

Neural Lyapunov Model Predictive Control

Supplementary Material

We provide proofs of the Theorems 1 and 2, introduced in the paper, in Appendix A. In Appendix B, we describe the formulation of the model predictive controller as a sequential quadratic program (SQP). In Appendix C, we discuss the experimental setup that comprises of the implementation specifics, details about baseline controllers, parameters for the experiments, and more description of the control problem setting. In Appendix D, we provide additional plots and results for the inverted pendulum and car kinematics examples. Finally, in Appendix E we discuss an algorithm for probabilistic safety verification.

A PROOF OF THE THEOREMS

Here we provide the proofs of Theorems stated in the paper. We write the proof for Theorem 2 before Theorem 1 since it is simpler and helps in proving the latter.

PROOF OF THEOREM 2

Proof. To prove the result, we first write:

$$\begin{aligned} \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}^*(x)] - \mathbf{E}_{x \in \mathcal{D}}[J_{V^*}^*(x)] = & \underbrace{\mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}^*(x)] - \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)]}_{I_1} + \\ & \underbrace{\mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)] - \mathbf{E}_{x \in \mathcal{D}}[J_{V^*}^*(x)]}_{I_2}, \end{aligned} \quad (13)$$

where $\mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)]$ denotes the MPC performance when a perfect model is used for predictions.

For the term I_2 , Lowrey et al. (2018) provide a bound on the performance of the MPC policy. It is important to note that in their problem formulation, the MPC's objective is defined as a maximization over the cumulative discounted reward, while in our formulation Equation 11 we consider a minimization over the cost. Consequently, compared to inequality presented by Lowrey et al. (2018), there is a change in sign of the terms in the left-hand side of the inequality. This means:

$$\mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)] - \mathbf{E}_{x \in \mathcal{D}}[J_{V^*}^*(x)] \leq \frac{2\gamma^N \epsilon}{1 - \gamma^N}. \quad (14)$$

We now focus on the term I_1 in Equation 13. Let us denote $x^*(i)$ and $u^*(i)$ as the optimal state and action predictions respectively, obtained by using the correct model, f , and the MPC policy at time i . By the principle of optimality, the optimal sequence for the MPC using the correct model, \underline{u}_f , can be used to upper-bound the optimal cost for the MPC using the surrogate model:

$$\mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}^*(x)] - \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)] \leq \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}(x, \underline{u}_f)] - \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)]. \quad (15)$$

Since the input sequence is now the same for both terms in right-hand side of Equation 15, the difference in the cost is driven by the different state trajectories cost (the cost on x over the horizon which includes the state-constraint violation penalty, as defined in Equation 23) as well as the terminal

cost. In form of equation, this means:

$$\begin{aligned}
& \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}(x, \underline{u}_f)] - \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC},f}^*(x)] \\
&= \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}(x, \underline{u}_f) - J_{\text{MPC},f}^*(x)] \\
&= \mathbf{E}_{x(0) \in \mathcal{D}} \left[\sum_{j=0}^{N-1} \gamma^j \left\{ \hat{x}(j)^T Q \hat{x}(j) - x^*(j)^T Q x^*(j) \right\} \right. \\
&\quad \left. + \gamma^N \alpha \{ V(\hat{x}(N)) - V(x^*(N)) \} \right. \\
&\quad \left. + \sum_{j=0}^N \{ \ell_{\mathbb{X}}(\hat{x}(j)) - \ell_{\mathbb{X}}(x^*(j)) \} \right]. \tag{16}
\end{aligned}$$

Recall that we assume the surrogate model is Lipschitz with constant $L_{\hat{f}_x}$. This means that $\forall \tilde{x}, x \in \mathbb{R}^{n_x}$ and the same input $u \in \mathbb{R}^{n_u}$, we have:

$$\|\hat{f}(\tilde{x}, u) - \hat{f}(x, u)\|_2 \leq L_{\hat{f}_x} \|\tilde{x} - x\|_2,$$

Further, from Equation 8, $\forall (x, u) \in \tilde{\mathbb{X}} \times \mathbb{U}$, we have:

$$\|w(x, u)\|_2 = \|f(x, u) - \hat{f}(x, u)\|_2 \leq \mu.$$

Under the optimal policy for the correct model, let us denote the deviation in the state prediction when the MPC input prediction is applied with a different model, \hat{f} , as $\hat{d}(j) := \hat{x}(j) - x^*(j)$.

At step $j = 1$:

$$\begin{aligned}
\|\hat{d}(1)\|_2 &= \|\hat{x}(1) - x^*(1)\|_2 \\
&= \|\hat{f}(x^*(0), u^*(0)) - f(x^*(0), u^*(0))\|_2 \\
&= \|w(x^*(0), u^*(0))\|_2 \\
&\leq \mu.
\end{aligned}$$

At step $j = 2$:

$$\begin{aligned}
\|\hat{d}(2)\|_2 &= \|\hat{x}(2) - x^*(2)\|_2 \\
&= \|\hat{f}(x^*(1) + \hat{d}(1), u^*(1)) - f(x^*(1), u^*(1))\|_2 \\
&= \|\hat{f}(x^*(1) + \hat{d}(1), u^*(1)) - \hat{f}(x^*(1), u^*(1))\|_2 \\
&\quad + \|\hat{f}(x^*(1), u^*(1)) - f(x^*(1), u^*(1))\|_2 \\
&\leq \underbrace{\|\hat{f}(x^*(1) + \hat{d}(1), u^*(1)) - \hat{f}(x^*(1), u^*(1))\|_2}_{\leq L_{\hat{f}_x} \|\hat{d}(1)\|_2} \\
&\quad + \underbrace{\|\hat{f}(x^*(1), u^*(1)) - f(x^*(1), u^*(1))\|_2}_{=\|w(x^*(1), u^*(1))\| \leq \mu} \\
&\leq L_{\hat{f}_x} \mu + \mu
\end{aligned}$$

By induction, it can be shown that:

$$\|\hat{d}(j)\|_2 = \|\hat{x}(j) - x^*(j)\|_2 \leq \sum_{i=0}^{j-1} L_{\hat{f}_x}^i \mu. \tag{17}$$

Alternately, if we assume the correct system that is to be controlled is Lipschitz with constant L_{f_x} , then proceeding as before:

At step $j = 1$:

$$\begin{aligned}
\|\hat{d}(1)\|_2 &= \|\hat{x}(1) - x^*(1)\|_2 \\
&\leq \mu.
\end{aligned}$$

At step $j = 2$:

$$\begin{aligned}
\|\hat{d}(2)\|_2 &= \|\hat{x}(2) - x^*(2)\| \\
&= \|\hat{f}(\hat{x}(1), u^*(1)) - f(\hat{x}(1) - \hat{d}(1), u^*(1))\| \\
&\leq \underbrace{\|\hat{f}(\hat{x}(1), u^*(1)) - f(\hat{x}(1), u^*(1))\|}_{=\|w(\hat{x}(1), u^*(1))\| \leq \mu} \\
&\quad + \underbrace{\|f(\hat{x}(1), u^*(1)) - f(\hat{x}(1) - \hat{d}(1), u^*(1))\|}_{\leq L_{fx} \|\hat{d}(1)\|} \\
&\leq \mu + L_{fx} \mu
\end{aligned}$$

By induction, again we have:

$$\|\hat{d}(j)\| = \|\hat{x}(j) - x^*(j)\| \leq \sum_{i=0}^{j-1} L_{fx}^i \mu. \quad (18)$$

Combining equations Equation 17 and Equation 18 and by letting $\bar{L}_f^i = \min(L_{\hat{f}x}^i, L_{fx}^i)$, we obtain:

$$\|\hat{d}(j)\| = \|\hat{x}(j) - x^*(j)\| \leq \sum_{i=0}^{j-1} \bar{L}_f^i \mu. \quad (19)$$

The following identity is used; $\forall \delta > 0$:

$$\|a + b\|_2^2 \leq \left(1 + \frac{1}{\delta}\right) \|a\|_2^2 + (1 + \delta) \|b\|_2^2.$$

Hence, we can write the cost over the predicted state as:

$$\begin{aligned}
&\hat{x}(j)^T Q \hat{x}(j) \\
&= \|Q^{1/2} \hat{x}(j)\|_2^2 = \|Q^{1/2}(x^*(j) + \hat{d}(j))\|_2^2 \\
&\leq (1 + \delta) \|Q^{1/2} x^*(j)\|_2^2 + \left(1 + \frac{1}{\delta}\right) \|Q^{1/2} \hat{d}(j)\|_2^2 \\
&\leq (1 + \delta) x^*(j)^T Q x^*(j) + \left(1 + \frac{1}{\delta}\right) \|Q\|_2 \|\hat{d}(j)\|_2^2.
\end{aligned} \quad (20)$$

From equations Equation 19 and Equation 20, $\forall j \in \{0, 1, \dots, N-1\}$, we obtain:

$$\begin{aligned}
&\hat{x}(j)^T Q \hat{x}(j) - x^*(j)^T Q x^*(j) \\
&\leq \underbrace{\delta x^*(j)^T Q x^*(j)}_{=\ell(x^*(j), 0)} + \left(1 + \frac{1}{\delta}\right) \|Q\|_2 \left(\sum_{i=0}^{j-1} \bar{L}_f^i \mu\right)^2 \\
&\leq \delta \ell(x^*(i), u^*(i)) + \left(1 + \frac{1}{\delta}\right) \|Q\|_2 \left(\sum_{i=0}^{j-1} \bar{L}_f^i \mu\right)^2.
\end{aligned} \quad (21)$$

Recall from Equation 4 that $V(x) \leq L_V \|x\|_2^2$. Proceeding as before, we can write the following for the terminal cost:

$$V(\hat{x}(N)) - V(x^*(N)) \leq \delta V(x^*(N)) + \left(1 + \frac{1}{\delta}\right) L_V \left(\sum_{i=0}^{N-1} \bar{L}_f^i \mu\right)^2. \quad (22)$$

The final part of the proof concerns the constraints cost term. Let the state constraints be defined as a set of inequalities:

$$\mathbb{X} = \{x \in \mathbb{R}^n : g(x) \leq 1\},$$

where g is a convex function. For the optimal solution, x^* , the violation of the constraint is represented through the slack variable:

$$s^* = s(x^*) = \frac{(g(x^*) - 1) + |g(x^*) - 1|}{2}.$$

Since the constraints are convex and compact, and they contain the origin, then at the optimal solution, x^* , we have that there exists a \mathcal{K}_∞ -function, $\bar{\eta}(r)$, such that:

$$\left| \ell_{\mathbb{X}}(s(x^* + \hat{d})) - \ell_{\mathbb{X}}(s(x^*)) \right| \leq \bar{\eta}(\|\hat{d}\|).$$

