Artificial intelligence and fusion plasma control : application to the WEST tokamak

Samy Kerboua-Benlarbi

Intelligence artificielle et contrôle des plasmas de fusion: Application au tokamak WEST

Samy KERBOUA-BENLARBI

Laboratoire Jean-Alexandre Dieudonné (LJAD)

Présentée en vue de l'obtention du grade de docteur en Mathématiques **d'**Université Côte d'Azur

Dirigée par : Blaise Faugeras
Co-direction : Rémy Nouailletas

Soutenue le : 5 Novembre 2024

Devant le jury, composé de :

Feda Almuhisen
Ingénieure-chercheuse, CEA, IRFM

Blaise Faugeras
Ingénieur-chercheur, CNRS, UCA

Federico Felici
Chercheur spécialiste, Google DeepMind

Sylvain Lamprier
Professeur, Université d'Angers, LERIA

Laurent Lefèvre
Professeur, Grenoble INP Esisar - UGA, LCIS

Rémy Nouailletas
Ingénieur-chercheur, CEA, IRFM

Olivier Sauter
Senior scientist, EPFL, SPC

Capgemini engineering
cea irfm
RÉGION SUD PROVENCE ALPES CÔTE D'AZUR

Intelligence artificielle et contrôle des plasmas de fusion: Application au tokamak WEST

Artificial intelligence and fusion plasma control:
Application to the WEST Tokamak

Jury :

Président du jury

Laurent Lefèvre	Professeur	Grenoble INP Esisar - UGA, LCIS

Rapporteurs

Federico Felici	Chercheur spécialiste	Google DeepMind
Sylvain Lamprier	Professeur	Université d'Angers, LERIA

Examinateurs

Olivier Sauter	Senior scientist	Ecole Polytechnique Fédérale de Lausanne, SPC

Directions

Blaise Faugeras	Ingénieur-chercheur	CNRS, UCA
Rémy Nouailletas	Ingénieur-chercheur	CEA, IRFM

Invitée

Feda Almuhisen	Ingénieure-chercheuse	CEA, IRFM

RÉSUMÉ

La fusion dans un plasma magnétiquement confiné relève encore du domaine de la recherche fondamentale : en plus de la nécessaire progression de nos connaissances théoriques, l'opération des tokamaks actuels reste délicate, car elle nécessite un effort humain substantiel à chaque fois qu'un nouveau scénario expérimental est mis au point. En outre, la combinaison habituelle de rétroactions linéaires et de contrôle en boucle ouverte, n'est pas complètement robuste vis à vis du comportement non-linéaire des dynamiques du plasma. L'approche dont il est question n'est donc pas compatible avec la fiabilité d'un futur réacteur, et un meilleur contrôle du plasma est nécessaire par le biais d'algorithmes de contrôle utilisant des connaissances plus théoriques ou empiriques. Récemment, l'apprentissage par renforcement a démontré son utilité dans de nombreux domaines, notamment le contrôle des tokamaks. Par essai-erreur, un agent interagit avec un environnement, pour apprendre une politique décisionnelle maximisant une récompense formalisant les tâches à accomplir. Une fois associé aux réseaux de neurones, l'apprentissage profond par renforcement devient un candidat pertinent pour répondre à ces situations en grande dimension, aux nombreuses incertitudes, et non-linéaires. Ces travaux visent à appliquer et étendre ces méthodes au tokamak WEST, au travers du développement d'une plateforme logicielle pour entraîner un agent sur un code d'equilibre à frontière libre, qui simule l'équilibre magnétique et l'évolution des profils au coeur du plasma. Le contrôle magnétique par rétroaction de la forme, de la position et du courant du plasma est alors réalisé sur plusieurs scénarios de contrôle, avec des temps d'entraînement considérablement réduits grâce à l'utilisation de l'apprentissage par curriculum. Plusieurs procédures sont discutées, non seulement pour améliorer la généralisation dans le domaine opérationnel du système de contrôle en temps-réel de WEST, mais aussi pour explorer la robustesse des agents appris par renforcement face aux perturbations et aux incertitudes du plasma. Les résultats obtenus offrent de nombreuses perspectives en faveur de ce paradigme, et de l'intelligence artificielle plus généralement, dans le cadre d'une utilisation de routine au sein des opérations d'un dispositif de fusion confiné magnétiquement. Les liens inhérents établis entre l'apprentissage par renforcement et le contrôle classique sont étayés, et pourraient conduire à une meilleure interprétabilité des réseaux de neurones en tant que politiques puissantes et robustes, potentiellement capables de gérer un plus grand nombre d'actionneurs, comme d'objectifs, pendant de longues décharges de plasma.

MOTS CLÉS

Apprentissage par renforcement, Contrôle des tokamaks, Réseaux de neurones, Calcul distribué, Inférence probabiliste

ABSTRACT

Fusion in a magnetically confined plasma is still in the realm of fundamental research: in addition to the necessary progress in our theoretical knowledge, the operation of current tokamaks remains delicate, as it requires substantial human effort each time a new experimental scenario is developed. Moreover, the usual combination of linear feedback and feedforward control is not very robust with respect to the nonlinear behavior of plasma dynamics. The overall approach is then not compatible with the reliability of a future reactor, and a better control of the plasma is necessary through control algorithms using more theoretical or empirical knowledge. Reinforcement Learning (RL) has recently demonstrated its usefulness in many fields, notably tokamak control. By trial-and-error, an agent interacts with an environment, to learn a behavioral policy maximizing a reward which formalizes the overall objectives. Paired with neural networks, deep reinforcement learning becomes a suitable candidate to fulfill these high-dimensional, uncertain and nonlinear situations. This thesis aims to apply and extend the said methods on the WEST tokamak, with the development of a general framework to train an agent on a free-boundary equilibrium solver, which simulates the magnetic equilibrium and core profiles evolution. Magnetic feedback control of plasma's shape, position, and current is achieved on multiple control scenarios, with training times significantly reduced by the use of curriculum learning. Several procedures are discussed, not only to enhance generalization within the operational domain of the WEST real-time control system, but also to explore the robustness of RL agents against disturbances and plasma uncertainties. The obtained results offer many perspectives in favor of reinforcement learning, and more generally artificial intelligence, for routine use in the operation of a magnetically confined fusion device. The inherent connections drawn between reinforcement learning and classical control are studied, and could lead to a more interpretable use of neural networks as powerful and robust policies, potentially able to manage more actuators, and objectives, during long plasma discharges.

KEYWORDS

Reinforcement Learning, Tokamak control, Neural networks, Distributed computing, Probabilistic Inference

Acknowledgments

The words on this page fill me with joy and emotion as I contemplate this manuscript, which could not have seen the light of day without the help of people whom my mind, like my heart, cannot forget.

First of all, I would like to thank my supervisors, Dr. Blaise Faugeras and Dr. Rémy Nouailletas, for giving me this fantastic opportunity to join the fusion world. This long-held dream came true because you believed in me and shared your passion for this fantastic field, supporting me throughout this journey. I would also like to thank Philippe Moreau for his sound advice throughout my thesis. I have learned much from his expertise on tokamaks. The whole GPAM team, and STEP in general, comprises brilliant people with whom every moment spent around the coffee machine made everyday life even better. I would also like to thank the people at Capgemini Engineering, who played an essential role in this project's realization and supported me throughout.

I am genuinely thankful to all the members of my thesis jury for providing insightful feedback, notably Dr. Federico Felici and Pr. Sylvain Lamprier, whose reviews helped me enhance my research.

The IT department must be acknowledged for being resilient in the face of my training sessions, which had the merit of creating a running gag like few others. I also thank the Jean Zay servers for their extremely helpful waiting times.

As soon as I arrived at CEA, the discussions and debates I had with you, Kirill, led us to put the world to rights on many occasions. I sincerely want to thank you for these warm and enriching moments, where I discovered many concepts and ideas that will follow me from now on. Moreover, I cannot forget all the passionate discussions and laughter I shared with you, Nathan, and I thank you for all that. So many memories, from France to Japan through motorcycling and memes, will stay with me forever. Thank you, Kevin, for everything we talked about: science, linguistics, music and life. At all hours, our talks were as many discoveries as moments that changed my perspective on the world.

Thank you to my dear friends with whom I shared the ASTHEC office: Guilherme, Virginia, Nicolas, and Theo. I am proud to have learned from and rubbed shoulders with you over these three years, between the afterworks, our challenges, and successes. You guys are great!

Sharing laughs and joys is something I have found in each of my colleagues: Mathieu, Jessica, James, Elisa, Nicolas, Olivier, Quentin, Alexis, Yann, Pierre, Raffael, Maxime and many others. Meeting all of you and sharing those incredible moments was a privilege.

To you, Mathilde, thank you from the bottom of my heart. Ideas fail me to express all the gratitude I wish to express to you. Your words have motivated me to hold on in the face of adversity and smile when I thought it was impossible.

I could not write this section without thanking my family. Those who helped me grow up in the best environment: my aunts, including whom I call my second mother, my uncles, my cousins, my brother, my grandparents, those who are there and those who can no longer be. Your memory encouraged me during the entirety of this work. My family also continues with my closest friends, who form it in many ways, from the benches of

Paris 7 to those of Paris 6 and more: Bryan, Antoine, Edgar, Pierre, Clémence, Caroline, Lydia, Yani, Victor, and Arthur. Not to mention Yacine, Ilan, Antoine and Clément. Knowing you is an honour, and I need an entire manuscript to recount your exploits! I continue to grow thanks to each of you, having the most sincere feelings for your help and remembering that year after year, you are my brothers and sisters in time and space.

To you, Mom and Dad, as I reflect on the profound impact you both had on my life. Mom, you taught me to believe in myself, and Dad, you showed me the importance of thinking. Your unwavering support has lifted me up and helped me stand tall, filled with pride to be your son. Thank you for everything and beyond.

Finally, I have spent my life raising my eyes to the sky to contemplate the stars. These last few years have taught me that all I have to do is look around me to see the most sparkling ones I have been given to cherish.

There is nothing like looking, if you want to find something. You certainly usually find something, if you look, but it is not always quite the something you were after.

J.R.R Tolkien, The Hobbit

Journal publications

- S. Kerboua-Benlarbi, R. Nouailletas, B. Faugeras, E. Nardon and P. Moreau, *Magnetic Control of WEST Plasmas Through Deep Reinforcement Learning*, in IEEE Transactions on Plasma Science - 2024
- S. Kerboua-Benlarbi, R. Nouailletas, B. Faugeras and P. Moreau, *Curriculum Reinforcement Learning for Tokamak Control*, in Lecture Notes on Artificial Intelligence, part of IJCAI 2024 proceedings - Accepted, 2025

Conference and workshop posters

- Symposium On Fusion Engineering (SOFE), Oxford, United Kingdom - 2023
- Workshop on Artificial Intelligence for Accelerating Fusion and Plasma Science, IAEA headquarters, Vienna, Austria - 2023
- Workshop on Artificial Intelligence for Research at the International Joint Conference on Artificial Intelligence (IJCAI), Jeju island, South Korea - 2024

Oral presentations

- 41st MHD, Disruptions and Control Topical Group Meeting, International Tokamak Physics Activity (ITPA), ITER Organization, France - 2023

Contents

1 Introduction **15**
- 1.1 Plasma control and Artificial Intelligence . . . 16
  - 1.1.1 Thermonuclear fusion and tokamaks . . . 16
  - 1.1.2 The need for robust magnetic control . . . 19
  - 1.1.3 Machine Learning and company at the rescue . . . 22
- 1.2 Objectives and main contributions . . . 25
- 1.3 Outline . . . 27

2 Inference-based reinforcement learning **29**
- 2.1 Fundamentals . . . 30
- 2.2 Connections with optimal control . . . 34
- 2.3 Finding optimal policies . . . 36
  - 2.3.1 A taxonomy of algorithms . . . 36
  - 2.3.2 The road to the actor-critic's pantheon . . . 38
  - 2.3.3 Neural approximators . . . 39
  - 2.3.4 Exploring or exploiting: a core dilemma . . . 43
- 2.4 An inference-based interpretation of deep RL . . . 44
  - 2.4.1 Introduction on probabilistic inference . . . 45
  - 2.4.2 Maximum a posteriori Policy Optimization . . . 49
- 2.5 An agent ready for interactions . . . 57

3 PILOT: a general framework for magnetic control **60**
- 3.1 Creating a numerical twin for WEST . . . 62
  - 3.1.1 Machine description . . . 62
  - 3.1.2 Control scenarios . . . 63
  - 3.1.3 A NICE environment to train them all . . . 67
- 3.2 In a world of scenarios and rewards . . . 79
  - 3.2.1 References generator . . . 79
  - 3.2.2 Reward definition . . . 80
  - 3.2.3 A digression on the WEST plasma control system . . . 85
- 3.3 Assembling a distributed architecture . . . 88
  - 3.3.1 The wonderful story of how C++ met Python . . . 88
  - 3.3.2 Nodes galore . . . 89

3.3.3 A glimpse of the agent's distinctive features . . . 91
3.4 A framework ready for training . . . 94

4 The need for speed in PILOT 95
4.1 Accelerating training through curriculum learning . . . 96
4.1.1 An inspiration from human learning . . . 97
4.1.2 One does not simply generate a curriculum . . . 98
4.1.3 Connections to the state-of-the-art . . . 100
4.1.4 Limitations of the current approach . . . 104
4.2 A structural view against catastrophic forgetting . . . 105
4.3 A procedure ready for benchmark . . . 107

5 Performance of RL-based magnetic control 109
5.1 Evaluation on the scenarios of interest . . . 110
5.1.1 Plasma centroid, elongation and minor radius . . . 110
5.1.2 Careful calibration of the reward hyperparameters . . . 115
5.2 Issues regarding the LCFS and the plasma current . . . 117
5.2.1 Myopic exploration . . . 117
5.2.2 An issue regarding plasma current . . . 119
5.3 The Good, the Bad and the Ugly of Curriculum learning . . . 123
5.4 A paradigm under careful calibration . . . 126

6 Conclusion and perspectives 127

A Formal comparison between RL and OC 132

B Precisions on value and gradient-based methods 140
B.1 A focus on value learning . . . 141
B.2 Looking at policy learning . . . 143

C Precisions on WEST 145
C.1 A more precise representation of WEST geometry . . . 146
C.2 A noisy description of delays . . . 146
C.3 May the snapshot be with you . . . 148

D Reward definitions 149
D.1 Reward components . . . 150
D.2 Reward definition . . . 151
D.3 Curriculum definition . . . 153

E Training hyperparameters 154
E.1 The role of each hyperparameter . . . 155
E.1.1 Generally for PILOT . . . 155
E.1.2 Specifically for MPO . . . 155
E.2 Current configuration . . . 156

E.3 Neural architectures 157

List of Figures

1.1 Nuclear fusion from Deuterium and Tritium . . . 16
1.2 Plasma definition . . . 17
1.3 Effects of a uniform magnetic field on plasma's particles . . . 18
1.4 Particles' vertical drift in tokamaks . . . 18
1.5 Influence of toroidal and poloidal fields on particles' trajectory . . . 19
1.6 Core components of a tokamak . . . 20
1.7 Classical feedback control on WEST . . . 22
1.8 The russian dolls of modern artificial intelligence . . . 24
1.9 Reinforcement learning-based feedback control on WEST . . . 25

2.1 Interaction loop between the agent and the environment. . . . 31
2.2 General formulation of policy search . . . 37
2.3 A fully connected neural network . . . 40
2.4 Gradient descent methods . . . 41
2.5 Unrolling a recurrent neural network . . . 42
2.6 Summary of the Maximum a posteriori Policy Optimization algorithm . . . 59

3.1 The WEST tokamak at CEA, Cadarache . . . 61
3.2 WEST cross-section with control coils . . . 62
3.3 WEST transition between limiter and X-point configurations . . . 63
3.4 The (r,ϕ,z) coordinate system in tokamaks . . . 64
3.5 The vertical instability issue . . . 66
3.6 Generic view of the poloidal plane of ta tokamak . . . 68
3.7 Initialization procedure of the NICE environment . . . 71
3.8 Stability study after plasma profiles variations in NICE . . . 74
3.9 Closed-loop training and inference using PILOT . . . 77
3.10 Magnetic probes and flux loops locations on WEST . . . 78
3.11 The four horsemen of RL-based control experiments . . . 80
3.12 The *Softplus* transformation . . . 82
3.13 The *Smoothmax* combination . . . 83
3.14 Full environment to mimic WEST and its control system . . . 84
3.15 Event scheduler of the WEST plasma control system . . . 85
3.16 WEST control tolerance using envelopes . . . 86

3.17 Communications between the policy and the NICE client 89
3.18 The complete PILOT framework in the context of distributed actor-critic agents 90
3.19 Architecture of the critic network 92
3.20 Architecture of the policy network 93

4.1 A curriculum from a restriction of the general case 99
4.2 Comparison between the curriculum and the *chunk* procedure 101
4.3 Curriculum overview 102
4.4 An illustration of catastrophic forgetting 105
4.5 Progressive neural networks 106

5.1 Stabilization of a fixed limiter plasma with constant references 111
5.2 Tracking of a limiter plasma with several moving targets 112
5.3 Stabilization of an X-point plasma with constant references 113
5.4 Transition from of a limiter plasma to an X-point configuration 114
5.5 The difficult reward calibration 115
5.6 Reward decomposition within an example trajectory 116
5.7 Unusual evolution of PF coils currents 117
5.8 Time evolution of the LCFS and the X-point with fixed references 118
5.9 Time evolution of the LCFS and the X-point with moving references 119
5.10 X-point configuration falls into two sub-optimal policies 120
5.11 The two sides of the plasma current control 121
5.12 Circuit equations for the central solenoid and the plasma 122
5.13 Quantitative comparison between the basic and curriculum approaches . . . 124
5.14 Episodic return for both the curriculum approach and the vanilla one 125

A.1 Open and closed loop interactions with an environment 135
A.2 Online Model Predictive Control 137
A.3 Optimal trajectory under *Bellman's principle of Optimality* 138

C.1 Realistic view of the WEST poloidal plane 147
C.2 Examples of snapshots used to build the plasma scenarios 148

List of Tables

1.1 Comparison between existing learning paradigms 23

3.1 Description of NICE modes 70
3.2 PID gains and their impact on system's response 87

5.1 Root mean squared error of principal reward components 123

List of Abbreviations

Control aspects

DP Dynamic Programming

MPC Model Predictive Control

OC Optimal Control

PID Proportional-Integral-Derivative

Implementation

PILOT Plasma reInforcement Learning fOr Tokamaks

TCP Transfer Communication Protocol

UDS Unix Domain Sockets

Artificial intelligence

AC Actor-Critic

CL Curriculum learning

EM Expectation-Minimization

LSTM Long-Short-Term-Memory

MLP Multi-Layered Perceptron

MPO Maximum a posteriori Policy Optimization

RL Reinforcement Learning

Tokamak physics

LCFS Last Closed Flux Surface

PCS Plasma Control System

PF Poloidal Field

TF Toroidal Field

Introduction

Contents

1.1 Plasma control and Artificial Intelligence 16

1.1.1 Thermonuclear fusion and tokamaks 16

1.1.2 The need for robust magnetic control 19

1.1.3 Machine Learning and company at the rescue 22

1.2 Objectives and main contributions 25

1.3 Outline 27

1.1 Plasma control and Artificial Intelligence

1.1.1 Thermonuclear fusion and tokamaks

Global demand for energy has been growing steadily since the advent of the industrial era and the technological developments which have followed. In a context where energy production is facing the decline of fossil fuels, new approaches need to be considered to reduce the carbon footprint of many societies. Nuclear energy has been mastered through fission since the middle of the 20th century, knowing that the reaction does not emit greenhouse gases when carried out. Nevertheless, its combustible is only available in limited quantities on Earth. In parallel, thermonuclear fusion could offer interesting possibilities concerning the shortcomings of fossil energy sources. With no direct byproducts, no risks of chain reaction and fuel partly available in large quantities, it has many advantages over carbon-emitting energy sources [9]. However, it should be qualified regarding the current expectations of its usefulness for climate change. Indeed, we must consider the research and engineering time scales needed for the complete development of such technology, which will not be available sufficiently soon to counteract and limit its consequences. Consequently, mastering nuclear fusion will not instantaneously open the way to a miraculous means of energy production, but it could definitely join the energy mix as a strong asset.

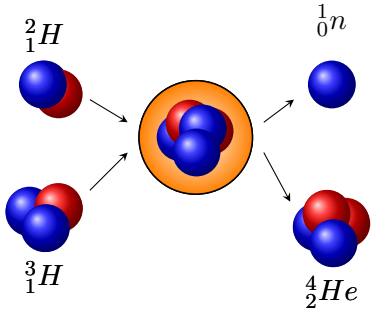


Figure 1.1: Deuterium and tritium fuse, producing helium, a neutron and releasing 14.1 MeV.

To understand how we experimentally control this phenomenon, we must closely examine the conditions under which the fusion reaction occurs. Two positively charged light nuclei must be brought together to achieve it, overcoming electrostatic repulsion. For example, a helium nucleus could be obtained from two deuterium and tritium nuclei, by emitting a neutron and releasing energy (Figure 1.1). Thanks to quantum tunneling, there is a non-zero probability of overcoming the Coulomb barrier without needing a colossal amount of energy, which even the Sun's properties could not respect. With that in hand, a sufficiently high energy level must still be reached under specific constraints of density and temperature. Those conditions are attained within a plasma, the universe's fourth and most common state of matter (Figure 1.2). Observed commonly in stars like our beloved

Sun, this combination of charged particles is heated at several millions of degrees celsius[1], undergoing high gravity, which produces a fusion plasma.

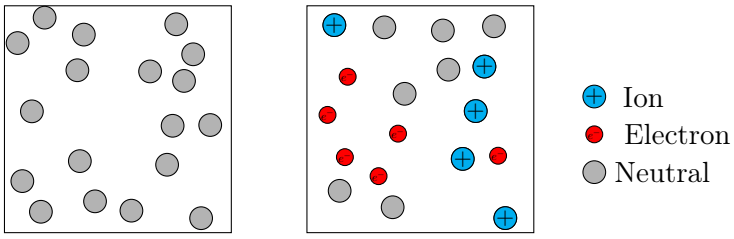


Figure 1.2: Turning a gas into a plasma in which fusion reactions are observed under specific constraints.

The system must be sustained with a plasma that is sufficiently hot (T), dense (n), and with a long confinement time τ_e. The latter corresponds to the characteristic time it takes for the plasma to empty itself of its energy content if the injected sources are cut off. Such triple product defines the **Lawson criterion**, which appears as a strong empirical indicator of how well the plasma ignites to obtain a large number of fusion reactions:

$$nT\tau_e \geq 3 \times 10^{21}[m^{-3}keVs]$$

Reproducing the exact conditions that one would find in stars is not sustainable on Earth. Indeed, without important gravitational forces, we must rely on other physical concepts, such as magnetic fields. This definition leads to the two known practical ways of reproducing what we observe on the solar scale:

- **magnetic confinement fusion** where τ_e is of the order of a second and n is 10^{20} particles per m^3 (10^{-5} the density of air)
- **inertial confinement fusion** where τ_e is of the order of 10^{-10} second and n is 10^{30} particles per m^3 (10^4 the density of liquid materials in general).

In both situations, the ionic temperature T_i is approximately around 10keV. This PhD is solely focused on *Tokamaks* (Russian acronym for *toroidal'naya kamera s magnitnymi katushkami*), which belong to the first kind of devices. Their purpose follows the previous distinction: to compensate for the low plasma density by optimizing its temperature and confinement time. As a result, it produces plasmas with temperature gradients more significant than any natural phenomenon, at the risk of breaking its components. A question arises regarding how tokamaks function in this context and how they try to maximize confinement performance to obtain stable plasmas. As stated earlier, plasma is made out of charged particles. Under the influence of a magnetic field, particles move helically along its field lines (Figure 1.3). The gyration or *Larmor* radius of a particle,

[1] 15 million degrees celsius for the Sun notably

depends on its mass, charge, energy, and the magnetic field's intensity. If the intensity increases, the Larmor radius diminishes, even more when the particle mass is small at the same energy levels. If its energy augments, the Larmor radius follows accordingly. Knowing that particles of interest are highly energetic, the gyration radius of several species, such as Helium, can reach about tens of centimeters. However, if one only considers this kind of transport, it would be relatively easy to maintain a particle inside the plasma. However, abnormal and turbulent transport make it difficult to keep all of them in the latter [130]. To produce fusion reactions, we need a device capable of controlling magnetic fields efficiently, to confine the particles inside the plasma and optimize their trajectory despite the elements stated before.

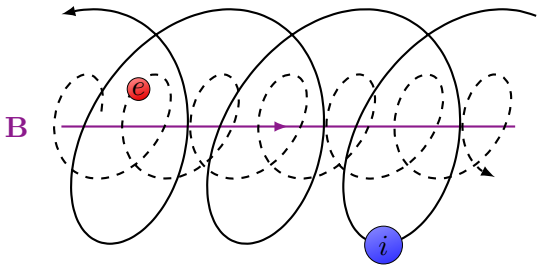


Figure 1.3: Electron and Ion trajectory in a straight and uniform magnetic field.

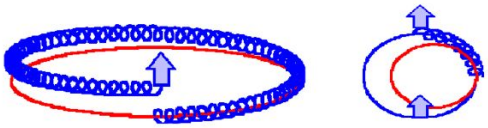


Figure 1.4: Schematic view of the particles' vertical drift.

Until now, we considered a straight and uniform magnetic field in what we could call a *linear device*. This configuration would create losses on both ends, so we close this cylinder-like object to create a torus. *Toroidal* magnetic coils surround this structure in order to generate a magnetic field that allows circular trajectories meant to confine the particles. However, a drift appears, which makes the overall setup insufficient for achieving stable plasmas. Indeed, if we see the plasma as a conductive wire, the toroidal magnetic field B_T can be expressed through Ampere's law:

$$B_T = \frac{\mu_0 I_T}{2\pi R}$$

where μ_0 is the vacuum permeability, R the major radius of the magnetic field line, and I_T the current in the toroidal coils. B's magnitude is inversely proportional to R, meaning that a reduction of the major radius leads to a higher magnetic field intensity. Generalizing to a regular plasma's representation, the magnetic field is not uniform as it becomes stronger while going from the outside edge of the torus to its inside one. In this way, particles tend to pull away from the field lines once the *lower field side* has been reached. Taking into account the remaining factors, such as the magnetic field gradient and the centrifuge force, we get a vertical drift whose direction depends on the respective sign of each particle (Figure 1.4). Thus, a simple torus with toroidal coils is insufficient to maintain a plasma, even if the toroidal magnetic field was supposed to be a first step towards properly optimizing particle trajectories. By transformer effect, if a strong current

is induced in the plasma, it would generate a *poloidal magnetic field* added to the toroidal one. Adding a central solenoid at the center of the torus alleviates this structural notion to produce the said current and define the full meaning of the tokamak concept. By combining both components, magnetic field lines have a helical trajectory around the plasma so that particles on the outer side of the torus go back on the inside. More precisely, a plasma is made out of nested magnetic flux surfaces, and the combined magnetic field exhibits field lines that run over the flux surfaces helically (Figure 1.5). The addition of several poloidal coils allows the refinement of generated fields and thus controls the plasma's position and geometry. Despite not being important for our subject of interest, it is worth mentioning that superconductive coils are preferred, as they allow for longer discharges, contrary to regular coils (copper, for example), which would overheat really fast under the stated conditions.

Figure 1.5: Magnetic fields in both poloidal (vertical) and toroidal (horizontal) directions are at the foundations of the tokamak concept.

Therefore, tokamaks are torus-shaped configurations of coils that make it possible to build a magnetic trap efficient enough to confine particles inside the plasma, and sustain the necessary conditions for fusion to happen. Under specific temperature and density conditions [130], they rely on magnetic fields generated by *toroidal* and *poloidal* field coils to control the plasma. In fact, maintaining a plasma according to a scenario of interest is performed using tokamak's various actuators ranging from coils, to heating systems, through fueling ones. During this succession of events, the plasma is subjected to several modifications which create instabilities that compromise systems safety. A natural question emerges on how control is performed on the device of interest and what is needed for a better mastery of plasma's unstable evolution.

1.1.2 The need for robust magnetic control

As stated previously, control systems are required to perform tracking of quantities intrinsically linked to plasma's evolution, like the position of the plasma centroid m, its boundary or *Last Closed Flux Surface* (LCFS), elongation κ and current I_p (Figure 1.6). Future devices like ITER[2] integrate many systems to tackle the numerous control issues that appear in tokamaks. For example, ITER specifications regarding plasma current

[2] *International Thermonuclear Experimental Reactor* - `https://www.iter.org/`

range from 1 to 15MA, hence the need for an efficient control apparatus. Specifically, magnetic control plays a crucial role in modifying the plasma's position and geometry, which impact the stability and performance of plasma's confinement within tokamaks. Through the active adjustment of the voltages applied to the poloidal field (PF) coils, magnetic fields are carefully manipulated within the said devices (Figure 1.6). This work will then focus entirely on magnetic control, despite the fact that other domains could benefit from the same investigation found in this work.

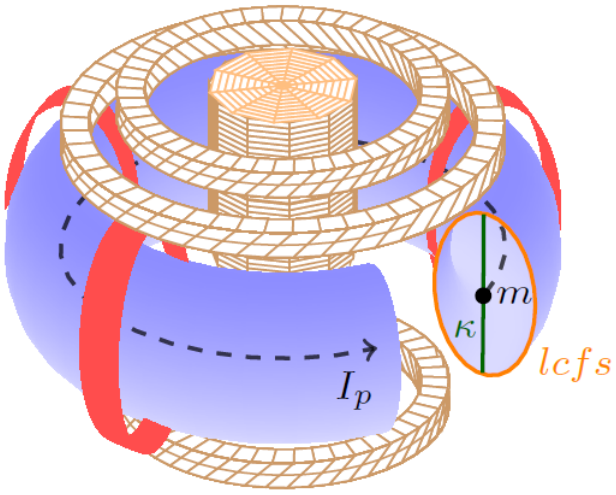


Figure 1.6: Schematic view of a tokamak with toroidal (red) and poloidal (striped gold) coils. Many quantities must be controlled.

