Neural Co-state Policies: Structuring Hidden States in Recurrent Reinforcement Learning

David Leeftink, Max Hinne, Marcel van Gerven
Department of Machine Learning and Neural Computating
Donders Institute for Brain, Cognition and Behaviour
Radboud University, Nijmegen, the Netherlands.
{david.leeftink, max.hinne, marcel.vangerven}@ru.nl

Abstract

A key capability of intelligent agents is operating under partial observability: reasoning and acting effectively despite missing or incomplete state observations. While recurrent (memory-based) policies learned via reinforcement learning address this by encoding history into latent state representations, their internal dynamics remain uninterpretable black boxes. This paper establishes a formal link between these hidden states and the Pontryagin minimum principle (PMP) from optimal control. We demonstrate that for standard recurrent architectures, latent representations map directly to PMP co-states, which allows the readout layer to be interpreted as performing Hamiltonian minimization. Because standard reward maximization does not naturally discover this alignment, we introduce a PMP-derived co-state loss to explicitly structure the internal dynamics. Empirically, this approach matches or improves performance on partially observable DMControl tasks, and is robust against zero-shot out-of-distribution sensor masking. By framing recurrent networks as dynamic processes governed by the minimum principle, we provide a principled approach to designing robust continuous control policies.

1 Introduction

A fundamental challenge for intelligent agents is operating effectively under partial observability. Real-world physical tasks are inherently obscured and often characterized by noisy sensors, measurement delays, or missing data. Consequently, biological and artificial systems in continuous control tasks rarely have full access to the true state of their environment. Because a single instantaneous observation is typically insufficient to determine the underlying state of the system, an agent must learn to integrate a history of past events to infer what cannot be directly seen (Kaelbling et al., 1998).

In deep reinforcement learning (RL), recurrent policies address partial observability by maintaining a hidden state that continuously accumulates information. While optimizing these networks purely for reward yields strong performance, their internal dynamics remain an uninterpretable black box (Wierstra et al., 2010; Hausknecht & Stone, 2015). Without explicit structural constraints, recurrent policies are prone to memorizing fragile heuristics rather than learning a grounded representation of the control task, leaving them vulnerable to breaking down entirely under unfamiliar conditions.

To structure these internal dynamics, this paper establishes a formal link between recurrent policies and the Pontryagin minimum principle (PMP) (Pontryagin, 1987). PMP states that optimal continuous control is governed by a Hamiltonian system, where co-state variables co-evolve with the environment to encode optimality conditions. As shown in Fig. 1, we formalize this relationship by introducing neural co-state policies (NCP); a framework that explicitly aligns neural memory updates with co-state dynamics. We demonstrate that standard architectures like continuous-time recurrent neural networks (CT-RNNs) (Beer, 1995) and gated recurrent units (GRUs) (Chung et al., 2014) inherently possess

Preprint.

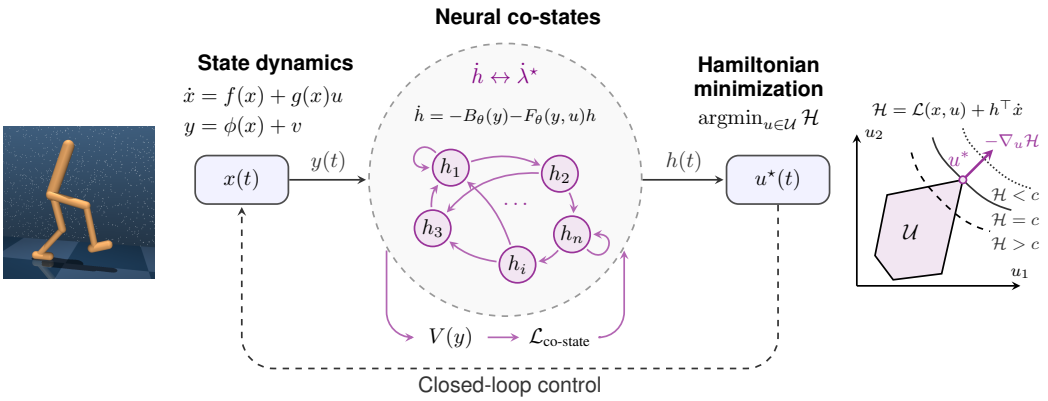

Figure 1: **Recurrent Reinforcement Learning via Neural co-state policies.** NCP s are regularized during training to mirror the optimality conditions implied by the minimum principle. By structuring the hidden states as representations of the underlying optimal control co-states, the read-out layer acts as a control-Hamiltonian minimizer.

this exact mathematical structure. By mapping their latent memory directly to optimal co-states, the network's final readout layer can be interpreted as performing Hamiltonian minimization.

While recurrent policies have the capacity for optimal Hamiltonian dynamics, standard reward maximization does not naturally converge to this alignment. To bridge this gap, we introduce an auxiliary co-state loss derived from the Hamilton-Jacobi-Bellman (HJB) equation (Bellman, 1966). Because HJB theory establishes the theoretical co-state as the gradient of the value function (Vinter, 1986), these targets can be dynamically extracted directly from the learned critic in standard actor-critic architectures. Supervising the actor with these targets explicitly structures the network's hidden states to track the latent optimality conditions of the environment.

We evaluate our approach on partially observable continuous control tasks from DeepMind Control Suite (DMControl; Tassa et al. (2018)). Empirically, applying the co-state loss to CT-RNN and GRU architectures matches or improves performance over recurrent policies trained with proximal policy optimization (PPO) baselines (Schulman et al., 2017). We furthermore show that the learned, structured internal dynamics demonstrate robustness to increased out-of-distribution sensor dropout.

Ultimately, this work bridges a critical gap between classical optimal control and deep RL by anchoring previously black-box hidden states in the minimum principle's mathematical structure. By framing recurrent architectures as dynamic processes governed by this principle, we provide a theoretically aligned foundation to design robust continuous control policies.

2 Background: Recurrent Policy Optimization

2.1 Continuous control under partial observability

Many real-world reinforcement learning (RL) domains, such as robotics and autonomous navigation, involve physical systems governed by continuous-time dynamics where the agent only receives partial, noisy sensor readings. We formalize this setting as a continuous dynamical system described by a differential equation and an observation equation:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{2.1}$$
$$y(t) = \phi(x(t)) + v(t) \tag{2.2}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ represents the true underlying system state, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{d_u}$ is the control input applied by the agent, and $y(t) \in \mathcal{O} \subseteq \mathbb{R}^{d_y}$ is the noise-corrupted observation with sensor noise $v(t)$. We assume the true drift dynamics f and control-influence matrix g are continuous and differentiable with respect to x. In the standard RL paradigm, these underlying transition dynamics are unknown to the agent, but they can be evaluated by executing actions and collecting observation trajectories over multiple learning episodes.

Finding the optimal control sequence $u(t)$ requires iteratively minimizing the cost objective:

$$J(u) = \mathbb{E}_v \left[\Phi(x(t_f)) + \int_{t_0}^{t_f} \Big(q(x(\tau)) + c(u(\tau)) \Big) d\tau \right]. \tag{2.3}$$

Here, $\Phi(x)$ is the terminal cost, $q(x)$ is the state cost, and $c(u)$ is the control input cost. We assume q and c are differentiable and can be evaluated empirically, though their underlying analytical forms are not known a priori. Because the sequence of observations is stochastic, finding the optimal control requires minimizing this expected cost. If the goal is to maximize the reward, instead $-J(u)$ can be maximized by using the running reward function as integrand.

