

A APPENDIX

Algorithm 1 RAMPS

Input: Policy π , environment \mathcal{E} , initial datasets D, D_{val} , model-update period T_{model} , policy-update period T_{π}

- 1: Train initial linear model $\hat{F} = (A, B, c)$ on D ; compute validation errors $\{\varepsilon_i\}$ on D_{val}
- 2: Set error bound $\varepsilon \leftarrow \text{quantile}(\{\varepsilon_i\}, q)$
- 3: Precompute matrices for all horizons up to H_{max} (matrix powers A^k , constraint templates $G(H)$, accumulators M_h)
- 4: **loop**
- 5: $a_{\pi} \leftarrow \pi(s)$
- 6: Observe lifted state z from s
- 7: $u_{\text{best}} \leftarrow \text{None}$; $\text{best_H} \leftarrow \text{None}$
- 8: $H_{\text{lo}} \leftarrow H_{\text{min}}$; $H_{\text{hi}} \leftarrow H_{\text{max}}$
- 9: **while** $H_{\text{lo}} \leq H_{\text{hi}}$ **do**
- 10: $H_{\text{mid}} \leftarrow \lfloor (H_{\text{lo}} + H_{\text{hi}})/2 \rfloor$
- 11: Build constraint matrices $G(H_{\text{mid}})$ and $h(H_{\text{mid}})$ using (A, B, c) , ε , and precomputed $\{A^k\}$
- 12: Solve QP $_{H_{\text{mid}}}$: $\min_{u_{0:H-1}} \|u_0 - a_{\pi}\|_2^2$
 s.t. $G(H_{\text{mid}})u \leq h(H_{\text{mid}})$, $u_k \in \mathcal{U}$
- 13: **if** QP $_{H_{\text{mid}}}$ feasible **then**
- 14: $u_{\text{best}} \leftarrow \text{solution}$; $\text{best_H} \leftarrow H_{\text{mid}}$
- 15: $H_{\text{lo}} \leftarrow H_{\text{mid}} + 1$
- 16: **else**
- 17: $H_{\text{hi}} \leftarrow H_{\text{mid}} - 1$
- 18: **end if**
- 19: **end while**
- 20: **if** $u_{\text{best}} = \text{None}$ **then**
- 21: Apply backup action $u \leftarrow u_{\text{backup}}(z)$
- 22: **else**
- 23: Apply shielded action $u \leftarrow u_{\text{best}}[0]$
- 24: **end if**
- 25: Execute u in environment, store transition in D
- 26: Periodically refit (A, B, c) and recompute ε and QP precomputations (every T_{model} steps)
- 27: Periodically update policy π from D (every T_{π} steps)
- 28: **end loop**

A.1 COMPUTATIONAL COMPLEXITY ANALYSIS

The computational efficiency of our shielding framework is critical for its real-time applicability. The architecture of our method is designed to front-load the most intensive computations into a single, state-independent pre-computation phase, leaving the per-timestep solve phase remarkably lightweight. We analyze the complexity of these two phases below, defining s as the state dimension, u as the action dimension, H as the maximum prediction horizon, and m as the number of faces in the safety polyhedron.

One-Time Pre-computation Cost. The computationally intensive construction of the QP’s state-independent components is performed in a pre-computation phase, which is executed only when the underlying Koopman dynamics model is updated. This pre-computes and caches all components of the QP that are independent of the current state z_k . The construction of the QP constraint matrices dominates the complexity of this phase.

- **Matrix Power and Affine Term Pre-computation:** Calculating the powers of the state matrix A up to A^H requires $\mathcal{O}(H \cdot s^3)$ operations. The cumulative affine terms are subsequently computed in $\mathcal{O}(H \cdot s^2)$.
- **Constraint Matrix Construction:** The primary cost lies in constructing the matrices for the full-horizon QP. The constraint matrix G_{all} has dimensions $(N_c \times Hu)$, where the number of

constraints $N_c \leq mH$. The vectorized right-hand-side matrices, M_h and v_h , are constructed with a complexity of approximately $\mathcal{O}(\mathbf{m} \cdot \mathbf{H} \cdot \mathbf{s}^2)$.

The dominant term arises from the matrix power calculation, making the total complexity of the pre-computation phase $\mathcal{O}(\mathbf{H} \cdot \mathbf{s}^3 + \mathbf{m} \cdot \mathbf{H} \cdot \mathbf{s}^2)$. This cost is incurred only once per model update, not at every control step.

Real-Time Solve Cost. The shield is executed at each timestep and is designed for high-frequency operation. Its complexity is significantly lower due to the extensive pre-computation.

- **State-Dependent Calculation:** The only significant state-dependent computation is the calculation of the right-hand-side vector h_{all} . Leveraging the pre-computed matrices, this is reduced to a single matrix-vector product, $h_{\text{all}} = M_h z_k + v_h$, which has a complexity of $\mathcal{O}(\mathbf{m} \cdot \mathbf{H} \cdot \mathbf{s})$.
- **Binary Search and QP Solution:** The binary search for the largest feasible horizon performs $\mathcal{O}(\log H)$ iterations. Within each iteration, we update the QP’s bounds and solve it. The ‘update’ operation is linear in the number of constraints, $\mathcal{O}(N_c)$. Crucially, the ‘solve’ call is warm-started from the previous iteration’s solution, making its average-case complexity, which we denote $T_{\text{qp, warm}}$, substantially lower than solving from scratch.

Therefore, the total real-time complexity of the adaptive horizon selection is approximately $\mathcal{O}(\mathbf{m} \cdot \mathbf{H} \cdot \mathbf{s} + \log(\mathbf{H}) \cdot \mathbf{T}_{\text{qp, warm}})$. This low polynomial complexity ensures that the shield can operate efficiently in real-time control loops.