We usually refer to a plasma *discharge* as an experiment during which the plasma undergoes structural changes from its creation to its shutdown. The plasma is initiated from the ionization of prefilled gas by the generation of a toroidal electric field via transformer action using the central solenoid, which ends up being similar (to a certain extent) to a Townsend avalanche discharge [28]. Essentially, the current is ramped up in the central solenoid, and a pre-magnetization map is built from poloidal coils. A rapid drop of the central solenoid current induces an acceleration of the electrons by the electric field. They collide and ionize more neutrals, producing a plasma at high loop voltage. This so-called plasma breakdown will notably differ on ITER, as it will need additional heating from the low loop voltage at which it should operate plasma initiation. The plasma is traditionally launched in a circular fashion, usually known as a *limiter* configuration. Once formed, the plasma evolves depending on the purpose of the experiment, with examples ranging from:

- reaching an X-point configuration, i.e., a configuration limiting the pollution of plasmas from wall materials, and on which will go back later on;
- to modifying the plasma current and exploring shapes within the operational domain of the machine.

In many modern devices, especially on *WEST*[3], numerous plasma experiments transition between limiter configurations and the first option, consistently ramping up the plasma current. This evolution is meant to reach a stable operating regime during the so-called *flat-top* phase. The last part of a discharge, known as the *ramp-down* phase, is used to shut down the plasma safely. However, disruptions can occur prematurely. These quick and brutal losses of magnetic confinement can damage the machine and are, in fact, strictly prohibited in the operation of future devices. This hazard could be related to escaped particles but also linked to mechanical constraints placed on the tokamak elements. Some of them are sensitive to magnetic fields, and the high rate of change in plasma current over time generates strong forces on those structures. New control approaches are being studied to answer such phenomenon, but this question is outside the scope of this manuscript. More precisely, this work focuses entirely on plasma evolution involving limiter and X-point plasmas since this phase is unmissable in the landscape of magnetic control, as it allows the exploration of shapes and configurations between the latter. Throughout a discharge, interactions between the plasma and external circuits are governed by non-linear partial differential equations. Conventionally, several Single-Input-Single-Output *Proportional-Derivative-Integral* controllers (PID) are deployed to regulate the system [8], all of which must be designed and structured not to interfere with each other. Controller design is then based on linearization around an equilibrium point, leaving a time-invariant, low-order differential system. Ultimately, classical control needs a physical model that is sufficiently detailed to reproduce the involved phenomena, yet simple enough for a controller to be built. The said approximations highly depend on the geometric parameters of the plasma to be controlled and typically include the following:

- plasma's behavior is assumed to be axisymmetric,
- it can be described with a finite number of global parameters,
- plasma mass can be neglected,
- plasma's resistivity is assumed to be known.

The simplification of such non-linear physical reality appears unavoidable in the case of classical fusion plasma control. Moreover, the plasma's geometry and location can not be directly observed and are instead inferred in real-time from magnetic sensors using reconstruction codes [36, 20]. This overall setup (Figure 1.7) requires substantial engineering effort to tune those classical controllers whenever control objectives undergo variations, and show limits to the coupled behavior of plasma dynamics, i.e. magnetic equilibrium is strongly influenced by other physics such as transport. Indeed, linear control laws are suitable for holding stability in a narrow operating range within known targets, but non-linear control may be required for more advanced exploration. Advanced methods such as Multi-Input-Multi-Output plasma control [78], Linear Quadratic Regulator [95], or H_∞ approach [89], have been explored in many ways. Nevertheless, most functional solutions

[3]Tungsten (W) Environment in Steady-state Tokamak

tend to be linear, and the related model reduction ultimately leads to costly controller synthesis.

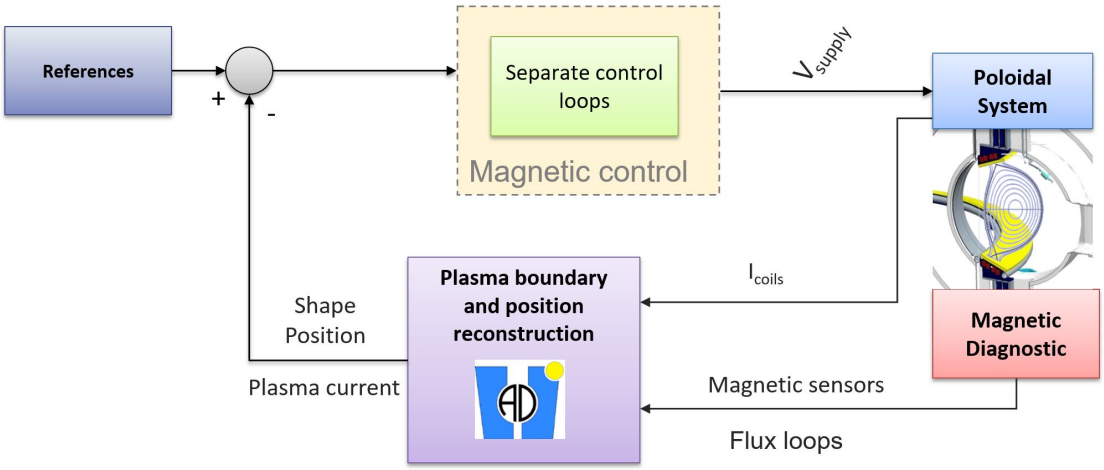


Figure 1.7: The magnetic diagnostics (probes and flux loops) are used to reconstruct the magnetic equilibrium, with a defined boundary and position. A feedback loop treats the obtained information to send the next voltages to the poloidal system.

No matter the method, scientists rely on these tools to study the effects of various configurations on plasma dynamics, such as elongated shapes and their related vertical instabilities [89, 43, 125, 25]. Therefore, there is an essential need for flexibility, adaptability, and robustness of magnetic control systems throughout the device's lifetime, without which no stable plasma could be sustained.

1.1.3 Machine Learning and company at the rescue

An interest arises with *machine learning*, a subfield of artificial intelligence. Any phenomenon, whether artificial or natural, is governed by deterministic or stochastic dynamics. All data related to these dynamics are generated through sampling from an underlying unknown distribution. If known, it would mean that we could explain the said system almost entirely, which is not the case in most situations. Machine learning aims to model the latter from related data and generalize it afterwards to new occurrences of the same phenomenon. In a sense, it enables the premise of learning without being explicitly programmed to do so. While using neural networks as powerful function approximators, the ability to efficiently address the nonlinearities of tokamak physics becomes a cornerstone of this investigation. Indeed, they could directly learn a mapping from magnetic measurements to the desired actuation. A number of applications have already been identified by the fusion community [57], and discussions on their various aspects are intensifying. Even more precisely, *reinforcement learning* is an exciting paradigm within machine learning, given its intrinsic connections with classical control. In the latter, an agent interacts se-

Supervised	Principle	Correct answers are given by labels in the dataset
	Example	Is it a cat ? → Yes: Wrong! The right answer was dog. Is it a dog ? → Yes: Right! Keep going.
Unsupervised	Principle	No known answers affiliated to each sample
	Example	Is it a cat ? → Perhaps, let's gather similar ones. Is it a dog ? → I don't know, just looking for similarities.
Reinforcement	Principle	Answers are given by the reward and the environment
	Example	Is it a cat ? → Yes: Wrong! You receive a negative reward (-1). Is it a dog? → Yes: Right! You receive a positive reward (+1).

Table 1.1: A comparison between the different learning approaches: Supervised (S), Unsupervised (U) and by Reinforcement(R)

quentially with an environment through actions and uses information sent back by the second. This is composed of a state, which informs how the environment evolved from one timestep to the other, and a reward. The latter is a core concept as it formalizes the control targets under a simple scalar function, which can combine multiple objectives. In reinforcement learning, data are classically from the interaction with the environment, which differs from supervised learning, i.e. the most common learning paradigm, which uses a fixed dataset (Table 1.1). The choice for an approach is motivated by the fact that accessing complete datasets of magnetic measurements and related actions on WEST was not evident at some point in time. At the same time, models of plasma evolution were readily available for the environment's definition. Implicitly, the fusion of reinforcement learning and neural networks, better known as *deep reinforcement learning* (Figure 1.8), could overcome the issues of linear plasma control, as well as enabling the use of only one feedback loop in the magnetic control system (Figure 1.9). To summarize such interest, we can interface it with previous statements regarding classical control, identifiable on the vast majority of existing tokamaks:

- Multiple control loops are usually required. With deep reinforcement learning, we are content with a single function grouping the targets set as a scalar.

- Conventional controllers usually require each control loop to be optimized separately, based on linear models and assumptions about the physical system involved. Advanced methods exist to fine-tune them effectively, but we still have multiple controllers to achieve stable magnetic control. In deep reinforcement learning, we have a joint solution for the non-linear control problem and closed-loop stability, i.e., the solution performs non-linear feedback control without multiple control loops to achieve overall stability.

- Precise knowledge of the involved system and its dynamics is usually needed to effectively model each part of the control problem. This requires an in-depth understanding of the underlying dynamics. In deep reinforcement learning, domain knowledge is in the environment, and we must only define the reward accordingly. No assumption is made on the latter, and even if domain expertise could be valuable, reinforcement learning relies on trial and error without any strict requirement for system identification.
- No need for reconstruction codes since the agent can directly learn a non-linear mapping from raw magnetic measurements to predicted actions, instead of mandatorily relying on inferred quantities like classical controllers.

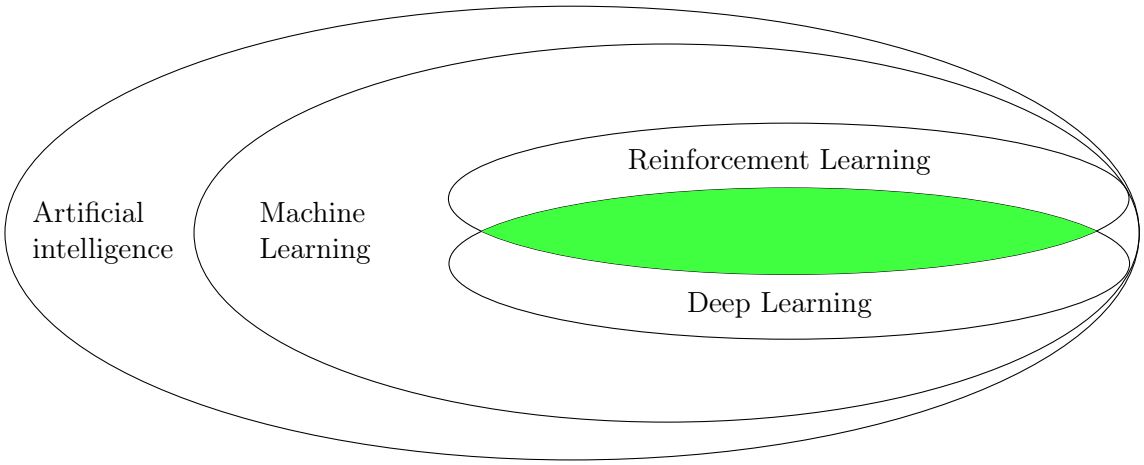


Figure 1.8: The russian dolls of modern artificial intelligence.

Consequently, deep reinforcement learning is becoming increasingly popular among the tokamak control community. For example, it has been used for model-based control [22], for the stabilization of vertically unstable plasmas [32, 27], to build feedforward trajectories of plasma parameters [107], for temperature and profile control [122, 123], or even for disruption avoidance [106]. Recent works [29] designed a reinforcement learning-based system which achieved magnetic control of the *Tokamak à Configuration Variable* (TCV) in Lausanne, Switzerland. The learned controller demonstrates the capability of the RL-based approach to tackle various plasma configurations of different complexities, while simultaneously tracking many quantities of interest. These examples highlight a shift of focus from classical plasma control, designed using a priori knowledge on how control should be performed with respect to the physical dynamics of the device, to controllers learning by trial-and-error to act on the system following what should be achieved in terms of final targets. Although reinforcement learning has many advantages, it comes with several pitfalls. Indeed, such a form of trial-and-error interaction requires the existence of realistic simulations, which are not necessarily guaranteed for all types of control scenarios.

Similarly, the paradigm does not benefit from deterministic measures of stability and robustness, as might be understood in the sense of classical control. These issues will be discussed throughout the manuscript, notably in chapter 2. Therefore, we have defined the context in which this thesis evolves, justifying the desired approach. The scope of this work is then to explore the use of deep reinforcement learning for magnetic control of a tokamak and assess the efficiency of this alternative path. A description of the main contributions follows in the next section, as well as the outline of this manuscript.

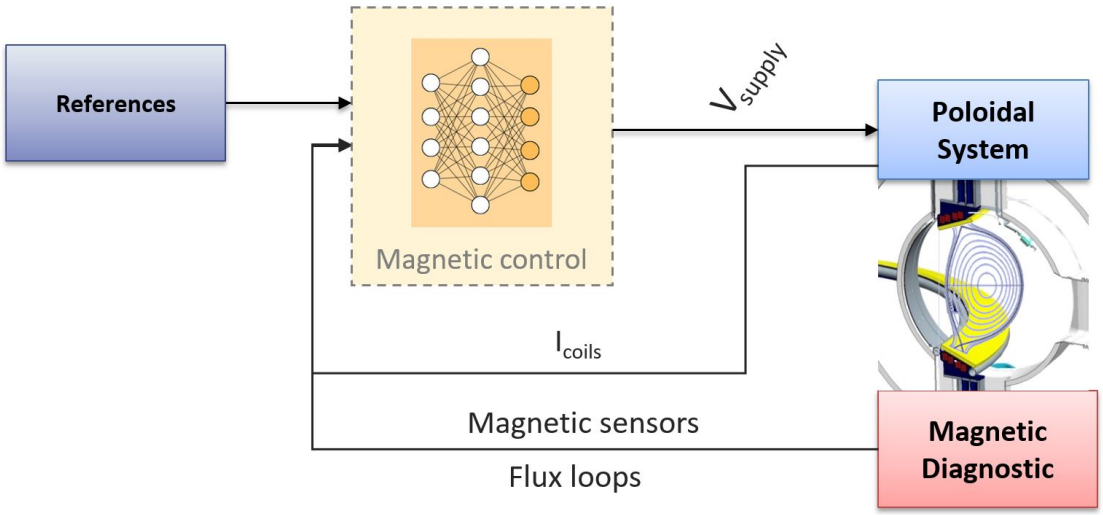


Figure 1.9: The diagnostics are directly fed with references to a neural network trained with reinforcement learning. As a result, no reconstruction codes are needed.

1.2 Objectives and main contributions

This PhD aims to study the use of deep reinforcement learning in the context of magnetic control of fusion plasmas on the WEST tokamak located in France. The previously developed state-of-the-art led to the first practical demonstration of how an RL-based system could shape, modify, and position the plasma. This study's outcomes opened several questions related to its reproducibility on other machines and how we could perform the related training efficiently. The availability of tools enabling such an approach is not identifiable in a straightforward manner despite its description in the related work, and it is then necessary to develop a general framework that would act as a strong basis for further enhancements. Consequently, the first objective is to reproduce the results from the TCV experiments in Switzerland, as it would prove that the method works indifferently of the device. The second objective is to find means of making reinforcement learning part of the routine operations on WEST, especially by speeding up training, as the initial proposal

exhibited training times lasting for several days. This is not in line with the flexibility and adaptability sought in the fusion domain, with operational controllers obtainable much more rapidly from one experimental campaign after the other. As we will see later on, the system is meant to learn only one scenario at a time, which limits the expressiveness of the controller towards handling multiple scenarios at once. This last concept will be briefly discussed as part of the perspectives, knowing that reducing convergence time is already a first step in this direction. A final objective rests on presenting the control performance of the RL-based solution for several control scenarios of interest, as well as evaluating the developed extensions. Hence, this study will feature three questions linked to the previous questioning, from which the main contributions stem from:

Problematic 1: How can we use reinforcement learning for magnetic control on WEST ?

1. Setup a numerical twin of WEST which will be used as the environment
2. Identify the right implementation and dynamics of the *Maximum a posteriori Policy Optimization* algorithm used in this thesis
3. Develop PILOT, a general framework for training of RL-based control systems on WEST
4. Define proper control scenarios for reproduction of TCV's trials in the WEST use-case
5. Train controllers comparable to what was observed in the literature

Problematic 2: How can we make this method a routine part of tokamak operations?

1. Identify limitations of the initial method and potential bottlenecks including the reward definition
2. Speed up training using *curriculum learning*
3. Discuss the use of Progressive Neural Networks to fight against catastrophic forgetting
4. Present *knowledge distillation* as a tool to make the latter in line with real-time constraints

Problematic 3: How can we assess the performance and effectiveness of reinforcement learning methods?

1. Analyze how reinforcement learning performs in the presence of disturbances, and with different available modes in the numerical twin

2. Analyze the need for integral effects which are mathematically needed for plasma current control

3. Evaluate the training speed-up offered by curriculum learning

1.3 Outline

This thesis is described as follows:

- Chapter 2 defines the path towards the algorithm used in this PhD, namely the *Maximum a posteriori Policy Optimization* (MPO). This will go through the basics of reinforcement learning, which exhibits strong connections with optimal control. Finding efficient behaviors for the agent can be done using various methods, but only actor-critic approaches are discussed. Finally, inference-based deep reinforcement learning is presented as the foundation of the sole algorithm used in this work. Even though numerous algorithms could have been applied in our context, this chapter concludes with the reasons why MPO is kept as a suitable choice.

- Chapter 3 contains the first main contribution in the complete presentation of the PILOT framework developed to answer the first problematic around the reproducibility of TCV's findings. Enabling the interaction loop inherent to reinforcement learning, it asynchronously connects flexible blocks. The WEST tokamak and its numerical twin are described at first. This is followed by a discussion on the notion of control scenario, and a schematic view of all the ones studied during this PhD is presented. A trajectory generator is showcased from these target discharges, yielding references used in the reward definition. Finally, a concrete explanation of PILOT's design is given, from communication protocols to the agent's specifications.

- Chapter 4 presents the remaining contributions which help answer the second problematic, namely improvements of the training procedure. An approach is proposed using curriculum learning to speed up training and improve final performance by several orders of magnitude. This incorporates existing attempts from the literature, notably hierarchical learning. Despite these improvements, the first implementation of the curriculum procedure displays potential issues regarding stability, as catastrophic forgetting might occur. This phenomenon is observed when the agent suddenly loses knowledge previously acquired. To overcome this problem, Progressive Neural Networks (PNN) could be leveraged as a prospective asset, at the cost of increased computational complexity. A brief discussion on knowledge distillation follows, which could help PNNs to meet WEST real-time constraints. These ideas pave the way towards an RL-based experimental routine on the WEST tokamak.

- Chapter 5 analyses the results related to each contribution. A qualitative and quantitative comparison is done on controllers trained with different modes of the WEST simulator, proving the need for integral control. Curriculum learning is analyzed and appears to be a valuable training procedure enhancement.
- Chapter 6 will conclude the study, summarizing the advantages and limitations of the paradigm of interest, as well as perspectives which could appear on short and long-term time scales, such as *Structured State Spaces* for interpretability of deep RL.

Inference-based reinforcement learning

Contents

2.1 Fundamentals 30
2.2 Connections with optimal control 34
2.3 Finding optimal policies 36
2.3.1 A taxonomy of algorithms 36
2.3.2 The road to the actor-critic's pantheon 38
2.3.3 Neural approximators 39
2.3.4 Exploring or exploiting: a core dilemma 43
2.4 An inference-based interpretation of deep RL 44
2.4.1 Introduction on probabilistic inference 45
2.4.2 Maximum a posteriori Policy Optimization 49
2.5 An agent ready for interactions 57

Reinforcement Learning (RL) [113] is a machine learning (ML) paradigm considered as an innovative approach to real-time control. Over the past decade, RL has been successfully applied to a wide range of problems, showing capabilities close, if not similar, to human-level skills. Examples could be drawn from games [109, 121] to control [65, 29] through natural language processing [23, 119]. Despite such achievements, it undergoes several constraints that must be put in perspective of classical control, especially in the context of robustness and stability assessment of controllers meant to act on critical plants. In this chapter, a thorough description of reinforcement learning is conducted to contextualize it formally. This aims first to define the fundamental notions behind reinforcement learning, then describe how the latter is closely related to optimal control and closed-loop systems. The next sections pave the way to deep reinforcement learning and the actor-critic algorithm used in this thesis. Finally, we will discuss how its interpretation aligns with the constraints of plasma applications.

2.1 Fundamentals

A classical RL framework defines a sequential decision-making problem in which an agent interacts with an environment formalized as a *Markov Decision Process (MDP)* denoted $\mathcal{M}$. This MDP is defined by a quadruple $\{\mathcal{S}, \mathcal{A}, P, \mathcal{R}\}$:

- a finite state space $\mathcal{S}$
- a finite action space $\mathcal{A}$
- its state transition distribution $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$
- the instantaneous reward function $\mathcal{R}(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

This environment is designed to represent the physical plant in the footsteps of classical control, with timestep $t \geq 0$. The environment's *state* is denoted as $s_t \in \mathcal{S}$, and sent to the agent with a *reward* $r_t = \mathcal{R}(s,a) = \mathbb{E}[r_{t+1}|s_t = s, a_t = a]$, as the *expected value* over the next reward. This scalar feedback signal indicates how well the agent performs with respect to the overall objectives, e.g. the reward hypothesis. It is a core concept in reinforcement learning as it defines what the agent should aim at rather than how it should do it, combining several constraints into a weighted sum, for example.

Definition (Reward hypothesis)**.** All of what we mean by goals and purposes can be well thought of as the maximization/minimization of the expected value of the cumulative sum of a received scalar signal (called reward/cost).

We will preferably aim at non-negative instantaneous reward functions, as they work better in many real-world problems like ours. As an example, we can picture a situation in which we would like to perform a turn with a car from point A to point B. We could set a reward of -1 at each timestep, implicitly asking the agent to perform the task as fast as possible. Since it wants to maximize G_t, the agent will try out possibilities to get the

episodic reward closer to 0, possibly leading to large negative cumulative rewards since the agent does not have knowledge of what a "good" trajectory is. Failing faster could avoid such accumulation, with a return lower than what it could have been if more attempts had been performed. In essence, this myopic behavior follows our objective but is not set correctly in case of actual deployment of the device on the road. That is why the reward is even more critical: if it is too simple, the signal will not be expressive enough to learn the task, and if it is too descriptive, it will guide the agent without letting it explore possible outcomes.

Consequently, the agent sends back *actions* a_t to the environment, and the latter evolves to a new state with transition $P(s_{t+1} = s'|s_t = s, a_t = a)$. More precisely, actions are set according to a *policy*:

$$\begin{aligned} \pi \colon \mathcal{S} \times \mathcal{A} &\longrightarrow [0,1] \\ (s,a) &\longmapsto P(a_t = a | s_t = s) \end{aligned}$$

It is worth noticing that a policy can also be deterministic, associating the same action with probability 1 for each observed state, i.e. $\forall s \in \mathcal{S}, \pi(.|s) = 1$. One could observe that the decision is performed solely based on current knowledge, which is a direct consequence of the Markovian property of the environment, which states that the system's *future is independent of the past given the present*, i.e. the next state only depend of the current one. The hypothesis becomes wrong if the state does not contain all the available information or if transitions depend on time, for example.

Definition (Markov property)**.** The state transition has the Markov property if and only if

$$P(s_{t+1}|\tau) = P(s_{t+1}|s_t, a_t), \text{ with } \tau = (s_0, a_0, ..., s_t, a_t)$$

An episode is a trajectory $\{(s_k, r_k, a_k)\}_{0 \leq k \leq T}$, which has started from state s_0 sampled from initial distribution P_0, and has reached a terminal condition at timestep T. Meant to model constraints on the MDP, their trigger resets the environment to its initial conditions while usually giving a strong penalty.

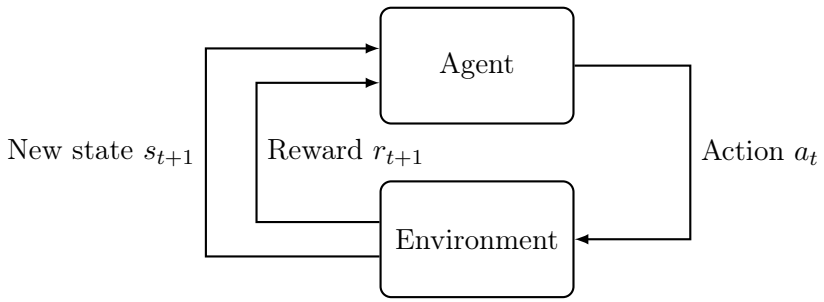


Figure 2.1: Interaction loop between the agent and the environment.

Through this feedback loop (Figure 2.1), the goal of RL is to make the agent learn an optimal policy $\pi_* : \mathcal{S} \to \mathcal{A}$ which maximizes the discounted cumulative reward over time, or expected return:

$$\pi_* = \arg\max_{\pi} \mathbb{E}_{\tau \sim P(.|\tau)}[G_0]$$

with $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, and the discount factor $\gamma \in [0, 1]$. This parameter is useful for considerations on finite horizon formulation since it starts from an infinite sum. Also, it penalizes long-term rewards since interest in a specific task tends to decrease over time. From behavioral neuroscience, we focus on close rewards, compared to those that arrive later. However, we still have to account for the long-term consequences of our actions. So, the choice of γ model such balance, to specify the agent's strategy over the course of a trajectory. It is important to notice that this does not relate directly to learning or prediction, in that it only refers to how we evaluate the reward impact in time from the observed interactions. One question arises about how to use the discounted cumulative reward over time in a practical manner. The *Value function* $V_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$ and the *Action-Value function* $Q_\pi(s, a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$ comes into play as fundamental definitions, closely related to the maximization problem previously stated. Those functions can be understood as an assessment of the quality of being in state s, and the quality of taking action a while being in state s. Both functions express a direct relationship between a state's value and its successor states' values. The following development for V_π is easily generalizable to Q_π:

$$\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi[G_t|s_t = s] \\
&= \mathbb{E}_\pi[r_t + \gamma G_{t+1}|s_t = s] \\
&= \mathbb{E}_\pi[r_t] + \mathbb{E}_\pi[\gamma G_{t+1}|s_t = s] \\
&= \sum_a \pi(a|s_t) \sum_{s'} P(s'|s, a)\mathcal{R}(s, a) + \sum_a \pi(a|s_t) \sum_{s'} P(s'|s, a)\gamma V_\pi(s') \\
&= \sum_a \pi(a|s_t) \sum_{s'} P(s'|s, a)[\mathcal{R}(s, a) + \gamma V_\pi(s')]
\end{aligned}$$

We obtain the *Bellman equations*, which express that the value of a state (respectively state-action pair) is equal to the immediate reward for that state and the value of adjacent states for all actions. By adjacent, we mean accessible regarding the transition distribution P from the particular state of interest.

Definition (Bellman Equation)**.**

$$\begin{aligned}
V_\pi(s) &= \sum_a \pi(a|s_t) \sum_{s'} P(s'|s, a)[\mathcal{R}(s, a) + \gamma V_\pi(s')] \\
Q_\pi(s, a) &= \sum_{s'} P(s|s, a)[\mathcal{R}(s, a) + \gamma \sum_{a'} \pi(a'|s')Q_\pi(s', a'))] \\
V_\pi(s) &= \mathbb{E}_\pi[Q_\pi(s, \pi(.|s))]
\end{aligned}$$

Now, we can directly express how two policies could compare each other:

$$\pi \geq \pi^{'} \text{ if and only if } \forall s \in \mathcal{S},\ V_{\pi}(s) \geq\ V_{\pi'}(s)$$

The notion of optimality follows as $\forall s \in \mathcal{S}, \forall \pi,\ V_*(s) \geq\ V_{\pi}(s)$, with $V_*(s) = \max_{\pi} V_{\pi}(s)$ and $Q_*(s,a) = \max_{\pi} Q_{\pi}(s,a)$ accordingly. This would require a search among all, possibly infinite policies, which is impractical. Hence, there is a need for a tool capable of checking if the value function and the policy are optimal. The *Bellman optimality equations* are then meant to show that the value of a state under an optimal policy is equal to the expected return for the best action from that state. Again, all developments for V_* are easily generalizable to Q_*:

$$\begin{aligned} V_*(s) &= \max_a Q_*(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[r_t \gamma G_{t+1} | s_t = s, a_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[r_t + \gamma V_*(s_t + 1) | s_t = s, a_t = a] \\ &= \max_a \sum_{s'} P(s'|s,a)[\mathcal{R}(s,a) + \gamma V_*(s_t + 1)] \end{aligned}$$

From a computational perspective, it exhibits an intuitive way of computing the value (respectively the action-value) of any state if we know P. Specifically, if we have found an optimal value function, then acting greedily with respect to it (i.e. choosing the best action in a particular state) defines the optimal policy. That is why basic algorithms such as *Policy and Value iteration* [113] recursively exploit the structure of the Bellman equations and the value function to find an optimal problem for a given MDP. However, as stated before, it relies on an exhaustive search of all possible outcomes, which would result in expensive computation to find all the related probabilities of occurrences and expected rewards. Such reality intuitively leads to an overall idea: we could find a way to approximately solve the Bellman optimality equations. This is the purpose of many RL methods, like the actor-critic approach discussed in the next section, relying only on sampled experiences instead of concrete knowledge of transition dynamics.

Definition (Bellman Optimality Equation)**.**

$$V_*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a) + \gamma V_*(s'))$$

$$Q_*(s,a) = \sum_{s} P(s'|s,a)(R(s,a) + \gamma \max_{a'} Q_*(s',a'))$$

Overall, Bellmann equations and their optimality counterparts enable the evaluation and optimization of policies to solve the MDP for π_*. The Bellman Optimality equations guarantee convergence of classic iterative algorithms such as *Policy and Value iteration* [113], which have been mentioned previously. These methods are defined in the presence of a perfect MDP, meaning we would know the transition dynamics in advance. However,

the vast majority of real-world applications does not exhibit knowledge of the probability transition function straightforwardly. So called model-based methods would focus on learning a representation of the probability transitions before planning optimally. Nevertheless, this PhD focuses entirely on model-free methods, where there is no knowledge of P and $\mathcal{R}$ (no perfect reward known beforehand). Based on this idea and starting from Bellman principles, two general principles are discussed and shared among many, if not all, modern RL algorithms;

- since $V_\pi(s) = \mathbb{E}_\pi[Q_\pi(s,a)]$, $Q_\pi(s,a) > V_\pi(s) \Rightarrow$ action a is better in average. So we would modify π to increase the probability of good actions by acting on gradients of the value functions;
- if we have a policy π and a known $Q_\pi(s,a)$, we can improve the first by looking at the action that maximizes the second. The obtained policy will always be at least as good as the initial one.

A comparison must be made to properly understand the advantages of RL over classical control. The next section gives a high-level explanation of the latter.

