



Young, Rory (2026) *Reinforcement learning at the edge of chaos*. PhD thesis.

<https://theses.gla.ac.uk/86020/>

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk/>
research-enlighten@glasgow.ac.uk

Reinforcement Learning at the Edge of Chaos

Rory Young

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow



University
of Glasgow

April 2026

Abstract

Reinforcement learning offers a powerful data-driven framework for solving complex control tasks. However, despite strong empirical performance in simulated domains, reinforcement learning methods remain rarely deployed in real-world systems. We argue that a key reason for this is because the learned control policies do not account for the predictability of the closed-loop control interaction. Consequently, these policies frequently induce chaotic state dynamics, where small changes to the control system can lead to dramatic and unpredictable long-term behaviour. This unpredictability poses a critical barrier for real-world deployment, as chaotic behaviour can lead to potentially unsafe outcomes.

To address this issue, we propose two methods for embedding predictability into the training process. First, we introduce a Lyapunov-based regularisation method that constrains the divergence of predicted trajectories by penalising the predicted Maximal Lyapunov Exponent during policy updates. This regularisation encourages the controller to generate smoother, more predictable closed-loop dynamics and introduces a tuneable trade-off between task performance and predictability. Second, we develop a fine-tuning framework that updates pre-trained policies with a stabilising intrinsic reward function. This approach improves global stability without retraining from scratch and is compatible with any existing reinforcement learning controllers.

Together, these contributions tackle a key limitation in current reinforcement learning practice: without guaranteed predictability, even high-performing policies remain unsuitable for real-world deployment. By explicitly shaping the dynamics of closed-loop behaviour, this work advances deep reinforcement learning towards robust and reliable control in real-world control systems.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Nicolas Pugeault, for his invaluable guidance, encouragement, and insight throughout this PhD. His support has been instrumental in shaping both the direction and quality of this work.

I am also deeply grateful to Bjørn Sand Jensen and Valentin Charvet, who first introduced me to reinforcement learning and sparked the curiosity that has driven much of this research. Their early mentorship laid the foundation for my continued interest in the field.

Many thanks to the Computer Vision and Autonomous Systems group for their thoughtful feedback and stimulating discussions. Their collective input contributed greatly to the development of this thesis.

Finally, I would like to thank Karen, Robert, Calum, and Jess for their unwavering support and patience while I made my robots less wobbly. Their encouragement has been a constant source of strength, and this work would not have been possible without them.

Declaration

With the exception of chapters 1 – 3, which contain introductory material, all work in this thesis was carried out by the author unless otherwise explicitly stated.

Contents

Abstract	i
Acknowledgements	ii
Declaration	iii
List of Figures	ix
List of Tables	xi
List of Algorithms	xii
List of Symbols	xiv
List of Abbreviations	xvi
List of Publications	xviii
1 Introduction	1
1.1 Outline of this Thesis	3

2	Background	4
2.1	Sequential Decision-Making	4
2.1.1	Partially Observable Markov Decision Process	5
2.1.2	DeepMind Control Suite	7
2.2	Reinforcement Learning	11
2.2.1	Policy Architecture	11
2.2.2	Estimating Performance	12
2.2.3	Model-Free Reinforcement Learning	14
2.2.4	Model-Based Reinforcement Learning	16
2.3	Dynamical Systems	17
2.4	Chaos Theory	18
2.4.1	Sensitivity	18
2.4.2	Topological Transitivity	19
2.4.3	Dense Periodic Trajectories	19
2.4.4	Chaotic Attractor	20
2.5	Lyapunov Exponents	21
2.5.1	Definition	21
2.5.2	Geometric Interpretation	22
2.5.3	Analytic Interpretation	23

2.5.4	Estimating LEs in Differentiable Systems	24
2.5.5	Estimating LEs in Non-Differentiable Systems	25
3	Related Work	27
3.1	Control theory	28
3.1.1	Linear State Dynamics	28
3.1.2	Non-linear State Dynamics	30
3.1.3	Chaotic State Dynamics	32
3.1.4	Unknown State Dynamics	33
3.2	Reinforcement Learning	35
3.2.1	Chaos in Reinforcement Learning	35
3.2.2	Stable Reinforcement Learning	36
3.2.3	Safe Reinforcement Learning	37
3.2.4	Robust Reinforcement Learning	39
4	Identifying Chaos	40
4.1	Preliminaries	42
4.1.1	Model Training	42
4.1.2	POMDP as a Dynamical System	43
4.2	System Convergence	44
4.3	Sensitivity to Initial Conditions	48

4.3.1	Estimating Lyapunov Exponents	48
4.3.2	Spatial Distribution of the Maximal Lyapunov Exponent	51
4.4	Topological Transitivity	52
4.5	Dense Periodic Trajectories	56
4.6	Reward Predictability	58
4.7	Identifying the Impact of RL Hyperparameters on Predictability	60
4.7.1	Control Frequency	60
4.7.2	Total Time Steps	62
4.8	Discussion	64
4.8.1	Predictability of the Control Interaction	64
4.8.2	Devaney vs Lyapunov	65
5	Minimising Chaos with Predictions	68
5.1	Methods	70
5.2	Results	72
5.2.1	Experimental Setup	73
5.2.2	Predictability	73
5.2.3	Performance	75
5.2.4	Robust Control	76
5.3	Discussion	78

5.4	Limitations	79
6	Minimising Chaos with Reward	80
6.1	Methods	82
6.1.1	Pre-training for Performance	82
6.1.2	Generating the Intrinsic Reward	83
6.1.3	Fine-tuning for Predictability	84
6.2	Results	85
6.2.1	Experimental Setup	85
6.2.2	Predictability	86
6.2.3	Performance	90
6.3	Discussion	93
6.4	Limitations	94
7	Conclusion	96
7.1	Summary of Contributions	96
7.2	Future Work	101
7.2.1	Chaos in the Physical Parameter Space	101
7.2.2	Chaotic Exploration for Combinatorial Optimisation	103
7.3	Closing Remarks	105

A Appendix **114**

A.1 Hyperparameters 114

A.1.1 Chapters 4 and 5 114

A.1.2 Chapter 6 115

List of Figures

2.1	Partially Observable Markov Decision Processes	6
2.2	DeepMind Control Suite Environments	9
4.1	System sensitivity	41
4.2	Total reward	43
4.3	Attractor surface area	45
4.4	<i>Pointmass</i> convergence	46
4.5	<i>Walker Stand</i> convergence	47
4.6	<i>Walker Walk</i> convergence	47
4.7	Estimated MLE and SLE	49
4.8	<i>Walker Walk</i> sensitivity	50
4.9	Spatial distribution of the MLE	52
4.10	Wasserstein distance	54
4.11	<i>Walker Walk</i> mixing	55
4.12	Recurrence rate	57

4.13	<i>Walker Stand</i> and <i>Walker Walk</i> recurrence plots	58
4.14	Reward MLE	59
4.15	Reward sensitivity	60
4.16	Control frequency ablation study	61
4.17	Training interval ablation study	63
5.1	Maximal Lyapunov Exponent Regularisation	69
5.2	Summary statistic for Maximal Lyapunov Exponent Regularisation	74
5.3	<i>Walker Walk</i> sensitivity	74
5.4	<i>Walker Stand</i> recurrence plot	75
5.5	<i>Walker Walk</i> environment render	76
5.6	Total episode reward subject to observation noise	77
5.7	<i>Walker Stand</i> sensitivity with observation noise	77
6.1	<i>Walker Stand</i> Reward Function	81
6.2	<i>Humanoid Walk</i> sensitivity with GIFT	87
6.3	<i>Humanoid Run</i> recurrence plot with GIFT	88
6.4	Temporal analysis of GIFT state dynamics	89
6.5	Distribution of total rewards	91
6.6	<i>Humanoid Walk</i> reward stability with GIFT	92
6.7	<i>Humanoid Run</i> total reward spatial distribution	93

List of Tables

2.1	Analytic interpretation of the Lyapunov Exponents	24
4.1	Classification of the system dynamics produced by deep RL policies	67
6.1	GIFT’s impact on the average Maximal Lyapunov Exponent	87
6.2	GIFT’s impact on the average total reward	91
A.1	Hyperparameters used in Chapters 4 and 5	114
A.2	Hyperparameters used in Chapter 6	115

List of Algorithms

1	Maximal Lyapunov Exponent Regularisation	72
2	G lobal stabilisation via I ntrinsic F ine T uning	84

List of Symbols

General

Symbol	Definition
x	Scalar
$\boldsymbol{x}$	Vector
$\boldsymbol{X}$	Matrix
f	Scalar function
$\mathbf{f}$	Vector function
$\mathcal{X}$	A set
$\mathbb{P}(\mathcal{X})$	Probability distribution over $\mathcal{X}$
$\mathcal{N}$	The normal distribution

Partially Observable Markov Decision Process

Symbol	Definition
$\mathcal{S}$	State space
$\mathcal{O}$	Observation space
$\mathcal{A}$	Action space
$\boldsymbol{s}_t$	The state at time t
$\boldsymbol{o}_t$	The observation at time t
$\boldsymbol{a}_t$	The action at time t
r_t	The reward at time t
ρ_0	The initial state distribution
γ	Discount factor
$\mathbf{f}(\boldsymbol{s}, \boldsymbol{a})$	The deterministic transition function
$\mathbf{o}(\boldsymbol{s})$	The deterministic observation function
$\mathbf{r}(\boldsymbol{s}, \boldsymbol{a})$	The deterministic reward function

Reinforcement Learning

Symbol	Definition
π_θ	Control policy
Π	The set of all control policies
θ	Control policy parameters
Θ	The set of all control policy parameters
$J(\pi)$	The objective function
V_ϕ	Value function
Q_ϕ	Q function
ϕ	Value / Q function parameters

Lyapunov Exponents

Symbol	Definition
λ_1	The Maximal Lyapunov Exponent
λ_i	The i^{th} Lyapunov Exponent

List of Abbreviations

Control Systems

Abbreviation	Definition
MDP	Markov Decision Process
POMDP	Partially Observable MDP
PM	The <i>Pointmass</i> environment
CB	The <i>Cartpole Balance</i> environment
CS	The <i>Cartpole Swingup</i> environment
WS	The <i>Walker Stand</i> environment
WW	The <i>Walker Walk</i> environment
WR	The <i>Walker Run</i> environment
CR	The <i>Cheetah Run</i> environment
HS	The <i>Humanoid Stand</i> environment
HW	The <i>Humanoid Walk</i> environment
HR	The <i>Humanoid Run</i> environment

Reinforcement Learning

Abbreviation	Definition
RL	Reinforcement Learning
SAC	Soft Actor-Critic
DDPG	Deep Deterministic Policy Gradient
TD3	Twin Delayed DDPG
PPO	Proximal Policy Optimization
RSSM	Recurrent State Space Model
DR3	Dreamer V3
MLE DR3	Dreamer V3 with MLE regularisation
GIFT	Global stabilisation via Intrinsic Fine Tuning

Lyapunov Exponent

Abbreviation	Definition
LE	Lyapunov Exponent
MLE	Maximal Lyapunov Exponent
SLE	Sum of Lyapunov Exponents

List of Publications

The material presented in Chapters 4 - 6 is shared with the reinforcement learning community through these works:

- Published Work
 - Rory Young and Nicolas Pugeault. Enhancing Robustness in Deep Reinforcement Learning: A Lyapunov Exponent Approach. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 86102–86123. Curran Associates, Inc., 2024
- Work Under Review
 - Rory Young and Nicolas Pugeault. Chaos in deep reinforcement learning: Analysis and stabilisation with lyapunov exponents. 2026a
 - Rory Young and Nicolas Pugeault. Gift: Global stabilisation via intrinsic fine tuning. 2026b

Chapter 1

Introduction

The successful operation of autonomous systems relies on closed-loop controllers. Whether in autonomous vehicles navigating dense traffic or nuclear reactors maintaining thermal stability, feedback-driven control is essential to safe and stable operation in real-world systems. These controllers regulate behaviour by continuously adjusting control inputs in response to system observations, thereby forming a non-linear feedback loop that ensures sustained optimal performance.

Control theory provides the mathematical framework for designing high-performing controllers and is most commonly used for real-world control systems. These methods leverage the known system dynamics to produce control policies that make the system converge to a set of target states, while avoiding hazardous system configurations. However, while control theory excels in systems with well-defined dynamics, its applicability is limited in high-dimensional systems where these dynamics are unknown or difficult to model.

In contrast, deep reinforcement learning (RL) [Mnih et al., 2015, Lillicrap et al., 2016, Silver et al., 2016, 2017, 2018, Sutton and Barto, 2018] employs a data-driven approach, learning to produce control policies in high-dimensional systems with unknown dynamics. RL agents can discover effective control strategies without access to explicit models of system behaviour by interacting with the control system and optimising a reward signal. However, while these methods attain impressive results in simulated domains, their application to real-world control systems remains limited [Dulac-Arnold et al., 2021].

A critical and often overlooked reason for this limitation is that deep RL controllers focus primarily on reward maximisation and rarely consider the predictability of the resulting control interaction. This poses a serious issue for real-world domains, as policies can produce chaotic system dynamics, even when they achieve good system performance. In multi-agent environments, this lack of predictability is particularly problematic, as it prevents other actors from accurately anticipating an agent’s actions. For example, in the autonomous driving setting, a policy might produce driving styles that optimise travel time but are unpredictable, thereby increasing the risk of traffic collisions. In effect, this lack of predictability severely limits the applicability of deep RL in real-world domains, as reliable behaviour cannot be guaranteed.

This thesis addresses this gap by analysing the predictability of closed-loop dynamics induced by state-of-the-art deep RL policies in realistic simulated control tasks. Specifically, we examine whether deterministic policies induce chaotic state dynamics [Lyapunov, 1992, Devaney, 1992] when applied to deterministic robotic systems with non-linear contact dynamics. These chaotic behaviours are characterised by a high sensitivity to initial conditions, where small state perturbations lead to significantly different long-term outcomes despite evolving deterministically. We argue that such dynamics are detrimental for real-world applications, as it becomes impossible to accurately predict the long-term behaviour of these chaotic control systems. Consequently, this unpredictability means that safe and reliable behaviour cannot be guaranteed. To mitigate this issue, we develop novel deep RL methods that explicitly optimise not only task performance but also the predictability of the induced control interaction. By promoting predictable closed-loop dynamics, these methods improve the consistency of learned policies and enhance the feasibility of deploying deep RL in real-world control systems.

1.1 Outline of this Thesis

The remainder of this thesis is organised in the following chapters:

- In Chapter 2, we provide an overview of the relevant background necessary for this thesis. This begins by outlining the problem of sequential decision-making in the form of the Partially Observable Markov Decision Process, followed by an introduction to model-free and model-based deep reinforcement learning. We then introduce chaos theory, a branch of dynamical systems theory which analyses the predictability of non-linear systems. Finally, we provide an overview of the spectrum of Lyapunov Exponents and how they quantify the predictability of dynamical systems.
- In Chapter 3, we examine recent developments relevant to the predictability of reinforcement learning. We first examine how control theory methods establish stability under different assumptions on the system dynamics. We then review chaos in reinforcement learning and methods that incorporate stability, safety, and robustness into the learning process.
- In Chapter 4, we quantify the predictability of the control interaction produced when trained state-of-the-art deep reinforcement learning policies control simulated robots. We empirically demonstrate that these non-linear control systems produce chaotic state dynamics and thus cannot provide the predictability requirements necessary for real-world deployment.
- In Chapter 5, we focus on improving the predictability of the control interaction by introducing Maximal Lyapunov Exponent regularisation for Dreamer V3. This method adds an additional term to the policy loss, which constrains the variance between predicted future states during updates. We demonstrate that weighting this additional term introduces a trade-off between predictability and performance.
- In Chapter 6, we focus on improving the predictability of the control interaction by introducing a fine-tuning method that stabilises policies with an intrinsic reward function. This additional reward function can be used in a supplementary training phase to stabilise standard high-performing policies. Empirically, we demonstrate that this method can maintain performance comparable to that of an unstable policy while significantly improving predictability.
- In Chapter 7, we summarise the findings of this thesis and propose new research directions.

Chapter 2

Background

This chapter introduces the fundamentals of reinforcement learning, chaos theory, and the spectrum of Lyapunov Exponents. In doing so, we formalise the notation that will be used throughout this thesis. We start by outlining the general problem of sequential decision-making in the form of the Partially Observable Markov Decision Process in Section 2.1 and then outline how model-free and model-based reinforcement learning can be used to find approximate solutions in Section 2.2. We then briefly introduce dynamical systems in Section 2.3 and subsequently discuss chaos theory, a branch of dynamical systems theory that is used to interpret complex, high-dimensional systems with non-linear feedback loops in Section 2.4. Finally, Section 2.5 outlines how Lyapunov Exponents can be used to identify the predictability of a dynamical system.

2.1 Sequential Decision-Making

In this section, we formalise the problem of sequential decision-making for continuous control systems. We start by outlining the general definition of the deterministic Partially Observable Markov Decision Process and the objective of a sequential decision-making agent in Section 2.1.1. In Section 2.1.2, we then provide an overview of the continuous control systems which are used throughout this thesis.

2.1.1 Partially Observable Markov Decision Process

Control systems play a crucial role in the management of various industrial processes, ranging from high-altitude weather balloons [Bellemare et al., 2020] to nuclear fusion reactors [Degraeve et al., 2022]. Although the specifics of these tasks can vary significantly, the underlying structure of the interaction can generally be represented by the deterministic Partially Observable Markov Decision Process (POMDP) [Åström, 1965], a generalisation of the Markov Decision Process (MDP) [Sutton, 1991] that accounts for incomplete state information. This framework represents the control interaction as the repeated interaction between a control policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and an environment with the following attributes:

- A set of all possible states $\mathcal{S} \subseteq \mathbb{R}^N$
- A set of all possible observations $\mathcal{O} \subseteq \mathbb{R}^L$
- A set of all possible actions $\mathcal{A} \subseteq \mathbb{R}^M$
- A set of initial states $\rho_0 \subseteq \mathcal{S}$
- A deterministic transition function $\mathbf{f} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
- A deterministic observation function $\mathbf{o} : \mathcal{S} \rightarrow \mathcal{O}$
- A deterministic reward function $\mathbf{r} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$
- A discount factor $\gamma \in [0, 1)$

Initially, the system starts in the state $\mathbf{s}_0$, which is drawn from the initial state distribution ρ_0 . For each discrete time step thereafter, the observation function encodes this state into an observation ($\mathbf{o}_t = \mathbf{o}(\mathbf{s}_t)$) which is passed to a control policy. The policy then generates an action ($\mathbf{a}_t$) which is applied to the control system. As a result of this signal, the environment transitions to the next state according to the transition function $\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)$. Finally, the agent receives a reward signal $r_t = \mathbf{r}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ which evaluates the optimality of the current state–action pair. The agent continues applying actions until it reaches a terminal state or a predefined time limit is exceeded, at which point the environment is reset for a new episode. An illustration of this process is provided in Figure 2.1.

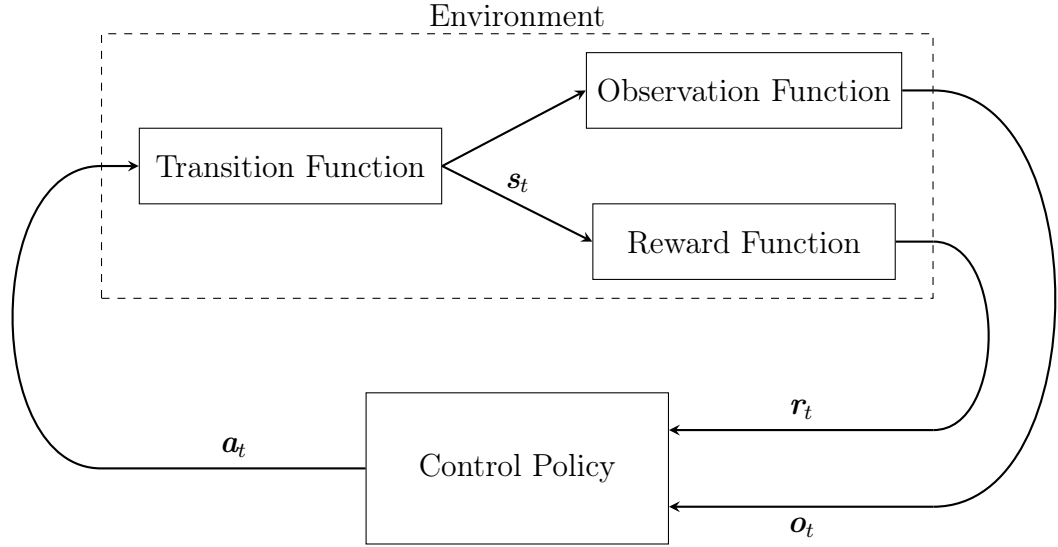


Figure 2.1: Partially Observable Markov Decision Processes

The high-level objective in the POMDP is to generate a policy $\pi \in \Pi$ which consistently produces good system performance over a fixed time horizon $T \in \mathbb{N}$. Formally, this objective is known as the return and is defined as the sum of discounted rewards (Equation 2.1), with the discount factor (γ) being used to incentivise more certain short-term performance over potential long-term gains.

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0} \left[\sum_{t=0}^T \gamma^t \times r(s_t, a_t) \mid s_{t+1} = \mathbf{f}(s_t, a_t), a_t = \pi(s_t) \right] \quad (2.1)$$

Equation 2.1 therefore provides an ordering over all possible policies (Π), with the optimal policy being defined as the policy which satisfies:

$$\pi^* = \arg \max_{\pi \in \Pi} (J(\pi)) \quad (2.2)$$

Abstracting control interactions as a deterministic Partially Observable Markov Decision Process offers a unified framework for modelling a wide range of sequential decision-making problems. A key advantage of this approach is its reliance on the Markov property, which allows decisions to be made based solely on the current state, without requiring access to the full history. Furthermore, because the transition function, observation function, and policy are all deterministic at inference, the system will always generate the exact same state and action trajectory given the same initial state. Consequently, this formulation enables systematic reasoning about both the performance of control strategies and the evolution of system dynamics over time.

While this deterministic formulation provides a rigorous analytical baseline, it represents a significant simplification of inherently stochastic real-world control systems. Specifically, it assumes the absence of sensor noise in the observation function, the exact execution of the policy’s actions without actuator variance, and a transition function free from unmodelled dynamics. However, by restricting the environment to perfect determinism, any unpredictability observed in the closed-loop system can be directly attributed to the learned policy and control system, rather than unmodelled external noise.

2.1.2 DeepMind Control Suite

Throughout this thesis, we utilise continuous control environments sampled from the DeepMind Control Suite [Tunyasuvunakool et al., 2020]. This framework has emerged as the standard benchmark for evaluating reinforcement learning algorithms in continuous domains, as it provides a diverse set of physically grounded deterministic control tasks, each with an interpretable reward structure. As a result, this suite offers a consistent baseline that enables fair comparison between deep reinforcement learning methods.

While this suite contains a diverse set of control environments, we restrict our investigation to systems in which the resulting closed-loop behaviour can be meaningfully analysed. Specifically, we focus on continuous deterministic control systems which meet the following criteria:

1. **Physical Rigidity:** The control system must contain only rigid bodies. This ensures that any observed instability arises from the non-linear feedback loop of the controller and not the stochastic complexities of soft-body or fluid simulations.
2. **Targeted Actuation:** The control system must exhibit a structure where the primary degrees of freedom are directly influenced by at least one actuator. This ensures that the control policy maintains sufficient authority to shape the resulting closed-loop dynamics.
3. **Purely Dynamical State Space:** The control system must use a state space consisting exclusively of joint positions and velocities. This ensures that episode-specific contextual information does not obscure the analysis of the system’s long-term state evolution.

4. **Algorithmic Feasibility:** The control system must demonstrate high performance when controlled by state-of-the-art deep reinforcement learning methods. This ensures that any observed instability arises from the optimised policy and not a learning failure.

Applying these criteria limits our investigation to the *Pointmass*, *Cartpole*, *Walker*, *Cheetah*, and *Humanoid* control systems (Figure 2.2). Furthermore, each system is associated with one to three distinct tasks, thereby giving a total of ten environments for this study.

Each of these environments is defined using the MuJoCo physics engine [Todorov et al., 2012], which provides high-fidelity simulation of multi-body dynamics. The environments evolve deterministically according to Newtonian mechanics, with control inputs interpreted as torques applied to the actuated joints of each robot. All physical quantities are computed in single-precision floating-point format and expressed in SI units, with lengths in metres and angles in radians. Episodes terminate after a fixed horizon of 1000 steps, which corresponds to 10 - 25 seconds of simulated time. The reward function for each environment is deterministic and generates a continuous scalar value between zero and one, with a reward of one indicating near-optimal performance and zero reflecting failure. Consequently, the total reward achievable per episode is 1000 for every environment. Despite these common design elements, each environment introduces distinct control challenges, which are outlined below.

Pointmass

The *Pointmass* control system consists of a simple point mass confined to a frictionless two-dimensional square with side lengths of 0.6 metres. When the system is initialised, the point mass is placed at a random location within the plane with zero initial velocity. At each time step, the control policy receives the full state vector as the observation, which includes the x and y positions of the point mass and their corresponding velocities. The policy then outputs accelerations in the x and y directions, which are applied to the point mass. This control system only contains one task, which provides a reward of 1 when the point mass is within 0.015 metres of the centre of the plane and the magnitude of the control input is low.

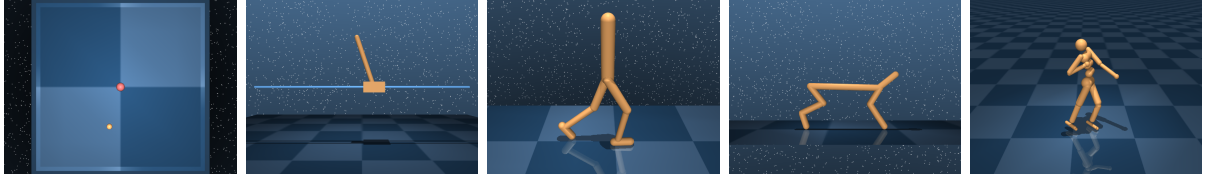


Figure 2.2: Visualisation of the control systems sampled from the DeepMind Control Suite which are used throughout this thesis. From left to right, we have: *Pointmass*, *Cartpole*, *Walker*, *Cheetah*, and *Humanoid*.

Cartpole

The *Cartpole* control system consists of a pole attached to a cart that moves along a one-dimensional frictionless track. The pole is connected to the cart via an unactuated hinge joint, which is free to swing in the vertical plane. The underlying state vector comprises the cart position, cart velocity, pole angle, and pole angular velocity. The observation provided to the control policy is the full state vector, but with the pole angle replaced with its sine and cosine. The action is a scalar horizontal force applied to the cart, and the objective is to keep the pole balanced in the upright position while keeping the cart near the centre of the track. A large reward is given at each time step when the pole is vertical, the cart is near the centre of the track, and the control input remains small.

This control system provides two tasks. In the *Balance* variant, the episode starts with the pole already upright and the cart near the centre, requiring the policy to maintain stability. In the *Swingup* variant, the episode begins with the pole hanging downwards, and the policy must first swing the pole upwards before balancing it.

Walker

The *Walker* control system features a planar bipedal robot composed of a torso, two thighs, two shins, and two feet, operating in the x - z plane. The state is represented by an 18-dimensional vector comprising the torso position in the x and z plane, torso angles, hip angles, knee angles, and ankle angles, along with their respective velocities. The control policy receives a 24-dimensional observation vector, which includes the torso height, the sine and cosine of each angle, and the velocities corresponding to the state vector. Importantly, this observation does not include the horizontal position, which means this is a partially observable system. Finally, the actions consist of continuous-valued torques applied to the actuated joints (i.e., hips, knees, and ankles).

The control system includes three tasks of increasing difficulty: *Stand*, *Walk*, and *Run*. Each task provides a high reward when the torso height is maintained 1.2 metres above the ground and the torso angle is close to vertical. In addition, the *Walk* and *Run* tasks require the robot’s centre of mass to move horizontally at 1 and 8 metres per second, respectively.

Cheetah

The *Cheetah* control system features a planar quadruped with two legs and a torso, operating in the x - z plane. The state is represented by an 18-dimensional vector comprising the torso position in the x and z planes, torso angles, each joint angle and velocities. The observation vector provided to the control policy is identical to the state vector, except that it excludes the torso’s horizontal position, thereby making this a partially observable system. The policy then generates a 6-dimensional continuous vector of joint torques. This control system is used exclusively in the *Run* task, which provides high rewards when the centre of mass is moving horizontally at 10 metres per second.

Humanoid

The *Humanoid* control system features a full-body bipedal robot composed of a head, torso, two arms, and two legs operating in three spatial dimensions. The state vector comprises torso position, torso orientation, torso velocities, joint angles, and angular velocities. The observation provided to the control policy is the state vector, with the joint angles replaced with their respective sines and cosines and without the x and y positions of the torso, thereby making it a partially observable system. The action space consists of continuous torques applied to each actuated joint.

The control system includes three tasks of increasing difficulty: *Stand*, *Walk*, and *Run*. Each task provides a high reward when the head height is maintained 1.4 metres above the ground, the torso angle is close to vertical, and the magnitude of control input is low. In addition, the *Stand*, *Walk*, and *Run* tasks require the centre of mass to move horizontally at 0, 1 and 10 metres per second, respectively.

2.2 Reinforcement Learning

Deep reinforcement learning (RL) is a branch of machine learning which finds approximate solutions to POMDPs with unknown transition, reward, and observation functions. In this setting, an RL agent interacts with the black box environment, identifies which states achieve high rewards, and iteratively optimises a policy ($\pi \in \Pi$) to maximise $J(\pi)$. As a result, policies trained via deep RL can control complex, continuous, high-dimensional control systems without the need for explicit transition dynamics.

In this section, we introduce the fundamentals for deep RL, which are used throughout this thesis. Specifically, Section 2.2.1 describes the types of policies used in the deep RL framework, while Section 2.2.2 outlines methods for evaluating their performance. Sections 2.2.3 and 2.2.4 then discuss how model-free and model-based approaches generate effective control policies. For a more comprehensive introduction to reinforcement learning, we refer the reader to the work by Sutton and Barto [2018].

2.2.1 Policy Architecture

In the deep RL framework, control policies are commonly modelled as a neural network (π_θ) with control parameters θ . Neural networks are parameterised universal function approximators, capable of learning complex, non-linear mappings between arbitrary input and output distributions. This allows deep RL control policies to map high-dimensional or visual observations directly to continuous actions. Consequently, deep RL becomes more feasible for real-world control systems with complex dynamics and sensory inputs.

Formally, a deep RL policy is defined as a mapping from the observation space to a probability distribution over actions, ($\pi_\theta : \mathcal{O} \rightarrow \mathbb{P}(\mathcal{A})$). While this mapping can theoretically represent any valid probability distribution, it is most commonly parameterised as a multivariate Gaussian distribution for continuous control systems. In this setting, the neural network policy outputs a mean vector $\mu_\theta(\mathbf{o})$ and a diagonal covariance matrix $\Sigma_\theta(\mathbf{o})$, which define the multivariate Gaussian distribution:

$$\pi_\theta(\mathbf{o}) = \mathcal{N}(\mu_\theta(\mathbf{o}), \Sigma_\theta(\mathbf{o})) \quad (2.3)$$

This probabilistic formulation encourages exploration during training by allowing the agent to sample different actions for the same observation and discover effective control strategies. During evaluation or deployment, the policy typically acts deterministically by selecting the most probable action (i.e. $\pi_\theta = \mu_\theta$). This ensures consistent and predictable behaviour, which is especially important in real-world applications.

Finally, as the policy is defined by its parameter vector θ , the space of all policies Π is the set of all possible network weights $\Theta \subseteq \mathbb{R}^d$. As such, the goal of an RL agent is to identify the parameters $\theta^* \in \Theta$ such that:

$$\theta^* = \arg \max_{\theta \in \Theta} (J(\pi_\theta)) \quad (2.4)$$

This objective means that RL becomes an optimisation problem over the policy parameters. The goal is to find the neural network weights (θ) that yield the highest expected return. Importantly, this optimisation must be performed without access to the underlying reward, transition, or observation functions. Instead, the agent must learn entirely from experience by interacting with the environment, observing outcomes, and updating θ to improve long-term performance.