A.2 PROOF OF THEOREM 2

We restate the theorem for convenience:

Theorem. Let $\epsilon_1, \dots, \epsilon_N$ be a set of i.i.d. sampled model errors from our learned model \hat{F} . Assume that the probability of any two samples being equal is zero. Choose a quantile $0 < q < 1$ and let ϵ be the $\lceil qN \rceil$ ’th smallest value among $\epsilon_1, \dots, \epsilon_N$. Then

$$\Pr[\|F(s_k, u_k) - \hat{F}(s_k, u_k)\|_\infty > \epsilon] \leq 1 - q + \frac{1}{(2N)^{1/3}} + \frac{1}{4(2^{1/3})N^{2/3}}.$$

Proof. Let E be a random variable defining the errors in the learned model so that $\epsilon_1, \dots, \epsilon_N$ are i.i.d. samples from E . Let ϵ be a conservative q -quantile of these errors (that is, $\epsilon = \epsilon_{(\lceil qN \rceil)}$ where $\epsilon_{(i)}$ is the i ’th order statistic of the sampled errors). Define a random variable X such that

$$X = \begin{cases} 1 & E > \epsilon \\ 0 & \text{otherwise} \end{cases}.$$

Then X is a Bernoulli random variable with success probability $P[E > \epsilon]$ so that in particular $\mathbb{E}[X] = P[E > \epsilon]$ and $\text{Var}[X] = P[E > \epsilon](1 - P[E > \epsilon])$. We can now view our error samples $\epsilon_1, \dots, \epsilon_n$ in terms of X . By construction, exactly $\lceil qN \rceil$ of the error samples are less than or equal to ϵ , so we compute the sample mean of X

$$\hat{\mu}_X = \frac{N - \lceil qN \rceil}{N} = \frac{\lfloor N - qN \rfloor}{N} \leq 1 - q.$$

Applying Chebyshev’s inequality to $\hat{\mu}_X$, we find that for any positive c

$$P[|\hat{\mu}_X - P[E > \epsilon]| \geq c] \leq \frac{P[E > \epsilon](1 - P[E > \epsilon])}{Nc^2}$$

Notice that for all $0 \leq p \leq 1$ we have $p(1 - p) \leq 1/4$ so that

$$P[|\hat{\mu}_X - P[E > \epsilon]| \geq c] \leq \frac{1}{4Nc^2} \implies P[P[E < \epsilon] - \hat{\mu}_X \geq c] \leq \frac{1}{4Nc^2}.$$

Plugging in, $\hat{\mu}_X \leq 1 - q$ we find

$$P[P[E > \epsilon] \geq 1 - q + c] \leq \frac{1}{4Nc^2} \implies P[E > \epsilon] \leq 1 - q + c + \frac{1}{4Nc^2}.$$

Since this bound holds for any positive c , we set $c = (2N)^{-1/3}$, which minimizes the value of the right-hand side. Plugging in this value of c , we find the bound

$$P[E > \varepsilon] \leq 1 - q + \frac{1}{(2N)^{1/3}} + \frac{1}{4(2^{1/3})N^{2/3}}.$$

□

A.3 ABLATIONS

A.3.1 EXPLORING THE EFFECT OF THE ROBUSTNESS TERM

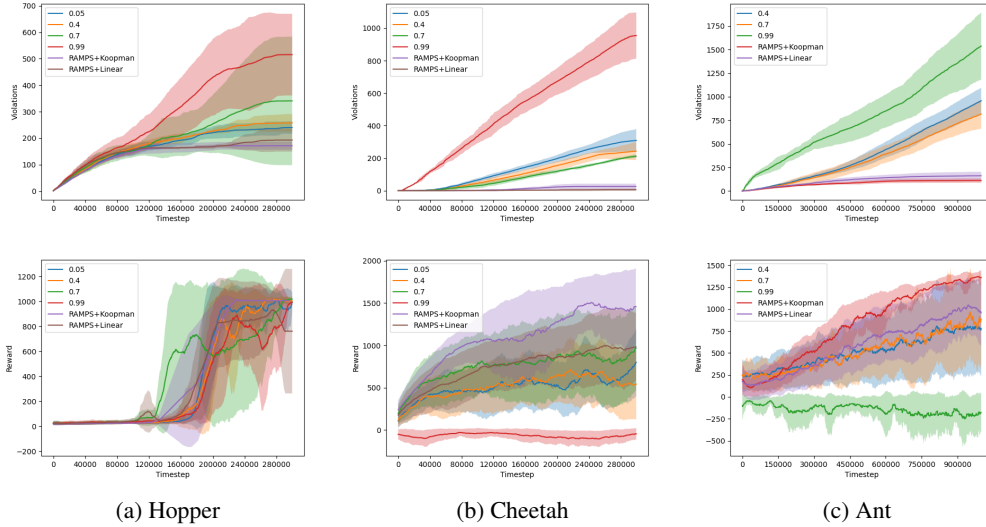


Figure 2: The critical role of the robustness term in ensuring safety. This figure compares the cumulative safety violations of the full RAMPS framework against a non-robust version that operates without the error-aware tightening term. While the non-robust shield continuously accumulates violations across all settings for the decay rate λ , the full RAMPS framework successfully learns to operate safely, evidenced by the flattening of its violation curves.

To isolate the contribution of our robust formulation, we performed a critical ablation study, presented in Figure 2. In this experiment, the shield operated without its error-aware tightening term. For direct comparison, each plot includes the performance of the full RAMPS framework using both the learned Koopman and a baseline Linear model.

While all non-robust configurations continuously accumulate safety violations, the full RAMPS framework’s violation curve consistently flattens, demonstrating its ability to learn to operate safely. This failure of the ablated models occurs because they operate on an overly optimistic view of the dynamics; they consistently certify actions that are safe within their flawed model but lead to catastrophic failures in the physical system. This result provides definitive evidence that while the multi-step CBF is a necessary structure, the explicit robustness to model uncertainty is the essential component that enables our framework to achieve strong safety guarantees.