Using the above inequality and Equation 19, it follows that, $\forall j \in \{0, 1, \dots, N\}$:

$$\ell_{\mathbb{X}}(\hat{x}(j)) - \ell_{\mathbb{X}}(x^*(j)) \leq \bar{\eta} \left(\sum_{i=0}^{j-1} \bar{L}_f^i \mu \right) = \bar{\eta}_j. \quad (23)$$

By combining equations Equation 13, Equation 14, Equation 15, Equation 16, Equation 21, Equation 22 and Equation 23, we obtain the bound stated in the Theorem:

$$\begin{aligned} & \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}^*(x)] - \mathbf{E}_{x \in \mathcal{D}}[J_{V^*}^*(x)] \\ & \leq \frac{2\gamma^N \epsilon}{1 - \gamma^N} + \left(1 + \frac{1}{\delta}\right) \|Q\|_2 \sum_{i=0}^{N-1} \gamma^i \left(\sum_{j=0}^{i-1} \bar{L}_f^j \right)^2 \mu^2 \\ & \quad + \left(1 + \frac{1}{\delta}\right) \gamma^N \alpha L_V \left(\sum_{i=0}^{N-1} \bar{L}_f^i \right)^2 \mu^2 + \underbrace{\sum_{j=0}^N \bar{\eta}_j}_{=: \bar{\psi}(\mu)} \\ & \quad + \delta \mathbf{E}_{x \in \mathcal{D}}[J_{\text{MPC}}^*(x; f)]. \end{aligned}$$

□

PROOF OF THEOREM 1

We prove the following, extended version of the theorem.

Theorem 3. Stability and robustness Assume that $V(x)$ satisfies (5), with $\lambda \in [0, 1)$, $\mathbb{X}_T = \{0\}$. Then, for any horizon length $N \geq 1$ there exist a constant $\bar{\alpha} \geq 0$, a minimum discount factor $\bar{\gamma} \in (0, 1]$, and a model error bound $\bar{\mu}$ such that, if $\alpha \geq \bar{\alpha}$, $\mu \leq \bar{\mu}$ and $\gamma \geq \bar{\gamma}$, then, $\forall x(0) \in \mathcal{C}(\mathbb{X}_s)$

1. If $N = 1$, $\mu = 0$, then the system is asymptotically stable for any $\gamma > 0$, $\forall x(0) \in \Upsilon_{N, \gamma, \alpha}$.
2. If $N > 1$, $\mu = 0$, then the system reaches a set \mathbb{B}_γ that is included in \mathbb{X}_s . This set increases monotonically with decreasing discount factors, γ , $\forall x(0) \in \Upsilon_{N, \gamma, \alpha}$. $\gamma = 1 \Rightarrow \mathbb{B}_\gamma = \{0\}$.
3. If $N > 1$, $\mu = 0$, and once in \mathbb{X}_s we switch to the expert policy, then the system is asymptotically stable, $\forall x(0) \in \Upsilon_{N, \gamma, \alpha}$.
4. If $\alpha V(x)$ is the optimal value function for the discounted problem, $\mu = 0$, and if $\mathcal{C}(\mathbb{X}_s) = \mathbb{X}_s$, then the system is asymptotically stable, $\forall x(0) \in \Upsilon_{N, \gamma, \alpha}$.
5. If $\alpha V(x)$ is the optimal value function in \mathbb{X}_s for the problem, $\mu = 0$, and if $\mathcal{C}(\mathbb{X}_s) \neq \mathbb{X}_s$, then the system is asymptotically stable, $\forall x(0) \in \Upsilon_{N, \gamma, \alpha}$.
6. The MPC has a stability margin. If the MPC uses a surrogate model satisfying (8), with one-step error bound $\|w\|_2^2 < \bar{\mu}^2 = \frac{1-\lambda}{L_V L_{f_x}^{2N}} l_s$, then the system is Input-to-State (practically) Stable (ISpS) and there exists a set $\mathbb{B}_{N, \gamma, \mu} : x(t) \rightarrow \mathbb{B}_{N, \gamma, \mu}$, $\forall x(0) \in \beta \Upsilon_{N, \gamma, \alpha}$, $\beta \leq 1$.
7. If $\mu = 0$, then $\alpha \geq \bar{\alpha}$ implies that $\alpha V(x) \geq V^*(x)$, $\forall x \in \mathbb{X}_s$, where V^* is the optimal value function for the infinite horizon problem with cost (3) and subject to (2).

Proof. In order to prove point 1 in the theorem, we first use the standard arguments for the MPC without terminal constraint [Mayne et al. \(2000\)](#); [Limon et al. \(2003\)](#) in the undiscounted case. We then extend the results to the discounted case.

Nominal stability First, when an invariant set terminal constraint is used, which in our case corresponds to the condition $V(x(N)) \leq l_s$ with $\mathbb{X}_s \subseteq \mathbb{X}$, then [Mayne et al. \(2000\)](#) have provided conditions to prove stability by demonstrating that $J_{\text{MPC}}^*(x)$ is a Lyapunov function. These require the terminal cost to be a Lyapunov function that satisfies Equation 5. Hence, we start by looking for values of α such that $\alpha V(x)$ satisfies Equation 5. In other words, we wish to find an $\bar{\alpha}_1 \geq 1$ such that, for all $\alpha \geq \bar{\alpha}_1$ and for some policy K_0 (in our case, the demonstrator for V), the following condition holds:

$$\alpha V(f(x, K_0(x))) - \alpha V(x) \leq -\ell(x, K_0(x)). \quad (24)$$

Let us denote $x^+ = f(x, K_0(x))$ for brevity. We have, by assumption, that:

$$\alpha(V(x^+) - \lambda V(x)) \leq 0. \quad (25)$$

This implies that:

$$\begin{aligned} \alpha V(x^+) - \alpha V(x) + \alpha V(x) - \alpha \lambda V(x) &\leq 0, \\ \Rightarrow \alpha V(x^+) - \alpha V(x) &\leq -\alpha(1 - \lambda)V(x). \end{aligned} \quad (26)$$

Recall that the loss function satisfies $l_\ell \|x\|_2^2 \leq \ell(x, u)$. Since the MPC is solved using a sequence of convex quadratic programs, it is also Lipschitz [Bemporad et al. \(2000\)](#). Similarly, if K_0 is Lipschitz or (uniformly) continuous over the closed and bounded set \mathbb{U} , then since \mathbb{X} is also closed and bounded, there also exists a local upper bound for the loss function on this policy, namely, $\ell(x, K_0(x)) \leq L_\ell \|x\|_2^2$.

Further, recall from Equation 4 that $l_\ell \|x\|_2^2 \leq V(x)$. Using the above notions, we have:

$$\begin{aligned} \alpha V(x^+) - \alpha V(x) &\leq -\alpha(1 - \lambda)l_\ell \|x\|_2^2, \\ &= -\alpha(1 - \lambda)l_\ell \frac{L_\ell}{L_\ell} \|x\|_2^2 \\ &\leq -\frac{\alpha(1 - \lambda)l_\ell}{L_\ell} \ell(x, K_0(x)) \\ &= -\beta \ell(x, K_0(x)). \end{aligned} \quad (27)$$

To satisfy the above condition, solving for a $\beta \geq 1$ is sufficient. From Equations Equation 26 and Equation 27, it implies that:

$$\alpha \geq \frac{L_\ell}{l_\ell(1 - \lambda)} = \bar{\alpha}_1 \geq 1. \quad (28)$$

Now, the function $\alpha V(x)$ satisfies all the sufficient conditions stated by [Mayne et al. \(2000\)](#) for the stability of an MPC under the terminal constraint $\hat{x}(N) \in \mathbb{X}_s$ which is equivalent to $V(\hat{x}(N)) \leq l_s$, without discount (with $\gamma = 1$).

Since we do not wish to have such a terminal constraint, we wish for another lower bound $\hat{\alpha}_2 \geq 1$ such that, if $\alpha \geq \bar{\alpha}_2$, then $V(\hat{x}(N)) \leq l_s$ at the optimal solution. The computation of this $\bar{\alpha}_2$ has been outlined by [Limon et al. \(2009\)](#) for the undiscounted case. Since our constraints are closed, bounded and they contain the origin, our model and the MPC control law are both Lipschitz, we directly use the result from [Limon et al. \(2009\)](#) to compute $\bar{\alpha}_2$:

$$\bar{\alpha}_2 = \frac{\sum_{i=0}^{N-1} \ell(\tilde{x}(i), \tilde{u}(i)) - N d}{(1 - \rho)l_s} \quad (29)$$

where $\tilde{x}(i), \tilde{u}(i)$ represent a sub-optimal state-action sequence for which $V(\tilde{x}(N)) \leq \rho l_s$ with $\rho \in [0, 1)$, and d is a lower bound for the stage loss ℓ for all x outside \mathbb{X}_s and all u in \mathbb{U} .

Then, one can take:

$$\alpha \geq \max(\bar{\alpha}_1, \bar{\alpha}_2) = \bar{\alpha} \quad (30)$$

to guarantee stability when $\gamma = 1$.

When the discount factor ($\gamma < 1$) is used, condition Equation 24 is still respected by the same range of α since

$$\gamma V(x^+) - V(x) \leq V(x^+) - V(x). \quad (31)$$

However, from the discussion in Gaitsgory et al. (2015), for infinite horizon optimal control, it appears that Equation 24 is not sufficient for $J_{\text{MPC}}^*(x)$ to be a Lyapunov function, even when a terminal constraint is used.

We wish to find a lower-bound $\bar{\gamma}$ such that, given α satisfying Equation 30, the MPC is stable for $\gamma \geq \bar{\gamma}$. For infinite-horizon optimal control, this was done by Gaitsgory et al. (2015). First, recall that:

$$\begin{aligned} \alpha V(x) &\leq \alpha L_V \|x\|_2^2 \leq \alpha \frac{L_V}{l_\ell} \ell(x, 0) \\ &= C \inf_{u \in \mathbb{U}} \ell(x, u). \end{aligned} \quad (32)$$

In Gaitsgory et al. (2015), it is shown that $1 \leq C < 1/(1 - \gamma)$ is sufficient for stability of an infinite-horizon discounted optimal control problem, when $\alpha V(x)$ is its value function. This means that:

$$\frac{\alpha L_V}{l_\ell} < \frac{1}{1 - \gamma}, \quad (33)$$

which implies that:

$$\gamma > 1 - \frac{l_\ell}{\alpha L_V} = \bar{\gamma}_1 \in [0, 1). \quad (34)$$

For MPC, we will instead present an additional condition to the above one that leads to at least convergence to the safe set. This results in a bounded and safe solution. Exact convergence to the origin will be then confirmed when V is the actual value function, as in Gaitsgory et al. (2015), or if we switch to the demonstrating policy, K_0 , once in the terminal set. Finally, we will remove the terminal constraint as done for the undiscounted case with a final bound on α and γ .