2.2 Connections with optimal control

Optimal Control (OC) allows to find a control trajectory u over a dynamical system defined by a state space $\mathbb{S}$, that optimizes an instantaneous cost $l(x,u)$, with $x \in \mathbb{S}$ [35]. Control design then specifies the dynamics model and the desired constraints that align with the overall objectives. For example, one would like to build an optimal driving trajectory towards a desired location while combining several other goals, such as maintaining a comfortable driving behavior for human passengers, or avoiding dangerous obstacles. The cost must be non-negative and penalize undesirable states. Moreover, it is usually described as a weighted sum of individual costs, each scaling the contribution of a sub-task (tracking, effort, etc) to the entire control objective.

It shows a shift of focus with classical control, from considering what the control should do rather than designing how it should do it, which is precisely what is intended by RL. Indeed, classical controllers are designed mainly with graphical tools (Bode plots, Nyquist, etc). Laplace transforms and linearization give a simple and intuitive approach to computing, analyzing, and tuning controllers without extensive computations. That is why most real-world applications usually count on PID control [8], as it is sufficient to achieve a particular system's behavior (overshoot, steady-state error, etc.) by focusing on a desired response. However, many of those applications could benefit from more advanced and nonlinear approaches, that look for strong performance under explicit constraints that classical control struggles to incorporate. Optimal control tries to answer these questions more robustly at the cost of computational efficiency.

A critical idea appears as MDPs are discrete stochastic formulations of optimal control problems [113]. However, RL is mainly described as a *closed-loop* system where the policy is learned by feedback from the environment. Oppositely, optimal control is primarily seen

as an *open-loop* planning problem. Such difference implies that the first might benefit from real-time information, while the second might undergo modeling errors from its dynamics model. A closed-loop optimal control formulation could be implemented using Model Predictive Control (MPC)[92]. This can be solved online and recursively by *Dynamic Programming* (DP)[12], which is at the basis of RL. Indeed, policy and value iteration were mentioned as algorithms used to find an optimal policy in the presence of a perfect MDP. They are derived from DP and are initially a way of solving MPC using the said Bellman equations. Reinforcement learning then appears as a stochastic interpretation of the latter, which does not need system identification. A concise formal study of these connections can be found in Appendix A and in-depth resources in [14]. This might help readers from both fields to connect their respective knowledge, even if it is not directly related to the subject of this manuscript.

Aside from differences in formalism, it is essential to question the overall philosophy emanating from the two domains, which has not been studied in many works [46, 19]. Historically, control has always focused on loop stability. More precisely, it looks at maintaining a bounded system from its initial conditions to all of its future evolution, and state convergence towards a finite set of values. However, RL focuses more on reward performance, and only learning convergence towards a near-optimal solution is analyzed, either asymptotically or quantitatively via ϵ bounds, i.e. $G_\infty \geq \epsilon$. Characterization and deviation from the said bounds in the sense of closed-loop control and bounded systems are often eluded. It mainly comes from the relatively harmless nature of numerous applications of RL, like games or ones where the physics properties of the environment can be bounded heuristically. The discounted rewards themselves do not need any meaning apart from being related to the problem definition. This can differ from optimal control where unstable states would reflect arbitrarily large costs-to-go, i.e. costs usually express stability concerns, while even RL terminal conditions do not follow such rigorous pattern on bounds validation. Most importantly, control practitioners start from deterministic settings to get an initial controller, assuming some knowledge about the system and adding uncertainties through robust control tools (Lyapunov stability, etc). This approach allows for a refined definition of stability guarantees since the system's behavior is studied beforehand. On the contrary, RL discards any information regarding the system's dynamics and starts directly from a stochastic perspective, making the deterministic computation of stability bounds challenging. Because of that, RL stability is still an open area of research, and many algorithms' viewpoints aim at proving convergence rather than stipulating stability bounds as expected from control applications. This PhD will mostly rely on a statistical analysis of the control obtained after training, which remains classical regarding RL research.

To conclude on the links between OC and RL, both are sequential decision-making problems designed on constraints and costs, with similarities but notable differences in their inherent properties and conventions. In a sense, RL might be seen as online adaptative optimal control that directly uses available information like real-time measurements without relying on a deep understanding of the involved dynamics [71]. This statement will become especially important in fusion, as complete understanding of plasma dynamics and transition probabilities is yet to be commonly defined, as well as access to all possi-

ble information during tokamak operations. We must now unravel how to explicitly find (near-)optimal policies in a model-free setup since real-world applications usually do not exhibit MDP dynamics straightforwardly.

2.3 Finding optimal policies

2.3.1 A taxonomy of algorithms

In modern reinforcement learning, classifying existing methods to find optimal policies would give an extensive set of options. Considering our use case, we need to contextualize our focus to justify the usefulness of the algorithm used for this study. Let us recall that an RL problem is framed as a *Markov Decision Process* (MDP) denoted $\mathcal{M}(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$, and we look for a stochastic policy π which maximizes the expected cumulative reward over time. As stated in a previous section, knowledge of P and $\mathcal{R}$ is a decisive characteristic which allows to partition algorithms into two major families:

- *Model-based* methods: P is estimated and used afterwards for planning, or to improve an already existing policy. If we can learn the MDP precisely, it converges to an optimal policy;
- *Model-free* methods: No knowledge P and $\mathcal{R}$ is assumed, and the entire procedure relies on trial-and-error.

Several precisions must be added when it comes to the knowledge of $\mathcal{R}$. For each MDP, there is no unique reward signal that might come from human intuition. Model-free algorithms rely on this assumption, as they do not try to estimate the optimal $\mathcal{R}_*$ in any way. Furthermore, model-based methods tend to be, in general, more difficult to achieve than discovering a (sub-)optimal policy without a model of the world. Likewise, this approach would perform control through planning, which might be too slow compared to the timescales of many, if not all, plasma events. Magnetic control applications motivate the interest in the model-free paradigm since we will not try to learn $\mathcal{P}$. Intuitively, this could also be related to the uncertainties in plasma simulations, which could form a strong bottleneck to learning the MDP precisely. In model-free methods, learning is performed through trial-and-error, where outcomes of the policy's actions are observed and then used to *reinforce* a policy, leading to better rewards.

The search for an optimal policy could be represented by three steps, namely *samples generation*, *policy evaluation*, and *policy improvement* (Figure 2.2). Samples are generated from the interactions with the environment and then employed to estimate the returns, which are finally used to optimize the policy. Traditionally performed online, it is worth noticing that offline RL exists and performs the very same training loop whilst using a predefined dataset of interactions. Despite offline RL gaining more interest among researchers, even in fusion [22], we stay within the scope of online methods, recalling the first objective of this PhD, i.e. reproducing the state-of-the-art [29] on a different device. Let us recall value and action-value functions $V^\pi(s)$ and $Q^\pi(s, a)$ for policy π, $s \sim P$ and $a \sim \pi(.|s)$.

It is worth mentioning that relying on $V^\pi(s)$ is difficult in many real-world applications such as fusion, since they do not exhibit proper knowledge of the probability transition function P. Because of that, making actions explicit is an interesting way of computing the expected return, as state-action pairs can be easily sampled throughout learning. For this reason, we will only mention the action-value function. Two ideas emerge out of considerations between these structures and the policy:

- if we know $Q^\pi(s, a)$, we can use it to improve π since they are intrinsically related;
- we could compute gradients of the policy, and performing gradient descent to increase the probability of good actions.

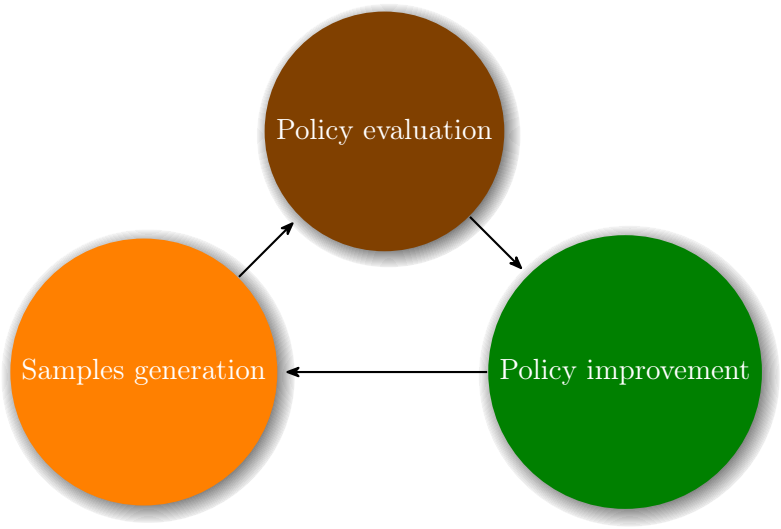


Figure 2.2: After generation of samples through interactions with the environment, returns are estimated before optimizing π accordingly.

Value-based methods define the first case: policy evaluation is realized while fitting Q^π, and the policy is easy to set, e.g. $\arg\max_\pi Q^\pi$. Since we optimize the value when learning the policy, value-based RL is closer to the implicit objective of the field. However, it may suffer from poor convergence when associated with function approximation, such as neural networks, when small value errors can lead to totally different actions, i.e. larger policy errors. The second case describes *gradient-based* methods: policy evaluation is done by computing G_τ on the entire trajectory $\tau = \{a_0, s_0, \cdots, a_T, s_T\}$, with $T > 0$. The policy is improved through gradient based optimization $\pi \longleftarrow \pi + \alpha \nabla_\pi \mathbb{E}_\pi[G_\tau]$, with learning rate α. Such formulation is directly linked to RL's real objective, which lies in learning a policy. Specifically, it makes working with continuous action spaces, or stochastic policies, easier since the related search space is smoother than the value one. Nevertheless, it can converge to a local minimum or knowledge so specific that generalization can not occur, resulting in poor performance. *Actor-critic* (AC) approaches combine the previous methods by

fitting $Q^\pi(s,a)$ for policy evaluation, and performing $\pi \longleftarrow \pi + \alpha \nabla_\pi \mathbb{E}_\pi[Q^\pi]$ during policy improvement. RL presents a huge number of approaches within these three families, which could then be classified depending on the nature of the action and state spaces (discrete or continuous), the stochasticity induced in the dynamics or in the policy, or whether it is easier to represent P or π. To conclude this taxonomy, two terms must be shown which relate to how the generated data for the environment is used to optimize the policy:

- *on-policy*: Algorithms characterized as such perform training with data related to the current policy only. This states that if the policy changes or gets updated, data has to be gathered immediately after;
- *off-policy*: These methods execute training with data from *any* policy, including previous ones. This increases sample efficiency, i.e. the number of examples needed to perform policy search, by reusing past experiences.

It is crucial to realize that looking for π using reinforcement learning is an entirely different task than learning the latter in a supervised way. This is especially true regarding training stability and convergence, which is not guaranteed when function approximation appears [41], or assumptions about how much of the environment's state we can observe. That is why recent advancements in AC methods are particularly interesting. Not only do they prove to leverage the use of nonlinear function approximation efficiently, but they also exhibit strong empirical performance in various domains [44]. We will focus on the off-policy actor-critic paradigm: it is known to have better sample efficiency, which helps regarding the computing timescales of plasma simulations, up to several hours for an episode; the strong results of those approaches constitute the state-of-the-art in terms of continuous control, which is our case since actions performed on the tokamak as well as the environment's state are characterized as such, and in high dimensions.

2.3.2 The road to the actor-critic's pantheon

Actor-critic RL aims to combine value-based and policy-based methods in a unified framework. The actor learns to make decisions, and the critic advices on potential good actions. In this case, the actor is related to the policy π, while the critic refers to estimated returns and the value function. The general formulation of these methods [66] to optimize the policy is defined as such:

$$\nabla_\pi \mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \mathcal{P}(.)} \left[\sum_{t=0}^{T-1} \nabla_\pi log\, \pi(a_t|s_t) Q^\pi(s_t, a_t) \right]$$

for trajectory $\tau = \{s_0, a_0, \cdots, s_T, a_T\}, T > 0$. Actually, a more general expression is worth mentioning [104], unifying all views regarding gradient-based methods:

$$\nabla_\pi \mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \mathcal{P}(.)} \left[\sum_{t=0}^{T-1} \nabla_\pi log\, \pi(a_t|s_t) \Psi^\pi(s_t, a_t) \right]$$

where Ψ^π can be the total reward, the action-value, etc. More details are present in Appendix B. Often, actor-critic algorithms are on-policy like policy-gradient methods. Since we focus on off-policy settings, we directly refer to the proper formulation [30], in which previous experiences are stored in a dedicated data structure. In the case of RL using neural networks, we will see that this structure is referred to as the *Replay Buffer*. Samples are drawn from the latter, and an estimator $\hat{Q}^\pi$ of Q^π is fitted in a supervised manner. This is done by minimizing the squared loss between predictions from the estimator and target returns y_t computed using value-based notions. Summing over the samples set x, we end up with the following cost:

$$\mathcal{L}(\hat{Q}^\pi) = \frac{1}{2} \sum_{i=0}^{|x|-1} ||\hat{Q}^\pi(s_t, a_t) - y_t||^2$$

with y_t obtained through any valid target computation, such as estimates of the full return using bootstrapped values like *n-step bootstrap returns* [113]. This target computation finds its source in value learning, and a dedicated explanation of the latter can be found in Appendix B. Once policy evaluation is performed, policy improvement is done in a policy-based fashion by computing the gradients of $\mathcal{J}$ with respect to π. The following expression appears accordingly to perform the gradient optimization:

$$\nabla_\pi \mathcal{J}(\pi) \approx \frac{1}{N} \left[\sum_{i=0}^{|x|-1} \nabla_\pi log\, \pi(a_i|s_i) \hat{Q}^\pi(s_i, a_i) \right]$$

This method reduces the variance of policy-gradient methods. By that, we relate to how sensitive this estimator gets to changes in the cumulative return from two trajectories starting from the same state. However, it displays bias since the critic is fitted and consequently not perfect, i.e. n-step estimates might be inaccurate as the initial values could be ill-defined. Modifying the n hyperparameter in the n-step bootstrapping operation helps mitigate the bias-variance trade-off: the bigger n gets, the bigger the variance follows, lowering bias. This concise explanation of the actor-critic algorithm serves as a basis for understanding that it addresses the flaws of value-based and policy-based ideas. The actor and the critic can be parameterized by neural networks, which could share weights, displaying costs and return estimates in various ways [44].

2.3.3 Neural approximators

Most methods we discussed mainly considered finite state and action spaces. However, when RL is applied to real physical systems like a tokamak, $\mathcal{S}$ and $\mathcal{A}$ are mostly continuous. Because of this, there are no proper ways of defining exact and closed forms of V, Q, or π, and discrete approaches tend to create tabular representations of the latter. Function approximation constitutes an efficient tool to turn these continuous learning problems into tractable instances while optimizing for a finite amount of parameters. Neural Networks (NN) are an unmissable type of architectures, working as general function approximators (Figure 2.3). Drawing inspiration from the brain, they are made of layers composed of

neurons, which were first modeled as individual linear approximators [76]. Consequently, connections between neurons and layers are represented by matrices of weights optimized by gradient descent, using a cost function that depends on the objective (mean squared error, cross-entropy, etc). This cost function must be differentiable to compute its gradients with respect to learnable parameters, i.e. the weights of the neural network, which are *back-propagated* through the network [96]. When training such architectures, one can perform gradient descent in different ways (Figure 2.4).

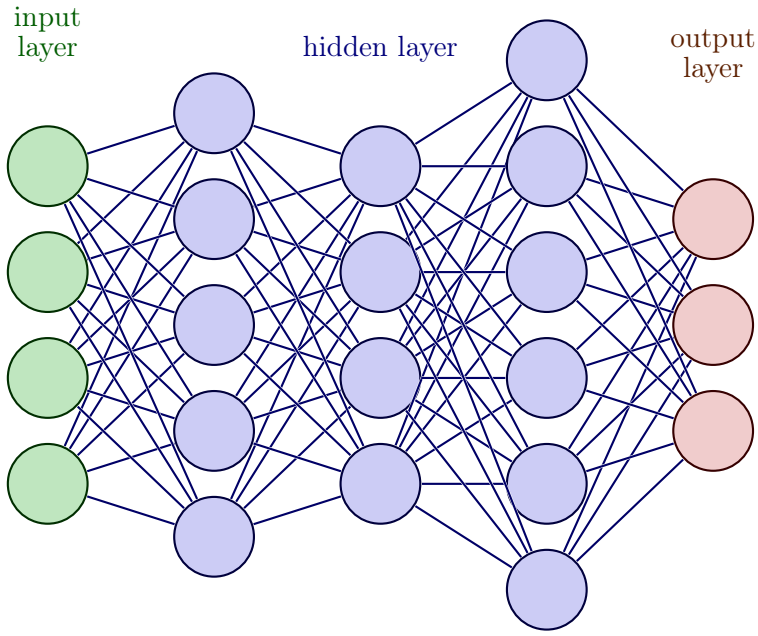


Figure 2.3: A simple fully connected neural network. It can have many hidden layers, each one with a different amount of neurons. Bias in each neuron is not represented here.

In the *stochastic* method, training is done one example at a time, meaning that the neural network would compute the cost related to a single output, look for its gradient, and optimize the weights accordingly. This approach is swift but suffers from strong oscillations and noise in the gradient information. Conversely, the *batch* method considers the entire dataset when computing the cost, which significantly stabilizes the overall scheme, at the cost of training times that can become intractable. Finally, the *mini-batch* idea helps to find the right balance between the previous methods by performing gradient descent using subsets of examples. Like numerous works, we will rely on the last one to perform neural network optimization. The *learning rate* of the gradient descent is a hyperparameter which can decay over time, or be adaptative [94]. Deep learning arises when multiple layers are stacked to increase the depth of the neural network. Nonlinearities are introduced between each layer in the form of activation functions [33], taking inspiration

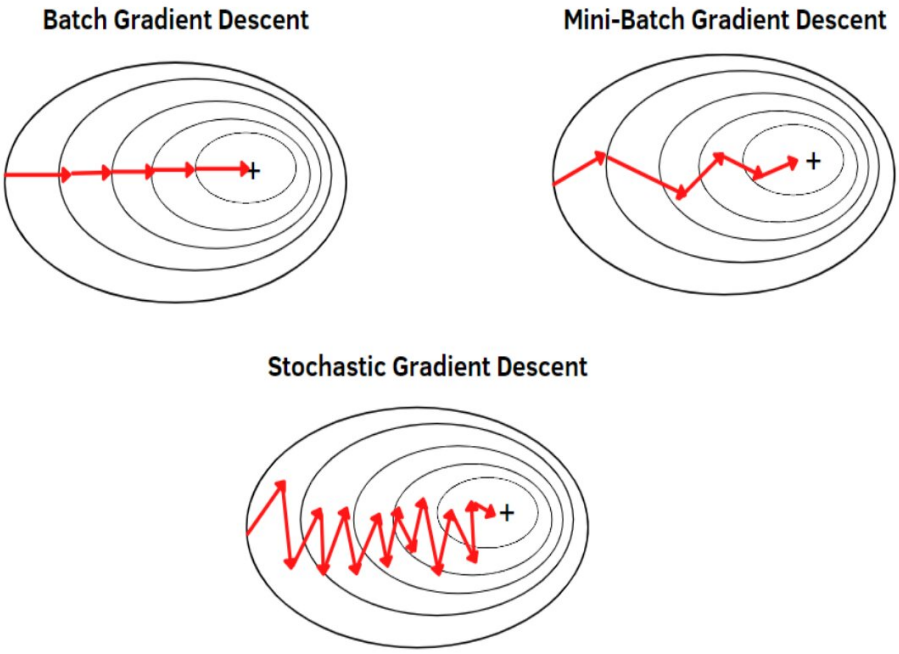


Figure 2.4: Gradient descent can be done either one example at a time (stochastic), considering the whole dataset available (batch) or considering a sampled subset of the latter (mini-batch).

from what we observe in the biological world. Assuming the smoothness of the function to be learned and considering that the latter can be decomposed into a composition of simpler functions, neural networks have proven to be very efficient when dealing with many types of data (time series, images, etc.). They exist in many forms nowadays, with architectures ranging from fully connected neural networks, to recurrent ones improving learning on sequences [108], through convolution networks efficient with images [87], or even transformers architectures at the basis of large language models [120]. We will focus entirely on fully connected neural networks, namely *Multi-layered perceptrons* (MLP), and Recurrent Neural Networks (RNN) in this manuscript. Indeed, MLPs are sufficiently expressive to control a plasma and small enough to fit on a control system. On the other hand, RNNs are structurally better at handling sequences of information. This is important in our use case since the data from the developed environment comprises multivariate time series. Such a network keeps an internal state that is unrolled over time, computing outputs for various sequence lengths, which is opposed to MLP where the input size is fixed. Their flexibility allows for many applications ranging from time series forecasting to language translation. Indeed, they can output one or multiple values per sequence. However, they might suffer from exploding and vanishing gradients, which raises a risk of non-convergence [88]. Solutions have been developed by modifying the computation of the hidden state, like the *Long-Short-Term-memory* unit (LSTM) [54], or the *Gate recurrent*

unit [24]. Despite many advantages, and no matter what the architecture is, deep learning applied to RL has several downsides:

- NN are universal approximators [102, 74]. Because of that, the space of potential functions towards which an NN can converge easily becomes intractable. Combined with RL and its non-stationary distributions, optimization and training are inherently noisier than supervised learning;
- because of the previous point, gradients must be representative of the entire state space without being biased towards a subset of the latter. Nevertheless, even if gradients are averaged through the mini-batch procedure where samples are supposed to respect state coverage, it might end up in exploding or vanishing gradients without any meaningful information for back-propagation;
- we mentioned on several occasions the mini-batch approach, in which the batch size becomes a hyperparameter of the training procedure. Moreover, the number of layers, the number of neurons per layer, and the chosen nonlinearities are just a small subset of hyperparameters that must be fine-tuned for each use case. Combined with large architectures' learnable weights, NN complexity becomes tremendous. Even if many RL algorithms have been benchmarked [7], they still have to undergo hyperparameter search, which can be even more expensive with the addition of neural architectures.

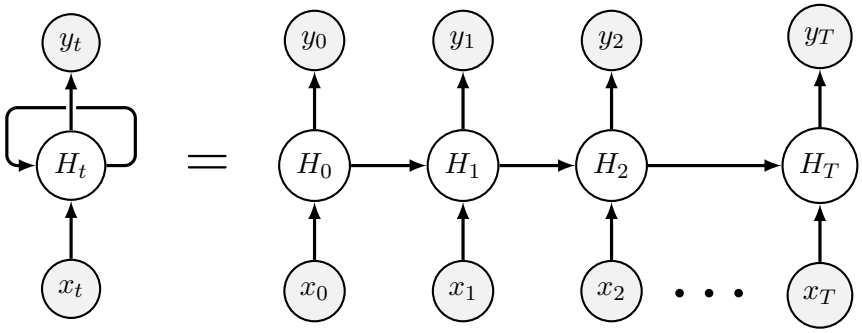


Figure 2.5: A recurrent neural network is unrolled in time, displaying the hidden unit H, input sequence x and outputs y.

In theory, deep RL seems like the perfect combination, but many pitfalls line up in practice as RL exhibits nosier learning dynamics. A few methods help stabilize the procedure and sort out the best out of these nonlinear predictors. First, n-step bootstrap returns tend to be a preferable choice for deep RL, especially regarding neural networks [79]. It makes the learned action-value function, i.e., its predictions, uncorrelated with the computed targets to stabilize action-value fitting. Many other approaches were studied [103, 19], out of which a few examples can be drawn:

- *Target networks*: The value function is define as a neural network. Its learning is set as a supervised problem using bootstrapped estimates for targets. Then, the objective is to minimize the error between the predictions and the said targets. However, both elements are highly correlated, which could cause divergence [80]. The value network is copied to overcome such an issue, creating a *target* value network. Its weights are periodically updated by retrieving those of the online and optimized network. This target network is used logically to compute targets, uncorrelating them from the predictions. This idea is extended trivially to policies parameterized by a neural network, uncorrelating actions similarly.

- *Experience replay*: Consecutive experiences from interactions with the environment are stored in a fixed-size buffer [37]. Old experiments are erased sequentially in a *first-in-first-out* manner (FIFO), with each entry designed as a tuple of the form $(s_t, a_t, s_{t+1}, r_t) \forall t > 0$. Batches are sampled randomly, removing temporal correlations. It is particularly well suited for off-policy settings and ensures that if a sudden policy change occurs, the gradients will evolve slowly, improving convergence and sample efficiency.

- *Normalization*: In a neural network, the scale of the outputs of each layer might be tiny, from which gradients will not be informative enough to update the NN's weights properly. Normalization is then required and concerns many parts of the neural network. Where possible, we normalize inputs to the range [-1,1] for all features to remain at the same relative scale. Outputs of each hidden layer are normalized through dedicated operations, that is to say, *Batch* or *Layer* normalization layers [58, 11]. We will use only the second option, as it empirically gives better results with the algorithm considered for this work [38]. Gradients can also be clipped to prevent catastrophic updates of the NN's weights and augment training speed [136].

All these elements will be used extensively during this PhD. Whether neural networks are used, one core element of RL has not been presented yet. The notion of exploration and its dilemma with exploitation are presented in the next section.

2.3.4 Exploring or exploiting: a core dilemma

We saw that online RL techniques are twofold: a sample-based version of dynamic programming where transitions obtained by a fixed policy are used to approximate the value function, and then the policy is improved. The second method involves directly optimizing the (parameterized) policy from observed data without any direct considerations of the value function. AC methods lie at the crossroads of these two paths. In the context of value-based methods and Q-learning, we know that convergence is asymptotic to Q^{π_*} if $\sum_{t=0}^{\infty} \alpha_t^2$ is finite, and $\sum_{t=0}^{\infty} \alpha_t$ is infinite, i.e. all state-action pairs are visited infinitely often [113]. This second statement dictates a core element of reinforcement learning, namely *exploration*: all states must be reachable by available actions. Thus, a policy should explore actions and increase state space coverage as much as possible. This explains the

interest in stochastic policies, as exploration mechanisms are inherently part of the sampled actions. We can picture a situation where two outcomes are accessible to an agent. The first one would always give a reward of 5. In contrast, the second one would give a reward of -2 with a non-zero probability of giving 20 occasionally: the agent must explore the landscape of possible outcomes to understand that exploiting the first path is beneficial, but exploration of the second option is sometimes needed to maximize the gains. We can extend this *exploitation-exploration* dilemma to any method, as generalization is enhanced through the process. One famous example comes from the notion of *greedy* interaction we mentioned earlier, which goes for the best action while acting on the environment. The most classical way of handling exploration follows through the ϵ-*greedy* strategy:

$$a_t \sim \pi(.|s_t) = \begin{cases} \arg\max_{a \in \mathcal{A}} Q^\pi(s_t, a_t) & \text{with probability } 1-\epsilon \\ \text{random sampling} & \text{with probability } \epsilon \end{cases}$$

The action is chosen randomly with probability ϵ or taken as the highest Q-value with probability $1-\epsilon$. This ensures that the agent explores the problem while using its current knowledge of the best (greedy) behavior obtained so far. Such an exploration scheme is similar to an excitation condition in control [19]. Indeed, even if we do not have a proper model of the environment, the action-values embody an implicit scheme of the MDP's dynamics, and of the policy as a consequence. Excitation is needed to identify the optimal trajectory through the Q-function persistently, and any policy like ϵ-*greedy* can be applied to control the system. This dilemma can take many forms in more advanced actor-critic methods, from exploration bonuses in the reward design [116], to noise-induced in various parts of the MDP [56], through notions related to Bayesian RL [39] or probabilistic inference as we will see right afterwards. On another note, the discount factor γ that we have seen many times also influences the balance between exploration and exploitation. When close to 1, the agent is more likely to explore the possibilities to find long-term beneficial strategies. When tending towards 0, the agent may exploit strategies known to favor immediate rewards, potentially at the risk of missing long-term dependencies.

Now that the taxonomy of common RL approaches has been described and discussed through their core challenges, we need to link the method chosen for this PhD, already seen in the state-of-the-art, to the preceding ideas. This will be viewed through the prism of probabilistic inference that we present pithily.

2.4 An inference-based interpretation of deep RL

Viewing RL through the scope of probabilistic inference is not straightforward. Usually, RL is fundamentally different from the latter because it acknowledges stochasticity only in the sense of MDP dynamics, and the reward is only a signal determined during interactions with the environment. Reward design is indeed a field of research, with heuristics depending on the control problem to solve. But could we model the reward differently, accounting for probabilistic effects? Indeed, learning to match a distribution simultaneously on the dynamics and the reward is an appealing approach, which gives new ways of conceiving

exploration and flexible formulations for MDPs of many complexities. By that, we mean that in many applications, such as plasma control, the environment's state s would not always be entirely observable, and observations rely on a subset of $o(s)$. Inference-based tools provide a straightforward way of incorporating such context into learning dynamics, hence their usefulness in this research. The algorithm used in this PhD is intrinsically linked to this family of methods, so our first need lies in introducing probabilistic inference and variational approaches in the RL context.

2.4.1 Introduction on probabilistic inference

Learning problem

Let us define an MDP $\mathcal{M}(\mathcal{S}, \mathcal{A}, p, r)$. For the sake of simplicity, we restrict this study to a finite horizon problem with finite episode time T instead of an infinite sum relying on a discount factor γ. Considerations on the latter will appear when necessary. We place this discussion under a closed-loop formulation in a model-free setup, in which no knowledge nor estimation of p and r is assumed. Let us also remember that deep RL was chosen for this work, so the policy is parameterized by a neural network. In classical RL, the main objective can be described as:

$$\pi^* = \arg\max_{\pi} \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t)}[r(s_t, a_t)]$$

under trajectory distribution expressed by the likelihood:

$$\begin{aligned} p(\tau) &= p(s_1, a_1, \cdots, s_T, a_T) \\ &= p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \end{aligned}$$

The return is optimized in order to a get a good control policy, from interactions with the environment. However, nothing guarantees that the related data will always be optimal, and no considerations on the optimal control trajectory exist. By conditioning the reward and the transition dynamics on the latter, a probabilistic model could be built to perform control. Let us define a binary random variable $\mathcal{O}_t$ over the reward, which is conditioned to be true if the timestep is said to be optimal and set to False otherwise:

$$p(\mathcal{O}_t = 1|s_t, a_t) = exp(r(s_t, a_t))$$

This definition assumes that the reward is bounded, as it would produce negative $p(\mathcal{O}_t = 0|s_t, a_t)$ otherwise [70]. The choice of the exponentiated reward is common in many studies [38], even if a few examples used a different formalism [128, 86]. This random variable offers a refined interpretation of $p(\tau)$, now conditioned on $\mathcal{O}_T = \{\mathcal{O}_t, \forall t \in \{1, \cdots, T\}\}$:

$$
\begin{aligned}
p(\tau|\mathcal{O}_T) &\propto p(\tau, \mathcal{O}_T) \\
&= \left[p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t) \right] exp(\sum_{t=1}^{T} r(s_t, a_t))
\end{aligned}
$$

The first term relates to the probability of observing trajectory τ based on p, while the second defines the probability of τ being optimal as an exponential sum of rewards. Here, the optimal policy $\pi^* = p(a_t|s_t, \mathcal{O}_T)$ is different than what is usually depicted by the RL formulation π^*_θ. Indeed, our new objective has the optimal trajectory conditioning the distribution of interest, which does not depend on the parameterized representation.