Furthermore, the reward curves reveal that ineffective shielding directly harms task performance. The most conservative non-robust setting ($\lambda = 0.99$), which also suffers from high violations, yields the worst reward, often collapsing to negative values. This occurs because its overly strict constraints lead to frequent QP infeasibility, forcing the agent to rely on a simple backup policy that is not designed for task progression. In contrast, the full RAMPS framework not only achieves the best safety but also learns the highest-performing reward policy. This demonstrates that the shield is not overly aggressive; rather, by being robust and minimally invasive, it creates a stable learning environment that allows the agent to safely explore and find a more optimal policy.

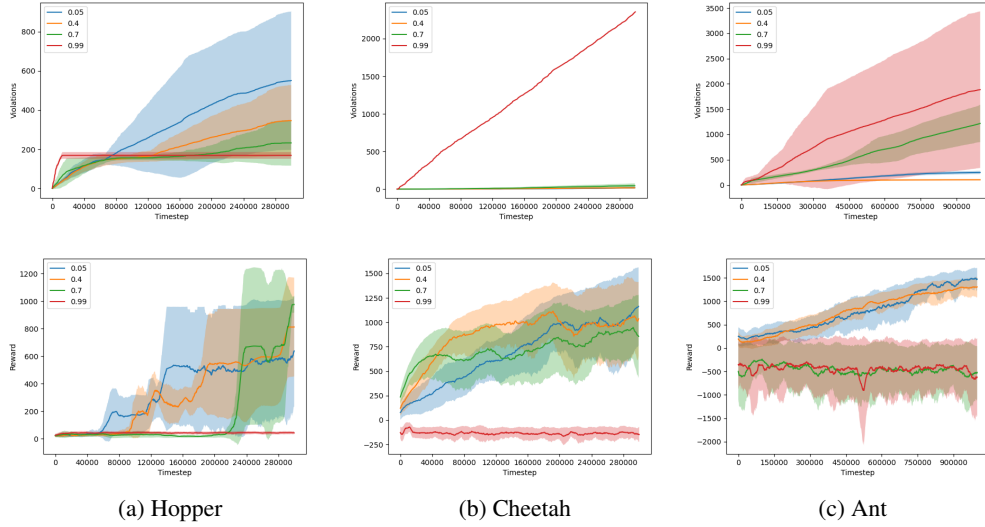


Figure 3: The impact of CBF conservatism (λ) on safety and performance. Moderately conservative decay rates (e.g., $\lambda = 0.4$, $\lambda = 0.7$) achieve the best balance of low safety violations and high task reward. The most conservative setting ($\lambda = 0.99$) paradoxically performs the worst due to frequent QP infeasibility.

A.3.2 THE EFFECT OF CBF λ CONSERVATISM

A key hyperparameter in our framework is the CBF decay rate λ , which dictates the conservatism of the shield. Figure 3 explores this parameter’s effect on both safety and reward performance when using the full, robust RAMPS framework. The results reveal a critical trade-off between constraint strictness and the feasibility of the shielding problem, a trade-off that varies with the complexity of the environment dynamics.

In the **Hopper** environment, the most conservative setting ($\lambda = 0.99$) results in a *safe failure*; the lowest violation count but also near-zero reward. This occurs because the strict requirement to preserve 99% of the safety margin makes the QP problem frequently infeasible, forcing the agent to over-rely on a passive backup policy that prevents task progression. Conversely, a setting of $\lambda = 0.7$ achieves the best performance in both safety and reward, indicating that the learned model is sufficiently accurate to consistently find feasible solutions under this demanding safety requirement.

The **Cheetah** environment paints a similar but distinct picture. Here, the $\lambda = 0.99$ setting again results in the worst performance, but leads to high violations and low reward, suggesting that when the primary QP fails, the simple backup policy is insufficient to manage Cheetah’s unstable dynamics. The best results are achieved with more permissive values ($\lambda \in \{0.05, 0.4\}$), suggesting that for more complex systems, the shield requires greater flexibility to ensure the underlying optimization problem remains feasible.

This trend is further emphasized in the highly unstable **Ant** environment. As with Cheetah, the $\lambda = 0.99$ setting is catastrophically poor in both safety and reward. A setting of $\lambda = 0.7$ is also too restrictive, hampering the agent’s ability to learn. The best overall performance is achieved with $\lambda = 0.4$, which provides a strong safety guarantee while allowing enough flexibility for the agent to learn a high-reward policy. A highly permissive setting like $\lambda = 0.05$ enables good policy learning but at the cost of higher safety violations. Ultimately, these results show that λ is a critical tuning parameter, with the optimal value becoming more permissive as the inherent instability of the environment increases.

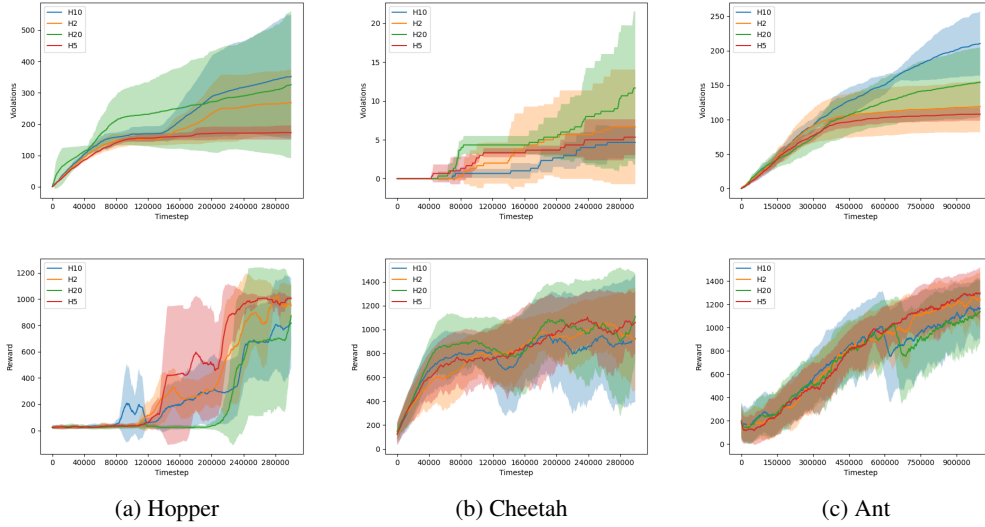


Figure 4: The trade-off between foresight and model reliability for the prediction horizon H . A moderate horizon ($H = 5$) achieves the best balance of low violations and high reward, avoiding both the myopic failures of a short horizon ($H = 2$) and the unreliable predictions of a long horizon ($H = 20$).