2.2 Policies with memory

To minimize this cost functional, consider a policy $\pi_\theta \colon \mathcal{O} \to \mathcal{U}$, such that $u_\theta(t) = \pi_\theta(y(t))$ and $\theta \in \Theta$ are the policy parameters. This is a *memory-less* controller that only considers the (observation of) the current state to decide what action to take. The control inputs generated by this policy can be said to be optimal if they minimize the cost objective:

$$\begin{aligned} & \theta^\star \coloneqq \underset{\theta \in \Theta}{\operatorname{argmin}}\, J(\pi_\theta(x(t))) \\ \text{s.t.} \quad & \dot{x}(t) = f\left(x(t)\right) + g\left(x(t)\right) \pi_\theta(y(t)) \quad \text{for } t \in [t_0, t_f] \\ & x(t_0) = x_0 \ , \end{aligned} \tag{2.4}$$

where x_0 is the initial system state.

Because the true state $x(t)$ cannot be fully inferred from a single observation $y(t)$, this memory-less controller is fundamentally suboptimal. This necessitates a policy with memory, which maintains an internal latent state $h(t) \in \mathbb{R}^{d_h}$ that acts as a dynamical process coupled to the environment, where d_h is the dimensionality of the hidden state. In continuous time, this takes the generic form:

$$\dot{h}(t) = \psi_\theta(y(t), h(t)), \tag{2.5}$$

$$u(t) = \pi_\theta(h(t)), \tag{2.6}$$

where ψ_θ determines the evolution of the latent state. Common architectures for such dynamic policies CT-RNN or GRU (Wierstra et al., 2010). However, in standard deep RL practice, these methods treat the hidden state $h(t)$ as an unstructured and uninterpretable black-box.

3 Recurrent Policies as Optimal Dynamic Processes

In continuous control, optimizing policy parameters is fundamentally governed by the underlying structure of the stochastic optimal control problem in Eq. (2.3). This structure is characterized by the *necessary conditions of optimality*, traditionally approached via the Hamilton-Jacobi-Bellman (HJB) equation or PMP (Pontryagin, 1987; Kirk, 2004; Bryson & Ho, 1975). While HJB characterizes optimality globally across the state space, PMP directly describes optimality conditions along the trajectory. In what follows, we leverage PMP to demonstrate that the hidden state of a recurrent policy can be mathematically mapped onto the theoretical co-states of an optimal Hamiltonian system. While the stochastic optimal control problem is addressed via the stochastic minimum principle, we leverage the property that the internal dynamics of recurrent networks operate deterministically and ground our architectural mapping in the deterministic PMP.

3.1 Pontryagin Minimum Principle and Hamiltonian Systems

To study optimality over time, we consider the control-Hamiltonian, which evaluates the rate of change of the optimal value over time along the trajectory:

$$\mathcal{H}(x, \lambda, u) \coloneqq \mathcal{L}(x, u) + \lambda^\top \dot{x} = q(x) + c(u) + \lambda^\top \big(f(x) + g(x)u \big). \tag{3.1}$$

Here, $\lambda \in \mathbb{R}^{d_x}$ represents the *co-state* (or adjoint) vector. Acting as a continuous-time Lagrange multiplier, the co-state tracks the sensitivity of the value function to infinitesimal perturbations in the current state, and is connected to the value function via $\lambda(t) = \nabla V^\star(x)$ Vinter (1986). By differentiating the Hamiltonian w.r.t. the states and co-states, we obtain a coupled, continuous-time optimal boundary value problem. PMP states that an optimal control $u^\star(t)$ must minimize the Hamiltonian at all times $t \in [t_0, t_f]$.

Proposition 3.1 (Optimal Hamiltonian System)**.** *Let $u^\star(t)$ be an optimal solution for the control problem. The optimal trajectory $x^\star(t)$ and its corresponding co-state $\lambda^\star(t)$ must satisfy the following two-point boundary value problem for $t \in [t_0, t_f]$:*

$$\begin{aligned} \dot{x}^\star &= \nabla_\lambda \mathcal{H} = f(x^\star) + g(x^\star)u^\star \\ \dot{\lambda}^\star &= -\nabla_x \mathcal{H} = -\nabla_x q - (\nabla_x \dot{x}^\star)\,\lambda^\star \\ u^\star &= \operatorname*{argmin}_{u \in \mathcal{U}} \mathcal{H}(x^\star, \lambda^\star, u) \\ \textit{s.t.} \quad x^\star(t_0) &= x_0 \quad \textit{and} \quad \lambda^\star(t_f) = \nabla_x \Phi(x^\star(t_f)) \ . \end{aligned} \tag{3.2}$$

where the notation $(\nabla_y f)_{i,j} \coloneqq \partial f_j / \partial y_i$ denotes Jacobians, such that the gradient of the scalar Hamiltonian yields the column vector $\nabla_x \mathcal{H} = (\frac{\partial \mathcal{H}}{\partial x_1}, \ldots, \frac{\partial \mathcal{H}}{\partial x_{d_x}})^\top$.

Proof: The derivation follows standard extremization of the cost functional using variational calculus. A full derivation is provided in Appendix A.

3.2 The Neural Co-state Policy

The PMP framework thus describes optimality conditions through two coupled continuous-time processes: the physical system states x, and the co-states λ, which dictate the optimal control inputs. These systems are inherently coupled: the co-states generate actions that drive the physical environment, while the evolution of the environment continuously updates the co-states via the Hamiltonian gradients.

The central premise of this work is that this coupled structure mirrors the abstraction of recurrent reinforcement learning. In partially observable RL, recurrent policies aim to control an environment x by generating actions from an internal, latent dynamical process h. However, standard practice treats this hidden state as an uninterpretable black box. We propose the view that to achieve optimal control, the latent recurrent process $h(t)$ should act as a high-dimensional neural embedding of the theoretically optimal co-state $\lambda^\star(t)$.

This analogy provides a mathematical blueprint for designing recurrent architectures. To formalize this connection, we introduce the neural co-state policy (NCP) class that mimics the theoretical optimal Hamiltonian system through learned neural representations.

Definition 3.2 (Neural Co-state Policy)**.** A neural co-state policy (NCP) is a system defined by:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \qquad y = \phi(x) + v, \\ \dot{h} &= -B_\theta(y) - F_\theta(y, u)h \\ u &= \operatorname*{argmin}_{u \in \mathcal{U}} \{c(u) + u^\top G_\theta(y)h\} \\ \text{s.t.} \quad x(t_0) &= x_0, \quad \text{and} \quad h(t_f) = \nabla_y \Phi(y(t_f)), \end{aligned} \tag{3.3}$$

where $h \in \mathbb{R}^{d_h}$ acts as the memory state of the network. The policy class separates various learnable components to represent the unknown environmental dynamics: F_θ models the state transition Jacobians $\nabla_x \dot{x}$, G_θ models the control-influence matrix $g(x)$, and B_θ models the state cost derivative $\nabla_x q(x)$. By defining the system strictly through its initial state $x(t_0)$ and its terminal hidden state $h(t_f)$, the NCP forms a two-point boundary value problem (TPBVP).

Assuming a standard continuous action space and a quadratic control penalty $c(u) = u^\top R u$ (where $R \succ 0$), evaluating $\nabla_u \mathcal{H} = 0$ yields a closed-form optimal control law for the NCP: $u^\star = -\frac{1}{2} R^{-1} G_\theta(y) h^\star$.

Proposition 3.3 (NCP Optimality Condition)**.** *Consider the optimal Hamiltonian system from Proposition 3.1 and the NCP system from Definition 3.2. For $\theta^\star$ to yield optimal trajectories, it is a necessary condition for the NCP components to act as functional representations of the true Hamiltonian terms:*

$$B_{\theta^\star}(y) \approx \nabla_x q(x), \qquad F_{\theta^\star}(y, u) \approx \nabla_x \dot{x}, \qquad G_{\theta^\star}(y) \approx g(x)^\top.$$

Consequently, the optimal latent state $h(t)$ is encouraged to converge to a structural embedding of the true theoretical co-state $\lambda^\star(t)$.