Finding an optimal policy under probabilistic inference

Now that the learning problem is stated, the search for a policy must be discussed. To find $\pi(a_t|s_t, \mathcal{O}_T)$ we compute *backward messages* [69] of the form:

$$
\begin{aligned}
\beta_t(s, a) &= p(\mathcal{O}_T|s, a) \\
\beta_t(s) &= p(\mathcal{O}_T|s) = \int_{\mathcal{A}} \beta_t(s, a) p(a|s) da
\end{aligned}
$$

where $p(a|s)$ does not describe any optimal actions and only exists as a prior action distribution. Since we want to start in any possible state, $\mathcal{O}_T$ can be restricted to $\mathcal{O}_{t:T} = \{\mathcal{O}_t, \forall t \in \{t, \cdots, T\}\}$. These messages denote the probability of the trajectory being optimal if it starts in state s (and action a in the corresponding case). The backward computation is described by:

$$
\beta_t(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t) = \int_{\mathcal{A}} \beta_{t+1}(s_{t+1}) p(s_{t+1}|s_t, a_t) p(\mathcal{O}_t|s_t, a_t) ds_{t+1}
$$

Because of the Markovian property, $\pi(a_t|s_t, \mathcal{O}_{t:T}) = \pi(a_t|s_t, \mathcal{O}_T)$. From Bayes' rule [129], the optimal action distribution becomes:

$$
\pi(a_t|s_t, \mathcal{O}_{t:T}) = \frac{p(a_t, s_t|\mathcal{O}_{t:T})}{p(s_t|\mathcal{O}_{t:T})}
$$

If we assume the prior $p(a_t|s_t)$ to be uniform, the derivation leads to:

$$
\begin{aligned}
\pi(a_t|s_t, \mathcal{O}_{t:T}) &= \frac{p(a_t, s_t|\mathcal{O}_{t:T})}{p(s_t|\mathcal{O}_{t:T})} \\
&= \frac{p(\mathcal{O}_{t:T}|s_t, a_t) p(a_t|s_t)}{p(\mathcal{O}_{t:T}|s_t)} \\
&\propto \frac{p(\mathcal{O}_{t:T}|s_t, a_t)}{p(\mathcal{O}_{t:T}|s_t)} = \frac{\beta_t(s_t, a_t)}{\beta_t(s_t)}
\end{aligned}
$$

How does it relate to standard reinforcement learning? Actually, β_t in log-space corresponds to soft interpretations of Q and V, so that:

$$\begin{cases} Q(s_t, a_t) = log\, \beta_t(s_t, a_t) \\ V(s_t) = log\, \beta_t(s_t) \end{cases} . \tag{2.1}$$

By marginalizing over actions, and depending on how large is Q, we get:

$$V(s_t) = log \int_{\mathcal{A}} exp(Q(s_t, a_t))da_t \approx \max_{a_t} Q(s_t, a_t)$$

Finally, the backward computation becomes [69]:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma log\, \mathbb{E}_{s_{t+1} \sim p(.|s_t, a_t)}(exp(V(s_{t+1})))$$

Even if it looks similar, this is not the expected value displayed by many regular RL approaches. This is similar to a soft maximum of the expectation, which is an optimistic behavior. Such a situation is problematic, as the agent will take specific actions if it has a non-zero probability of achieving a high reward. Among all possible and likely actions, the agent will favor high value actions, even if they are risky or exhibit uncertainties. Said differently, the agent could go towards actions that have high value, if there is a non-zero probability of achieving good rewards. Now that we have described the inference framework and how it relates to classical views on RL, we must look for the objective optimized throughout learning. We will also try to display a more realistic behavior through a better update rule.

Objective definition

We want to find $\pi(a_t|s_t, \mathcal{O}_T)$ so that actions are conditioned on all the optimality variables. This is interesting because it matches situations where the Markov property does not hold. The distribution over the entire trajectory is given by:

$$p(\tau|\mathcal{O}_T) = \left[p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t) \right] exp(\sum_{t=1}^{T} r(s_t, a_t)) \tag{2.2}$$

We want to look for an approximation $\hat{p}$ close enough to $p(\tau|\mathcal{O}_T)$:

$$\hat{p}(\tau|\mathcal{O}_T) = \left[\hat{p}(s_1) \prod_{t=1}^{T-1} \hat{p}(s_{t+1}|s_t, a_t) \right] \hat{\pi}(a_t|s_t) \tag{2.3}$$

By exact inference, the framework looks at minimizing the *Kullback-Liebler* (KL) divergence between the two distributions of interest. The KL divergence is defined as:

$$D_{KL}(p(\tau)||\hat{p}(\tau)) = \mathbb{E}_{\tau \sim p(.)} log \left(\frac{p(\tau)}{\hat{p}(\tau)} \right) \tag{2.4}$$

This measure stipulates how much p and $\hat{p}$ differ where we consider p to be true. It is not symmetric; hence, it is not a distance metric. Depending on the order, two interpretations can be shown:

- the forward KL $D_{KL}(p(\tau)||\hat{p}(\tau))$ where we draw samples from p and maximize their probabilities under $\hat{p}$, i.e. high probabilities in p impose high probabilities in $\hat{p}$. This is known as *mean-seeking* behavior since $\hat{p}$ must cover all modes of high probabilities in p;
- the reverse KL $D_{KL}(\hat{p}(\tau)||p(\tau))$ where we draw samples from $\hat{p}$ and maximize their probabilities under p, i.e. high probabilities in $\hat{p}$ impose high probabilities in p. This is considered as *mode-seeking* behavior since each sample in $\hat{p}$ must be within a specific mode of p. The approximation $\hat{p}$ then looks for a mode with high probability and wide support.

The difference is subtle, as the first allows the approximation to cover all modes of the true distribution, not collapsing in any mode in particular. The second allows the approximation to cover one mode of high probability of the true distribution, preventing $\hat{p}$ from collapsing in one which would have a strong narrow support, i.e. an overspecialized one. Going back to equations (2.2) and (2.3), the RL objective is expressed as maximizing: $-D_{KL}(p(\tau|\mathcal{O}_T)||\hat{p}(\tau|\mathcal{O}_T)) = 0$. The forward KL becomes difficult to compute since we do not have access to the true distribution $p(\tau|\mathcal{O}_T)$. However, we can sample from $\hat{p}$ because of interactions with the environment and realize that the previous expression is similar to a reverse KL: $-D_{KL}(p(\tau|\mathcal{O}_T)|\hat{p}(\tau|\mathcal{O}_T)) = D_{KL}(\hat{p}(\tau|\mathcal{O}_T)|p(\tau|\mathcal{O}_T))$. Now, we see that $\hat{p}$ looks for a mode of high probability and wide support of p. Since the latter is conditioned on the optimal trajectory, optimization will look for a mode close to optimality without collapsing in one which would harm generalization. To simplify notations, we will hide conditions on $\mathcal{O}_T$. Negating and developing the KL objective using linearity of the expectation, we obtain:

$$\begin{aligned}
-D_{KL}(\hat{p}(\tau)||p(\tau)) &= \mathbb{E}_{\tau\sim\hat{p}(.)}[log\, p(s_1) + \sum_{t=1}^{T-1} log\, p(s_{t+1}|s_t, a_t) + \sum_{t=1}^{T} r(s_t, a_t) \\
&\quad - log\, \hat{p}(s_1) - \sum_{t=1}^{T-1} log\, \hat{p}(s_{t+1}|s_t, a_t) - log\, \hat{\pi}(a_t|st)] \qquad (2.5) \\
&= \mathbb{E}_{\tau\sim\hat{p}(.)}[log\, p(s_1) + \sum_{t=1}^{T-1} log\, p(s_{t+1}|s_t, a_t) + \sum_{t=1}^{T} r(s_t, a_t)] \\
&\quad - \qquad \mathcal{H}(\hat{\pi}(\tau))
\end{aligned}$$

with the entropy $\mathcal{H}(p) = -\mathbb{E}[log\, p]$. This objective is difficult to optimize because of the lop-probabilities $logp(s_{t+1}|s_t, a_t)$ and $log\hat{p}(s_{t+1}|s_t, a_t)$. Again, this makes the agent too optimistic since it has control over the transition dynamics. It would allow the algorithm to remove terrible outcomes of risky actions from the model while we want to take them into account. Moreover, accessing the transition dynamics is unrealistic in many settings,

as the resulting policies would be sub-optimal in practice. One could force $\hat{p}(s_1) = p(s_1)$ as well as $\hat{p}(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t)$, letting only $\hat{\pi}(a_t|s_t)$ vary. This would lead to a modification of $\hat{p}(\tau)$:

$$\hat{p}(\tau) = \left[p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t) \right] \hat{\pi}(a_t|s_t)$$

From it, we derive a new expression from the KL divergence with several elements canceling each other:

$$\begin{aligned} -D_{KL}(\hat{p}(\tau)||p(\tau)) &= \mathbb{E}_{\tau \sim p(.)}[\sum_{t=1}^{T} r(s_t, a_t) - log\, \hat{\pi}(a_t|st)] \\ &= \mathbb{E}_{\tau \sim p(.)}[\sum_{t=1}^{T} r(s_t, a_t)] - \mathcal{H}(\hat{\pi}(\tau)) \end{aligned}$$

We now seek to optimize the expected reward and entropy, which differs from traditional RL because of the second. We refer to this framework as *Maximum Entropy* RL, from which we can also derive a formulation for Q and V by recovering a proper Bellman operator [70]. It essentially formalizes that policies generating similar rewards should be equally probable. By that, it enhances exploration and generalization in the presence of disturbances [47]. Thus, it is entirely in line with the fact that learning a proper controller for magnetic control might benefit from enhanced exploration mechanisms. This objective will be extended in the next section using variational inference, starting the description of the algorithm used in this PhD.

2.4.2 Maximum a posteriori Policy Optimization

The *Maximum a posteriori Policy Optimization* algorithm (MPO) is an off-policy actor-critic, which sees the search of a policy from the perspective of *Expectation-Minimization* (EM) algorithms [5]. Generally speaking, EM methods are applied where there is a latent random variable from which data is not supposed to be visible in the first place [16]. In our context, this means obtaining the mode of a posterior distribution q over the optimal trajectory yet to be optimized. To understand MPO, we must start with the variational inference approach and use it to extend the initial inference-based formulation. Let us recall the trajectory distribution:

$$p(\tau) = p(s_1) \prod_{t=1}^{\infty} p(s_{t+1}|s_t, a_t) \pi_p(a_t|s_t) \tag{2.6}$$

and $\mathcal{O}_T$ the optimal trajectory, under the MDP $\mathcal{M}(\mathcal{S}, \mathcal{A}, p, r)$. We denote $\pi_p(a_t|s_t) = p(a_t|s_t)$ for consistency regarding previous policy notations, and use a discount factor γ to turn the problem into a finite horizon one, instead of considering the finite sum which has been seen until now. We then have:

$$p(\tau|\mathcal{O}_T) \propto p(\tau, \mathcal{O}_T) = [p(s_1) \prod_{t=1}^{\infty} p(s_{t+1}|s_t, a_t)] \, exp(\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t))$$

Similarly, we define q as the approximation of the posterior distribution we seek through optimization. For the incoming derivation, let us place ourselves in the continuous case, even if the discrete one would follow the same principle. Performing the optimization using variational inference principles relates to looking for an *evidence lower bound*, starting its derivation from:

$$\begin{aligned} log\, p(\mathcal{O}_T) &= log \int p(\mathcal{O}_T, \tau) d\tau \\ &= log \int \frac{q(\tau)}{q(\tau)} p(\mathcal{O}_T, \tau) d\tau \\ &= log\, \mathbb{E}_{\tau \sim q(.)} \left[\frac{p(\mathcal{O}_T, \tau)}{q(\tau)} \right] \end{aligned}$$

Jensen's inequality stipulates that:

$$log\, \mathbb{E}_{\tau \sim q(.)} \left[\frac{p(\mathcal{O}_T, \tau)}{q(\tau)} \right] \geq \mathbb{E}_{\tau \sim q(.)} \left[log\, \frac{p(\mathcal{O}_T, \tau)}{q(\tau)} \right]$$

Using log decomposition and the Bayes' rule [129], we obtain:

$$\begin{aligned} \mathbb{E}_{\tau \sim q(.)} \left[log\, \frac{p(\mathcal{O}_T, \tau)}{q(\tau)} \right] &= \mathbb{E}_{\tau \sim q(.)} [log\, p(\mathcal{O}_T, \tau) - log\, q(\tau)] \\ &= \mathbb{E}_{\tau \sim q(.)} [log\, (p(\mathcal{O}_T|\tau) p(\tau)) - log\, q(\tau)] \\ &= \mathbb{E}_{\tau \sim q(.)} [\sum_{t=1}^{\infty} \frac{1}{\alpha} \gamma^t r(s_t, a_t) - (log\, q(\tau) - log\, p(\tau))] \end{aligned}$$

The $\frac{1}{\alpha}$ term is usually induced as a temperature parameter in the expression of $p(\mathcal{O}_t = 1|s_t, a_t)$. Finally, using linearity of the expected value, the variational lower bound is expressed as:

$$log\, p(\mathcal{O}_T) \geq \mathbb{E}_{\tau \sim q(.)} \left[\sum_{t=1}^{\infty} \frac{1}{\alpha} \gamma^t r(s_t, a_t) \right] - D_{KL}(q(\tau) \,||\, p(\tau)) = \mathcal{J}(q, p)$$

This is a KL term similar to what was displayed in Equation (2.5), this time incorporating the discount factor. Many choices are valid for q, and we choose to define it with $q(s_1) = p(s_1)$ and $q(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t)$, $\forall t > 0$. Hence, $q(\tau)$ is expressed in a similar way as Equation (2.6) for $p(\tau)$, letting only $q(a_t|s_t)$ vary. We will write $\pi_q(a_t|s_t) = q(a_t|s_t)$, for consistency with π_p. Developing the Kullback-Leibler term, we end up with:

$$log\, p(\mathcal{O}_T) \geq \mathbb{E}_{\tau \sim q(.)} \left[\sum_{t=1}^{\infty} \frac{1}{\alpha} \gamma^t r(s_t, a_t) \right] - D_{KL}(\pi_q \,||\, \pi_p) = \mathcal{J}(\pi_q, \pi_p)$$

We want to find the policy which maximizes $\mathcal{J}(q, p)$. Indeed, we want to identify posterior distribution q close to prior distribution p, controlling the rate of change in policy updates at each learning iteration. Again, this reverse KL scheme makes π_q look for a mode of high probability and wide support of π_p, which is conditioned on the optimal trajectory. Once this lower bound is formally defined, we follow the same operations seen in actor-critic algorithms. First, we estimate new representations for the Q-values and improve the policy from them. This actor-critic setup allows to work in a distributed manner, as described in [5, 79], which increases the sample efficiency by multiplying many actors in favor of one critic, filling a replay buffer with many interactions with the environment. The MPO procedure refers to policy evaluation and improvement in the same way it has been described previously:

1. policy evaluation: We learn an estimation of the critic Q^{π_p} using a standard on-policy algorithm. Q^{π_p} will be parameterized by the neural network with learnable parameters λ. It is worth noticing that initial works [5] used the *Retrace* algorithm [81]. However, we did not see any enhancements compared to the regular n-step bootstrap targets, so the latter is kept in the remaining part of this manuscript;
2. policy improvement: We use the new Q^{π_p} to perform the EM scheme:
   (a) the E-step optimizes $\mathcal{J}(\pi_q, \pi_p)$ with respect to π_q while fixing $\pi_p = \pi_\theta$, a parameterized policy in the form of a fully-connected neural network. Since we do not have knowledge of which actions are optimal, we infer π_q from which actions would lead to improvement of the policy;
   (b) the M-step optimizes $\mathcal{J}(\pi_q, \pi_p)$ with respect to π_p while fixing the obtained π_q. Since π_q gives a lower bound, it maximizes $\mathcal{J}$, improving the policy.

As seen previously in this chapter, target networks are known to make learning in off-policy algorithms run smoother. We will use this principle for the actor and the critic and denote them:

- $Q_\lambda^{\pi_\theta}$ the online parameterized critic fitted at each policy evaluation;
- $Q_{\lambda'}^{\pi_\theta}$ the target critic that is updated periodically with weights from $Q_\lambda^{\pi_\theta}$;
- π_θ the online parameterized policy, i.e. the one that is improved at each EM iteration;
- $\pi_{\theta'}$ the target parameterized policy that is updated periodically with weights from π_θ.

At each iteration k of MPO, $Q_{(.)}^{\pi_\theta^{(k)}}$ will refer to the current online or target critic, and $Q_{(.)}^{\pi_\theta^{(k-1)}}$ to the previous ones. The same idea applies to $\pi_{(.)}$. Since weights of target networks

are kept fixed for N iterations, we almost always have $(.)^{(k)}_{(.)'} = (.)^{(k-1)}_{(.)'} = \cdots = (.)^{(k-N)}_{(.)'}$. Experiences used to perform learning are tuples (s_t, a_t, s_{t+1}, r_t) with $t \in \{1, ..., T\}$ for $T > 0$, sampled from a replay buffer. Later in this manuscript, we will see that MPO is trained with sequences instead of one-step interactions, for use by a recurrent critic [29, 135]. Considering maximum length L of a random episode, this means that s_t will be defined on $\{0, \cdots, L-1\}$, and s_{t+1} will span $\{1, \cdots, L\}$. N-step bootstrap returns are then computed for each state in the sequence, with practical implementations padding the end of the latter with n repetitions of the last observed Q-values.

Policy Evaluation

A simple TD-learning approach is considered to fit $Q_\lambda^{\pi_\theta^{(k)}}$ in a supervised way, minimizing the following TD error by gradient descent:

$$\min_\lambda \left(\underbrace{Q_{\lambda'}^{\pi_\theta^{(k-1)}}(s_{t+1}, a_{t+1})}_{\text{targets}} - \underbrace{Q_\lambda^{\pi_\theta^{(k)}}(s_t, a_t)}_{\text{predictions}} \right)^2$$

We have $a_{t+1} \sim \pi_{\theta'}^{(k)}(s_{t+1})$, and targets computed using n-step bootstrap returns. This means (for sequences but not only) that $Q_{\lambda'}^{\pi_\theta^{(k-1)}}(s_{t+1}, a_{t+1})$ is evaluated for each state, and the related target is built using the next n states as expected by the method. Now that $Q_\lambda^{\pi_\theta^{(k)}}$ is updated, the EM procedure can start.

Policy Improvement

Let us recall that the lower bound of interest is defined as:

$$\mathcal{J}(\pi_q, \pi_p) = \mathbb{E}_{\tau \sim q(.)} \left[\sum_{t=1}^{\infty} \frac{1}{\alpha} \gamma^t r(s_t, a_t) \right] - D_{KL}(\pi_q \,||\, \pi_p)$$

We can fold $1/\alpha$ in the reward function, i.e. as an implicit hyperparameter, or multiply the expression by α to get a similar expression in which the KL divergence is weighted [47]. This might be seen as an interpolation between the standard RL objective and the KL one.

E-step To start with, we fix $\pi_p = \pi_{\theta'}^{(k)}$, and rely on the target critic $Q_{\lambda'}^{\pi_\theta^{(k)}}$ since it is preferable regarding training stability. Without them, all derivations would just assume $\pi_p = \pi_\theta^{(k)}$ and $Q_\lambda^{\pi_\theta^{(k)}}$. So, we get:

$$\mathcal{J}(\pi_q, \pi_{\theta'}^{(k)}) = \mathbb{E}_{\tau \sim q(.)} \left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \right] - \alpha D_{KL}(\pi_q \,||\, \pi_{\theta'}^{(k)})$$

We can realize that J is associated with a general Q-function Q^d like any MDP, for any distribution d, state and action trajectories. Traditionally, $Q^d = \mathbb{E}_{\tau\sim d}\left[\sum_{t=1}^{\infty}\gamma^t r(s_t,a_t)\right]$. In our situation, we have:

$$Q^d = \mathbb{E}_{\tau\sim d(.)}\left[\sum_{t=1}^{\infty}\gamma^t r(s_t,a_t) - \alpha D_{KL}(\pi_q \,||\, \pi^{(t)})\right]$$

This shows that optimizing $\mathcal{J}$ with respect to π_q can be expressed as solving the MDP with a regularized Q^d, i.e. π_q is regularized towards the best current policy $\pi^{(t)} = \pi_{\theta'}^{(k)}$. Here, we have $Q^{\pi_{\theta'}^{(k)}} = \mathbb{E}_{\tau\sim q(.)}\left[\sum_{t=1}^{\infty}\gamma^t r(s_t,a_t)\right]$. This formulation is critical since it introduces Q-values. Ultimately, the approximation q must have the following property:

$$\mathbb{E}_{\tau\sim\pi_{\theta'}^{(k)}}\left[Q^{\pi_{\theta'}^{(k)}}\right] \leq \mathbb{E}_{\tau\sim\pi_q^{(k)}}\left[Q^{\pi_q^{(k)}}\right]$$

We can modify $\mathcal{J}$ by marginalizing out actions and states, and by playing with the previous inequality, we get the extended cost [5]:

$$\mathcal{J}(\pi_q,\pi_{\theta'}^{(k)}) = \mathbb{E}_{s\sim q(.)}\left[\mathbb{E}_{a\sim\pi_q(.|s)}\left[Q_{\lambda'}^{\pi_\theta^{(k)}}(s,a)\right] - \alpha D_{KL}(\pi_q \,||\, \pi_{\theta'}^{(k)})\right]$$

A more complete derivation can be found in [5, 4]. It is challenging to compute the expectation $\mathbb{E}_q$ with respect to π_q [38], so s is sampled from the replay buffer which exhibits a stationary distribution μ. A final issue arises with the α parameter, which is complex to fine-tune because of the relative scale of the KL term and Q. To overcome it, this regularization problem is turned into a hard constraint on the KL divergence:

$$\begin{aligned}\max_q \quad & \mathbb{E}_{s\sim\mu(.)}\left[\mathbb{E}_{a\sim\pi_q(.|s)}\left[Q_{\lambda'}^{\pi_\theta^{(k)}}(s,a)\right]\right] \\ s.t. \quad & \mathbb{E}_{s\sim\mu(.)}\left[D_{KL}(\pi_q \,||\, \pi_{\theta'}^{(k)})\right] < \epsilon\end{aligned} \tag{2.7}$$

One can interpret the ϵ hyperparameter as a tool to control the exploration rate during learning, allowing deviation from the target policy. To infer π_q, we consider that it is given by a non-parametric formulation, built with actions sampled from $\pi_{\theta'}^{(k)}$ for each state in the current batch. Parametric formulations of π_q would remove the need for an M-step, leading to algorithms such as *Soft actor-critic*, *Trust Region Policy Optimization* or *Proximal Policy Optimization* [48, 103, 105]. Once we have a proper formulation for this E-step, the algorithm is executed in the following way while taking time back into account:

1. We sample m actions from $\pi_{\theta'}^{(k)}$, for each state $\{s_{j,t+1}\}_{1\leq j\leq n}$ in the current batch, i.e. $a_{i,t+1}\sim\pi_{\theta'}^{(k)}(s_{j,t+1})$ with $i\in\{1,\cdots,m\}, j\in\{1,\cdots,n\}, t\geq 0$. For the sake of simplicity, let us note $a_{i,t+1}$ simply by a, and similarly for s, except when precisions will be required. By doing this, we build state-action pairs to create targets from the fixed policy.

2. Shaping $\pi_q(a|s)$ comes to adjusting the probabilities of each action for each state, so that actions with higher action-value have higher probabilities [4]. This goes back to the reverse KL definition: we want to shape π_q close to a mode of high probability and wide support of the Q-values for the best current policy $\pi_{\theta'}^{(k)}$. Any valid transformation should guarantee $\forall(i,j) \in \{1,\cdots,m\} \times \{1,\cdots,n\}$, $\pi_q(a_i|s_j) > 0$ and $\sum_i \pi_q(a_i|s_j) = 1$. We enhance the constraints as a consequence:

$$\begin{aligned} \max_q \quad & \mathbb{E}_{s\sim\mu(.)}\left[\mathbb{E}_{a\sim\pi_q(.|s)}\left[Q_{\lambda'}^{\pi_\theta^{(k)}}(s,a)\right]\right] \\ s.t. \quad & \mathbb{E}_{s\sim\mu(.)}\left[D_{KL}(\pi_q \,||\, \pi_{\theta'}^{(k)})\right] < \epsilon \\ & \forall s \sum_{i=0}^{m} \pi_{\theta'}^{(k)}(a_i|s) = 1 \end{aligned}$$

   By doing that, we ensure that normalized $\pi_q(a|s)$ stay close enough to the last policy distribution. Expected Q-values increase while preventing $\pi_q(a|s)$ from collapsing into one narrow mode at the start.

3. A derivation of this problem is obtainable through a Lagrangian formulation [5], giving:

$$\pi_q(a|s) \propto \pi_{\theta'}^{(k)}(a|s) exp\left(\frac{Q_{\lambda'}^{\pi_\theta^{(k)}}(s,a)}{\eta}\right)$$

   where η is a temperature parameter representing the hard constraint on ϵ. To understand the naming of this parameter in machine learning, we can start from the *Boltzmann distribution* that historically emanates from statistical mechanics. Let us define a system's state as a function of its temperature and energy, with $p(s_n)$ the probability of being in state s_n:

$$p(s_n) = \frac{exp(-\frac{E_n}{kT})}{\sum_l^{|\Omega|} exp(-\frac{E_l}{kT})}$$

   with a finite set of reachable state Ω, k set to the *Boltzmann* constant and T the temperature of the system. We can see that states with lower energy will have a high probability of occurrence. More than that, if T is increased, the overall distribution flattens out, increasing entropy. However, if T is reduced, the distribution sharpens with a dominating mode. Said differently, if T is increased, high probabilities become lower, and low ones become higher. In the opposite case, high probabilities become even higher, and low ones become even lower. While being used intensively in Machine Learning, the function $Softmax : \mathbb{R}^m \longrightarrow [0,1]^m$, $m > 0$ can be linked to the Boltzmann distribution. The Softmax is mainly used to output a vector of probabilities whose dimension relates to the classification problem at hand. For example, we could try to output the probability that one picture will be a dog, a cat,

or a panda. Placed at the end of a neural network, it leads to a vector with three probabilities summing up to 1. Each of the latter's components comes from the expression:

$$Softmax(\mathbf{s})_n = \frac{exp(-\beta s_n)}{\sum_l^m exp(-\beta s_l)}$$

for component n of vector $\mathbf{s}$. We connect the two mathematical tools, setting $\beta = \frac{1}{T}$ and $s_n = E_n$. This might be one reason why machine learning refers to this naming convention. In our case, the Q-values replace the energy. However, the interpretation still holds: a higher η would flatten the distribution, making the probabilities of states with lower Q-values higher and those with higher Q-values lower. A smaller η would follow the opposite behavior we just discussed. Exploration is intrinsically related to this statement since the ϵ parameter from Equation (2.7) can be interpreted as such. In addition, η impacts the distribution of Q-values, which directly connects to the reverse KL mode-seeking formulation. Finding this temperature is the ultimate operation required by this E-step and corresponds to a convex dual optimization of the form:

$$\eta = \arg\min_{\eta} \ \eta\epsilon + \eta \sum_{j=1}^{n} \frac{1}{n} \, log \, \sum_{i=1}^{m} \frac{1}{m} exp\left(\frac{Q_{\lambda'}^{\pi_\theta^{(k)}}(s_j, a_i)}{\eta}\right)$$

In practice, only one gradient descent step is needed to update π_q, using the *Adam* optimizer [64] as it ensures η positive. The initialization of the temperature is important because it will influence how exploration happens at the beginning of learning. Because of the stationary distribution $\mu(s)$ given by samples from the replay buffer and the target Q network used to evaluate the system over actions a, we face a complete off-policy learning procedure. It has reduced variance of the estimate [5] at the cost of performing only a partial optimization of $\mathcal{J}$, since $\pi_p = \pi_{\theta'}$ and $Q^{\pi_{\theta'}^{(k)}}$ as a consequence, are seen as constant with respect to q. This shows that policy improvement is performing an alternate coordinate ascent in π_q and π_p.

M-step Now that a new estimate $\pi_q(a|s)$ is obtained based on sampled actions, we must extend it to the state space so that better actions can be taken in *unknown* states. That is when the M-step steps in by optimizing $\mathcal{J}(\pi_q, \pi_\theta)$ with respect to π_θ to improve the policy, i.e. obtain $\pi_\theta^{(k+1)}$. In $\mathcal{J}$, we drop terms independent of $\pi_\theta^{(k)}$ and we fix π_q to the previously inferred distribution. It leads to the following problem:

$$\max_{\theta} \mathcal{J}(\pi_q, \pi_p) = \max \mathbb{E}_{q\sim\mu(.)} \left[\mathbb{E}_{a\sim\pi_q(.|s)} log \, \pi_\theta^{(k)}(a|s)\right] \tag{2.8}$$

Unfortunately, this maximum likelihood can overfit easily to the the sampled based distribution obtained in the E-step. Furthermore, if the policy evaluation gives poor

signals, the sampled-based distribution can be inaccurate. These elements might result in strong oscillations of the action distribution. A solution comes from adding a new KL constraint to Equation (2.8) to limit the change of the parametric policy, which is different than what was done during the E-step, which limited the rate of change in the approximate action distribution. Consequently, it would better generalize without overfitting the samples, going beyond the state-actions pairs used previously. This gives the following constrained M-step:

$$\begin{aligned} \max_{\pi_\theta} \quad & \mathbb{E}_{q\sim\mu(.)}\Big[\mathbb{E}_{a\sim\pi_q(.|s)} log\, \pi_\theta^{(k)}(a|s)\Big] \\ s.t. \quad & \mathbb{E}_{s\sim\mu(.)}\Big[D_{KL}(\pi_{\theta'}^{(k)} \,||\, \pi_\theta^{(k)})\Big] < \epsilon_{\pi_\theta} \end{aligned}$$

where ϵ_{π_θ} is a hyperparameter of the expected change over state distribution in KL divergence for the policy. If we consider $\pi_{\theta'}^{(k)}$ to be the true stable action distribution, and $\pi_\theta^{(k)}$ the online one yet to be approximated, this can be seen as a forward KL formulation. More than the rate of change, it ensures that updates in the policy do not collapse in a specific mode, covering all high probabilities modes from the old representation. Using a Lagrangian formulation again [5, 4], we obtain a primal optimization problem which can be solved by gradient ascent and descent:

$$\max_{\pi_\theta}\min_{\alpha} \sum_{j=1}^{n}\sum_{i=1}^{m} \pi_q(a_i|s_j)\, log\, \pi_\theta^{(k)}(a_i|s_j) + \alpha\left(\epsilon_{\pi_\theta} - \sum_{j=1}^{m}\frac{1}{m} D_{KL}(\pi_{\theta'}^{(k)} \,||\, \pi_\theta^{(k)})\right)$$

with $\alpha > 0$. It is solved in practice by alternate optimization of the inner and outer objectives independently, with one gradient optimization step performed for each routine. A new $\pi_\theta^{(k+1)}$ is obtained, and we repeat policy evaluation and improvement until convergence.