A.3.3 THE EFFECT OF PREDICTION HORIZON

The selection of the prediction horizon H presents a critical trade-off between predictive foresight and the reliability of the learned dynamics model. Figure 4 explores this trade-off, revealing that the optimal choice for H is environment-dependent but follows a clear pattern.

A long horizon ($H = 20$) consistently leads to poor performance in both safety and reward. This occurs because the multi-step predictions of the Koopman model become increasingly inaccurate as errors compound over the extended rollout. The shield is forced to make decisions based on this unreliable information, leading to suboptimal or unsafe interventions.

Conversely, a short horizon ($H = 2$) is also suboptimal. While the model is accurate over this brief window, the limited lookahead is insufficient to resolve the high relative-degree traps present in the dynamics, a core challenge this paper aims to address. The shield becomes myopic, failing to prevent safety violations that are inevitable several steps in the future.

The results show that a moderate horizon ($H = 5$) provides the optimal balance. It is long enough to provide the necessary foresight to handle control delays and traps, yet short enough that the learned model’s predictions remain reliable. This empirical finding validates our choice of $H = 5$ for the main experiments presented in this paper.

A.3.4 THE EFFECT OF ERROR BOUND CONFIDENCE

The robust tightening term in our framework is calibrated using an error bound, ϵ , derived from a hold-out validation set. The confidence of this bound is a critical hyperparameter, which we control by taking a percentile of the absolute one-step prediction errors. Figure 5 explores the impact of this choice on safety and reward.

The results show a clear and direct correlation between the confidence of the error bound and the safety of the resulting shield. A lower percentile (e.g., 25 or 50) provides an error bound that is too optimistic; it underestimates the true model error, leading to a high number of safety violations. As the confidence increases, the shield becomes more conservative and effective, with the 99th percentile (99) achieving the best safety performance by a significant margin.

Crucially, this improved safety directly enables better reward performance. By providing a more reliable and stable training environment, the 99th percentile configuration allows the RL agent to

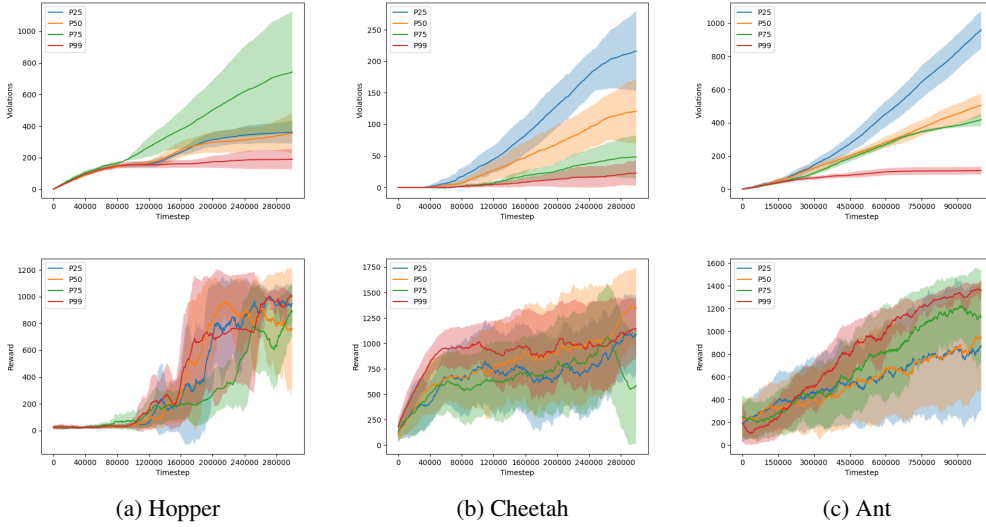


Figure 5: The impact of error bound confidence (percentile) on performance. A higher percentile (99), corresponding to a more conservative error bound, achieves the best safety (lowest violations) and enables the agent to learn a higher-reward policy.

explore more effectively and learn a higher-reward policy. Less confident settings, while seemingly more permissive, lead to catastrophic failures that terminate episodes, ultimately harming the agent’s ability to learn the task. This demonstrates that a sufficiently conservative, high-confidence error bound is not a hindrance to performance but is in fact a prerequisite for achieving both safety and high reward in complex environments.

A.3.5 SUMMARY OF ABLATION STUDIES

Taken together, our ablation studies validate the core design principles of the RAMPS framework and illuminate the critical trade-offs inherent in learning-based predictive safety. The results consistently demonstrate that achieving robust safety is not a matter of maximizing any single parameter, but of finding a carefully calibrated balance between competing factors.

The most crucial finding is that *explicit robustness to model error is the essential component for safety*. As shown in Figure 2, removing the error-aware tightening term is catastrophic; the shield becomes overly optimistic and fails to prevent a continuous accumulation of violations, regardless of other hyperparameter settings. This confirms that the ability to reason about its own model’s uncertainty is a prerequisite for the shield’s success.