Optimal Hamiltonian system

$$\dot{x}^\star = f(x^\star) + g(x^\star)u^\star,$$
$$\dot{\lambda}^\star = -\nabla_x q - (\nabla_x \dot{x}^\star)\lambda^\star,$$
$$u^\star = \underset{u\in\mathcal{U}}{\operatorname{argmin}}\{c(u) + u^\top g(x^\star)^\top \lambda^\star\},$$
$$\text{s.t.} \quad x^\star(t_0) = x_0,$$
$$\lambda^\star(t_f) = \nabla_x \Phi(x^\star(t_f)).$$

Neural Co-state Policy

$$\dot{x}^\star = f(x^\star) + g(x^\star)u^\star, \quad y^\star = \phi(x^\star) + v,$$
$$\dot{h}^\star = -B_\theta(y^\star) - F_\theta(y^\star, u^\star)h^\star,$$
$$u^\star = \underset{u\in\mathcal{U}}{\operatorname{argmin}}\{c(u) + u^\top G_\theta(y^\star)h^\star\},$$
$$\text{s.t.} \quad x^\star(t_0) = x_0,$$
$$h^\star(t_f) = \nabla_y \Phi(y^\star(t_f)).$$

Figure 2: **Structural parallel between the optimal Hamiltonian system (left) and the NCP (right)**: an optimal NCP system will learn a representation such that the hidden state $h^\star$ is a function approximator of the optimal co-state $\lambda^\star$ of the optimal Hamiltonian system. This results in a policy that follows an optimal dynamic control law via a latent continuous-time process.

This proposition establishes our theoretical goal: if the neural hidden state $h(t)$ can be explicitly regularized to functionally align with the true co-state $\lambda^\star(t)$, the policy naturally recovers the theoretically optimal control law through its learned latent projections. Figure 2 makes this alignment explicit.

3.3 The Control-Hamiltonian Structure of Recurrent Networks

The NCP framework does not necessarily demand a fundamentally new network architecture; rather, it provides a unifying mathematical lens through which to reinterpret existing models. To demonstrate this, consider the optimal latent dynamics and corresponding actions defined by the NCP class:

$$\text{(NCP)} \quad \dot{h}^\star = -B_\theta(y) - F_\theta(y, u^\star)h^\star, \quad \text{and} \quad u^\star = -\frac{1}{2}R^{-1}G_\theta(y)h^\star. \tag{3.4}$$

We can examine standard architectures, such as CT-RNN and GRU, against these theoretical conditions. By isolating the core affine transformations that drive their hidden state updates (omitting time-constants and leak rates for structural clarity), we observe the following algebraic parallel:

$$\text{(CT-RNN)} \quad \dot{h} \propto \phi\big(W_{\text{in}}y + W_h h + b\big), \qquad \text{and} \quad u = W_{\text{out}}h \tag{3.5}$$

$$\text{(GRU)} \quad \dot{h} \propto \phi\big(W_{\text{in}}y + W_h(r \odot h) + b\big), \qquad \text{and} \quad u = W_{\text{out}}h \tag{3.6}$$

where $\phi(\cdot)$ denotes the non-linear activation function and r is the GRU reset gate (Chung et al., 2014).

While these standard equations apply non-linearities over the hidden state differential and include bias terms, their underlying affine structure inherently mirrors the coupled systems of PMP. By treating the learned weights as state-dependent matrix operators, we can extract a direct functional mapping to the NCP dynamics:

- **The state-cost gradient (B_θ):** The input projection matrix $W_{\text{in}}y$ processes the current observation, acting as the structural equivalent to the marginal state cost $-B_\theta(y)$. Notably, if the environment's true state cost $q(x)$ is quadratic, its gradient is strictly linear. This makes standard RNNs well-suited to capture optimal control in tasks with quadratic reward formulations.
- **The dynamics Jacobian (F_θ):** The recurrent weight matrices W_h serve the role of the dynamics Jacobian $-F_\theta$, capturing the state-to-state sensitivity of the system over time.
- **The readout layer (Hamiltonian minimization):** In the PMP framework, optimal control requires minimizing the Hamiltonian. If we approximate the control-influence mapping $G_\theta(y)$ as a static matrix G_θ, the Hamiltonian minimization term $-\frac{1}{2}R^{-1}G_\theta$ collapses into a single constant matrix. The linear readout layer $u = W_{\text{out}}h$ can then be interpreted as a closed-form execution of Hamiltonian minimization.

For the recurrent matrices to correctly integrate the theoretical co-state, they must explicitly contain a representation of the local system dynamics (capturing the state derivatives via F_θ and the control-influence via G_θ). This requirement directly reflects the core premise of the *good regulator theorem* (Conant & Ross Ashby, 1970) and the *internal model principle* (Francis & Wonham, 1976) in the context of ordinary differential equations (ODEs). This posits that any strictly optimal controller must inherently contain a model of the system it regulates.

Algorithm 1 Neural Co-state Optimization for Recurrent Networks for PPO

Initialize: actor parameters $\theta = \{E_\theta, \text{GRU}_\theta, W_{\text{out}}\}$, critic parameters $\phi = \{V_\phi\}$, batch size M, loss coefficients c_1, c_2, c_3.
for iteration $= 1, 2, \ldots$ **do**
  Initialize hidden state h_0
  Run policy in environment for T timesteps, collecting (y_t, h_t, u_t, r_t):
    Encode observation: $\tilde{y}_t = E_\theta(y_t)$
    Integrate hidden state: $h_t = \text{GRU}_\theta(\tilde{y}_t, h_{t-1})$
    Apply implicit minimum principle: $\mu_t = \text{argmin}_{u \in \mathcal{U}} \mathcal{H} = W_{\text{out}} h_t$
    Sample action $u_t \sim \mathcal{N}(\mu_t, \sigma)$
  Compute advantage estimates $\hat{A}_t$ using critic $V_\phi(\tilde{y}_t, h_t)$
  Compute HJB co-state targets $\hat{\lambda}_t = \text{stop\_gradient}(\nabla_{\tilde{y}} V_\phi(\tilde{y}_t, h_t))$
  Optimize joint PPO objective with K epochs and minibatch size M:
    $\mathcal{L}_{\text{total}}(\theta, \phi) = \mathcal{L}_{\text{actor}} + c_1 \mathcal{L}_{\text{critic}} - c_2 \mathcal{L}_{\text{entropy}} + c_3 \mathcal{L}_{\text{co-state}}$
    where $\mathcal{L}_{\text{co-state}}(\theta) = \mathbb{E}_t \left[1 - \frac{h_t \cdot \hat{\lambda}_t}{\|h_t\|_2 \|\hat{\lambda}_t\|_2} \right]$
end for

4 Structuring Internal Dynamics in Recurrent Policies

In standard model-free RL, however, policies are optimized purely via scalar reward maximization. While the recurrent architectures discussed in the previous section possess the structural capacity to represent neural co-states, pure policy optimization provides no guarantee that their hidden states will actually converge to the optimal deterministic PMP co-state representations during learning. To bridge this optimization gap, we introduce a neural co-state loss that aligns the hidden states of the network with the theoretical co-states. By leveraging standard actor-critic methods, a co-state loss is derived that is applicable to any recurrent policy that belongs to the NCP class.

4.1 The Actor-Critic Optimization Framework

Standard recurrent RL operates via the actor-critic paradigm, utilizing an actor (π_θ) to selects actions and a critic (V_ϕ) to estimate returns. Optimization commonly relies on PPO (Schulman et al., 2017) alongside generalized advantage estimation (GAE) (Schulman et al., 2015). PPO stabilizes updates by clipping the objective function to prevent destructively parameter shifts, while GAE reduces policy gradient variance by exponentially smoothing the critic's temporal difference errors.