Recall that condition (24) applies. If the terminal constraint was met at time t , namely, if $V(x^*(N)) \leq l_s$, then at the next time step, $t + 1$ we have that $u(N + 1) = K_0(x^*(N))$ is feasible. Hence, the optimal MPC solution can be upper-bounded by the shifted solution at the previous time t , with the K_0 policy appended at the end of the horizon Mayne et al. (2000). Denote this policy as \tilde{u} and \tilde{x} as the predictions. We have that:

$$\begin{aligned} \Delta J_{\text{MPC}}^*(x) &= J_{\text{MPC}}^*(x^+) - J_{\text{MPC}}^*(x) \\ &\leq J_{\text{MPC}}^*(x^+, \tilde{u}) - J_{\text{MPC}}^*(x). \end{aligned}$$

Hence,

$$\begin{aligned} \Delta J_{\text{MPC}}^*(x) &\leq \sum_{i=1}^N \gamma^{i-1} \ell(\tilde{x}(i), \tilde{u}(i)) + \gamma^N \alpha V(\tilde{x}^+(N)) - \ell(x, \tilde{u}(0)) - \sum_{i=1}^{N-1} \gamma^i \ell(\tilde{x}(i), \tilde{u}(i)) \\ &\quad - \gamma^N \alpha V(\tilde{x}(N)) \\ &= (1 - \gamma) L_{N-1}(x) - \ell(x, \tilde{u}(0)) \\ &\quad + \underbrace{\gamma^{N-1} (\gamma \alpha V(\tilde{x}^+(N)) - \gamma \alpha V(\tilde{x}(N)) + \ell(\tilde{x}(N), K_0(x)))}_{\leq 0 \Leftarrow \gamma \alpha \geq \bar{\alpha}_1} \\ &\leq (1 - \gamma) L_{N-1}(x) - \ell(x, \tilde{u}(0)), \end{aligned}$$

where $L_{N-1}(x) = \sum_{i=1}^{N-1} \gamma^{i-1} \ell(\tilde{x}(i), \tilde{u}(i))$ and we have taken α such that $\gamma \alpha \geq \bar{\alpha}_1$.

Now, for $\gamma = 1$, the effect of L_{N-1} disappears and the MPC optimal cost is a Lyapunov function as in the standard MPC stability result from Mayne et al. (2000). By inspection of L_{N-1} , since the cost is bounded over bounded sets, also a small enough γ could be found such that $L_{N-1}(x) < \ell(x, \tilde{u}(0))$. This γ , however, depends on x . Consider $x \notin \mathbb{X}_s$, for which there exist a feasible solution, namely a solution providing $\tilde{x}(N) \in \mathbb{X}_s$. Then, since ℓ is strictly increasing, $\ell(0, 0) = 0$, \mathbb{X}_s contains the origin and the constraints are bounded, we have that there exist a $v \geq 1$ such that for any feasible x :

$$\bar{L}_{N-1} = v(N-1) \inf_{x \notin \mathbb{X}_s} \ell(x, 0),$$

is an upper bound for $L_{N-1}(x)$. For instance,

$$v = \frac{\sup_{(x,u) \in \epsilon \mathbb{X} \times \mathbb{U}} \ell(x, u)}{\inf_{x \notin \mathbb{X}_s} \ell(x, 0)},$$

is sufficient for any closed set of initial conditions $x(0) \in \epsilon \mathbb{X} \supset \mathbb{X}_s$, with $\epsilon > 0$. In order to have stability, it suffices to have $(1 - \gamma)\bar{L}_{N-1} - \ell(x, \tilde{u}(0)) \leq 0$ which requires:

$$\gamma \geq 1 - \frac{\ell(x, \tilde{u}(0))}{\bar{L}_{N-1}} = \bar{\gamma}(x). \quad (35)$$

In the above condition $\bar{\gamma}(x)$ can be less than 1 only outside a neighborhood of origin. Consider again

$$d = \inf_{x \notin \mathbb{X}_s} \ell(x, 0). \quad (36)$$

Then taking

$$\gamma \geq 1 - \frac{d}{\bar{L}_{N-1}} = \bar{\gamma}_2 \in (0, 1), \quad (37)$$

provides that the system trajectory will enter the safe set \mathbb{X}_s , hence $\mathbb{B}_\gamma \subseteq \mathbb{X}_s$. Finally, once $x \in \mathbb{X}_s$, we that the policy $K_0(x)$ is feasible and:

$$\ell(x, K_0(x)) \leq \alpha V(x) - \alpha V(x^+) \leq \alpha V(x).$$

Hence, we can use this policy to upper bound the MPC cost:

$$J_{MPC}^*(x) \leq \sum_{i=0}^{N-1} \gamma^i \alpha V(\tilde{x}(i)) + \gamma^N \alpha V(\tilde{x}(N)).$$

If the above is true with equality, then we can proceed as in Theorem 3.1 of [Gaitsgory et al. \(2015\)](#), with $\gamma > \bar{\gamma}_1$. This would require $\alpha V(x)$ to be also a value function for the discounted problem.

From the above considerations, we can conclude that that:

1. If $N = 1$, then $\bar{L}_{N-1} = 0$ and the system is asymptotically stable for any $\gamma > 0$.
2. If $N > 1$, $\gamma \geq \bar{\gamma}_2$, then the system reaches an bound \mathbb{B}_γ that is included in \mathbb{X}_s .
3. If $N > 1$ $\gamma \geq \bar{\gamma}_2$ and once in \mathbb{X}_s we switch to the policy $K_0(x)$ then the system is asymptotically stable.
4. If $\alpha V(x)$ is the global value function for the discounted problem and if $\mathcal{R}(\mathbb{X}_s) = \mathbb{X}_s$, then $\gamma > \bar{\gamma}_1$ provides that the system is Asymptotically stable.
5. If $\alpha V(x)$ is only the value function in \mathbb{X}_s for the discounted problem, and if $\mathcal{R}(\mathbb{X}_s) \neq \mathbb{X}_s$, then $\gamma > \max(\bar{\gamma}_1, \bar{\gamma}_2)$ provides that the system is Asymptotically stable.

Finally, following Theorem 3 from [Limon et al. \(2003\)](#), the terminal constraint can be removed for all points $x \in \mathcal{R}^N(\rho \mathbb{X}_s)$, with $\rho \in [0, 1)$, by setting:

$$\alpha \geq \bar{\alpha} = \max(\bar{\alpha}_1/\gamma, \bar{\alpha}_3), \quad (38)$$

$$\bar{\alpha}_3 = \frac{\bar{L}_N - \frac{1-\gamma^N}{1-\gamma} d}{(1-\rho)\gamma^N l_s} > 0. \quad (39)$$

In fact, by the same argument of [Limon et al. \(2003\)](#), for any states for which it exists a feasible sequence $\underline{\tilde{u}}$ taking $\hat{x}(N)$ to $\rho \mathbb{X}_s$ we have that:

$$J_{MPC}^*(x) \leq J_{MPC}(x, \underline{\tilde{u}}) = L_N(x) + \rho \alpha \gamma^N l_s. \quad (40)$$

If α satisfies (38), then we also have that:

$$(1-\rho)\alpha \gamma^N l_s \geq L_N(x) - \frac{1-\gamma^N}{1-\gamma} d, \quad (41)$$

from which we can directly verify that the set defined in (12) is a ROA (for either asymptotic or practical stability):

$$\Upsilon_{N,\gamma,\alpha} = \left\{ x \in \mathbb{R}^{n_x} : J_{MPC}^*(x) \leq \frac{1-\gamma^N}{1-\gamma} d + \gamma^N \alpha l_s \right\}.$$

Robustness For point 6, the stability margins of nominal MPC have been studied in [Limon et al. \(2009\)](#). In particular, in a setup without terminal constraint, under nominal stabilising conditions, with a uniformly continuous model (in our case even Lipschitz), cost functions being also uniformly continuous, then the optimal MPC cost is also uniformly continuous ([Limon et al., 2009](#), Proposition 1). In other words, from ([Limon et al., 2009](#), Theorem 1), there is a \mathcal{K}_∞ -function, σ , such that at the optimal solution, \underline{u}^* , we have:

$$|J_{MPC}^*(x+w) - J_{MPC}^*(x)| \leq \sigma(\|w\|). \quad (42)$$

Using the above identity, one wish to bound the increase in the MPC cost due to uncertainty. At the same time, we wish the MPC to remain feasible and perform a contraction, namely, to have a stability margin. Since we are using soft constraints, then the MPC remains always feasible, however, we need the predictions at the end of the horizon to be in an invariant set \mathbb{X}_s even under the effect of uncertainty. In particular, we will use $V(x)$ and its contraction factor λ to compute a smaller level set $\zeta\mathbb{X}_s$, for some $\zeta \in (0, 1)$ which is invariant under the uncertainty. Once this is found then we can compute a new α for this set according to (38). In particular, under the policy K_0 , we have that:

$$\begin{aligned} V(x^+ + w) - V(x) &\leq V(x^+) - V(x) + L_V\|w\|_2^2 \\ &\leq (\lambda - 1)V(x) + L_V\|w\|_2^2. \end{aligned}$$

We wish this quantity to be non-positive for $x \notin \zeta\mathbb{X}_s$, which means that $V(x) \geq \zeta l_s$. For this it is sufficient to have:

$$\|w\|_2^2 \leq \frac{1-\lambda}{L_V}\zeta l_s \leq \frac{1-\lambda}{L_V}V(x) \quad (43)$$

Since we apply the function V to the prediction at time N , and at the next step the measured state (prediction at index 0) differs from the previous MPC prediction of a quantity w , for the MPC this condition translates into:

$$\|w\|_2^2 \leq \frac{1-\lambda}{L_V L_{f_x}^{2N}}\zeta l_s < \bar{\mu}^2 = \frac{1-\lambda}{L_V L_{f_x}^{2N}}l_s, \quad (44)$$

Therefore, given the model error w , if the MPC remains feasible and if α and γ exceed their lower bounds given the restricted set $\zeta\mathbb{X}_s$, we have that $\|w\|_2 < \bar{\mu}$ implies that:

$$\begin{aligned} \Delta J_{MPC}^*(x) &= J_{MPC}^*(x^+ + w) - J_{MPC}^*(x) + J_{MPC}^*(x^+) - J_{MPC}^*(x^+) \\ &\leq J_{MPC}(x^+, \tilde{u}) - J_{MPC}^*(x) + \sigma(\|w\|) \\ &\leq (1-\gamma)\bar{L}_{N-1} - \ell(x, \tilde{u}(0)) + \sigma(\|w\|) \\ &\leq -l_\ell\|x\|_2^2 + \sigma(\|w\|) + \bar{d}(N), \end{aligned}$$

which is the definition of Input-to-State practical Stability [Khalil \(2014\)](#); [Limon et al. \(2009\)](#), where we have defined $\bar{d}(N) = (1-\gamma)\bar{L}_{N-1}$. The trajectory of the system is therefore bounded by the level-set of $J_{MPC}^*(x)$ outside which $\sigma(\mu) + \bar{d}(N) \leq l_\ell\|x\|_2^2$. Since σ is strictly increasing and \bar{d} is strictly decreasing, we can also conclude that the size of this bound increases with increasing model error μ and with the horizon length N . Note that the term \bar{d} vanishes if $\gamma = 1$ but the term σ will also increase with N if $\bar{L}_f > 1$. From the restriction of the terminal set, it also follows that the ROA defined in Equation 12 will also be restricted unless we recompute a larger α for this new set.