2.2.2 Estimating Performance

Value functions are typically used to estimate the total expected return for a given observation or observation–action pair. These functions play a crucial role in guiding the agent’s decision-making by indicating the most advantageous actions across different scenarios. Two fundamental types of value functions in RL are the value function and the Q function.

Value Function

The value function ($V : \mathcal{O} \rightarrow \mathbb{R}$) evaluates the performance of a policy starting with an observation $\mathbf{o} \in \mathcal{O}$ and following π_θ thereafter. Formally, it is defined as:

$$V(\mathbf{o}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\mathbf{o}_t)} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \times V(\mathbf{o}_{t+1}) \mid \mathbf{o}_{t+1} = \mathbf{o}(\mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)) \right] \quad (2.5)$$

In effect, the value function estimates the long-term benefit of being in a given state and following the current policy thereafter. By learning accurate value estimates, the agent can evaluate the desirability of different states without simulating full trajectories.

Q Function

Similarly, the Q function ($Q^\pi : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$) evaluates the performance of a policy when taking an action $\mathbf{a} \in \mathcal{A}$ after observing $\mathbf{o} \in \mathcal{O}$ and following the policy thereafter. Formally, this function is defined as:

$$Q(\mathbf{o}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \times \mathbb{E}_{\mathbf{a}_{t+1} \sim \pi(\mathbf{o}_{t+1})} \left[Q(\mathbf{o}_{t+1}, \mathbf{a}_{t+1}) \mid \mathbf{o}_{t+1} = \mathbf{o}(\mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)) \right] \quad (2.6)$$

The Q function represents the expected long-term return of taking a specific action in a given state and subsequently following the current policy. By estimating action-specific returns, the agent can compare alternative actions and prioritise those that provide higher expected rewards.

Learning Value Functions

Temporal difference learning [Sutton, 1988] is a widely used approach for estimating value functions in reinforcement learning. This method updates parameterised versions of the value function (V_ϕ) and Q function (Q_ϕ) based on bootstrapped estimates of future rewards. Specifically, given an observed transition $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1}, r_t)$ and learning rate $\alpha \in \mathbb{R}$, the value function and Q function can be iteratively updated using Equation 2.7 and Equation 2.8 respectively.

$$V_\phi(\mathbf{o}_t) \leftarrow V_\phi(\mathbf{o}_t) + \alpha [r_t + \gamma V_\phi(\mathbf{o}_{t+1}) - V_\phi(\mathbf{o}_t)] \quad (2.7)$$

$$Q_\phi(\mathbf{o}_t, \mathbf{a}_t) \leftarrow Q_\phi(\mathbf{o}_t, \mathbf{a}_t) + \alpha \left[r_t + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_\phi(\mathbf{o}_{t+1}, \mathbf{a}') - Q_\phi(\mathbf{o}_t, \mathbf{a}_t) \right] \quad (2.8)$$

Learning value functions and Q functions provide a principled mechanism for estimating the long-term performance of a control policy. Together, they support efficient learning by reducing the need for exhaustive trajectory sampling and facilitating generalisation across similar states and actions.

2.2.3 Model-Free Reinforcement Learning

During training, a model-free RL agent iteratively updates its control policy parameters to maximise the objective function (Equation 2.4). However, as the transition dynamics and reward function of the environment are unknown, the gradient of this objective cannot be computed analytically. Instead, model-free methods estimate the gradient and update the parameters of the policy via gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla J(\pi_\theta) \quad (2.9)$$

To approximate the gradient, model-free methods rely on sampled interactions with the environment. These methods estimate the expected return of a policy by collecting trajectory data and applying either stochastic or deterministic policy gradient techniques.

Stochastic Policy Gradient

Stochastic policy gradient methods leverage the policy gradient theorem [Williams, 1992] to iteratively update θ in the direction that maximises $J(\pi_\theta)$. This theorem states that for a probabilistic control policy ($\pi_\theta : \mathcal{O} \rightarrow \mathbb{P}(\mathcal{A})$), the gradient of $J(\pi_\theta)$ is given by:

$$\nabla J(\pi_\theta) = \mathbb{E}_{\mathbf{s}_0 \sim \rho_o, \mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t)} \left[\sum_{t=0}^{\infty} \left(\nabla_\theta \log(\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)) \times \sum_{k=0}^{\infty} r_{t+k} \right) \right] \quad (2.10)$$

Intuitively, the gradient vector increases the probability of selecting action $\mathbf{a}$ after observing $\mathbf{o}$ in proportion to the measured return. As a result, the policy parameters move in the direction in which actions produce higher total rewards.

Applying the policy gradient theorem to Equation 2.9 gives the REINFORCE policy estimator. While this method is straightforward, it often suffers from high variance in the gradient estimates, leading to slow convergence. Several methods exist for reducing this variance, including: substituting the measured sum of reward with a learned Q function [Sutton et al., 1999], enforcing a constraint on the step size to avoid large deviations from the current policy [Schulman et al., 2015], or using a clipped objective [Schulman et al., 2017].

Deterministic Policy Gradient

Deterministic policy gradient methods [Silver et al., 2014] estimate the gradient of the return function using a deterministic policy and a differentiable Q function. Unlike their stochastic counterparts, which rely on sampling actions from a probability distribution, deterministic policy gradient methods directly optimise the policy using the gradient of the Q function with respect to the policy parameters:

$$\begin{aligned}\nabla J(\pi_\theta) &= \mathbb{E}_{\mathbf{s}_0 \sim \rho_0} [\nabla_\theta Q_\phi(\mathbf{o}_t, \pi_\theta(\mathbf{o}_t))] \\ &= \mathbb{E}_{\mathbf{s}_0 \sim \rho_0} [\nabla_\theta \pi_\theta(\mathbf{o}_t) \nabla_{\mathbf{a}_t} Q_\phi(\mathbf{o}_t, \mathbf{a}_t) | \mathbf{a}_t = \pi_\theta(\mathbf{o}_t)]\end{aligned}\tag{2.11}$$

While this exhibits lower gradient variance than the stochastic policy gradient, it suffers from practical limitations including sensitivity to Q function approximation errors, poor exploration, and instability in high-dimensional action spaces. These issues can lead to suboptimal policies and inefficient training. To address these challenges, Deep Deterministic Policy Gradient (DDPG) [Lillicrap et al., 2016] introduces experience replay, target networks, and temporally correlated noise for improved stability and exploration. Twin Delayed Deep Deterministic Policy Gradient (TD3) [Fujimoto et al., 2018] further reduces overestimation bias with clipped double Q-learning, delayed policy updates, and target policy smoothing. Finally, Soft Actor-Critic (SAC) [Haarnoja et al., 2018] takes a stochastic approach, incorporating entropy regularisation to encourage exploration and avoid premature convergence.

2.2.4 Model-Based Reinforcement Learning

Model-based reinforcement learning refers to any RL approach that incorporates a learned model of the environment’s transition dynamics. Unlike model-free methods, which rely solely on observed transitions from the environment, model-based RL constructs an internal predictive model that approximates how the environment responds under the influence of a given control policy. This model enables the agent to simulate future trajectories, plan more effectively, and refine value estimates without requiring constant interaction with the real environment. As a result, model-based RL can reduce the number of real-world samples needed during training, leading to improved sample efficiency.

Dreamer V3 [Hafner et al., 2025] is a notable state-of-the-art model-based RL algorithm that leverages a learned model of the system dynamics to update its policy. Specifically, this dynamics model is a Recurrent State Space Model (RSSM) operating over a discrete hidden space $\mathcal{Z}$ and continuous hidden space $\mathcal{H}$ using:

- An Encoder $\mathbf{q}_\phi : \mathcal{O} \times \mathcal{H} \rightarrow \mathbb{P}(\mathcal{Z})$
- A Decoder $\mathbf{p}_\phi : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{P}(\mathcal{O})$,
- A Dynamics Predictor $\mathbf{q}_\phi : \mathcal{H} \rightarrow \mathbb{P}(\mathcal{Z})$
- A Sequence Model $\mathbf{f}_\phi : \mathcal{H} \times \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{H}$

These components are used to generate predicted trajectories of observations ($\mathbf{o}_t$), bootstrapped λ -return trajectories (R_t^λ) [Sutton and Barto, 2018], and state value trajectories ($V_\phi(\mathbf{o}_t)$) over a short time horizon $T \in \mathbb{N}$. The policy is then trained to maximise the normalised advantage estimates using REINFORCE gradients [Williams, 1992] and an entropy regulariser ($H[\cdot]$) [Williams and Peng, 1991] with weighting coefficient η :

$$\mathcal{L}^{\text{Dr3}}(\theta) \doteq - \sum_{t=1}^T \left[\log(\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)) \times \text{sg} \left(\frac{R_t^\lambda - V_\phi(\mathbf{o}_t)}{\max(1, S)} \right) + \eta H[\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)] \right] \quad (2.12)$$

This formulation enables Dreamer V3 to train effective policies entirely within its learned model, using imagined rollouts to optimise behaviour. As a result, it achieves state-of-the-art performance across diverse benchmarks, matching or surpassing model-free methods while requiring significantly fewer environment interactions, thereby demonstrating the scalability and sample efficiency of modern model-based reinforcement learning.

2.3 Dynamical Systems

In this section, we define discrete dynamical systems, a general-purpose mathematical framework for describing the evolution of systems over time. This perspective provides the foundation for subsequent discussions of chaos and predictability in reinforcement learning environments.

Formally, a discrete-time dynamical system is defined by a state space $\mathcal{S} \subseteq \mathbb{R}^N$ and deterministic transition function $\mathbf{f} : \mathcal{S} \rightarrow \mathcal{S}$. Each state $\mathbf{s} \in \mathcal{S}$ represents a unique system configuration, with each dimension of $\mathbf{s}$ corresponding to a specific degree of freedom. Given an initial state $\mathbf{s}_0 \in \mathcal{S}$, the evolution of the system is governed by repeated application of the transition function. Specifically, the state at time $t \in \mathbb{N}$ is given by the repeated composition of $\mathbf{f}$:

$$\mathbf{s}_t = \mathbf{f}^t(\mathbf{s}_0) = \underbrace{\mathbf{f} \circ \mathbf{f} \circ \cdots \circ \mathbf{f}}_{t \text{ times}}(\mathbf{s}_0). \quad (2.13)$$

The resulting sequence of states $\mathbf{s}_t$ forms a trajectory, which reveals how the system evolves from its initial condition. Analysing such trajectories reveals key dynamical properties, including fixed points, periodic trajectories, attractors, and divergent behaviour. These geometric features underpin much of our understanding of complex control systems, particularly when dealing with non-linear dynamics or feedback-driven interactions. In high-dimensional settings, trajectory analysis becomes essential for identifying patterns of stability, sensitivity, and long-term predictability.

2.4 Chaos Theory

In this section, we introduce the fundamentals of chaos theory as set out by Devaney [1992]. This definition states that a dynamical system is chaotic if and only if it is sensitive to initial conditions, displays topological transitivity, and contains dense periodic trajectories. These three elements of chaotic dynamics are addressed in Sections 2.4.1 - 2.4.3 respectively. Finally, we introduce the chaotic attractor in Section 2.4.4.

2.4.1 Sensitivity

Sensitivity to initial conditions is often regarded as the most significant feature of chaotic systems. Intuitively, it captures the idea that small differences in initial states can lead to large divergences in long-term behaviour. This property underpins the unpredictability commonly associated with chaos, where even infinitesimal perturbations in a system's starting point may render long-term forecasts practically impossible. Formally, sensitivity to initial conditions is defined as follows:

Definition 1 (Sensitivity)

A dynamical system with continuous state space ($\mathcal{S} \subseteq \mathbb{R}^N$) and discrete deterministic transition function ($\mathbf{f} : \mathcal{S} \rightarrow \mathcal{S}$) is sensitive to initial conditions if there exists $\beta > 0$ such that for all $\mathbf{x} \in \mathcal{S}$ and $\epsilon > 0$ there exists $\mathbf{y} \in \mathcal{S}$ and $k \in \mathbb{N}$ such that:

$$0 < |\mathbf{x} - \mathbf{y}| < \epsilon \implies |\mathbf{f}^k(\mathbf{x}) - \mathbf{f}^k(\mathbf{y})| \geq \beta \quad (2.14)$$

This definition of sensitivity implies that for any given initial state ($\mathbf{x}$) it is always possible to select another state ($\mathbf{y}$) which is arbitrarily close to $\mathbf{x}$, such that the trajectory of $\mathbf{y}$ eventually diverges from that of $\mathbf{x}$ by a non-trivial distance β . Importantly, this definition does not require the trajectories of $\mathbf{x}$ and $\mathbf{y}$ to remain separated by β for all future time steps. Rather, it suffices that there exists at least one point in time where the trajectories exhibit significant divergence.

2.4.2 Topological Transitivity

Topological transitivity formalises the notion that a chaotic system can evolve from one region of its state space to any other, given sufficient time. It implies that the system cannot be partitioned into smaller, invariant subsystems with disjoint dynamics. This property is crucial to the unpredictability of chaotic behaviour, ensuring that local perturbations may have global consequences. Formally, topological transitivity is defined as follows:

Definition 2 (Topological Transitivity)

A dynamical system with continuous state space ($\mathcal{S} \subseteq \mathbb{R}^N$) and discrete deterministic transition function ($\mathbf{f} : \mathcal{S} \rightarrow \mathcal{S}$) is transitive if for all open subsets of the state space $A, B \subseteq \mathcal{S}$ there exists $k \in \mathbb{N}$ such that:

$$\mathbf{f}^k(A) \cap B \neq \emptyset \quad (2.15)$$

This definition of transitivity states that the system's dynamics can connect any two open regions within the state space, irrespective of their spatial separation. In other words, for any pair of regions, there exists a trajectory that begins in one and eventually reaches the other. This ensures that the system cannot be decomposed into separate parts that evolve independently, and that its long-term behaviour is fundamentally intertwined across the entire state space.

2.4.3 Dense Periodic Trajectories

The final requirement for a system to exhibit chaotic behaviour is that periodic trajectories must densely cover the state space. This condition ensures the presence of underlying regularity, distinguishing chaotic dynamics from purely random processes. Formally, it is defined as follows:

Definition 3 (Dense Periodic Trajectories)

A dynamical system with continuous state space ($\mathcal{S} \subseteq \mathbb{R}^N$) and discrete deterministic transition function ($\mathbf{f} : \mathcal{S} \rightarrow \mathcal{S}$) contains dense periodic trajectories if for all $\mathbf{x} \in \mathcal{S}$ and $\epsilon > 0$ there exists $\mathbf{y} \in \mathcal{S}$ and $k \in \mathbb{N}$ such that:

$$|\mathbf{x} - \mathbf{y}| < \epsilon \quad \text{and} \quad \mathbf{f}^k(\mathbf{y}) = \mathbf{y}$$

This definition of dense periodic trajectories states that for any given state ($\mathbf{x}$) it is always possible to select another state ($\mathbf{y}$) which is arbitrarily close to $\mathbf{x}$, such that the trajectory of $\mathbf{y}$ repeats every k steps. In other words, periodic trajectories are dense in $\mathcal{S}$. This condition ensures that the state space contains repeating structures, even in regions where most trajectories behave aperiodically and exhibit sensitive dependence on initial conditions.

2.4.4 Chaotic Attractor

In many dynamical systems, the conditions for chaos are not satisfied throughout the entire state space, but rather within a restricted subset of it. Formally, this region is known as a chaotic attractor and is defined as:

Definition 4 (Chaotic Attractor)

A dynamical system with continuous state space ($\mathcal{S} \subseteq \mathbb{R}^N$) and discrete deterministic transition function ($\mathbf{f} : \mathcal{S} \rightarrow \mathcal{S}$) contains a chaotic attractor at $\mathcal{A} \subseteq \mathcal{S}$ if the following properties hold:

- **Attracting:** *There exists an open neighbourhood $\mathcal{U}$ with $\mathcal{A} \subset \mathcal{U} \subseteq \mathcal{S}$ such that for all $\mathbf{x} \in \mathcal{U}$ we have $\lim_{k \rightarrow \infty} \mathbf{f}^k(\mathbf{x}) \in \mathcal{A}$.*
- **Invariant:** *For all $\mathbf{x} \in \mathcal{A}$ we have $\mathbf{f}(\mathbf{x}) \in \mathcal{A}$.*
- **Chaotic:** *The dynamics of $\mathbf{f}$ restricted to $\mathcal{A}$ satisfy the three conditions of chaos.*

Chaotic attractors often arise in dissipative systems, where trajectories from a wide range of initial conditions are drawn into a lower-dimensional subset of the state space exhibiting chaotic behaviour. Classic examples include the Lorenz [Lorenz, 1963] and Rössler [Rössler, 1976] attractors, which demonstrate complex, aperiodic dynamics confined to well-defined geometric structures. The notion of a chaotic attractor is fundamental in understanding how deterministic systems produce long-term behaviour that is both bounded and unpredictable.

2.5 Lyapunov Exponents

In this section, we introduce the spectrum of Lyapunov Exponents (LEs) [Lyapunov, 1992, Ruelle, 1979]. We begin by providing a formal definition of LEs in Section 2.5.1, then explain their geometric and analytic interpretation in Sections 2.5.2 & 2.5.3 respectively. In Sections 2.5.4 & 2.5.5, we then describe how these values can be estimated in systems with differentiable and non-differentiable transition functions.

2.5.1 Definition

The spectrum of Lyapunov Exponents characterises the long-term predictability of non-linear dynamical systems. In general, a dynamical system operating over an N -dimensional state space has N ordered Lyapunov Exponents ($\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$), each representing the exponential deformation rate of the state space in a unique direction. Formally, given an initial state $\mathbf{s}_0 \in \mathcal{S}$, these values are defined as:

$$\lambda_i = \lim_{T \rightarrow \infty} \frac{1}{T} \ln \left(\sigma_i \left(\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_0} \right) \right) \quad (2.16)$$

where $\sigma_i \left(\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_0} \right)$ are the singular values of the Jacobian matrix of the transition function between $t=0$ and $t=T$ [Cencini et al., 2009, Parker and Chua, 2011]. In effect, the Lyapunov Exponents are a generalisation of the singular values of the Jacobian matrix over an extended state trajectory. Each exponent quantifies the asymptotic exponential rate of divergence or convergence of trajectories along orthogonal directions in the tangent space.

2.5.2 Geometric Interpretation

In a geometric setting, the Lyapunov Exponents describe how infinitesimal volumes in the tangent space evolve under the flow of a dynamical system. Specifically, the volume of the k -dimensional parallelepiped in the tangent space defined by $k \in \mathbb{N}$ orthonormal perturbation vectors ($\text{Vol}_k(t)$) grows at an exponential rate governed by the sum of the first k Lyapunov Exponents:

$$\text{Vol}_k(t) \approx \text{Vol}_k(0) \times e^{(t \times \sum_{i=1}^k \lambda_i)}. \quad (2.17)$$

In effect, the Lyapunov Exponents provide a general method for estimating the long-term k -dimensional volume between k state trajectories in the N -dimensional state space, with two significant cases occurring when $k = 1$ and $k = N$.

Maximal Lyapunov Exponent

When $k = 1$, Equation 2.17 provides an estimate of the linear distance between two state trajectories which have similar but not identical initial positions. Specifically, the distance between two state trajectories $\mathbf{s}_t, \hat{\mathbf{s}}_t \in \mathcal{S}$ with $0 < \|\mathbf{s}_0 - \hat{\mathbf{s}}_0\| \ll 1$ can be approximated by:

$$\|\mathbf{s}_t - \hat{\mathbf{s}}_t\| \approx \|\mathbf{s}_0 - \hat{\mathbf{s}}_0\| e^{\lambda_1 \times t} \quad (2.18)$$

As a result, the Maximal Lyapunov Exponent (MLE/ λ_1) governs the exponential rate at which these two trajectories diverge in the state space and is formally defined as:

$$\lambda_1 = \lim_{t \rightarrow \infty} \lim_{|\mathbf{s}_0 - \hat{\mathbf{s}}_0| \rightarrow 0} \frac{1}{t} \ln \left(\frac{|\mathbf{s}_t - \hat{\mathbf{s}}_t|}{|\mathbf{s}_0 - \hat{\mathbf{s}}_0|} \right) \quad (2.19)$$

A positive MLE indicates the system is highly sensitive to small changes in initial conditions, as the state trajectories diverge at an exponential rate. In contrast, a negative MLE indicates that the system is predictable as the state trajectories converge over time.

Sum of Lyapunov Exponents

Similarly, the growth rate of an N -dimensional volume is governed by the Sum of Lyapunov Exponents ($\text{SLE} / \sum_{i=1}^N \lambda_i$). This quantity reflects the exponential rate at which infinitesimal volumes in the full N -dimensional state space expand or contract over time. A negative SLE indicates that the system is dissipative and that volumes contract to a lower-dimensional attractor. Furthermore, an SLE of zero indicates that volumes are preserved under the influence of the transition function, while a positive SLE indicates that volumes expand indefinitely.

2.5.3 Analytic Interpretation

Overall, the predictability of a dynamical system can be determined using the MLE and the SLE, as outlined in Table 2.1. When the SLE is negative, the system forms an attractor as the contraction rate of the state space outweighs the expansion rate [Shao et al., 2018]. Furthermore, if all LEs are negative, the system will approach a fixed point, and if the LEs are less than or equal to zero, then the system will converge to a trajectory.

However, when the MLE is positive and the SLE is negative, similar trajectories will diverge at an exponential rate but remain confined to a chaotic attractor [Devaney, 1992, Lorenz, 1963]. This bounded exponential divergence means that trajectories in this region of the state space are unstable and only replicable given the exact same starting state. Small perturbations to the starting state will produce significantly different long-term outcomes, which appear random and uncorrelated. As a result, it is impossible to predict the long-term behaviour of a chaotic system given an approximation of the initial state.

Finally, when the SLE is positive, the system does not form an attractor as the expansion rate exceeds the contraction rate. In this instance, trajectories experience persistent divergence, meaning that small perturbations in the initial state lead to rapidly amplifying differences over time. Unlike chaotic attractors, where trajectories remain confined within a bounded region of the phase space, systems with a positive SLE exhibit unbounded growth. Such behaviour is characteristic of systems undergoing runaway dynamics, where instability drives uncontrolled state evolution.

Table 2.1: Classification of system dynamics characterised by the spectrum of Lyapunov Exponents

Type of dynamics	Lyapunov Exponents
Attracting Fixed Point	$0 > \lambda_1 \geq \dots \geq \lambda_n$
Attracting Trajectory	$\lambda_1 = 0$ $0 > \lambda_2 \geq \dots \geq \lambda_n$
Chaotic Attractor	$\lambda_1 > 0$ $\sum_{i=1}^n \lambda_i < 0$
Hyperchaotic Attractor	$\lambda_1 \geq \dots \geq \lambda_k > 0$ $\sum_{i=1}^n \lambda_i < 0$
Divergent	$\sum_{i=1}^n \lambda_i \geq 0$

2.5.4 Estimating LEs in Differentiable Systems

To measure the predictability of a known dynamical system with differentiable transition dynamics, the full spectrum of Lyapunov Exponents can be estimated by approximating Equation 2.16 for sufficiently large T :

$$\lambda_i \approx \frac{1}{T} \ln \left(\sigma_i \left(\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_0} \right) \right) \quad (2.20)$$

This expression captures the system's global predictability over the interval $[0, T]$ via the cumulative effect of its dynamics, with longer horizons yielding more accurate estimates of the Lyapunov Exponents. Additionally, in the case of a discrete time system with differentiable dynamics ($\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t)$), the total Jacobian is equivalent to the product of the individual Jacobians along the state trajectory:

$$\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_0} = \prod_{n=1}^T \frac{\partial \mathbf{s}_n}{\partial \mathbf{s}_{n-1}} = \frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_{T-1}} \frac{\partial \mathbf{s}_{T-1}}{\partial \mathbf{s}_{T-2}} \dots \frac{\partial \mathbf{s}_1}{\partial \mathbf{s}_0} \quad (2.21)$$

This method provides an exact theoretical computation of the spectrum of Lyapunov Exponents. However, directly using the product of Jacobians to determine these values introduces several numerical challenges. First, the product of Jacobians can become ill-conditioned, as all eigenvectors align with the dominant expansion direction. This causes smaller growth directions to be lost due to numerical precision limits. Furthermore, for chaotic systems, at least one Lyapunov Exponent is positive, and thus, the product of Jacobian matrices becomes increasingly ill-conditioned as $T \rightarrow \infty$. Despite these shortcomings, directly measuring the LEs using this method provides a very accurate estimation of the MLE.

2.5.5 Estimating LEs in Non-Differentiable Systems

To measure the predictability of a known dynamical system with non-differentiable transition dynamics, the full spectrum of Lyapunov Exponents can be estimated using the approach outlined by Benettin et al. [1980a,b]. In this method, the spectrum is represented by a set of $N \in \mathbb{N}$ perturbation vectors, each with length $0 < \epsilon \ll 1$, which are arranged in a column matrix:

$$\mathbf{V}_0 = [\mathbf{v}_0^1, \dots, \mathbf{v}_0^N] = \epsilon \times \mathbf{I}_N. \quad (2.22)$$

These vectors are then iteratively updated via Equation 2.23, thereby giving an approximation of local divergence in the tangent space.

$$\mathbf{V}_{t+1} = [\mathbf{f}(\mathbf{s}_t + \mathbf{v}_t^1) - \mathbf{f}(\mathbf{s}_t), \dots, \mathbf{f}(\mathbf{s}_t + \mathbf{v}_t^N) - \mathbf{f}(\mathbf{s}_t)] \quad (2.23)$$

However, over time, these perturbation vectors will tend to align with the direction of maximal growth. This occurs because components of the vectors along directions with smaller growth rates diminish exponentially relative to components along the most unstable direction. As a result, the ensemble of perturbation vectors becomes increasingly collinear, eventually capturing only the Maximal Lyapunov Exponent.

To counteract this collapse, the perturbation vectors are orthonormalised using the Gram-Schmidt process every $h \in \mathbb{N}$ steps. In practice, this is achieved by calculating the scaled QR decomposition of $\mathbf{V}_t/\epsilon$ (Equation 2.24) every h steps and replacing the tangent vectors $\mathbf{V}_t$ with the updated orthonormal basis $\epsilon \times \mathbf{Q}_t$. This ensures that each vector maintains a unique direction in the phase space.

$$\frac{\mathbf{V}_t}{\epsilon} = \mathbf{Q}_t \mathbf{R}_t \quad (2.24)$$

Furthermore, the diagonal entries of $\mathbf{R}_t$ represent the local expansion of the perturbation vectors in the updated orthogonal reference basis ($\mathbf{Q}_t$). As a result, these entries quantify the deformation of an infinitesimal volume element, with the first diagonal entry reflecting growth along the most unstable direction and the subsequent entries corresponding to growth in increasingly orthogonal subspaces. As such, they serve as local measures of divergence along linearly independent directions of the phase space and can therefore be used to determine the full spectrum of Lyapunov Exponents via Equation 2.25.

$$\lambda_i \approx \lim_{T \rightarrow \infty} \frac{1}{h \times T} \sum_{t=0}^{T/h} \ln(\mathbf{R}_{h \times t}^{(i,i)}) \quad (2.25)$$

Overall, this method provides a principled approach for identifying the predictability of dynamical systems with known but non-differentiable transition functions. By orthonormalising the perturbations before they converge to the direction of maximal growth, this method can accurately capture both positive and negative Lyapunov Exponents, thereby providing a reliable estimate of the MLE and SLE.

Chapter 3

Related Work

This chapter reviews the literature most relevant to the predictability of control systems. We focus on two complementary perspectives: classical control theory and reinforcement learning. These frameworks address similar control objectives but differ fundamentally in how they model system dynamics and reason about predictability.

Control theory provides a principled framework for designing controllers with predictable closed-loop behaviour. By exploiting structural assumptions about the system dynamics, it is often possible to derive controllers with explicit stability guarantees. However, these guarantees rely on increasingly restrictive assumptions as system complexity grows, limiting their applicability in real-world settings.

In contrast, reinforcement learning learns control policies directly from interaction with the system, avoiding the need for explicit modelling assumptions. This flexibility enables control in complex, high-dimensional environments where analytical models are unavailable. However, the lack of explicit structure introduces new challenges, particularly in ensuring the stability and predictability of the resulting control interaction.

The remainder of this chapter reviews how these two perspectives address the problem of predictable control under increasing system complexity. Section 3.1 examines how control theory establishes stability across linear, non-linear, chaotic, and uncertain dynamical regimes. Section 3.2 then reviews recent work in reinforcement learning, focusing on chaos in policy learning, methods for enforcing stability, and related challenges in safety and robustness. Together, these works highlight a fundamental trade-off between performance and predictability, which motivates the methods proposed in this thesis.

3.1 Control theory

Control theory provides a mathematical framework for constructing controllers that regulate the behaviour of dynamical systems. These methods typically exploit structural assumptions about the transition dynamics to design control policies with predictable closed-loop behaviour. In particular, when the system dynamics are known and well-structured, it is often possible to construct controllers with explicit stability guarantees.

However, the applicability of these methods depends critically on the nature of the underlying dynamics and the availability of an accurate system model. In systems with linear dynamics, stability can be characterised globally and verified analytically. As the dynamics become non-linear, these guarantees weaken and are often only local or dependent on specific structural properties. In systems with chaotic dynamics, stability becomes inherently trajectory-dependent and highly sensitive to modelling errors. Finally, when the transition dynamics are unknown or only partially observed, control methods must rely on additional assumptions, such as parameterised models or structured uncertainty.

This section reviews how control theory addresses these increasingly general settings. Across these regimes, a common trade-off emerges: strong stability guarantees require strong assumptions about the system dynamics, and relaxing these assumptions significantly complicates the design of reliable controllers.

3.1.1 Linear State Dynamics

We begin with control systems that assume known linear transition functions. In this setting, the state dynamics evolve according to the function $\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$ define the influence of the state and action on the system dynamics. While this assumption is restrictive in real-world control systems, it allows the closed-loop behaviour to be analysed analytically.

A widely used heuristic approach for stabilising such systems in practice is the Proportional Integral Derivative (PID) controller [Åström and Hägglund, 1995]. This method constructs a control signal based on the error between the current system state and a desired reference. Specifically, the control input is defined as a weighted combination of three terms: a proportional term that scales with the current error, an integral term that accumulates past errors, and a derivative term that anticipates future error based on the rate of change. Together, these components enable the controller to reduce steady-state error, damp oscillations, and improve transient response.

However, despite its simplicity and widespread use, PID control does not provide general guarantees of stability. The behaviour of the closed-loop system critically depends on the choice of control parameters, which are typically tuned empirically for a specific system. While stability can often be achieved in practice, it is not guaranteed analytically, and small changes in the system dynamics or operating conditions can lead to degraded or unstable behaviour.

In contrast, Linear Quadratic Regulator (LQR) [Anderson and Moore, 1990] provides a more principled framework for constructing stabilising controllers. This framework considers linear systems with a quadratic cost function that penalises both deviations from a target state and excessive control effort. The objective is to minimise the cumulative cost induced by a sequence of control inputs, thereby encouraging the system to remain close to the desired state while avoiding unnecessarily large actions.

Under these assumptions, the optimal control policy takes the form of a linear state-feedback controller ($\mathbf{a}_t = -\mathbf{K}\mathbf{s}_t$), where the control input at each time step is computed directly from the current system state. Intuitively, the controller applies corrective actions that are proportional to the deviation from the target state, with the proportionality determined by the system dynamics and the relative weighting of state error and control effort. This results in a feedback mechanism that continuously steers the system towards the desired equilibrium.