To optimize the recurrent parameters over finite trajectories, gradients are computed recursively backwards through the unrolled network via back-propagation through time (BPTT). The BPTT error propagation takes the standard form:

$$\delta_{t-1} = \delta_t \frac{\partial h_t}{\partial h_{t-1}} + \frac{\partial \mathcal{L}}{\partial h_{t-1}}, \tag{4.1}$$

where δ_t is the accumulated error gradient of the objective function with respect to the hidden state at time t. Mathematically, this recursive gradient chaining is the discrete-time computational equivalent of integrating the co-state (or adjoint) equation.

4.2 Extracting Co-state Targets from HJB

While standard actor-critic algorithms excel at credit assignment, they optimize recurrent parameters purely for expected return maximization, leaving the differential structure of the hidden state unconstrained. To align the hidden state with the theoretical PMP co-state, we bridge PMP with dynamic programming via the Hamilton-Jacobi-Bellman (HJB) equation. A foundational property linking these frameworks is that the optimal continuous-time co-state is strictly equivalent to the spatial gradient of the optimal value function: $\lambda^\star(t) = \nabla V^\star(x(t))$.

In the actor-critic paradigm, the learned critic network $V_\phi(y_t)$ approximates this true value function. Using automatic differentiation, we can directly compute its gradient with respect to the encoded environment observation to obtain a functional co-state approximation: $\hat{\lambda}_t = \nabla_{\tilde{y}} V_\phi(\tilde{y}_t)$, where $\tilde{y} = E(y_t)$ is the encoded observation.

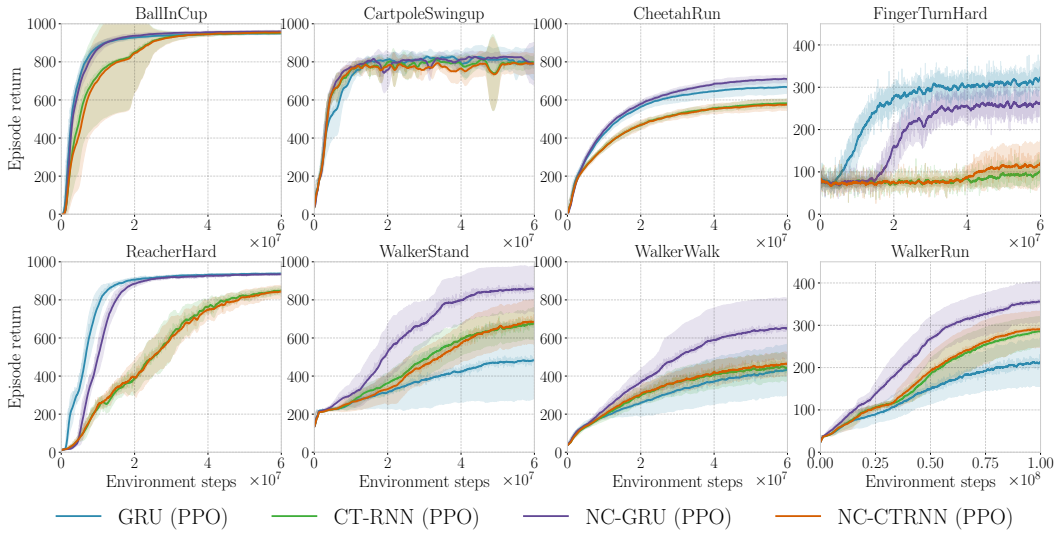

Figure 3: **DMControl tasks under partial observability.** Learning curves for standard and NCP-regularized recurrent architectures (GRU and CT-RNN) evaluated under a sensor dropout regime. During training, the entire observation vector is zero-masked at each timestep with probability $p = 0.5$. Solid lines denote the mean episode return across 10 independent random seeds, and shaded regions represent ± 1 standard deviation. For visual clarity, curves are smoothed using a simple moving average over 30 logged checkpoints (equivalent to roughly 10^6 environment steps).

Because theoretical co-states can take on arbitrarily large magnitudes while recurrent hidden states are typically bounded by non-linear activations, minimizing the unnormalized Euclidean distance can cause instability. Therefore, we regularize the directional alignment using the cosine distance:

$$\mathcal{L}_{\text{co-state}}(\theta) = \mathbb{E}_t \left[1 - \frac{h_t \cdot \hat{\lambda}_t}{\|h_t\|_2 \|\hat{\lambda}_t\|_2} \right] \tag{4.2}$$

Algorithm 1 summarizes our approach to neural co-state optimization.

5 Experiments and Results

We empirically evaluate the NCP framework across 8 continuous control tasks from the DeepMind Control Suite (Tassa et al., 2018), using the PPO framework. To explicitly test the memory capacity of the learned latent dynamics, we introduce partial observability via a stochastic sensor dropout mechanism. At each timestep t, the entire observation vector y_t is completely masked through a scalar mask $m_t \sim \text{Bernoulli}(1-p)$, resulting in the corrupted observation $\tilde{y}_t = m_t y_t$. This total sensor blackout requires the recurrent policy to integrate past interactions to handle sensor dropouts. All models are trained with a fixed masking probability ($p_{\text{train}} = 0.5$). Experimental details are described in Appendix E, while the implementation code is open-source available at `github.com/DavidLeeftink/neural-costate-policies`.

Through this design, our experiments address three core questions: (1) **Performance:** does the co-state loss improve sample efficiency and expected returns? (2) **Sensitivity:** how sensitive is training stability to the co-state loss coefficient? (3) **Robustness:** do NCP hidden states generalize better in zero-shot learning with out-of-distribution (OOD) masking?

5.1 Continuous Control and Zero-Shot Robustness on the DeepMind Control Suite

Addressing (1) **Performance**, we benchmarked NC-GRU and NC-CTRNN against their unregularized counterparts (Figure 3). The impact of the PMP-derived co-state loss varies across task complexity. On simpler, lower-dimensional tasks (e.g., *CartpoleSwingup*, *BallInCup*), all models successfully solve the environment. However, the auxiliary co-state regularization introduces a marginally slower

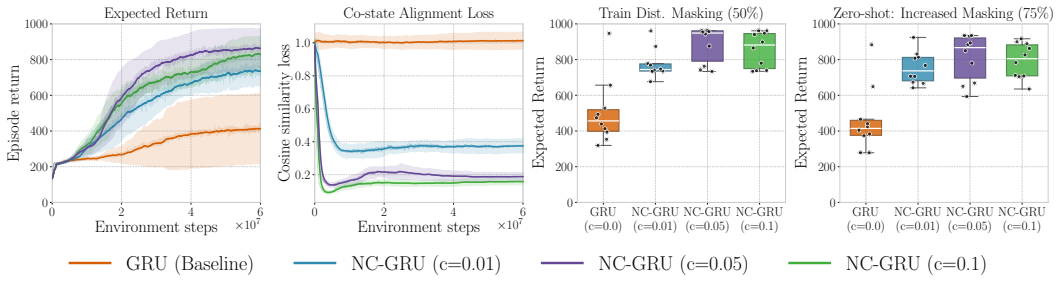

Figure 4: **Ablation of the co-state penalty coefficient on the WalkerStand task.** *(Left)* Expected returns during training under 50% observation masking. *(Middle left)* Co-state cosine similarity loss, demonstrating the structural alignment of the latent space over time. *(Middle right)* Final performance distributions evaluated within the training distribution. *(Right)* Out-of-distribution zero-shot robustness evaluated under 75% sensor masking. Solid lines denote the mean episode return across 10 independent random seeds, and shaded regions represent ± 1 standard deviation. For visual clarity, curves are smoothed using a simple moving average over 30 logged checkpoints (equivalent to roughly 10^6 environment steps). Boxplots aggregate 100 evaluations per seed across varying initial conditions, with white horizontal lines indicating the median.

initial convergence for the NCP variants compared to the unregularized baselines. On *FingerTurnHard*, all architectures struggle to achieve high returns, showing that dexterous manipulation under observation masking remains challenging. For this task, the co-state loss marginally weakens the performance compared to the standard GRU. Lastly, on locomotion tasks (*WalkerStand*, *WalkerWalk*, and *WalkerRun*, *CheetahRun*), the NC-GRU model yields substantial improvements in both final asymptotic return and overall stability. Whereas the NC-GRU is outperforming the standard GRU on most cases, the NC-CTRNN achieves mostly similar performance to the original CTRNN policy, suggesting the co-state loss term is not as effective for this architecture.

