HIDDEN STATES AS VALUE GRADIENTS: THE PONTRYAGIN STRUCTURE OF RECURRENT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

A key capability of intelligent agents is to act effectively under incomplete state observations. Recurrent policies address this by compressing observation histories into a hidden state. In this work, we show that the hidden state of a recurrent policy admits a control-theoretic interpretation: it plays the role of the co-state in Pontryagin's minimum principle, and the readout that maps it to actions implements control-Hamiltonian minimization. The hidden state thus tracks the gradient of the value function, encoding the optimality structure of the underlying control problem. We formalize this correspondence through a class of policies we refer to as co-state policies (CPs) and show that several modern recurrent cells implicitly realize this structure. The correspondence also allows for a co-state loss for actor-critic training, in which the critic's gradient serves as a target for the actor's hidden state. Empirically, we find that hidden states trained with the co-state loss encode co-state information beyond what is linearly decodable from the environment state alone, and that the loss improves policies both within and outside this class on challenging locomotion tasks, including the H1 and Berkeley humanoids. By connecting the minimum principle to recurrent memory, we provide a control-theoretic account of what hidden states compute in continuous control and a mechanism for shaping them toward optimality.

1 INTRODUCTION

A fundamental challenge for intelligent agents is to act effectively under partial observability. Physical environments are characterized by noisy sensors, measurement delays, or missing data, such that a single instantaneous observation rarely determines the underlying state of the system. To succeed, an agent must learn to integrate a history of observations and actions to infer what cannot be directly seen (Kaelbling et al., 1998). In deep reinforcement learning (RL), this is commonly handled by recurrent policies, which carry a hidden state that accumulates information over time.

A natural question is what this memory should compute. The standard reading is that the hidden state tracks a belief over the unobserved state. This view is rooted in the sufficiency of the belief state (Åström, 1965): a policy acting on the posterior over states loses nothing relative to one acting on the full history, so estimation and control may be treated as separate stages.

In this work, we develop a complementary account of what the hidden state represents, by drawing a formal link between recurrent policies and the Pontryagin minimum principle (PMP) (Pontryagin, 1987). We argue that the hidden state does not need to represent the full belief, but only the quantity that the optimal action depends on: the gradient of the value function, $\nabla_x V(x)$, which appears in Pontryagin's principle as the co-state. In this view, the hidden state is a statistic of the belief: the part on which the optimal action depends.

We formalize this relationship by introducing co-state policies (CPs), the class of recurrent policies whose memory mirrors the optimal co-state dynamics of the minimum principle. In PMP, the co-state evolves alongside the state and measures the sensitivity of the optimal value function to the current state, with dynamics that are affine in the co-state and driven by the running cost. In a CP, the hidden state plays the role of the co-state. The defining features of this class of architectures are (1) that the recurrence is affine in the hidden state, and (2) that the readout performs Hamiltonian minimization.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

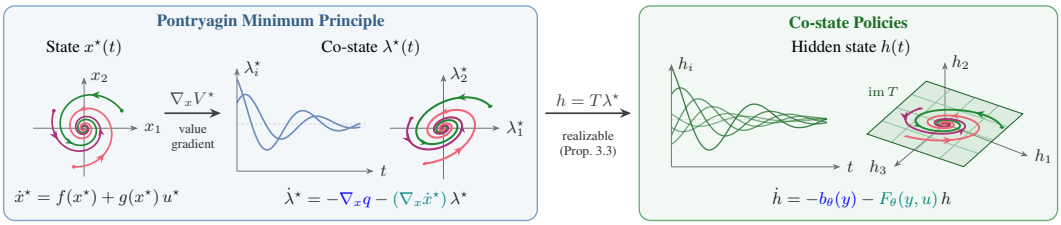

Figure 1: **The co-state structure of recurrent policies.** (left) Along optimal trajectories, the co-state is the value gradient $\lambda^\star = \nabla_x V^\star$ and evolves affinely in $\lambda^\star$. (right) Co-state policies update their memory by a recurrence of the same affine form, so their d_h-dimensional hidden state can carry the d_x-dimensional co-state as a linear image $h = T\lambda^\star$ in $\mathbb{R}^{d_h}$. Colors mark corresponding terms.

We demonstrate that the CP class includes a wide range of recurrent cells, such as structured state space models (SSMs) and linear RNNs (Gu et al., 2021; Orvieto et al., 2023), while gated cells such as the LSTM and GRU fall outside the class. Furthermore, because the co-state and the value gradient coincide along the optimal trajectories (Vinter, 1986), the critic's gradient supplies a target for the actor's hidden state that can be used as an auxiliary objective in the actor-critic setting. On a non-linear stabilization task with numerically computed optimal co-states, we find that under reward maximization alone most hidden states do not encode co-state structure beyond what is linearly decodable from the state, but that the co-state objective lifts nearly all of them above this ceiling. On humanoid locomotion (Unitree H1, Berkeley), the co-state objective improves policies both inside and outside the class.

Ultimately, this work asks what recurrent policies compute under uncertainty. By framing recurrent policies' hidden states as dynamic processes governed by the minimum principle, we provide a control-theoretic account of recurrent memory in deep RL, and a means of supervising it directly.

2 Background: Recurrent Policies in Reinforcement Learning

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:

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

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 RL, these transition dynamics are unknown to the agent and can only be sampled through interaction.

The optimal control minimizes the cost objective:

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

with terminal cost Φ, state cost q, and control cost c, all assumed differentiable and evaluable empirically though not known analytically. We write $\mathcal{L}(x,u) = q(x) + c(u)$ for the running cost. In RL the objective is commonly discounted, weighting the running cost by $e^{-\rho(t-t_0)}$, the continuous-time counterpart of γ^t with $\gamma = e^{-\rho\Delta}$ for step size Δ.

2.2 Recurrent Policies

A memory-less policy $\pi_\theta\colon \mathcal{O} \to \mathcal{U}$ considers only (the observation of) the current state to decide what action to take, such that $u(t) = \pi_\theta(y(t))$ where $\theta \in \Theta$ are the policy parameters. Because the

true state $x(t)$ cannot be fully inferred from a single observation $y(t)$ under partial observability, this memory-less controller is suboptimal in general. This necessitates a policy with memory, which maintains a latent state $h(t) \in \mathbb{R}^{d_h}$ that evolves as a dynamical process coupled to the environment, where d_h is the dimensionality of the hidden state:

$$\dot{h}(t) = \phi_\theta(y(t), h(t)), \qquad u(t) = \pi_\theta(h(t)). \tag{2.4}$$

A standard reading of h is as a filter that compresses a sequence of partial observations $y_{0:t}$ and actions $u_{0:t-1}$ into a belief $b_t = p(x_t \mid y_{0:t}, u_{0:t-1})$. This belief is a sufficient statistic for control (Åström, 1965; Kaelbling et al., 1998), and a recurrent policy is understood as approximating it.