Applying this controller produces a closed-loop system whose behaviour is fully determined by the resulting system matrix ($\mathbf{A} - \mathbf{BK}$). Crucially, the parameters of the controller are chosen such that this matrix has stable eigenvalues, thereby ensuring that the state trajectory converges over time. As a result, LQR provides explicit guarantees of global asymptotic stability under its modelling assumptions, while simultaneously optimising the trade-off between performance and control effort.

However, these guarantees rely on several restrictive assumptions. In particular, the transition dynamics must be known exactly and must be well approximated by a linear model. In real-world control systems, these assumptions are often violated, as the dynamics are typically non-linear, uncertain, and only partially observed. This reliance on accurate linear models limits the applicability of classical linear control in complex systems. In contrast, reinforcement learning methods are appealing precisely because they can learn control policies directly from interaction without requiring such models.

3.1.2 Non-linear State Dynamics

We now consider control systems with known non-linear transition functions of the form $\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)$. While the objective remains to construct controllers that induce stable closed-loop dynamics, the presence of non-linearities makes both analysis and design significantly more challenging. Consequently, the synthesis of robust control laws must account for the intricate topological features of the state space to maintain performance under varying operating conditions.

A common approach for controlling non-linear systems is to approximate their dynamics locally. This is achieved by linearising the transition function around a nominal operating point, thereby producing a locally linear system to which standard linear control techniques (e.g., PID, LQR) can be applied. In this setting, the resulting controller inherits the stability properties of the linear approximation, ensuring convergence within a neighbourhood of the operating point. However, these guarantees are inherently local and rely on the assumption that the system remains close to the linearisation point. When the system operates outside this region, the approximation breaks down and stability is no longer guaranteed.

An alternative approach is to recover linear structure directly through feedback linearisation [Drazin, 1992]. Rather than approximating the dynamics, this method constructs a control input that cancels the non-linear components of the dynamics, thereby transforming the system into an equivalent linear form. When this transformation is valid, linear control techniques can be applied to the transformed system, and stability guarantees comparable to linear systems can be obtained. However, this approach requires strong structural assumptions, including precise knowledge of the system dynamics and the ability to invert components of the transition function. As a result, feedback linearisation is only applicable to a restricted class of control systems.

In contrast to these approaches, Model Predictive Control (MPC) [Rawlings et al., 2017] provides a general optimisation-based framework for controlling non-linear systems. At each time step, MPC selects a sequence of control inputs that optimises a finite-horizon objective, applying only the first action before re-solving the optimisation problem at the next step. By repeatedly re-evaluating the optimal action sequence, MPC can account for non-linear dynamics and adapt to changes in the system state. However, while MPC can produce stable behaviour in practice, it does not inherently provide general guarantees of closed-loop stability. Stability must be enforced through additional constraints or terminal cost design, which typically rely on detailed knowledge of the system dynamics. Furthermore, solving an optimisation problem at every time step introduces significant computational overhead, limiting its applicability in systems with high control frequencies.

More generally, the Lyapunov function [Lyapunov, 1992] provides a framework for analysing and certifying stability in non-linear systems. This approach constructs a positive semi-definite scalar function ($V : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$) that decreases along trajectories of the closed-loop system ($V(\mathbf{s}_{t+1}) < V(\mathbf{s}_t)$). If such a function can be found, it provides a formal certificate of stability, even in the presence of non-linear dynamics. However, while Lyapunov methods offer strong theoretical guarantees, identifying suitable Lyapunov functions is challenging and typically requires detailed knowledge of the system dynamics, thereby limiting their applicability in complex settings.

Overall, these methods demonstrate that stable control is achievable in non-linear systems, but the resulting guarantees are significantly weaker than in the linear case. Stability is often local, depends on restrictive structural assumptions, or requires explicit knowledge of the transition dynamics. Furthermore, as system complexity increases, these guarantees become increasingly difficult to establish, even when a model of the system is available. Reinforcement learning methods address this limitation by learning effective control policies without requiring an explicit model of the system dynamics.

3.1.3 Chaotic State Dynamics

In systems with chaotic dynamics, the limitations of non-linear control become even more apparent. Although these systems evolve deterministically, they exhibit sensitivity to initial conditions, meaning that arbitrarily small changes in the system state can lead to exponentially diverging state trajectories. In this setting, local measures of stability become insufficient for characterising long-term behaviour, as trajectories diverge within arbitrarily small neighbourhoods. As a result, long-term behaviour becomes effectively unpredictable, and stability can no longer be understood purely in terms of convergence to a fixed point or trajectory. Instead, stability is more naturally characterised by the spectrum of Lyapunov Exponents.

One approach for controlling chaotic systems is the Ott–Grebogi–Yorke (OGY) method [Ott et al., 1990]. Rather than attempting to eliminate the chaotic dynamics, this approach aims to stabilise unstable periodic trajectories embedded within the chaotic attractor. This is achieved by applying small actions to counteract the unstable dynamics when the system is near an unstable periodic trajectory. As a result, the system can be steered towards a periodic trajectory despite its underlying chaotic behaviour.

While this method does provide a principled approach for controlling systems with chaotic dynamics, control is only applied when the system state lies within a small neighbourhood of the target trajectory. Outside this region, the system evolves according to its natural dynamics. Consequently, convergence may be extremely slow, as corrective behaviour is only applied when the system state re-enters a small neighbourhood of the target trajectory.

To address the limited intervention of OGY, [Pyragas, 1992] proposed Delayed Feedback Control (DFC). Instead of applying a corrective action when the system is close to the target trajectory, this method attempts to stabilise a periodic trajectory by applying an action at every state. Specifically, to stabilise the system to a periodic trajectory with period T , this method applies a carefully designed linear transformation ($\mathbf{K} \in \mathbb{R}^{M,N}$) to the difference between the current state and the state T steps previously ($\mathbf{a}_t = \mathbf{K}(\mathbf{s}_t - \mathbf{s}_{t-T})$). In effect, the method constructs a signal that vanishes exactly when the system lies on the desired periodic trajectory, while suppressing deviations from it. As a result, the control is applied continuously rather than intermittently, and does not require explicit identification of the unstable trajectory.

Despite providing principled mechanisms for stabilising chaotic systems, these approaches share several important limitations. Specifically, both methods rely on the existence of a desirable periodic trajectory embedded within the system dynamics, and implicitly assume that stabilisation can be achieved by regulating deviations relative to this trajectory. As a result, they are inherently tied to local structure in the state space, thereby restricting their applicability in settings where the desired behaviour is not well-described by a fixed periodic trajectory. Motivated by this research gap, Chapters 5 and 6 propose data-driven methods for learning globally stabilising control policies in chaotic control systems without requiring explicit knowledge of the transition dynamics.

3.1.4 Unknown State Dynamics

The control methods discussed thus far assume that the transition dynamics are known and can be exploited directly when constructing the controller. In many real-world systems, however, this assumption does not hold as the system dynamics may be partially known, difficult to model accurately, or subject to changes over time. In this setting, the objective is to design controllers that achieve stable behaviour without relying on an exact model of the transition function. Importantly, these approaches are not mutually exclusive with the methods introduced in the previous sections. Instead, they are often combined with linear, non-linear, or chaotic control techniques to compensate for uncertainty in the system dynamics.

A common approach is adaptive control [Åström and Wittenmark, 1989], where the controller updates its internal parameters online in response to observed system behaviour. Rather than relying on a fixed model, the controller estimates unknown components of the dynamics and adjusts the control policy accordingly. This allows the system to compensate for modelling errors and adapt to changes in the environment. However, while adaptive control can maintain stability under certain conditions, these guarantees typically rely on assumptions such as slow variation in the dynamics or the existence of a suitable parameterisation. In practice, these assumptions are often difficult to verify, particularly in complex high-dimensional systems.

Another approach is robust control based on structured uncertainty [Zhou et al., 1996], where the controller is designed to operate over a class of possible models rather than a single known system. In this setting, the dynamics are not known exactly but are assumed to lie within a predefined set of possible models. The controller is then constructed to achieve acceptable performance across this entire set, often by considering worst-case scenarios. While this approach reduces reliance on precise model knowledge, it requires the uncertainty to be specified a priori and typically leads to conservative behaviour, as the controller must account for the most adverse realisations within the model class.

More generally, learning-based control methods [Ljung, 1999] infer system dynamics directly from data and incorporate these models into the control process. In practice, this is often realised through a two-stage pipeline in which a dynamics model is first learned and then used within a model-based controller, such as model predictive control. This enables control in systems where analytical models are unavailable or difficult to derive. However, the effectiveness of this approach depends critically on the accuracy of the learned model. Even small approximation errors can compound over time, thereby leading to incorrect state predictions and degraded control behaviour. As a result, stability and performance guarantees are difficult to obtain, particularly in high-dimensional or highly non-linear systems.

Overall, these approaches demonstrate that the assumption of known dynamics can be relaxed, but only under additional structural constraints. Adaptive methods require suitable parameterisations, robust approaches rely on predefined uncertainty sets, and learning-based methods depend on the accuracy of inferred models. As system complexity increases, these requirements become increasingly difficult to satisfy, thereby limiting the applicability of classical control techniques in real-world settings. In contrast, reinforcement learning removes these assumptions by learning control policies directly from interaction with the system. However, this shift introduces new challenges, as stability is no longer explicitly enforced. Motivated by this research gap, Chapter 4 investigates whether reinforcement learning policies naturally induce predictable or chaotic state dynamics when controlling non-linear control systems. Building on this, Chapters 5 and 6 propose methods for improving the predictability of the RL control interaction.

3.2 Reinforcement Learning

While the previous section reviewed how classical control theory establishes stability under increasingly weak assumptions about the transition dynamics, this section reviews how related challenges have been addressed within reinforcement learning. In particular, we focus on work that studies chaos in policy learning, methods that incorporate stability guarantees into the RL objective, and approaches that address safety and robustness.

3.2.1 Chaos in Reinforcement Learning

The influence of chaotic dynamics in reinforcement learning was initially explored within the context of model-based approaches. In this setting, policy gradients are obtained by predicting the long-term state dynamics using a one-step dynamics model and assessing how changes in the policy impact the predicted returns. However, Parmas et al. [2018] demonstrated that this learning process can be highly unstable. Specifically, they showed that over long prediction horizons, the variance in policy gradients grows uncontrollably, thereby causing the gradients to be effectively random. They argue that this instability occurs because the learned non-linear dynamics model acts like a chaotic system, and the sensitive nature of this system renders long-term policy gradients ineffective.

More broadly, Metz et al. [2021] demonstrated that this instability is not just an artefact of the learned dynamics model but also occurs when the transition function is explicitly known. Specifically, they proved that the analytic policy gradient explodes when the dominant eigenvalue of the Jacobian matrix of the transition function is greater than one. This occurs because the Jacobian over T steps $\left(\frac{\partial s_T}{\partial s_0}\right)$ grows exponentially as $T \rightarrow \infty$. In effect, chaotic control systems with positive MLE have unstable policy gradients over long horizons.

Finally, Wang et al. [2023] provided a theoretical explanation for policy gradient instability by studying the regularity of the value function in deterministic control systems. They showed that even when the policy, reward function and transition function are Lipschitz continuous, the value function is only Lipschitz continuous when the system’s MLE satisfies $\lambda_1 < -\ln(\gamma)$. When this condition is violated, the value function and return become α -Hölder continuous, with $\alpha = -\ln(\gamma)/\lambda_1$. This result implies that both the value function and return exhibit fractal-like structures, meaning that small perturbations in the state or policy parameters can cause disproportionately large fluctuations in performance.

Overall, these findings demonstrate that chaotic state dynamics can have an adverse effect on policy learning. They show that chaotic behaviour in the environment influences the smoothness of the objective function, and this in turn leads to unreliable policy updates. However, while prior work draws on chaos theory to motivate this instability, it does not explicitly demonstrate that the underlying system is chaotic. Instead, chaos is used as a theoretical tool to explain the instability observed in policy gradients.

Motivated by this research gap, Chapter 4 empirically determines if the interaction between trained state-of-the-art deep RL policies and continuous control environments induces chaotic state dynamics. Chapters 5 and 6 then propose methods for minimising the Maximal Lyapunov Exponent, thereby increasing the Hölder exponent and improving the smoothness of the objective function.

3.2.2 Stable Reinforcement Learning

Recent work has investigated how reinforcement learning algorithms can be augmented with control-theoretic structure to enforce closed-loop stability, a critical requirement in real-world control tasks. This ensures that the system’s trajectories remain bounded and converge to a desired set point or trajectory. Classical non-linear control typically guarantees such behaviour via Lyapunov functions: a scalar energy-like function is constructed to prove that the system’s state asymptotically converges to an equilibrium. This framework has recently been extended to the RL setting by embedding stability constraints directly into the learning objective.

The first deep RL algorithm to incorporate Lyapunov stability theory was the Lyapunov Actor-Critic [Han et al., 2020]. This method trains a neural network to approximate a Lyapunov function, which is then incorporated as a critic to guide policy learning. In doing so, it aims to certify the stability of the control interaction by ensuring that the Lyapunov function decreases along state trajectories. While this is presented as a formal stability guarantee, the policy is trained to stabilise the task-specific reward, which typically depends on a low-dimensional projection of the state (e.g., torso velocity in the *Cheetah Run* environment). As a result, the learned Lyapunov function reflects stability in terms of reward-aligned features, rather than guaranteeing convergence across the full system state.

Furthermore, Chang and Gao [2021] proposes an extension to Lyapunov Actor-Critic that learns an almost Lyapunov function independently of the environment’s reward and then incorporates it into the policy update by augmenting the advantage estimate. In doing so, it penalises actions that violate the Lyapunov function, thereby encouraging stabilising behaviour. However, while this is presented as an environment-agnostic formal stability guarantee, it requires a predefined target state, which is often defined over a reduced number of axes (e.g., stabilising joint angles to hold upright positions in the *Walker Walk* environment).

Overall, these approaches represent important steps towards integrating formal control-theoretic stability guarantees into the RL training loop. However, they typically enforce stability over a restricted subset of the system state, often aligned with manually specified task-relevant states. This poses two issues for the general deployment of stable reinforcement learning methods. First, during learning, the agent requires environment-specific information (i.e., the target state) to provide stability certificates. This restricts the application of these agents, as stable target positions are often unknown prior to training. Second, when controlling high-dimensional control systems (e.g., *Walker*, *Cheetah*), the target position is set to a subset of the state. As a result, instability can still arise in the unconstrained dimensions. Motivated by this research gap, Chapters 5 and 6 propose methods for stabilising the full system dynamics without relying on environment-specific target states or reward shaping during training.

3.2.3 Safe Reinforcement Learning

Traditional reinforcement learning algorithms are typically designed to maximise cumulative reward over an episode. However, this unconstrained optimisation process often ignores the physical or operational limitations of the environment. Consequently, when deployed in real-world scenarios, a trained agent may readily navigate into hazardous regions of the state space or execute catastrophic control actions. To address this test-time vulnerability, safe reinforcement learning methods explicitly introduce safety constraints into the control objective. This ensures that the deployed policy maintains the control system within a predefined safe set of states without compromising task performance.

This safety objective is conventionally formalised using the Constrained Markov Decision Process [Altman, 1999]. To enforce hard constraints on the agent’s behaviour, the Constrained MDP extends the standard MDP by introducing an auxiliary cost function $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and a predefined cost limit $d \in \mathbb{R}$. In this setting, the objective (Equation 3.1) is to find a policy which maximises the expected return while strictly bounding the expected cumulative cost.

$$\max_{\pi} J(\pi) \quad \text{subject to} \quad \mathbb{E}_{s_0 \sim \rho_0} \left[\sum_{t=0}^{\infty} c(s_t, a_t) \mid s_{t+1} = \mathbf{f}(s_t, a_t), a_t = \pi(s_t) \right] \leq d \quad (3.1)$$

One approach to solving the Constrained MDP is to incorporate safety constraints directly into the deep RL policy optimisation process. Specifically, Constrained Policy Optimisation (CPO) [Achiam et al., 2017] extends Trust Region Policy Optimization [Schulman et al., 2015] by solving a constrained policy update that jointly enforces a trust-region constraint and a constraint on expected cost. This yields near-monotonic improvement in reward while approximately satisfying the safety constraint at each update. However, while CPO successfully reduces unsafe behaviours, this approach does not eliminate them entirely, as the agent can still occasionally violate safety constraints.

Subsequently, Cheng et al. [2019] provided stricter safety guarantees by integrating control-theoretic constraints into the deep RL loop. Specifically, they demonstrated that Control Barrier Functions can be used to enforce forward invariance of a predefined safe set by modifying the agent’s actions at execution time. In practice, this is achieved by mapping potentially unsafe control inputs to the closest action that satisfies the barrier constraints. This mechanism can be implemented as a differentiable projection layer within the RL architecture, allowing gradients to propagate through the safety filter and enabling the policy to learn to avoid the boundary of the safe set. However, this approach relies heavily on environment-specific prior knowledge, as it typically requires a partially known dynamics model or a manually defined safe set to construct the barrier function. This restricts their applicability in high-dimensional systems where the transition dynamics are complex or entirely unknown prior to training.

Overall, Safe RL methods provide essential mechanisms for restricting an agent’s behaviour to safe regions of the state space. However, constraining an agent to a predefined safe zone does not inherently improve the predictability of the control interaction. An agent can operate entirely within safety constraints while still exhibiting highly chaotic behaviour. Consequently, to achieve reliable real-world deployment, controllers must respect task-specific safety boundaries and ensure the predictability of the underlying dynamics.

3.2.4 Robust Reinforcement Learning

A related challenge for the deployment of reinforcement learning in real-world control systems is robustness to small perturbations in the system observations. In practice, the observations received by an agent are often corrupted by sensor noise, modelling errors, or environmental variability. For a control policy to be reliable, it must produce consistent and predictable outcomes under such perturbations.

Recent work has shown that deep reinforcement learning policies can be highly sensitive to small changes in their inputs. Specifically, Huang et al. [2017] demonstrated that applying small, imperceptible perturbations to the agent’s observations can significantly degrade overall performance. These perturbations are computed using a modification to the Fast Gradient Sign Method [Goodfellow et al., 2015], which leverages both the policy and value function to shift the observed state in a direction that induces suboptimal actions. Importantly, the magnitude of the perturbation required to cause this degradation is often extremely small, highlighting a fundamental sensitivity of deep RL policies to their inputs.

Subsequent work has shown that such vulnerabilities are not limited to adversarial settings. Kos and Song [2017] demonstrated that injecting small amounts of random noise at infrequent intervals is sufficient to disrupt policy performance. Similarly, Lin et al. [2017] showed that selectively applying perturbations at critical decision points can degrade performance while minimising the frequency of intervention. These results suggest that RL policies can exhibit highly non-smooth input-output mappings, where small changes in the observed state lead to disproportionately large changes in the selected action.

Overall, these findings demonstrate that even small perturbations to the system observations can significantly degrade the performance of deep RL policies. Crucially, this sensitivity implies that minor deviations in the observed state can propagate through the control policy and lead to substantial changes in the resulting state trajectories. Chapter 4 builds on this finding by analysing the effect of a single small perturbation on the long-term system dynamics, thereby providing insight into the mechanisms underlying this sensitivity.

Chapter 4

Identifying Chaos

Deep reinforcement learning (RL) offers a powerful framework for real-time control of complex autonomous systems, and its success in simulated domains has led to growing interest in applying RL to real-world control tasks. However, during the learning process, state-of-the-art deep RL controllers focus primarily on maximising performance and rarely consider the predictability of the resulting control interaction.

In this chapter, we argue that these deep RL policies are unsuitable for real-world deployment, as they can induce chaotic state dynamics [Lorenz, 1963, Devaney, 1992], even when the environment is fully deterministic. Consequently, a single small change in the system’s state can have a dramatic impact on long-term state trajectories, as shown in Figure 4.1. This instability poses two problems for the safe deployment of deep RL in real-world environments:

1. **Unstable learning:** Chaotic dynamics disrupt policy learning by inducing sharp irregularities in the value function and return [Metz et al., 2021, Wang et al., 2023]. This instability leads to unreliable policy gradients and poor generalisation, as small perturbations in state or policy can cause large fluctuations in performance.
2. **Unpredictable behaviour:** It is impossible to accurately predict the long-term behaviour of chaotic control systems which rely on noisy, partially observable sensors to attain an observation. This unpredictability means that safe and reliable behaviour cannot be guaranteed for real-world systems.

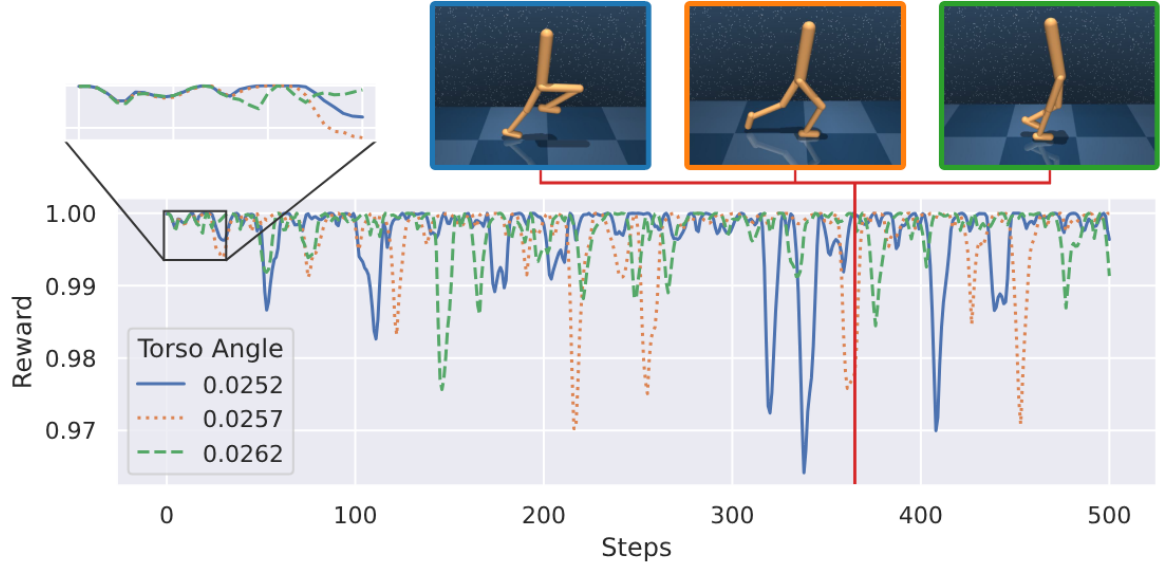


Figure 4.1: Reward attained when a trained deterministic Soft Actor-Critic agent controls the deterministic *Walker Walk* environment. Each system has the same initial configuration, other than the torso angle, which is perturbed by $\pm 5 \times 10^{-4}$ degrees. This small perturbation causes the systems to significantly diverge after 50 steps due to the chaotic nature of the control interaction, causing significant variation in the total reward attained.

These challenges highlight the need for a more principled understanding of how deep RL policies interact with the underlying dynamics of the systems they control. If the policies systematically induce chaotic behaviour, then performance alone is an insufficient criterion for their evaluation. This motivates a formal investigation into whether the dynamics induced by deep RL policies satisfy established definitions of chaos from dynamical systems theory.

To this end, the goal of this chapter is to determine whether the definitions of chaos outlined by Devaney [1992] and Lyapunov [1992] can be used to identify the predictability of the deterministic control interaction, and if optimising for reward alone induces chaotic dynamics in continuous control domains. To address this, we investigate if the deterministic control interaction forms an attractor in Section 4.2, quantify if they are sensitive to initial conditions in Section 4.3, determine if they display topological transitivity in Section 4.4, and identify whether they contain periodic trajectories in Section 4.5. By evaluating these properties, we can determine whether state-of-the-art deep RL policies induce predictable, diverging, or chaotic dynamics, as defined by Lyapunov and Devaney.

Furthermore, Section 4.6 investigates the impact of a chaotic control system on the predictability of the reward attained, while Section 4.7 then examines how model training influences this predictability. Through this analysis, we determine if state-of-the-art deep RL policies induce predictable or chaotic state dynamics in a simplified deterministic setting. We argue that the presence of chaotic state dynamics renders these policies unsuitable for real-world deployment, as sensor noise and environmental disturbances would further amplify this unpredictability.

4.1 Preliminaries

This section outlines the preliminaries required for analysing the predictability of deep RL policies in continuous control tasks. We first describe the model’s training regime, followed by a reformulation of the POMDP as a dynamical system. This enables the resulting closed-loop interaction to be studied through the lens of chaos theory and Lyapunov Exponents.

4.1.1 Model Training

To closely match real-world applications, we estimate the predictability of tasks from the DeepMind Control Suite [Tunyasuvunakool et al., 2020] when controlled by deep RL policies. These simulated environments provide a range of deterministic control problems with continuous state spaces. For each control task, we independently train three instances of Soft Actor-Critic (SAC) [Haarnoja et al., 2018], Twin Delayed Deep Deterministic Policy Gradients (TD3) [Fujimoto et al., 2018], and Dreamer V3 [Hafner et al., 2025], as these represent state-of-the-art model-free and model-based methods for continuous control tasks. Furthermore, to determine if trained deep RL policies directly influence the predictability of each control system, we also introduce a passive controller that takes no actions (No-action). All models are based on the Stable Baselines 3 [Raffin et al., 2021] implementation with the parameters outlined in Appendix A.1 and trained using an Intel Core i7-8700 CPU workstation with an Nvidia RTX 2080 Ti GPU and 32GB of RAM.

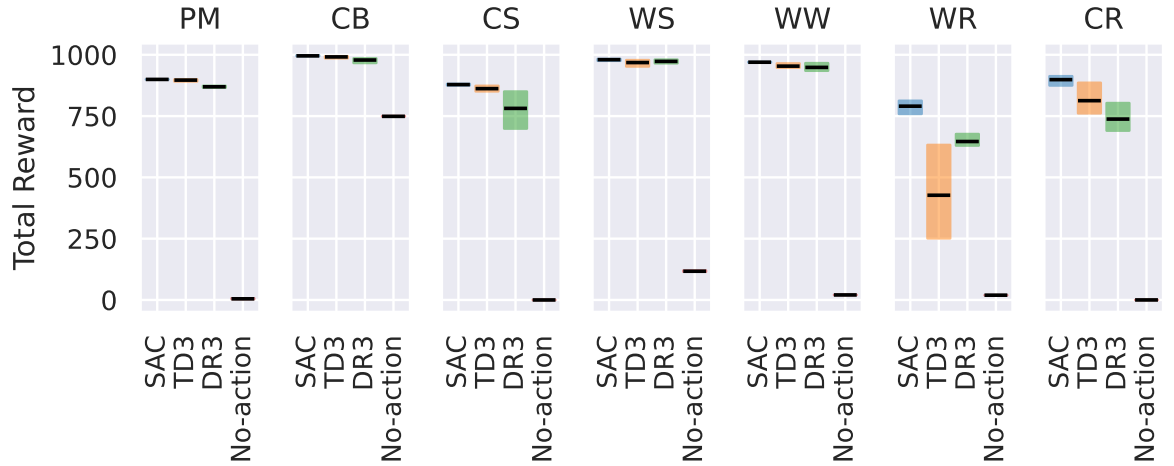


Figure 4.2: Total episode reward for the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments when controlled by trained instances of SAC, TD3, Dreamer V3 (DR3), and an agent that takes no actions (No-action). Each policy–environment combination is independently trained with three random seeds, and the average interquartile episode reward with a bootstrapped 95% confidence interval is reported over 80 evaluation episodes, each with a fixed length of 1000 steps.

The average interquartile reward and bootstrapped 95% confidence interval [Agarwal et al., 2021] for each policy type are reported in Figure 4.2. These results show that all three algorithms learn policies which performed well and significantly better than the No-action baseline. However, this does not speak to the types of dynamics produced or the predictability of the control interaction.

4.1.2 POMDP as a Dynamical System

To analyse the closed-loop dynamics induced by learned policies, we model the control interaction as a discrete-time dynamical system. Specifically, we define the phase space of the dynamical system as the environment’s state space, and use the composition of the deterministic policy (π_θ), deterministic observation function ($\mathbf{o}$) and the deterministic environment’s transition function ($\mathbf{f}$) as the update function:

$$\mathbf{s}_{t+1} \doteq \mathbf{f}(\mathbf{s}_t, \pi_\theta(\mathbf{o}(\mathbf{s}_t))) \quad (4.1)$$

This composition yields a parameterised update function that governs the system’s evolution over a single step of the POMDP. Since the policy, observation and the environment’s transition function are deterministic, the resulting dynamical system is deterministic. Consequently, the system will always produce the same state trajectories when starting from a given initial state. However, unlike conventional dynamical systems, this function depends explicitly on the policy parameters θ , with changes in θ inducing differing state dynamics. Framing the POMDP as a dynamical system in this way enables a principled analysis of how the environment behaves under the influence of learned control policies.

4.2 System Convergence

In this section, we determine whether the state trajectories produced by trained deep RL controllers converge to a fixed point, diverge indefinitely, or converge to an attractor with positive K -dimensional volume (where $K \in \mathbb{N}$ is less than or equal to the dimension of the state space N). The presence of an attractor provides an indication of the long-term dynamics produced by the policy–environment interaction. Point attractors indicate that the policy guides the system to a singular point in the state space and can maintain the system at this position. In contrast, diverging dynamics suggest that the policy fails to maintain control, leading to unbounded trajectories. Finally, convergence to an attractor with positive volume suggests that persistent dynamics occur in a specific region of the state space. The existence of a positive volume attractor is a prerequisite for Devaney and Lyapunov chaos, as these require diverging dynamics within a bounded chaotic attractor.

To determine if the interaction between deep RL policies and continuous control environments forms an attractor in the state space, we sample 80 states randomly from the initial state distribution of each environment and extend each of these for 1 POMDP episode (1000 steps). This process generates a set of 80 state trajectories, which can be used to assess the convergence properties of the control interaction. Increasing the number of initial states will provide a better coverage of the state space, albeit at the expense of computation time. To this end, we use 80 initial states as this provides good coverage of the initial state distribution without introducing significant computational overhead.

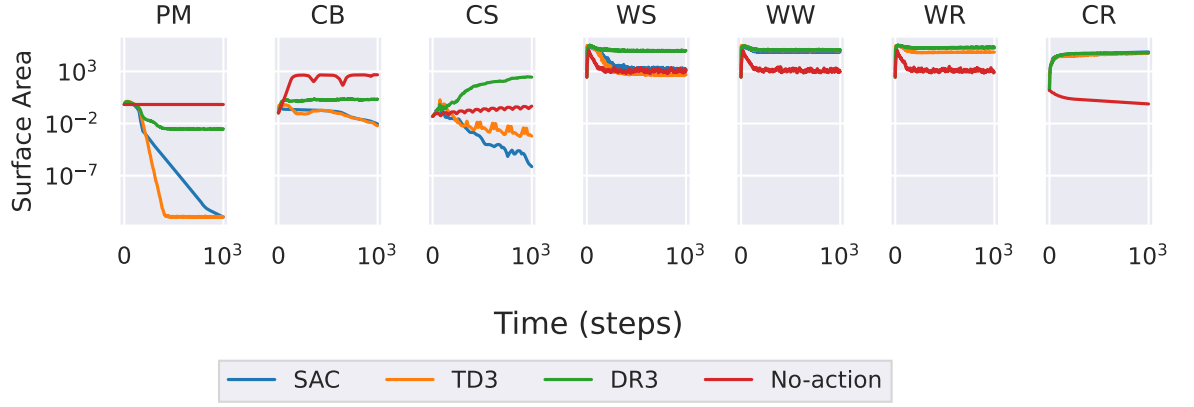


Figure 4.3: Surface area of the attractor produced when SAC, TD3, Dreamer V3 (DR3), and the no action baseline (No-action) control the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments.

With this set of example episodes, we then measure the surface area of the minimum hyperrectangle that contains all the states at each time step. This metric can identify the type of attractor, as convergence to zero indicates a point attractor, unbounded values indicate diverging dynamics, and convergence to a positive value indicates an attractor with positive K -dimensional volume. Additionally, by measuring the surface area we can identify the convergence properties of the state space even if one dimension fully converges.