Analysis of the results suggests that the co-state priors perform well at regularizing rhythmic locomotion, however the performance degrades in contact-heavy tasks like *FingerTurnHard*. This challenge possibly could be related to extracting the co-state targets using the w.r.t. the noisy partial derivative observation rather than the true state, which forms a challenge for continuous-time co-state tracking.

5.2 The effect of the co-state loss coefficient

To address (2) **Sensitivity**, we compare the standard GRU model against the neural co-state GRU model with co-state coefficients of $c_3 \in \{0.01, 0.05, 0.1\}$ on the *WalkerStand* task, while we use the Brax (Freeman et al., 2021) parameter configurations for the remaining training coefficients. Figure 4 shows that incorporating the co-state loss during training substantially improves the expected returns.

Unlike the standard GRU that plateaus early, all co-state configurations improve performance, peaking optimally at $c_3 = 0.05$ before degrading at $c_3 = 0.1$. Larger coefficients accelerate the reduction of co-state alignment loss, ensuring tighter adherence to the optimal control prior. In contrast, the unregularized GRU fails to naturally align, maintaining a high loss near one throughout training despite improving expected returns.

To address (3) **Robustness**, we evaluate model resilience in a zero-shot setting by increasing observation masking from 50% to 75%. While all models experience performance degradation under this out-of-distribution regime, the NC models do not demonstrate inherent robustness to the increased masking frequency itself. However, the policies are able to largely retain the higher median returns inherited from its superior training-time performance. This suggests that while co-state regularization significantly elevates the agent's operating point, the resulting performance floor in difficult settings is a direct reflection of the gains achieved during the training distribution.

6 Related work

Recurrent policies and hidden state representations. Moving beyond explicit belief-state tracking, modern deep RL encodes historical context as an unstructured latent state via RNNs. To stabilize

long-horizon credit assignment, recent architectures introduce inductive biases, such as structured state space models (Gu et al., 2021) or continuous-time oscillatory dynamics (Rusch & Mishra, 2020). While these methods enforce a general *dynamical* prior, NCPs enforce an *optimal control* prior. Although earlier work has considered the relationship between optimal control and neural optimization (Liu et al., 2021; Bensoussan et al., 2023), NCPs are unique by bridging neural memory and classical control theory; regularizing the latent space to track the necessary optimality conditions.

Physics-informed latent representations and internal forward models. This control-theoretic regularization aligns with physics-informed machine learning, where architectures like Hamiltonian neural networks (Greydanus et al., 2019) and deep Lagrangian networks (Lutter et al., 2019) embed physical conservation laws. Whereas these works primarily model *passive* environmental dynamics, NCPs explicitly structure the agent's *internal* belief state to realize the internal model principle (Francis & Wonham, 1976).

Pontryagin's Minimum Principle in Deep RL. While the Hamilton-Jacobi-Bellman (HJB) equation forms the theoretical backbone of value-based RL, its trajectory-centric counterpart, PMP, has only recently gained traction. PMP in RL has been restricted strictly to model-based paradigms: offline policy evaluation (Jin et al., 2020), deterministic trajectory optimization (Gu et al., 2022; Eberhard et al., 2025) and planning under uncertainty (Leeftink et al., 2025). NCPs diverge from this lineage by operating fundamentally model-free, requiring only a targeted co-state loss to ground closed-loop recurrent policies in optimal control theory.

7 Discussion

In this work, we established a structural correspondence between the hidden states of recurrent reinforcement learning policies and the theoretical co-states of the Pontryagin minimum principle. By conceptualizing standard architectures like CT-RNNs and GRUs as dynamic processes governed by Hamiltonian minimization, we introduced the neural co-state policy (NCP) framework. Our results demonstrate that extracting PMP co-state targets from the HJB value gradient provides a tractable auxiliary loss that successfully grounds internal memory in optimal control theory, matching or improving performance on partially observable continuous control tasks.

Limitations and future work. While bridging a critical theoretical gap, NCPs derive the co-state targets via the critic's spatial gradient. Because this relies on automatic differentiation, these targets can be noisy — a known vulnerability in deterministic policy gradients (Lillicrap et al., 2016). A conceptual limitation is that standard recurrent architectures require bounded non-linear activations for numerical stability, deviating from the unconstrained integration of theoretical PMP co-states.

This motivates three avenues for future research. First, the NCP equivalence framework can be extended to identify and map broader classes of memory architectures, such as coupled oscillator or long-short term memory (LSTM; Hochreiter & Schmidhuber (1997)) models. Second, theoretically principled readout layers can be designed to explicitly solve time- and fuel-optimal tasks that require discontinuous bang-bang or bang-off-bang controls (App. C). Finally, integrating probabilistic value network ensembles allows for increased data-efficiency, providing optimal dynamic representations under epistemic uncertainty (App. D).

Broader Impact. Beyond algorithmic control, our framework shares deep conceptual connections with biological motor control. The human brain operates as an exceptionally intricate recurrent policy, to resolve partial observability in continuous motor tasks (Mastrogiuseppe & Ostojic, 2018; Tsay & Ivry, 2026). Viewing the brain through the lens of NCPs could provide a theoretical framework for understanding how biological networks encode dynamic optimality. Furthermore, continuous control under partial observability is the defining challenge of physical robotics (Schneider et al., 2022). Real-world tasks such as bipedal locomotion, dextrous manipulation, and autonomous navigation are inherently plagued by noisy sensors, temporary occlusions, and hardware latency. By grounding the memory state of the policy in the mathematics of optimal control, NCPs offer a principled path forward for robotic control. We discuss broader societal impact in Appendix A.

Ultimately, this work demonstrates that previously black-box neural memory can be anchored in the mathematics of the minimum principle, providing a new theoretical perspective on recurrent computation in continuous control.

Acknowledgements

This publication is part of the project ROBUST: Trustworthy AI-based Systems for Sustainable Growth with project number KICH3.LTP.20.006, which is (partly) financed by the Dutch Research Council (NWO), ASMPT, and the Dutch Ministry of Economic Affairs and Climate Policy (EZK) under the program LTP KIC 2020-2023. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Beer, R. D. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

Bellman, R. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Bensoussan, A., Han, J., Yam, S. C. P., and Zhou, X. Value-gradient based formulation of optimal control problem and machine learning algorithm. *SIAM Journal on Numerical Analysis*, 61(2): 973–994, 2023.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Katariya, Y., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Bryson, A. E. and Ho, Y.-C. *Applied Optimal Control: Optimization, Estimation, and Control*. Taylor & Francis, 1975.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Conant, R. C. and Ross Ashby, W. Every good regulator of a system must be a model of that system. *International Journal of Systems Science*, 1(2):89–97, 1970.

Eberhard, O., Vernade, C., and Muehlebach, M. A pontryagin perspective on reinforcement learning. In *Proceedings of the Seventh Annual Learning for Dynamics & Control Conference*, volume 283 of *Proceedings of Machine Learning Research*, pp. 233–244, 2025.

Francis, B. A. and Wonham, W. M. The internal model principle of control theory. *Automatica*, 12 (5):457–465, 1976.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Gu, C., Xiong, H., and Chen, Y. Pontryagin optimal control via neural networks. *arXiv preprint arXiv:2212.14566*, 2022.

Hausknecht, M. J. and Stone, P. Deep recurrent Q-learning for partially observable MDPs. In *AAAI fall symposia*, volume 45, pp. 141, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., and Wang, W. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022.