Convergence issues

The overall policy improvement procedure has a significant flaw. The expected regularized reward is updated through actions sampled from the last policy distribution. In this case, the optimal solution gives equal probabilities to equally good actions based on $Q^{\pi_{\theta'}^{(k)}}$ and 0 probabilities to others. This induces a mode collapse on the best actions even if they are not truly optimal because of a wrong approximation of Q. A first intuitive solution comes from adding a KL constraint, as done during the M-step, regarding the rate of change in the parametric policy. However, we would still lose entropy to cover the best actions so far, i.e. the algorithm would prematurely converge towards one sub-optimal mode. This also depends on the hyperparameters for ϵ_{π_θ}, and the underlying topological structure of Q [4]. The solution has been studied in other works such as [3], and presents a decoupling of the objective into the policy mean and its covariance matrix:

1. we first maximize one objective to update the mean of the policy distribution, with the covariance matrix fixed to the one from the target policy, i.e. the last best policy;

2. we then maximize the other objective to update the covariance matrix while similarly fixing the mean.

This is particularly useful with action spaces in high dimensions, which is the case of our application. Indeed, it avoids ill-definitions of the covariance matrix but enables fast learning by allowing significant changes in the mean. More than that, gradients from the covariance are independent of variations of the mean. Because of that, if one wants to increase the likelihood of good samples far away from the mean, the policy must stretch along the value manifold, which is allowed by the decoupling without introducing any additional entropy term to control the policy distribution [114]. Again, through Lagrangian relaxation, the M-step objective becomes:

$$
\begin{aligned}
\max_{\mu_\theta, \Sigma_\theta} \quad & \mathbb{E}_{q\sim\mu(.)}\Big[\mathbb{E}_{a\sim\pi_q(.|s)}\Big[log\ \pi_\theta(a|s, \mu = \mu^k)\Big]\Big] \\
& + \mathbb{E}_{q\sim\mu(.)}\Big[\mathbb{E}_{a\sim\pi_q(.|s)}\Big[log\ \pi_\theta(a|s, \Sigma = \Sigma^k)\Big]\Big] \\
s.t. \quad & \mathbb{E}_{s\sim\mu(.)}\Big[D_{KL}(\pi^{(k)}_{\theta'}(a|s)\ ||\ \pi^{(k)}_{\theta}(a|s, \mu = \mu^k))\Big] < \epsilon_\mu \\
s.t. \quad & \mathbb{E}_{s\sim\mu(.)}\Big[D_{KL}(\pi^{(k)}_{\theta'}(a|s)\ ||\ \pi^{(k)}_{\theta}(a|s, \Sigma = \Sigma^k))\Big] < \epsilon_\Sigma
\end{aligned}
$$

with $\pi_\theta = \mathcal{N}(\mu_\theta, \Sigma_\theta)$. ϵ hyperparameters can again be interpreted as a decoupled tolerance over exploration. Finally, an action penalty is added to the learning scheme similarly to [2]. The decoupling shows that we use Gaussian policies, and enhance exploration through proper initial mean and standard deviations. Nevertheless, such distributions have infinite support. A penalty for the actions trespassing their bounds is added to the policy loss in the M-step, forcing the mean of the policy to stay within the range of the action space. In our context, it helps avoid modes of the action distribution in which the agent's actions saturate to physical limits of the device, stopping premature convergence to a non-optimal policy. A final summary of the MPO algorithm is given in (Figure 2.6).

2.5 An agent ready for interactions

This chapter allowed us to define the proper terminology of RL and display its close links with optimal control. Many methods exist, and we decided to focus on off-policy actor-critic methods, defining their core elements as neural networks. Deep reinforcement learning has then been described to better contextualize the advantages and drawbacks of the approach. Probabilistic inference is an interesting interpretation of RL, which is at the basis of the Maximum a posteriori Policy Optimization algorithm used in this PhD. The latter was finally presented, notably its implementation and how it inserts itself in

the family of inference-based methods. It combines the estimation of the Q-values, policy evaluation, and policy improvement as an EM-based procedure. Hard constraints on the objectives and Gaussian policies explicitly handle exploration. One must notice that the MPO formulation slightly differs from the introduction we discussed regarding inference-based approaches. Indeed, it does not rely on a complete probabilistic model but only on a lower bound defined for classical policy search, which results in risk-seeking behavior [69]. Furthermore, it does not try to solve a maximum entropy objective framing the entire problem but focuses on solving such formulation at each learning iteration. This is done through a bound on the KL divergence, which is not between the new policy and an exponentiated reward but between the new policy and the old one (in the sense of target networks). This gives more stable training, avoiding previous issues raised by the difference with classical maximum entropy RL. It exhibits several advantages of inference-based algorithms, and its off-policy setting shows strong empirical performance in a variety of domains [115]. Moreover, its sample complexity is lower than state-of-the-art on-policy methods [103, 105], and it is easily scalable to gather interaction data in a distributed manner. This is particularly interesting knowing that the environment used in this PhD is computationally expensive, as we will see in chapter 3. Its use of probabilistic inference allows a refined control solution in the context of Markov Decision Processes where the transition dynamics are uncertain, and the environment's state is not completely visible to the agent. Finally, considering the objective stated in the introduction, replicating the state-of-the-art might benefit from using the same algorithm despite the number of approaches in the deep RL landscape. Now that everything has been displayed for the agent, the next chapter will present the environment simulating everything needed for magnetic control on the WEST tokamak.

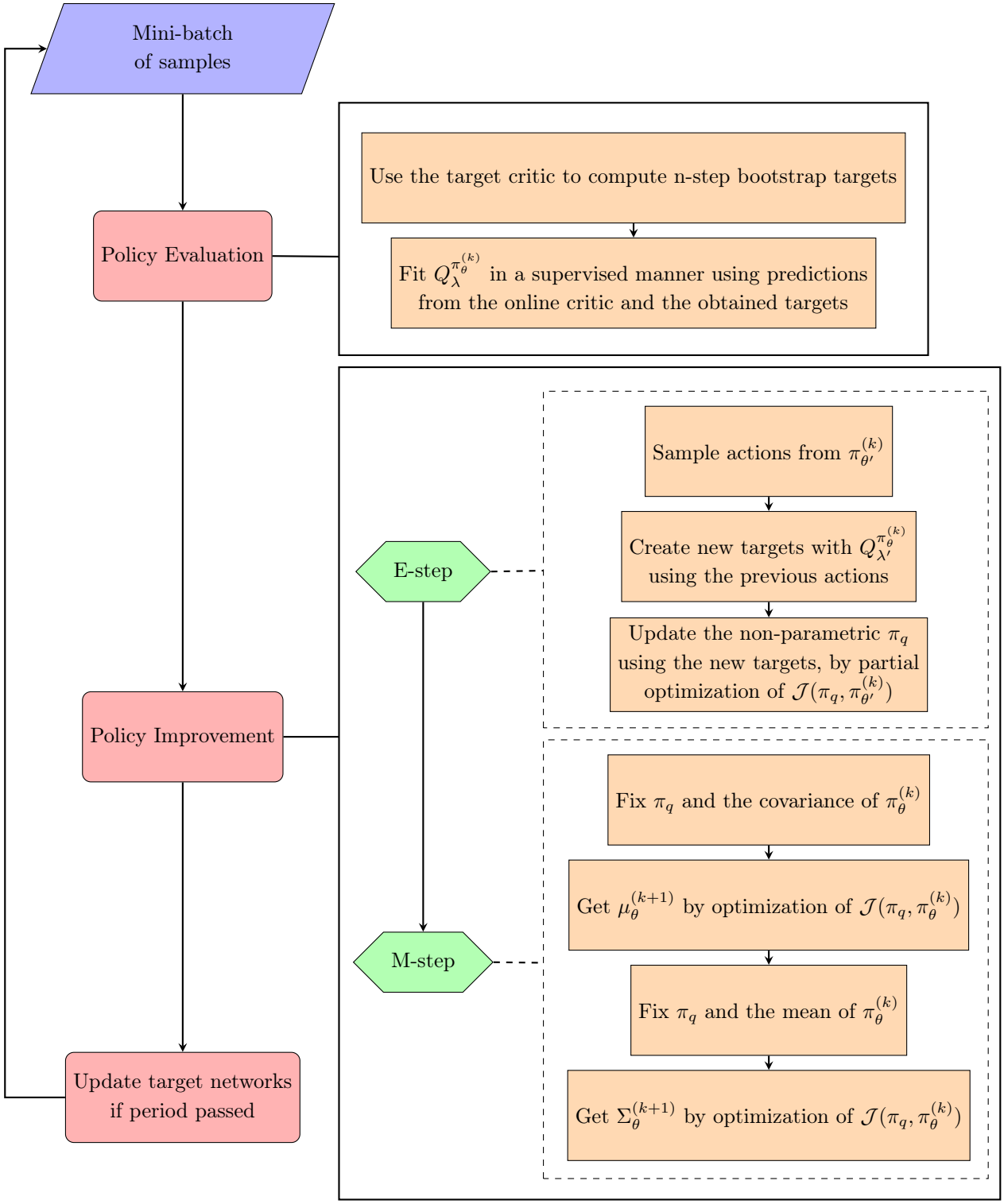


Figure 2.6: Summary of the Maximum a posteriori Policy Optimization algorithm

PILOT: a general framework for magnetic control

Contents

3.1 Creating a numerical twin for WEST 62
3.1.1 Machine description 62
3.1.2 Control scenarios 63
3.1.3 A NICE environment to train them all 67
3.2 In a world of scenarios and rewards 79
3.2.1 References generator 79
3.2.2 Reward definition 80
3.2.3 A digression on the WEST plasma control system 85
3.3 Assembling a distributed architecture 88
3.3.1 The wonderful story of how C++ met Python 88
3.3.2 Nodes galore 89
3.3.3 A glimpse of the agent's distinctive features 91
3.4 A framework ready for training 94

To produce RL-based magnetic controllers, a proper interaction loop is needed, composed of all the inherent building bricks of reinforcement learning. However, such dedicated software was not directly available for this PhD. During its entirety, the *PILOT*[1] framework was born to create a platform on which environments and agents could be easily modified across projects. In the previous section, we defined the chosen algorithm for this study, called the Maximum a posteriori Policy Optimization (MPO), which has several advantages in complex and slow environments. In fusion applications, available simulations have constrained specifications, notably regarding magnetic equilibrium evolution. No available numerical solver is provided for possible integration into an RL framework. Codes usually run entirely until they terminate since there is no reason to pause the execution, as expected from the RL interaction loop. Because of that, an extension of *NICE*[2] has been conducted to create a stable environment for our agent. This chapter thoroughly describes PILOT's components when applied to the WEST tokamak.

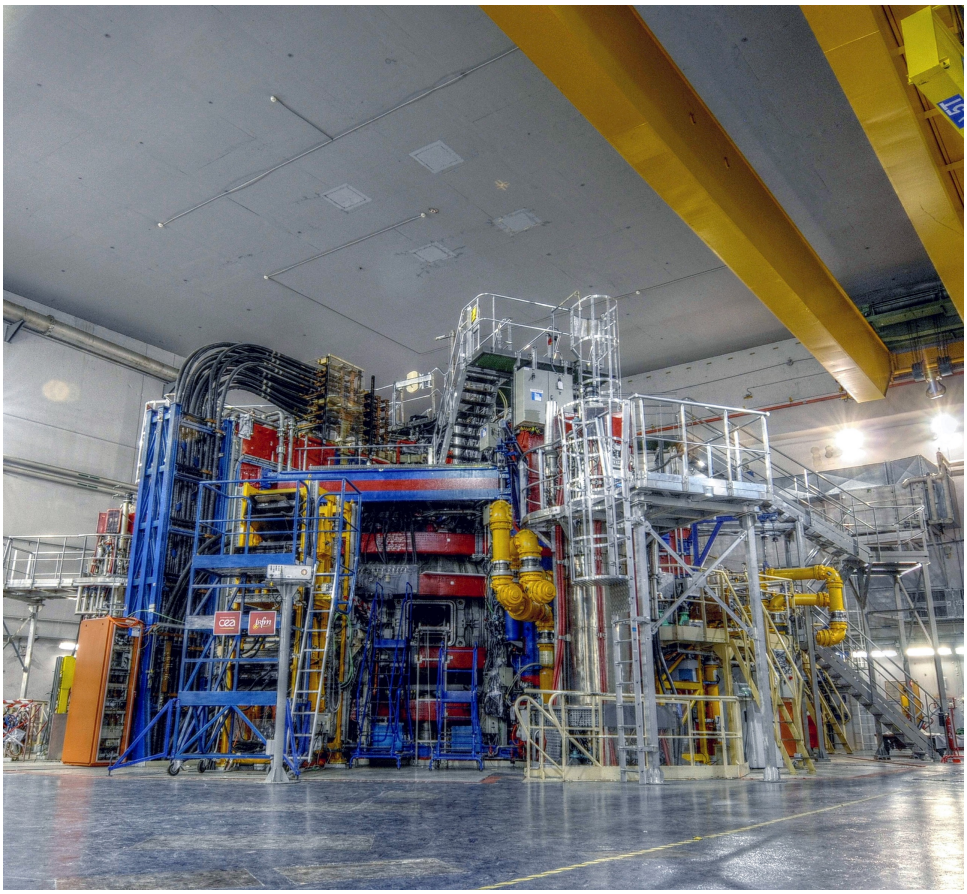


Figure 3.1: The WEST tokamak at CEA, Cadarache.

[1]Plasma rEinforcement Learning cOntrol for Tokamaks
[2]Newton direct and Inverse Computation for Equilibrium

3.1 Creating a numerical twin for WEST

3.1.1 Machine description

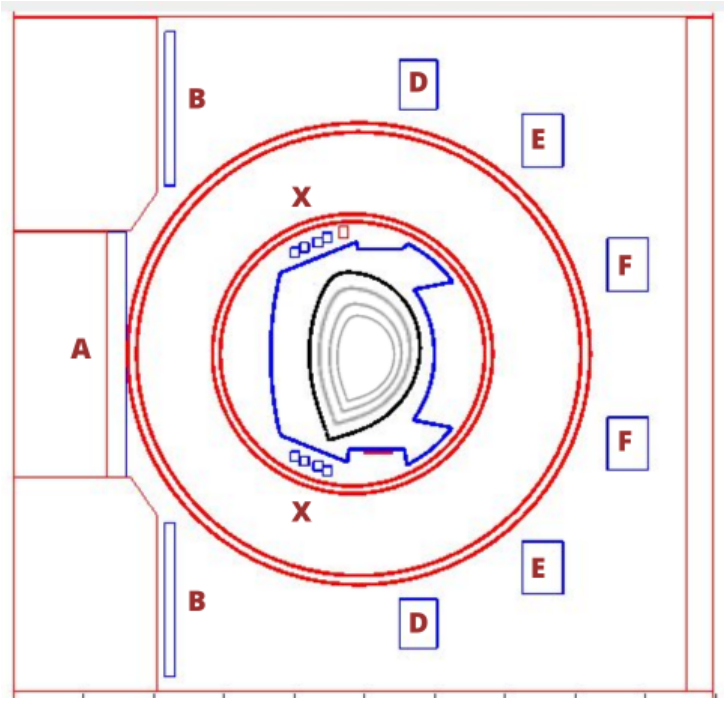


Figure 3.2: WEST cross-section with surrounding poloidal field coils.

WEST is a full tungsten environment superconducting tokamak located at CEA, Cadarache in France [17, 18] (Figure 3.1). It is an upgrade of *Tore Supra*, which was built in 1988 and achieved a first record plasma duration of 6 minutes and 30 seconds in 2003. In preparation for the ITER establishment, Tore Supra was transformed into WEST in 2016 to master the heat exhaust in a tungsten environment for long discharges. Consequently, the WEST experiment's main objectives gravitate around testing actively cooled plasma-facing components, especially its new divertor[3], as well as performing stable plasma discharges over 1000 seconds. WEST operates with a plasma current up to 1MA, exhibits a minor radius of 50cm, a major one of 2.5m, and a plasma volume of $20m^3$. Its heating systems incorporate Ion Cyclotron Resonance Heating and Lower Hybrid antennas, with respectively 9 and 7MW of power. Electron Cyclotron Resonance Heating was recently added and delivers 3MW of power. One of WEST main features lies in its superconductive toroidal coils which generate a magnetic field of 3.7T (at 2.5 meters radius) for long pulses. For control, nine poloidal coils surround the vacuum chamber, with two remaining inside as part of the divertors (Figure 3.2). WEST established a new record of

[3]Parts of the vacuum vessel in contact with the plasma in many configurations. Its utility will be discussed in the following sections

discharge duration in 2024, maintaining a plasma at $4keV$ for 6 minutes and 4 seconds, with an electron density twice greater than what was obtained on *Tore Supra*.

3.1.2 Control scenarios

Main magnetic configurations

In tokamak physics, the plasma can be seen as nested closed magnetic flux surfaces, i.e. iso-contours numerically speaking, where the *Last Closed Flux Surface* (LCFS) defines the plasma boundary. Magnetic control intervenes to target specific scenarios in which the plasma follows different configurations. In WEST, the latter is initiated in the *limiter* configuration, which sees the plasma leaning on an interaction point present on the *limiter*, a succession of manufactured tiles shaping a bumper that prevents damages on fragile components. Its circular-like boundary is then considered the last closed flux surface that is not in contact with the limiter. This can be performed using the nine control coils outside the vacuum vessel. Historically, elongated plasmas gained a lot of interest because advanced configurations with improved confinement properties originate from them. On WEST, they are achieved by adding two in-vessel coils to the overall system, namely the *divertor* coils, located in eponym structures (Figure 3.3). Each of these coils is composed of 4 sub-coils connected in series.

Each of their currents $\{I_{X_{up}}, I_{X_{down}}\}$ can be set in the same direction as the plasma current I_p so that the LCFS gets stretched and the elongation κ increases ($\kappa > 1$). At some point, one or more saddle points appear where the gradient of the magnetic flux is null, and the sign of its second-order derivative depends on the currents' direction. These null field areas form *X-points*, and the LCFS becomes surrounded by open surfaces. The chosen coil current values dictate the saddle point locations, which can appear on the upper

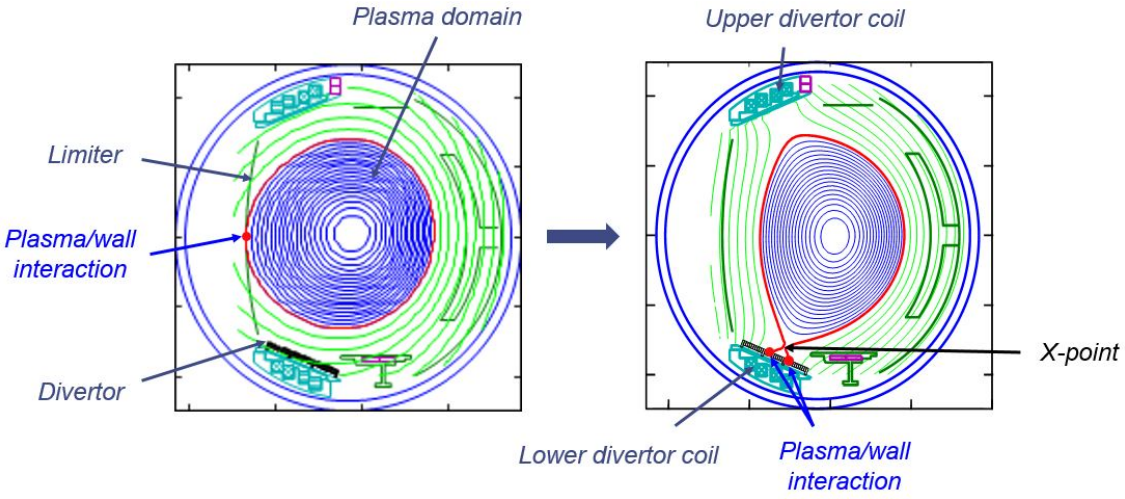


Figure 3.3: On WEST, many discharges describe a transition between limiter and X-point configurations.

half side of the vacuum chamber, on its lower one, or at both simultaneously. For each X-point, the plasma interacts with the wall materials at two *strike points* located on the related divertors. An important part of the plasma energy deposits on the latter, which is then subject to strong mechanical and physical constraints studied in many modern tokamaks. In this PhD, we only consider *lower single null* configurations, where a unique X-point is produced on the lower divertor coil side. This configuration is the basis of many experiments on WEST, in which classical control was used. Hence, it logically serves as a baseline to assess the efficiency of RL-based solutions before future work on more advanced configurations. It is worth noticing that the aforementioned enhanced confinement properties might come from the fact that X-points work as exclusive bounded zones, which limit contamination of the plasma by impurities caused by erosion of the chamber walls. This is particularly important when the latter are made out of tungsten, a heavy element with high radiating power: a small number of impurities can significantly degrade plasma confinement to the point of disruption. On WEST, most experiments tend to achieve higher plasma elongation, to benefit from these more efficient operation regimes. Therefore, scenarios start from a limiter configuration and evolve towards an X-point one. Then, the main objective is to ensure transitions as fast and precise as possible, without parasite oscillations in the plasma location, as well as undesired contacts between the confined plasma and many parts of the vacuum vessel. Nevertheless, relying on elongated plasmas comes at the risk of growing vertical instabilities that must be taken care of.

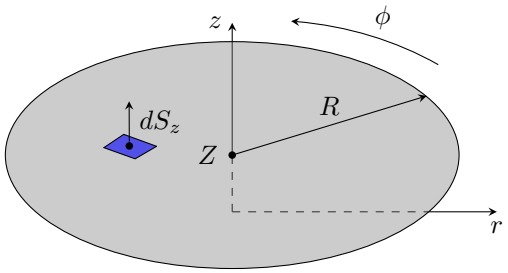


Figure 3.4: The (r,ϕ,z) coordinate system used to describe the origins of magnetic instabilities. The spatial integral over the disc of radius R and centered at Z is used to model the poloidal magnetic flux.

Plasma vertical instabilities

We now consider a filament model for a massless plasma in a tokamak, which was mentioned in the introduction, and a cylindrical right-handed coordinate system (r, ϕ, z) [101], with ϕ the toroidal angle. Recalling the foundations of tokamak physics, a strong current in the central solenoid generates a poloidal magnetic field, which combines with the toroidal one. From unit vectors $\hat{\mathbf{r}}, \hat{\mathbf{z}}, \hat{\boldsymbol{\phi}}$, the total magnetic field in a tokamak becomes:

$$\mathbf{B} = B_r(r,z)\hat{\mathbf{r}} + B_z(r,z)\hat{\mathbf{z}} + B_\phi(r,z)\hat{\boldsymbol{\phi}} = B_p(r,z) + B_\phi(r,z)$$

with B_p and B_ϕ, respectively the poloidal and toroidal components. The poloidal magnetic flux can then be described as [130]:

$$\psi(r,z) = \iint_S \mathbf{B}\, dS_z = \iint_S B_z(r',z) r' dr' d\phi = 2\pi \int_0^R B_z(r',z) r' dr'$$

with dS_z a surface element normal in the z direction and R the plasma major radius (Figure 3.4). Taking partial derivatives, we get:

$$\begin{aligned} \frac{\partial \psi}{\partial r} &= 2\pi r B_z \\ \frac{\partial \psi}{\partial z} &= 2\pi \int_0^R \frac{\partial B_z}{\partial r} r' dr' \end{aligned}$$

From the tokamak axisymmetry property on ϕ, we know that $\nabla \cdot \mathbf{B} = \frac{1}{r}\frac{\partial r B_r}{\partial r} + \frac{\partial B_z}{\partial z} = 0$. By rearranging the previous equations, we obtain:

$$\begin{cases} B_r = -\frac{1}{2\pi r}\frac{\partial \psi}{\partial z} \\ B_z = \frac{1}{2\pi r}\frac{\partial \psi}{\partial r} \end{cases}$$

Now that B_r and B_z are properly defined, they will be used to identify the mentioned instabilities. On the r axis, an outward force F_r is initially observed [130], and consequently, there is a need for a counteracting inward force to stabilize the plasma. This is done by generating a vertical magnetic field $\mathbf{B_v}$ using the control coils. This generation process is performed in the aim of giving $\mathbf{I_p} \times \mathbf{B_v}$ orientated in the $-\hat{\mathbf{r}}$ direction, with a resulting Lorentz force $\mathbf{F_L} \sim I_p \hat{\boldsymbol{\phi}} \times B_v \hat{\mathbf{z}}$. It is important to note that in the chosen coordinate system, the right-hand rule gives us that the sign of I_p is opposed to that of B_v. Knowing this, all plasma of interest are radially stable, and radial position control through the generation of a radial field $\mathbf{B_v}$ is only needed because of changes in the plasma characteristics (current, profiles, etc). Furthermore, we denote the curvature index n [67], which defines the curvature of the vertical field generated by the control coils:

$$n = -\frac{R}{B_{z_p}} \frac{\partial B_v}{\partial r}$$

with R the major radius, B_{z_p} the vertical field $\mathbf{B_v}$ generated by the coils and evaluated at z_p the plasma location on the z axis. That should equal 0 for a limiter plasma, and lower than 0 in the case of an elongated one [124]. Vertically, the plasma is at equilibrium where $B_r = 0$. Knowing the resulting Lorentz force $\mathbf{F_z} \sim \mathbf{I_p} \times \mathbf{B_r}$, and using the sign convention mentioned previously, we get that:

$$\frac{\partial F_z}{\partial z}\Big|_{z=z_p} < 0 \Longleftrightarrow \frac{\partial B_r}{\partial z}\Big|_{z=z_p} > 0$$

In the plasma, $\nabla \cdot \mathbf{B_p} = 0$, which leads to:

$$\frac{\partial B_v}{\partial r} = \frac{\partial B_h}{\partial z}$$

.

By rearranging the curvature index and recalling that if $I_p > 0 \rightarrow B_{z_p} < 0$, we get that $n > 0$. However, it goes against what was previously stated for an elongated plasma, with $n < 0$. This shows that any elongated plasmas will be vertically unstable. A stability analysis can also be performed from a control perspective, exhibiting an unstable mode under which the system undergoes vertical instabilities, and even if the vacuum vessel induces passive currents counteracting this issue, the instability timescale is too fast to entirely mitigate the phenomenon [67].

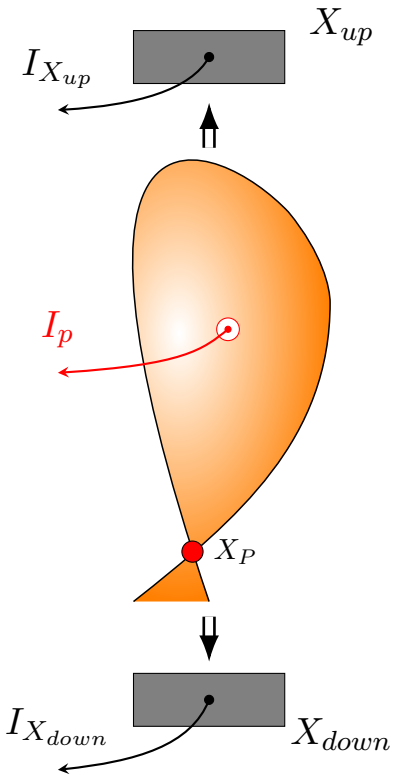


Figure 3.5: If we see the divertor coils and the plasma as conducting wires with currents flowing in the same direction, they attract each other. This attraction creates a vertical force responsible for vertical instabilities while the plasma stretches.

To conclude, the two divertor coils stretch and move the plasma. However, a small vertical displacement will increase the vertical force in this direction (Figure 3.5), accelerating the so-called vertical instabilities. Vertical control is an essential part of tokamak operations, and it will be discussed several times in this PhD, as we will look for magnetic control of the plasma position and shape. Thus, if we want to properly evaluate the approach of this PhD on a vast number of scenarios, these shapes must be handled

accordingly. Scenarios are then made of a succession of transitions between the two configurations of interest (Figure 3.3). The simulator described in the next section has been configured for stable initialization of both shapes and many possible intermediate ones.