Building upon this robust foundation, the remaining hyperparameters tune the balance between safety and performance. The prediction horizon H must balance foresight against model reliability; a moderate horizon (Figure 4) is optimal, as short horizons are too myopic to handle control delays, while long horizons suffer from compounding prediction errors. Similarly, the CBF decay rate λ must balance constraint strictness against QP feasibility (Figure 3), where an overly conservative setting can paradoxically harm both safety and reward by causing frequent reliance on a simple backup policy. Finally, our analysis of the error bound confidence (Figure 5) resolves a key trade-off, showing that a more conservative, high-confidence error bound (99th percentile) does not hamper performance but instead enables it by creating a more stable learning environment. Collectively, these results show that RAMPS is a co-designed system where each component is critical for achieving both high performance and strong safety guarantees.

A.4 ACTIVE RECOVERY BACKUP POLICY

In the rare event that the primary multi-step QP is infeasible, a deterministic backup policy is invoked. The use of such a policy is a standard practice in model-predictive shielding to ensure the agent

can always take an action. The active recovery approach detailed here is therefore **not a novel contribution of this work**, but rather follows a common pattern established in prior literature. Similar geometric recovery strategies are employed in other prominent shielding frameworks, such as SPICE Anderson et al. (2023), DMPS Banerjee et al. (2024), and MASE Wachi et al. (2023). For completeness, we describe our specific implementation of this widely-used technique below.

The policy, detailed in Algorithm 2, identifies the single most critical safety constraint, which is the one the agent is closest to violating; and solves a secondary, lightweight QP. The objective of this QP is to find a control action that maximally steers the system away from that constraint’s boundary, leveraging the inward-pointing normal vector of the safe set.

Algorithm 2 Active Recovery Backup Policy (following e.g., Anderson et al. (2023); Banerjee et al. (2024))

```

1: Input: Current latent state  $z$ 
2: Initialize: Minimum barrier value  $h_{\min} \leftarrow \infty$ , critical normal vector  $p^* \leftarrow \text{null}$ 
3: for all polyhedron face  $(p_i, b_i)$  in the definition of the safe set  $\mathcal{C}$  do
4:   Compute barrier value  $h_i(z) \leftarrow -(p_i^\top z + b_i)$ 
5:   if  $h_i(z) < h_{\min}$  then
6:      $h_{\min} \leftarrow h_i(z)$ 
7:      $p^* \leftarrow p_i$ 
8:   end if
9: end for
10: if  $p^*$  is null then
11:   {This occurs only if the state is not within any defined polyhedron.}
12:   return 0
13: end if
14: Define QP cost vector  $q \leftarrow (p^*)^\top B$ 
15: Solve the following QP for the recovery action  $u_{\text{backup}}$ :

```

$$\begin{aligned}
& \min_u \quad q^\top u \\
& \text{s.t.} \quad u \in \mathcal{U} \quad (\text{action bounds})
\end{aligned}$$

```

16: return  $u_{\text{backup}}$  if QP is solved, else return 0.

```

Limitations of Backup Policy. This recovery strategy is designed for computational efficiency, which necessitates several trade-offs common to such backup controllers. First, it is **non-robust**, relying on the nominal dynamics matrix B without accounting for model error. Second, it is **myopic**, operating as a one-step greedy controller without the foresight of a multi-step planner. Third, it focuses on the **single most critical constraint**, which may be insufficient when multiple constraints are nearly violated. Despite these inherent limitations, it provides a more principled fallback than a simple passive policy.

A.5 EXPERIMENTAL DETAILS

We evaluate our approach on five distinct environments, ranging from classic control tasks to more complex locomotion challenges, to demonstrate its efficacy across varying dimensionalities and dynamics.

Pendulum In this classic control benchmark, the goal is to swing up and stabilize an inverted pendulum. The environment has a 2-dimensional state space (encoding the pendulum’s angle and angular velocity) and a 1-dimensional action space (torque). A state is considered unsafe if the pendulum’s angle $|\theta|$ exceeds 0.4 radians. We allow all baselines 200000 environment interactions.

SafeHopper To test our method on more complex dynamics, we use the SafeHopper environment. This task involves controlling a two-legged robot, presenting a higher-dimensional challenge with an 11-dimensional state space and a 3-dimensional action space. Safety is defined by constraints on

the robot’s velocity ($-0.37315 \leq v \leq 0.37315$.) to ensure stable hopping. We allow all baselines 300000 environment interactions.

SafeCheetah The SafeCheetah environment is another complex benchmark. The agent must control a planar cheetah-like robot to run forward. This environment features a 17-dimensional state space and a 6-dimensional action space. Similar to SafeHopper, the safety specifications impose constraints on the robot’s velocity ($-2.8795 \leq v \leq 2.8795$.) to prevent unstable or dangerous movements. We allow all baselines 300000 environment interactions.

SafeAnt We further increase the complexity with the SafeAnt environment, which involves controlling a quadrupedal robot. The agent must learn to walk forward in a high-dimensional state space of 105 dimensions, with an 8-dimensional action space. Safety is defined by constraints on the robot’s velocity ($-2.3475 \leq v \leq 2.3475$.) We allow all baselines 1000000 environment interactions.

SafeHumanoid We also test RAMPS on SafeHumanoid, a highly challenging benchmark with a 348-dimensional state space and 17-dimensional action space. The agent must learn to coordinate full-body locomotion while remaining within prescribed safety limits, defined by velocity constraints ($-2.3475 \leq v \leq 2.3475$). As with the other environments, each baseline receives 1000000 environment interactions. Due to its dimensionality and instability, SafeHumanoid is known to be difficult for safe-RL algorithms, making it a strong stress test for both the learned dynamics and the shielding mechanism.

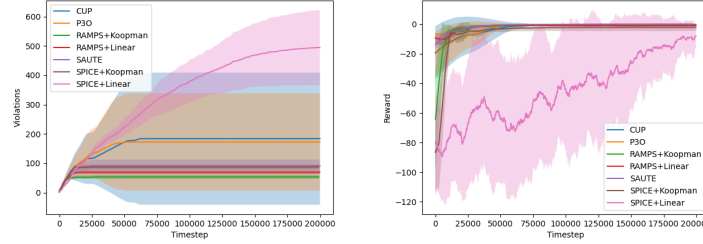


Figure 6: Safety violations (top) and episodic reward (bottom) for Pendulum.