Jin, W., Wang, Z., Yang, Z., and Mou, S. Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems*, 33: 7979–7992, 2020.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Kirk, D. E. *Optimal Control Theory: An Introduction*. Courier Corporation, 2004.

Leeftink, D., Yildiz, C., Ridderbusch, S., Hinne, M., and Van Gerven, M. Optimal control of probabilistic dynamics models via mean Hamiltonian minimization. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pp. 4146–4153, 2025. doi: 10.1109/CDC57313.2025.11312001.

Liberzon, D. Calculus of variations and optimal control theory: a concise introduction. 2011.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, 2016.

Liu, G.-H., Chen, T., and Theodorou, E. Dynamic game theoretic neural optimizer. In *International Conference on Machine Learning*, pp. 6759–6769. PMLR, 2021.

Lutter, M., Ritter, C., and Peters, J. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

Mastrogiuseppe, F. and Ostojic, S. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.

Pontryagin, L. S. *Mathematical Theory of Optimal Processes*. Routledge, 1st edition, 1987. doi: 10.1201/9780203749319.

Rusch, T. K. and Mishra, S. Coupled oscillatory recurrent neural network (CORNN): An accurate and (gradient) stable architecture for learning long time dependencies. *arXiv preprint arXiv:2010.00951*, 2020.

Schneider, T., Belousov, B., Abdulsamad, H., and Peters, J. Active inference for robotic manipulation. *arXiv preprint arXiv:2206.10313*, 2022.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind Control Suite. *arXiv e-prints*, art. arXiv:1801.00690, January 2018. doi: 10.48550/arXiv.1801.00690.

Tsay, J. S. and Ivry, R. B. Cerebellar contributions to action and cognition: Prediction, timescale, and continuity. *Proceedings of the National Academy of Sciences*, 123(15):e2524258123, 2026.

Vinter, R. Is the costate variable the state derivative of the value function? In *1986 25th IEEE Conference on Decision and Control*, pp. 1988–1989. IEEE, 1986.

Wierstra, D., Förster, A., Peters, J., and Schmidhuber, J. Recurrent policy gradients. *Logic Journal of IGPL*, 18(5):620–634, 2010.

Appendix

A Impact Statement

This work provides a foundation for reinforcement learning under partial observability. The proposed theoretical link makes previously black-box policies more interpretable and improves their performance. Practically, controllers capable of sustaining stable behavior partial observations are vital for safety-critical autonomous systems, such as robotics and medical devices.

Naturally, advancements in continuous control carry dual-use implications. Algorithms capable to tolerate sensor failure are equally applicable to military robotics and autonomous weapons. Although our current focus is strictly on theoretical algorithmic design and simulated benchmarks, translating these highly resilient RL agents to the real world will require careful, domain-specific oversight to prevent misuse.

B Optimal Hamiltonian System

B.1 Additional background on co-states and the control-Hamiltonian

Repeating the main text for completeness, the control-Hamiltonian expresses how the cost-to-go changes over time with the dynamics:

$$\mathcal{H}(x, \lambda, u) \coloneqq \mathcal{L}(x, u) + \lambda^\top \dot{x} = q(x) + c(u) + \lambda^\top \big(f(x) + g(x)u\big). \tag{B.1}$$

Here, $\lambda \in \mathbb{R}^{d_x}$ are referred to as co-states, and encode the sensitivity of the value of a state with respect to the dynamics. These follow from solving the constrained optimization problem in Eq. 2.4 using the Lagrange multiplier function $\lambda(t)$ for $t \in [t_0, t_f]$. By defining the differential function of the co-states, we obtain a coupled ODE:

$$\dot{x}(t) = \nabla_\lambda \mathcal{H}(x, \lambda, u) = f(x(t)) + g(x(t))u, \tag{B.2}$$

$$\dot{\lambda}(t) = -\nabla_x \mathcal{H}(x, \lambda, u) = -\nabla_x q - (\nabla_x \dot{x})\, \lambda(t) \tag{B.3}$$

for $t \in [t_0, t_f]$, where the notation $(\nabla_y f)_{i,j} \coloneqq \partial f_j / \partial y_i$ denotes Jacobians, in this case $\nabla_x \mathcal{H} = (\frac{\partial \mathcal{H}}{\partial x_1}, \ldots, \frac{\partial \mathcal{H}}{\partial x_{d_x}})^\top$. The interaction dynamics between the states and co-states form a coupled dynamical system, which we refer to as the Hamiltonian system. Since the co-states incorporate the cost function, this representation lends itself well to the inclusion of a notion of optimality. The minimum principle by Pontryagin (1987), for historical reasons also referred to as the maximum principle, states:

$$u^\star = \operatorname*{argmin}_{u \in \mathcal{U}} \mathcal{H}(x^\star, \lambda^\star, u) \tag{B.4}$$

This tells us that if the optimal control $u^\star(t)$ is applied, producing corresponding unique optimal paths $x^\star(t)$ and $\lambda^\star(t)$, the Hamiltonian function is minimized at all time points $t \in [t_0, t_f]$. This is a necessary but not a sufficient condition of optimality, implying the following optimal system:

B.2 Derivation of the optimal Hamiltonian system

In this section, we provide the derivation of the necessary optimality conditions for the deterministic continuous-time control problem, drawing on standard results from the calculus of variations (Bryson & Ho, 1975; Kirk, 2004). For brevity, we consider an unbounded control set $\mathcal{U} = \mathbb{R}^{d_u}$ which we approach in continuous-time by extremizing the Gateaux derivative. For bounded control sets, the proof is significantly more involved and involves the *needle-variation*, originally proposed by Pontryagin (1987). We refer the reader to Chapter 4 in Liberzon (2011) for the needle-variation proof.

Let $u^\star(t)$ be an optimal control that minimizes the cost functional $J(u)$ over the set of admissible controls $\mathcal{U}$. Let the optimal trajectory be $x^\star(t)$, with corresponding co-states $\lambda^\star(t)$, and let the optimal Hamiltonian be evaluated as $\mathcal{H}^\star \coloneqq \mathcal{H}(x^\star(t), \lambda^\star(t), u^\star(t))$. A necessary condition for optimality is that the gradient of the Hamiltonian with respect to the control vanishes.

Proof. To enforce the dynamic equality constraint $\dot{x} = f(x) + g(x)u$, we introduce the Lagrange multiplier function $\lambda(t)$ (the co-state) that ensures this constraint for all $t \in [t_0, t_f]$. Because the

dynamic constraint equals zero along any valid trajectory, adding it to the integral does not change the value of $J(u)$:

$$J(u) = \Phi(x(t_f)) + \int_{t_0}^{t_f} \Big[q(x) + c(u) + \lambda(t)^\top \Big(f(x) + g(x)u - \dot{x} \Big) \Big] \, dt. \tag{B.5}$$

By substituting the definition of the control-Hamiltonian, $\mathcal{H}(x, \lambda, u) = q(x) + c(u) + \lambda^\top (f(x) + g(x)u)$, we can rewrite the objective as:

$$J(u) = \Phi(x(t_f)) + \int_{t_0}^{t_f} \big[\mathcal{H}(x, \lambda, u) - \lambda(t)^\top \dot{x} \big] \, dt. \tag{B.6}$$