Value function upper bound Denote $\hat{V}(x) = \alpha V(x)$. Recall that, if $\alpha \geq \alpha_1$, as defined in Equation 28, and if $\|w(t)\| = 0 \forall t$, then we have that, $\forall x(t) \in \mathbb{X}_s$:

$$\hat{V}(x(t)) = \alpha V(x(t)) \geq \ell(x(t), K_0(t)) + \alpha V(x(t+1)), \quad (45)$$

which in turn implies that, by means of $\gamma \leq 1$ and of induction:

$$\hat{V}(x(t)) \geq \ell(x(t), K_0(t)) + \gamma \hat{V}(x(t+1)) \quad (46)$$

$$\geq \ell(x(t), K_0(t)) + \gamma \left(\ell(x(t+1), K_0(t+1)) + \gamma \hat{V}(x(t+2)) \right) \quad (47)$$

$$\geq \sum_{i=0}^{\infty} \gamma^i \ell(x(t+i), K_0(t+i)) = V_{K_0}(x(t)), \quad (48)$$

which is the value of the policy K_0 . This is, by definition, an upper bound of the optimal value function, $V^*(x)$. Hence $\alpha V(x) \geq V^*(x), \forall x \in \mathbb{X}_s$.

□

In practice, the ISS bound $\sigma(\mu)$ from Theorem 1 has a form similar to the one discussed for the constraint penalty in the proof of Theorem 2, see Equation 23. Its explicit computation is omitted for brevity; however, in general, we can expect the bound to become worse for systems that are open-loop unstable as the horizon length increases.

B MPC AS SQP FORMULATION

We solve the MPC problem through iterative linearisations of the forward model, which gives the state and input matrices:

$$A(i) = \left. \frac{\partial \hat{f}}{\partial x} \right|_{\bar{x}(i)}, \quad B(i) = \left. \frac{\partial \hat{f}}{\partial u} \right|_{\bar{u}(i)}. \quad (49)$$

These are evaluated around a reference trajectory:

$$\underline{x}^* = \{\bar{x}^*(i), i = 0, \dots, N\}, \quad (50)$$

$$\underline{u}^* = \{\bar{u}^*(i), i = 0, \dots, N-1\}. \quad (51)$$

This is initialised by running the forward model with zero inputs, and then updated at each iteration by simulating the forward model on the current optimal solution.

The Lyapunov function V is expanded to a second order term by using Taylor expansion and is evaluated around the same trajectory. The Jacobian and Hessian matrices, respectively, Γ and H , are:

$$\Gamma = \left. \frac{\partial V}{\partial x} \right|_{\bar{x}(N)}, \quad H = \frac{1}{2} \left. \frac{\partial^2 V}{\partial^2 x} \right|_{\bar{x}(N)}. \quad (52)$$

All the quantities in Equation 49 and Equation 52 are computed using automatic differentiation. Using these matrices, we solve the convex optimization problem:

$$\begin{aligned} \delta \underline{u}^* = \arg \min \quad & \gamma^N \alpha \left(\|H^{1/2} \delta \hat{x}(N)\|_2^2 + \Gamma^T \delta \hat{x}(N) \right) + \sum_{i=0}^{N-1} \gamma^i \ell(\hat{x}(i), \hat{u}(i)) \quad (53) \\ \text{s.t.} \quad & \hat{x}(i+1) = A(i) \delta \hat{x}(i) + B(i) \delta \hat{u}(i) + \hat{f}(\bar{x}(i)), \\ & \hat{x}(i) - \delta \hat{x}(i) = \bar{x}(i) \\ & \hat{u}(i) - \delta \hat{u}(i) = \bar{u}(i) \\ & \hat{x}(i) \in \mathbb{X}, \forall i \in [0, N], \\ & \hat{u}(i) \in \mathbb{U}, \forall i \in [0, N-1], \\ & \hat{x}(0) = x(t), \\ & \|\delta \hat{u}(i)\|_\infty \leq r_{\text{trust}}, \forall i \in [0, N-1], \end{aligned}$$

where the state constraints are again softened and the last inequality constraint is used to impose a trust region with a fixed radius, r_{trust} . This notably improves the search for an optimal solution, as shown for the inverted pendulum case in Figure 13.

Once problem Equation 53 is solved, we obtain the delta sequence $\delta \underline{u}^*$. The new optimal solution is then computed according to the update rule:

$$\underline{u}^* \leftarrow \underline{u}^* + lr \delta \underline{u}^*,$$

where $lr < 1$ is a learning rate, which is annealed after each iteration. Finally, the reference trajectories used for the next linearisation are \underline{u}^* and the state series resulting from simulating the forward model on \underline{u}^* , namely

$$\underline{x}^* = \hat{f} \circ \underline{u}^*.$$

This is summarised in Algorithm 2. Interested readers can find more details on SQP in the work by Nocedal & Wright (2006); Hadfield-Menell et al. (2016), where adaptive schemes for computing the trust radius are discussed.

Algorithm 2 Neural Lyapunov MPC solver

In: $x(t), \hat{f}, \alpha, V, N_{\text{steps}}, \epsilon_{lr} \in [0, 1], r_{\text{trust}} > 0, lr$

Out: $u(t)$

```

 $\underline{x}^* \leftarrow \{x(t)\}^{N+1}, \quad \underline{u}^* \leftarrow \{0\}^N$ 
for  $j = 0, \dots, N_{\text{steps}}$  do
     $\{A(i)\}, \{B(i)\} \leftarrow$  linearisation of  $\hat{f}$  using Equation 49
     $(\Gamma, H) \leftarrow$  Taylor expansion of  $V$  using Equation 52
     $\delta \underline{u}^* \leftarrow$  solution of optimization problem Equation 53
     $\underline{u}^* \leftarrow \underline{u}^* + lr \delta \underline{u}^*$ 
     $\underline{x}^* \leftarrow \hat{f} \circ \underline{u}^*$ 
     $lr \leftarrow (1 - \epsilon_{lr}) lr$ 
 $u(t) \leftarrow \underline{u}^*(0)$ 

```

C EXPERIMENTAL SETUP

We demonstrate our approach on an inverted pendulum and vehicle kinematics control problem. We also provide additional results and plots in Appendix D.²

C.1 IMPLEMENTATION SPECIFICS

A practical consideration for implementing Algorithm 1 is tuning the MPC terminal cost scaling, α , via grid-search. The MPC needs to run over the entire dataset of initial points, $\mathcal{X}_0 = \{x_m(0)\}_{m=1}^M$, with different configurations. In order to speed up the search for α , we run the MPC only on a sample of 20% of the initial dataset. Once the optimal α is found, only then we run the MPC over the entire dataset and use this data to compute the next $V(x)$.

Additionally to what presented in the main text, we parameterize $V(x)$ with a trainable scaling factor, β , as:

$$V(x) = \text{softplus}(\beta)x^T (l_\ell I + V_{\text{net}}(x)^T V_{\text{net}}(x)) x, \quad (54)$$

where $\text{softplus}(x) = \log(1 + \exp(x))$. The parameter β is initialized to 1 in all experiments except for the inverted pendulum without LQR loss, i.e. for results in Figure 12.

The full set of parameters for the experiments can be found in Table 5, ??.

C.2 BASELINE CONTROLLERS

Our Neural Lyapunov MPC has a single-step horizon and uses the learned Lyapunov function as the terminal cost. To compare its performance, we consider two other MPCs and RL baselines:

- **Long-horizon MPC (demonstrator/demo):** An MPC with a longer horizon and a quadratic terminal cost $x^T P x$. This MPC is also used to generate the initial demonstrations for alternate learning.
- **Short-horizon MPC:** An MPC with a single-step horizon and the same terminal cost as the long-horizon MPC. All other parameters are the same as the Neural Lyapunov MPC except α , which is tuned manually.
- **Model-free RL:** Neural-network parameterized policy trained using on-policy algorithm, PPO (Schulman et al., 2017), and off-policy algorithm, SAC (Haarnoja et al., 2018).
- **Model-based RL:** Neural-network parameterized policy and ensemble of dynamic models trained using the algorithm, MBPO (Janner et al., 2019).

²Our code, including the world models, is implemented using PyTorch Paszke et al. (2019). This allows us to exploit automatic differentiation to linearise the dynamics for the SQP-based MPC.

Table 5: **Configuration for experiments in main paper.** We specify the parameters used for the simulation of the system dynamics, the demonstrator, the Neural Lyapunov MPC as well as for the alternate learning algorithm.