Pendulum is a low-dimensional, easy-to-control system where safety constraints and dynamics pose minimal difficulty. As expected, all methods achieve near-perfect safety and high reward. We include the results here for completeness; the main paper focuses on higher-dimensional domains where RAMPS’ multi-step shielding and scalable model-based prediction are essential.

A.5.1 DEEP KOOPMAN OPERATOR IMPLEMENTATION

For all experiments involving a learned latent dynamics model (RAMPS + K and SPICE + K), we use the Deep Koopman Operator framework introduced by Shi & Meng Shi & Meng (2022). The central idea of this approach is to represent the nonlinear system dynamics with a linear model in a suitably constructed higher-dimensional *lifted* space. This is achieved by augmenting the original state with learned features that allow the dynamics to evolve linearly in the expanded coordinate system.

Lifted State Representation. Let x_k denote the original system state at time k . A deep encoder network $g_\theta(\cdot)$ generates additional latent coordinates, producing the lifted state

$$z_k = \begin{bmatrix} x_k \\ g_\theta(x_k) \end{bmatrix}.$$

In this lifted space, the system evolution is modeled by a linear dynamical system:

$$z_{k+1} = \mathbf{A}z_k + \mathbf{B}u_k,$$

where u_k is the control input and the matrices \mathbf{A} and \mathbf{B} are learned jointly with the encoder.

Network Architecture. The encoder is realized as a three-layer multilayer perceptron (MLP) with two hidden layers of 256 units each. Hidden layers use SiLU activations and the final layer uses a Tanh activation. The dimensionality of the lifted latent features is environment-specific, chosen to balance model accuracy and computational tractability: we use 2 for Pendulum, 22 for SafeHopper, 34 for SafeCheetah, and 100 for both SafeAnt and SafeHumanoid. These dimensions were selected empirically based on prediction accuracy and shield feasibility profiles.

Training Objective. The Koopman model is trained end-to-end using the loss function proposed by Shi & Meng Shi & Meng (2022), consisting of two complementary components:

1. **State Prediction Loss:** Measures the discrepancy between the true next state x_{k+1} and the predicted next state extracted from $z'_{k+1} = \mathbf{A}z_k + \mathbf{B}u_k$. This encourages accurate one-step predictions in the original state space.
2. **Linearity Loss:** Enforces consistency between the learned feature representation and linear evolution under the Koopman operator. Specifically, it penalizes the difference between the predicted lifted features $g_\theta(x'_{k+1})$ and the lifted features of the true next state $g_\theta(x_{k+1})$.

Together, these losses encourage g_θ to discover a set of basis functions that linearize the system dynamics, effectively serving as a global regularizer and promoting long-horizon predictive stability.

Constraint Handling in the Lifted Space. Safety constraints are defined in the original state space via polyhedral sets. Rather than imposing constraints directly on the learned latent coordinates, we preserve the original safety specifications by *zero-padding* the additional Koopman features. That is, the lifted constraint set is constructed by embedding the original constraint polytope into the higher-dimensional space as:

$$\mathcal{C}_{\text{lifted}} = \mathcal{C} \times \{0\}^{d_{\text{latent}}}.$$

This approach ensures that safety checks and subsequent CBF/QP computations remain well-defined without introducing arbitrary or unverified restrictions on the latent variables. While conservative, this choice has two benefits: (i) it guarantees compatibility with the underlying safety definitions, and (ii) it avoids imposing structural constraints on the learned features whose semantics are not directly interpretable. Investigating learned or data-driven safety projections in the latent space is an interesting direction for future work.

Summary. In combination, these design choices allow the Deep Koopman Operator to model high-dimensional, nonlinear systems with a compact linear approximation in the lifted space, enabling fast multi-step predictions and efficient computation of safety certificates needed for RAMPS and SPICE. Despite its simplicity, the model is sufficiently expressive for the complex locomotion domains we consider, while maintaining the computational tractability required for real-time shielding.

A.5.2 ERROR BOUND CALIBRATION

The dynamics model and its corresponding error bound, ϵ , are periodically recalibrated throughout training to adapt to newly collected data. An initial model is trained after the first 10,000 environment steps. Subsequently, the model is fine-tuned at progressively doubling intervals (e.g., at 20,000, 40,000, and 80,000 steps). At each training stage, the error bound is calibrated using a hold-out validation set comprising 20% of all collected experience. To maintain computational efficiency, the size of this validation set is capped at a maximum of 100,000 samples.

A.5.3 ANALYSIS OF BASELINE FAILURES

As noted in Table 1, the SPICE+L baseline failed on all high-dimensional locomotion tasks (**SafeHopper**, **SafeCheetah**, and **SafeAnt**). This is attributable to the representational limits of a simple linear model in capturing the complex, nonlinear dynamics of these environments. The resulting model exhibited large prediction errors, which, within SPICE’s one-step shielding formulation, rendered the safety QP persistently infeasible. This forced the agent to over-rely on its backup policy, preventing it from learning the task.

More critically, **SPICE+K** also failed on the most complex environment, **SafeAnt**. This failure occurred even when using the *exact same* pre-trained Deep Koopman Operator that was successful for

our method, RAMPS + K . This finding isolates the failure to SPICE’s underlying shielding technique. Its myopic, one-step approach is not sufficiently robust to the larger, yet unavoidable, prediction errors of a learned model in such a high-dimensional space. In contrast, our multi-step formulation is explicitly designed to tolerate these errors, explaining the significant performance difference.