Gated cells such as LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Chung et al., 2014) are common choices for ϕ_θ in deep RL (Wierstra et al., 2010). Minimal variants such as minGRU and minLSTM (Feng et al., 2024) remove the hidden-state dependence from the gates, restoring an affine recurrence. A second family of recurrent cells are structured state space models (SSMs) (Gu et al., 2021; Smith et al., 2022), which model their hidden states via a linear recurrence:

$$\dot{h}(t) = A(t)h(t) + B(t)y(t), \qquad u(t) = Ch(t) \tag{2.5}$$

discretized with step Δ to $h_t = \bar{A}h_{t-1} + \bar{B}y_t$. The affine structure of SSMs admits parallel scans over the sequence, enabling credit assignment over long horizons. Selective SSMs such as S6 (Gu & Dao, 2023) retain this structure while conditioning $(\bar{A}, \bar{B})$ on the current input.

3 Recurrent Policies as Optimal Dynamic Processes

Optimizing a policy for the cost in Eq. (2.3) is governed by the structure of the underlying optimal control problem. Whereas the Hamilton-Jacobi-Bellman (HJB) equation characterizes optimality globally across the state space, PMP characterizes it locally along a trajectory (Pontryagin, 1987; Kirk, 2004; Bryson & Ho, 1975). The trajectory-local view is a natural fit for a recurrent policy, as its hidden state also evolves along the trajectory it controls.

3.1 Pontryagin Minimum Principle and Hamiltonian Systems

To study optimality along a trajectory, we consider the control-Hamiltonian, which augments the running cost with the dynamics weighted by a multiplier λ:

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

Here, $\lambda \in \mathbb{R}^{d_x}$ is the *co-state*: a continuous-time Lagrange multiplier enforcing the dynamics, which measures the sensitivity of the optimal cost-to-go to the current state. Wherever $V^\star$ is differentiable, it coincides with the value gradient along optimal trajectories, $\lambda(t) = \nabla V^\star(x(t))$ (Vinter, 1986).

Theorem 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 (TPBVP) for $t \in [t_0, t_f]$:*

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

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 produces the column vector $\nabla_x \mathcal{H} = (\frac{\partial \mathcal{H}}{\partial x_1}, \ldots, \frac{\partial \mathcal{H}}{\partial x_{d_x}})^\top$.[1]

Proof: The derivation follows standard extremization of the cost functional using variational calculus. A full derivation is provided in Appendix A.2.

Two structural properties of Eq. (3.2) are particularly important for what is to follow. First, the co-state equation is affine in λ, meaning that non-linearity in the environment does not require non-linearity in the co-state. Second, because the action u enters the control-Hamiltonian $\mathcal{H}$ linearly, the action that minimizes this Hamiltonian depends on the co-state only through the projection $g(x)^\top \lambda$.

[1] For the discounted infinite-horizon objective, the current-value co-state is $\dot{\lambda}^\star = \rho\lambda^\star - \nabla_x q - (\nabla_x \dot{x}^\star)\lambda^\star$.

Optimal Hamiltonian system

$$\dot{x}^\star = f(x^\star) + g(x^\star)u^\star,$$
$$\dot{\lambda}^\star = -\nabla_x q - (\nabla_x \dot{x}^\star)\lambda^\star,$$
$$u^\star = \arg\min_{u\in\mathcal{U}}\{c(u) + u^\top g(x^\star)^\top \lambda^\star\},$$

Co-state Policy

$$\dot{x}^\star = f(x^\star) + g(x^\star)u^\star, \quad y^\star = \psi(x^\star) + v,$$
$$\dot{h}^\star = -b_\theta(y^\star) - F_\theta(y^\star, u^\star)h^\star,$$
$$u^\star = \arg\min_{u\in\mathcal{U}}\{c(u) + u^\top G_\theta(y^\star)h^\star\},$$

Figure 2: **Structural parallel between the optimal Hamiltonian system (left) and the CP (right)**: an optimal CP represents the co-state as a linear embedding. This results in a policy that follows an optimal dynamic control law via a latent continuous-time process. Color indicates direct correspondence under full observability (3.2) and indirect correspondence under partial observability (3.3).

3.2 Co-state Policies

The PMP describes optimality through two coupled processes: the physical state x, and the co-state λ, with the control readout from the latter. We claim that this is the structure of a recurrent policy: the hidden state h plays the role of λ, and the policy readout that of the Hamiltonian minimization.

Definition 3.2 (Co-state Policy)**.** A co-state policy (CP) is a recurrent policy of the form

$$\dot{h} = -b_\theta(y) - F_\theta(y,u)h, \qquad u = \arg\min_{u\in\mathcal{U}}\{c(u) + u^\top G_\theta(y)h\}, \tag{3.3}$$

where $h \in \mathbb{R}^{d_h}$ is the memory state, and $b_\theta, F_\theta, G_\theta$ may depend arbitrarily on the observation and control but not on h. For $c(u) = \frac{1}{2}u^\top Ru$ with $R \succ 0$ and $\mathcal{U} = \mathbb{R}^{d_u}$, the readout has the closed form $u = -R^{-1}G_\theta(y)\,h$.

The defining property is that the recurrence is affine in h for a given action. Next, we consider when such a network reproduces the optimal co-state exactly. Because recurrent policies are higher-dimensional ($d_h \geq d_x$), we state it through a fixed embedding T.

Proposition 3.3 (Co-state Representation)**.** *Let $c(u) = \frac{1}{2}u^\top Ru$ with $R \succ 0$, assume full observability (ψ invertible, $v = 0$), and let $T \in \mathbb{R}^{d_h \times d_x}$ have full column rank and be fixed along the trajectory. If*

$$b_\theta(y) = T\,\nabla_x q(x), \qquad F_\theta(y,u)\,T = T\,\nabla_x \dot{x}, \qquad G_\theta(y)\,T = g(x)^\top, \tag{3.4}$$

and $h(t_0) = T\lambda^\star(t_0)$, then the CP realizes the optimal co-state dynamics $h(t) = T\lambda^\star(t)$ for all $t \in [t_0, t_f]$ and emits $u(t) = u^\star(t)$ for all $t \in [t_0, t_f]$.

Proof: We verify that $h(t) = T\lambda^\star(t)$ satisfies the CP dynamics. Substituting the first two conditions into the CP dynamics gives $\dot{h} = -b_\theta(y) - F_\theta(y,u)\,T\lambda^\star = -T\nabla_x q - T(\nabla_x \dot{x})\lambda^\star = T\dot{\lambda}^\star$, so h and $T\lambda^\star$ follow the same dynamics. The relation $h = T\lambda^\star$ is therefore invariant, and $h(t_0) = T\lambda^\star(t_0)$ places the trajectory on it for all t. The readout then returns $u = -R^{-1}G_\theta(y)\,T\lambda^\star = -R^{-1}g(x)^\top\lambda^\star = u^\star$ by the third condition. □

This makes the analogy between hidden states and co-states precise: a CP whose operators represent the Hamiltonian terms is an optimal controller, with h a linear image of $\lambda^\star$. We summarize this in Fig. 2. While Theorem 3.1 describes a TPBVP that fixes λ at the terminal time, a causal recurrent policy must go forward in time and cannot satisfy the end-point boundary directly. However, along optimal trajectories the co-state admits a feedback form, $\lambda^\star(t) = \nabla_x V^\star(x^\star(t), t)$, so it is determined by the current state and time. As such, a recurrent policy can track the co-state forward in time by realizing this value gradient.