PARAMETER	SYMBOL	VALUE	
		CAR KINEMATICS	PENDULUM
GENERAL			
MASS	m	-	0.2 kg
LENGTH	l	-	0.5 m
ROTATIONAL FRICTION	λ_F	-	$0.1 \text{ N m s rad}^{-1}$
GRAVITY	g	-	9.81 ms^{-2}
SAMPLING TIME	dt	0.2 s	0.01 s
STATE CONSTRAINT	\mathbb{X}	$[-1, 1]^2 \times [-\pi, \pi]$	$[-\pi, \pi] \times [-2\pi, 2\pi]$
INPUT CONSTRAINT	\mathbb{U}	$[-10, 10] \times [-2\pi, 2\pi]$	$[-0.64, 0.64]$
STATE PENALTY	Q	$\text{DIAG}(1, 1, 0.001\pi)$	$\text{DIAG}(0.1, 0.1)$
INPUT PENALTY	R	$\text{DIAG}(100, 20\pi)$	0.1
DISCOUNT FACTOR	γ	1	1
DEMONSTRATOR			
TYPE	K_0	MPC	MPC
HORIZON	N	5	5
# OF LINEARISATIONS	N_{steps}	3	10
TRUST REGION	r_{trust}	∞	2.5
LEARNING RATE	lr	0.9	0.9
DECAY RATE	ϵ_{lr}	0.2	0.2
TERMINAL PENALTY	P	$400 Q$	$500 P_{\text{LQR}}$
NEURAL LYAPUNOV MPC			
HORIZON	N	1	1
# OF LINEARISATIONS	N_{steps}	18	18
TRUST REGION	r_{trust}	0.005	0.5
LEARNING RATE	lr	0.9	0.9
SCALING OF V	α	TABLE 8	TABLE 10
DECAY RATE	ϵ_{lr}	0.02	0.02
V_{net} ARCHITECTURE	MLP	(128, 128, 128)	(64, 64, 64)
V_{net} OUTPUT	$n_V \times n_x$	400×3	100×2
ALTERNATE LEARNING			
OUTER ITERATIONS	N_{ext}	3	2
ENLARGEMENT FACTOR	ϵ_{ext}	0.1	0.1
MPC LINE SEARCH	α_{list}	$\{1, 6, ..., 36\}$	$\{0.2, 0.4, ..., 2\}$
LYAPUNOV EPOCHS	N_V	500	200
LOSS EQUATION 10	ρ	0.0001	0.0001
CONTRACTION FACTOR	λ	0.99	0.99
LYAPUNOV LEARNING RATE	lr	0.001	0.001
LYAPUNOV WEIGHT DECAY	wd	0	0.002

In order to train the RL baselines, we consider two variations of our proposed models. In the first version (marked v1), the environment considers the stage cost used in demonstrator MDP as the penalty and terminates the episodes whenever the state constraints are violated. In the second version (marked v2), there is no environment termination. Instead an additional penalty term is added for violating the constraint. This is further described in Table 6. We show the results of training in these variants of the environments for SAC and PPO in Figure 4.

Table 6: **Reward and termination conditions for RL MDPs.** We train RL baselines on two variants of the inverted pendulum and car kinematics models. Note, for inverted pendulum: $\beta_1 = 10, \beta_2 = 200$, while car kinematics: $\beta_1 = 1, \beta_2 = 200$. The symbol $\mathbb{I}_{(\cdot)}$ is the indicator function.

VERSION	REWARD TERM	TERMINATION CONDITION
v1	$-\eta^T Q \eta - u^T R u$	$\eta \notin \mathbb{X}$
v2	$-\beta_1(\eta^T Q \eta - u^T R u) - \beta_2 \mathbb{I}_{\eta \notin \mathbb{X}}$	-

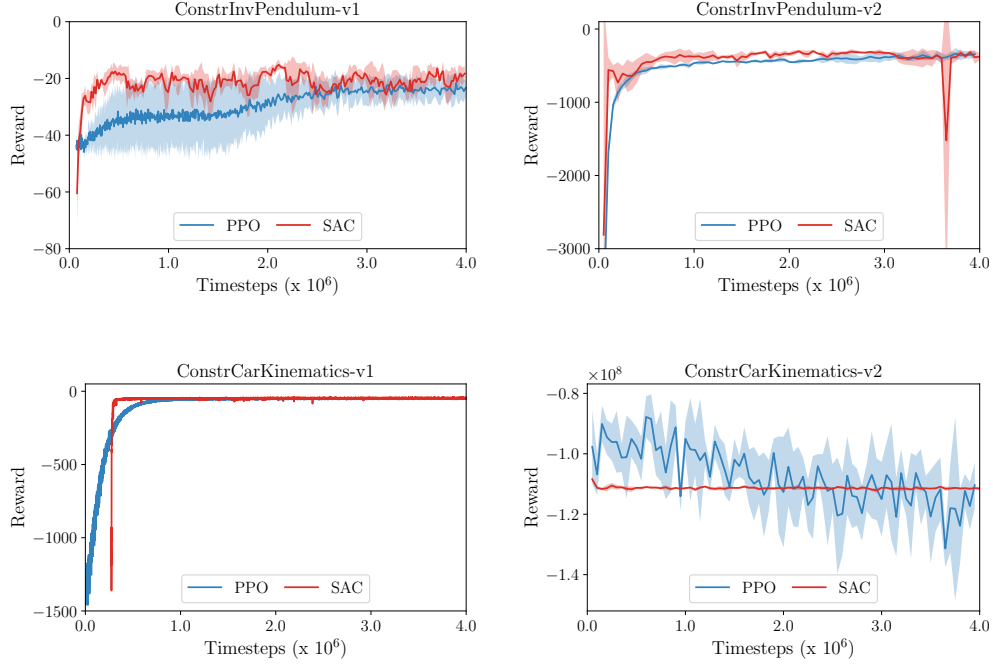


Figure 4: **Training of RL agents.** The plots show the learning curves for PPO and SAC in the two variants each of inverted pendulum and car kinematics environments. We train the agent on 4 different seeds for 4×10^6 timesteps. The solid line and filled region indicates the mean and one standard deviation reward across the seeds.

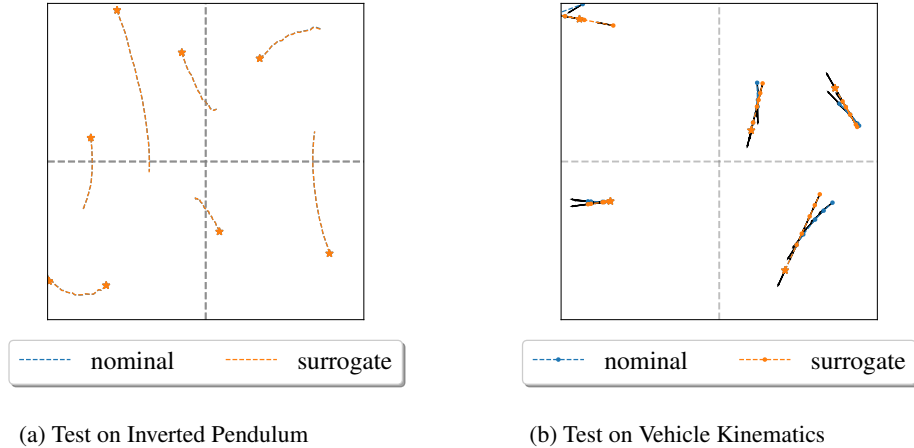


Figure 5: **Testing the surrogate models.** We generate each trajectory by propagating an initial state $\eta(0)$ with input sequence $\{u(t)\}_{t=0}^{T-1}$ through the nominal system and the learned surrogate model. (a) For the inverted pendulum: $u(t) \sim \mathcal{U}([-0.4, 0.4])$ and simulation is performed for $T = 25$. (b) For the vehicle kinematics: $u(t) \sim \mathcal{U}([-0.1, 0.1] \times [-0.1, 0.1])$ and simulation is performed for $T = 5$.

C.3 FORWARD MODELS

We use an Euler forward model for the environments. Consider dt as the sampling time, then the state transition is:

$$\eta(t+1) = \eta(t) + dt f_u(\eta(t), u(t)), \quad (55)$$

where $\eta(t)$ is the state, $u(t)$ is the input and $f_u(\eta(t), u(t))$ is the time-invariant, deterministic dynamical system.

C.3.1 VEHICLE KINEMATICS

World model For the non-holonomic vehicle, $\eta = (x, y, \phi) \in \mathbb{R}^3$ is the state, respectively, the coordinates in the plane and the vehicle orientation, and $u = (v, \omega) \in \mathbb{R}^2$ is the control input, respectively, the linear and angular velocity in the body frame. $f_u(\eta, u)$ encodes the coordinate transformation from the body to the world frame [Fossen \(2011\)](#):

$$f_u(\eta(t), u(t)) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} v(t) \cos \phi(t) \\ v(t) \sin \phi(t) \\ \omega(t) \end{pmatrix} = \underbrace{\begin{pmatrix} \cos \phi(t) & 0 \\ \sin \phi(t) & 0 \\ 0 & 1 \end{pmatrix}}_{J(\eta)} \begin{pmatrix} v(t) \\ \omega(t) \end{pmatrix} \quad (56)$$

Surrogate model We build a gray-box model using a neural network to model $J(\eta)$, similar to the work by [Quaglino et al. \(2020\)](#). The input feature to the network is $\sin \phi$ and $\cos \phi$, where ϕ is the vehicle’s orientation. The network consists of a hidden layer with 20 hidden units and tanh activation function, and an output layer without any activation function. The weights in the network are initialized using Xavier initialization [Glorot & Bengio \(2010\)](#) and biases are initialized from a standard normal distribution.

Training the surrogate model We generate a dataset of $10K$ sequences, each of length $T = 1$. For each sequence, the initial state $\eta(0)$ is sampled uniformly from \mathbb{X} , while the input $u(0)$ is sampled uniformly from \mathbb{U} . We use a training and validation split of 7 : 3. Training is performed for 300 epochs using the Adam optimizer [Kingma & Ba \(2014\)](#) and the mean-squared error (MSE) loss over the predicted states. The learning rate is 0.01 and the batch size is 700.

C.3.2 INVERTED PENDULUM

World model Inverted pendulum is one of the most standard non-linear systems for testing control methods. We consider the following model [Berkenkamp et al. \(2017\)](#):

$$ml^2\ddot{\theta} = mgl \sin \theta - \lambda_F \dot{\theta} + u \quad (57)$$

where $\theta \in \mathbb{R}$ is the angle, m and l are the mass and pole length, respectively, g is gravitational acceleration, λ_F is the rotational friction coefficient and $u \in \mathbb{R}$ is the control input. We denote the state of the system as $\eta = (\theta, \dot{\theta}) \in \mathbb{X} \subset \mathbb{R}^2$ and input as $u \in \mathbb{U} \subset \mathbb{R}$. We use an Euler discretization, as in Equation 55, to solve the initial-value problem (IVP) associated with the following equation:

$$f_u(\eta(t), u(t)) = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta}(t) \\ \frac{mgl \sin \theta(t) + u(t) - \lambda_F \dot{\theta}(t)}{ml^2} \end{pmatrix} \quad (58)$$

Surrogate model We use a neural network to predict the acceleration of the pendulum, $\ddot{\theta}(t)$. The input to the network is the state $\eta(t)$ and action $u(t)$ at the current time-step t . The network is a three layer feedforward network with 64 hidden units and tanh activation in each hidden layer. The output layer is linear. All the layers have no biases and their weights are initialized as in [Glorot & Bengio \(2010\)](#).