Figure 4.3 provides the average and 95% confidence interval for the minimum surface area of the hyperrectangle for each policy–environment interaction. From this, we can establish that when the model-free methods (SAC, TD3) control low-dimensional systems (*Pointmass*, *Cartpole*), the resulting interaction produces a point attractor as the surface area decays to zero. Furthermore, inspecting the time-series projection plot for the *Pointmass* environment (Figure 4.4), we find that the attractor is formed at $\mathbf{s} = (0, 0, 0, 0)$. These findings are consistent with the definition of the reward function for this task (as well as the *Cartpole* tasks), as high reward is given when the policy maintains the system in a fixed location.

In contrast, when Dreamer V3 controls the low-dimensional systems or any deep RL policy controls the high-dimensional systems (*Walker*, *Cheetah*), the surface area of the hyperrectangle plateaus at a non-zero value. This indicates that the state trajectories do not converge to a single point but diverge while remaining in a bounded region of the state space. However, this does not indicate if the system orbits within this attractor or if the state trajectories converge to a series of point attractors.

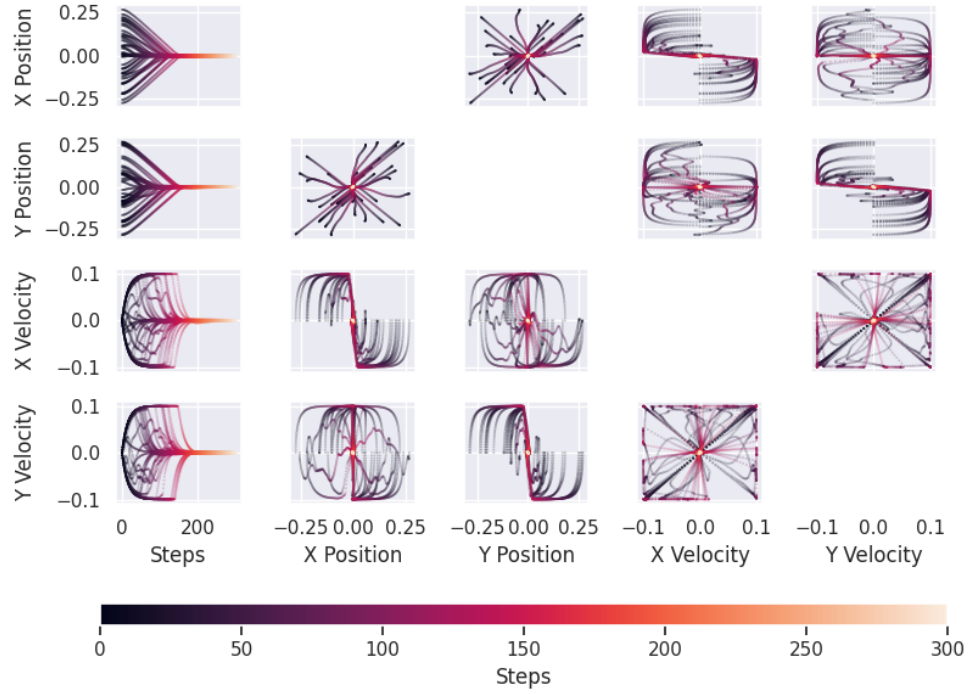


Figure 4.4: Axial hyperplane projections of 80 trajectories produced when a policy trained using SAC controls the *Pointmass* environment.

To clarify this, we examine the orthogonal projection of the *Walker Stand* state trajectories onto axial hyperplanes (Figure 4.5). These projections indicate that the system converges to a set of point attractors with high torso height and low torso angle, as these provide high rewards. However, while convergence occurs in the task-relevant dimensions, the remaining state variables do not exhibit the same behaviour. This suggests that the system displays partial convergence but not global convergence.

Finally, inspecting the orthogonal projection of the *Walker Walk* state trajectories onto axial hyperplanes (Figure 4.6), we find the attractor is moving as all the state axes are bound except for the horizontal position, which monotonically increases. Consequently, state trajectories in these systems will converge to and orbit within a consistent region of the state space, provided they have the same initial horizontal position. This finding is consistent with the definition of the reward function for this task (as well as the *Walker Run* and *Cheetah Run* tasks), as high reward is given when the policy maintains the system at a fixed positive horizontal velocity.

Overall, these results indicate that trained deep RL policies induce point attractors, partial point attractors and positive volume attractors in continuous control domains. Furthermore, these attractors are consistent with the reward function for each environment as they emphasise convergence to a fixed state or a subset of state dimensions.

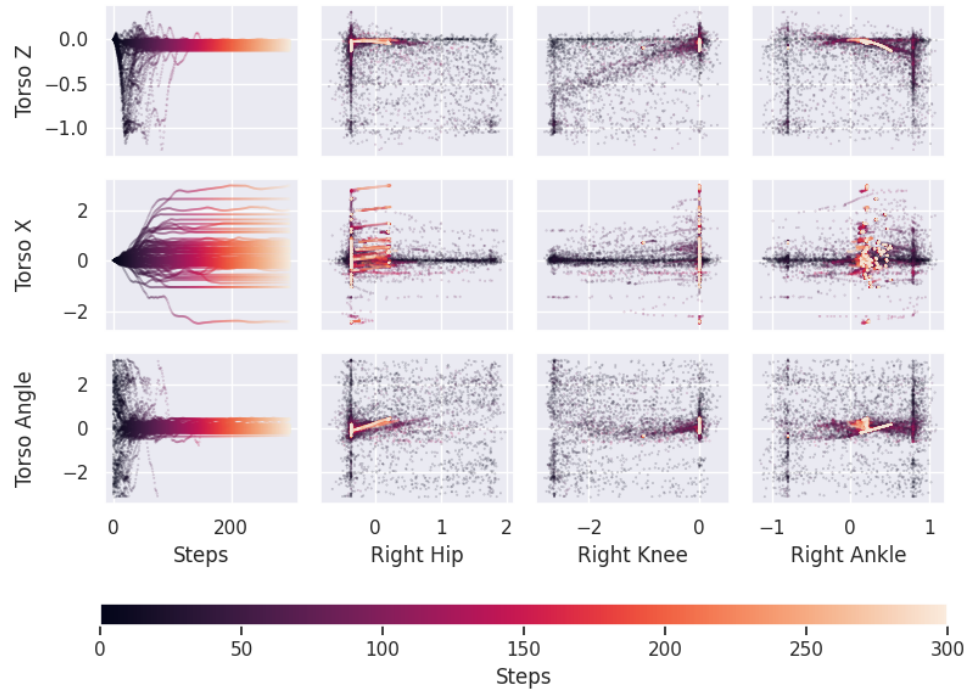


Figure 4.5: Axial hyperplane projections of 80 trajectories produced when a policy trained using SAC controls the *Walker Stand* environment.

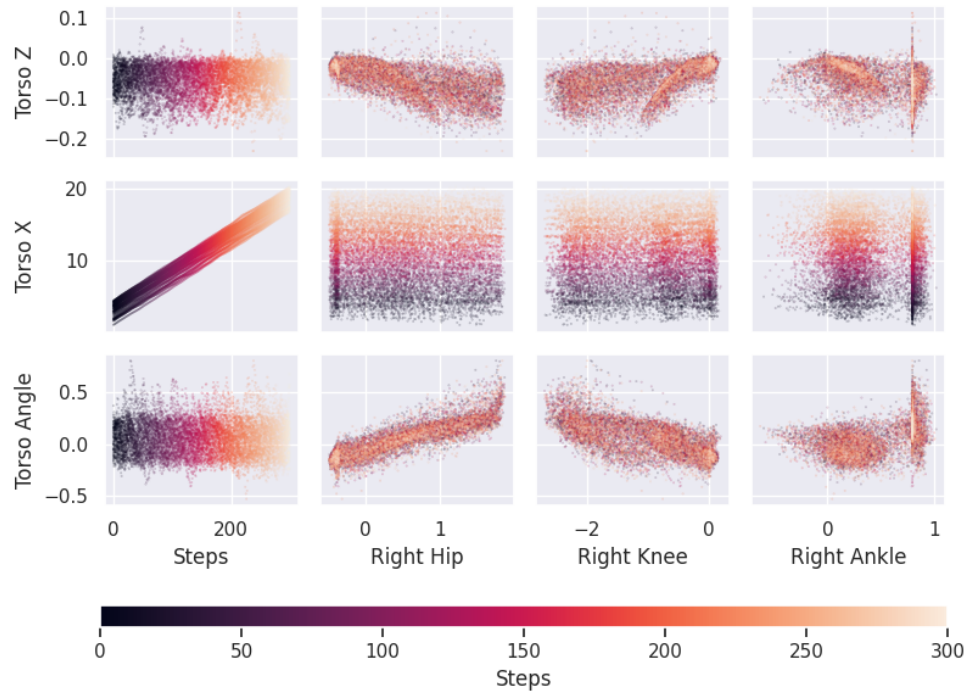


Figure 4.6: Axial hyperplane projections of 80 trajectories produced when a policy trained using SAC controls the *Walker Walk* environment.

4.3 Sensitivity to Initial Conditions

In this section, we use the spectrum of Lyapunov Exponents to empirically measure how sensitive deep reinforcement learning policies are to small changes to the initial system state. These values provide a direct method for quantifying the predictability characteristic of the control system, and the largest of these values can be used to identify high sensitivity to initial conditions, a criterion necessary for Devaney chaos.

4.3.1 Estimating Lyapunov Exponents

To identify the predictability of each policy–environment interaction, we estimate the full spectrum of Lyapunov Exponents using the method proposed by Benettin et al. [1980a,b]. We evaluate this spectrum over 1000 steps (1 POMDP episode), starting from states sampled from the initial distribution for each environment. A perturbation vector is initialised for each state dimension at a distance of 10^{-4} from the sample state. This represents an arbitrarily small change to the control system without introducing numerical precision errors. These perturbation vectors are orthonormalised every 10 steps so that they maintain a linear approximation of the tangent space (smaller than 10^{-1} units) and don't converge to the direction of maximal growth. Furthermore, given these perturbation vectors grow exponentially (Equation 2.18), this normalisation frequency directly imposes an upper bound on the largest viable MLE our method can detect. Specifically, Equation 4.2 shows that the linearity assumption breaks when the MLE is larger than 0.69, as the expected divergence after the normalisation period (10 steps) is greater than 10^{-1} . Consequently, this method is invalid if $\lambda_1 > 0.69$.

$$10^{-4} \times e^{10 \times \lambda_1} < 10^{-1} \implies \lambda_1 < \frac{3 \times \ln(10)}{10} \approx 0.69 \quad (4.2)$$

This process is then repeated with 20 states randomly sampled from the initial state distribution, and the average value for each exponent is used to calculate the MLE and SLE. Increasing the number of initial states will provide a more accurate measure of the MLE and SLE. However, as this is a computationally expensive process, we limit this to 20 initial states.

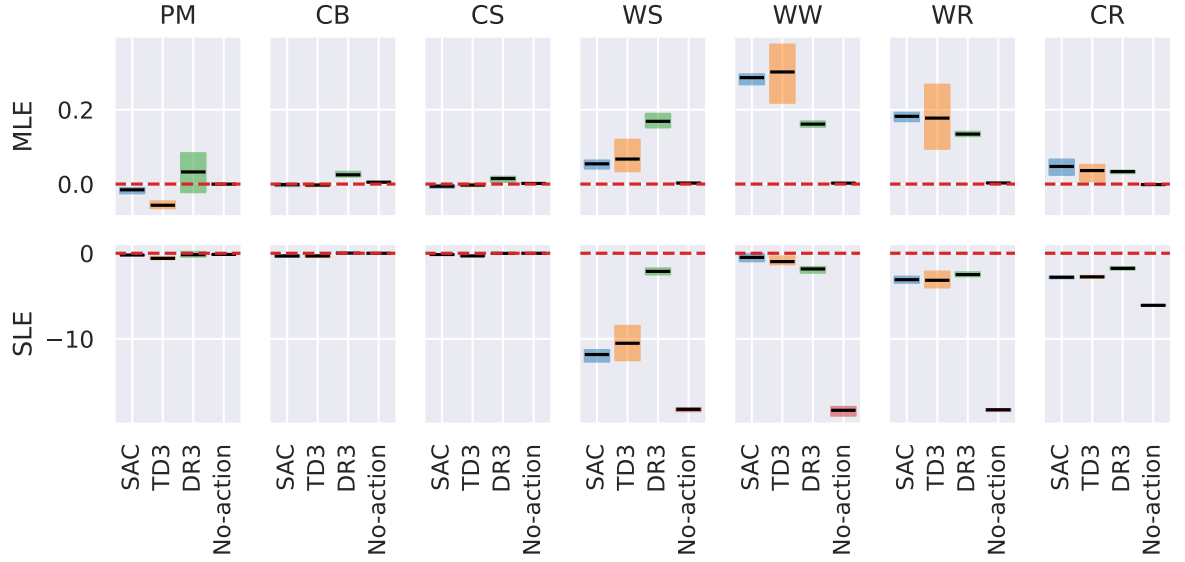


Figure 4.7: Estimated Maximal Lyapunov Exponent (MLE) and Sum of Lyapunov Exponents (SLE) for the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments when controlled by a trained instance of SAC, TD3, Dreamer V3 (DR3), and an agent that takes no actions (No-action). Each policy–environment combination is independently trained with three random seeds, and the interquartile average MLE and SLE for each seed is calculated using 20 initial states. A bootstrapped 95% confidence interval is included to show the variation in MLE and SLE across random seeds.

Figure 4.7 provides the bootstrapped 95% confidence interval for the MLE and SLE produced by each policy–environment pair. In all cases, the empirical MLE remains well below the theoretical upper bound ($\lambda_1 = 0.69$), indicating that the chosen orthonormalisation frequency and perturbation size are valid.

Furthermore, these results indicate that all the environments are naturally invariant to small perturbations, as the No-action baseline has $\lambda_1 \approx 0$. Conversely, when these environments are controlled by deep RL policies, λ_1 can be non-zero, indicating that deep neural network controllers directly influence the predictability of these control systems.

We find that SAC and TD3 produce predictable state dynamics in simple low-dimensional environments (*Pointmass*, *Cartpole Balance*, *Cartpole Swingup*) as they have negative MLE. Therefore, if a small perturbation is applied to any of the states in these environments, the subsequent state trajectories converge. This result is consistent with Section 4.2 and the definition of the reward function for each of these tasks, as they provide high rewards for maintaining the system at a fixed location.

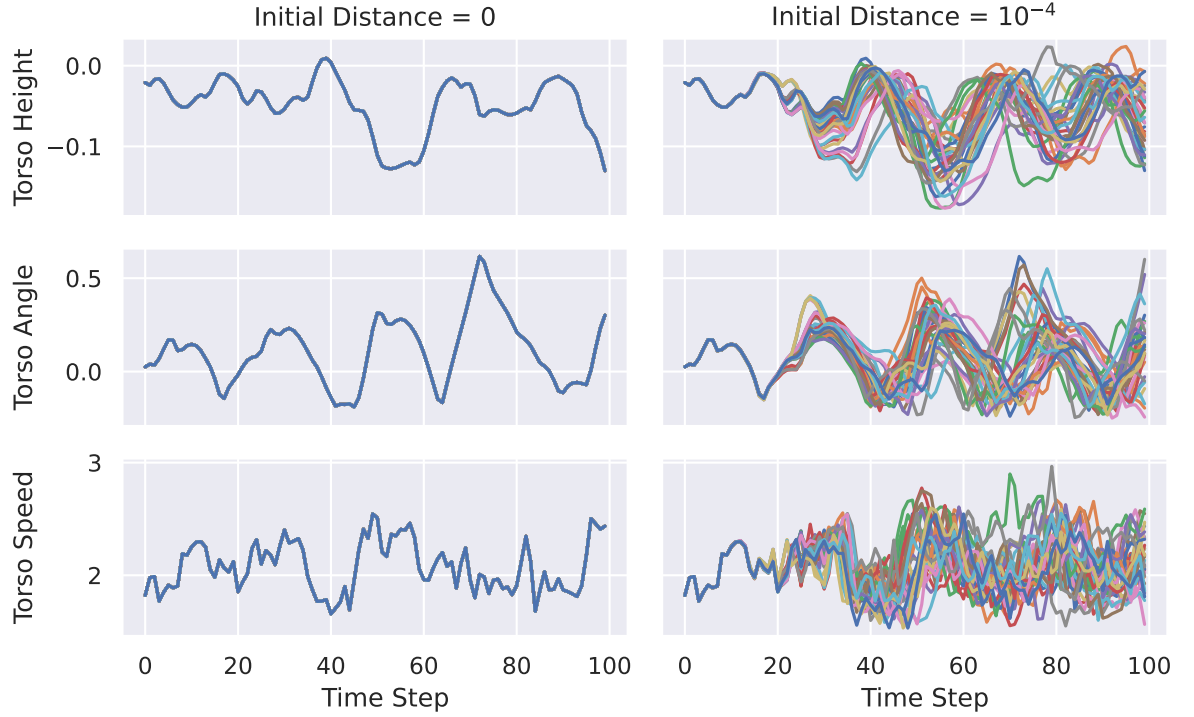


Figure 4.8: Partial state trajectory produced by SAC when controlling the *Walker Walk* with and without an initial state perturbation. All initial states produce the same state trajectories when unperturbed (left) but deviate significantly over 100 steps when initially perturbed by 10^{-4} (right).

However, when controlled by Dreamer V3, these simple systems exhibit low levels of chaos, as they have positive MLE and negative SLE. As a result, state trajectories in these environments are more sensitive to initial conditions, with similar states producing significantly different long-term outcomes. Instead of converging to a single state, these trajectories exhibit sensitive behaviour, continuously orbiting within a region that yields high rewards.

In more complex, high-dimensional environments (*Walker Stand*, *Walker Walk*, *Walker Run*, *Cheetah Run*), all methods induce chaotic dynamics, as indicated by a positive MLE and negative SLE. This means that these policies cannot account for arbitrarily small changes in the system’s state since small changes produce exponentially diverging state trajectories, as illustrated in Figure 4.8. In effect, the policy has a predictability horizon of roughly 10 steps before an initial error grows by a factor of e . This brief Lyapunov time poses a significant problem for real-world applications of RL, as observation perturbations are easily introduced via imperfect measurements or sensor noise. Therefore, for complex high-dimensional environments controlled by deep RL policies, it is impossible to guarantee predictability, as the system cannot correct itself after a single inaccurate observation.

4.3.2 Spatial Distribution of the Maximal Lyapunov Exponent

To further investigate the predictability properties of deep RL policies, we analyse the spatial distribution of the Maximal Lyapunov Exponent across different regions of the state space. Rather than assuming a uniform distribution, we estimate the MLE at various initial positions within each environment to examine how chaotic behaviour manifests spatially. The MLE is computed using the same methodology outlined in Section 4.3.1 but with initial states sampled from a uniform grid spanning the state space. We focus on the four-dimensional systems (*Pointmass*, *Cartpole*) as this allows us to visualise the spatial MLE when the initial velocity is zero.

Figure 4.9 illustrates the spatially distributed MLE produced by a trained instance of SAC and the No-action baseline. These results indicate that the deep RL policy directly influences the predictability of the control interaction, as there is a significant difference in MLE between SAC and the No-action baseline. Notably, the MLE is negative in regions corresponding to the agent’s initial state distribution and high-reward areas, suggesting that the agent has learnt to stabilise these regions. In contrast, the MLE is positive in regions lacking training incentives, where the agent may not have learned optimal behaviours.

This contrast in MLE is particularly evident when comparing the *Swingup* and *Balance* variations of the *Cartpole* environment when controlled by SAC. These systems share the same underlying dynamics and only differ in their initial state distribution. Despite these similarities, the trained deep RL policy produces markedly different spatial MLEs in each environment. These findings suggest that, while deep RL policies can exhibit predictability within their training data distribution, they do not necessarily generalise this predictability to unseen regions of the state space. This lack of robustness outside the training distribution poses a significant challenge for real-world deployment.

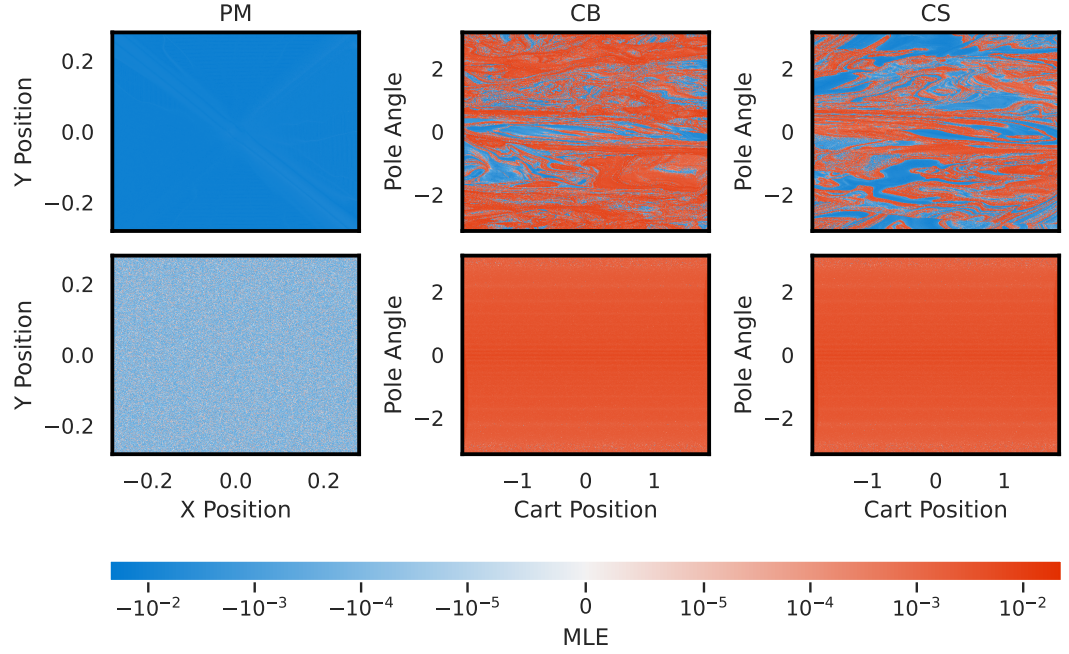


Figure 4.9: Spatial distribution of the Maximal Lyapunov Exponent (MLE) for different environments controlled by trained RL policies. Warmer colours indicate regions with high sensitivity to initial conditions (chaotic behaviour), while cooler colours indicate predictable regions. The first row is generated using SAC while the second row is generated by the No-action baseline.

4.4 Topological Transitivity

In this section, we investigate whether the interaction between deep reinforcement learning controllers and continuous environments produces topologically transitive state dynamics. Specifically, we examine if the state trajectories originating from any open subsets of the state space $(A, B \subseteq \mathcal{S})$ converge after $\mathbf{f}$ iterates A for $n \in \mathbb{N}$ steps:

$$f^n(A) \cap B \neq \emptyset \quad (4.3)$$

This property ensures that the system’s trajectories are sufficiently mixed over time, thereby enabling trajectories from distinct regions of the state space to eventually overlap. However, verifying this property across all possible subsets of a continuous state space is computationally intractable. Therefore, to approximate topological transitivity, we introduce a relaxed definition that restricts attention to reachable regions of the state space. Specifically, we partition the initial state distribution into two disjoint subsets $(A, B \subset \rho_0)$ and assess if the corresponding state trajectories converge after both sets are iterated for $n \in \mathbb{N}$ steps:

$$f^n(A) \cap f^n(B) \neq \emptyset. \quad (4.4)$$

This relaxed definition of topological transitivity has two practical benefits. First, it restricts the state space to the set of states that the control system can attain. Second, it allows multiple sets to be evaluated collectively. Specifically, for $n \in \mathbb{N}$, $A' = f^n(A)$, and $B' = f^n(B)$, we have:

$$f^n(A) \cap f^n(B) \neq \emptyset \Rightarrow f^n(A) \cap B' \neq \emptyset \quad (4.5)$$

$$f^n(A) \cap f^n(B) \neq \emptyset \Rightarrow A' \cap f^n(B) \neq \emptyset. \quad (4.6)$$

However, demonstrating that trajectories originating from a specific pair of disjoint subsets eventually overlap is not a sufficient condition for the formal proof of global topological transitivity (Definition 2). Instead, global transitivity requires that this property holds for any arbitrary pair of open subsets within the continuous state space. Consequently, this approach identifies a weaker notion of topological transitivity, restricted both to the regions of the state space reachable from the initial state distribution and to the specific subsets selected for evaluation. Despite these theoretical limitations, this measure can still provide a narrow indication of whether the underlying dynamics facilitate state-space mixing within the boundaries of the induced attractor.

To empirically test whether this relaxed condition holds within the deep RL control interaction, we partition the initial state distribution of each environment into two disjoint regions and sample 100 initial states from each. These samples serve as representative point estimates for each subset and form the basis for assessing whether the regions exhibit transitive behaviour under the policy–environment interaction. Further increasing the number of samples will provide a more accurate representation of the initial state distribution at the expense of increased computational burden. We therefore use 100 states, as this provides a good empirical distribution without incurring significant overhead. Each sampled state is then propagated forward via the POMDP for 1 episode (1000 steps). This process produces two ensembles of trajectories that evolve from distinct initial conditions, which can be used to determine if the system produces topologically transitive dynamics.

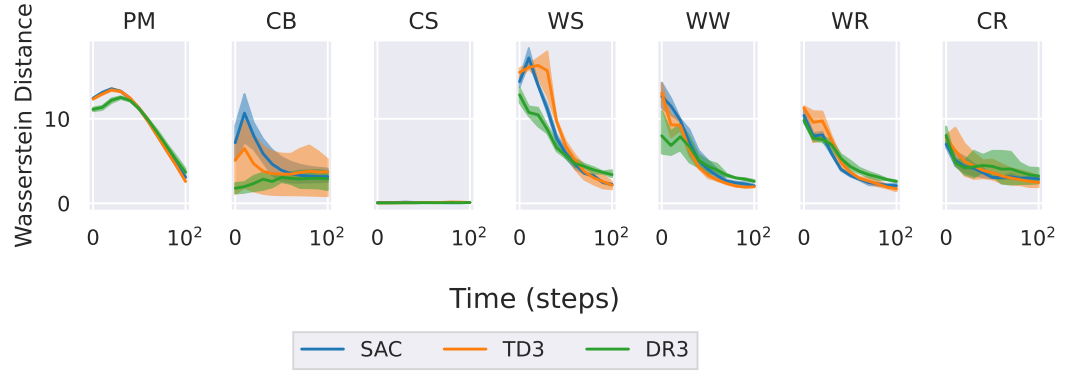


Figure 4.10: Wasserstein distance between two sets of initially disjoint state trajectories when the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments when controlled by trained instances of SAC, TD3, and Dreamer V3 (DR3). Convergence to zero indicates topologically transitive state dynamics.

To identify the similarity between these ensembles of trajectories, we measure the Wasserstein distance [Vaserstei, 1969, Dobrushin, 1970] at each time step (Figure 4.10). We find that the distance between these initially disjoint sets consistently decays to zero as time increases. This indicates that the initially disjoint sets become increasingly similar, thus indicating topological transitivity. This result is intuitive for the *Pointmass* and *Cartpole* environments, as they produce a fixed point attractor in the state space when controlled by deep RL policies. Consequently, the states sampled from the initially disjoint regions converge to the same position over time.

In contrast, as the dynamics in the *Walker* and *Cheetah* environments converge to an attractor with non-zero volume, the presence of topological transitivity in these systems implies that the trajectories originating from distant initial conditions become increasingly entangled within this attractor. This phenomenon becomes more apparent when viewing the hyperplane projections of the state trajectories with initially distinct states (Figure 4.11). From this, we can identify that the long-term state trajectories occupy the same space despite starting from distinct initial positions.

Overall, these results provide empirical evidence that the control interaction induces state-space mixing within the reachable regions of the state space. Although these findings are restricted to the specific subsets evaluated and do not provide a formal guarantee of global transitivity, the observed convergence of the representative ensembles suggests that trajectories within the induced attractor become increasingly entangled. This localised mixing contributes to the long-term unpredictability of the resulting dynamics, as the state trajectories eventually visit the neighbourhood of any state in the attractor.

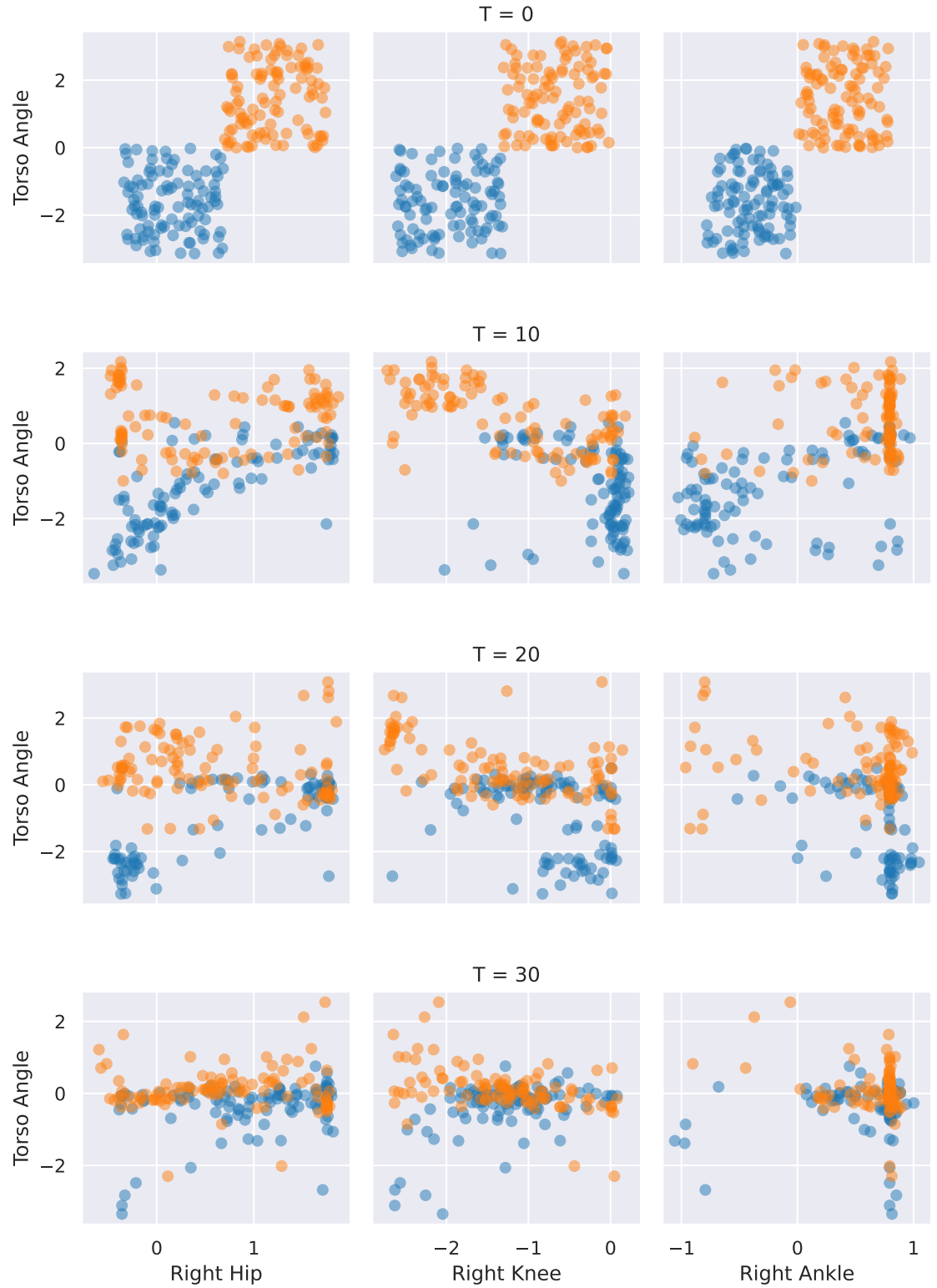


Figure 4.11: Empirical distribution of states attained when SAC controls the *Walker Walk* environment after $t = \{0, 10, 20, 30\}$ steps. The initial states for each trajectory are sampled from a partition of the initial state distribution.