3.3 Partial Observability: Linear-Quadratic-Gaussian Analysis

Whereas Prop. 3.3 assumes that the network observes the true state, recurrent policies are deployed to resolve partial observability. We analyze partial observability in the linear-quadratic-Gaussian setting, where the compression of the sequence of observations and actions into a sufficient statistic is exact. Let $\hat{x}$ be the steady-state Kalman estimate and P the solution to the Riccati equation, so that $\lambda^\star = Px$. Setting $h = P\hat{x}$, the CP is written as:

$$\dot{h} = P\dot{\hat{x}} = \underbrace{PLy}_{-b_\theta(y)} + \underbrace{P\left(A - LC - BR^{-1}B^\top P\right)P^{-1}}_{-F_\theta} h, \tag{3.5}$$

Table 1: **Recurrent cells as co-state policies.** The components of each cell (single-layer) are written as Eq. (4.1). Discrete-time cells use $\Delta_t = 1$; for zero-order-hold cells b_θ holds up to input scaling.

Cell	CP	$\boldsymbol{\Delta_t}$	$\boldsymbol{b_\theta}$	$\boldsymbol{F_\theta}$	$\boldsymbol{G_\theta}$
LSTM (1997), GRU (2014)	✗		non-linear gates on h_{t-1}		–
minGRU (2024)	✓	1	$-\sigma(W_z y) \odot \tilde{h}(y)$	$\mathrm{diag}\big(\mathrm{softplus}(W_z y)\big)$	I
minLSTM (2024)	✓	1	$-i_t/(f_t + i_t) \odot \tilde{h}(y)$	$\mathrm{diag}\big(\log(1 + i_t/f_t)\big)$	I
S4D (2022), S5 (2022)	✓	Δ	$-B\,y$	$\mathrm{diag}(\nu - i\omega)$	C
LRU (2023)	✓	1	$-\gamma \odot B\,y$	$\mathrm{diag}(\nu - i\omega)$	C
S6 (2023)	✓	$\Delta_t(y)$	$-B_t(y)\,y^\top$	$\mathrm{diag}(\nu)$	$C_t(y)$
Mamba-2 (SSD) (2024)	✓	$\Delta_t(y)$	$-B_t(y)\,y^\top$	$\nu\, I$	$C_t(y)$
DeltaNet (2021)	✓	1	$-\beta_t\, v_t k_t^\top$	$-\log(1-\beta_t)\, k_t k_t^\top$	$q_t(y)$

where L is the Kalman gain (Appendix B). This means that the CP operators carry a belief-state reading: b_θ applies the Kalman innovation in co-state coordinates, while F_θ combines the environment dynamics with the observer correction and the control feedback. This makes CPs applicable to the partially observed settings where recurrent policies are used in practice.

In non-linear settings, local linearization can approximate the optimal solution via time-varying Jacobians $A(t) = (\nabla_x \dot{x})^\top$ and $B(t) = g(x(t))$, following the standard construction of iterative LQG and the extended Kalman filter (Todorov & Li, 2005). Because the Riccati solution $P(t)$ and observer gain $L(t)$ now evolve along the trajectory, differentiating the co-state mapping $h(t) = P(t)\hat{x}(t)$ introduces an additional $\dot{P}\hat{x}$ term. Substituting the continuous-time Riccati equation for $\dot{P}$ gives (suppressing t for readability):

$$\dot{h} = P\dot{\hat{x}} + \dot{P}\hat{x} = \underbrace{PLy}_{-b_\theta(y)} - \underbrace{\big(A^\top + QP^{-1} + PLCP^{-1}\big)}_{F_\theta} h. \tag{3.6}$$

This preserves the affine structure of the recurrence: F_θ incorporates the fully observable adjoint dynamics ($A^\top$ and QP^{-1}) as well as the partial-observability correction ($PLCP^{-1}$). As such, estimation and control are not two separate stages, but enter the same coordinates additively.

3.4 The readout as Hamiltonian minimization

In standard actor architectures, the readout that maps the hidden state to the action is chosen by convention. In CPs, it can be chosen by minimizing the control-Hamiltonian, which depends on the co-state and is captured by the switching function $\sigma(t) \coloneqq -G_\theta(y)h(t)$ ($= -g(x,t)^\top \lambda(t)$ on the manifold). The form of the policy readout is then determined by the control cost (illustrated in Fig. 5).

4 The Control-Hamiltonian Structure of Recurrent Networks

4.1 Which recurrent cells are co-state policies?

The CP class (Definition 3.2) asks that the recurrence h enters its own update affinely. After discretizing the continuous-time dynamics with step Δ_t, we can write the following discrete-time CP:

$$h_t = \bar{A}_t h_{t-1} + \bar{b}_t, \qquad \bar{A}_t = \exp\big(-\Delta_t F_\theta(y_t, u_t)\big), \quad \bar{b}_t = -\Delta_t\, b_\theta(y_t). \tag{4.1}$$

To decide membership, we determine whether a recurrent cell's transition can be written in this form.

Gated recurrent family. GRUs and LSTMs contain non-linear updates in the hidden state. The GRU update $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h y_t + U_h(r_t \odot h_{t-1}))$ is affine in neither term, while LSTM is affine in its cell state c_t but not in its gates, which are non-linear functions of h_{t-1}. As such, neither GRU nor LSTM fall inside the CP class. Their minimal variants minGRU and minLSTM (Feng et al., 2024), however, remove the hidden-state dependence from the gates, leaving $h_t = (1 - z_t(y)) \odot h_{t-1} + z_t(y) \odot W_h y_t$ (equivalent to Eq. (4.1) with $\bar{A}_t = 1 - z_t(y)$). Therefore, these are CPs.

Linear RNNs and state-space models. These satisfy Eq. (4.1) by construction. S4D (Gu et al., 2022), S5 (Smith et al., 2022) and the LRU (Orvieto et al., 2023) use a fixed diagonal $\bar{A}$; selective cells such as S6 (Gu & Dao, 2023) and Mamba-2 (Dao & Gu, 2024) condition Δ_t, $\bar{B}_t$ and the readout on the observation, making them time-variant. Interestingly, the condition that makes a cell a CP is the same one that makes it parallelizable: an affine recurrence composes associatively and can be evaluated using scans. These conditions align because the co-state equation is itself affine in λ. We identify the components $b_\theta, F_\theta, G_\theta$ of each cell for various recurrent cells in Table 1.