Training the surrogate model To train the surrogate model, we generate a dataset of $10K$ sequences, each of length $T = 1$. We use MSE loss and Adam optimizer [Kingma & Ba \(2014\)](#) to train the model. The rest of the parameters are kept the same as those used for the vehicle kinematics model training.

D ADDITIONAL RESULTS

Here we provide additional results and plots related to the experiments specified in the paper.

D.1 VEHICLE KINEMATICS

The vehicle model results are run over different machines. The resulting losses and α are discussed. In all experiments, the parameters of V are reinitialised after every outer epoch. The Lyapunov loss does not make use of the LQR loss ℓ .

Choosing number of outer iterations N_{ext} We run our algorithm for more iterations on a machine with different operating system. This leads to a slight difference from the results mentioned in the paper. As observed from Table 7, the third iteration leads to the best performance in terms of verified and not verified percentages. We set $N_{ext} = 3$ in all experiments.

Table 7: **Car Kinematics: Choosing number of outer iterations.** Lyapunov Function learning performed on a different machine (results are slightly different from the ones in the paper).

ITER.	LOSS ($\log(1+x)$)	VERIFIED (%)	NOT VERIFIED (%)
1	1.42	92.83	3.95
2	0.91	93.08	4.71
3	0.62	94.26	3.89
4	0.46	93.65	4.38
5	0.53	92.18	5.63

Alternate learning on nominal model We train the Neural Lyapunov MPC while using a nominal model for internal dynamics. In Figure 6, we plot the training curves, the resulting Lyapunov function, and line-search for MPC in each outer epoch. The results are also shown in Table 8. As can be seen, tuning the MPC parameter α helps in minimizing the loss Equation 10 further. Points near the origin don’t always verify. The MPC obtained after the third iteration achieves the best performance. This can further be validated from Figure 7, where we plot trajectories obtained by using the Neural Lyapunov MPC from each iteration.

Alternate learning on surrogate model In order to test the transfer capability of the approach, we perform the training of Neural Lyapunov MPC using an inaccurate surrogate model for the internal dynamics. This model is also used for calculating the Lyapunov loss Equation 10 and evaluating the performance of the Lyapunov function. We plot the training curves, the resulting Lyapunov function, and line-search for MPC in each outer epoch in Figure 3. The results of the training procedure are presented in Table 9. The MPC obtained from the second iteration achieves the best performance. In the third iteration, the Lyapunov loss increases and number of verified and not verified points becomes worse. The poor performance also reflects in the trajectories obtained by using the Lyapunov MPC from the third iteration, as shown in Figure 8.

Table 8: **Car Kinematics: Learning on nominal model.** Results for training Neural Lyapunov MPC while using a nominal model for internal dynamics. We use the Lyapunov loss Equation 10 for both learning the Lyapunov function and tuning the MPC. This is specified in the $\log(1+x)$ scale.

(a) Lyapunov Function Learning			
ITER.	LOSS ($\log(1+x)$)	VERIFIED (%)	NOT VERIFIED (%)
1	1.55	92.20	4.42
2	0.87	93.17	4.89
3	0.48	94.87	3.89

(b) MPC Parameter Tuning			
ITER.	LOSS ($\log(1+x)$) BEFORE	PARAMETER AFTER α^*	
1	1.55	1.07	26.00
2	0.87	0.71	31.00
3	0.48	0.52	36.00

Table 9: **Car Kinematics: Learning on surrogate model.** Results for training Neural Lyapunov MPC while using the surrogate model for internal dynamics as well as in Lyapunov training. We use the Lyapunov loss Equation 10 for both learning the Lyapunov function and tuning the MPC. This is specified in the $\log(1+x)$ scale.

(a) Lyapunov Function Learning			
ITER.	LOSS ($\log(1+x)$)	VERIFIED (%)	NOT VERIFIED (%)
1	1.84	91.74	8.26
2	1.43	92.26	7.74
3	1.65	91.61	8.39

(b) MPC Parameter Tuning			
ITER.	LOSS ($\log(1+x)$) BEFORE	PARAMETER AFTER α^*	
1	1.84	1.41	36.00
2	1.43	1.05	36.00
3	1.65	1.30	36.00

D.2 INVERTED PENDULUM

Differently from the car kinematics, for the inverted pendulum task, the parameters of V are not re-initialized after every outer epoch, and the Lyapunov loss makes use of the LQR loss, ℓ , for all the experiments except for the results in Figure 12. In this section, we discuss the results obtained from the alternate learning on the nominal model. We also provide an ablation study on: 1) a solution based solely on a contraction factor, and 2) the effect of having an imperfect solver, in particular the instability resulting from wrongly tuning the trust-region radius.

Alternate learning Since the trained surrogate model has a high accuracy, we only consider the scenario for alternate learning with the nominal model. The main results for this scenario are in Figure 9 and Table 10. We notice a slight improvement in the MPC’s performance in the second iteration of the training procedure. In Figure 9, it can be noticed that a small α needs to be used, which contradicts the ideal theoretical result. In practice, a very large value of this parameter results in bad conditioning for the QPs used by the MPC and causes the solver to fail.³ Since the pendulum

³When the solver fails, we simply set the solution to zero.

Table 10: **Inverted Pendulum: Learning on nominal model.** Results for training Neural Lyapunov MPC while using a nominal model for internal dynamics. We use the Lyapunov loss Equation 10 for both learning the Lyapunov function and tuning the MPC. This is specified in the $\log(1+x)$ scale.

(a) Lyapunov Function Learning			
ITER.	LOSS ($\log(1+x)$)	VERIFIED (%)	NOT VERIFIED (%)
1	3.21	13.25	0.00
2	1.08	13.54	0.00

(b) MPC Parameter Tuning			
ITER.	LOSS ($\log(1+x)$) BEFORE	AFTER	PARAMETER α^*
1	3.21	2.47	1.40
2	1.08	1.28	1.00

is open-loop unstable, an increase of the Lyapunov loss can be noticed for larger values of α . This demonstrates that it is necessary to perform a search over the parameter and that we cannot simply set it to a very large value.

In Figure 10, we show the trajectories obtained by running the Neural Lyapunov MPC obtained from the first and second iterations. The initial states are sampled inside the safe level-set by using rejection sampling. The trajectories obtained from both the iterations are similar even though the Lyapunov function is different. The Lyapunov function from second iteration has a larger ROA.

We also compare the Neural Lyapunov MPC from the second iteration with the baseline MPCs in Figure 11. The baselines controllers behave quite similarly in this problem, although they have different prediction horizons. This is because, for both of them, the LQR terminal cost is a dominating term in the optimization’s objective function. The Neural Lyapunov MPC achieves a slightly slower decrease rate compared to the demonstrator; however, it still stabilizes the system. The transfer from nominal to surrogate model is very successful for all the controllers, though in this case, the surrogate model is particularly accurate.

It should be kept in mind that in order to produce these results, it was necessary to impose in the Lyapunov loss Equation 10 that the decrease rate of $V(x)$ needs to be upper bounded by the LQR stage loss, as in Equation 5. This resulted in the most effective learning of the function $V(x)$, contrarily to the vehicle kinematics example.

Alternate learning without LQR loss We now consider the case when the learning is performed while using a contraction factor of $\lambda = 0.9$ and without the LQR loss term in the Lyapunov loss (i.e., $v = 0$). The results are depicted in Figure 12. In order to obtain these results, the Lyapunov NN scaling β , in Equation (54), was initialized with:

$$\beta_0 = \text{softplus}^{-1}(25),$$

according to a rough estimate of the minimum scaling α from Equation 28. This was able to produce a Lyapunov function that makes the MPC safe with $\alpha = 12$. However, the learning becomes more difficult, and it results in a smaller safe region estimate with a slower convergence rate for the system trajectories.

Effects of the trust region In Figure 13, we show the result of varying the trust radius of the SQP solver on the inverted pendulum. While a larger value can result in further approximation, given the limited number of iterations, in this case a small value of the radius results in a weaker control signal and local instability near the origin.

E PROBABILISTIC SAFETY VERIFICATION

A probabilistic verification is used to numerically prove the physical system stability with high probability. The resulting certificate is of the form $\mathcal{P}(\mathbb{X}_s \setminus \mathbb{X}_T) \geq \epsilon_P$, where the ϵ_P increases with increasing number of samples. Following the work of Bobiti (2017), the simulation is evaluated at a large set of points within the estimated safe set \mathbb{X}_s , while we search for an annulus where the 24 is verified. Monte Carlo rejection sampling is performed with PyMC (Salvatier et al., 2016).

In practical applications, several factors limit the convergence of the trajectory to a neighborhood of the target (the *ultimate bound*, Blanchini & Miani (2007)). For instance, the policy structural bias, discount factors in RL methods or persistent uncertainty in the model, the state estimates, and the physical system itself. Therefore, we extended the verification algorithm of (Bobiti, 2017) to estimate the ultimate bound as well as the invariant set, as outlined in Algorithm 3. Given a maximum and minimum level, l_l, l_u , we first sample initial states uniformly within these two levels and check for a robust decrease of V over the next state distribution. If this is verified, then we sample uniformly from inside the minimum level set l_l (where V may not decrease) and check that V does not exceed the maximum level l_u over the next state distribution. Note that Algorithm 3 is highly parallelizable.

Probabilistic verification is expensive but necessary, as pathological cases could result in the training loss (10) for which the safe set could be converging to a local minima with a very small set. Results can also vary where different random seeds are used. For these reasons, we also perform an informal verification over a fixed validation set and use this to select the best V from all training epochs.

Algorithm 3 Probabilistic safety verification

In: $N, V, K, \delta > 0$
Out: (SAFE, l_u, l_l)

 SAFE \leftarrow False **for** $l_u = 1, 1 - \delta, 1 - 2\delta, \dots, 0$ **do**
for $l_l = 0, \delta, 2\delta, \dots, l_u$ **do**

 draw N uniform x -samples s.t.:

 $l_l \leq V(x) \leq l_u$
if $V(\hat{x}^+) - V(x) \leq 0, \forall x$ **then**

 draw N uniform x -samples s.t.:

 $V(x) \leq l_l$
if $V(\hat{x}^+) \leq l_l, \forall x$ **then**

 SAFE \leftarrow True

return SAFE, l_u, l_l

 Verification failed.