4.5 Dense Periodic Trajectories

In this section, we investigate whether the interaction between deep reinforcement learning policies and continuous control environments contains dense periodic trajectories, a necessary criterion for establishing if dynamical systems are Devaney chaotic. However, in the context of reinforcement learning, locating these trajectories within the control loop presents significant challenges as the system’s transition function is generally unknown. Moreover, for systems that exhibit sensitivity to initial conditions, any small deviations from a dense periodic trajectory will cause significant deviations over time. As a result, the identification of dense periodic trajectories within the RL control loop is practically infeasible.

Despite these limitations, it is still possible to identify approximately periodic behaviour using the recurrence plot [Marwan et al., 2007]. This method constructs a distance matrix ($\mathbf{D} \in \mathbb{R}^{N \times N}$) (Equation 4.7), which captures the pairwise Euclidean distances between states across a trajectory of length N .

$$D_{i,j} = \|\mathbf{s}_i - \mathbf{s}_j\|_2. \quad (4.7)$$

Visually, this matrix reveals the underlying system dynamics through distinct geometric signatures. Near-constant state trajectories produce a matrix dominated by near-zero values, while non-repeating trajectories produce large distance values. In contrast, periodic dynamics are represented by a range of both large and small values that form structured diagonal patterns.

Furthermore, the recurrence rate (Equation 4.8) [Marwan et al., 2007] provides a quantitative measure of these patterns. In effect, this metric identifies the proportion of non-trivial pairs of states at distinct time steps where the state trajectory returns within an ϵ ball of itself.

$$RR(\epsilon) = \frac{1}{N(N-1)} \sum_{i \neq j}^N \begin{cases} 1 & \text{if } D_{i,j} < \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (4.8)$$

Values approaching 1 indicate converging state dynamics, while values approaching 0 suggest non-repeating behaviour. Values between 0 and 1 represent periodic behaviour, indicating that the system undergoes repeated state visits over time. While this does not provide a formal guarantee of the existence of dense periodic trajectories, it offers a practical proxy for assessing periodic behaviour when the transition function is unavailable.

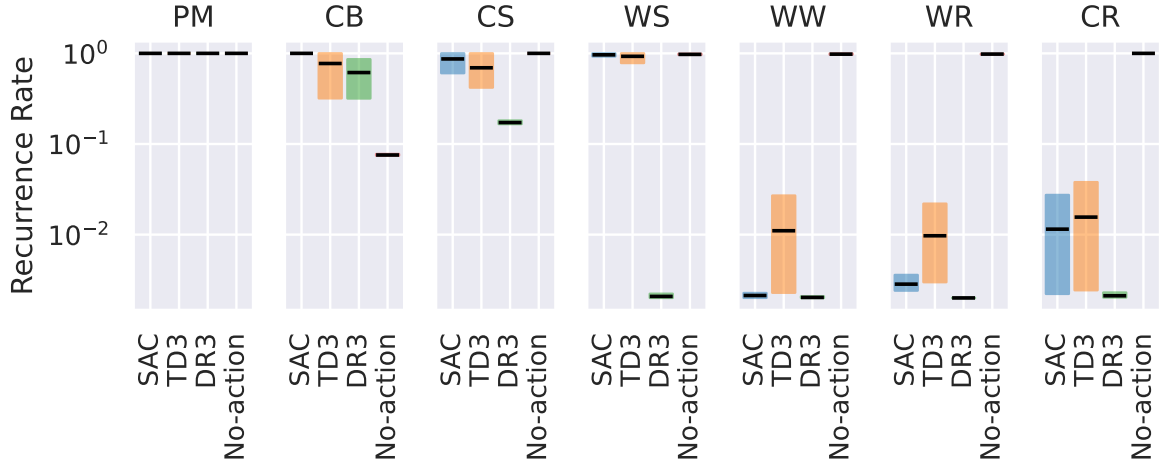


Figure 4.12: Recurrence rate for the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments when controlled by trained instances of SAC, TD3, Dreamer V3 (DR3), and an agent which takes no actions (No-action). Values close to 1 indicate static state dynamics, values close to 0 indicate non-repeating dynamics, and values between 0 and 1 indicate periodic state dynamics.

To measure the recurrence rate, we generate 40 state trajectories of length 1000 (1 POMDP episode) for each policy–environment interaction and calculate the distance between each trajectory and its delayed embedding. Increasing the number of initial states will provide a more accurate measure of the recurrence rate. However, we limit this to 40 initial states as this is a computationally expensive process. Furthermore, when calculating the distance between states, we exclude any dimensions that are monotonically increasing, as this corresponds to a drifting attractor. We then calculate the proportion of times this distance is less than $\epsilon = 0.1$, excluding the trivial case where no delay occurs. This value is selected to approximate a local neighbourhood that is small relative to the overall state space but large enough to be numerically stable.

Figure 4.12 provides the average recurrence rate with 95% bootstrapped confidence intervals for each control interaction. This indicates that the *Pointmass*, *Cartpole*, and *Walker Stand* have a recurrence rate of approximately 1, thereby indicating near-constant state dynamics (Figure 4.13). This behaviour is consistent with the presence of static (partial) attractors and the design of their reward functions, as they constrain the position dimensions to fixed values.

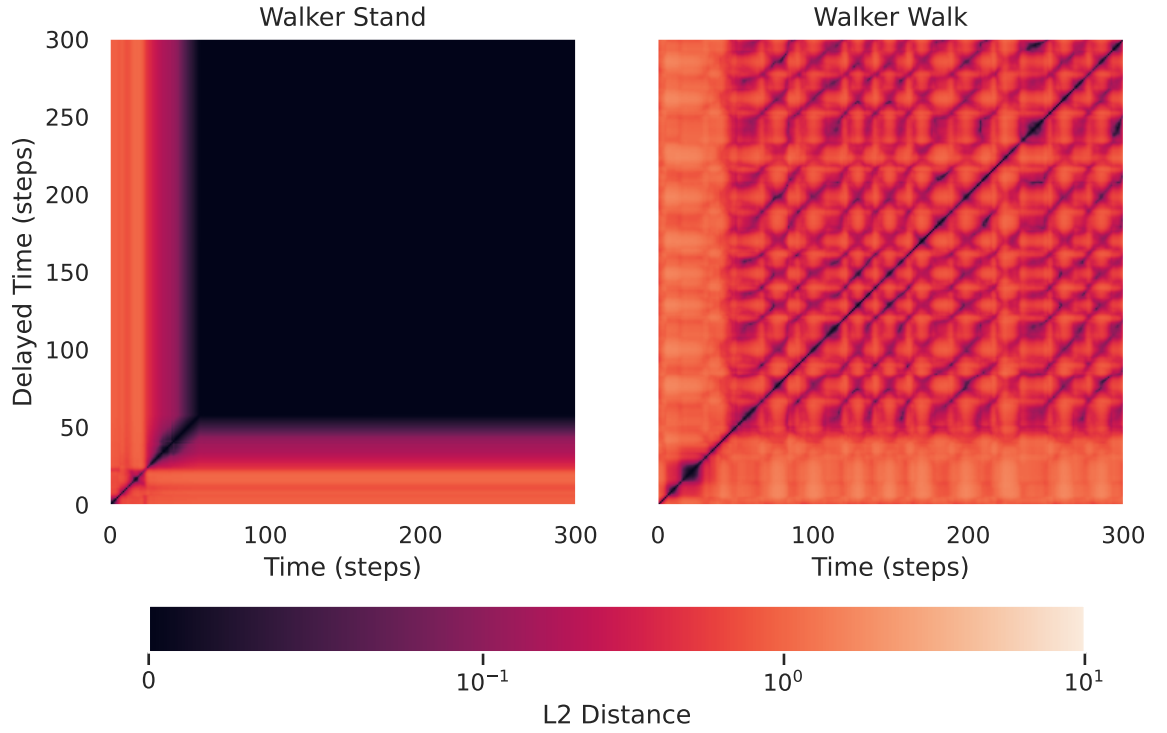


Figure 4.13: Recurrence plots for the *Walker Stand* (a) and *Walker Walk* (b) environments when controlled by a trained instance of SAC.

In contrast, the *Walker Walk*, *Walker Run*, and *Cheetah Run* environments produce a recurrence rate of approximately 10^{-3} . This suggests that these systems occasionally return to previously visited states but do so with low frequency. Furthermore, inspecting the recurrence plot for the *Walker Walk* environment when controlled by SAC (Figure 4.13) shows that the state dynamics are aperiodic, as indicated by the inconsistent off-diagonal lines. Together, these findings suggest that deep reinforcement learning policies induce periodic behaviour in high-dimensional control tasks, which provide high reward for attaining a fixed velocity.

4.6 Reward Predictability

In this section, we investigate how the chaotic state dynamics impact a policy’s performance. Using the Maximal Lyapunov Exponent, we identify the predictability of the reward attained and show that chaotic control systems create a fractal return surface, in which small state perturbations produce significantly different total rewards. This makes chaos particularly detrimental for reward, as policies with high average returns can still produce highly variable and unreliable outcomes.

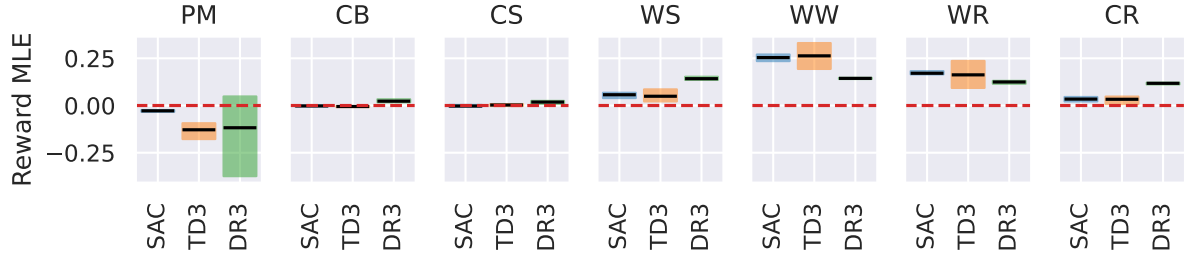


Figure 4.14: Reward MLE interquartile mean for the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) when controlled by SAC, TD3, and Dreamer V3 (DR3). Each policy–environment combination is independently trained with three random seeds, and the reward MLE for each seed is calculated using 20 initial states. A bootstrapped 95% confidence interval is included to show the variation in reward MLE across random seeds.

By considering the reward over a state trajectory as a trajectory in a one-dimensional reward space, the predictability of the reward can be measured using Lyapunov Exponents. Given that this space is one-dimensional, only one Lyapunov Exponent exists, and this single exponent can reliably identify the predictability of reward trajectories. A negative MLE indicate that small perturbations to the system’s state produce converging long-term rewards. Moreover, for a bounded reward function, the reward trajectories cannot diverge indefinitely, so a positive MLE indicates that the long-term reward is chaotic, as small changes to the system state produce exponentially diverging bounded reward trajectories.

Figure 4.14 provides the average reward MLE and bootstrapped 95% confidence interval for each policy–environment pair. This indicates that the reward trajectories produced in simple low-dimensional environments (*Pointmass*, *Cartpole Balance*, *Cartpole Swingup*) are predictable as they have negative reward MLE. As a result, a single small state perturbation does not harm the total reward attained as the subsequent reward trajectories converge. Conversely, in high-dimensional control systems (*Walker Stand*, *Walker Walk*, *Walker Run*, *Cheetah Run*), the reward MLE is positive, indicating that the reward is highly sensitive to arbitrarily small changes in the system state. Consequently, reward trajectories in these systems are unpredictable, as similar initial states produce significantly different sequences of rewards.

Plotting the reward trajectories for one of these chaotic interactions (Figure 4.15) shows that a single perturbation with varying magnitude can have a huge detrimental impact on the total reward attained despite performing well on average. This poses a significant problem for real-world applications of RL, as it is impossible to guarantee worst-case performance in a chaotic control system.

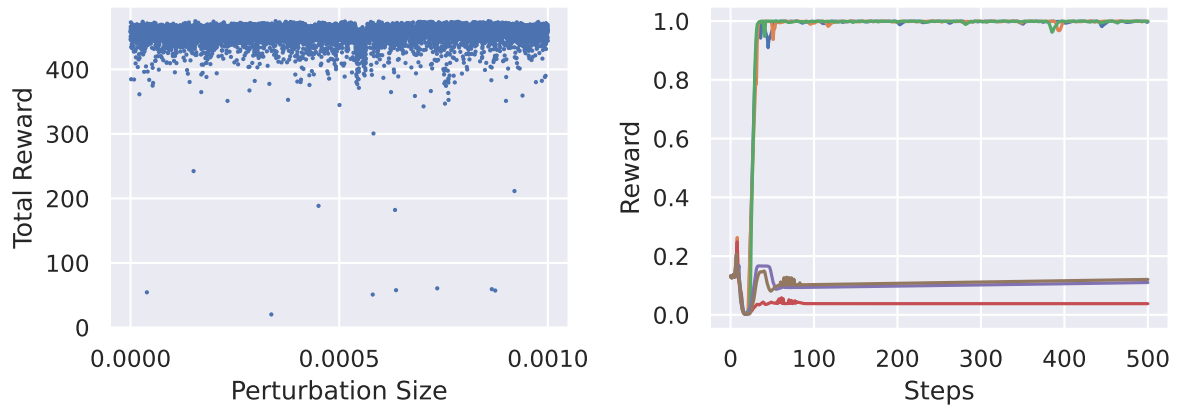


Figure 4.15: Left: Total reward attained by a deterministic SAC policy when controlling the deterministic *Walker Walk* environment subject to an initial perturbation with fixed direction and varying magnitude. Right: Rewards attained by the three best and three worst state trajectories subject to this perturbation.

4.7 Identifying the Impact of RL Hyperparameters on Predictability

Training deep RL policies involves numerous design choices that influence how an agent learns, explores, and generalises to different regions of the state space. While these choices are typically tuned to maximise reward, they may also have unintended consequences on the predictability of the learned policy. To understand the extent of this influence, we train three instances of SAC while systematically varying hyperparameters that are common amongst model types.

4.7.1 Control Frequency

The rate at which new actions are applied to the environment has a pivotal impact on the overall control interaction as it influences the system’s responsiveness and the complexity of decision-making. A high decision rate causes new actions to be applied frequently, allowing for fine-grained adjustments and precise control. However, this comes at the cost of an increased computational burden and potentially noisier decision-making due to frequent policy evaluations. Conversely, a low decision rate, where new actions are applied less frequently, reduces the computational overhead and can smooth out rapid fluctuations. However, this may also introduce delayed reactions and decreased adaptability to fast-changing dynamics.

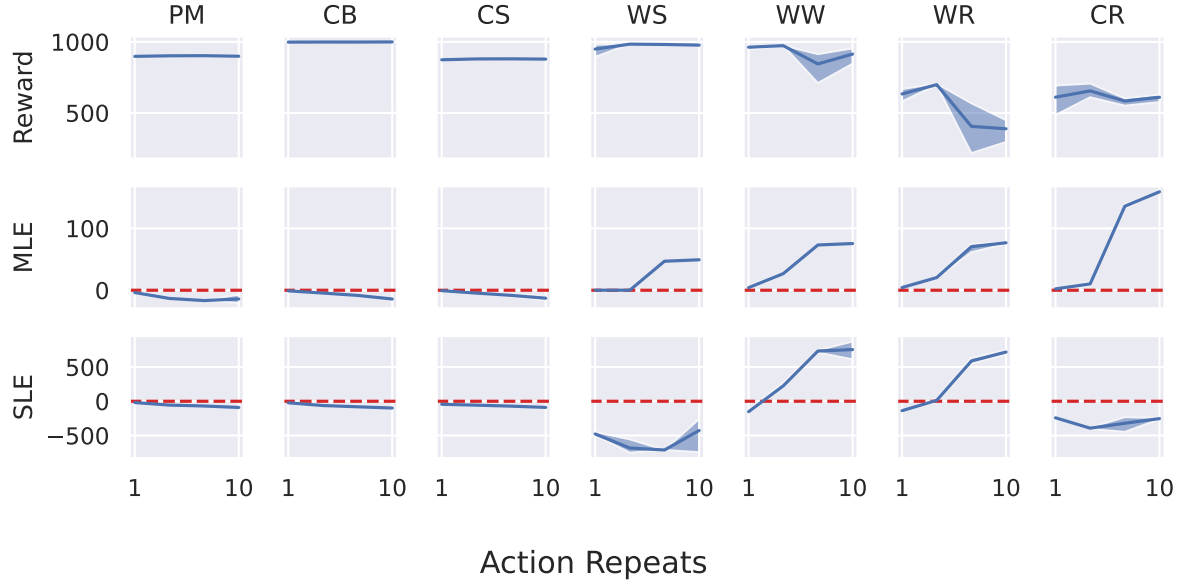


Figure 4.16: Bootstrapped 95% confidence interval of the average reward, MLE, and SLE produced by SAC when controlling the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments with the number of action repeats varying between one and ten.

To identify the impact that the decision frequency has on the predictability of the control interaction, we train three instances of SAC, varying the number of action repeats from 1 to 10. This produces a set of control interactions for each environment, each operating over a unique temporal horizon. Consequently, the methods for measuring the total reward and Lyapunov Exponents need to account for this change in decision frequency. First, we set each environment type to have the same overall control interaction time and control frequency while varying the number of times the same action is repeatedly applied to the environment. Furthermore, to maintain consistency between different control interaction rates, the reward for each POMDP step is set to the sum of the individual substeps. Consequently, the total possible reward over an entire episode is 1000, despite varying control interaction frequencies and lengths. Finally, when calculating the Lyapunov Exponents, we divide the log divergence by the control frequency, as this causes the exponents to be measured over seconds instead of steps.

Figure 4.16 provides the bootstrapped 95% confidence interval for the average total reward, MLE and SLE when varying the number of times the same action is applied to the control system. We find that environments that need to attain a static position to achieve high reward (*Pointmass*, *Cartpole*, *Walker Stand*) maintain similar performance when the decision frequency decreases. Conversely, systems in which a high reward is provided for maintaining a specific velocity (*Walker Walk*, *Walker Run*, *Cheetah Run*) show a slight degradation in performance as the decision frequency decreases. This result is intuitive,

as systems required to maintain a fixed velocity continuously traverse the state space and must rapidly adjust to minor state variations. In contrast, systems tasked with reaching a fixed position remain largely static once the target position is attained, reducing the necessity for frequent adjustments.

The differences between systems with static and dynamic solutions, subject to differing decision frequencies, are particularly noticeable when considering the MLE and SLE. The static systems (with the exception of *Walker Stand*) consistently produce predictable dynamics despite the differing control frequencies. This is because these systems always reach a fixed attractor. Consequently, any small initial changes do not have a long-term impact on the system despite the differing decision frequencies. The exception to this is the *Walker Stand* environment, which has an increased level of chaos. This is because the fixed point it needs to attain is only specified by the torso height and angle, and thus the system can stabilise the range of differing fixed points. The increase in MLE introduced by a larger number of action repeats is due to the system moving further away from the original position. Finally, we find that systems with transient solutions (*Walker Walk*, *Walker Run*) produce positive SLE as the decision frequency decreases. Therefore, these systems are not bound by an attractor and thus produce diverging dynamics.

4.7.2 Total Time Steps

The total duration of training plays a crucial role in shaping the effectiveness of learned policies, affecting both the convergence and the generalisation capabilities of the resulting policy. Training over an extended period allows the agent to explore a broader range of scenarios, thereby refining its decision-making. However, prolonged training also increases computational costs and the risk of overfitting to specific environment characteristics. In contrast, shorter training durations limit resource expenditure but may lead to suboptimal policies due to insufficient exploration. Typically, the total number of training steps is defined by a benchmark problem or tuned to maximise the total reward via hyperparameter optimisation techniques, neither of which accounts for the predictability of the learnt policy.

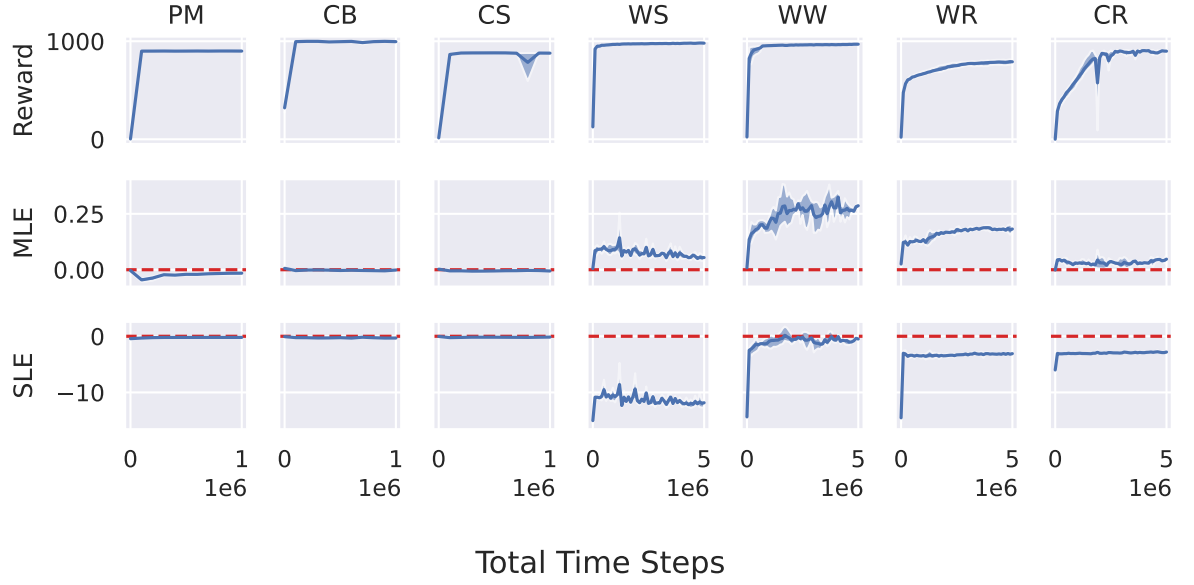


Figure 4.17: Bootstrapped 95% confidence interval of the average reward, MLE, and SLE produced when training SAC on the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments.

To identify how the predictability of the control interaction evolves throughout the learning process, we train three instances of SAC for 1 million (*Pointmass*, *Cartpole*) or 5 million (*Walker*, *Cheetah*) environment steps and measure the reward and Lyapunov Exponents of the learnt policy every 100,000 steps. Training for this long duration allows the policy to converge to a high-reward solution, as shown in Figure 4.17. Furthermore, this figure also shows that at the beginning of training, the MLE is approximately zero, which aligns with the No-action baseline. This suggests that the initial dynamics are neutral, and the initialised policy does not introduce significant deviations from the passive system. As training progresses, we observe that the learnt policy induces predictable dynamics in the *Pointmass* and *Cartpole* environments, while in *Walker* and *Cheetah*, the policy introduces chaotic behaviour, as evidenced by a positive MLE. Notably, in the *Walker Stand* environment, we observe an initial spike in MLE, followed by a gradual decline as training proceeds. This trend suggests that the system initially exhibits chaotic dynamics but becomes progressively more predictable as the policy refines its control strategy.

Overall, these findings demonstrate that the predictability characteristics of the learnt policy continue to evolve throughout training, even after the reward has plateaued. This highlights the need to consider predictability as a distinct criterion when evaluating reinforcement learning policies for real-world deployment.

4.8 Discussion

In this chapter, we set out to identify if deep reinforcement learning controllers induce predictable, diverging, or chaotic state dynamics when controlling continuous control systems. Following the frameworks established by Lyapunov and Devaney, we identified if these systems form an attractor in Section 4.2, are sensitive to initial conditions in Section 4.3, display topological transitivity in Section 4.4, and contain periodic trajectories in Section 4.5. A summary of these findings is presented in Table 4.1.

The purpose of this investigation was to determine whether state-of-the-art deep RL controllers induce predictable state dynamics in a continuous control environment, and to identify which of the frameworks of chaos outlined by Lyapunov or Devaney is more suitable for RL control systems. These objectives are addressed in Sections 4.8.1 and 4.8.2 respectively.

4.8.1 Predictability of the Control Interaction

Overall, our results demonstrate that deep reinforcement learning policies induce predictable and chaotic state dynamics when controlling environments with continuous state and action spaces. Specifically, we find that when the model-free agents control the low-dimensional systems (*Pointmass*, *Cartpole*), the resulting state dynamics are consistently predictable and converge to a fixed-point attractor. These results are intuitive, as the reward function for these systems provides high rewards for maintaining the control system at a fixed point with zero velocity. Therefore, achieving strong task performance should induce predictable state dynamics.

The exception to this occurs when the low-dimensional systems are controlled by Dreamer V3. We find that this control interaction produces an attractor with non-zero volume, has a positive Maximal Lyapunov Exponent and a negative Sum of Lyapunov Exponents, thereby indicating that the control interaction is Lyapunov chaotic. However, this does not match with the Devaney framework, as the system does not produce periodic behaviour. This mismatch in results occurs because the Lyapunov Exponents measure local short-term unpredictability, whereas the periodic behaviour is measured over a longer period of time.

Furthermore, we find that when any deep RL policy controls the high-dimensional systems (*Walker*, *Cheetah*), the resulting dynamics are both Lyapunov and Devaney chaotic. This can be established by observing that the state trajectories in these systems converge to an attractor with positive volume, are highly sensitive to small changes in the system’s state, and exhibit topological transitivity and periodicity. Taken together, this means that these policies cannot account for arbitrarily small changes in the system’s state since small perturbations produce exponentially diverging state trajectories.

This unpredictability poses two threats to the application of deep RL to real-world problems, where it is infeasible to attain an accurate measurement of the system state. First, small state perturbations can have a significant impact on the performance of trained deep RL policies, even when the average performance is good. This can create hazards in real-world conditions where observation noise is prevalent and can be exploited by adversarial attack methods. Second, even when deep RL policies perform well, unpredictable behaviours are undesirable in most real-world applications.

Collectively, these findings highlight a critical limitation of current deep RL methods. Primarily focusing on maximising reward alone does not yield control policies that are predictable. Predictability is neither an emergent property of high reward nor guaranteed by convergence during training. Instead, it must be explicitly incorporated into the training objective.

4.8.2 Devaney vs Lyapunov

To evaluate the predictability of deep reinforcement learning controllers, we compared two definitions of chaos. Devaney’s definition characterises chaos as requiring three properties: sensitivity to initial conditions, topological transitivity, and the presence of dense periodic trajectories. This formulation provides a useful structural characterisation of chaotic systems, especially in idealised settings. However, in our empirical setting, Devaney’s definition of chaos proved too rigid for empirical verification, as its structural conditions are difficult to verify in high-dimensional RL systems with unknown transition dynamics.

In contrast, the Lyapunov approach defines chaos in terms of the exponential divergence of nearby trajectories. This directly measures the unpredictability that arises from small perturbations, which is arguably the defining feature of chaotic systems. In practice, we find this measure to be more relevant for the control settings, as a positive Maximal Lyapunov Exponent indicates unpredictable behaviour. Moreover, although computing exact Lyapunov Exponents requires differentiable dynamics, various approximation techniques allow their estimation even for non-differentiable or stochastic systems.

Overall, we recommend the Lyapunov framework as it is a more appropriate tool for analysing chaos in RL-controlled systems. It provides a quantitative measure of the system's sensitivity to initial conditions, which is the most important aspect of chaos from the perspective of control.

Table 4.1: Classification of the system dynamics produced by SAC, TD3, and Dreamer V3 (DR3) when controlling continuous environments sampled from the DeepMind Control Suite.

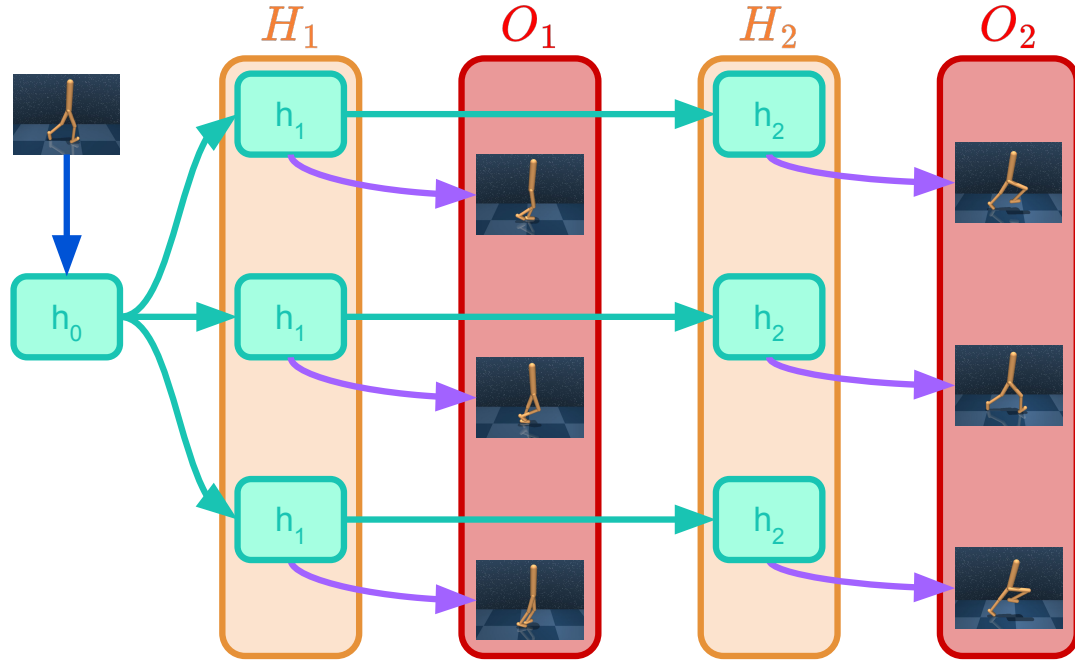
Environment	Agent	Attractor	Sensitive	Mixing	Periodic	System Dynamics
<i>Pointmass</i>	SAC	Stationary fixed point	✗	✓	✗	Predictable
	TD3	Stationary fixed point	✗	✓	✗	Predictable
	DR3	Stationary attractor	✓	✓	✗	Aperiodic Sensitive Mixing
<i>Cartpole Balance</i>	SAC	Stationary fixed point	✗	✓	✗	Predictable
	TD3	Stationary fixed point	✗	✓	✗	Predictable
	DR3	Stationary attractor	✓	✓	✗	Aperiodic Sensitive Mixing
<i>Cartpole Swingup</i>	SAC	Stationary fixed point	✗	✓	✗	Predictable
	TD3	Stationary fixed point	✗	✓	✗	Predictable
	DR3	Stationary attractor	✓	✓	✓	Chaotic
<i>Walker Stand</i>	SAC	Moving attractor	✓	✗	✗	Unpredictable Fixed Points
	TD3	Moving attractor	✓	✗	✗	Unpredictable Fixed Points
	DR3	Moving attractor	✓	✓	✓	Chaotic
<i>Walker Walk</i>	SAC	Moving attractor	✓	✓	✓	Chaotic
	TD3	Moving attractor	✓	✓	✓	Chaotic
	DR3	Moving attractor	✓	✓	✓	Chaotic
<i>Walker Run</i>	SAC	Moving attractor	✓	✓	✓	Chaotic
	TD3	Moving attractor	✓	✓	✓	Chaotic
	DR3	Moving attractor	✓	✓	✓	Chaotic
<i>Cheetah Run</i>	SAC	Moving attractor	✓	✓	✓	Chaotic
	TD3	Moving attractor	✓	✓	✓	Chaotic
	DR3	Moving attractor	✓	✓	✓	Chaotic

Chapter 5

Minimising Chaos with Predictions

In Chapter 4, it was shown that deep reinforcement learning policies can exhibit chaotic dynamics when applied to continuous control tasks. Consequently, any small changes to the system’s state can produce significantly different state trajectories, which can severely degrade task performance. This lack of robustness presents a significant barrier for the safe deployment of deep RL controllers in real-world settings, where local disturbances are common.