4.2 Supervising the co-state

A suitable architecture alone does not guarantee that co-state representations naturally emerge under reward maximization, however. One can use the identity $\lambda^\star = \nabla_x V^\star$ to steer it towards latent representations that align with co-states. In the actor-critic setting, the critic already estimates the value function and hence its gradient. By taking the critic's gradient as a proxy for the value gradient, we construct a hidden state target $\hat{\lambda}_t = \nabla_{y_t} V_\psi(y_t, h^c_{t-1})$. The critic's hidden state h^c_{t-1} is detached, since the goal is for the actor's hidden state to mirror the critic's gradient instead of the other way around. Mirroring Prop. 3.3, we align the two through a linear embedding:

$$\mathcal{L}_{\text{co-state}}(\theta, W) = \mathbb{E}_t \left[\left\| W h_t - \mathtt{sg}\big[\hat{\lambda}_t\big] \right\|^2 \right], \tag{4.2}$$

where h_t is the recurrent state, $W \in \mathbb{R}^{d_y \times d_h}$ is a learned projection matrix and $\mathtt{sg}$ denotes the stop gradient operator. Here, $\nabla_{y_t} V_\psi$ is the observable proxy for $\nabla_x V^\star$. Supervising a value gradient makes this loss adjacent to the field of dual heuristic programming (Werbos, 1992). Since the loss objective is based purely on the actor and the critic, it can also be applied to recurrent policies outside the CP class.

5 Experiments

Our experiments address two questions: i) do hidden states of reward-maximizing policies embed optimal co-states, beyond state estimation? and ii) does supervising the hidden states with the value gradient of the critic improve this embedding as well as the return?

Throughout, we use a recurrent actor to select actions and a recurrent critic to estimate returns. We optimize both with recurrent proximal policy optimization (PPO) (Schulman et al., 2017) and generalized advantage estimation (GAE) (Schulman et al., 2015), computing gradients via back-propagation through time (BPTT). Our implementation extends RSL-RL (Schwarke et al., 2025); all code and training scripts are available via the anonymized repository `https://anonymous.4open.science/r/tls0ts0hxw7v` and will be publicly released.

5.1 Do hidden states encode co-states?

To determine whether hidden states truly embed value gradients, we distinguish two hypotheses: either h strictly estimates the unobserved state, or it tracks the co-state λ explicitly. We evaluate these hypotheses on the non-linear FitzHugh-Nagumo system, with the objective to stabilize the system's unforced limit cycle to the origin from partial voltage observations (Appendix E). Because the system's control influence function $g(x) = (R_{\text{sys}}, 0)^\top$ does not vary with the state, the third condition of Prop. 3.3 is met by a constant G_θ. Consequently, the CP readout is linear for this task, ensuring that the experiment isolates the recurrence.

We first generate $m = 500$ ground-truth optimal trajectories $\{x^\star(t), \lambda^\star(t)\}$ by solving the exact Hamiltonian two-point boundary value problem (Theorem 3.1) via indirect single shooting (Gros & Diehl, 2022). Independently, we train recurrent PPO agents on the identical undiscounted objective, restricting the actor to this linear readout. Evaluating the trained policies from the same initial conditions allows us to test Prop. 3.3, which predicts that h is a linear image of the optimal co-state. We fit a single ridge regression model $\hat{\lambda}_t = W h_t$ across all time steps on 80% of the trajectories, discarding the initial 5% for the uninformative $h = \mathbf{0}$ initialization, and report the coefficient of determination (R^2) on the held-out 20%.

An important baseline ceiling is obtained by fitting a linear model from the true optimal state to the optimal co-state. This is the best any policy could achieve if its memory were to perfectly track the

Table 2: **Linear decodability of optimal co-states on FitzHugh–Nagumo.** We train several policies with a linear readout layer across 10 seeds (mean $\pm$ std), and evaluate them from the same initial condition as the optimal state and co-state trajectories. The value gradient loss is applied with coefficient $c_{\text{co-state}} = 0.2$. The held-out trajectories are fitted across all time steps of the window $[0.5, 10)$ with $d_h = 64$ and $\Delta_t = 0.02$ without truncated BPTT. Δ_{ceil} is the difference to the linear ceiling. Across the optimal trajectories, this is 0.870; we recompute it across the states visited by each policy in the rollouts. Bolded values indicate the Δ is more than two standard deviations above the linear ceiling.

	R^2: **PPO (reward only)**			R^2: **PPO** $+ \boldsymbol{\mathcal{L}}_{\text{co-state}}$		
Cell	$h \to \lambda^\star$	Δ_{ceil}	$h \to x^\star$	$h \to \lambda^\star$	Δ_{ceil}	$h \to x^\star$
LSTM	.727±.025	−.140	.810±.018	.932±.024	**+.066**	.971±.018
GRU	.870±.025	+.003	.929±.018	.950±.004	**+.088**	.985±.003
minGRU	.885±.014	+.024	.960±.010	.915±.012	**+.055**	.971±.007
minLSTM	.864±.020	+.003	.953±.012	.920±.010	**+.060**	.974±.004
LRU	.865±.024	+.008	.947±.016	.867±.025	+.033	.932±.031
S6	.909±.003	**+.049**	.959±.006	.920±.004	**+.060**	.974±.003

state. Across the optimal trajectories this gives $R^2(x^\star \to \lambda^\star) = 0.870$; we recompute it per seed across the states each policy visits, and report the difference Δ_{ceil}. We furthermore report how well h tracks the unobserved state x via $R^2(h \to x^\star)$.

Table 2 shows that membership in the CP class does not by itself produce co-state structure beyond the ceiling. Under reward maximization, S6 exceeds it, while minGRU, minLSTM, GRU and LRU sit at it, and the non-CP LSTM falls well below. Supervising the hidden state with the critic's value gradient lifts every architecture except LRU above the ceiling. The loss is most effective for the cells outside the class: the GRU attains the highest decodability overall ($R^2 = .950, \Delta_{\text{ceil}} = +0.088$). Representing the co-states is therefore driven more by the training objective than by the type of recurrence. Improved co-state alignment is accompanied throughout by improved state tracking, consistent with the analysis in Sec. 3.3, where the co-state is an image of the state estimate. State estimation is a component of co-state tracking, but the positive Δ_{ceil} shows it is not sufficient.

5.2 The Effect of the Co-state Coefficient and Readout Layer

To study the effect of the co-state loss on the mean return, we consider a partially observable variant of the `CheetahRun` task from the DeepMind Control Suite without velocity observations (-V) (Tassa et al., 2018; Han et al., 2019). We first evaluate the effect of the co-state loss by augmenting the recurrent PPO objective with the co-state objective (Eq. (4.2)). We restrict this analysis to the GRU cell with a linear readout layer, and consider the co-state coefficients $c = \{0., 0.025, 0.05, 0.1, 0.2\}$. Because the loss is with respect to the critic, the objective has to be balanced with the other PPO objectives whose constants are kept fixed throughout (an overview of training hyperparameters is given in Table 3).