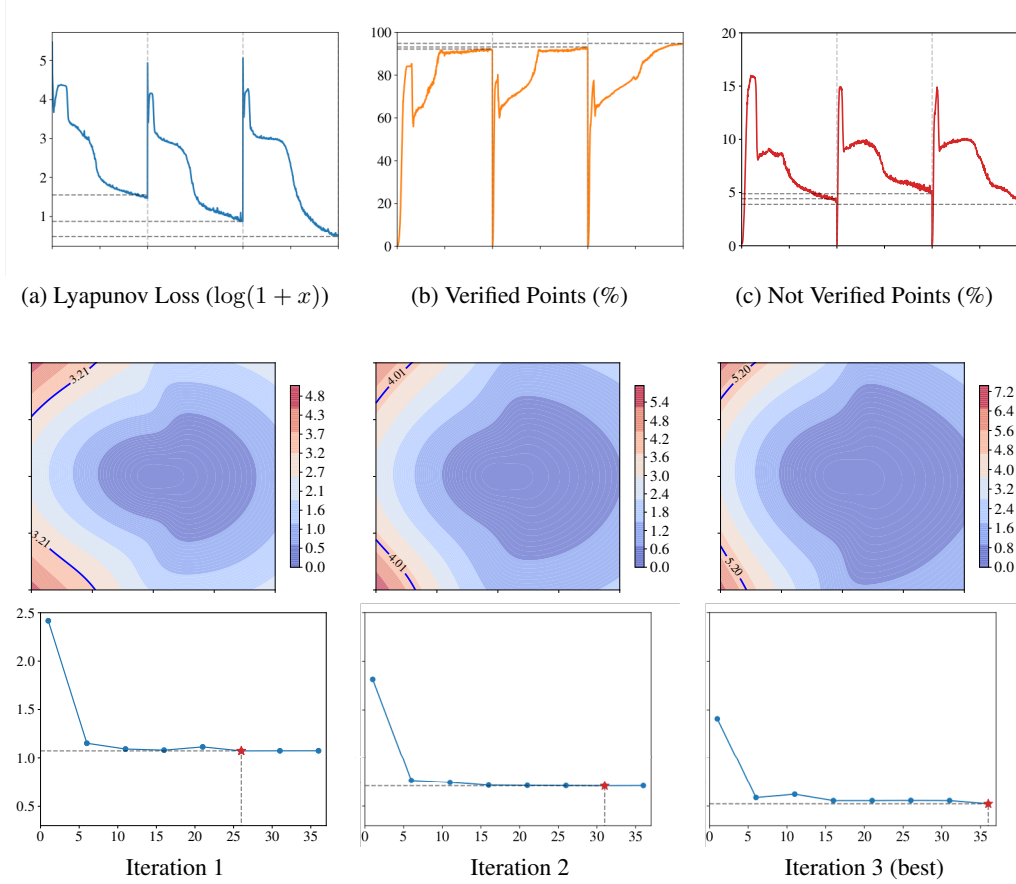


Figure 6: **Car kinematics: Alternate learning on nominal model.** After every $N_V = 500$ epochs of Lyapunov learning, the learned Lyapunov function is used to tune the MPC parameters. **Top:** The training curves for Lyapunov function. Vertical lines separate iterations. **Middle:** The resulting Lyapunov function V at $\phi = 0$ with the best performance. **Bottom:** Line-search for the MPC parameter α to minimize the Lyapunov loss (10) with V as terminal cost. The loss is plotted on the y-axis in a $\log(1+x)$ scale. The point marked in red is the parameter which minimizes the loss.

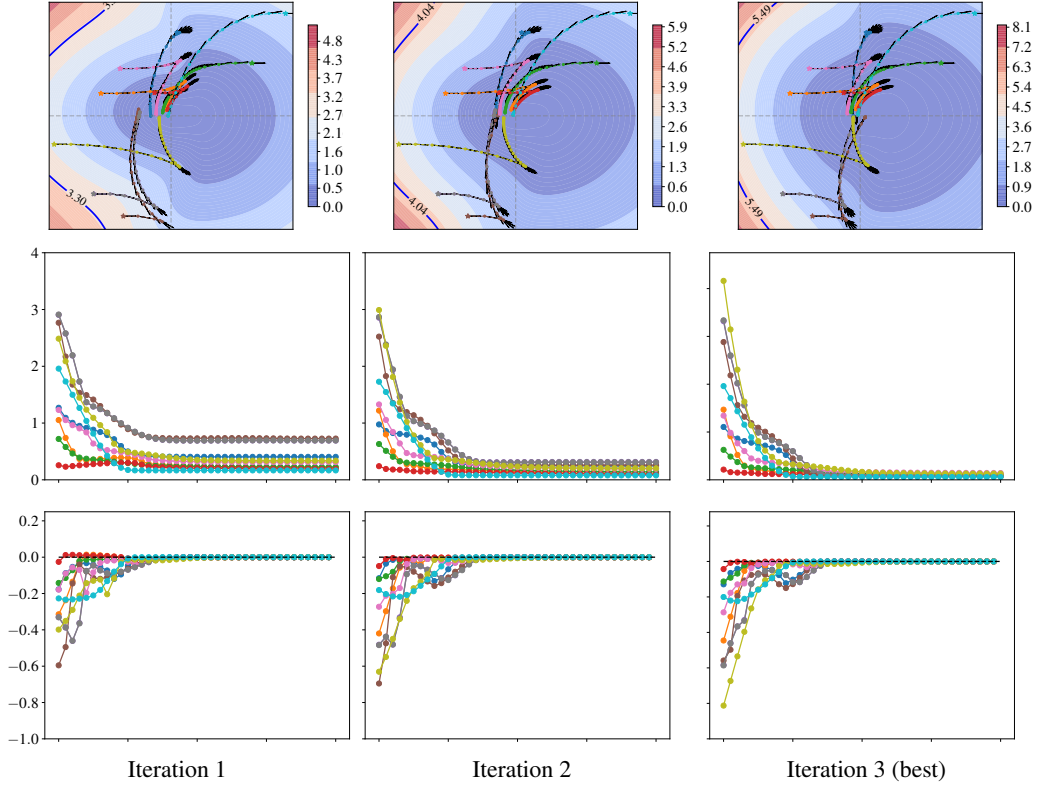


Figure 7: **Car kinematics: Testing Neural Lyapunov MPC obtained from training on nominal model.** For each iteration, we show the trajectories obtained through our Neural Lyapunov MPC while using the resulting Lyapunov function and the MPC parameter selected from the line-search. **Top:** The Lyapunov function at $\phi = 0$ with trajectories for 40 steps at each iteration. **Middle:** The evaluated Lyapunov function. **Bottom:** The Lyapunov function time difference.

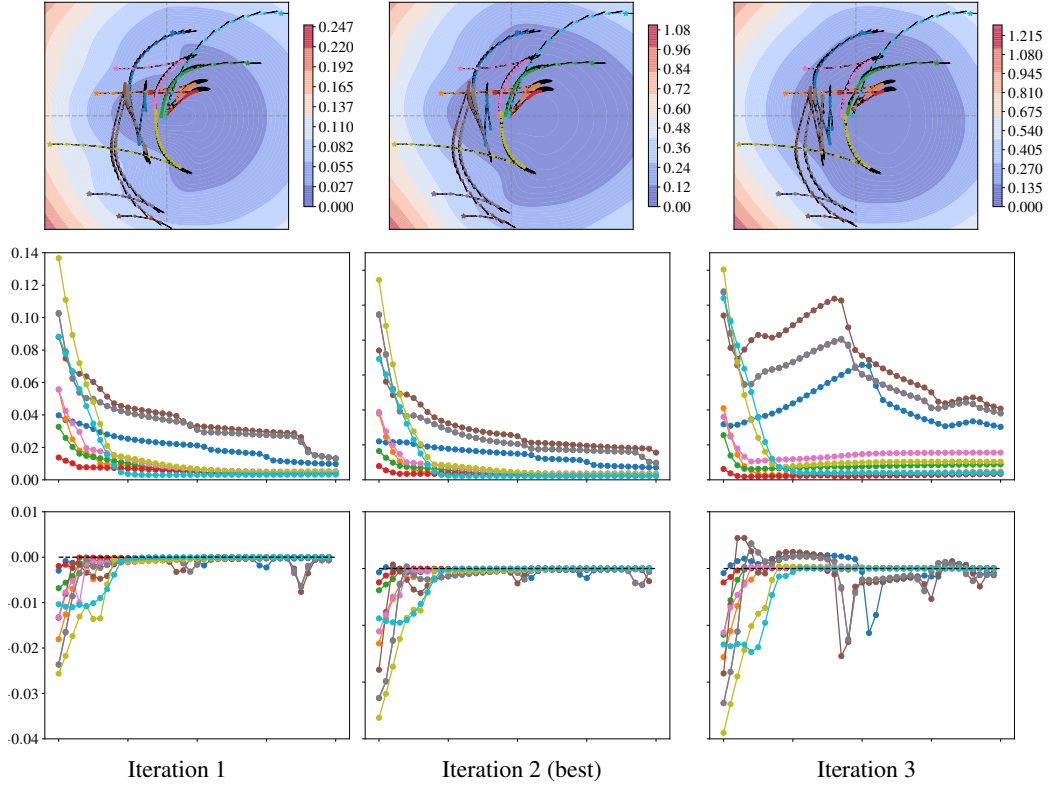


Figure 8: **Car kinematics: Testing Neural Lyapunov MPC obtained from training on surrogate model.** For each iteration, we show the trajectories obtained through our Neural Lyapunov MPC while using the resulting Lyapunov function and the MPC parameter selected from the line-search. **Top:** The Lyapunov function at $\phi = 0$ with trajectories for 40 steps at each iteration. **Middle:** The evaluated Lyapunov function. **Bottom:** The Lyapunov function time difference.