To address this issue, this chapter seeks to understand if constraining the predicted MLE during policy updates can improve the predictability of deep RL controllers without compromising task performance. To this end, we propose Maximal Lyapunov Exponent regularisation for Dreamer V3. This novel technique uses the stochastic recurrent state space model to estimate possible future state trajectories under the influence of the current policy. The divergence between these predicted trajectories is incorporated as an additional term in the policy loss, thereby constraining the predicted MLE during policy updates (Figure 5.1). In effect, this forces the agent to optimise its confidence in future trajectories jointly with its expectations of rewards.



$$\mathcal{L}^{\lambda_1}(\theta) \doteq \underset{L}{\text{Var}}(H_1) + \underset{L}{\text{Var}}(O_1) + \underset{L}{\text{Var}}(H_2) + \underset{L}{\text{Var}}(O_2) + \dots$$



Figure 5.1: Schematic illustration of MLE regularisation for Dreamer V3. The first observation is encoded into the latent space via the Encoder. Three independent variants of this internal state are unrolled using the stochastic Recurrent State Space Model (RSSM) and decoded into observations via the Decoder. During policy updates, MLE regularisation constrains the variance between the set of predicted internal states and the variance between the set of predicted observations at each time step.

We demonstrate that incorporating this regularisation term in Dreamer V3’s policy loss function significantly reduces the chaotic dynamics produced by this state-of-the-art deep RL controller. Furthermore, this increased predictability dramatically improves the policy’s robustness to observation noise, thereby enhancing the feasibility of deep RL agents for real-world continuous control tasks. Finally, we show that this additional term introduces a trade-off between predictability and performance, as excessively weighting it over-constrains the policy, resulting in highly predictable but low-reward controllers. Taken together, these findings support the hypothesis that explicitly constraining chaotic dynamics can enhance the predictability of Dreamer V3.

5.1 Methods

The Maximal Lyapunov Exponent quantifies how sensitive a control system is to local state perturbations under the influence of a control policy. It emerges from the repeated composition of the policy and environment transition functions and can only be directly measured when these functions are provided. In the standard RL paradigm, this poses a significant challenge as the transition function is not explicitly represented and can only be measured via sampling. However, for model-based RL methods, the MLE can be approximated without the need for an explicit transition function, as these policies learn a differentiable representation of the system dynamics. As a result, model-based RL offers the most practical approach for directly minimising the MLE during training.

However, controlling the MLE using model-based RL introduces additional complexities, as these methods often use an internal representation for action generation and observation prediction. Even under a deterministic policy, the same observation may yield different actions due to variation in the internal representation. Therefore, to minimise the long-term state divergence produced by these controllers, the internal state must be constrained in addition to the predicted observation. Consequently, to reduce the chaotic dynamics produced by deep RL controllers, we aim to minimise $\lambda_o(\theta)$ in the observation space $\mathcal{O}$ (Equation 5.1) and $\lambda_h(\theta)$ in the hidden space $\mathcal{H}$ (Equation 5.2).

$$\lambda_o(\theta) = \lim_{t \rightarrow \infty} \lim_{\epsilon \rightarrow 0} \frac{1}{t} \ln \left(\frac{|\mathbf{o}_t - \hat{\mathbf{o}}_t|}{\epsilon} \right) \quad (5.1)$$

$$\lambda_h(\theta) = \lim_{t \rightarrow \infty} \lim_{\epsilon \rightarrow 0} \frac{1}{t} \ln \left(\frac{|\mathbf{h}_t - \hat{\mathbf{h}}_t|}{\epsilon} \right) \quad (5.2)$$

While Equations 5.1 and 5.2 provide a formal definition of the MLE in the observation and hidden spaces, the presence of two nested limits renders direct estimation infeasible in practice. The limit as $t \rightarrow \infty$ demands long trajectories that are computationally expensive and prone to compounding prediction errors, while the limit as $\epsilon \rightarrow 0$ requires infinitesimal perturbations, which are challenging to realise numerically. However, these limits can be avoided by measuring divergence over a short prediction horizon ($T \in \mathbb{N}$) and using a stochastic dynamics model to approximate a trivially small perturbation ($0 < \epsilon \ll 1$). Furthermore, as T is fixed, $\ln(\cdot)$ is monotonically increasing and ϵ has been approximated via the stochastic dynamics model, the observation MLE and hidden state MLE are approximately at a minimum when the predicted states and predicted hidden states converge (Equations 5.3 and 5.4).

$$\arg \min_{\theta} \lambda_o(\theta) \approx \arg \min_{\theta} |\mathbf{o}_T - \hat{\mathbf{o}}_T| \quad (5.3)$$

$$\arg \min_{\theta} \lambda_h(\theta) \approx \arg \min_{\theta} |\mathbf{h}_T - \hat{\mathbf{h}}_T| \quad (5.4)$$

Therefore, minimising $\lambda_o(\theta)$ and $\lambda_h(\theta)$ requires reducing the difference between state and hidden state trajectories predicted using the stochastic dynamics model over a short time horizon. This can easily be achieved by incorporating $|\mathbf{o}_T - \hat{\mathbf{o}}_T|$ and $|\mathbf{h}_T - \hat{\mathbf{h}}_T|$ as weighted terms in the policy loss. However, only measuring the distance between two sampled states can introduce estimation bias. Therefore, instead of directly minimising the distance between two predicted states, we propose using the stochastic dynamics model to make $L \in \mathbb{N}$ state trajectories ($\mathbf{O}_t = \{\mathbf{o}_{t,1}, \mathbf{o}_{t,2}, \dots, \mathbf{o}_{t,L}\}$) and internal representation trajectories ($\mathbf{H}_t = \{\mathbf{h}_{t,1}, \mathbf{h}_{t,2}, \dots, \mathbf{h}_{t,L}\}$). Each of these provides a plausible estimate of the future states and thus using the variance between these predicted states (Equation 5.5) provides a less biased regularisation term.

$$\mathcal{L}_t^{\lambda_1}(\theta) \doteq \text{Var}_L(\mathbf{O}_t) + \text{Var}_L(\mathbf{H}_t). \quad (5.5)$$

Finally, as we want convergence along the full trajectory and not a state T steps in future, we also constrain the distance between predicted trajectories at $t \in [1, T]$. This gives the regularisation term:

$$\begin{aligned} \mathcal{L}^{\lambda_1}(\theta) &\doteq \sum_{t=1}^T \mathcal{L}_t^{\lambda_1}(\theta) \\ &\doteq \sum_{t=1}^T \left(\text{Var}_L(\mathbf{O}_t) + \text{Var}_L(\mathbf{H}_t) \right), \end{aligned} \quad (5.6)$$

which can be incorporated as a weighted term into Dreamer V3's policy loss:

$$\mathcal{L}^{\text{Policy}}(\theta) \doteq \mathcal{L}^{\text{Dr3}}(\theta) + \alpha \times \mathcal{L}^{\lambda_1}(\theta). \quad (5.7)$$

Algorithm 1 Maximal Lyapunov Exponent Regularisation**Require:**

Policy ($\pi_\theta : \mathcal{H} \times \mathcal{Z} \rightarrow \mathcal{P}(\mathcal{A})$)
 Encoder ($q_\phi : \mathcal{O} \times \mathcal{H} \rightarrow \mathcal{P}(\mathcal{Z})$)
 Decoder ($p_\phi : \mathcal{H} \times \mathcal{Z} \rightarrow \mathcal{P}(\mathcal{O})$)
 Dynamics Predictor ($f_\phi : \mathcal{H} \rightarrow \mathcal{P}(\mathcal{Z})$)
 Sequence Model ($f_\phi : \mathcal{H} \times \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{H}$)

 Current Observation ($\mathbf{o} \in \mathcal{O}$)
 Current Hidden State ($\mathbf{h} \in \mathcal{H}$)
 Time Horizon ($T \in \mathbb{N}$)

Ensure: $\mathcal{L}^{\lambda_1}(\theta)$

$\mathcal{L}^{\lambda_1}(\theta) \leftarrow 0$ ▷ Initialise the MLE regularisation loss
 $\mathbf{Z} = \{\mathbf{z}_l \sim q_\phi(\mathbf{o}, \mathbf{h})\}_{l=1}^L$ ▷ Update stochastic representations
for $t = 1, 2, \dots, T$ **do**
 $\mathbf{A} \leftarrow \{\mathbf{a}_l \sim \pi_\theta(\mathbf{h}_l, \mathbf{z}_l)\}_{l=1}^L$ ▷ Generate a set of sample actions
 $\mathbf{H} \leftarrow \{\mathbf{h}_l = f_\phi(\mathbf{h}_l, \mathbf{z}_l, \mathbf{a}_l)\}_{l=1}^L$ ▷ Update hidden representations
 $\mathbf{Z} \leftarrow \{\mathbf{z}_l \sim q_\phi(\mathbf{h}_l)\}_{l=1}^L$ ▷ Update stochastic representations
 $\mathbf{O} \leftarrow \{\mathbf{o}_l \sim p_\phi(\mathbf{h}_l, \mathbf{z}_l)\}_{l=1}^L$ ▷ Generate a set of predicted states
 $\mathcal{L}^{\lambda_1}(\theta) \leftarrow \mathcal{L}^{\lambda_1}(\theta) + \text{Var}(\mathbf{O}) + \text{Var}(\mathbf{H})$ ▷ Update the MLE regularisation loss
end for
return $\mathcal{L}^{\lambda_1}(\theta)$

Including this regularisation term as an additional weighted term in the policy loss forces agents to consider the predictability of the system during the optimisation process. This promotes policies that produce predictable state trajectories which attain high rewards, instead of solely optimising the expected return. The complete algorithm for calculating the MLE regularisation term is provided in Algorithm 1.

5.2 Results

In this section, we investigate the impact that the proposed MLE regularisation has on Dreamer V3. Specifically, we examine whether constraining the Maximal Lyapunov Exponent reduces the system's sensitivity to initial conditions and how this affects overall performance. Through this analysis, we aim to determine whether MLE regularisation provides a principled approach to mitigating chaos in model-based RL while preserving task performance.

5.2.1 Experimental Setup

To assess the effect of the proposed regularisation, we train three instances of Dreamer V3 with MLE regularisation using $\alpha \in \{0, 0.5, 1, 1.5, 2\}$. When estimating state divergence, the RSSM predicts $L = 3$ plausible future trajectories over which the state and internal representation variance is measured. Increasing L will produce a more accurate estimate of state divergence at the cost of increased computational resources. Therefore, to maintain a similar training time to that of Dreamer V3, we set $L = 3$. All other hyperparameters are consistent with the Dreamer V3 baseline outlined in Appendix A.1.

5.2.2 Predictability

To evaluate the influence of MLE regularisation on the predictability of Dreamer V3, we measure the Maximal Lyapunov Exponents and the Sum of Lyapunov Exponents produced by each policy. The Lyapunov Exponents are estimated using the methodology outlined in Section 4.3.1, with Figure 5.2 providing the summary statistic for each metric. These results indicate that MLE regularisation directly influences the predictability of the control interaction, as increasing α decreases the MLE in all environments. Furthermore, Figure 5.3 demonstrates the effect of this improved predictability as it shows the partial state trajectories produced by the *Walker Walk* when controlled by the unregularised ($\alpha = 0$) and regularised ($\alpha = 1$) policies. This indicates that the state trajectories diverge less when MLE regularisation is applied, as the control interaction is more predictable.

This improvement in predictability is most pronounced in the *Walker Stand* environment, as even small weighting coefficients substantially decrease the Maximal Lyapunov Exponent. To identify the source of this improvement in predictability, Figure 5.4 provides the recurrence plot for the unregularised ($\alpha = 0$) and regularised ($\alpha = 1$) policies. This indicates that the standard Dreamer V3 policy does not stabilise the system to a fixed point, as there is consistently a significant distance between the sample trajectory and its delayed version. In contrast, when the regularised policy controls this system, the distance is near zero after 100 steps, indicating that the system stabilises at a fixed point.

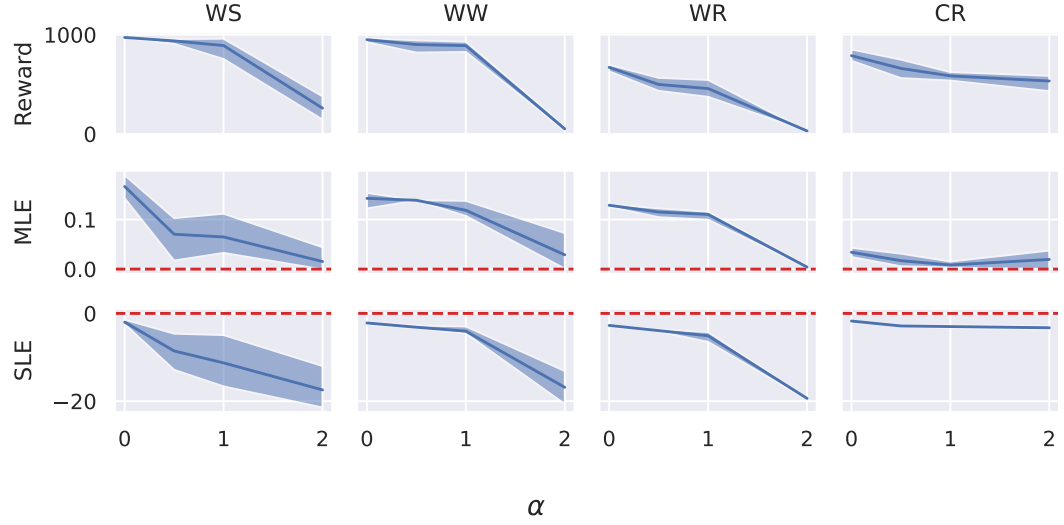


Figure 5.2: Total Reward, Estimated Maximal Lyapunov Exponent (MLE), and Sum of Lyapunov Exponents (SLE) for the *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments when controlled by a trained instance of Dreamer V3 with varying levels of MLE regularisation. Each policy–environment combination is independently trained with three random seeds. The total reward is evaluated over 80 evaluation episodes, each with a fixed length of 1000 steps, and the interquartile MLE and SLE for each seed are estimated using 20 initial states. A bootstrapped 95% confidence interval is included to show the variation in MLE and SLE across random seeds.

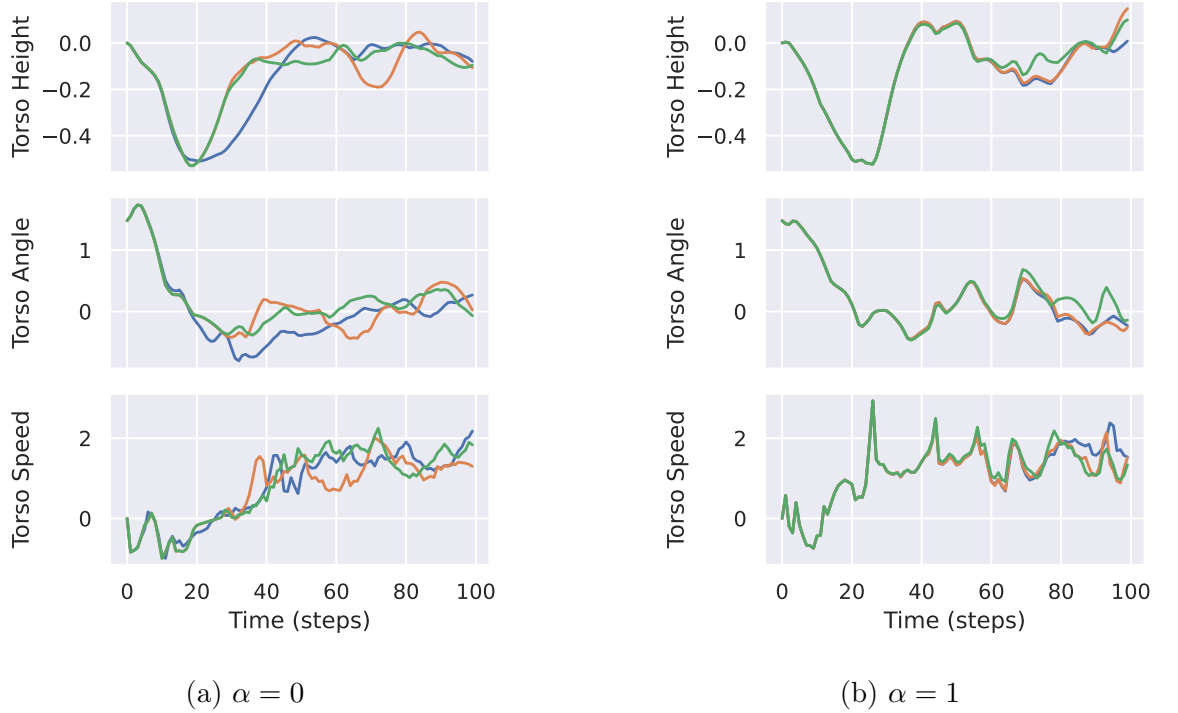


Figure 5.3: Partial state trajectories produced by Dreamer V3 and Dreamer V3 + MLE regularisation when controlling the *Walker Walk* environment.

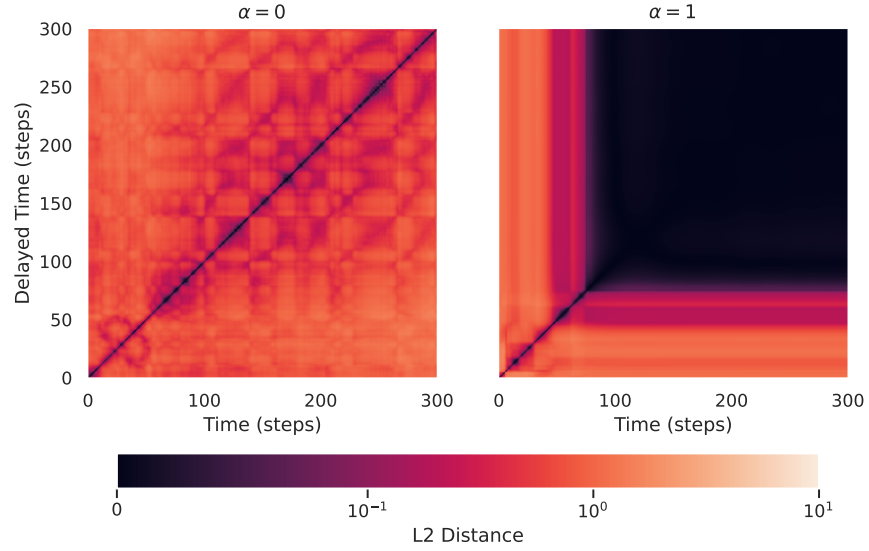


Figure 5.4: Recurrence plot produced when Dreamer V3 (left) and Dreamer V3 + MLE regularisation (right) control the *Walker Stand* environment.

5.2.3 Performance

To evaluate the influence of MLE regularisation on the performance of Dreamer V3, we measure the total episodic reward produced by each control system. Specifically, we measure this metric over 80 unique POMDP episodes, as this provides an accurate representation of the total reward distribution without incurring significant computational overhead. Figure 5.2 provides the summary statistic for this empirical distribution and indicates that the improvement in predictability comes at the cost of overall performance as increasing α decreases the total reward.

Furthermore, Figure 5.5 provides a render of the physical motion of the *Walker Walk* environment when controlled by two independently trained instances of Dreamer V3 with MLE regularisation weighted at $\alpha = 2$. From this, we can identify that both systems approximate qualitatively good behaviour but are unable to maintain the speed necessary to achieve high rewards. This indicates that enforcing lower MLE values can overly constrain the policy into producing more predictable low-reward state trajectories.

Overall, these results suggest that MLE regularisation reduces the Maximal Lyapunov Exponent across environments, thereby improving the predictability of Dreamer V3 policies. However, this gain in predictability is offset by a reduction in task performance, especially under strong regularisation. For most environments, setting $\alpha = 1$ provides a favourable compromise, yielding lower state divergence without severely degrading episodic return.

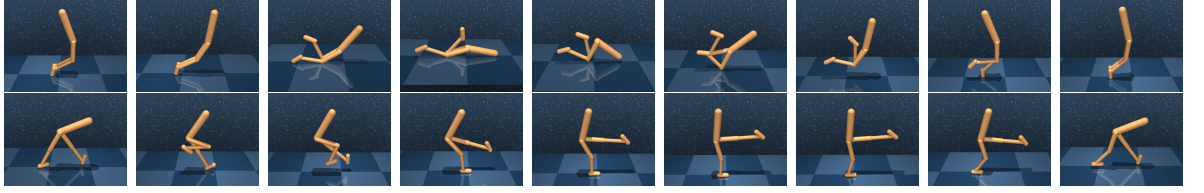


Figure 5.5: Environment render of the *Walker Walk* environment when controlled by independent instances of Dreamer V3 with MLE regularisation and weighting coefficient of 2. These systems move right in a more predictable manner than standard Dreamer V3, but not fast enough to attain high rewards.

5.2.4 Robust Control

Finally, to identify how robust these regularised Dreamer V3 policies are to continual state perturbations, we measure their performance when Gaussian noise is added to each observation. Each policy is trained without observation noise ($\sigma = 0$) in the fully deterministic variant of each environment and with $\alpha = 1$. When evaluating, Gaussian noise ($\mu = 0$ and $\sigma \in [0, 1]$) is applied at each time step, and performance is measured over 240 episodes with the standard unroll length of 1000 steps (1 POMDP episode). Further increasing the number of episodes used for this will provide a better empirical estimation of the average total reward when Gaussian noise is applied. However, as this requires an increased computation expense, we limit our experiment to 240 episodes.

Figure 5.6 shows the interquartile mean and bootstrapped 95% confidence interval for the total reward attained by each policy–environment interaction subject to various levels of Gaussian noise. This shows that MLE regularisation significantly improves the performance of Dreamer V3 in four of the noisy control systems, while the other two environments (*Walker Stand*, *Cheetah Run*) attain similar performance to the Dreamer V3 baseline. Furthermore, examining the state trajectories produced by Dreamer V3 and Dreamer V3 + MLE regularisation when controlling the *Walker Stand* task with Gaussian observation noise (Figure 5.7) shows that the regularisation improves the predictability of the control interaction as the trajectories do not diverge significantly. These findings indicate that MLE regularisation can improve the robustness of Dreamer V3 subject to observation noise, as it produces more consistent and reliable behaviour in the face of uncertainties.

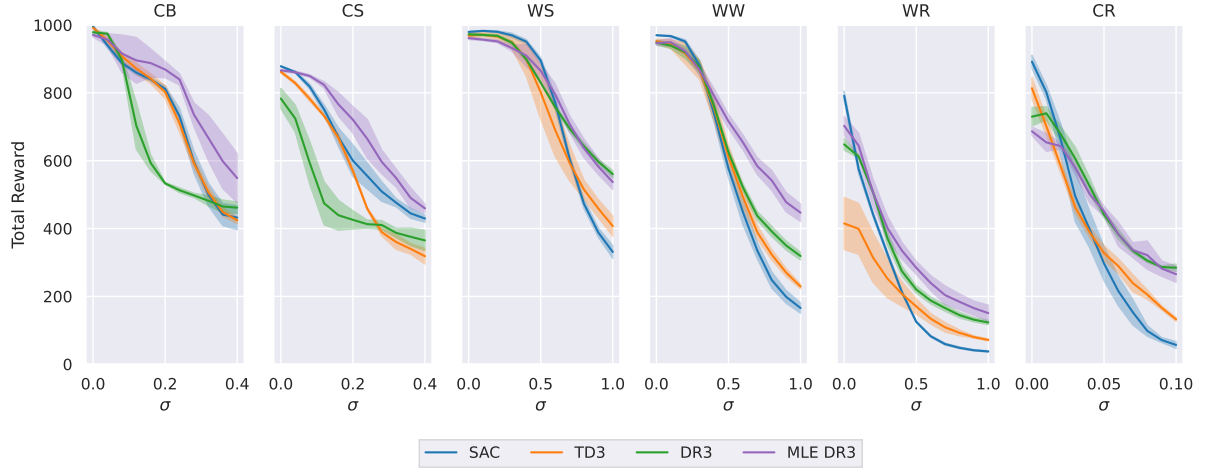


Figure 5.6: Total episode reward for the *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR), and *Cheetah Run* (CR) environments when controlled by trained instances of SAC, TD3, Dreamer V3 (DR3), and Dreamer V3 with MLE regularisation (MLE DR3) subject to $\mathcal{N}(0, \sigma)$ Gaussian observation noise. Each policy–environment combination is independently trained with three random seeds, and the average episode reward with a bootstrapped 95% confidence interval is reported over 240 evaluation episodes, each with a fixed length of 1000 steps.

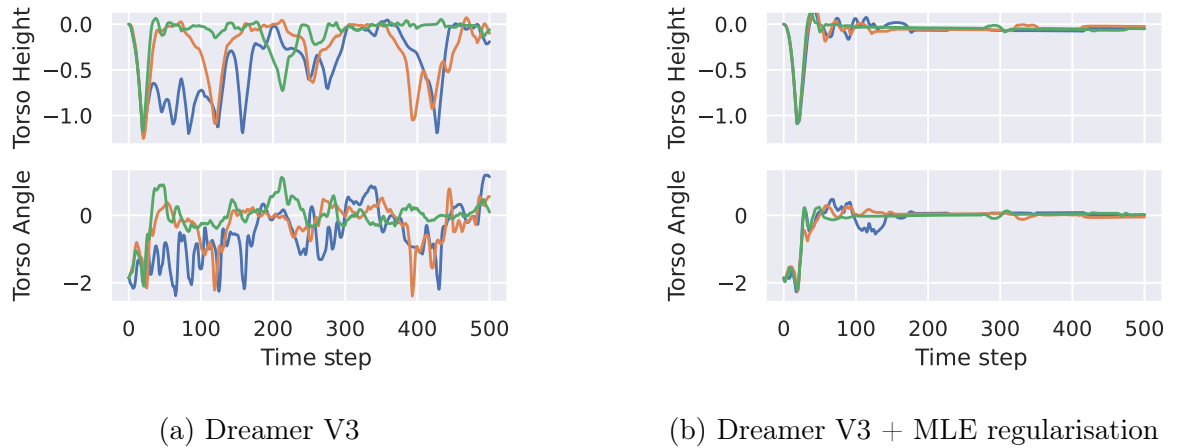


Figure 5.7: State trajectories produced when Dreamer V3 and Dreamer V3 + MLE regularisation control the *Walker Stand* environment with Gaussian observation noise ($\mathcal{N}(0, 0.5)$).

5.3 Discussion

In this chapter, we investigated whether constraining the Maximal Lyapunov Exponent during policy updates could improve the predictability of deep RL controllers. To this end, we proposed Maximal Lyapunov Exponent regularisation for Dreamer V3. This method uses the Recurrent State Space Model to generate a differentiable estimate of local state divergence under the current policy. We hypothesised that minimising this prediction during training would reduce the policy’s sensitivity to initial conditions, thereby producing more predictable control interaction. By embedding these predictability constraints directly into the policy loss, we aimed to determine whether this approach provides a principled mechanism for mitigating chaotic state dynamics in model-based reinforcement learning.

Overall, our empirical results validate this hypothesis as the inclusion of MLE regularisation produces policies which generate significantly lower MLE. However, this additional loss also introduces a direct trade-off between performance and predictability, induced by the regularisation weighting coefficient. Underweighting the term results in high-performing controllers that remain highly sensitive to initial conditions, much like the unregularised Dreamer V3. Meanwhile, introducing a moderate level of regularisation produces policies that maintain strong task performance while significantly improving predictability. However, assigning a high weight to the MLE regularisation term over-constrains the optimisation process, thus producing highly predictable but low-performing control systems.

We argue that the performance degradation is a consequence of the multi-objective optimisation problem introduced by the regulariser. Specifically, the policy parameters are subject to two potentially conflicting objectives, where $\mathcal{L}^{Dr3}$ maximises the task-specific return while $\mathcal{L}^{\lambda_1}$ improves the predictability of the control interaction. In complex control tasks, these gradients do not necessarily align, as high-reward regions frequently reside in unpredictable areas of the state space. For example, the *Walker Run* and *Cheetah Run* environments provide high rewards when the robot moves at high velocity. However, these high-energy manoeuvres often transition the system into regions of the state space that are inherently less predictable. Consequently, these objectives are fundamentally misaligned as they optimise for conflicting behaviours.

5.4 Limitations

While MLE regularisation provides an effective tool for improving predictability, several limitations remain that restrict its applicability in certain domains. Firstly, the method introduces a significant tuning burden via the weighting coefficient α . As demonstrated in our ablation studies, this hyperparameter must be carefully chosen to balance the multi-objective optimisation problem. An inappropriate weight can lead to either continued unpredictability or a catastrophic loss in performance, making the training process sensitive to initial parameter selection.

Secondly, the effectiveness of this approach is severely impacted by the accuracy of the Recurrent State Space Model. Because the predictability gradient is derived from the model’s own predictions, the regularisation is only as reliable as the model’s estimation of state divergence. If the RSSM fails to capture the true non-linear dynamics of the environment accurately, the policy may fail to reduce true MLE. This issue is particularly limiting in chaotic control systems, as these have inherently unpredictable long-term state dynamics. In effect, using a predictive model to optimise predictability in an unpredictable control system creates a paradoxical dependency.

Finally, a key limitation of this approach is that it is fundamentally underspecified. While the regulariser constrains all dimensions of the observation space by penalising variance, it does not define a specific target for the system to follow. It acts as a directionless penalty on divergence, pushing the agent away from unpredictable states without providing a high-performing reference to converge toward. This lack of a specific target trajectory often leads the agent to adopt overly conservative, low-energy behaviours simply to minimise the variance penalty. This suggests that a targeted approach is required to improve the predictability of the control interaction by explicitly defining a high-performing state trajectory for the system to follow.

Chapter 6

Minimising Chaos with Reward

In Chapter 5, we demonstrated that the chaotic state dynamics induced by deep reinforcement learning policies in continuous control tasks can be reduced using Maximal Lyapunov Exponent regularisation. However, this method introduces a trade-off between task performance and predictability, as these objectives don't necessarily align. In this chapter, we adopt a different approach and investigate whether the reward function alone can be used to improve predictability. To this end, we make the following two observations.

First, control systems with identical transition dynamics can exhibit significantly different predictability characteristics depending on the reward signal used to train the policy. For example, the *Stand*, *Walk*, and *Run* variations of the *Walker* environment generate distinct Maximal Lyapunov Exponents despite sharing the same transition dynamics. The only distinguishing factor between these tasks is the reward function used to train the policy.

Second, predictable behaviour only emerges when the reward function constrains all the dimensions of the system's state (*Pointmass*, *Cartpole*). In contrast, reward functions that only constrain a subset of the state dimensions (*Walker*, *Cheetah*) stabilise the system in the constrained dimensions but overall produce chaotic state dynamics. This can clearly be seen when examining the state dynamics produced by *Walker Stand* when controlled by SAC (Figure 6.1). As the environment provides a high reward when the torso is high and vertical, the policy implicitly learns to constrain these dimensions during policy updates. In contrast, the remaining state dimensions do not influence the reward function, and thus the agent does not directly consider the stability of these dimensions during policy updates.