The left panels in Fig. 3 show that the co-state loss increases the mean episode return for all coefficient values. The second left panel shows that a stronger regularization constant indeed decreases the loss as intended. To investigate the effect of the readout layer, another architectural component of the CP class, we consider a linear readout, an MLP readout, and a selective readout in the form of $C_t(y)h$, and train these with and without co-state loss ($c_{\text{co-state}} = 0.1$, based on the left panel results). The right panels show that the readout layers result in similar performances when trained without co-state loss. With the co-state loss, however, all three readouts improve, including the MLP readout outside the CP class.

5.3 Humanoid locomotion

We next evaluate the co-state objective on two partially observable humanoid locomotion tasks: the Berkeley humanoid ($d_y = 52, d_u = 12$), which requires tracking joystick velocity commands over rough terrain, and the Unitree H1 ($d_y = 113, d_u = 19$), which navigates flat terrain with a joint

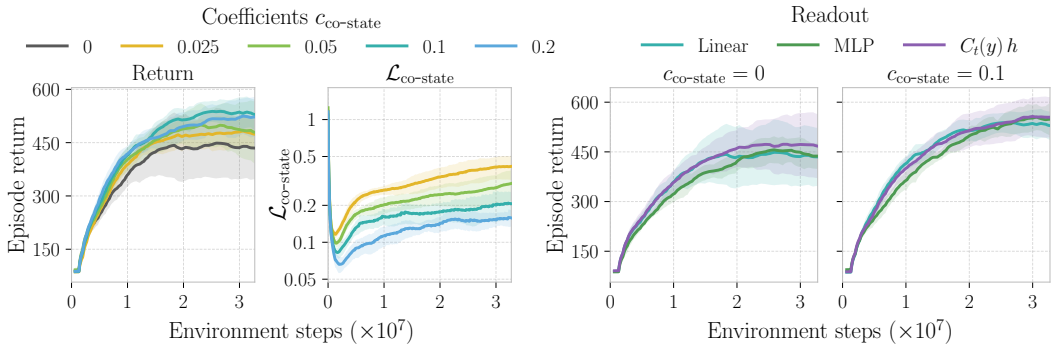

Figure 3: **Effect of the co-state coefficient and the readout layer on partially observable CheetahRun (velocity removed).** We train a GRU policy with a linear readout layer across 10 seeds (mean $\pm$ std), and evaluate: **(left)** episode return and co-state loss for $c_{\text{co-state}} \in \{0, 0.025, 0.05, 0.1, 0.2\}$. (**right**) episode return for the linear, MLP, and selective readout layers, with and without the co-state loss.

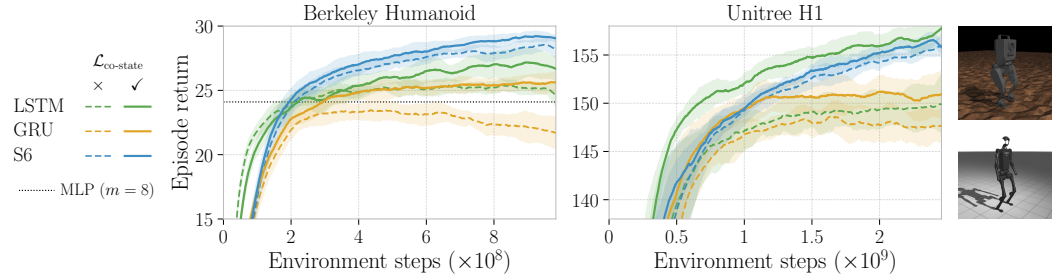

Figure 4: **Berkeley Humanoid Joystick on Rough Terrain (left) and Unitree H1 Joystick Gait tracking (right).** We train LSTM, GRU and S6 policies across 10 seeds (mean $\pm$ standard deviation), where each cell uses the most beneficial readout for the task (linear for GRU; non-linear for LSTM). Each model is trained *with* (full lines) and *without* (colored dashed lines) the co-state loss with $c_{\text{co-state}} = 0.05$. The black dotted line shows the final return of the frame-stacked MLP baseline ($m = 8$; Appendix D.1), 24.1 ± 0.6 after 4×10^8 steps on the Berkeley humanoid; on the H1, it attains 99.1 ± 17.2 after 8×10^8 steps, below the plotted range. Both exceed the single-frame results reported by Zakka et al. (2025).

objective of velocity and gait tracking (Zakka et al., 2025). Both environments feature strong partial observability and contact discontinuities. As a non-recurrent baseline, we use an MLP with frame stacking ($m = 8$), tuned over stack length and learning rate (Appendix D.1). We compare GRU (linear readout), LSTM (MLP readout), and S6 cells, each trained with and without the co-state loss ($c_{\text{co-state}} = 0.05$, selected in Appendix D.2).

Fig. 4 shows that the co-state objective improves the mean episode return for all cell types on both tasks. The GRU attains the lowest return on both tasks and suffers from long-term training instability on the Berkeley robot; the co-state loss improves its return and resolves this instability. On the H1, the LSTM gains most from the co-state loss, achieving the highest reward on this task and improving substantially on the Berkeley task. Finally, the S6 cell attains the highest return of all baseline models prior to applying the co-state loss, but improves only marginally with it. The architectures that gain most in return are also those whose co-state alignment improves most in Sec. 5.1.

6 Related Work

Recurrent policies and hidden state representations. Recurrent policies have been used effectively in deep RL for partially observable environments and remain a strong baseline for POMDPs (Wierstra et al., 2010; Hausknecht & Stone, 2015; Ni et al., 2021). Their memory states are commonly read as

a belief over the unobserved state, justified by the sufficiency of the belief for control (Åström, 1965; Kaelbling et al., 1998; Bertsekas et al., 1995). We read the same memory through optimal control, where the hidden state carries the co-state of the minimum principle and the readout minimizes the control-Hamiltonian. These two readings are nested rather than opposed: the belief-state sufficiency establishes what information the memory may use, while the minimum principle identifies what part the action requires. This is best seen in the linear-quadratic-Gaussian case (Sec. 3.3), where the observer correction enters the same co-state coordinates as the adjoint dynamics.

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, the minimum principle, has played a smaller role. The closest precedent is adaptive dynamic programming, where dual heuristic programming trains a critic to output the value gradient directly and derives the control from this (Werbos, 1992). This is model-based and uses the value gradient as a training signal for a memoryless actor. More recently, PMP has been used in model-based paradigms, where the Pontryagin conditions are differentiated to obtain gradients through an optimal trajectory (Jin et al., 2020), and for planning under uncertainty (Leeftink et al., 2025). The CP framework instead is model-free and uses the minimum principle to characterize what recurrent memory represents.