We consider the Gâteaux derivative of this cost functional:

$$\delta J(u) = \lim_{\epsilon \to 0} \frac{J(u + \epsilon \delta u) - J(u)}{\epsilon}, \tag{B.7}$$

where $\delta u(t)$ is an arbitrary control variation defined over $[t_0, t_f]$. The first variation can be obtained by isolating the first-order terms around the optimal solution $u^\star(t)$:

$$J(u^\star + \epsilon \delta u) - J(u^\star) \approx \delta J|_{u^\star}(\delta u)\epsilon, \tag{B.8}$$

where higher-order terms are neglected as $\epsilon \to 0$. Since, by definition, $u^\star(t)$ is a minimum for J, its first variation must be zero. Note that a variation in the control $\delta u(t)$ naturally induces a corresponding variation in the state trajectory, denoted as $\delta x(t)$. A well-known result in optimal control theory (Kirk, 2004, Eq. 5.1.13) states that to successfully absorb the variations in the state trajectory $\delta x(t)$, the co-state multiplier $\lambda(t)$ must be chosen to satisfy the differential equation:

$$\dot{\lambda}^\star(t) = -\nabla_x \mathcal{H}(x^\star, \lambda^\star, u^\star), \tag{B.9}$$

with the terminal transversality condition $\lambda^\star(t_f) = \nabla_x \Phi(x^\star(t_f))$. With the state variations eliminated by this specific choice of optimal co-state dynamics, the remaining first variation of the cost with respect to the control is given by (Kirk, 2004, Eq. 5.1.14):

$$\delta J|_{u^\star}(\delta u) = \int_{t_0}^{t_f} \left[\nabla_u \mathcal{H}^\star \right]^\top \delta u(t) dt. \tag{B.10}$$

By the fundamental lemma of the calculus of variations, since $\delta u(t)$ is an arbitrary variation, the integral can only equal zero if the integrand itself vanishes identically. It follows that:

$$\nabla_u \mathcal{H}^\star = 0, \quad \text{for all } t \in [t_0, t_f]. \tag{B.11}$$

For unconstrained control sets, this stationary condition defines the optimal control law. For bounded, closed control sets $\mathcal{U}$, this local stationary condition is generalized by the PMP to the global minimization of the Hamiltonian across the action space:

$$u^\star(t) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \, \mathcal{H}(x^\star(t), \lambda^\star(t), u). \tag{B.12}$$

This establishes the optimality condition for the control sequence.

C Readout Layer Design via Optimal Control Principles

In standard deep RL architectures of the actor, a readout layer (or policy head) that maps the final hidden state to an action is typically chosen heuristically. However, because the NCP class explicitly aims to align the hidden state with the theoretical Pontryagin co-state ($h \approx \lambda^\star$), the optimal readout layer is no longer arbitrary but instead has to minimize the control-Hamiltonian function with respect to the specific running cost function of the environment.

By defining the optimal switching function as $\sigma(t) = -g(x, t)^\top \lambda(t)$ and assuming symmetric actuator limits $\mathcal{U} \in [-u_{\max}, u_{\max}]$ for notational clarity, we can structurally design the policy head to match different classes of optimal control problems (illustrated in Fig. 5):

1. Quadratic Control (Smooth Action): As explored in the main text, environments with a quadratic penalty on control effort, $\mathcal{L}(x, u) = q(x) + u^\top R u$ and control-affine system dynamics

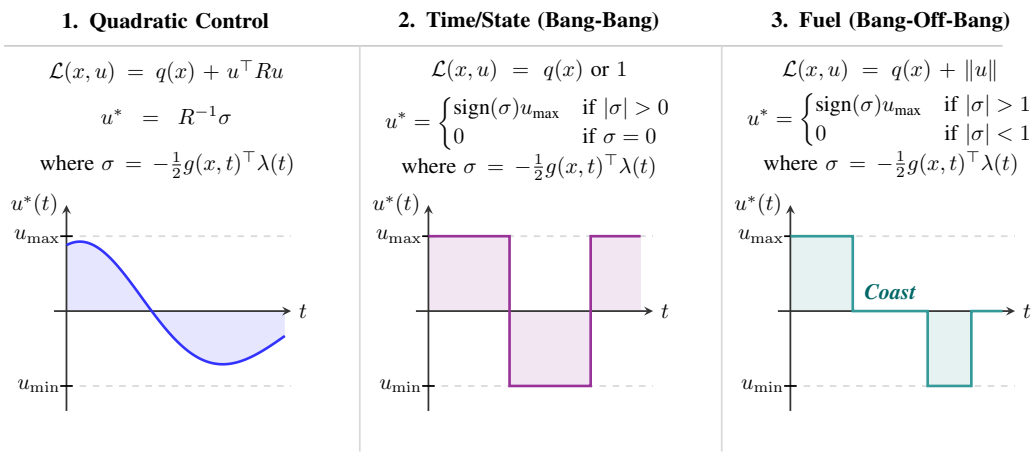

Figure 5: **Hamiltonian policies under varying cost functions:** Energy-optimal (1), time-optimal (2), and fuel-optimal (3). This can produce either smooth optimal control functions or discontinuous bang-bang controls. All of them leverage the same co-states, and can thus can serve as an 'actor' equivalent to the one that leverages the latent neural co-state.

$\dot{x} = f(x) + g(x)u$, yield a Hamiltonian that is convex with respect to u. Setting the gradient to zero results in the optimal control law: $u^\star = R^{-1}\sigma$. In our framework, this corresponds to a standard linear readout layer without bounding activation functions.

2. Time-Optimal and State-Only Costs (Bang-Bang Control): For many robotics tasks (e.g., minimum-time navigation), the cost consists only of state-dependent terms, meaning control effort is not penalized: $\mathcal{L}(x,u) = q(x)$. Because the Hamiltonian is linear with respect to u, and physical actuators possess strict limits $[u_{\min}, u_{\max}]$, the minimum principle dictates that the optimal solution lies exclusively on the boundaries. This results in *bang-bang* control, where the agent switches instantaneously between maximum and minimum effort based on the sign of the switching function:

$$u^\star = \begin{cases}\text{sign}(\sigma)u_{\max} & \text{if } |\sigma| > 0\\ 0 & \text{if } \sigma = 0\end{cases}$$

In the NCP framework, a bang-bang optimal controller can be enforced simply by applying a $\text{sign}(\cdot)$ activation function to the policy head.

3. Fuel-Optimal Costs (Bang-Off-Bang Control): If the environment penalizes the absolute magnitude of the control effort (e.g., conserving fuel in aerospace applications), an L_1 penalty is introduced: $\mathcal{L}(x,u) = q(x) + \|u\|$. The optimal control law develops a deadzone where it is optimal to coast (apply no control effort) when the sensitivity is low:

$$u^\star = \begin{cases}\text{sign}(\sigma)u_{\max} & \text{if } |\sigma| > 1\\ 0 & \text{if } |\sigma| < 1.\end{cases}$$

While the empirical evaluations in this work focus on the continuous quadratic case, the interpretation of recurrent hidden states as Pontryagin co-states allows one to design readout layers specific to the properties of the cost or reward function of the environment while leaving the underlying recurrent co-state dynamics unchanged.

D Epistemic Uncertainty and Mean Hamiltonian Minimization

An agent attempting to make an intelligent trade-off between exploration and exploitation must focus its exploration on *epistemic* uncertainty, associated with what the agent does not know yet. Unlike aleatoric uncertainty, which represents inherent and irreducible randomness, epistemic uncertainty can be reduced by gathering more data. In optimal control, epistemic uncertainty is often modeled as parametric uncertainty, obtained through Bayesian inference over the environment's dynamics. We can formulate this by considering a policy that aims to minimize the expected cost over a posterior

distribution of plausible dynamic models:

$$\pi^\star \coloneqq \operatorname*{argmin}_{\pi\in\Pi} \mathbb{E}_{\zeta\sim p(\zeta|\mathcal{D})}\left[J(\pi_\theta(x_\zeta(t)))\right] \tag{D.1}$$
$$\text{s.t.} \quad \dot{x}_\zeta(t) = f_\zeta\big(x_\zeta(t), u(t), t\big), \quad \text{for } t \in [t_0, t_f]$$
$$x_\zeta(t_0) = x_0.$$

where $J(\theta)$ denotes the deterministic cost functional evaluated under dynamics parameterized by ζ, and $p(\zeta \mid \mathcal{D})$ is the posterior distribution given past interaction data $\mathcal{D}$.