3.1.3 A NICE environment to train them all

Magnetic equilibrium

Let us recall that the total magnetic field in a tokamak is decomposed in toroidal and poloidal components, respectively named $\mathbf{B}_\phi$ and $\mathbf{B}_p$ so that $\mathbf{B} = \mathbf{B}_\phi + \mathbf{B}_p$. Similarly, let us use the (r, ϕ, z) coordinate system introduced in the previous section. The *NICE*[4] [36, 45] solver is a C++ free-boundary equilibrium code solving a non-linear 2D partial differential equation of the poloidal flux ψ in time and space in tokamaks. The said equation represents the force balance in the plasma and starts from Maxwell equations, which are augmented with *Faraday* and *Ohm* laws to model the system's evolution:

$$-\Delta^*\psi(r,z,t) = j_\phi(r,\psi(r,z,t),t) \tag{3.1}$$

The left-hand side of 3.1 considers a second order elliptic operator Δ^*:

$$\Delta^*\psi = \frac{\partial}{\partial r}\left(\frac{1}{\mu(\psi)r}\frac{\partial\psi}{\partial r}\right) - \frac{\partial}{\partial z}\left(\frac{1}{\mu(\psi)r}\frac{\partial\psi}{\partial z}\right) = \nabla.\left(\frac{1}{\mu(\psi)r}\nabla\psi\right)$$

where $\mu(\psi)$ is the magnetic permeability, and ∇ is the 2D operator in the (r,z)-plane. The toroidal component of the current density j_ϕ depends on the location inside the vacuum chamber (Figure 3.6), and several domains must be taken into account as a consequence:

- the iron structures Ω_f, which make μ non-linearly dependent on ψ. Outside of them, it is equal to the constant permeability of vacuum μ_0;
- the passive structures $\{\Omega_{ps_i}\}_{(1\leq i\leq N_{ps})}$ with conductivity $\sigma_i \neq 0$;
- the PF coils regions $\{\Omega_{c_j}\}_{(1\leq j\leq N_c)}$ with respective areas A_j and currents I_j;
- the limiter region Γ_L, where the plasma exists before reaching the walls;
- the plasma domain $\Omega_p \subset \Gamma_L$, which is unknown and defined by its boundary, namely the last closed iso-contour of ψ. The latter is either tangent to Γ_L or bounded by the presence of an X-point. The whole equation to solve is then referred to as the *Grad-Shafranov* equation (GS):

  $$-\Delta^*\psi = Rp'(\psi) + \frac{1}{\mu_0 R}ff'(\psi)$$

  with major radius R, pressure $p(\psi)$, and diamagnetic function $f(\psi) = RB_\phi$. Both p and ff' are 0 outside Ω_p.

[4] Newton direct and Inverse Computation for Equilibrium

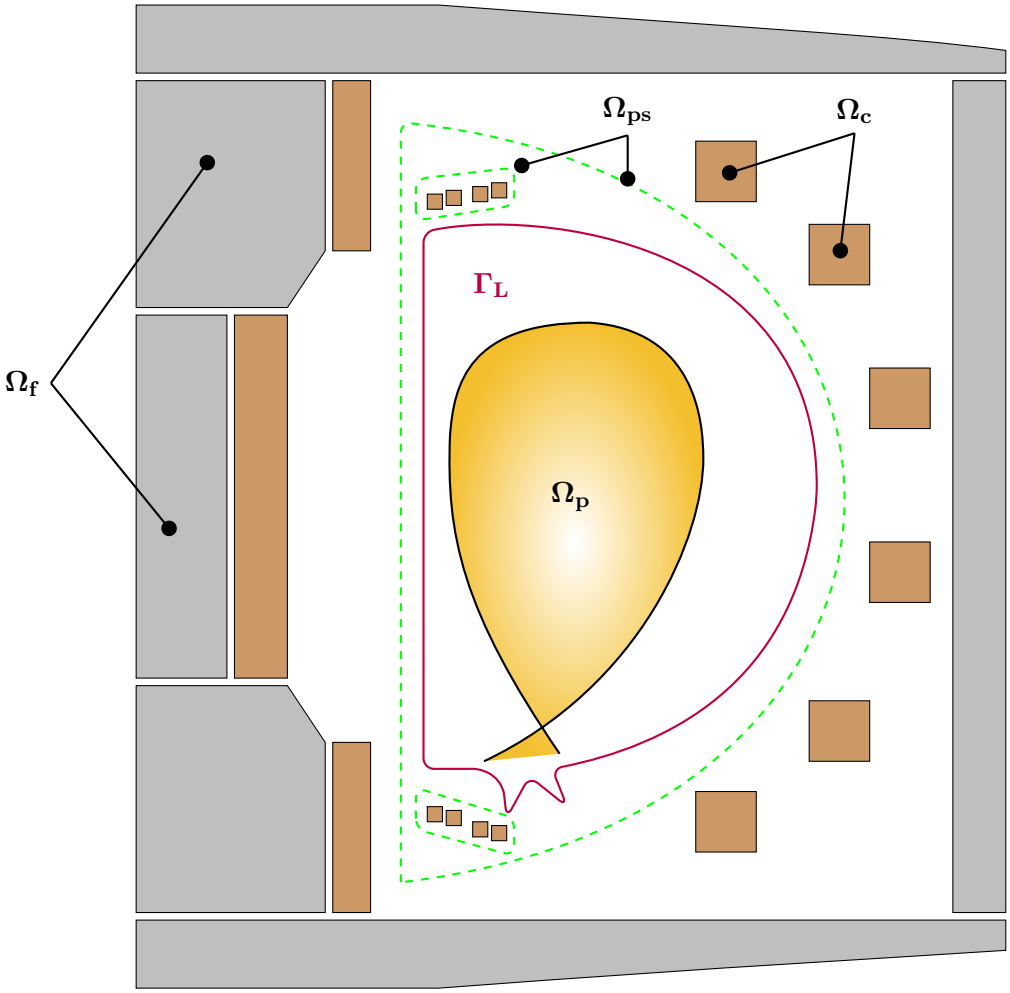


Figure 3.6: Schematic view of a tokamak poloidal plane, as modeled in NICE. Ferromagnetic iron structures (gray) introduce a non-linearity in the equilibrium equation. The passive structures (dashed contour) notably include the vacuum vessel. The latter contains the limiter region Γ_L (purple), where the plasma domain Ω_p lies. Coils surround the vessel (brown rectangles), and the two divertor coils are logically inside the chamber.

We can rewrite (3.1) so that it involves all domains:

$$-\Delta^*\psi(r,z,t) = \begin{cases} rp'(\psi,t) + \frac{1}{\mu_0 r} ff'(\psi,t) & \text{in} \quad \Omega_p \\ -\frac{\sigma_i}{r}\frac{d\psi}{dt} & \text{in} \quad \Omega_{ps_i} \\ -\frac{1}{A_j} I_j(\frac{d\psi}{dt},t) & \text{in} \quad \Omega_{c_j} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

A precise depiction of WEST geometry and related domains is presented in Appendix C. Part of the RL interaction loop described in Chapter 2 implies sending actions to

the NICE environment so that plasma evolution can be computed. The agent predicts the input voltages, which appear in the expression of coil currents. Taking into account mutual and self induction through circuit equations [53], $I_j(\frac{d\psi}{dt}, t)$ is expressed as:

$$I_j(\frac{d\psi}{dt}, t) = \sum_{k=1}^{N_s} \mathbf{S}_{jk} V_k(t) + \sum_{l=1}^{N_c} \mathbf{R}_{jl} \frac{1}{A_l} \int_{\Omega_{c_l}} \frac{d\psi}{dt} dr dz, \qquad \forall i \in 1, \dots, N_c$$

with N_s the number of power supplies, $\mathbf{S}$ of dimension $N_c \times N_s$ and $\mathbf{R}$ of dimension $N_c \times N_c$. It is worth mentioning that equation (3.2) is parabolic inside the coils or when the plasma evolves through time, but elliptic in static cases. In Γ_L, the magnetic axis of the plasma is taken as the global minimum of ψ, i.e. $\nabla\psi = 0$, and X-points are saddle points of ψ. Given prescribed $p(\psi)$, $ff'(\psi)$ and initial conditions $\psi(r, z, 0)$, we solve (3.2) at each timestep for ψ such that $lim_{||(r,z)||\to\infty}\psi(r, z, t) \to 0$. The timestep is always set to $\delta t = 10^{-3}$ s, following the frequency at which the real plasma control system of WEST works. The equation is discretized using a P1 finite element method based on a triangular mesh of the computation domain [15]. More precisely, the triangulation is performed on a restricted spatial domain named ABB[5], which is enclosed in a semi-circle containing the whole geometry of the tokamak, i.e. $\Omega_p \cup \Omega_p \cup_i \Omega_{ps_i} \cup_j \Omega_{c_j}$. In addition to the P1 approach, a second finite element method is implemented. It couples C^0 piecewise linear Lagrange finite elements in a region without the plasma, and C^1 piece-wise cubic reduced Hsieh–Clough–Tocher finite elements elsewhere [45]. Ultimately, it stabilizes the numerical scheme but is computationally more expensive, which explains our focus on the P1 formulation. Nevertheless, the latter also displays long training times that we compensate for through enhancements on the RL side (See Chapter 4).

NICE modes and initialization procedure

NICE features several modes, each with a specific purpose regarding tokamak operations, or more specifically for the training procedure used in PILOT (Table 3.1). The inverse and reconstruction modes do not solve directly the Grad-Shafranov equation. They perform least-squares optimization to minimize a cost function specified by the user, so that the equilibrium equation in the ABB domain is satisfied. The general formulation of this cost function could be simplified as such, with C the overall cost, D a quadratic term to make an iso-contour run through the desired boundary points $\{(R_k, Z_k)\}_{0\leq k\geq N_b}$ and R a regularization term for large coil currents $\{I_j\}_{0\leq j\geq N_c}$:

[5]It stands for *Albanase, Blum, de Barbieri* who first introduced the semi-circle integral method used to consider the previous conditions at infinity [6]

$$C = D + R$$
$$D = \frac{1}{2}\sum_{i}^{N_b}(\psi(R_i, Z_i) - \psi(R_0, Z_0))^2$$
$$R = \frac{1}{2}\sum_{i}^{N_c} w_i I_i^2$$

Mode	Purpose	Input	Output	Usual applications
Inverse	Find coil currents to match a desired plasma shape	Desired shape I_p p' and ff'	I_{coils} $\psi(R,Z)$	Development of magnetic configurations
Inverse evolution	Find coil currents and voltages to match a desired plasma shape evolution	Desired shape evolution $\psi(R,Z,t_0)$ $I_p(t)$, $p'(t)$ and $ff'(t)$	$I_{coils}(t)$ $V_{coils}(t)$ $\psi(R,Z,t)$	Not used in practice (too costly and difficult to fine tune).
Direct	Compute one equilibrium at a given time	I_{coils} I_p p' and ff'	$\psi(R,Z)$	Study of equilibrium stability
Direct evolution	Compute the equilibrium evolution	$\psi(R,Z,t_0)$ $V_{coils}(t)$ $I_p(t)$ $p'(t)$ and ff'(t)	$\psi(R,Z,t)$	Controller design and scenario development
Direct evolution with resistive diffusion	Refined computation of the equilibrium evolution	$\psi(R,Z,t_0)$ $V_{coils}(t)$ $p'(t)$	$\psi(R,Z,t)$	
Direct evolution with resistive diffusion and transport equation	Physically refined computation of the equilibrium evolution	$\psi(R,Z,t_0)$ $V_{coils}(t)$	$\psi(R,Z,t)$	
Reconstruction	Compute the equilibrium with given measurements	Magnetic measurements I_{coils} $I_{passive} = 0$	$\psi(R,Z)$ p' and ff'	Plasma state and shape estimation for real time control, diagnostic treatment and physics analysis

Table 3.1: NICE modes can be employed for theoretical studies and practical uses within the WEST operating routine.

In this PhD, the NICE code is used within a reinforcement learning loop, meaning interactions with the chosen MPO agent are performed through episodes. Ideally, the latter must be informative enough to make the data gathered in the replay buffer, useful at best for policy optimization. Nevertheless, one must realize that there is no guarantee that the simulation will converge because of possible local minima in the resolution of the Newton formulation. This could lead to numerical instabilities, shortening episode duration, while the agent's actions could have been efficient. Hence, a standard procedure is defined at the initialization of each new execution. This strategy ensures proper convergence of NICE for RL, either to reach the full desired duration of an episode, or at least to support enough direct evolutive iterations to shape informative data sequences for the agent (Figure 3.7). The procedure always follows the same pattern no matter what the scenario can be:

1. an inverse iteration based on configuration files for the initial shape is computed. Consequently, optimized currents of the poloidal system are obtained as an initial guess in preparation for the next step;

2. a first direct computation is performed to safely check that the previous equilibrium is a good starting point. Profiles are initialized from the ones obtained at the previous step, that is to say, normalized ψ_N, $p'(\psi)$, and $ff'(\psi)$;

3. the direct evolution starts, with initial profiles copied in the same way for consistency.

Moreover, executions (and episodes as a consequence) are always launched from configuration files extracted from recent experimental data, so that the configuration files contain specifications that are physically reachable on the WEST tokamak.

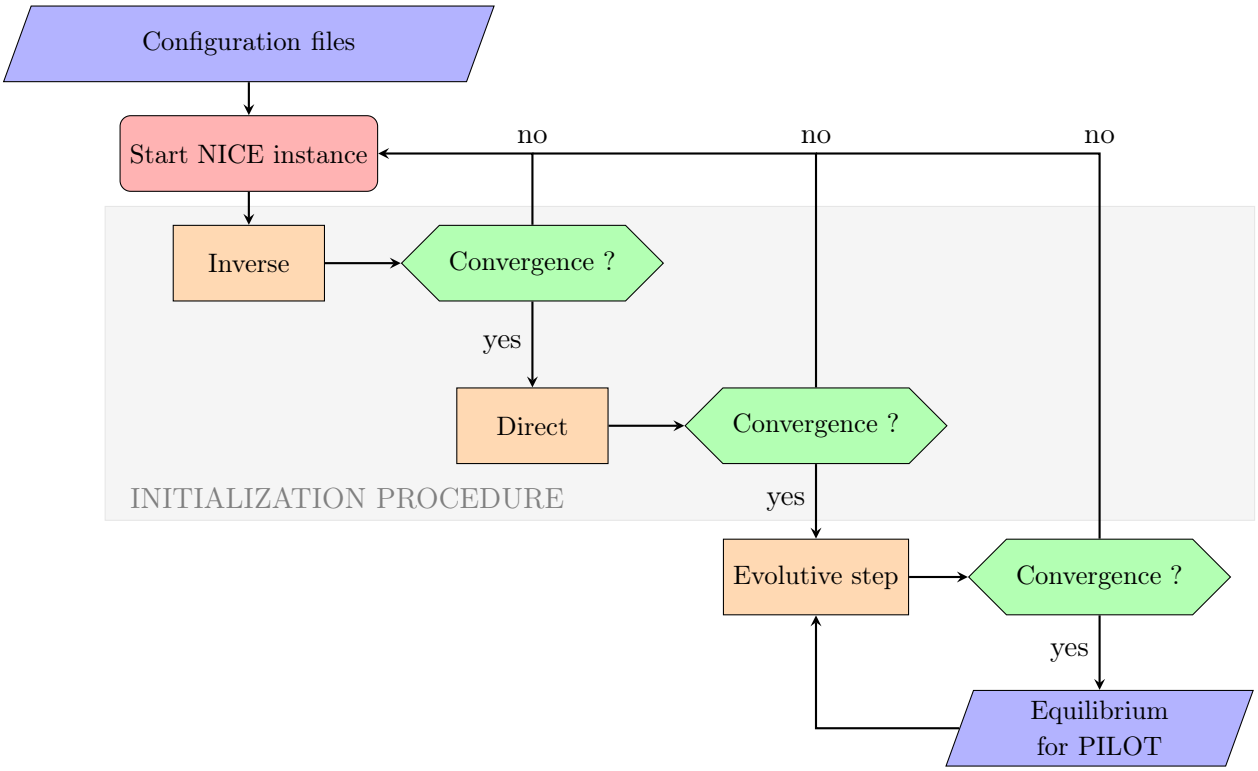


Figure 3.7: Summary of the interaction loop that is happening in the NICE environment, including convergence checks during initialization.

Improving numerical stability

The initialization strategy ensures that the computational workflow of the NICE environment is efficiently set. Despite its presence, NICE still exhibits numerical instabilities in

all modes involved. Over time, NICE was augmented with resistive diffusion [52], to better model plasma evolution. In classical formulations, p' and ff' must be prescribed as input data. Out of the two, ff' can lead to strong numerical instabilities. In most works, this term is computed from the resistive current diffusion equation, with the flux surface average Grad-Shafranov equation. Nonetheless, the resistive current diffusion equation is usually part of a transport model, which evolves parallel to the magnetic equilibrium one. This makes it difficult to ensure numerical stability since two representations are required for the poloidal flux. NICE implements an evolution equation directly for f [45], which leads to the following expression. Given initial conditions $f_0(t)$, and $\forall g(\psi) \text{so that} g(\psi_{LCFS}) = 0$:

$$\text{Find } f(\psi,t) \text{ w.r.t. } \int_{\Omega_p} \frac{df(\psi,t)}{dt}\frac{g(\psi)}{r}drdz + \int_{\Omega_p}\frac{d\psi}{dt}\frac{f(\psi,t)}{r}g'(\psi)drdz$$
$$+ \int_{\Omega_p} \eta^{||}(rp'f(\psi,t)) + \frac{f^2(\psi,t)+|\nabla\psi|^2}{\mu_0 r}f'(\psi,t) - r\mathbf{j}_{ni}\cdot\mathbf{B})g'(\psi)drdz = 0$$

with non-inductive current density $\mathbf{j}_{ni}$, resistivity $\eta^{||}$ (the parallel component coefficient of the anisotropic resistivity tensor of the plasma) and flux at the plasma boundary ψ_{LCFS}. $\mathbf{j}_{ni}$ originates from current sources created by additional heating systems, namely the ECRH and ICRH on WEST. Furthermore, $\mathbf{j}_{ni}$ is not well known, so it is approximated through $j_{ni}(\psi)\cdot|B|$ with $j_{ni}(\psi)$ an analytical formula discussed later in this chapter. An abstract analytical formula also represents $\eta^{||}$. This guarantees a more complete and physically accurate representation of the plasma's behavior, accounting for the dynamics of the plasma's magnetic flux profiles. In previous research found in the literature, different models have been defined, such as a 0D flux consumption model where a simple lumped-circuit equation describes the plasma current time evolution [85, 29]. The present extensions display noticeable benefits compared to previous alternatives, as they represent the current density distribution in the plasma. Hence, it better models the magnetic field lines' evolution over time as they diffuse, which is crucial for better simulation during each control scenario. In addition, it gives a more representative evolution of the total plasma current:

$$I_p := \int_{\Omega_p}\left(Rp'(\psi) + \frac{1}{\mu_0 R}ff'(\psi)\right)drdz$$

with Ω_p defining plasma domain. However, resistive diffusion does not entirely prevent convergence issues within the P1 formulation, as it sometimes results in sudden spikes of plasma current up to several kA [63] and of current density, which are not expected from reality. To overcome this problem, NICE was recently extended with an electron energy transport equation [45]. Through the definition of the electron temperature profiles, it realistically computes pressure profiles p, and better approximate $\mathbf{j}_{ni}\cdot\mathbf{B}$ and $\eta^{||}$ [100]. This approach refines the entire simulation as profiles include physical information instead of abstract representations. It results in even more stabilized and realistic calculations, which benefit the framework. Hence, the information observed by the agent is more in line with reality.

Diversity among samples

The agent used in this study appeals to a replay buffer, in which data from the NICE environment is stored. Once sampled, they are used by the MPO algorithm to optimize the behavioral policy. A lack of diversity among the samples could result in more extended training at best, or convergence towards a non-optimal policy in the worst-case scenario. Indeed, generalization and performance could rapidly drop if the observations processed by the learning algorithm are too similar, or if a few examples drift from the average trajectory displayed by the current behavior. There is a need to promote sample diversity throughout each episode. To do so, let us recall the said NICE enhancements using resistive diffusion and electron transport. Without the latter, p' and ff' are represented by $\mathcal{A}$ and $\mathcal{B}$, linear combinations of a set of basis functions defined on $[0, 1]$ which do not follow the same formulation depending on the involved mode. In the reconstruction mode, they can be piecewise linear or cubic splines, for example. In inverse, direct and evolution ones, which are used extensively in our initialization workflow, $\mathcal{A}$ and $\mathcal{B}$ can either:

- be prescribed by given profile points in the configuration files, and linearly interpolated afterwards;
- rely on parametric formations of the form:

$$\mathcal{A}(x) = \beta(1 - x^\alpha)^\gamma$$
$$\mathcal{B}(x) = (1 - \beta)(1 - x^\alpha)^\gamma$$

The second option is employed exclusively in this work, and proper randomization of α, β, and γ paves the way towards the desired diversity, encouraging initial profiles to vary between episodes. Nonetheless, one must remember that the NICE solver displayed convergence issues without its enhancements, so there must be regions of this 3D hyperparameter space where convergence is unlikely to happen. So, to perform the said randomization correctly, a stability search is conducted under variations of α, β, and γ for both $\mathcal{A}$ and $\mathcal{B}$. The same kind of layout is present on different simulators like FEEQS [53]. Analyzing parameters on these software shows that $\alpha \in \{0.5, 1, 2\}$ depending on plasma's type, $\beta \in [0.1, 2]$ adjusting the normalized pressure, and $\gamma \in [0, 3]$. In FEEQS, the latter is empirically linked to plasma's internal inductance through $\gamma = 2 \times l_i - 1$ with $l_i \in [0.5, 2]$. We then perform 200 runs of 200 timesteps kept at 10^{-3} s each, sampling a triplet at each new execution and recording failures occurring at any step of the initialization procedure. The time threshold is meant to check if the initialized profiles are well posed so that the execution goes further than the first steps of the starting procedure, which are the inverse and the direct operations. Indeed, the evolutive part could also crash before reaching a decent number of iterations. In this 3-dimensional parameter space, the obtained hypercube (Figure 3.8) is used to sample triplets at each episode during training, which minimizes the probability of NICE not converging. When resistive diffusion is present, let us recall that parameterized equations are employed for $\mathbf{j}_{ni}$ and $\eta^{||}$, which are required to compute f and chosen to be:

$$\eta^{||} = \frac{a}{(b - \psi_N)}$$
$$\mathbf{j}_{ni} \cdot \mathbf{B} = \max_{\psi} j_{ni}(\psi) \cdot |\mathbf{B}| \times (1 - \psi_N)$$

In the same way as for α, β and γ, we study NICE convergence under the randomization of a and b, with $\max_\psi j_{ni}(\psi) = 10^5$. The exact same runs are computed, with a sampled within $[10^{-9}, 10^{-8}]$ and b from $[1.005, 1.01]$. These intervals were identified after many attempts outside of their bounds were unsuccessful. This ends up in extended bounds, from which additional sampling of the two parameters helps promote diversity among the related profiles. Since this grid search is performed among all modes and for all shapes of interest, NICE stability becomes empirically bounded for this project.

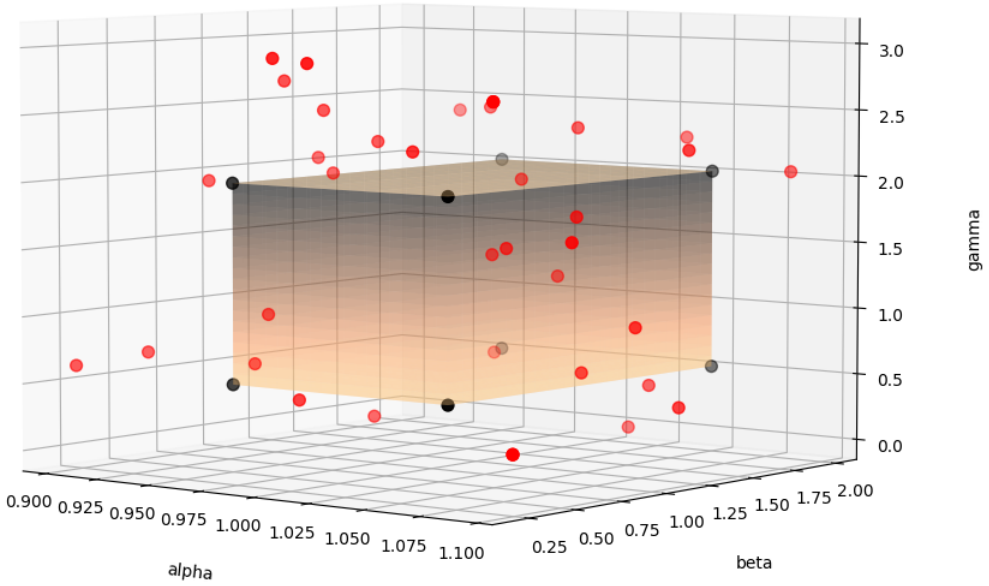


Figure 3.8: All points are execution examples which stopped at a certain step of the RL loop, either at initialization or during the evolutive part. The hypercube minimizes the number of failed attempts.

We must notice that p' and ff' do not evolve and are kept as such after getting randomized at initiation. The same idea can be said about $\eta^{||}$ and $j_{ni} \cdot B$ which are also fixed parameters. In fact, the second step of the initialization procedure marginally differs when resistive diffusion is considered since p and ff' are obtained through intermediate basis decomposition of f, p, η, $j_{ni} \cdot B$. When both resistive diffusion and electron energy transport are included, the initialization procedure still needs the randomization of initial profiles. Once the evolution section is reached, the evolving p and ff' profiles are then

directly part of the computation. Yet, most of the analyses conducted in this PhD and presented in Chapter 5, did not benefit from the electron energy transport equation, which explains why the previous procedure is put on show. One could have randomized the initial plasma shapes (position, LCFS coordinates, etc), but the starting references are fixed instead. Indeed, they have been specifically chosen for their numerical stability, as random updates could have harmed the entire procedure. Moreover, extending the previous searches for the whole plasma geometry is impractical, as we observed shapes between two functional ones from which NICE can not converge. Most importantly, the breakdown and formation of the first plasma are always the same: we can reasonably assume that the shape of the plasma will always be similar once it is formed and the trained policy takes over. In addition, the internal profiles and the resistivity depend on the device conditioning, the impurities, and a lot of uncontrolled parameters, which justifies even more why the previous random search is worth doing.

Assessing NICE performance

Several NICE parameters influence the average calculation time for magnetic equilibria. A parallel study was conducted to select values prioritizing calculation speed while limiting the risk of precision loss. Considering that NICE is expensive in terms of its calculations, a first optimization of the stopping criterion for the Newton solver is done. This threshold is set to 10^{-5} instead of the default value of 10^{-10}. Thus, it limits the maximum number of iterations considered for convergence in each direct evolution computation without losing accuracy in its inputs. Choosing the timestep depends on the control objective and may impact solver convergence. If the step size is too large, the situation will not be modeled correctly, and invalid information may be obtained. Conversely, a smaller step size will stabilize the simulation at the risk of increasing calculation times. After a certain threshold, reducing the step size will no longer modify the result, but the calculation time will continue to grow. Here, the timestep has been set to 10^{-3} second, as a value of 10^{-4} second is unnecessary. Indeed, this value enables us to model the learning process with the characteristic times observable on WEST while matching the frequency of the control system.

The mesh size directly impacts the search time when identifying nodes useful for interpolating a quantity like magnetic flux. The mesh is generated within the ABB domain, as presented earlier in this section. To do this, the mesh boundary must be defined by several points, up to 500. After several trials, 17 points look sufficient to achieve stable generation at the environment launch. Increasing or decreasing this number might augment the number of mesh generation failures, thus blocking the training launch. This research extended to the desired area for each triangle of the structure within the computational domain. This size corresponds to $3 \times 10^{-4}\ m^2$ in the limiter domain, and $2 \times 10^{-3}\ m^2$ in iron structures, PF coils and the vessel around the limiter domain. Finally, $2 \times 10^{-4}\ m^2$ is retained in passive structures, and $5 \times 10^{-3}\ m^2$ in the remaining space of the ABB domain. This way, the mesh is generated efficiently and finely enough to obtain accurate results without unnecessarily increasing size.

These specifications lead to an average computation time of 3 seconds per timestep for

each direct evolution step. However, this only considers equilibria computed in the middle of the vacuum chamber. It does not consider locations of the latter where equilibrium computation might be more difficult than others, especially at the start of training where the agent moves the plasma in a sub-optimal manner. With that in hand, the new average computation time per timestep was measured at 10 seconds. Adding resistive diffusion and electron energy transport equations increases it to 15 seconds. It is worth mentioning that an initial analysis showed that the previous combined mode computed each equilibria in 13 seconds on average. This was refined by looking more in-depth at the said plasma locations, as several pathological trajectories that increase computation time were not initially considered. Moreover, shared computational resources challenged the precise computation of the average computation time. So, this update considered different server loads.

Finalizing the environment

Once all of this is set up, is it enough for a real numerical twin? The environment only contains a simulation of the plasma's equilibrium evolution without considering what links the structure to actuators and diagnostics. For this purpose, power supply and diagnostic models are incorporated to represent the plasma control system on WEST accurately (Figure 3.9). The power supply model is kept simple with only a few parameters of the poloidal coils system reproduced from the real plant, including delays. This is programmed as a cycling matrix of dimensions $N_c \times d$ with d being the number of delayed timesteps for each coil, and $N_c = 11$ in the WEST case. This comes down to periodically shifting columns of the matrix with vectors containing the actions predicted by the agent for each of the 11 poloidal field coils. Moreover, limits on their currents and voltages are considered at 95% of the operational limits defined on WEST. This is done to replicate safety control requirements induced through the terminal conditions inherent to the RL loop. In addition, it characterizes the action penalty used for the agent in Chapter 2. The terminal conditions are triggered if the thresholds are reached or the safety factor at 95% of the magnetic flux, namely the q_{95}, goes below 2. The regular safety factor expresses the ratio of the number of poloidal turns per toroidal cycle on a field line running over a magnetic surface. Is it known that the plasma tends to disrupt when I_p/B_ϕ exceeds a limit expressed by the edge safety factor, conveniently expressed as $q_{edge} \propto \frac{a^2}{R}/\frac{B_\phi}{I_p}$ with B_ϕ the toroidal magnetic field, I_p the plasma current, and a, R minor and major radii. This limit equals 2 for limiter plasmas [90], but does not hold for diverted configurations. In the latter, we must use the safety factor just inside the LCFS, namely the q_{95}. A low factor might end up in a squished plasma that is unmaintainable. If triggered, the environment is reset, and the initialization procedure is performed.