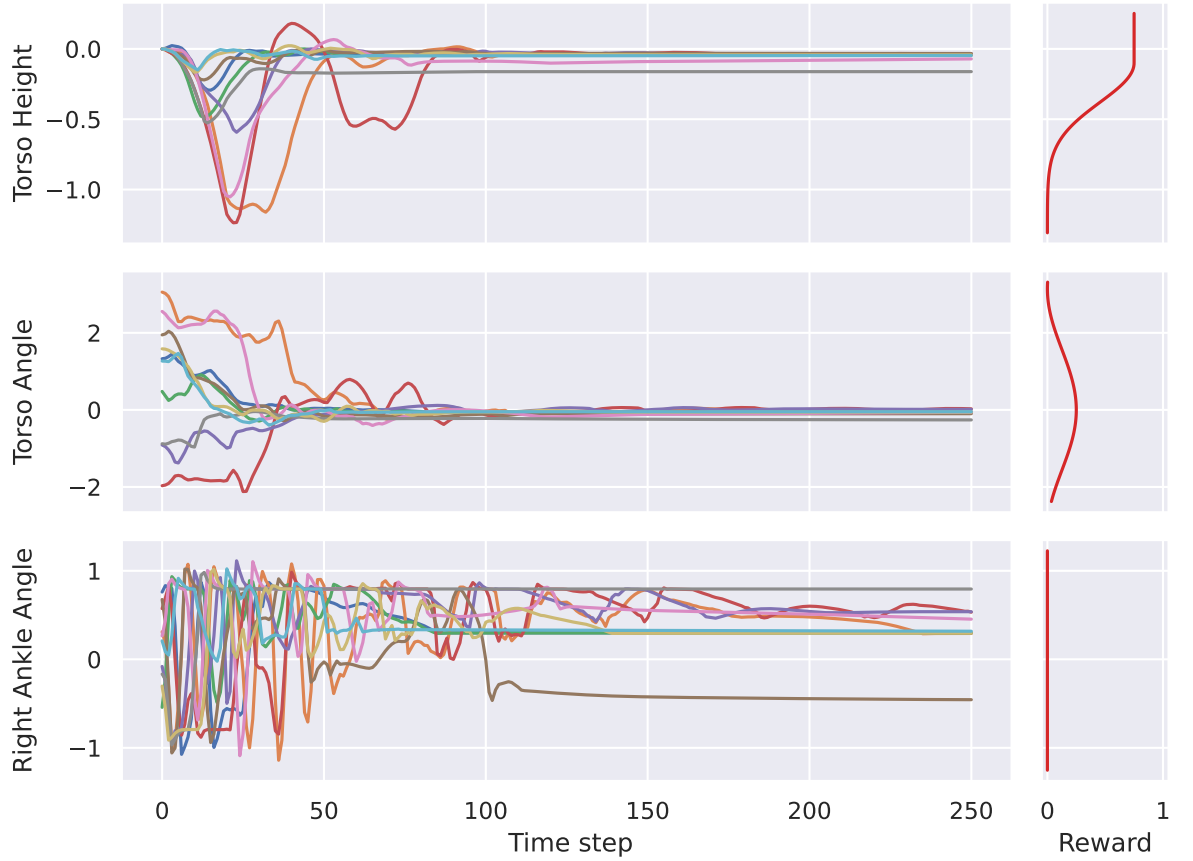


Figure 6.1: Ten partial state trajectories produced by the *Walker Stand* environment when controlled by SAC (left) and the per-step individual components of the reward function used to train the policy (right). The policy learns to constrain the dimensions which influence the reward (Torso Height and Angle) but does not stabilise the remaining dimensions.

Together, these observations reveal a fundamental limitation of reward-driven learning: the reward function not only defines the objective but also implicitly determines which parts of the state space the policy learns to stabilise. If the reward signal encompasses all degrees of freedom, the resulting policy can achieve globally stable dynamics. However, if any dimensions are omitted, the policy receives no corrective feedback in those axes, allowing unpredictability to persist.

In practice, however, constructing such fully constrained reward functions is often infeasible. In many tasks, the desired system dynamics are not known in advance, making it difficult to determine how the reward function should constrain the state space. As a result, simplified task-specific reward functions are commonly used to focus learning on key behavioural features (e.g., torso position). While this underspecificity is necessary for systems with unknown optimal dynamics, it introduces a critical shortcoming as the reward function only constrains a subset of the state space.

To address this dilemma, we propose **Global** stabilisation via **Intrinsic Fine Tuning** (GIFT), a lightweight framework that improves the global stability of deep RL policies. The approach begins by training a policy using a standard task-specific reward function, allowing it to acquire competent behaviour. This pre-trained policy is then used to construct an intrinsic reward function that constrains all dimensions of the observation space. The pre-trained policy is then fine-tuned with the intrinsic reward, thereby incentivising convergence across all observation dimensions. In effect, this encourages the policy to stabilise all dimensions of the observation space.

We demonstrate that applying GIFT to state-of-the-art deep RL policies in continuous control domains consistently produces significant improvements in predictability. Specifically, the resulting policies exhibit a Maximal Lyapunov Exponent that is an order of magnitude lower than the original policies, and this improvement in predictability does not compromise task performance. This makes GIFT a practical and broadly applicable tool for stabilising any RL agent in any continuous control domain.

6.1 Methods

In this section, we present the three sequentially executed processes which define GIFT (Algorithm 2). Specifically, in Section 6.1.1 we outline how a policy is trained to maximise the total reward in a given POMDP. In Section 6.1.2, we then define how to generate a task-agnostic intrinsic reward function which incentivises global stability. Finally, in Section 6.1.3 we describe how the original policy is fine-tuned with the intrinsic reward.

6.1.1 Pre-training for Performance

In the first stage of GIFT, a policy (π_θ) is trained in the POMDP with the goal of maximising the total episodic reward. This training can be conducted using any deep RL method and no modifications related to stability are introduced at this stage. The objective is to learn a policy that can achieve good performance in the original task.

6.1.2 Generating the Intrinsic Reward

In the second stage of GIFT, an intrinsic reward function ($\hat{R} : \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R}$) is constructed to promote convergence to a learned high-performing reference observation trajectory ($\mathbf{T}$). This trajectory acts as both a spatial and temporal template for successful behaviour, and the intrinsic reward encourages the agent to stay within a high-reward region and progress sequentially through it.

To construct $\mathbf{T}$, the pre-trained task-optimised policy is first used to generate a dataset of $L \in \mathbb{N}$ observation trajectories and reward trajectories, each of length $N \in \mathbb{N}$. These trajectories are obtained by rolling out the standard POMDP under varying initial conditions and the observation trajectory that attains the highest total reward is selected as the reference trajectory ($\mathbf{T} \in N \times \mathcal{O}$). Consequently, this sequence exemplifies optimal behaviour across all observation dimensions, not just those covered by the environment's reward function.

Given a sample observation transition $(\mathbf{o}_t, \mathbf{o}_{t+1})$ and reference trajectory $\mathbf{T}$, the intrinsic reward function is then defined as:

$$\hat{R}(\mathbf{o}_t, \mathbf{o}_{t+1}) = \frac{1}{(\kappa \cdot \|\mathbf{o}_{t+1} - \boldsymbol{\tau}(\mathbf{o}_t)\|_2)^2 + 1} \quad (6.1)$$

where $\kappa \in \mathbb{R}$ controls the rate of decay of the reward with distance and the target observation ($\boldsymbol{\tau}(\mathbf{o}_t)$) is defined as the immediate successor of the observation in $\mathbf{T}$ that is closest to the current observation $\mathbf{o}_t$:

$$\boldsymbol{\tau}(\mathbf{o}_t) = \mathbf{T} \left[1 + \arg \min_{n \in [0, N-2]} (\|\mathbf{T}[n] - \mathbf{o}_t\|_2) \right]. \quad (6.2)$$

This construction ensures that the policy is rewarded for progressing along the high-reward trajectory rather than merely remaining near it, as targeting the next step in $\mathbf{T}$ encourages forward motion through the sequence. Furthermore, unlike standard task-specific rewards, which typically constrain only a subset of observation dimensions, this intrinsic reward function provides corrective feedback across the entire observation space. As a result, it mitigates the underspecification problem by encouraging the policy to regulate all observable degrees of freedom, thereby improving the stability of the control interaction.

Algorithm 2 Global stabilisation via Intrinsic Fine Tuning

Require:

Policy (π_θ)
 Partially Observable Markov Decision Process (POMDP)
 Learning Method (e.g. SAC / PPO)

Ensure: π_θ

Pre-training for Performance
 Train π_θ in the POMDP using the Learning Method

Generating the Intrinsic Reward
 Unroll the POMDP for N steps using π_θ starting in L different initial positions
 Set T as the observation trajectory which corresponds to the highest total reward
 Generate the intrinsic reward function

Fine-tuning for Predictability
 Fine-tune π_θ with the intrinsic reward using the Learning Method

return π_θ

6.1.3 Fine-tuning for Predictability

In the final stage of GIFT, the pre-trained policy is fine-tuned with the intrinsic reward in order to achieve predictable behaviour. This fine-tuning is conducted by continuing to update the existing policy using any standard deep RL algorithm. Importantly, at this stage, the task-specific reward is no longer used and the policy is now optimised solely with respect to the intrinsic reward function. As a result, the fine-tuned policy is incentivised to control the observation dimensions that were not used for the original reward function.

6.2 Results

In this section, we evaluate the effectiveness of fine-tuning deep reinforcement learning policies using GIFT. The aim is to determine whether this additional training phase consistently enhances the predictability of control interactions without compromising system performance. Section 6.2.1 outlines our experimental setup, while Section 6.2.2 details how GIFT influences the predictability of the control interaction. Finally, Section 6.2.3 shows how the change of predictability induced by GIFT impacts the performance of the control interaction.

6.2.1 Experimental Setup

We apply GIFT to Soft Actor-Critic (SAC) [Haarnoja et al., 2018] and Proximal Policy Optimization (PPO) [Schulman et al., 2017] as these represent the state-of-the-art off-policy and on-policy deep RL methods. Each agent is trained in the *Walker* and *Humanoid* environments sampled from the MuJoCo Playground [Zakka et al., 2025] variation of the DeepMind Control Suite [Tunyasuvunakool et al., 2020]. These differentiable environments represent complex, continuous control, multi-body systems with non-linear contact forces, which can produce chaotic state dynamics.

Each policy–environment pair is independently trained with three initial seeds using an Intel Core i7-8700 CPU workstation, with an Nvidia RTX 2080 Ti GPU, and 32GB of RAM. After training the baseline policy, GIFT unrolls the POMDP for 1 full episode ($N = 1000$ steps) under the deterministic version of this policy, starting in $L = 128$ unique initial states. While increasing the initial state count would yield a more diverse set of target trajectories, we restrict the sampling to $L = 128$ to account for the high computational cost of the unrolling process. From these sampled trajectories, we select the observation trajectory associated with the highest cumulative reward to serve as the reference trajectory. Finally, we use the hyperparameters outlined in Appendix A.1 for both training phases and use $\kappa = \sqrt{17/3}$ when calculating the intrinsic reward, as this configuration was found to produce the best performance during hyperparameter optimisation.

6.2.2 Predictability

To assess GIFT’s impact on the predictability of the control interaction, we measure the Maximal Lyapunov Exponent (MLE) for each policy–environment pair. As these systems are differentiable, we compute the MLE using the Jacobian method (Equation 6.3), starting from 100 distinct initial states and unrolling for 1 POMDP episode ($T = 1000$ steps). While larger sample sizes would refine the MLE estimate, we restrict the evaluation to $L = 100$ initial states to balance statistical precision against the computational overhead.

$$\lambda_i \approx \frac{1}{T} \ln \left(\sigma_i \left(\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_0} \right) \right) \quad (6.3)$$

Table 6.1 provides the interquartile average MLE for each environment along with a bootstrapped 95% confidence interval to capture variability due to random seeds. These results demonstrate that GIFT substantially improves the predictability of control policies across all systems, as incorporating GIFT reduces the MLE by approximately one order of magnitude for both model types. This indicates that these systems produce more consistent state trajectories under small state perturbations.

Figure 6.2 further illustrates this effect as it shows a subset of the state trajectories produced by *Humanoid Walk* when controlled by SAC and SAC + GIFT, subject to an initial state perturbation of 10^{-4} units. This demonstrates that when maximising reward alone, SAC induces chaotic state dynamics, as this small change in system state compounds over time to produce substantially different long-term outcomes. In contrast, the SAC + GIFT policy induces predictable dynamics, as the perturbed trajectories do not diverge in the long term.

Furthermore, to examine the temporal structure of these state trajectories, we compute a recurrence plot for a state trajectory produced by SAC and SAC + GIFT when controlling the *Humanoid Run* environment (Figure 6.3). These figures visualise the distance between the trajectory and a delayed version of itself and can be used to highlight repeating patterns in the system’s dynamics. We find that when the system is controlled by SAC, the recurrence plot contains unstructured patterns, which is consistent with the chaotic nature of this control interaction. In contrast, SAC + GIFT generates consistent diagonal lines, indicating that the state trajectory repeats at regular intervals. This provides further evidence that GIFT not only improves local predictability but also imposes temporal consistency of the system’s evolution.

Table 6.1: Average Maximal Lyapunov Exponent produced by environments sampled from the DeepMind Control Suite when controlled by trained deep RL policies. A bootstrapped 95% confidence interval is included to show the variation introduced by initial seeds. Lower values indicate a more predictable control interaction.

Environment	SAC	SAC + GIFT	PPO	PPO + GIFT
<i>Walker Stand</i>	0.043 ± 0.032	0.004 ± 0.000	0.008 ± 0.002	0.007 ± 0.001
<i>Walker Walk</i>	0.066 ± 0.017	0.005 ± 0.000	0.046 ± 0.021	0.053 ± 0.019
<i>Walker Run</i>	0.084 ± 0.036	0.011 ± 0.002	0.107 ± 0.002	0.064 ± 0.013
<i>Humanoid Stand</i>	0.065 ± 0.022	0.006 ± 0.002	0.012 ± 0.002	0.004 ± 0.001
<i>Humanoid Walk</i>	0.192 ± 0.013	0.003 ± 0.000	0.012 ± 0.004	0.005 ± 0.002
<i>Humanoid Run</i>	0.206 ± 0.037	0.004 ± 0.001	0.010 ± 0.001	0.006 ± 0.004

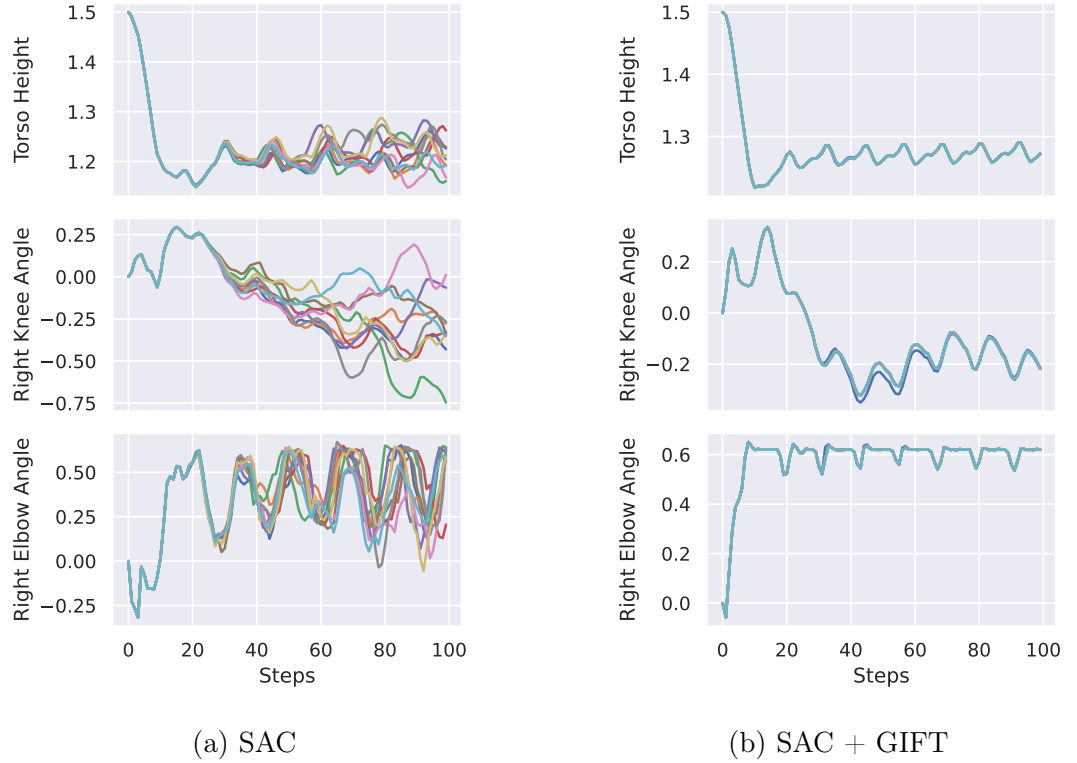


Figure 6.2: Ten partial state trajectories produced by SAC and SAC + GIFT when controlling the *Humanoid Walk* environment subject to a single initial state perturbation of size 10^{-4} . The standard *Humanoid Walk* reward function only constrains the torso height, angle and speed, while the GIFT reward function constrains all dimensions.

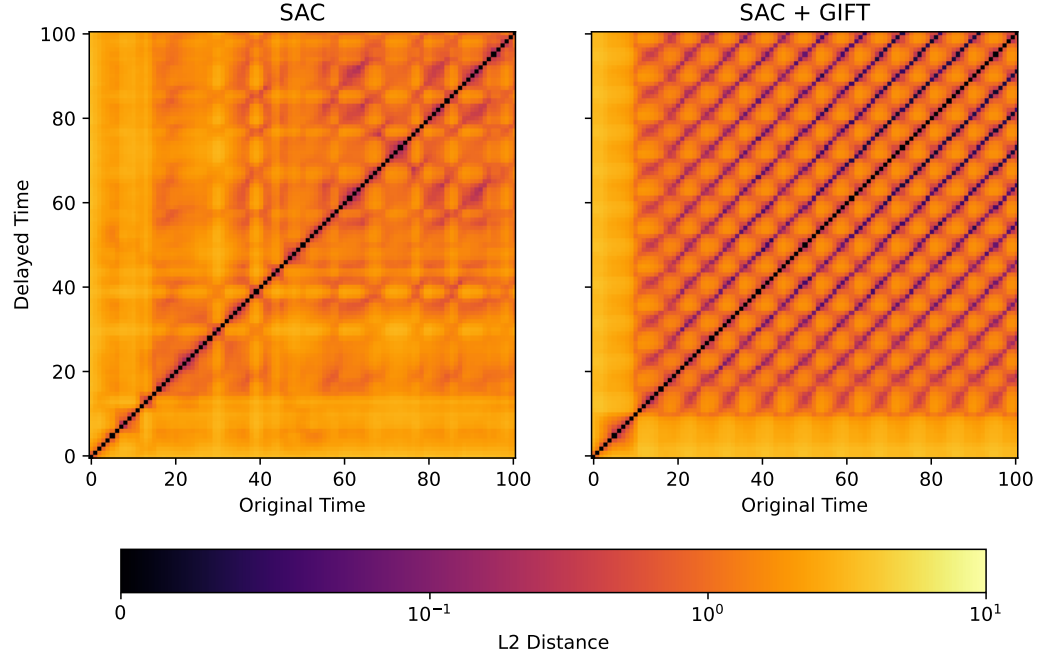
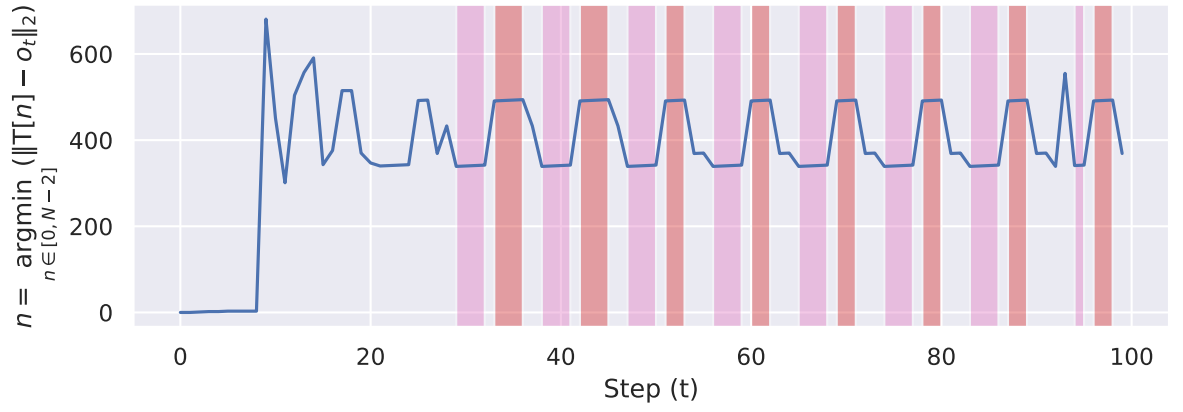


Figure 6.3: Recurrence plot showing the distance between a state trajectory and a delayed embedding of itself produced when SAC and SAC + GIFT control the *Humanoid Run* environment. SAC + GIFT produces several off-diagonal lines, indicating that this model produces periodic state dynamics.

Finally, to evaluate how the system progresses relative to the reference trajectory, we analyse the temporal alignment between the generated observation trajectory ($\mathbf{o}_t$) and the target trajectory ($\mathbf{T}$). For each time step, we identify the index of the nearest point on the reference trajectory in observation space ($\arg \min_{n \in [0, N-2]} \|\mathbf{T}[n] - \mathbf{o}_t\|_2$). As shown in Figure 6.4, the SAC + GIFT policy produces a regular cyclical pattern, where the system consistently transitions between two distinct regions of the trajectory. This pattern suggests that the system has stabilised into a periodic regime, repeatedly following multiple segments of the reference trajectory.

Overall, these results demonstrate that GIFT consistently induces predictable state dynamics across high-dimensional control tasks. We demonstrate that predictability can emerge not from altering the model architecture or training procedure, but from reshaping the objective to constrain all observable dimensions. The findings therefore support the chapter’s central claim that deep reinforcement learning policies generate predictable dynamics when the reward function provides feedback across the full observation space, rather than concentrating solely on task-specific outputs.



(a) Time index of the target trajectory closest to the observation trajectory

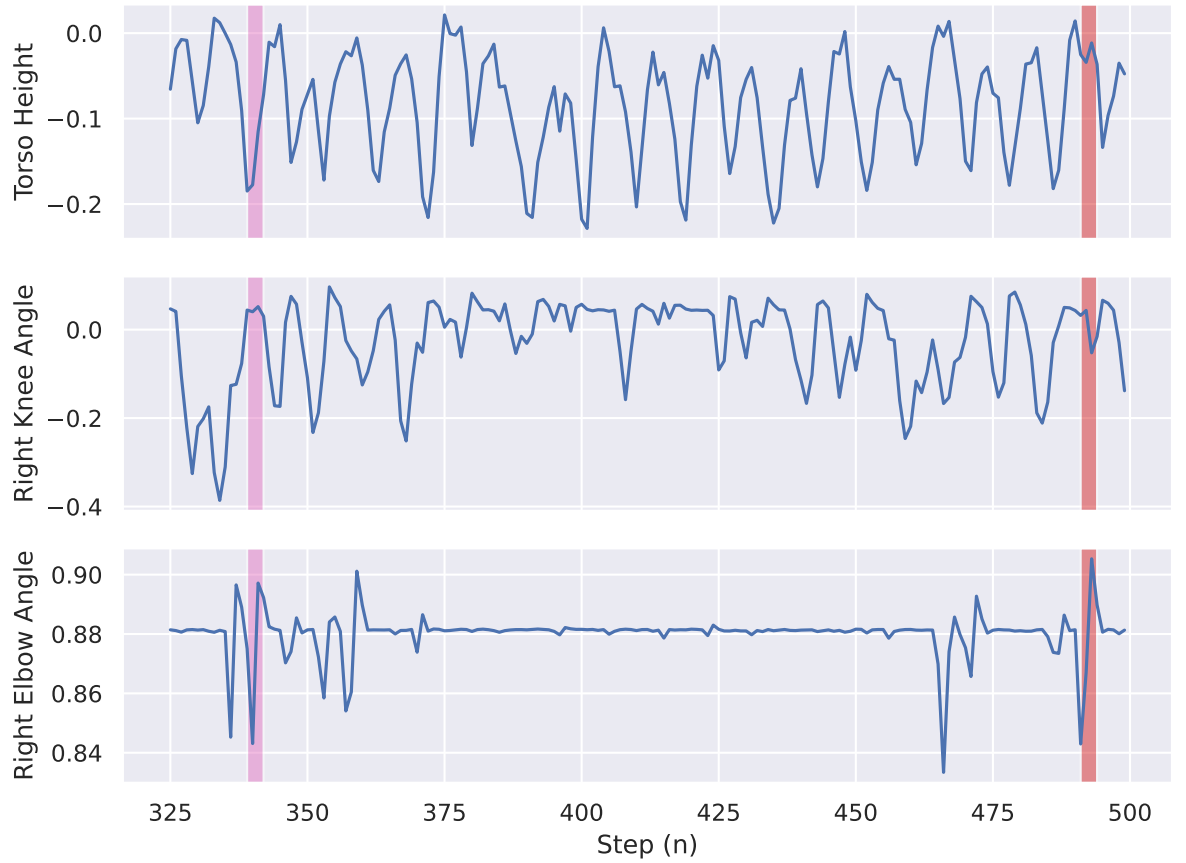
(b) Partial target observation trajectory for the *Humanoid Run* environment.

Figure 6.4: (a) Time index of the target observation trajectory which is closest to observation trajectory ($\mathbf{o}_t$). The system linearly progresses along two regions of the target trajectory (red and pink) while periodically switching. (b) Partial target observation trajectory produced by SAC + GIFT when controlling the *Humanoid Run* environment. The coloured bands correspond to the two stable nodes to which the policy converges.

6.2.3 Performance

To assess GIFT’s impact on the performance of the control interaction, we measure the total reward for each policy–environment pair. We compute this over 1000 steps starting from 100 distinct initial states using the deterministic variant of the policy. For each environment, we report the interquartile mean of the total reward in Table 6.2, along with a bootstrapped 95% confidence interval to capture variability due to random seeds during training. These results indicate that GIFT does not adversely affect the average performance of the control interaction, as incorporating GIFT preserves or modestly improves the average total reward across the majority of environments. This is especially apparent when SAC controls the *Humanoid* environments, as policies fine-tuned with GIFT produce on average a 9% increase in total reward. However, we also find that GIFT can increase the variability in performance between differing seeds, as indicated by the wider confidence intervals in the *Run* tasks. This occurs because the reference trajectories for each seed attain differing total rewards, so convergence towards these seed-specific trajectories leads to policies with varying final performance.

Furthermore, to understand the influence of the initial state distribution, we inspect the distribution of total rewards for each individual seed in Figure 6.5. These results indicate that GIFT significantly reduces the variation in total rewards subject to different initial configurations. This is particularly noticeable in the *Humanoid* environments when controlled by SAC as the increased predictability significantly improves the consistency of the policy. This occurs because the unstable dynamics produced by the standard SAC policy often cause the *Humanoid* robot to fall over, while SAC + GIFT mitigates such failures (Figure 6.6). This increase in consistency across the full initial state distribution is valuable in real-world control settings, where worst-case performance is often more critical than average performance.

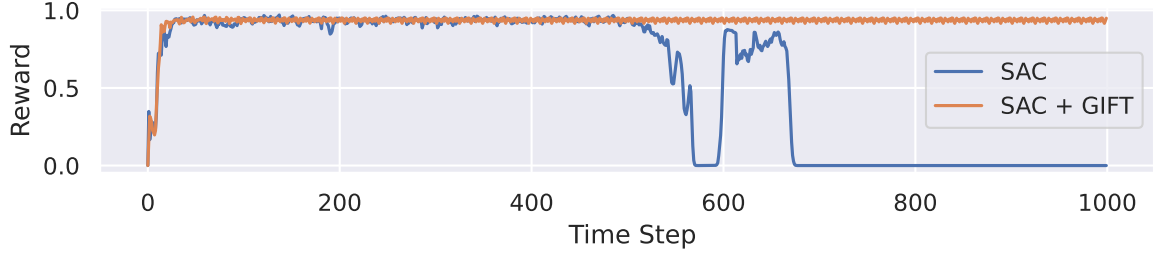
However, when GIFT is applied to PPO on the *Walker Walk* environment, we find that the variation in performance increases across different initial conditions. This may reflect the on-policy nature of PPO, which makes it more susceptible to overfitting to the specific behaviours present in the reference trajectory. Since GIFT constructs its stabilising reward from a reference trajectory, an on-policy algorithm may learn to exploit that trajectory too narrowly, thereby reducing robustness to unseen initial states. This suggests that the compatibility of GIFT with different underlying algorithms depends on their capacity to generalise beyond the training distribution.

Table 6.2: Average total reward produced by environments sampled from the DeepMind Control Suite when controlled by trained deep RL policies. A bootstrapped 95% confidence interval is included to show the variation introduced by initial seeds. Higher values indicate a better-performing control interaction.

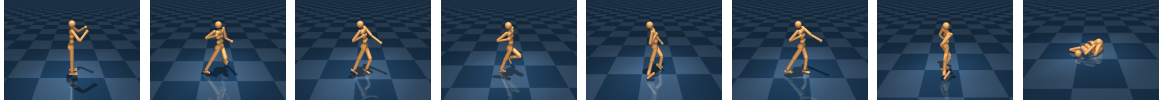
Environment	SAC	SAC + GIFT	PPO	PPO + GIFT
<i>Walker Stand</i>	985.3 ± 1.2	986.7 ± 1.1	959.2 ± 5.0	969.0 ± 3.2
<i>Walker Walk</i>	975.7 ± 1.1	974.4 ± 1.3	960.5 ± 3.9	903.8 ± 41.8
<i>Walker Run</i>	758.8 ± 5.5	735.1 ± 26.0	656.2 ± 133.8	634.9 ± 125.2
<i>Humanoid Stand</i>	937.1 ± 16.8	961.7 ± 11.6	83.5 ± 30.1	13.3 ± 2.6
<i>Humanoid Walk</i>	919.6 ± 12.6	939.7 ± 10.2	27.6 ± 5.6	3.7 ± 0.5
<i>Humanoid Run</i>	282.1 ± 14.2	345.5 ± 93.5	11.8 ± 2.1	7.6 ± 4.8



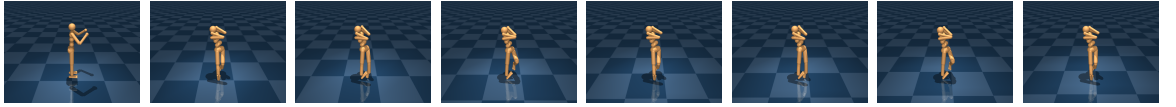
Figure 6.5: Violin plot showing the distribution of total rewards produced by each trained instance of SAC, SAC + GIFT, PPO and PPO + GIFT. Higher values indicate a better performing agent.



(a) Reward Trajectories



(b) SAC Render



(c) SAC + GIFT Render

Figure 6.6: Reward trajectory attained by SAC and SAC + GIFT when controlling the *Humanoid Walk* environment starting in the same initial position (Figure 6.6a). SAC attains a lower total reward as the unstable system dynamics cause the robot to fall over (Figure 6.6b). In contrast, when GIFT is applied to SAC, the robot remains vertical as the control interaction is more stable (Figure 6.6c).

Finally, to evaluate the local smoothness of the total reward surface, we measure the total reward attained by SAC and SAC + GIFT when controlling the *Humanoid Run* environment, subject to two orthogonal state perturbations with length varying between 0 and 10^{-4} (Figure 6.7). These results indicate that the variation in total reward produced by SAC also occurs over small changes in initial conditions. In contrast, the total reward attained by policies fine-tuned with GIFT is significantly more consistent, indicating that small changes to the initial state do not have a detrimental impact on total performance.

Together, these findings demonstrate that GIFT enhances the reliability of deep RL policies, as it smooths the reward landscape and reduces sensitivity to both large and small variations in the initial state. This behaviour is critical in real-world applications, as abrupt performance failures triggered by minor perturbations can compromise the viability of a control policy. GIFT therefore offers a principled mechanism for improving not only the average performance but also the consistency of deep RL policies in practice. Nonetheless, careful selection of the reference trajectory remains crucial, as it strongly influences the final reward.