To solve this probabilistically robust optimization problem, we can draw a finite set of m samples $\zeta_i \sim p(\zeta \mid \mathcal{D})$. These samples form an ensemble of deterministic dynamical systems driven by a shared control input $u(t)$:

$$\dot{x}_{\zeta_i}(t) = f_{\zeta_i}\big(x_{\zeta_i}(t), u(t), t\big), \tag{D.2}$$
$$\dot{\lambda}_{\zeta_i}(t) = -\nabla_x \mathcal{H}\big(x_{\zeta_i}(t), \lambda_{\zeta_i}(t), u(t), t\big), \tag{D.3}$$
$$\text{s.t.} \quad \lambda_{\zeta_i}(t_f) = \nabla_x \Phi\big(x_{\zeta_i}(t_f)\big). \tag{D.4}$$

If $u^\star(t)$ is an optimal control sequence for this ensemble, it must satisfy the Mean Hamiltonian Minimization principle (Leeftink et al., 2025):

$$u^\star(t) = \operatorname*{argmin}_{u\in\mathcal{U}} \mathbb{E}_{\zeta\sim p(\zeta|\mathcal{D})}\left[\mathcal{H}\left(x_\zeta(t), \lambda_\zeta(t), u, t\right)\right], \tag{D.5}$$

Implications for Neural Co-state Policies. While the NCP framework introduced in this work involves optimizing a deterministic point estimate, Eq. (D.5) provides a theoretical foundation for a *Bayesian* NCP. Because the theoretical co-state is linked to the value function gradient, $\lambda^\star = \nabla_x V^\star$, this uncertainty can be captured by an ensemble of m critics. By maintaining parallel hidden states $h_{\zeta_i}(t)$ across these critics, each representing a unique co-state hypothesis, the actor can take exploratory actions by minimizing the mean of the ensemble Hamiltonians:

$$u^\star(t) \approx \operatorname*{argmin}_{u\in\mathcal{U}} \frac{1}{m}\sum_{i=1}^{m}\left\{c(u) + u^\top G_\theta(y) h_{\zeta_i}(t)\right\}. \tag{D.6}$$

We leave the empirical validation of this Bayesian ensemble approach to future work.

E Implementation and training details

Here we provide an overview of the algorithmic components, architectural choices, and hyperparameters used in our implementation to ensure reproducibility. Our systems and default hyperparameter configurations follow standard continuous control setups, drawing heavily from the Brax environments (Freeman et al., 2021). The full implementation is written in JAX (Bradbury et al., 2018), leveraging `jax.vmap` for vectorized environment rollouts and `jax.lax.scan` for efficient BPTT.

E.1 Network Architecture

Both the NC-GRU and NC-CTRNN share a common actor-critic blueprint. To ensure stable learning, the actor and critic use a shared encoder. They differ only in their recurrent core:

- **Observation Encoder:** At each timestep, the raw observation y_t is dynamically normalized (see Appendix E.2) and passed through a linear encoder followed by a hyperbolic tangent activation: $x_t = \tanh(W_{\text{enc}} y_t + b_{\text{enc}})$.
- **Recurrent Cores:**
  - **GRU:** We use a standard GRU cell mapping the encoded observation and previous hidden state to the next state: $h_t = \text{GRU}(x_t, h_{t-1})$.
  - **CT-RNN:** The continuous-time variant employs leaky integration with a learnable time constant $\alpha = \sigma(\text{log\_alpha})$, initialized to 0 (defaulting to a 0.5 leak rate). The update is defined as: $h_t = (1-\alpha)h_{t-1} + \alpha \tanh(W_{\text{in}} x_t + W_{\text{rec}} h_{t-1})$.

- **Actor-Critic Heads:** The actor outputs the mean of a Gaussian distribution. The log standard deviation ($\log \sigma$) is maintained as a *state-independent* learnable array initialized to zero.
- **Code-Level Initialization:** All linear and recurrent weights use orthogonal initialization. Hidden layers and the critic output use a scale of 1.0, while the actor output layer uses a scale of 0.01 to ensure the initial action distribution is tightly centered around zero, preventing extreme actions early in training. All biases are initialized to 0.

E.2 Algorithmic Components

Our optimization relies on PPO, integrating several standard mechanisms and code-level optimizations for stabilized training described in Huang et al. (2022):

- **Observation and Reward Standardization:** We maintain running statistics (mean and variance) for observations. Raw observations are normalized dynamically and strictly clipped to $[-10, 10]$ before being passed to the networks. Additionally, raw rewards (negative costs) are scaled dynamically by dividing them by the standard deviation of the running discounted returns: $r_t^{\text{scaled}} = r_t^{\text{raw}} / \sqrt{\text{Var}(R) + \epsilon}$, where $\epsilon = 10^{-8}$, and are subsequently clipped to $[-10, 10]$.
- **Advantage Estimation and Minibatch Normalization:** Generalized Advantage Estimation (GAE) is computed backwards through time over the unrolled trajectories. The resulting advantages are normalized (subtracting the mean and dividing by the standard deviation) at the minibatch level during the PPO update epoch, rather than globally over the entire rollout buffer.
- **Sequential Minibatches:** Our implementation constructs minibatches by fetching sequential trajectory segments to maintain the temporal dependencies required for BPTT.
- **Action Clipping:** During environmental interaction, actions sampled from the actor's distribution $a_t \sim \mathcal{N}(\mu_t, \sigma^2)$ are clipped to the valid environmental bounds $[u_{\min}, u_{\max}]$ before being applied to the physics step. The unclipped actions and corresponding log probabilities are stored for the PPO update.
- **Clipped Surrogate Objective & Value Loss:** The policy is updated using the standard clipped PPO objective with a clipping parameter ϵ_{clip}. The value loss also employs clipping to constrain the value function update.

E.3 Co-state loss implementation

Following Algorithm 1, we compute the target co-state λ_{target} is isolated from the computational graph via the `stop_gradient` operator. The PMP loss is computed as the cosine distance between the recurrent hidden state h_t and the target co-state λ_{target}, numerically stabilized by $\epsilon = 10^{-8}$.

E.4 Hyperparameters

The hyperparameter configurations for our systems are detailed in Table 1. These parameters are based on the standard setup described in Freeman et al. (2021) to ensure fair evaluation, specifically adapted for BPTT sequence lengths and continuous control demands. We explicitly utilize the Adam optimizer with an epsilon parameter of 10^{-5}, which significantly stabilizes updates compared to standard defaults.

E.5 Hardware

Our experiments were conducted using NVIDIA RTX 2080 Ti and Quadro RTX 6000 GPUs, paired with Intel Xeon E5-2650 and AMD EPYC 7302 CPUs. Depending on the complexity of the environment, training a single seed for 60 million timesteps constituted between 1 (*Cartpole*) to 6 (*Walker*) hours for the given batch size used.

Table 1: PPO and Network Hyperparameters

Category	Hyperparameter	Value
Architecture	Hidden Layer Size	128
	Initialization Scale	1.0 (General), 0.01 (Actor Out)
	Initialization Type	Orthogonal
Optimization	Optimizer	Adam (via Optax)
	Learning Rate	2.5×10^{-4} (Annealed linearly)
	Epsilon (Adam)	10^{-5}
	Max Gradient Norm	0.5
PPO Details	Rollout Batch Size	32
	Timesteps	60m (100m for WalkerRun)
	Minibatches	4
	PPO Epochs	4
	Discount Factor (γ)	0.99
	GAE Parameter (λ)	0.95
	Clipping Epsilon (ϵ_{clip})	0.2
	Value Loss Coef (c_{vf})	0.5
	Entropy Coef (c_{ent})	0.01
	Co-state Coef (c_{costate})	0.05 (Ablated in experiments)