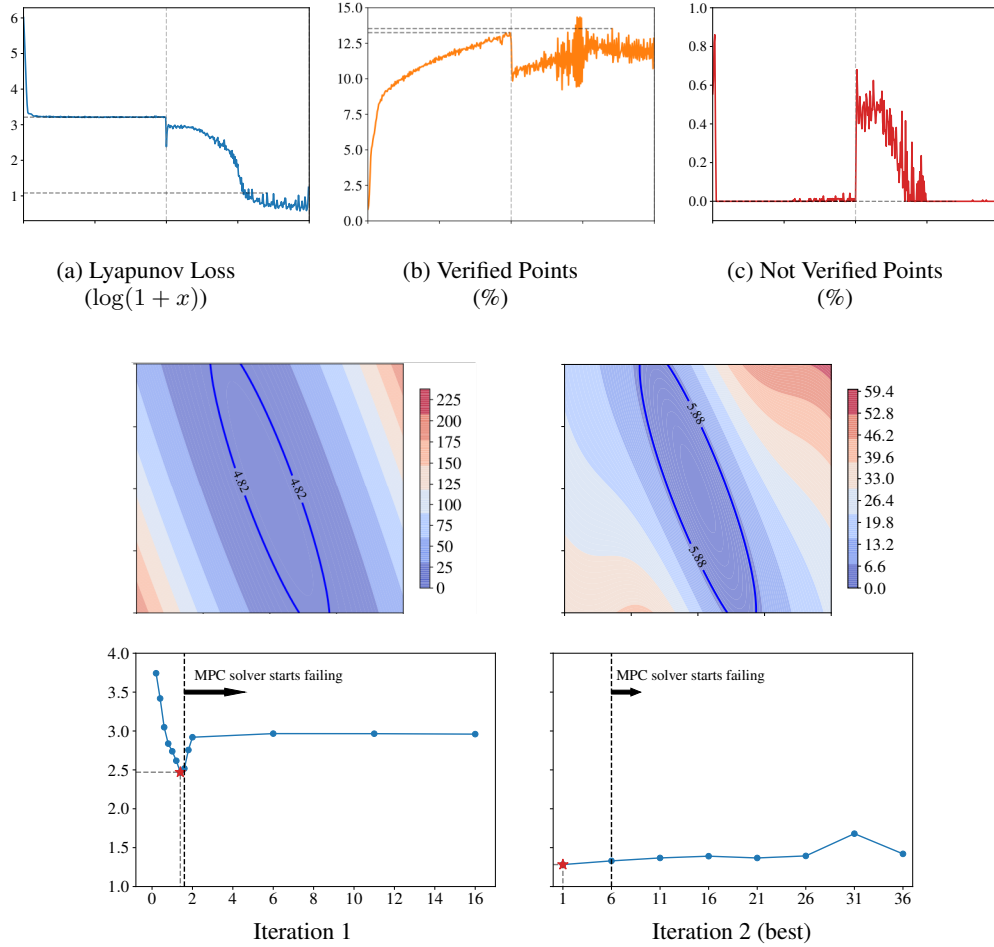


Figure 9: **Inverted Pendulum: Alternate learning on surrogate model.** After every $N_V = 200$ epochs of Lyapunov learning, the learned Lyapunov function is used to tune the MPC parameters. Unlike the vehicle kinematics example, we do not reinitialize V between the iterations. **Top:** The training curves for Lyapunov function. Vertical lines separate iterations. **Middle:** The resulting Lyapunov function V with the best performance. **Bottom:** Line-search for the MPC parameter α to minimize the Lyapunov loss (10) with V as terminal cost. The loss is plotted on the y-axis in a $\log(1+x)$ scale. The point marked in red is the parameter which minimizes the loss.

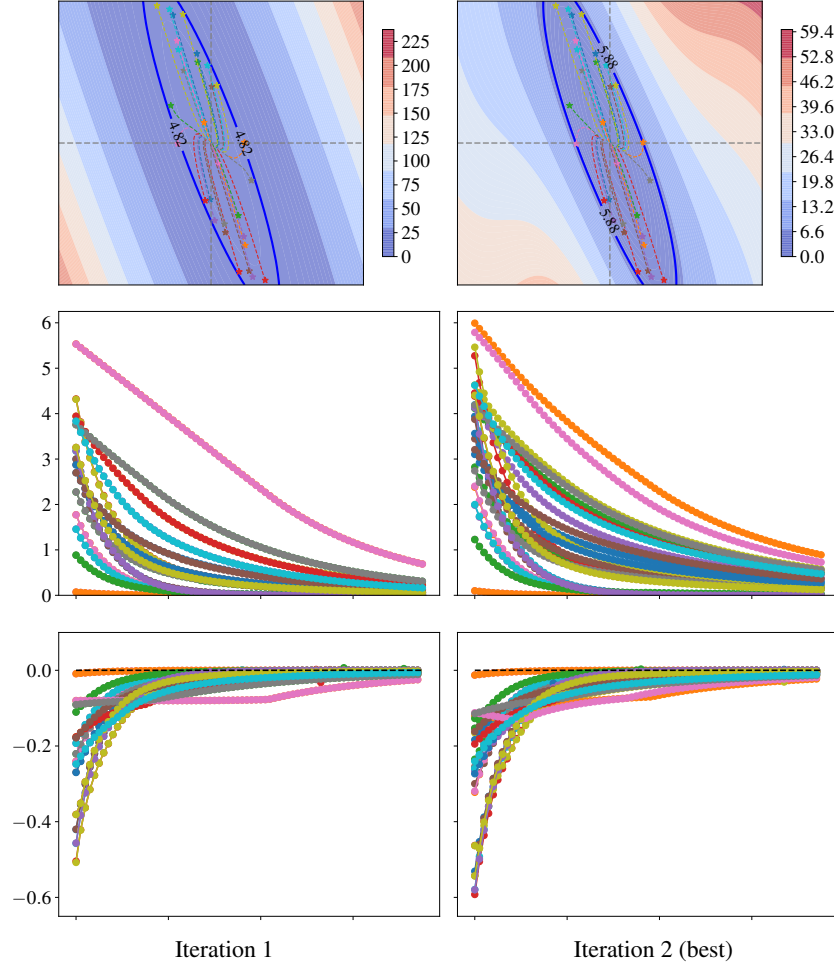


Figure 10: **Inverted Pendulum: Testing Neural Lyapunov MPC obtained from training on nominal model over iterations.** For each iteration, we show the trajectories obtained through our Neural Lyapunov MPC while using the resulting Lyapunov function and the MPC parameter selected from the line-search. The initial states are sampled inside the safe level-set using rejection sampling. **Top:** The Lyapunov function with trajectories for 80 steps at each iteration. **Middle:** The evaluated Lyapunov function. **Bottom:** The Lyapunov function time difference.

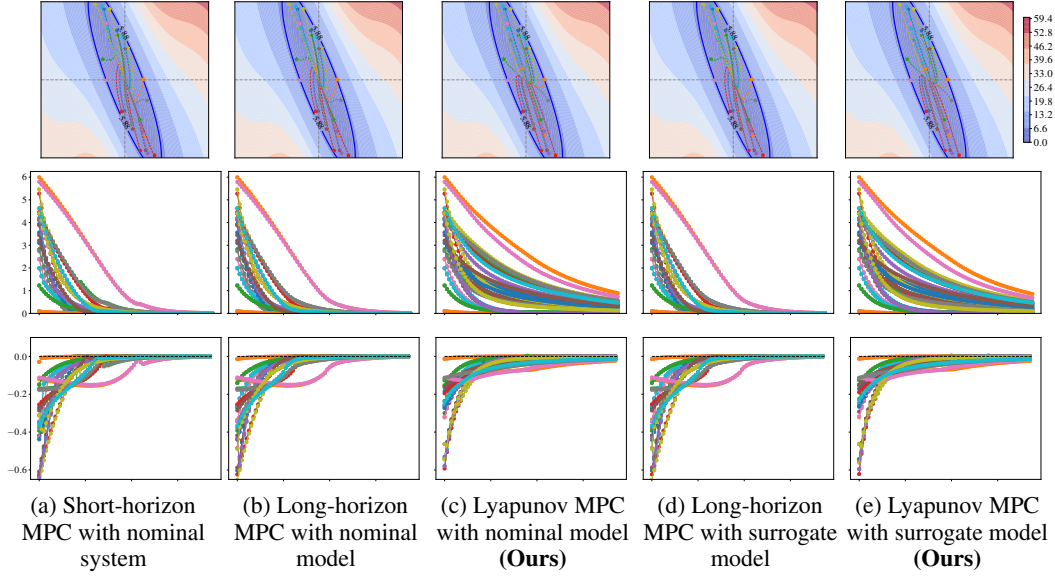


Figure 11: **Inverted Pendulum: Transfer from nominal to surrogate model.** **Top:** The Lyapunov function with overlaid trajectories for 80 timesteps. **Middle:** The Lyapunov function evaluated along trajectories. **Bottom:** The Lyapunov decrease evaluated along trajectories.

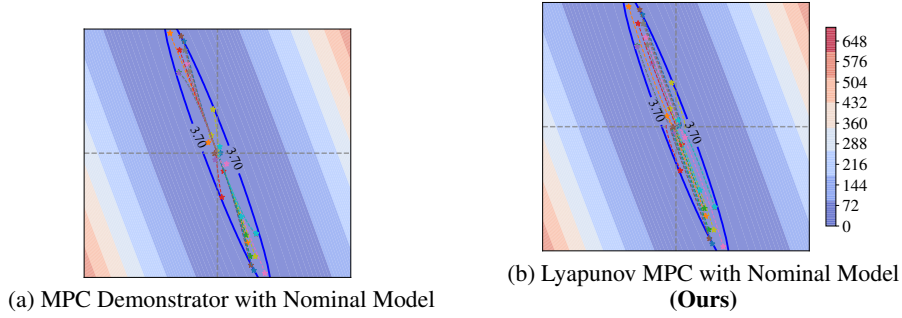


Figure 12: **Inverted Pendulum: Effect of using Lyapunov MPC with contractor factor and no LQR loss.** The Lyapunov function and safe-level set obtained from the first iteration of alternate learning with $\lambda = 0.9, v = 0$ in the Lyapunov loss Equation 10. This results in a smaller safe region estimate and slower closed-loop trajectories compared to the case when $\lambda = 0.99, v = 1$. Each trajectory is simulated for $T = 80$ timesteps.

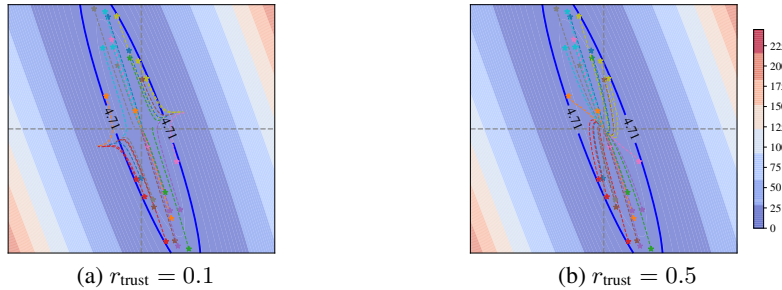


Figure 13: **Inverted Pendulum: Effect of trust region on MPC.** The solver hyperparameter, r_{trust} , can also affect the stability of the MPC and was tuned manually at this stage. Given the limited amount of solver iterations, a small trust region results in weaker control signals and local instability. A larger trust radius can in this case stabilize the system, while being possibly more sub-optimal. The depicted Lyapunov function is obtained from first iteration of alternate learning and is used by the MPC as its terminal cost.