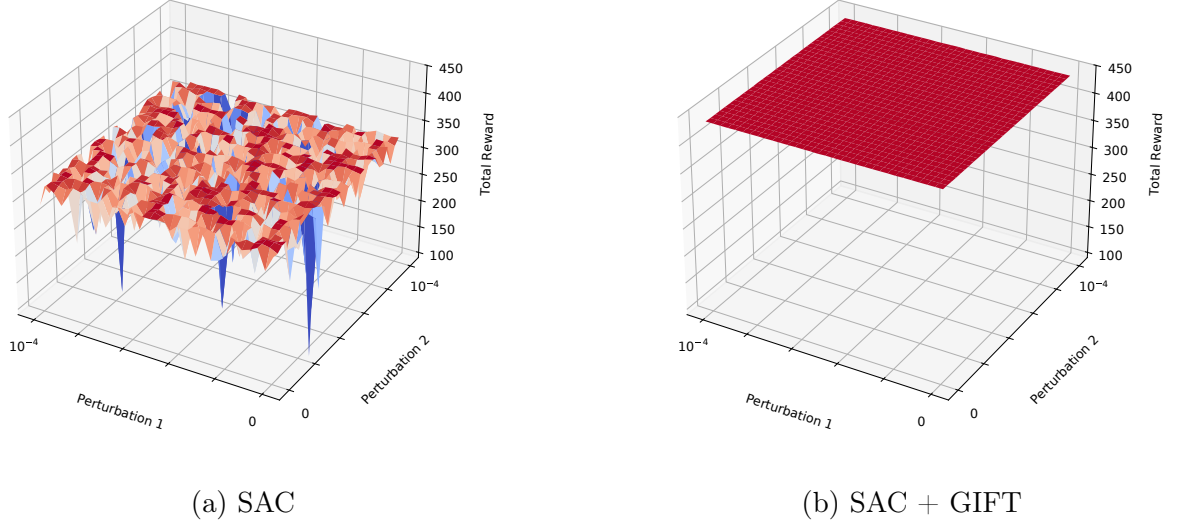


Figure 6.7: Surface plot showing how the total reward produced by SAC and SAC + GIFT varies spatially in the *Humanoid Run* environment. The initial position for each system is varied by adding a weighted sum of two orthogonal perturbation vectors with a minimum length of 0 and a maximum length of 10^{-4} .

6.3 Discussion

In this chapter, we argue that predictability in deep reinforcement learning policies is not solely determined by the choice of model architecture or transition dynamics, but is strongly shaped by the structure of the reward function. Specifically, in high-dimensional systems, the reward functions are typically fixed and defined over a limited subset of state dimensions. As a result, policies optimised purely for task performance often stabilise part of the control system while generally exhibiting chaotic dynamics.

To address this, we introduce **G**lobal stabilisation via **I**ntrinsic **F**ine **T**uning, a general-purpose lightweight framework for improving the stability of the state dynamics induced by deep reinforcement learning policies in continuous control domains. This method reframes reward design as a component that can be retrofitted to trained agents. Specifically, it constructs an intrinsic reward from a high-performing reference trajectory, which is dependent on all dimensions of the observation space. This task-agnostic reward function is used to fine-tune the original policy, thereby forcing the policy to consider stability in all observable dimensions. In effect, this shifts the optimisation objective from maximising task performance to promoting convergence to a single observation trajectory associated with high performance.

GIFT can be applied to any deep RL policy or environment, as it does not depend on specific architectural choices, task formulations or access to privileged information. It operates entirely on observed trajectories and learned policies, which makes it compatible with a wide range of continuous control settings and reinforcement learning algorithms. This flexibility is particularly valuable for real-world deployment, where constraints on model design and access to environment internals often limit the applicability of more specialised stability methods.

Our empirical results across six environments demonstrate that GIFT consistently reduces the Maximal Lyapunov Exponent by up to an order of magnitude, without compromising task performance. In the *Humanoid* environment in particular, the stability gains translated into higher total returns and significantly lower variance across differing initial configurations and seeds. These findings support the central claim of this chapter: constraining all state dimensions via an appropriately structured reward improves the predictability of learned policies.

6.4 Limitations

Despite these positive results, GIFT possesses several structural limitations that must be acknowledged. First, the fine-tuning process can generate policies that alternate between disjoint sections of the target trajectory. This produces temporal inconsistency and may degrade overall performance if the reference sections correspond to low task reward. This occurs because the intrinsic reward evaluates spatial proximity within the observation space without strictly enforcing chronological progression. However, explicitly enforcing this chronological sequence would violate the Markov assumption, as it would require the intrinsic reward function to depend on the full state history. Consequently, the presence of these temporal jumps is not just a learning flaw but an unavoidable compromise required to maintain a valid MDP.

Furthermore, as GIFT operates entirely within the observation space, its effectiveness is reduced in environments with substantial partial observability. This missing physical information creates a dichotomy in which the agent may successfully track the reference trajectory on the observable dimensions, while the unobserved dimensions can accumulate chaotic divergence. This hidden instability makes it difficult to define reliable global stabilisation targets or ensure consistent convergence across the true physical system.

Finally, relying on a single trajectory limits the suitability of the approach in diverse environments. Specifically, since GIFT incentivises movement towards a single high-performing reference trajectory, it can create a wide basin of attraction. Consequently, the fine-tuned policy may exhibit suboptimal behaviour in environments with significant diversity (e.g., a large initial state distribution) as the system attempts to converge to the target trajectory. In effect, this reliance on a single trajectory prioritises predictability over out-of-distribution generalisation.

Chapter 7

Conclusion

This chapter outlines the work presented in this thesis, which addresses the problem of chaos in deep reinforcement learning. We start by summarising the key contributions and their broader implications in Section 7.1. In Section 7.2, we then propose novel directions for future research. Finally, Section 7.3 reflects on the role of predictability within deep reinforcement learning.

7.1 Summary of Contributions

In an increasingly automated world, control policies are critical for the safe and reliable operation of complex real-world control systems. From autonomous vehicles navigating densely populated streets to robots working collaboratively in a factory, closed-loop feedback controllers must manage these systems to achieve high performance. However, performance alone is not a sufficient indicator of a successful controller as it must also maintain predictable behaviour. In practice, controllers that achieve strong average performance may still exhibit unpredictable behaviour, which is undesirable for real-world control systems. As a result, ensuring both performance and predictability remains a central challenge in the design of modern control systems.

Deep reinforcement learning provides a data-driven framework for learning control policies in complex environments. By interacting directly with a control system and optimising a reward signal, RL policies learn effective control strategies without requiring explicit models of the underlying dynamics. This makes it particularly well-suited to high-dimensional, non-linear systems, where traditional control approaches often struggle. As a result, RL has demonstrated strong empirical performance across a wide range of simulated control tasks, learning behaviours that would be difficult to design manually.

However, while RL has achieved strong results in simulated settings, its application to real-world systems remains limited. In this thesis, we argue that a key reason for this is that standard RL methods prioritise reward maximisation and do not consider the predictability of the resulting control interaction. We argue that this unpredictable behaviour poses two problems for the safe deployment of deep RL controllers in real-world environments. First, the unpredictable state dynamics create a fractal return surface which is highly sensitive to small changes in the system state. The high-frequency oscillations in this function cause a lack of robustness, as small state perturbations can produce significantly different total rewards. Second, even for high-performing policies with consistent returns, it remains impossible to accurately predict the long-term behaviour of these control systems as they rely on noisy, partially observable sensors to attain an observation. This unpredictability means that safe and reliable behaviour cannot be guaranteed, thereby imposing significant limitations on deep RL controllers in real-world control systems.

Motivated by this limitation, Chapter 4 investigated the predictability of control interactions by reframing the Markov Decision Process as a controllable dynamical system. Using the frameworks for chaos proposed by Devaney and Lyapunov, we demonstrated that state-of-the-art deep RL policies generated deterministically chaotic state dynamics when controlling high-dimensional multi-body robotic systems, as indicated by a positive Maximal Lyapunov Exponent. Consequently, these systems exhibited high sensitivity to initial conditions, as infinitesimal perturbations to the system state caused the resulting state trajectories to diverge at an exponential rate. This means that it is impossible to accurately predict the long-term state dynamics of the continuous interaction given an approximate observation. This lack of predictability poses a significant problem for real-world control systems where consistent and predictable behaviour is necessary.

Furthermore, we showed that chaotic state trajectories can also impact a policy’s performance and produce chaotic reward trajectories. Specifically, this unpredictability created a fractal return surface in which small state perturbations produced significantly different total rewards. We argued that adversarial attack methods could exploit these high-frequency oscillations by repeatedly injecting perturbations that cause the agent to follow state trajectories that attain the lowest total reward. This lack of robustness poses a significant problem for real-world control systems, where worst-case performance is often more important than average performance.

Additionally, Chapter 4 also demonstrated that the MLE generated by the control interactions is influenced by both the environment and the control policy. Specifically, we empirically showed that two environments with the same state dynamics but differing reward functions or initial state distributions generated significantly different MLEs. Additionally, we showed that the MLE varies during policy training even after reward plateaus, indicating that the predictability of the control interaction is implicitly affected by the deep RL policy. Together, these findings indicate that the level of chaos produced by the policy–environment interaction is directly influenced by the policy parameters.

Building on this finding, Chapter 5 attempted to improve the predictability of the control interaction by altering the policy objective. Specifically, we proposed Maximal Lyapunov Exponent regularisation for Dreamer V3, a novel regularisation method which penalises the predicted MLE during policy updates. This approach first uses the differentiable world model to predict several plausible future state trajectories and then reduces the variances between these predictions during policy updates. In effect, this additional term in the policy loss forces the agent to optimise its confidence in future trajectories jointly with its expectations of rewards. This shifts the optimisation focus from pure reward maximisation to a constrained search for high-performing policies which generate predictable dynamics.

Our results demonstrated that including this regularisation term significantly reduced the MLE produced by Dreamer V3. Consequently, the control interaction becomes more predictable given an approximate observation, as the exponential rate of divergence between local state trajectories is reduced. This was empirically verified by our finding that regularised policies exhibit significantly enhanced robustness to observation noise. Collectively, these findings demonstrate that the predictability of the deep RL control interaction is a controllable feature that can be improved by proactively altering the policy objective.

However, while the introduction of Maximal Lyapunov Exponent regularisation into the policy loss successfully reduced the MLE, it also reduced the total reward attained by the policy. Specifically, we found that the linear combination between the standard policy loss and the new regularisation term introduced a direct trade-off between performance and predictability. Consequently, underweighting the regularisation term had no significant impact on the standard Dreamer V3 policy, whereas overweighting it significantly reduced both the MLE and the total reward. As a result, careful selection of the hyperparameter is necessary for producing high-performing policies with predictable dynamics.

Furthermore, while this approach improves the predictability of the control interaction, it is limited to model-based deep RL methods, as the estimation of the MLE requires generating multiple predicted future trajectories using a learned differentiable dynamics model. As a result, this approach cannot be applied to model-free methods, where such a predictive structure is not available. Consequently, this excludes a large class of high-performing deep RL algorithms that do not learn explicit models of the environment dynamics, thereby limiting its applicability across the broader landscape of deep RL policies.

Chapter 6 addresses these limitations by introducing Global stabilisation via Intrinsic Fine Tuning, a model-agnostic fine-tuning framework for improving the predictability of any pre-trained deep RL policy. Rather than modifying the training process, GIFT operates as a secondary optimisation stage applied after a high-performing controller has been learned. In practice, this is achieved through a three-stage procedure. First, a policy is pre-trained using the standard task reward. A Stabilising POMDP is then constructed that shares the same transition dynamics as the original POMDP but replaces the reward function with an intrinsic signal that encourages similarity between the current state trajectory and a high-performing reference trajectory. Finally, the policy is fine-tuned on this modified reward signal, encouraging it to remain within stable regions of the state space while preserving its original behaviour. This decoupled approach allows predictability constraints to be introduced without interfering with the initial performance optimisation process.

Our results demonstrated that this fine-tuning procedure significantly reduced the chaotic behaviour of the control interaction. Specifically, we found that the inclusion of this additional training phase reduced the MLE by up to an order of magnitude across all evaluated environments, indicating a substantial reduction in chaotic behaviour. Furthermore, we found that the fine-tuned policies maintained high levels of task performance, achieving total rewards comparable to those of the pre-trained policies. Taken together, these results demonstrate that GIFT provides a model-agnostic mechanism for improving predictability without modifying the underlying training process.

Collectively, these contributions directly address the central research direction of this thesis: understanding and improving the predictability of the deep reinforcement learning control interaction. Chapter 4 establishes that deep RL policies can induce chaotic dynamics, as measured by the Maximal Lyapunov Exponent. Chapter 5 demonstrates that this unpredictability is not an inherent attribute, but a controllable property that can be influenced through modifications to the policy objective. Chapter 6 extends this insight by introducing a model-agnostic framework that improves predictability without compromising performance. Taken together, these contributions demonstrate that the deep RL policies naturally induce unpredictable state dynamics when controlling simulated control systems and that the level of predictability can be improved.

More broadly, these findings highlight the importance of predictability as a fundamental requirement for deploying deep reinforcement learning in real-world control systems. High-performing controllers must not only achieve strong average performance but also exhibit consistent and reliable behaviour under small perturbations and partial observations. Chaotic dynamics directly undermine this requirement, as they introduce sensitivity to initial conditions and limit the ability to anticipate long-term system behaviour. This provides a principled explanation for the gap between strong empirical performance in simulation and the challenges associated with real-world deployment. By demonstrating that the predictability of the control interactions can be both characterised and improved, this thesis positions predictability as a key consideration in the design of practical reinforcement learning controllers.

7.2 Future Work

The findings presented in this thesis establish a foundational framework for characterising and controlling chaotic dynamics in the reinforcement learning control loop. Building upon these results, we propose two novel research directions that can leverage the findings set out by this thesis. In Section 7.2.1, we propose extending the analysis of chaos into the physical parameter space to address environment variability within a Contextual-MDP. Furthermore, in Section 7.2.2, we investigate the use of chaotic dynamics as a deterministic mechanism for structured exploration in combinatorial optimisation problems. Collectively, these directions transition the study of chaos in reinforcement learning from a localised analysis of state-space predictability to a global perspective on how sensitivity governs the interplay between agent, environment, and the search for optimal solutions.

7.2.1 Chaos in the Physical Parameter Space

Throughout this thesis, we have analysed the chaotic state dynamics produced when deep reinforcement learning policies interact with deterministic continuous control systems. In this framework, the environment dynamics are assumed to be fully determined by the current state and action through the deterministic transition function:

$$\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t). \quad (7.1)$$

This formulation provides a useful abstraction for analysing closed-loop control dynamics and enables the predictability of the control interaction to be studied using tools from dynamical systems theory. However, it assumes that the underlying transition function remains fixed across all episodes. In practice, real-world control systems rarely satisfy this assumption. Physical systems often contain latent environmental or mechanical parameters which influence the system dynamics but remain constant during a single episode. For example, manufacturing defects may result in robots with varying component masses or joint friction. Although these parameters remain fixed for each robot, small variations between robots may cause significantly different state dynamics. As a result, the transition dynamics of real-world systems cannot always be described solely by the current state and action.

One way to formalise this variation in physical parameters is through the Contextual Markov Decision Process (CMDP) [Hallak et al., 2015, Modi et al., 2018]. Formally, a CMDP extends the standard MDP by introducing a set of contextual parameters ($\mathcal{C} \subseteq \mathbb{R}^K$) which characterise the constant physical properties of the environment and a probability distribution over this space ($\mathbb{P}(\mathcal{C})$). At the start of each episode, a context vector ($\mathbf{c} \in \mathcal{C}$) is sampled from $\mathbb{P}(\mathcal{C})$. This vector represents the physical properties of the control system for the duration of the episode and remains fixed throughout the interaction. For each discrete time step thereafter, the environment dynamics proceed as in a standard MDP, except that the transition function is now parameterised by the contextual variables:

$$\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t; \mathbf{c}). \quad (7.2)$$

In this formulation, the context vector acts as a fixed parameter of the transition function as it defines a specific instantiation of the environment. Therefore, different context vectors correspond to different realisations of the control system, such as variations in mass, friction, or actuator behaviour. Although the underlying dynamics remain deterministic once the context is fixed, the distribution over contexts induces a family of related environments with slightly different physical properties. The control policy must therefore operate across a range of transition dynamics that vary between episodes but remain constant within an episode.

This formulation also provides a natural way to analyse the predictability of the control interaction with respect to variations in the physical parameters of the environment. In the CMDP, each context vector defines a particular instantiation of the transition dynamics, meaning that a fixed policy induces a family of related closed-loop dynamical systems. Consequently, variations in the context vector correspond to perturbations of the underlying system dynamics rather than perturbations of the initial state.

Extending the analysis of chaotic dynamics to this setting presents an important direction for future research. In this thesis, the Maximal Lyapunov Exponent was used to quantify sensitivity to perturbations in the free state variables. An analogous sensitivity measure could be defined for perturbations in the contextual parameters while holding the initial state and control policy fixed. Specifically, consider two trajectories ($\mathbf{s}_t$ and $\hat{\mathbf{s}}_t$) which evolve from the same initial state under the same control policy but with slightly different contextual parameters ($\mathbf{c}$ and $\hat{\mathbf{c}}$). We can quantify whether this small perturbation in $\mathbf{c}$ leads to exponentially diverging state trajectories over time using Equation 7.3.

$$\lambda_c = \lim_{t \rightarrow \infty} \lim_{|c - \hat{c}| \rightarrow 0} \frac{1}{t} \ln \left(\frac{|s_t - \hat{s}_t|}{|c - \hat{c}|} \right) \quad (7.3)$$

If this quantity is large and positive, then small changes in the physical parameters of the environment may produce substantially different long-term system behaviour despite identical initial states and control policies. This metric provides a principled way to study policy generalisation across families of related environments, and may help explain why policies trained under nominal conditions often fail when deployed in systems with slight physical variation [Ghosh et al., 2021].

Future work could investigate this phenomenon empirically by measuring contextual sensitivity across families of related control environments with varying physical parameters. In particular, it would be valuable to examine whether policies that induce highly sensitive closed-loop dynamics with respect to contextual perturbations exhibit poorer generalisation across environments with slightly different physical properties. Such analysis may also inform the design of training procedures that explicitly minimise contextual sensitivity, thereby improving the robustness of reinforcement learning controllers when deployed in real-world systems.

7.2.2 Chaotic Exploration for Combinatorial Optimisation

Throughout this thesis, we have focused on reducing the chaotic state dynamics when deep reinforcement learning policies control simulated systems. While reducing chaos is crucial for the safe deployment of control policies in real-world environments, this reduction in chaos could be detrimental to combinatorial optimisation (CO) problems [Korte and Vygen, 2012]. In this setting, diverse solutions are beneficial as RL policies are tasked with finding an optimal solution within a large combinatorial search space [Bonnet et al., 2024, Berto et al., 2025].

Formally, a CO problem is defined by the tuple $(\mathcal{I}, \mathcal{F}, M)$ where $\mathcal{I}$ is the set of all domain instances, $\mathcal{F}$ is the combinatorial set of feasible solutions for a given domain instance and $M : \mathcal{I} \times \mathcal{F} \rightarrow \mathbb{R}$ is a metric which defines an ordering over feasible solutions. For a given domain instance $i \in \mathcal{I}$, the goal of an RL policy is to identify the feasible solution $\bar{x} \in \mathcal{F}$ such that:

$$\bar{x} = \arg \min_{x \in \mathcal{F}(i)} (M(i, x)) \quad (7.4)$$

The challenge with CO problems is that the size of the search space of feasible solutions ($|\mathcal{F}(i)|$) scales combinatorially. For example, in the Travelling Salesman problem [Applegate et al., 2007], an RL policy is tasked with finding the shortest Hamiltonian cycle between $N \in \mathbb{N}$ cities on a map. However, for N cities, there are $|\mathcal{F}(i)| = (N - 1)!$ solutions. Effectively, this produces a massive search space with multiple local minima.

To navigate the combinatorial complexity of the search space, state-of-the-art methods typically leverage stochastic sampling at inference time [Bello et al., 2017, Kool et al., 2019, de Puiseau et al., 2024]. These approaches generate a distribution over $\mathcal{F}(i)$ by unrolling the MDP multiple times while sampling actions from the policy ($a \sim \pi(s)$). The diverse solutions produced by this process are then used to identify the globally optimal configuration. However, this reliance on iterative randomness introduces a significant compounding risk. Specifically, because CO problems are highly sensitive to early structural decisions, sampling a single low-probability action a_t at any step t can cause the entire remaining trajectory to deviate into a suboptimal region of the search space. Consequently, while stochasticity provides the variance required to probe different areas of $\mathcal{F}(i)$, it often does so at the expense of solution integrity, as the agent is forced to sample actions that may ignore its own learned optimisation logic.

To address these limitations, we propose replacing the stochastic sampling approach with a deterministic policy and increasing the MLE produced by this control interaction. This shift is advantageous as:

1. The deterministic policy maintains improved local behaviour as it eliminates the need to sample from a probability distribution at each time step. Consequently, the agent never deviates into suboptimal regions of the state space due to low probability action selection. Instead, it strictly applies the learned optimal action, ensuring that every state transition is consistent with the agent’s value function.
2. The chaotic nature of the policy produces a diverse set of outcomes as infinitesimal perturbations to the initial state result in an exponential divergence of trajectories. Consequently, the required exploration of $\mathcal{F}(i)$ is driven by the sensitive dependence on initial conditions and not the stochasticity of the policy.

This allows the system to project a single, high-quality optimisation heuristic across vastly different regions of the search space. By shifting the source of variance from the decision-making process to the initial state, we can generate a diverse ensemble of locally optimised candidate solutions.

7.3 Closing Remarks

In this thesis, we argue that deep reinforcement learning methods must treat predictability not as a secondary constraint but as a primary objective alongside performance. When predictability is not explicitly encouraged during learning, the resulting policies frequently induce chaotic dynamics and unpredictable outcomes. However, this unpredictability is not inevitable, as it can be significantly reduced using the methods developed in this thesis. As reinforcement learning moves closer to real-world deployment, the ability to ensure predictable behaviour will be essential for building systems that can be trusted.

Bibliography

- Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 06–11 Aug 2017.
- Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021.
- E. Altman. *Constrained Markov Decision Processes*. Routledge, 1st edition, 1999. doi: 10.1201/9781315140223.
- Brian D. O. Anderson and John B. Moore. *Optimal control: linear quadratic methods*. Prentice-Hall, Inc., USA, 1990. ISBN 0136385605.
- David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The Traveling Salesman Problem : A Computational Study*. Princeton University Press, Princeton, UNITED STATES, 2007. ISBN 9781400841103.
- Karl J. Åström and Björn Wittenmark. *Adaptive control*. Addison-Wesley, Reading, Mass;Wokingham;, 1989.
- Karl Johan Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society, 1995. ISBN 1-55617-516-7.
- Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, Dec 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2939-8.

- Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations*, 2017.
- Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. Part 1: Theory. *Meccanica*, 15:9–20, 1980a. ISSN 1572-9648. doi: 10.1007/BF02128236.
- Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application. *Meccanica*, 15: 21–30, March 1980b. ISSN 1572-9648. doi: 10.1007/BF02128237.
- Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, Nayeli Gast Zepeda, André Hottung, Jianan Zhou, Jieyi Bi, Yu Hu, Fei Liu, Hyeonah Kim, Jiwoo Son, Haeyeon Kim, Davide Angioni, Wouter Kool, Zhiguang Cao, Jie Zhang, Kijung Shin, Cathy Wu, Sungsoo Ahn, Guojie Song, Changhyun Kwon, Lin Xie, and Jinkyoo Park. RL4CO: an Extensive Reinforcement Learning for Combinatorial Optimization Benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.
- Clément Bonnet, Daniel Luo, Donal John Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence Illing Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries Petrus Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn Neil Waters, Mohamed Ali Ali Mimouni, Ulrich Armel Mbou Sob, Ruan John de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnau Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable reinforcement learning environments in JAX. In *The Twelfth International Conference on Learning Representations*, 2024.
- Massimo Cencini, Fabio Cecconi, and Angelo Vulpiani. *Chaos*. World Scientific, 2009. doi: 10.1142/7351.
- Ya-Chien Chang and Sicun Gao. Stabilizing Neural Control Using Self-Learned Almost Lyapunov Critics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809, 2021. doi: 10.1109/ICRA48506.2021.9560886.
- Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control

- tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3387–3395, Jul. 2019. doi: 10.1609/aaai.v33i01.33013387.
- Constantin Waubert de Puiseau, Christian Dörpelkus, Jannik Peters, Hasan Tercan, and Tobias Meisen. Beyond training: Optimizing reinforcement learning based job shop scheduling through adaptive action sampling, 2024.
- Jonas Degraeve, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsim-poukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, Feb 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04301-9.
- Robert Devaney. *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. Addison-Wesley, first edition, 1992. ISBN 9780201554069;0201554062.
- R. L. Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, 15(3):458–486, 1970. doi: 10.1137/1115049.
- P. G. Drazin. *Nonlinear Systems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 1992.
- Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468, 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05961-4.
- Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 2018.
- Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25502–25515. Curran Associates, Inc., 2021.
- Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *3rd International Conference on Learning Representations*, 2015.

- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 2018.
- Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, Apr 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2.
- Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *Arxiv*, 2015.
- Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-Critic Reinforcement Learning for Control With Stability Guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, 2020. doi: 10.1109/LRA.2020.3011351.
- Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial Attacks on Neural Network Policies. In *5th International Conference on Learning Representations*, April 2017.
- Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.
- Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012. ISBN 3642244874.
- Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. In *5th International Conference on Learning Representations*, 2017.
- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, San Juan, Puerto Rico, Conference Track Proceedings*, 2016.
- Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3756–3762, 2017. doi: 10.24963/ijcai.2017/525.
- Lennart Ljung. *System identification: theory for the user*. Prentice Hall, Upper Saddle River, N.J, 2nd edition, 1999. ISBN 0132441934;9780132441933;.

- Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20:130 – 141, 1963. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.
- Aleksandr Lyapunov. *The General Problem of the Stability of Motion*. Control Theory and Applications Series. Taylor & Francis, 1992. ISBN 9780748400621.
- Norbert Marwan, M. Carmen Romano, Marco Thiel, and Jürgen Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438(5):237–329, 2007. ISSN 0370-1573. doi: <https://doi.org/10.1016/j.physrep.2006.11.001>.
- Luke Metz, C. Freeman, Samuel Schoenholz, and Tal Kachman. Gradients are Not All You Need. *Arxiv*, 11 2021.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharmashan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. ISSN 1476-4687. doi: 10.1038/nature14236.
- Aditya Modi, Nan Jiang, Satinder Singh, and Ambuj Tewari. Markov decision processes with continuous side information. In Firdaus Janoos, Mehryar Mohri, and Karthik Sridharan, editors, *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, pages 597–618. PMLR, 07–09 Apr 2018.
- Edward Ott, Celso Grebogi, and James A. Yorke. Controlling chaos. *Phys. Rev. Lett.*, 64: 1196–1199, Mar 1990. doi: 10.1103/PhysRevLett.64.1196.
- Thomas S. Parker and Leon O. Chua. *Practical Numerical Algorithms for Chaotic Systems*. Springer New York, NY, first edition, 2011. ISBN 978-1-4612-8121-4. doi: 10.1007/978-1-4612-3486-9.
- Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. PIPPS: Flexible Model-Based Policy Search Robust to the Curse of Chaos. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4065–4074. PMLR, 2018.
- K. Pyragas. Continuous control of chaos by self-controlling feedback. *Physics Letters A*, 170(6):421–428, 1992. ISSN 0375-9601. doi: [https://doi.org/10.1016/0375-9601\(92\)90745-8](https://doi.org/10.1016/0375-9601(92)90745-8).
- Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22:1–8, 2021.

- J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017. ISBN 9780975937730.
- O.E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976. ISSN 0375-9601. doi: [https://doi.org/10.1016/0375-9601\(76\)90101-8](https://doi.org/10.1016/0375-9601(76)90101-8).
- David Ruelle. SENSITIVE DEPENDENCE ON INITIAL CONDITION AND TURBULENT BEHAVIOR OF DYNAMICAL SYSTEMS. *Annals of the New York Academy of Sciences*, 316:408–416, 1979. doi: <https://doi.org/10.1111/j.1749-6632.1979.tb29485.x>.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.
- Hua Shao, Yuming Shi, and Hao Zhu. Lyapunov Exponents, Sensitivity, and Stability for Non-Autonomous Discrete Systems. *International Journal of Bifurcation and Chaos*, 28:1850088, 2018. doi: [10.1142/S0218127418500888](https://doi.org/10.1142/S0218127418500888).
- David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Beijing, China, 22–24 Jun 2014. PMLR.
- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. ISSN 1476-4687. doi: [10.1038/nature16961](https://doi.org/10.1038/nature16961).
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017. ISSN 1476-4687. doi: [10.1038/nature24270](https://doi.org/10.1038/nature24270).

- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362:1140–1144, 2018. doi: 10.1126/science.aar6404.
- Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, Aug 1988. ISSN 1573-0565. doi: 10.1007/BF00115009.
- Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.
- Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.
- Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: <https://doi.org/10.1016/j.simpa.2020.100022>.
- L. N. Vaserstei. Markov processes over denumerable products of spaces, describing large systems of automata. *Probl. Peredachi Inf*, 5:64–72, 1969.
- Tao Wang, Sylvia Herbert, and Sicun Gao. Fractal Landscapes in Policy Optimization. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 4277–4294. Curran Associates, Inc., 2023.
- Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. ISSN 1573-0565. doi: 10.1007/BF00992696.
- Ronald J. Williams and Jing Peng. Function Optimization Using Connectionist Reinforcement Learning Algorithms. *Connection Science*, 3:241–268, 1991. doi: 10.1080/09540099108946587.

- Rory Young and Nicolas Pugeault. Enhancing Robustness in Deep Reinforcement Learning: A Lyapunov Exponent Approach. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 86102–86123. Curran Associates, Inc., 2024.
- Rory Young and Nicolas Pugeault. Chaos in deep reinforcement learning: Analysis and stabilisation with lyapunov exponents. 2026a.
- Rory Young and Nicolas Pugeault. Gift: Global stabilisation via intrinsic fine tuning. 2026b.
- Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo Playground, 2025.
- Kemin Zhou, John C. Doyle, and K. Glover. *Robust and optimal control*. Prentice Hall, London;Upper Saddle River, N.J., 1996. ISBN 0134565673;9780134565675;.
- Karl Johan Åström. Optimal Control of Markov Processes with Incomplete State Information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965. ISSN 0022-247X. doi: 10.1016/0022-247X(65)90154-X.

Appendix A

Appendix

A.1 Hyperparameters

A.1.1 Chapters 4 and 5

Table A.1: Hyperparameters used to train SAC, TD3 and Dreamer V3 in Chapters 4 and 5.

SAC and TD3		Dreamer V3	
Parameter	Value	Parameter	Value
Environment steps	5×10^6	Environment steps	10^6
Buffer size	10^6	Buffer size	10^5
Parallel environments	8	Parallel environments	8
Update period	1	Update period	1
Updates per step	8	Updates per step	8
Discount factor	0.99	Discount factor	0.99
Learning rate	0.0003	Learning rate	10^{-4}
Batch size	1024	Batch size	15
Polyak update coefficient	0.005	Batch length	64
Networks activation	Tanh	Imagination horizon	15
Networks depth	5	Network activation	LayerNorm + SiLU
Networks width	256	Networks depth	4
		Networks width	512
		Recurrent state size	4096
		Number of latents	32
		Classes per latent	32

A.1.2 Chapter 6

Table A.2: Hyperparameters used to train SAC and PPO in Chapter 6.

SAC		PPO	
Parameter	Value	Parameter	Value
Environment steps	5×10^6	Environment steps	6×10^7
Buffer size	2^{22}	Parallel environments	128
Parallel environments	128	Unroll length	30
Update period	1	Batch size	32
Updates per step	8	Updates per batch	16
Discount factor	0.99	Discount factor	0.995
Learning rate	0.001	Learning rate	0.001
Batch size	512	Entropy constant	0.01
Polyak update coefficient	0.005	Networks activation	Swish
Networks activation	ReLU	Policy networks depth	6
Networks depth	4	Policy networks width	32
Networks width	256	Value networks depth	7
		Value networks width	256