We note explicitly that **DMPS** (Banerjee et al., 2024), **VELM** (Wang & Zhu, 2024) and **MASE** (Wachi et al., 2023) are model-predictive shielding (MPS) techniques which operate using a safety predicate or safe/unsafe-state specification rather than relying on dense cost or reward shaping. As such they are directly related to SPICE and RAMPS in design intent: all attempt to find safe actions via online planning under a safety specification. Despite operating with this stronger safety interface, the publicly available implementations of DMPS and VELM, and our re-implementation of MASE (Wachi et al., 2023), exhibited rapid violation growth and frequent infeasible/timeout planner returns on the MuJoCo locomotion benchmarks (Hopper, Cheetah, Ant), accumulating >1000 violations in the first 20–30k environment interactions. We also tested the **Conservative Safety Critics** approach (Bharadhwaj et al., 2021a), which similarly failed to train stably in these setting. Because these failure modes made them impractical and unstable as baselines for our main comparisons, we exclude them from the final baseline table. For transparency and validation, we will release all code and scripts for these baselines along with our framework implementation.

A.5.4 ANALYSIS OF COMPUTATIONAL EFFICIENCY

Table 2: Shield Computation Time Analysis. This table shows the mean and standard deviation of the per-step execution time of the safety shield (Policy action proposal, state space lifting via Koopman, constraint assembly and QP solving) across all training episodes for each environment. The consistently low average times (all under 0.5 ms) confirm the real-time feasibility of the RAMPS framework.

ENVIRONMENT	RAMPS	SPICE
PENDULUM	0.2289 ± 0.01	0.4061 ± 0.0063
HOPPER	0.2822 ± 0.04	0.5996 ± 0.0624
CHEETAH	0.3234 ± 0.03	0.5244 ± 0.0418
ANT	0.4038 ± 0.08	2.4820 ± 1.8329
HUMANOID	0.5061 ± 0.02	3.7105 ± 1.2512

The primary design goal of the RAMPS framework is to make predictive shielding computationally tractable for real-time applications. The timing results in Table 2 confirm the success of this approach. Across all tested environments, the mean per-step computation time for the shield is remarkably low, remaining well under half a millisecond.

Notably, the computation time scales gracefully with the complexity of the environment. For the simple **Pendulum** environment, the mean solve time is just 0.2289 ms. For the high-dimensional and dynamically complex **Ant** environment, this time increases to only 0.4038 ms. This sub-millisecond performance demonstrates that the extensive pre-computation phase is effective, leaving the online QP solve lightweight and suitable for high-frequency control loops. Furthermore, the low standard deviation across all environments indicates that the solver’s performance is consistent and predictable, a critical feature for reliable real-time systems. We use OSQP Stellato et al. (2020) for solving the QP.

A.5.5 ANALYSIS OF ACTION TYPE DISTRIBUTION

Table 3 details the ratio of actions selected by the primary shield, the original RL agent’s policy (Neural), and the fallback backup policy. This analysis reveals how the shield’s behavior adapts to the complexity of the environment.

The shield is the dominant actor in all environments, indicating its critical role in maintaining safety. This is most pronounced in the highly unstable **Ant** environment, where the shield intervenes in over 96% of steps, indicating that the RL agent rarely proposes a provably safe action on its own. It is critical to note that this high intervention rate does not prevent the agent from learning a high-reward policy. This is a direct result of the shield’s minimally invasive objective function, which finds the

Table 3: This table shows the per-episode average ratio of actions selected by the Shield, the original RL agent (Neural), and the Backup policy for results in Figure 1 and Table 1. The results show that the shield is highly active but allows the agent more freedom in less complex environments. The extremely low reliance on the Backup policy across all environments confirms the robustness and high feasibility rate of the primary multi-step QP shield.

ENVIRONMENT	SHIELD RATIO (%)	NEURAL RATIO (%)	BACKUP RATIO (%)
PENDULUM	74.72 ± 31.48	25.28 ± 31.48	0.00 ± 0.00
HOPPER	82.06 ± 4.11	17.08 ± 3.79	0.86 ± 0.93
CHEETAH	81.13 ± 4.55	17.27 ± 4.25	1.60 ± 0.59
ANT	96.50 ± 1.39	2.45 ± 1.25	1.05 ± 0.81
HUMANOID	96.28 ± 1.39	3.81 ± 0.81	0.00 ± 0.00

closest possible safe action to the agent’s original proposal. Consequently, many interventions are slight corrections that nudge the agent back towards safety without fundamentally disrupting its learned behavior.

In the moderately complex **Cheetah** and **Hopper** environments, the shield remains the primary actor but is significantly less invasive, allowing the agent’s neural policy to act directly approximately 17% of the time. This suggests that for these dynamics, the RL agent is better able to learn a policy that aligns with the safety constraints. The **Pendulum** environment shows the most interesting behavior; while the shield is active for 75% of the steps on average, the extremely high standard deviation (31.48%) suggests a bimodal behavior where the agent learns to operate safely for long periods before requiring periods of heavy intervention.

Finally, a crucial indicator of the primary shield’s robustness is the extremely low reliance on the backup policy. For all locomotion tasks, the backup policy is invoked only 1% of the time, and for Pendulum, it is never used at all. This demonstrates that the multi-step, adaptive-horizon QP is consistently able to find a feasible, provably safe solution, rarely needing to resort to its simpler fallback mechanism.

A.6 ANALYSIS OF SHIELD INTERVENTION

To better understand the trade-off between reward maximization and safety interventions, we analyze the average per-step action deviation ($\|u_{\text{shielded}} - u_{\text{agent}}\|$) during training in the Cheetah environment (Fig. 7).

We observe that **RAMPS+L** (linear model) produces consistently higher action deviations, indicating that the shield intervenes more aggressively to maintain safety. This stronger intervention translates into more reliable shielding performance, but it also limits the agent’s ability to explore freely, resulting in lower asymptotic reward.

By contrast, **RAMPS+K** (Koopman model) exhibits substantially smaller action deviations throughout training. This reduced level of intervention reflects the shield’s greater *invasiveness efficiency*: the agent is allowed to execute its intended actions more faithfully, leading to improved reward performance while still respecting safety constraints. In other words, RAMPS+K achieves a better balance between enforcing safety and preserving the agent’s autonomy.