Modern sequence models. A parallel line of work replaces attention with recurrences that can be evaluated by parallel scans, including linear transformers (Katharopoulos et al., 2020), fast-weight programmers and DeltaNet (Schlag et al., 2021), gated linear attention (Yang et al., 2025) and Mamba-2 (Dao & Gu, 2024). As shown in Sec. 4.1, the affine update that enables their parallel scans is exactly what places them in the CP class, giving a control-theoretic motivation for adopting them in reinforcement learning, beyond the computational one. Such recurrences are increasingly used as backbones in vision-language-action (VLA) models (Liu et al., 2024), whose memory the co-state loss could supervise directly during actor-critic fine-tuning.

7 Discussion

We propose that the hidden state acts as the Pontryagin co-state, while the readout minimizes the control-Hamiltonian. We formalize this through the co-state policy class, which encompasses many modern architectures from linear RNNs to selective state space models. Because the co-state coincides with the value gradient along optimal trajectories, the critic's gradient can directly supervise the actor's memory. Our experiments show that while co-state policies can realize optimal co-state dynamics, most architectures do not represent them when trained under reward maximization alone. Directly supervising the hidden state with the value gradient improves both co-state representation and mean return.

A limitation of the algorithmic use of the co-state identity $\lambda^\star = \nabla_x V^\star(x^\star)$ is that it only holds where $V^\star$ is differentiable. While our results show that the co-state objective still helps on contact-rich tasks such as humanoid locomotion, future work is needed to understand the consequences. A second limitation is that while our partial observability analytical results are exact for the linear-quadratic-Gaussian setting and extend to non-linear systems through local linearization, the global non-linear setting is studied only via our empirical results.

A promising direction of future work is to investigate the role of hidden state initialization. While the standard practice of initializing the hidden state at zero comes with little theoretical motivation, the CP class prescribes that the initial hidden states follow from the value gradient, $h(t_0) = T\nabla_x V^\star(x_0)$, which is not available at run time. A different direction is extending the correspondence to stochastic hidden states via the stochastic maximum principle (Peng, 1990). Finally, the connection between hidden states and PMP opens a direction for safe reinforcement learning under partial observability. State constraints enter the control-Hamiltonian through additional multiplier terms (Hartl et al., 1995), which recurrent policies of this class could carry in the same memory that represents the co-state.

Taken together, these results connect the minimum principle to recurrent memory. They offer an account of hidden states as carriers of the optimality structure of the underlying control problem, rather than of the state alone, together with a mechanism for shaping them toward it.

AI USE STATEMENT

Generative AI was used in this work to aid and polish writing of the work, assisting in implementing methods, assisting in the writing of proofs, and in assisting in formulating mathematical claims. We have not used generative AI tools for: generating synthetic data sets, helping develop theoretical models or conceptual frameworks, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, interpreting results, assisting with translation, designing or providing feedback on research methodology or experiments, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses. Additionally, we used generative AI tools for modifying scientific figures, editing software code, and determining keywords for the paper. We have reviewed all AI-assisted work. LLM-writing was checked carefully and rephrased; LLM-generated code was verified and tested for correctness. We take full responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work contributes to reinforcement learning under partial observability. Practically, controllers capable of sustaining stable behavior under 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 of tolerating 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.

REPRODUCIBILITY STATEMENT

All code and training scripts are available at the following anonymous repository: `https://anonymous.4open.science/r/tls0ts0hxw7v`, and will be made public. Full derivations of the theoretical results are given in Appendix A.2 (optimal Hamiltonian system) and Appendix B (LQG analysis). Environment specifications, algorithmic components, and all hyperparameters are listed in Appendix E and Table 3. Main results are reported over 10 random seeds; seed counts for the appendix ablations are given in the respective captions.

REFERENCES

Karl Johan Åström. Optimal control of Markov processes with incomplete state information i. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

Michael Athans. The role and use of the stochastic linear-quadratic-Gaussian problem in control system design. *IEEE Transactions on Automatic Control*, 16(6):529–552, 1971.

Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena scientific Belmont, MA, 1995.

Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Taylor & Francis, 1975.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were RNNs all we needed? *arXiv preprint arXiv:2410.01201*, 2024.

S. Gros and M. Diehl. *Numerical Optimal Control (Draft)*. 2022. Available online: `https://www.syscop.de/files/2020ss/NOC/book-NOCSE.pdf` (accessed on 28 June 2023).

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019.

Richard F Hartl, Suresh P Sethi, and Raymond G Vickson. A survey of the maximum principles for optimal control problems with state constraints. *SIAM Review*, 37(2):181–218, 1995.

Matthew J Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposia*, volume 45, pp. 141, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022.

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

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

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

David Leeftink, Cagatay Yildiz, Steffen Ridderbusch, Max Hinne, and Marcel Van Gerven. 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.

Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2011.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free RL can be a strong baseline for many POMDPs. *arXiv preprint arXiv:2110.05038*, 2021.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Shige Peng. A general stochastic maximum principle for optimal control problems. *SIAM Journal on Control and Optimization*, 28(4):966–979, 1990.

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

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. RSL-RL: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv e-prints*, art. arXiv:1801.00690, January 2018. doi: 10.48550/arXiv.1801.00690.

Emanuel Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005 American Control Conference*, pp. 300–306. IEEE, 2005.

Richard B Vinter. 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.

Paul Werbos. Approximate dynamic programming for real-time control and neural modeling. *Handbook of Intelligent Control*, 1992.

Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 18(5):620–634, 2010.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

APPENDIX

A OPTIMAL HAMILTONIAN SYSTEM

A.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:

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

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. 2.3 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:

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

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

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:

$$u^\star = \arg\min_{u \in \mathcal{U}} \mathcal{H}(x^\star, \lambda^\star, u) \tag{A.4}$$

This tells us that if the optimal control $u^\star(t)$ is applied, producing corresponding optimal trajectories $x^\star(t)$ 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.

A.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 $J(u)$:

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

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:

$$J(u) = \Phi(x(t_f)) + \int_{t_0}^{t_f} \left[\mathcal{H}(x, \lambda, u) - \lambda(t)^\top \dot{x} \right] \mathrm{d}t. \tag{A.6}$$

We consider the Gâteaux derivative of this cost functional:

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

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 $u^\star(t)$:

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

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:

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

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):

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

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:

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

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:

$$u^\star(t) = \arg\min_{u \in \mathcal{U}} \mathcal{H}(x^\star(t), \lambda^\star(t), u). \tag{A.12}$$

This establishes the optimality condition for the control sequence.

B Linear-Quadratic-Gaussian Analysis

We consider the LQG setting, with linear dynamics $\dot{x} = Ax + Bu + w$, observations $y = Cx + v$ with $v \sim \mathcal{N}(0, \Sigma_v)$, $w \sim \mathcal{N}(0, \Sigma_w)$, and quadratic cost $\mathcal{L}(x, u) = \frac{1}{2}(x^\top Q x + u^\top R u)$ with $Q \succeq 0$ and $R \succ 0$. Assume (A, B) stabilizable and (A, C) detectable, so that the steady-state Kalman gain L and the stabilizing solution $P \succ 0$ of the control algebraic Riccati equation $A^\top P + PA - PBR^{-1}B^\top P + Q = 0$ both exist.