Diagnostics are simulated through the direct mode, in which a specific tool has been developed (Table 3.1). It is worth discussing that the environment's state is initially defined as $s = \{y, I_a, m\}$ with y the plasma equilibrium information, I_a the currents in the active control coils, and m the raw magnetic measurements. y typically contains all quantities of interest used to compute the holy grail of RL, namely the reward. However, s is usually tricky to observe entirely in real-time. To overcome this issue, the NICE environment is

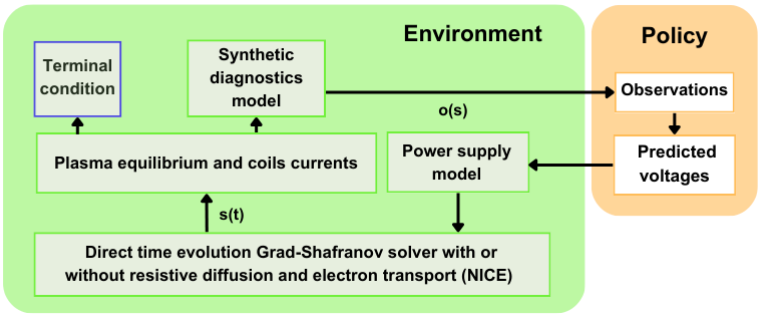


Figure 3.9: PILOT incorporates power supply and diagnostics models, so that training and inference can be performed. For example, a trained policy can be connected to the complete environment for tests or evaluation.

restricted to a *Partially Observable MDP* (POMDP) where the state space is limited to the observation space $\mathcal{O}$. As such, we have $o(s) = \{m_b, fl, I_a, \frac{dm_b}{dt}, V_{loop}\}$, with $\{m_b, fl\}$ magnetic probes and flux loops raw measurements, $\frac{dm_b}{dt}$, first-order temporal derivatives of a selection of magnetic probes signals, and the loop voltage V_{loop} taken as the temporal derivative of the first flux loop. As a side note, the loop voltage is the voltage created in a circular loop concentric with the plasma column due to the variation of the poloidal magnetic flux passing through it. More than the real-time observation issue, the agent will be able to learn directly from raw magnetic measurements, which aligns with the objectives stated in the introduction. Indeed, the purpose of using neural networks is to get rid of reconstruction codes during real-time operations. At each new equilibrium computation during the direct evolution part, synthetic data is generated, reproducing what is seen on the data acquisition system of WEST. More precisely, WEST boasts 110 magnetic sensors measuring the magnetic field locally in tangent and normal directions for each cross-section on which they are placed. Considering the axisymmetric assumption in tokamaks, and despite many sections, only one is considered for the final measurements (Figure 3.10). Similarly, 17 flux loops quantify the magnetic flux. At last, why is it interesting to add temporal derivatives of the magnetic probes if the previous sensors already give insights into the plasma's state?

The inclusion of temporal information leverages a missing aspect coming from static sensor data in the sense that the rate of change in the magnetic field measurements improves representation of the device dynamics. Combined with the initial raw measurements, they could leverage a robust identification of implicit plasma events during training. Even so all probes could have been retained, only 55 pairs of local measurements are uniformly sampled along the cross-section, to compute the said derivatives. Using half of the sensors reduces the agent's neural networks' potential input size, with enough physical details captured in the observations. Again, delays are implemented to ensure consistency within the actual plant. Finally, noise can be injected in $o(s)$ to model uncertainties and potential faulty diagnostics.

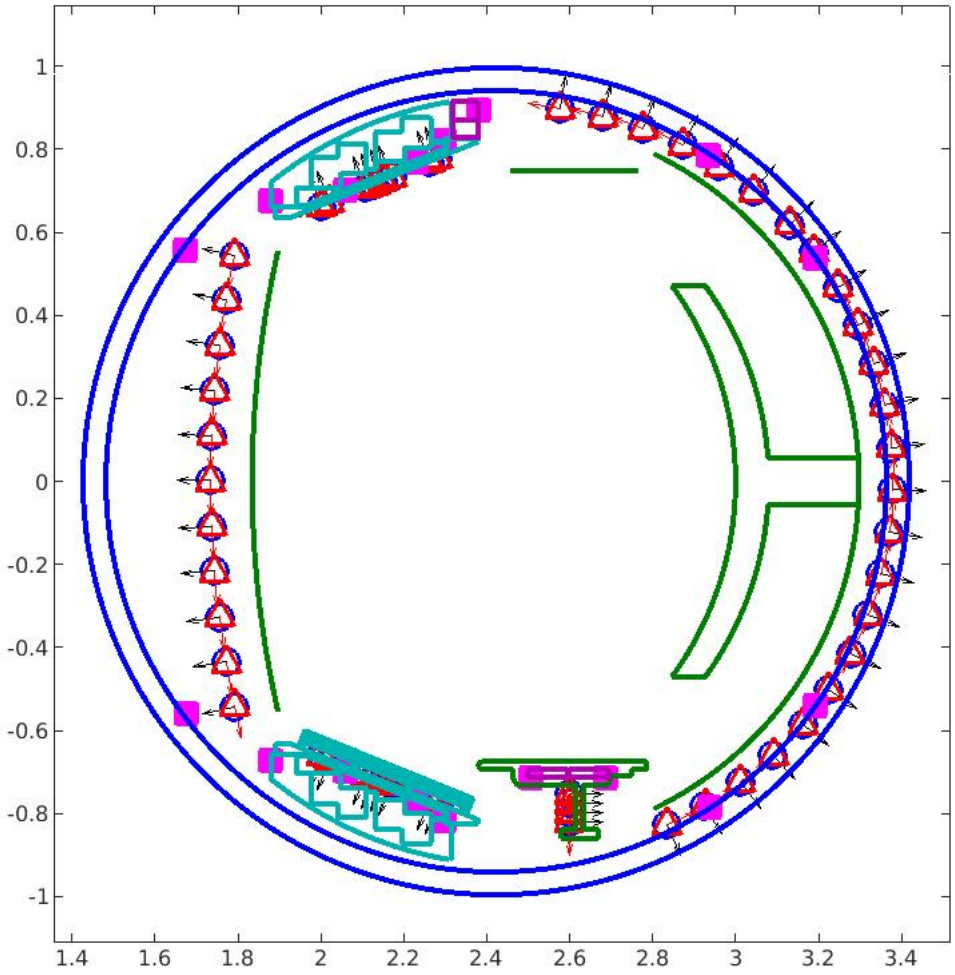


Figure 3.10: Magnetic probes with normal measurements (blue arrows) and tangential ones (red arrows), as well as flux loops (purple squares) circle the torus. We only consider one cross-section for the diagnostics model thanks to the axisymmetry assumption.

To conclude, PILOT uses an environment based on NICE, which computes the evolving magnetic equilibrium under incoming voltages for a timestep of 10^{-3} second, generating raw synthetic magnetic data for use by the other parts of the framework. This environment gives an accurate representation of the plasma, as well as the WEST control system. From now on, we will interchangeably refer to it as NICE or the environment.

3.2 In a world of scenarios and rewards

3.2.1 References generator

We want to train an agent capable of controlling plasma's shape, position, and current. Using RL means having a reward function designed to track such quantities along specified scenarios. However, to compute this signal, we must find a way to compare the agent's behavior with targets of interest. An easily modifiable reference generator is then integrated into PILOT. One must only specify snapshots depending on current needs and time intervals between them. Several examples of these snapshots can be found in Appendix A. The initial and final equilibriums can be both X-point or limiter configurations. Any specifications in between are possible, as intermediate sections are bounded by snapshots of the plasma containing all needed information. Intermediate equilibria, which compose the said sections, are linearly interpolated with respect to the environment timestep of 1 millisecond. By doing that, reward computation is available at every step of the training loop. The data structures produced by the generator store the 32 LCFS reference points, the elongation, limiter and/or X-point location, magnetic center coordinates, supplied voltages, as well as plasma and coil currents. On top of that, snapshots are taken from successful discharges extracted from the WEST database. In this way, references are not only feasible but also within the operational domain of the device. This builds an entire discharge with automatically generated transitions. In this work, four scenarios are showcased to answer the first problematic (Figure 3.11), closely related to the state-of-the-art and the routine operation on WEST:

1. *limiter maintain* is the baseline scenario for this PhD, as it looks for simple control of a stable plasma with all of its characteristics kept constant over time;

2. *limiter evolve* is looking for the displacement of the plasma magnetic center and boundary, varying the elongation while keeping all other information constant. This task exhibits an actual evolution of the targets, which gives insights on how the controller might track moving references;

3. *X-point maintain* is designed to evaluate the performance of RL-based solutions against references of an unstable elongated plasma kept constant over time;

4. *X-point evolve* corresponds to a classical transition between a limiter configuration and an X-point one; it is somehow the *final* level, because of the unstable moving targets.

All scenarios last for about 300 timesteps, as it appeared enough for generalization and inference on longer shots and to check if the trained controllers can handle vertical instabilities in the case of X-point transitions. For the latter, this duration is also close to actual transition times observed on WEST.

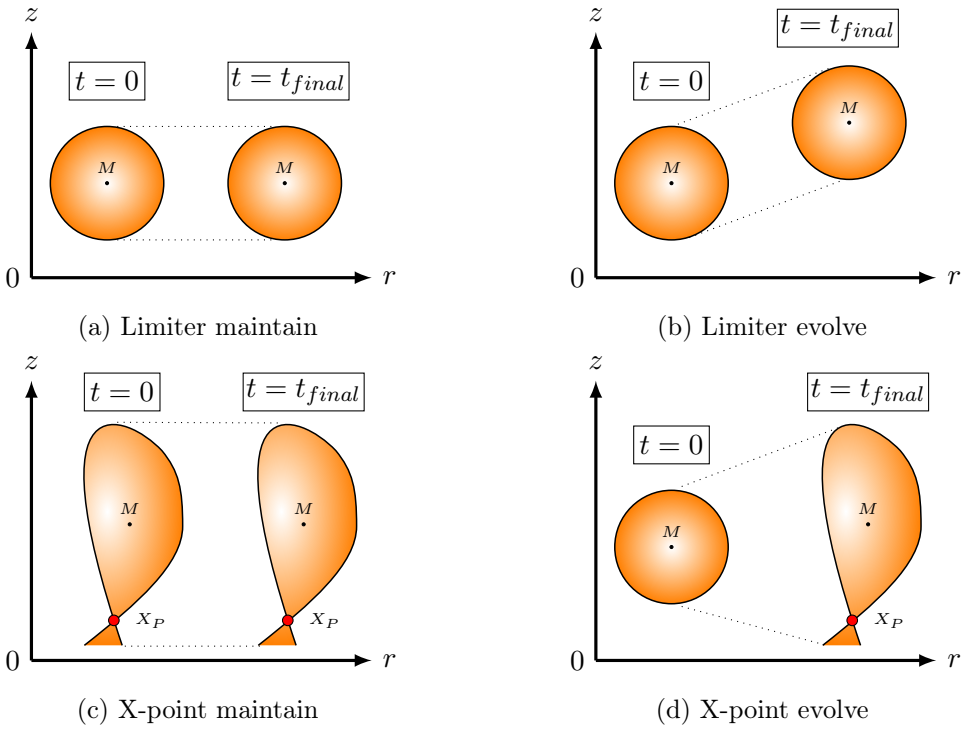


Figure 3.11: The four horsemen of RL-based control experiments.

3.2.2 Reward definition

Let us recall the classic RL interaction loop. The agent interacts with the environment through actions and obtains a state $s(t)$ and a reward $r(t)$ in return. The reward is a scalar feedback signal, which the agent uses to optimize its behavior in many ways, as described in Chapter 2. This work does not contradict this idea, as rewards are computed from $s(t)$, with target references $tr(t)$ coming from the reference generator. The chosen reward is a normalized combination of error signals, each describing a specific sub-task of magnetic control (shape, plasma current, etc). Each component c is computed as the absolute distance between its reference value $tr_c(t)$ and the corresponding state's property from the NICE environment $s_c(t)$. It is worth saying that while such distance is not differentiable, the remaining operations are meant to be. Since the agent tries to maximize the reward, we enforce positive ones, which worked best in our use case. Negative rewards could be used to model faster controller's response, but this potential objective is instead obtained by defining shorter transitions in the reference generator.

Even though the reward scale should not matter in theory, we observe steeper gradients as the range of reward values increases in practice. Such gradients might make the training process unstable compared to the initial networks' weights. Because of that, the final reward is normalized to $[0, 1]$ through non-linear transformations. Such a process goes through two steps: firstly, scaling is applied to each c using the *Softplus* function to get

sub-task rewards between 0 and 1. Suppose one component comprises several targets, like shape control using all last closed flux surface (LCFS) reference points. In that case, they are combined with the *Smoothmax* function to get a scalar value within the desired interval. At the end, a last combination is performed through the same function to get a final reward value within the range of interest. This bounds the reward, which is helpful for interpretation and stays in line with the reward assumption introduced in the probabilistic inference framework. The resulting method appears in Algorithm 1. It is worth mentioning that the final scalar is multiplied by 10^{-1} so that the maximum cumulative return for 100ms of plasma is 10. This last operation follows [29] and is useful to scale the reward signal well with the input features received by the critic during training. Finally, the agent gets a penalizing reward equal to -5, scaled accordingly to 0.5 when terminal conditions are triggered.

Looking at both functions, nonlinearity plays an important role. Ultimately, the procedure starts from errors computed as distances between targets and references. It would be beneficial to map the said distances to $[0, 1]$ and intelligently combine them, either promoting exploration or boosting precise control through exploitation. With that in hand, functions defining the reward computation should allow such branching depending on the overall scenario:

- for each component, one must let the agent know that there is wide room for improvements. Conversely, the agent should sometimes stay within narrow error bounds, by which the reward evolves steeply from a good behavior to a bad one;

- for combination, there should be a mechanism that follows the same principle. The agent could consider all components equally, which can be useful for proper exploration. On the contrary, it could focus on the worst component of them all at a given time so that precise tracking of the latter can be achieved.

Algorithm 1 Reward calculation pseudo-code

C, set of reward components, TR set of corresponding targets, W set of corresponding weights.

COMPUTE(C)
$R \leftarrow \{\}$
for all $c \in \mathcal{C}$ **do**
  if c scalar **then**
    $E \leftarrow |s_c(t) - tr_c(t)|$
    $R_c \leftarrow SOFTPLUS(E)$
  else
    $R_c \leftarrow SMOOTHMAX(\{SOFTPLUS(R_{c_i})\}_{1 \leq i \leq size(c)}, 1)$
  end if
  APPEND(R, R_c)
end for
return $SMOOTHMAX(R, W)$

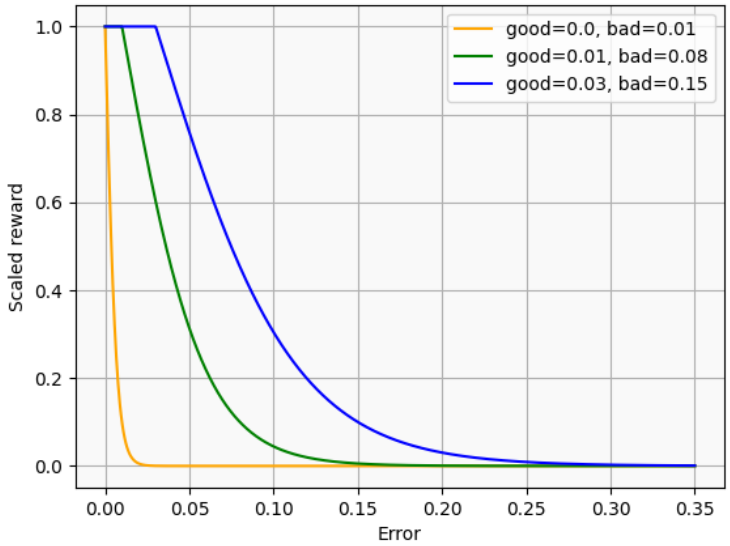


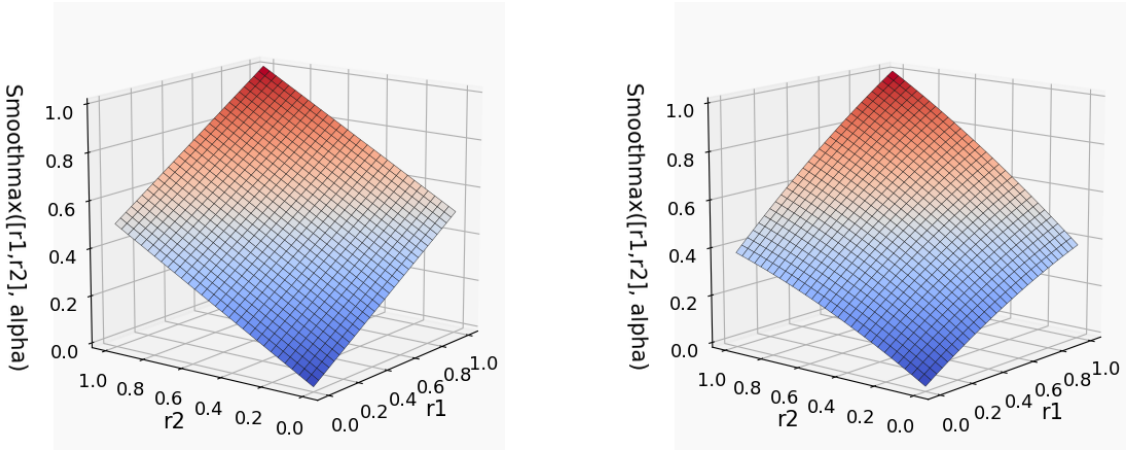
Figure 3.12: *Softplus* behavior. If the error is smaller than the good parameter, the reward will saturate to 1. If worse than bad, the reward decays to 0. ξ describes scaling steepness between the two anchor points and is fixed at $-log(19)$ like [29]. This value is chosen so that the *bad* parameter leads to a reward of 0.1 out of 1.

The two transformations used in this PhD explicitly act on two sides of this coin. More precisely, *good* and *bad* parameters in the *Softplus* formulation appears as core elements of the tool:

$$Softplus(e) = min(max(2 \cdot s, 0), 1), \text{with} \begin{cases} s = \sigma(-\xi(\frac{e-good}{bad-good})) \\ \xi = -log(19) \end{cases} \tag{3.3}$$

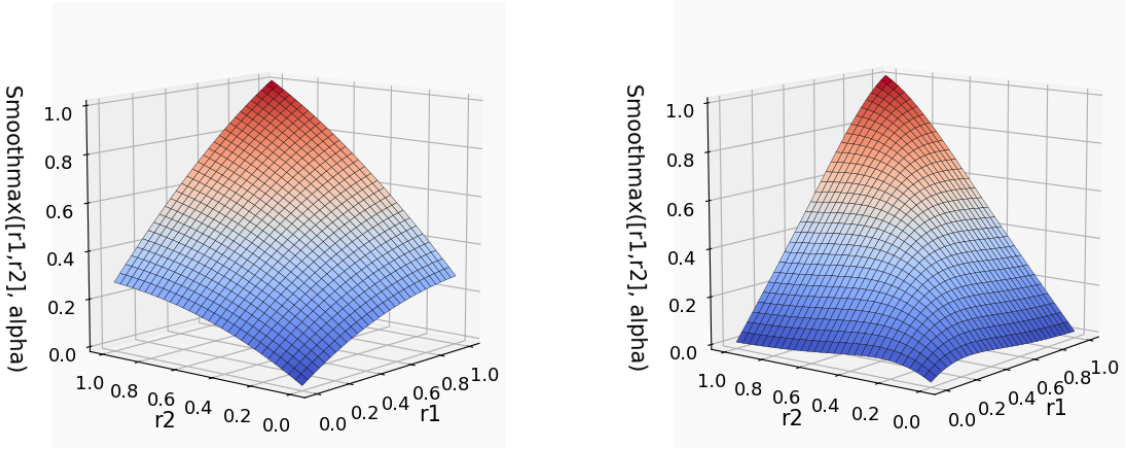
This operation scales the reward signal nonlinearly according to regions of interest in the reward space (Figure 3.12). Tight values in both parameters of the component will lead to higher focus to achieve high related reward, and smoother values will make the components easier to satisfy. Consequently, if the related error is below *good*, the corresponding reward will always be 1, meaning that anything in this region of the reward space can be considered a proper behavior towards the control objective. If the error goes above the *bad* threshold, the reward rapidly decays from 0.1 towards 0. This penalizes the current behavior but lets the agent attempt to recover in a small room for improvements. In between, the reward scale depends on the width of the parameters' interval, with an apparent drop-off passing the good parameter. The steepness is parameterized by ξ and set to $-log(19)$ following several standards proposed by [29]. To conclude on this matter, close parameters for the *Softplus* function help look for precise control, making it difficult to get valuable signals when the training scenario involves substantial variations, and exploration should occur. Oppositely, wide intervals favor in-depth exploration at the cost of precise control through exploitation. In the case of the present scenarios, *limiter maintain* and *X*-

point maintain exhibit initial conditions similar to their final targets. Choosing relatively tight intervals for each reward component is suitable since the region of interest does not need extensive exploration. For *limiter evolve* and *X-point evolve*, they explicitly show differences between initial and final target states. With such a context, exploration is needed and will be preeminent to gain efficient knowledge of the reference trajectories and their surroundings. In fact, *Sigmoid* and asymptotic scaling functions were tested, but the use of the *Softplus* function is more suitable, especially regarding interpretability.



(a) $\alpha = 0$

(b) $\alpha = -0.5$



(c) $\alpha = -1$

(d) $\alpha = -5$

Figure 3.13: $Smoothmax(r_1, r_2, \alpha)$, with r_1, r_2 scaled reward components in $[0, 1]$. Focus is directed towards the worst component as $\alpha \to -\infty$. Such nonlinear scaling allows the refinement of objectives specification during training.

Then, the *Smoothmax* transformation allows for a non-linear combination of its inputs:

$$Smoothmax(\mathbf{r}, \mathbf{w}) = \frac{\sum w_i r_i e^{\alpha r_i}}{\sum w_i e^{\alpha r_i}}$$

Weights affect the importance of each reward component, while the α parameter creates a balance between them (Figure 3.13). This setup becomes even more critical as each component, i.e. control tasks, does not display the same level of difficulty or the same importance regarding a configuration of interest. Indeed, a negative value will shape final rewards close to the least performing component, leaving others vaguely explored. Such trade-offs are essential since the closer α is to 0, the more all components are treated equally, i.e. all control objectives are put on an equal footing. The positive case should be discarded, as it entirely excludes any component that will perform badly. Since each kind of scenario displays a specific usage, the previous dilemma between exploitation and exploration is raised accordingly. With this aim in mind, let us compare both plasma configurations of interest to get a better intuition of *Smoothmax* utility. For instance, when the plasma is diverted, the LCFS component would be closely related to the X-point location component because both are intrinsically linked to the plasma's geometry. If the plasma is limited, we could draw the same idea regarding the LCFS and the reference elongation. In each situation, exploring the two components is correlated: this does not mean we should remove one of them, but it is pretty intuitive to consider all tasks equally. It becomes imperative when reference targets variate sufficiently between a scenario's beginning and end, as no components should be favored. This is directly linked to the mandatory nature of exploration in the *evolve* scenarios since the agent explores possibilities at the potential risk of losing accuracy. This loss could happen on several components to get better capabilities on others. Furthermore, knowledge acquired for one component does not interfere with what comes from the second one, and exploration remains general enough. However, in *maintain* use cases, initial conditions are the same as final targets, which favors focusing on precise control immediately at the start. The ini-

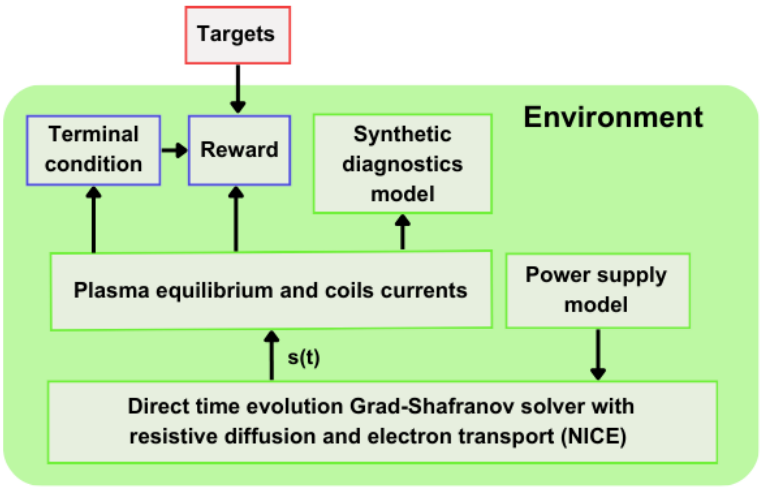


Figure 3.14: Full environment with target generation, to mimic WEST and its control system.

tial intuitive understanding of *Smoothmax* is put at half-mast. Even if components might be correlated, the system should focus on the current worst component to increase the reward without needing in-depth exploration by the agent. Many transformations could have been used for the final nonlinear combination since mappings from $\mathbb{R}_+^* \to [0, 1]$ can be drawn out of several possible operators (Softmax, Maximum, etc). This work relied on *SmoothMax* as it is flexible enough to approximate the common maximum if needed, while being convex. This latter point is an important feature with regards to neural networks since it can be differentiated many times. For a description of each component and how they are configured for all scenarios, please refer to Appendix D.

Training RL-based magnetic controllers requires building an environment mimicking WEST in the most realistic way. After receiving voltages processed by a power supply model, NICE advances in time the equilibrium, which data is used for the final reward calculation and to check terminal conditions. Thanks to the reference generator and the reward design with a tolerance over control accuracy, this environment (Figure 3.14) is ready for full integration into PILOT.

3.2.3 A digression on the WEST plasma control system

Previously, we stated that reference scenarios are sequences of transitions linearly interpolated between several desired snapshots, from which the reference generator yields targets for reward computation. The latter's definition employs multiple parameters (*Good*, *Bad*, α), which allow to refine the granularity of objectives we are aiming at. It is worth building a parallel with the WEST plasma control system to understand how the previous part of the framework relates to a certain logic already seen in real devices.

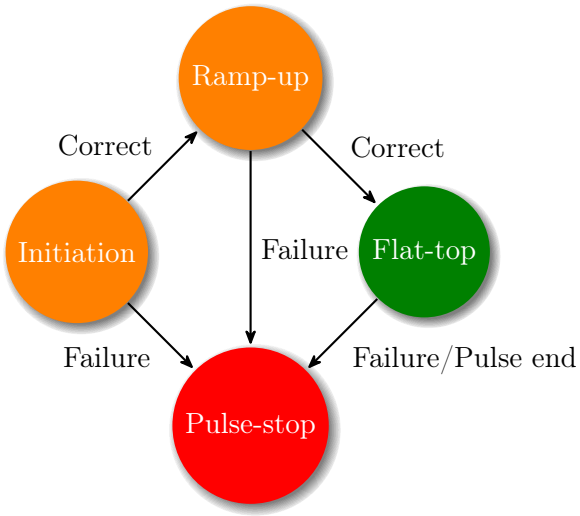


Figure 3.15: An example of the event scheduler with segments defined as cells, which transition depending on which even has been triggered.

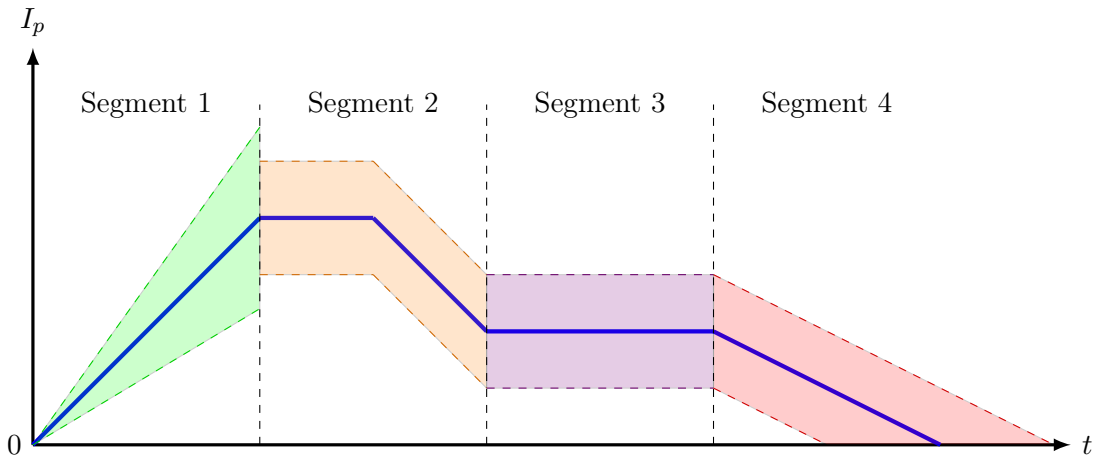


Figure 3.16: For example, a plasma current reference trajectory is made out of several segments, integrating different envelopes accounting for control tolerance.

Several parameters must be maintained close to some desired references to achieve a plasma discharge on WEST. This is done by managing actuators to react to specific situations and ensure machine protection. In practice, the *Plasma Control System* (PCS) handles magnetic control, with specifications on expected or unplanned events (disruptions, coils overheating, etc). For this, it must implement a real-time event manager (Figure 3.15) while keeping its structure flexible enough for future advanced control methods that might become useful. Like many other devices, diagnostics gather data in real-time not only for control but also for data storage. On WEST, this interplay between the diagnostics, the controllers, and the event manager goes through *segments*. One segment contains the variations occurring in the PCS during a pre-defined time interval. These changes might concern references to waveforms of plasma parameters (plasma current, etc) or even parameters of the controllers. Segments are switched depending on the plasma state, the status of the actuators and diagnostics, or any characterized event that could have been detected. To detect them, each reference waveform considers an envelope around its nominal trajectory (Figure 3.16). If observations from the diagnostics exceed the bounds, the event manager reacts quickly. Depending on the event, the pulse either keeps going with the initial plan or starts an alternative segment like discharge termination. This system permits a flexible definition of what is acceptable regarding reference tracking, allowing more or less deviation from the specific aim of the experiment. Without being its first inspiration, these segments are quite in line with the reference generator developed for the framework. The same idea applies to terminal conditions of the reinforcement learning loop and its connections with the said envelopes.

Just like many devices, most WEST controllers are based on PID control, a proven method that stands for *Proportional*, *Integral*, and *Derivative*. Its terms respectively reduce the error between references and measurements, minimize its integral over time, and act on the system's response through the error derivative. Their usual formulation [8] is described as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau))\, d\tau + K_d \frac{de(t)}{dt}$$

Gain	Purpose	Limits
K_p	Getting closer to the setpoint Limiting residual errors	Too high → Unstable system, noise sensitivity Too low → Poor tracking, high residual error
K_i	Cancel residual error Handles input disturbance	Too high → Unstable system, setpoint overshoot Too low → Low robustness to input disturbances
K_d	Improve system reaction Improve stability	Too high → Unstable system, noise sensitivity Too low → No handling of system instabilities

Table 3.2: Summary of PID gains and their impact on the system's response.

with control $u(t)$, and difference between the observation and the desired reference $e(t)$. As stated in the first chapter, the tuning of WEST controllers is fully realized when we start to look at PID gains, each having a specific impact on the system's response (Table 3.2). Such fine-tuning could be linked to the reward engineering process in RL. While PID gains modify the behavior of the classical controller, an RL-based alternative learns from the reward, in which parameters must be fine-tuned depending on the scenario. This recalls what was stated in the introduction, with a clear shift of interest from fine-tuning how control should perform to what should be achieved. To conclude on WEST magnetic controllers, diagnostics inform on raw measurements used to reconstruct the plasma information, which are sent to different PIDs through feedback. Moreover, control is coupled to feedforward trajectories to define reference poloidal coil currents, which end up dictating the voltages supplied to the poloidal system.