A.7 ANALYSIS OF MULTI-DIMENSIONAL CONSTRAINTS ON HUMANOID

To evaluate RAMPS under realistic multi-dimensional safety conditions, we conducted an additional experiment on SAFEHUMANOID with a 348-dimensional state space and imposed a 21-dimensional polyhedral safety constraint set. Specifically, we constrained all coordinate velocities (state indices 23–25) to lie in $[-2.3475, 2.3475]$ and all angular velocities (state indices 28–45) to lie in $[-20, 20]$. Figure 8b shows that RAMPS+SAC achieves high reward (approximately 5000) and continues improving throughout training, while CMDP baselines (P3O, CUP, PPO-Saute) plateau early and fail to make progress.

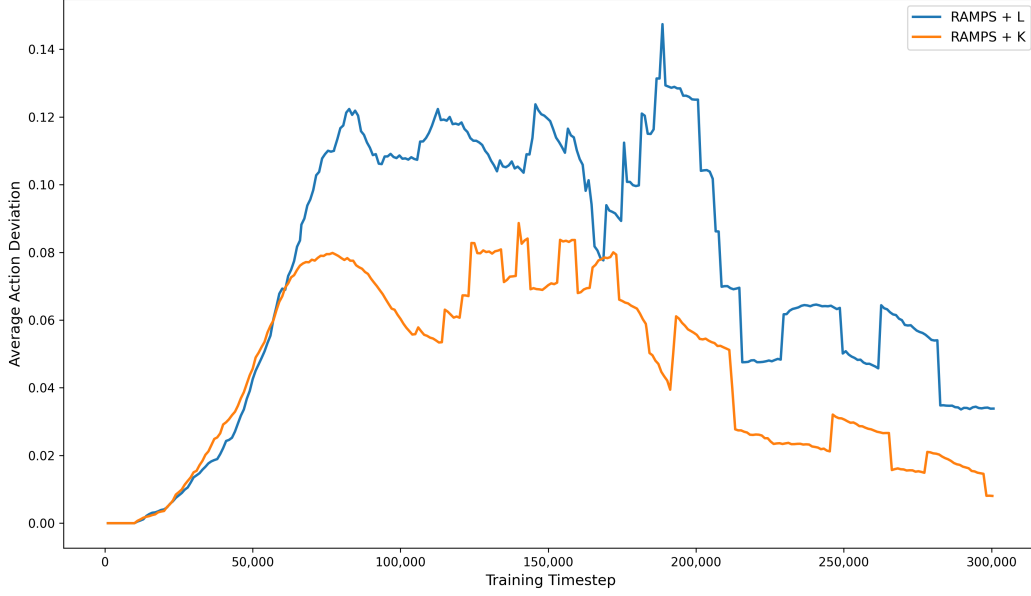


Figure 7: Average per-step action deviation ($\|u_{\text{shielded}} - u_{\text{agent}}\|$) during training on the Cheetah environment. Both variants show increasing intervention early in training as the agent explores unsafe behaviors, with RAMPS+K consistently yielding smaller deviations (less invasive interventions) and decaying sooner as the policy and model improve.

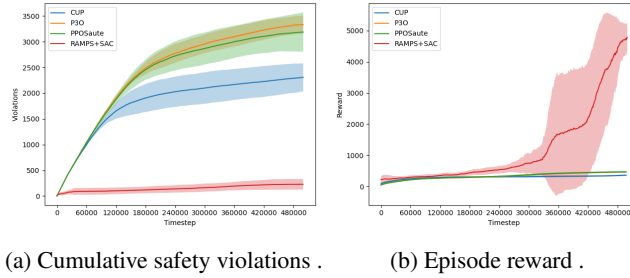


Figure 8: RAMPS evaluation on Humanoid with constraints on 21 dimensions (3 coordinate velocity constraints and 18 angular velocity constraints): (a) cumulative safety violations over training; (b) episode reward over training.

Figure 8a also demonstrates the safety behavior: RAMPS accumulates only about 256 cumulative violations over 500,000 steps, and reducing violations to 0 after 400,000 steps. CMDP baselines never learn to be safe, as evidenced by their increasing violation curves. These results confirm that RAMPS scales effectively to high-dimensional, coupled safety constraints and maintains both strong safety and high performance in settings where CMDP methods fail completely.

A.8 LIMITATIONS AND FUTURE WORK

Our framework operates under a common assumption in online learning: no a priori model of the environment is available. Consequently, the initial policy, π_0 , begins exploring without any prior knowledge of the environment, which can lead to safety violations while the dynamics model is being learned. These have also been reported in SPICE Anderson et al. (2023), VELM Wang & Zhu (2024) and DMPS Banerjee et al. (2024). A promising direction for future work is to mitigate these *cold start* violations by pre-training the Koopman model on relevant offline datasets, thereby enabling a safer initial policy. Furthermore, while our method provides a high-confidence probabilistic safety certificate, it does not offer a hard, worst-case guarantee on the number of violations. Future research

could focus on bridging this gap by employing formal verification techniques, such as abstract interpretation, to compute a rigorous upper bound on the number of potential safety violations throughout the learning process.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of our work. To this end, we provide the following resources. **Source Code:** The complete implementation of the RAMPS framework, including the Koopman dynamics model, the multi-step CBF shield, and all training scripts used to generate our results, is available as supplementary material. **Theoretical Foundations:** The mathematical formulation of our multi-step robust CBF, including the derivation of the safety constraints and the robust tightening term, is detailed in Section 4. The probabilistic safety guarantees relative to the learned model are established in 4. **Experimental Details:** Our experimental setup, including environment descriptions, safety specifications, and baseline implementations, is described in Section 5 and Section A. Furthermore, our extensive ablation studies, detailed in Appendix A.3, provide a clear analysis of hyperparameter sensitivity and justify our final configuration choices. We believe these resources provide a clear and complete path for reproducing our findings and building upon our work.