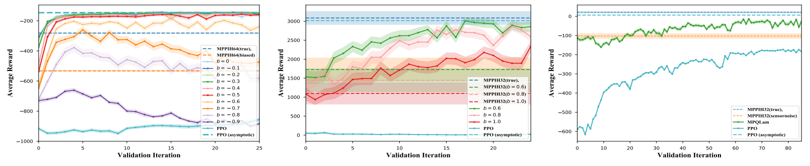
An action sampled from the resulting policy is then executed on the system. Periodically, the parameters θ are updated via stochastic gradient descent using mini-batches of experience tuples sampled from the replay buffer and the H-horizon MPC with blended Q-estimator from (6) is again invoked to calculate the targets. Refer to Appendix A.2 for full details.

MPC with the blended Q-estimator and an appropriate λ generates more stable Q-targets than using Q_θ or MPC with a terminal Q-function alone. We also slowly decay λ over time. Intuitively, in the early stages of learning, the bias in $\hat{Q}_\theta$ dominates and hence we want to rely more on the model. A larger λ is appropriate as it up-weights longer horizon estimates. Conversely, as $\hat{Q}_\theta$ improves, a smaller λ is favorable.



(a) Fixed λ (b) Fixed v/s Decaying λ (c) λ decay with $H = 64$ (d) λ decay with $H = 32$

Figure 1: CARTPOLESWINGUP experiments. Curves show rewards averaged over 30 validation episodes and 3 seeds. Training steps=$100k$, Validation freq=$4k$. When decaying λ, it is frozen during validation. λ_F is the λ at the end of training. Shaded region is standard deviation of MPPI reward.



(a) CARTPOLESWINGUP (b) INHANDMANIPULATION (c) SAWYERPEGINSERTION

Figure 2: Robustness and sample efficiency of MPQ(λ). (a) Varying bias over mass of cart and pole. (b) Varying bias over mass, inertia and friction of pen (c) Peg insertion with noisy perception. Same factor b is used for all altered properties. Results averaged over 30 validation episodes. In (a) λ is decayed in $[1.0, 0.75]$ and $[1.0, 0.85]$ in (b), (c)

5 Experiments

We evaluate MPQ(λ) on simulated robot control tasks (Appendix Fig.1) with a biased version of simulation used as the dynamics model for MPC. (1) CARTPOLESWINGUP: masses are set lower than the true values to make MPC persistently input smaller controls than desired, (2) SAWYERPEGINSERTION: Effects of inaccurate perception are tested by simulating a noisy position sensor at the target with MPC using a deterministic model, (3) INHANDMANIPULATION: mass, inertia and friction coefficients of the pen are set larger causing MPC policies to be overly aggressive. (Refer Appendix A.3 for details). Model Predictive Path Integral Control (MPPI) [17] is used as the MPC algorithm and simulation parameters are biased using $m = (1 + b)m_{true}$, where b is a bias-factor. We consider model-based and model-free baselines - (1) MPPI + true dynamics and no value function, (2) MPPI + biased dynamics and no value function (3) Proximal Policy Optimization (PPO) [12].

5.1 Analysis of Overall Performance

O 1. MPQ(λ) *is able to overcome model-bias in MPC for a wide range of* λ *values.*
Fig. 1(a) shows a comparison of MPQ(λ) with MPPI using true and biased dynamics with $b = -0.5$ and $H = 64$ for various settings of λ. There exists a wide range of λ values for which MPQ(λ) can efficiently trade-off the model and learned Q-function bias and out-perform MPPI with biased dynamics while achieving performance comparable to PPO in the limit. However, setting λ to a high value of 1.0 and 0.95, which weighs longer horizons heavily leads to poor performance as compounding effects of model bias are not being compensated for by Q_θ.

O 2. *Faster convergence can be achieved by decaying* λ *over time.*
As more data is collected on the system, we expect the bias in Q_θ to decrease, whereas model-bias remains constant. We decay λ in $[1.0, \lambda_F]$ using a fixed schedule (refer Appendix A.3) to reduce the dependence on MPC over time. This accelerates convergence than a fixed λ as shown in Fig. 1(b). Figures 1(c) and 1(d) show robustness of MPQ(λ) to a wide range of decay rates with $H = 64$ and 32 respectively. With true dynamics, MPPI with $H = 32$ outperforms $H = 64$ due to optimization issues with long horizons, while MPQ(λ) performs comparable with MPPI $H = 32$ and PPO upon convergence.

O 3. MPQ(λ) *is robust to large degree of model misspecification.*
Fig. 2(a) shows the effects of varying the bias on the mass of the cart and pole. MPQ(λ) outperforms MPPI ($H = 64$), both with biased and true dynamics, for a wide range of b, and convergence is generally faster for smaller bias. For large b, MPQ(λ) fails to improve. Similarly, in Fig. 2(b) MPQ(λ) achieves performance comparable to MPPI with true dynamics for different degrees of bias in properties of the pen. We conclude that while MPQ(λ) is robust to large amount of model-bias, if the model is extremely un-informative, relying on MPC can degrade performance making model-free RL the favorable option.

O 4. MPQ(λ) *is much more sample efficient compared to model-free RL on high-dimensional continuous control tasks, even with approximate models*
Figures 2(b) and 2(c) show comparison of MPQ(λ) with PPO on the INHANDMANIPULATION and SAWYERPEGINSERTION tasks respectively. In both cases, MPQ(λ) rapidly improves and achieves average reward comparable to MPPI with true dynamics, whereas PPO barely improves.

References

[1] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[2] Mohak Bhardwaj, Ankur Handa, Dieter Fox, and Byron Boots. Information theoretic model predictive q-learning. In *Learning for Dynamics and Control*, pages 840–850, 2020.

[3] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[5] Gilwoo Lee, Brian Hou, Sanjiban Choudhury, and Siddhartha S Srinivasa. Bayesian residual policy optimization: Scalable bayesian reinforcement learning with clairvoyant experts. *arXiv preprint arXiv:2002.03042*, 2020.

[6] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

[7] David Q Mayne, Erric C Kerrigan, EJ Van Wyk, and Paola Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353, 2011.

[8] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[10] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[11] Stephane Ross and J Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

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

[13] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pages 5779–5788, 2019.

[14] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[15] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[16] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[17] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[18] Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 100–107. IEEE, 2013.