By the separation principle, the Kalman estimate $\hat{x}$ is a sufficient statistic for the belief (Athans, 1971), and the optimal control is the full-information law evaluated at it,

$$u^\star = -R^{-1}B^\top P\,\hat{x}, \qquad \dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly. \tag{B.1}$$

The co-state of the deterministic problem satisfies $\lambda^\star = Px$, so in the partially observed case the corresponding quantity is $h = P\hat{x}$, an estimate of $\lambda^\star$ itself. This is the CP embedding of Prop. 3.3 with $T = I$.

Differentiating and substituting the control law gives

$$\dot{h} = P\dot{\hat{x}} = P\left(A - LC - BR^{-1}B^\top P\right)\hat{x} + PLy, \tag{B.2}$$

and rewriting $\hat{x} = P^{-1}h$ leads to Eq. (3.5), with $b_\theta(y) = -PLy$ and $F_\theta = -P(A - LC - BR^{-1}B^\top P)P^{-1}$. Both are independent of h, so the recurrence is affine in h and the policy is a CP. The readout $u = -R^{-1}B^\top P\hat{x} = -R^{-1}B^\top h$ is linear in h, matching $G_\theta = B^\top$ in Definition 3.2. □

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 CP 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 read out layer to match different classes of optimal control problems (illustrated in Fig. 5):

1. Quadratic Control (Smooth Action): Environments with a quadratic penalty on control effort, $\mathcal{L}(x, u) = q(x) + u^\top R u$ and control-affine system dynamics $\dot{x} = f(x) + g(x)u$, lead to 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

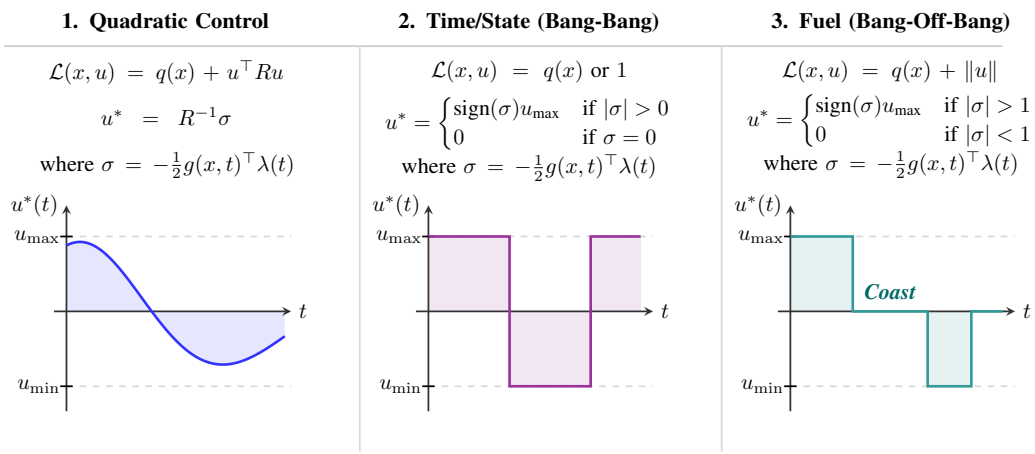

Figure 5: **Hamiltonian policies under varying cost functions:** Energy-optimal (1), time-optimal (2), and fuel-optimal (3). This can produce either smooth optimal control functions or discontinuous bang-bang controls.

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 CP framework, a bang-bang optimal controller can be enforced simply by applying a $\text{sign}(\cdot)$ activation function to the read out layer.

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 Additional humanoid experiments

D.1 Frame-stacked MLP baseline

As a non-recurrent baseline for the humanoid tasks, we train an MLP policy on a stack of the m most recent observations. We evaluate the stack lengths and learning rates shown in Fig. 6 ($m \in \{1, 8, 16\}$ on the H1 and $m \in \{1, 4, 8\}$ on the Berkeley humanoid), keeping all other hyperparameters as in Table 3, and train each for up to 8×10^8 (H1) and 4×10^8 (Berkeley) environment steps.

On the Unitree H1, frame stacking with $m = 8$ and learning rate 5×10^{-4} attains the highest return (99.1 ± 17.2), improving over the single-frame policy ($m = 1$, 77.9 ± 7.9). Longer histories do not help further: with $m = 16$, the return is lower and varies strongly across seeds, and with the larger learning rate of 10^{-3}, both $m = 8$ and $m = 16$ plateau at lower returns. Notably, the single-frame MLP in our pipeline already exceeds the score of ≈ 30 reported by Zakka et al. (2025), so we attribute most of this gap to our training setup rather than to frame stacking. The same configuration performs best on the Berkeley humanoid, where it attains 24.1 ± 0.6, compared to ≈ 17 reported by Zakka et al. (2025) for a single-frame MLP. We therefore use $m = 8$ with learning rate 5×10^{-4} as the baseline for both tasks in Fig. 4.

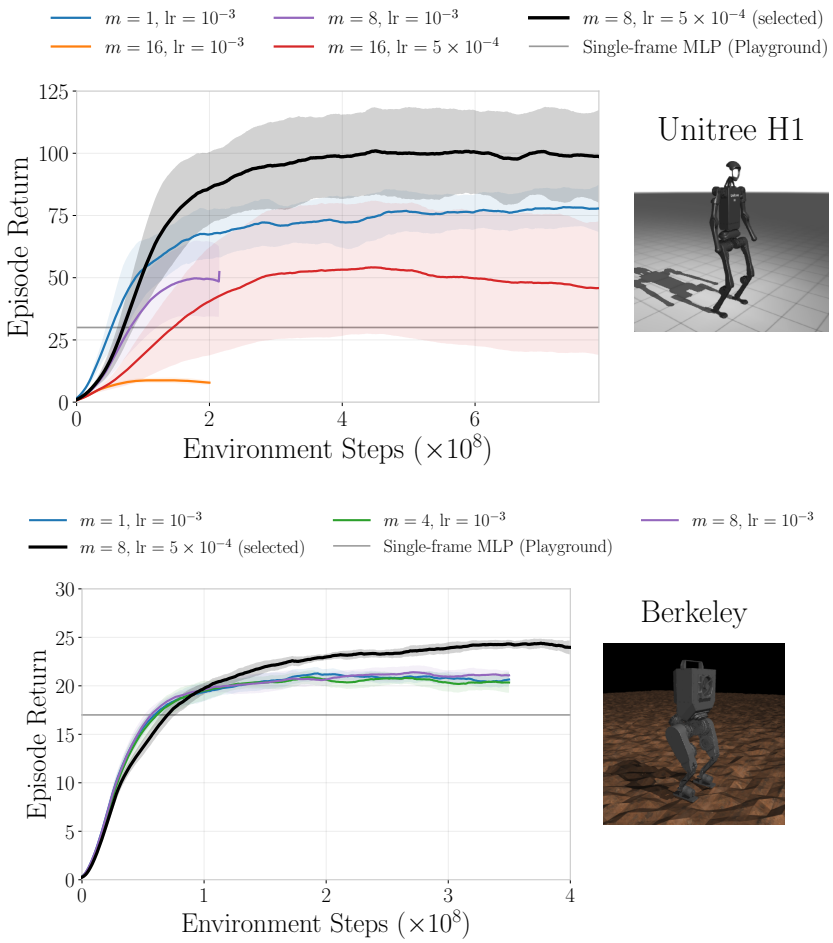

Figure 6: **Frame-stacked MLP baseline on the Unitree H1 Joystick Gait Tracking (top) and Berkeley Humanoid Joystick Rough Terrain (bottom).** We train MLP policies on the last m observations for different stack lengths and learning rates across 10 seeds (mean $\pm$ std). The grey line marks the single-frame MLP score reported by Zakka et al. (2025). On the H1, runs with learning rate 10^{-3} for $m \in \{8, 16\}$ were stopped early due to poor performance; on the Berkeley humanoid, runs with learning rate 10^{-3} are shown up to 3.5×10^8 steps, after which not all seeds were logged.

D.2 Co-state coefficient ablation on the Berkeley humanoid

To select the co-state coefficient for the humanoid experiments, we train the GRU policy with a linear readout on the Berkeley humanoid for $c_{\text{co-state}} \in \{0, 0.02, 0.05, 0.1\}$, keeping all other hyperparameters as in Table 3. Fig. 7 shows that every non-zero coefficient improves the return over reward-only training. The coefficient $c_{\text{co-state}} = 0.05$ attains the highest final return; we apply it to the H1 without further tuning.

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, as implemented by MuJoCo Playground (Zakka et al., 2025). We leverage parts of the RSL-RL framework for the implementation of recurrent PPO (Schwarke et al., 2025). The actor and critic are trained with their own parameters, and 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.

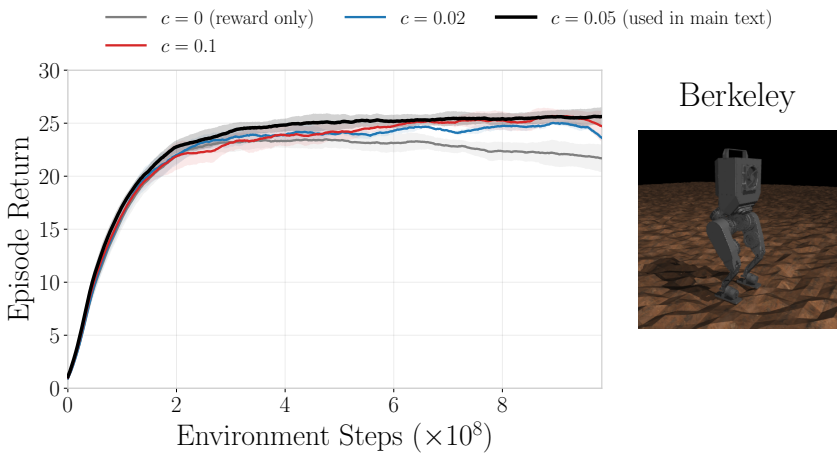

Figure 7: **Co-state coefficient sweep for the GRU on the Berkeley humanoid.** Mean $\pm$ std over 10 seeds for $c_{\text{co-state}} \in \{0, 0.05\}$ (the runs in Fig. 4), 5 seeds for $c_{\text{co-state}} = 0.1$, and 3 seeds for $c_{\text{co-state}} = 0.02$.

E.1 FitzHugh-Nagumo system

We evaluate our approach on a controlled FitzHugh-Nagumo oscillator. Setting the system parameters to $a = 0, b = 0, c = 1$, and the control gain to $R_{\text{sys}} = 1$, the continuous-time dynamics are given by:

$$\begin{aligned} \dot{v} &= v - \frac{v^3}{3} - w + u, \\ \dot{w} &= v, \end{aligned} \tag{E.1}$$

where $z = [v, w]^\top$ is the system state and $u \in [-1, 1]$ is a bounded continuous control input. The objective is to minimize a standard quadratic cost over a finite horizon $T = 10$:

$$J = 0.1\|z_T\|^2 + \int_0^T \left(0.1\|z_t\|^2 + 0.05\|u_t\|^2\right) \mathrm{d}t. \tag{E.2}$$

The system is integrated using a 4th-order Runge-Kutta (RK4) method with a physics substep of 5×10^{-3} and a control period of $dt = 0.02$, resulting in episodes of 500 time steps. Initial states are sampled uniformly from $[-3, 3]^2$. To enforce partial observability, the agent only receives the scaled voltage variable $(v_t/2)$ as its observation, requiring it to infer the gating variable w_t from history.

Policies are trained using 1,024 parallel environments for a total of 153.6 million time steps. The recurrent policies use a hidden state size of 64 and are optimized using a learning rate of 5×10^{-4}. Gradients are computed using full backpropagation through time (BPTT) over the entire 500-step episode without truncation.

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 HYPERPARAMETERS

The hyperparameter configurations for our systems are detailed in Table 3.

Table 3: PPO and Network Hyperparameters

Category	Hyperparameter	Value
Architecture	Hidden Layer Size	256
	Initialization Scale	1.0 (General), 0.01 (Actor Out)
Optimization	Optimizer	Adam
	Learning Rate	1×10^{-4} (CheetahRun), 3×10^{-4} (H1, Berkeley), 5×10^{-4} (FHN)
	Max Gradient Norm	1.
PPO Details	Rollout Batch Size	1024 (CheetahRun, FHN), 4096 (H1, Berkeley)
	Minibatches	32 (CheetahRun), 8 (H1, Berkeley, FHN)
	PPO Epochs	16 (CheetahRun), 5 (H1, Berkeley, FHN)
	Discount Factor (γ)	1. (FHN), 0.99 (H1, Berkeley, CheetahRun)
	GAE Parameter (λ)	0.95
	Clipping Epsilon (ϵ_{clip})	0.2
	Value Loss Coef (c_{vf})	1.
	Entropy Coef (c_{ent})	0.01
	Co-state Coef ($c_{\text{co-state}}$)	0.2 (FHN), 0.1 (CheetahRun), 0.05 (humanoids)

E.4 HARDWARE

Our experiments were conducted using Quadro RTX 6000 and A100 GPUs. For the FitzHugh-Nagumo and CheetahRun, training a single seed takes 4–5 hours depending on the recurrent cell on a Quadro RTX 6000. For the H1 and Berkeley simulations, training a single seed takes 10–12 hours on an A100.