This overall description serves as an additional basis for understanding where an RL-based controller could be deployed and, most importantly, what we intend to replace. Recalling the introduction, the RL-based controller would not need any reconstruction code. More than that, it could handle feedback control and feedforward trajectories as reward components. Ultimately, its deployment would rely on a binary of the behavioral policy, with inputs and outputs already integrated into the control system (measurements, currents, voltages). Such flexibility appears again as proof of RL's theoretical usefulness, comparing the integration of this approach to the actual way of performing plasma control.

3.3 Assembling a distributed architecture

3.3.1 The wonderful story of how C++ met Python

Now that the environment has been carefully described, as well as core definitions of rewards and references, a question remains regarding how the decisions taken by the agent can be communicated to the environment. Both elements are not written in the same programming language. Indeed, the agent and the framework are developed in Python. This is mainly done to benefit from various tools for off-policy RL and optimized deep learning libraries, e.g. TensorFlow [1]. This is different in the case of NICE, which is written in C++. This could be often seen in fusion, as simulations are usually written in such language, in Fortran or Matlab. Because of that, one must look at a simple yet safe communication protocol to let these building bricks interact with each other. The system here transfers predicted voltages to NICE, which processes them until the following magnetic equilibrium and synthetic measurements are sent back to Python. Several options were contemplated, each with its advantages and drawbacks. To choose between these options, the specifications of an RL training loop for plasma control must be in sight. Given the computing timescales of NICE, communication must be fast to optimize training duration without modifying the numerical solver in depth. It must also be reliable to avoid losing valuable information in the NICE data. Sockets fulfill these requirements, providing a proper platform for connecting the agent's decisions to the actual simulation across several viable protocols. From this idea, the full environment retains its last crucial concept. It spans an interface between a Python server and the NICE instance working as a client. Specifically, the said server is instantiated in the environment object. Right after, NICE is launched, and the server waits for the first data. If NICE initialization succeeds, the client responds by concatenating all needed information. This object is then proceeded to compute the reward and check terminal conditions. The agent observes the environment's measurements $o(t)$, coupled with all current targets to predict the next actions. As a side note, the said inputs are normalized, if possible, with respect to deep learning practices.

Thereafter, actions are sent to the NICE client, thanks to the server. This interaction loop goes on until a terminal condition is reached. Hence, the server sends a reset signal to NICE, which restarts from scratch, going again through the initialization procedure (Figure 3.17). One might wonder what happens if the initialization fails or NICE does not converge during an evolution time step. In this case, a reset of the simulation is directly performed, and the server receives a signal stating that a new episode has to start. Such signals management ensures that the whole interface handles exceptions at both Python and C++ levels, or any issue that will compromise data exchange. One must say that we rely on this procedure because NICE's implementation does not display any straightforward utilities for saving the solver's state at any moment during the execution, nor pausing the latter to perform the interaction with the agent. By using sockets, blocking Input/Outputs practically implement an RL loop. A first attempt relied on TCP (*Transmission Control Protocol*), which helped in creating a development routine for further extensions. *Unix domain sockets* (UDS) ends up being a valuable improvement, analyzing how the transfer

is observed at each side of the server-client interface. On average, sending times are under 21 microseconds, compared to TCP's 54 microseconds. Then, the time taken for communication is 2.5 times faster than within the previous procedure. Now, we have an exact definition of which elements compose the implemented environment:

- a launcher starting the NICE client, which contains both the power supply and diagnostics models;
- the server allowing communication with the latter;
- finally, utilities to compute rewards and check terminal conditions for episode reset.

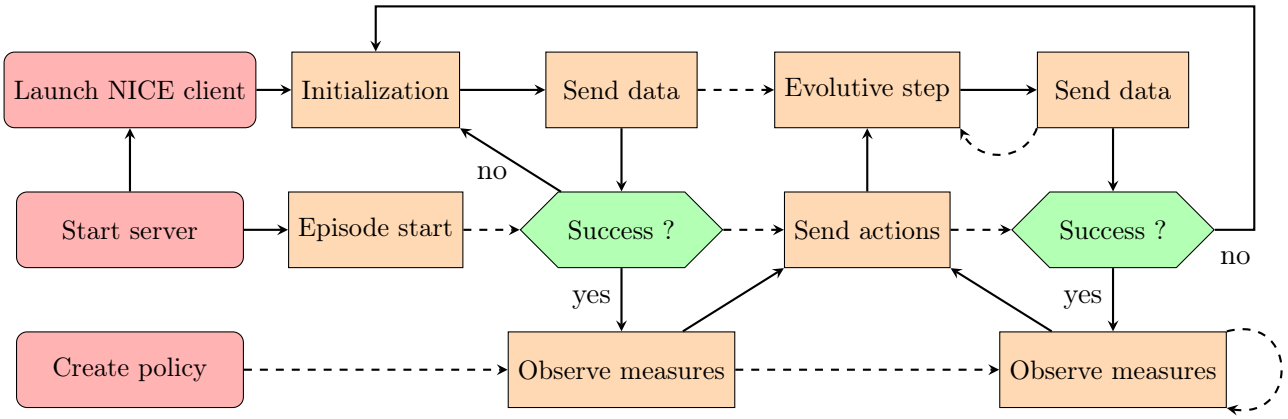


Figure 3.17: Schematic execution of the synchronous interaction between the policy and the NICE environment. After certain operations, the related component waits (dashed arrows) until the response is received. If an exception is raised, we go back to the initialization procedure in the client, and the server starts a new episode.

3.3.2 Nodes galore

The previous section only shows how information are shared in the present multi-language setup. Simply put, the socket approach only enables sending actions to the simulation and processing its data, but it lacks everything related to the training procedure. In Chapter 2, MPO is presented as an actor-critic algorithm, with each element in the expression respectively synonym of policy and action-value functions. Since we work in the context of deep RL, both are neural networks. By linking the policy to the actual environment, we finally get to predict actions necessary to form the interaction loop inherent to reinforcement learning. But we still have to realize the remaining connection with the replay buffer. Most importantly, executing NICE is costly, which makes training of MPO through only one actor intractable for the desired problem. Distributed reinforcement learning comes into play, as described in Chapter 2.

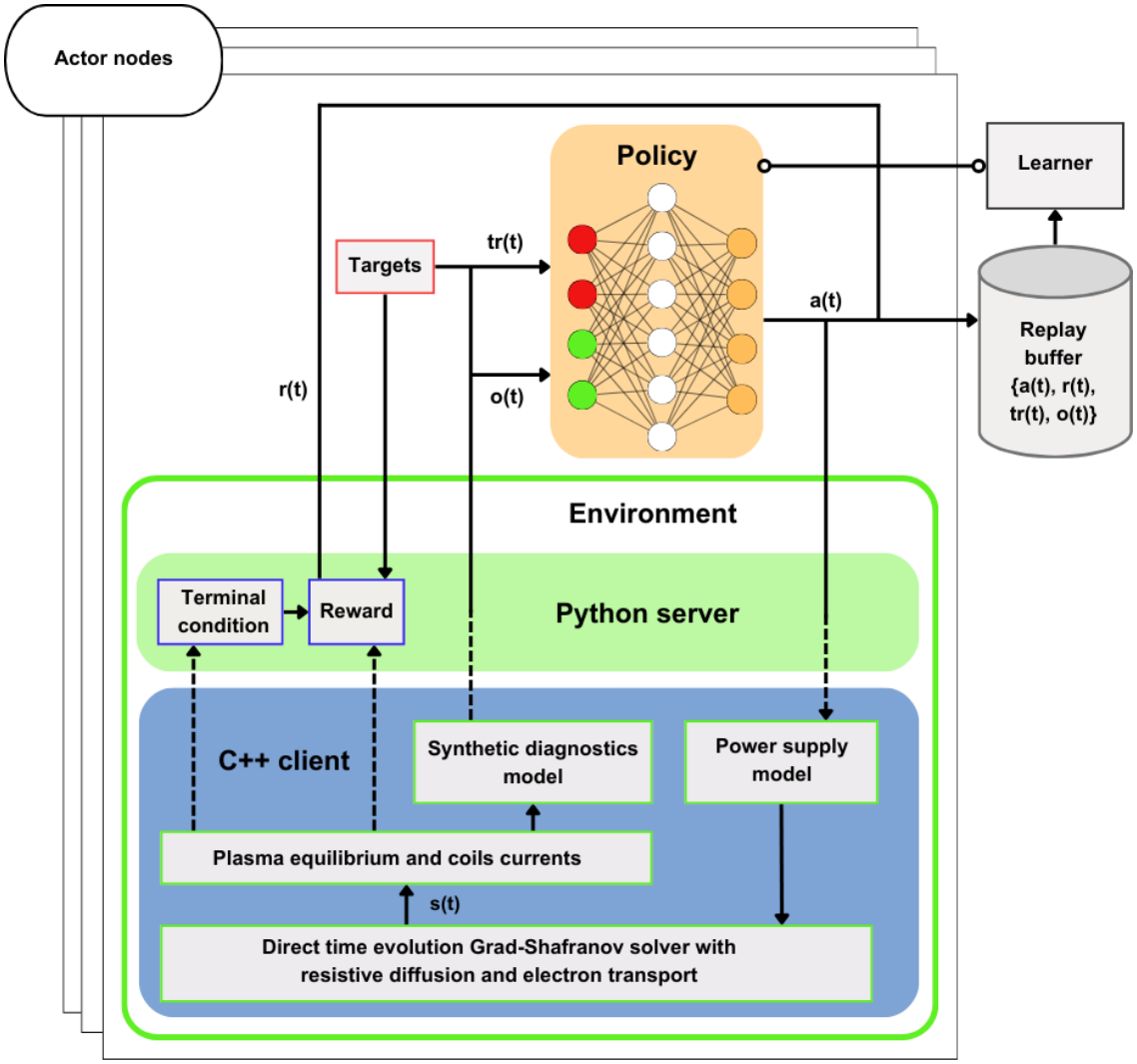


Figure 3.18: The complete framework in the context of distributed actor-critic agents. The environment is made up of a C++ client hosting NICE (blue) and the Python server (light green), enabling the use of observations for MPO. Dashed lines represent data exchange through sockets, with serialization performed at each sending.

Indeed, one can overcome the long duration of each episode by multiplying the number of actors running simultaneously. By doing so, the replay buffer is filled with more diverse information straight from the beginning, helping in shorter training times. Our setup would then require many actors to interact with their respective environments, gathering information for the replay buffer in parallel and without waiting for each other. Not to mention that the replay buffer must be connected to the critic neural network for MPO to perform policy optimization. We could have used sockets again, but the complexity of this undertaking would have increased tremendously. This issue exists because handling multi-

ple handcrafted connections produces bottlenecks and quickly becomes obsolete compared to ready-to-use open-source tools. The *Launchpad* library [134] appears as a powerful choice for this purpose, where gRPC (*Remote Procedure Call*) transactions connect nodes that represent different distributed services in or between networks. It is directly related to the distributed actor-critic layout we want to alleviate, with nodes representing threads for each block in PILOT. Assembling such architecture follows the present specification:

- the first node contains the replay buffer implemented with *Reverb* [21];
- the *learner* node uses information gathered within the replay buffer to optimize policy and critic stored neural networks;
- *actor* nodes work independently from each other, spanning a UDS protocol interface with its own random seed, in which the control policy interacts with a unique instance of NICE. All needed interaction data are sent to the replay buffer asynchronously. Each present node updates its policy by copying weights periodically from the learner.

This results in a fast and reliable, multi-language, multi-threaded framework, running numerous instances of the NICE environment in parallel to learn a control policy (Figure 3.18). As a side note, we can notice that all policy inputs, as well as data passing through the replay buffer, are normalized if possible, namely observations, actions, rewards, and targets. This is done with respect to usual practices in deep learning, as using inputs with similar scales speeds up neural network training.

3.3.3 A glimpse of the agent's distinctive features

For now, nothing was stated regarding neural network architectures used with MPO. Both actor and critic could share the same architecture, like a standard *Multi Layered Perceptron* (MLP), or use an asymmetric design with a *Recurrent* critic. Proved to be more efficient [29, 84], this second option is the one considered throughout the manuscript (Figures 3.19, 3.20). The use of a *Long-Short-Term Memory* (LSTM) [54] requires preparing and characterizing the data format, which will describe every element filling the replay buffer. For this purpose, stored data are defined as sequences of lengths ranging from 1 in case we would require a feedforward critic, to any $n > 0$ for the recurrent approach. By doing that, temporal dynamics are taken into account by the critic and emphasized anew if we recall the previous considerations on temporal derivatives in the observations. The *Sonnet*[6] and *Acme* [55] libraries were intensively used to implement MPO, and specifically to adapt its internal mechanisms to sequences. The latter are partitioned to perform a *burn-in* phase, which takes place at each learner step, i.e. a partition of each input sequence sampled from the replay buffer is used to initialize the LSTM core [62].

It is important to mention that despite MPO being theoretically robust to changes in its hyperparameters (meaning not ones related to the reward definition), a search over them is

[6] https://github.com/google-deepmind/sonnet/tree/v2

mandatory to extract the best out of the algorithm. However, their number is high, which showcases limitations in our computational budget. Even if it was enhanced throughout the project, a threshold must have been set to use the number of computing hours at hand realistically. To a lesser extent, it also argues in favor of reproducibility, considering what the vast majority of research laboratories in fusion might own. First, a few publications [38, 5, 4], as well as large-scale studies [7], inform on a set of viable values which mainly worked for toy problems. Coupled with an in-depth understanding of the MPO algorithm, we can realistically refine the intervals for each hyperparameter. We initially limit our focus on the scenario *limiter maintain* for simplicity. Next, *Bayesian optimization* [127] is employed to restrict the dimensions of the search space. Afterwards, a regular grid search is performed to find correct values for the scenario. Finally, validation is conducted by running trainings using the best set of hyperparameters for all remaining scenarios over two different seeds. One must notice that each actor node increments this seed value to be unique for each environment and neural network initialization. The conclusion follows what makes MPO an interesting choice: once a set of hyperparameters is found for one control scenario, it requires little if no updates when moving to a new one. Please refer to Appendix E for a complete description of each hyperparameter.

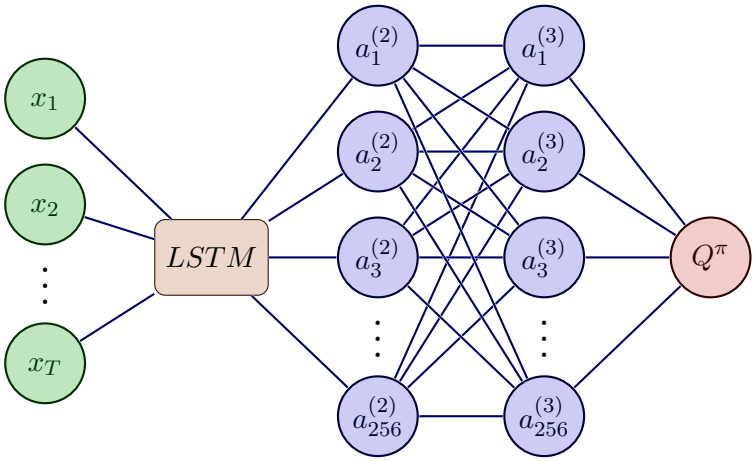


Figure 3.19: An LSTM with 256 units is used in the critic network. Inputs (green) are of dimensions $(T, B, |o(t)| + |a(t)|)$, where T is the length of the sequence and B is the batch size. The next hidden layers (blue) have 256 neurons, each fully connected. The output layer (red) contains only one neuron, which outputs the action-value. A detailed description of its implementation can be found in Appendix E.

Policy networks in actor nodes are all restricted to CPU to lower simulation to reality gaps, preparing the control policy for smooth transfer to the WEST control system. The C++ environments then run on Intel® Cascade Lake® 6248 at 2.50GHz. However, the learner node performs its computations on an NVIDIA® Tesla V100S to benefit from GPU advantages over deep learning architectures. Remote access to the Jean Zay super-

computer[7] gave access to several nodes with similar characteristics while expanding the number of GPUs to 4. The hybridism of this configuration is what makes the optimization of the framework not straightforward. We do not necessarily need a lot of GPUs to obtain correct training or inference times, but enough CPUs so that the NICE instances can run effortlessly one episode after the other. Like so, every aspect of the framework checks that training puts the agent in realistic conditions regarding WEST usual operation. As a side note, the framework is flexible enough to allow the addition of new control scenarios. Moreover, even if the NICE environment is used in the entirety of the thesis, any environment could be added as long as data specification is done rigorously, and functions inherent to the RL loop are implemented (reset, step). Data specification relates to the description of what we want to send and receive across the socket connections.

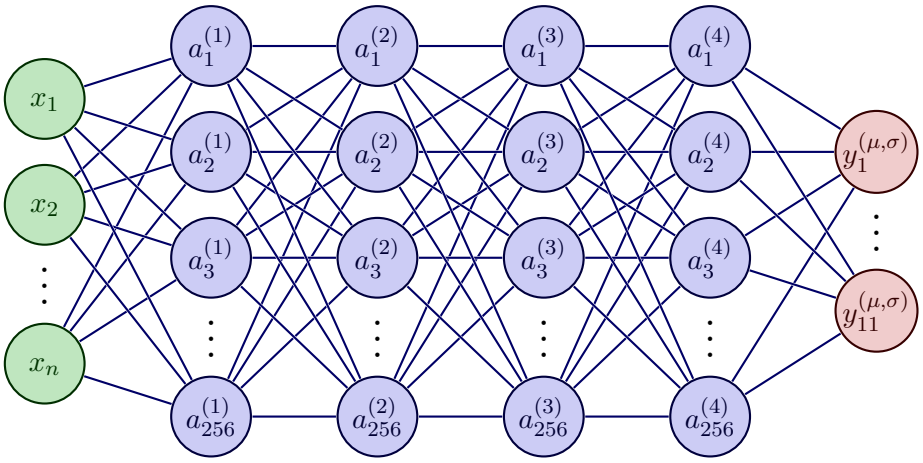


Figure 3.20: Input layer is of size $|o(t)| + |tr(t)|$, and all four hidden layers have 256 neurons. The output layer outputs parameters of a normal distribution for each of the 11 coils. Appendix E provides a detailed description of its implementation.

For predicted actions, voltages are sampled from Gaussian distributions, whose parameters are the outputs of the control policy, i.e. mean and standard deviation for each control coil. Thus, the agent does not directly learn the mapping from raw magnetic measurements to voltages, but an intermediate representation helpful for exploration. We stated that RL is deeply related to the exploitation/exploration dilemma, i.e how the agent exploits acquired knowledge or explores possible outcomes to refine the policy. We saw that a first implemented procedure acts on the environment to randomize initial conditions at best, leading to various observations that describe plasma evolution more efficiently. Another lever is linked to the reward definition and how the related functions can be configured. In particular, one can act on the agent by choosing an advanced exploration mechanism for our use-case. In Chapter 2, we briefly discussed stochastic policies, and how they could learn distributions used to sample actions during training. The agent's structure follows

[7] http://www.idris.fr/jean-zay/

this concept to maximize its exploration of state and action spaces. The sampled voltages are then supplied to each of the 11 PF coils of WEST. After exploring possible outcomes during training, only the mean of each distribution is kept at inference to predict optimal actions. It is worth mentioning that the policy is evaluated during training on a separate node, using only the mean in the same way. This helps in logging enough data for active monitoring. Most importantly, the weights of the policy network in this evaluation node are used to initialize the controller used for post-training analysis. This means all results presented in Chapter 5 are outputted by networks following the same architecture (Figure 3.20).

3.4 A framework ready for training

This chapter presented all the components of the PILOT framework needed to answer the first problematic. We first described which plasma configurations are usually displayed on the WEST tokamak and how this PhD defines scenarios targeting transitions between the limiter and X-point shapes. Then, we explained the NICE magnetic equilibrium simulation, from which we will compute episodes, extract rewards based on target scenarios, and gather data for learning by MPO. The initialization procedure is discussed since NICE was not initially designed to integrate a reinforcement learning interaction loop. Furthermore, a proper digital twin is set so the agent faces a realistic setup during training. Finally, the framework is shown, displaying how its building bricks are written in different languages and communicate through sockets. Nonetheless, NICE computation times are mentioned as a bottleneck. There is an evident need to find methods to reduce training times without thoroughly modifying the framework's core elements, especially the simulation. Indeed, modifying the NICE environment to speed up execution while staying within the scope of this PhD might be too complex. Similarly, changing the agent's structure or the training algorithm could give more insights into how different RL algorithms compare to each other. However, given the important work on hyperparameter choice, such an initiative would be counterproductive. Lower training times would give more arguments in favor of the adoption of RL as a promising alternative to classical control. The next chapter presents simple yet efficient methods to speed up training without an in-depth modification of PILOT.

The need for speed in PILOT

Contents

4.1 Accelerating training through curriculum learning 96

4.1.1 An inspiration from human learning 97

4.1.2 One does not simply generate a curriculum 98

4.1.3 Connections to the state-of-the-art 100

4.1.4 Limitations of the current approach 104

4.2 A structural view against catastrophic forgetting 105

4.3 A procedure ready for benchmark 107

PILOT enables the training of RL-based controllers at the cost of long training times. Each parallel environment runs NICE, a powerful yet costly free-boundary equilibrium solver. This slowness impacts many parts of the framework, including non-exhaustively the pace at which the replay buffer is filled, the ratio at which inserts and samplings are performed within the latter, and consequently, how fast optimization is performed in the learner. This chapter details a procedure based on *Curriculum Learning* (CL), which could speed up training by several orders of magnitude without in-depth modifications of PILOT's building bricks. This approach is one of the first attempts, along with [118], to look for practical means of speeding up training of magnetic controllers trained by reinforcement learning, and answers directly to our second problem. A discussion will follow on the method's shortcomings and approaches to prevent them.

4.1 Accelerating training through curriculum learning

Reinforcement learning can take several training days to reach minimal performance on relatively simple plasma scenarios [29, 63]. As stated previously, PILOT contains multiple elements, each being computationally expensive. Indeed, one NICE instance can take more than 10 minutes to simulate one second of plasma in the best-case scenario, while PILOT's learner node is faster but requires GPU usage to reach its full computing potential. Because of that, the computing timescales of each building brick of our framework vary greatly during training. Nonetheless, the routine operation of WEST and other tokamaks requires flexibility and adaptability in the design of controllers. Minimum engineering efforts should be targeted to train and adapt the controllers with respect to the control objectives of each new experimental campaign. This makes the framework valid to answer the first problematic of the manuscript but insufficient to answer the second. Several levers are considered to overcome this critical bottleneck. The first idea could focus on upgrading the NICE environment to reduce execution times. However, this might be too complex while staying within the scope of this PhD. Similarly, NICE instances could be restricted to their minimal mode, removing transport and resistive diffusion. Nonetheless, by trading a numerically stable simulation for a probable performance drop, this concept would lose out, with only a relative enhancement of computing costs. Another way would be to look over the MPO agent. Despite its theoretical advantages compared to other algorithms, it is still sensitive to changes in the hyperparameter settings. This means modifying the MPO algorithm would occasionally require many searches to find an optimal setup. In the same sense, completely changing the agent for a different one would give more insights regarding the efficiency of other approaches. Still, since the first problematic looks for ways to reproduce the state-of-the-art, this modification is considered counterproductive. On another note, enabling GPU usage does not prevent the expected expense of neural networks training. Because of these considerations, we must look for a different path to obtain magnetic controllers more rapidly, without an in-depth modification of PILOT's building bricks.

4.1.1 An inspiration from human learning

From the initial stages of human development to adulthood, learning is organized sequentially to transmit knowledge over time gradually. This process is structured to facilitate the understanding of new notions or tasks that occur later in life. This logic can be intuitively linked to how we learn to walk before learning to run. Therefore, a sequence of increasingly complex tasks helps in learning a final objective, which would have been difficult to train from scratch. Depending on the period of life, such a strategy is scheduled and designed to support the learner in acquiring transferable skills to guide its exploration during training. It promises increased performance and reduced learning time for a final set of tasks if done properly. This implicitly builds a curriculum, as knowledge must be transferred from one intermediate task to another. *Curriculum Learning* (CL) appears as a methodology to optimize the order in which experiences are sampled for training. Initially applied to supervised learning [13], this paradigm was built upon the hypothesis that learning could benefit from switching between the gradual introduction of increasingly complex pre-processed samples, and the use of specific ones that are neither too hard nor too easy. This handmade task-specific curriculum sped up training and yielded better generalization properties. When applied to RL, CL can be introduced to the training procedure in multiple ways. Recent works developed a classification of existing methods [111]. Most importantly, CL gained a mathematical framework for reinforcement learning domains [82] that we use in the present chapter. In most cases, each task considers its own Markov Decision Process, and three concepts arise with which CL taxonomy can be classified:

- the intermediate task generation, which is done manually or automatically;
- the partial ordering on the previous set, which schedules intermediate tasks;
- the knowledge transfer within the obtained schedule, as what is exactly shared one task to the other.

Indeed, task generation and sequencing can be handcrafted by human operators [112, 75], but both concepts could be built up automatically, either in an offline or an online manner [133, 59, 42]. Finally, *Transfer Learning* (TL) is the approach we refer to when describing how knowledge is shared among instances of the curriculum. This method discusses how the representations learned for one training can be extended to a new domain that can differ more or less from the initial one. For example, one could train a system to recognize pictures of cats and dogs, and perform transfer by augmenting the classification task with bird images. The identifiable primitives in the new class are indeed different, but the knowledge acquired in the initial task is inherently helpful for the second. In the context of RL, knowledge representation concerns several elements of the training loop, such as policies, value functions, rewards, etc [137]. As we will see, care must be taken while choosing the right combination of methods to avoid unwanted effects on the learning dynamics, i.e. finding the right approach avoids negative transfer, which could harm control performance [132]. CL then appears as a potential path to reduce training

times, especially in contexts like fusion, where poor reward and state representations slow down the training process. By focusing on simpler tasks at the start and appropriately transferring knowledge, convergence might be faster and more stable. Since the initial trainings last for more than a week, we look for increased performance at start of each new task, specializing exploration as training moves forward.

Considering the costly data sampling through WEST simulations, let alone on the actual device, curriculum learning could be of great assistance to stabilize the training procedure and reduce convergence time by several orders of magnitude. Each new experimental campaign on WEST requires the definition of multiple control scenarios. The latter might have similar plasma configurations and control objectives. Consequently, the same plasma targets can be used within different scenarios. Since we consider a sequence of transitions between desired plasma shapes, the ordering on the latter already implicitly defines a curriculum. Indeed, magnetic equilibriums must transition from one to another in a realistic and feasible way. One could go further by explicitly generating a curriculum on the reward function, looking for a sequence on its definition. By doing this, we can explicitly create a sequence of increasingly difficult control objectives, which might be similar between scenarios. For example, such curriculum could start with a simple reward on the shape, later including information regarding the X-point, etc. These two notions lead to the same conclusions when it comes to CL in fusion:

- the curriculum can be implicit because of the physical constraints required while transitioning between plasma configuration;
- curriculum generation and task ordering could explicitly describe tasks as sequences of desired events, or intermediate reward definitions. It is easy to realize that the two are equivalent since we want to track trajectories of target events through the reward;
- the two approaches show that a curriculum designed for a specific control scenario could be intuitively generalizable on similar ones, enhancing the production of controllers for multiple plasma discharges between experimental campaigns.

Now that we have described the general principle behind curriculum-based reinforcement learning, we should integrate this framework with ours. This will go through the three mandatory steps used to characterize the procedure of interest: the definition of magnetic control tasks, their ordering, and the concern raised by knowledge transfer and evaluation.

4.1.2 One does not simply generate a curriculum

Formalism

Let $\mathcal{T}$ be a finite set of tasks with corresponding $\mathcal{M}_i : (\mathcal{S}, \mathcal{A}, P_i, \mathcal{R}_i) \in \mathcal{T}, \forall i \in \{1, \cdots, \mathcal{T}\}$, all sharing the same state and action space. Moreover, we denote Θ^T, the set of all transitions belonging to $\mathcal{T}$, so that:

$$\Theta^{\mathcal{T}} = \{(s, a, r, s') \,|\, \exists\, \mathcal{M}_i \in \mathcal{T} \Rightarrow s \in \mathcal{S}, a \in \mathcal{A}, s' \sim P_i(.|s, a), r = \mathcal{R}_i(s, a, s')\}$$

A curriculum $\mathcal{C}$ can then be defined as a direct acyclic graph ($\mathcal{V}$, ε, H, $\mathcal{G}$), with $\mathcal{V}$ vertices, ε edges, $H : \mathcal{V} \rightarrow \mathcal{P}(\Theta^{\mathcal{T}})$, connecting $v \in \mathcal{V}$ to a subset of samples of Θ^{T}. An edge $< v_j, v_k >$ of $\mathcal{C}$ links two tasks, using all samples associated by H to v_j before transferring to v_k. For each $\mathcal{M}_i \in \mathcal{T}$, we have:

$$\Theta_i^{T} = \{(s, a, r, s') \,|\, s \in \mathcal{S}, a \in \mathcal{A}, s' \sim P_i(.|s, a), r = \mathcal{R}_i(s, a, s')\}$$

We need to associate all $v \in \mathcal{V}$ with corresponding $\mathcal{M}_i$ and Θ_i^{T}, meaning that each path on the graph directly influences how $H : \mathcal{V} \rightarrow \{\Theta_i^{T} | \mathcal{M}_i \in \mathcal{T}\}$ filter knowledge transfer between tasks, with edges built on properties of the samples associated with adjacent nodes. Tasks have to be ordered so that $\pi_*^{(i)}$ is useful in acquiring good samples at each transition to the current vertex. It is worth saying that $|\mathcal{T}| = 1$ characterizes many RL methods, and *Experience Replay* is one of them since we look for a way to organize incoming data in the replay buffer. In the current case, we restrict ourselves to a simple setup (Figure 4.1): a task is associated with only one vertex, and each intermediate vertex sinks in only one node until the terminal one is reached, which corresponds to the final task [82].

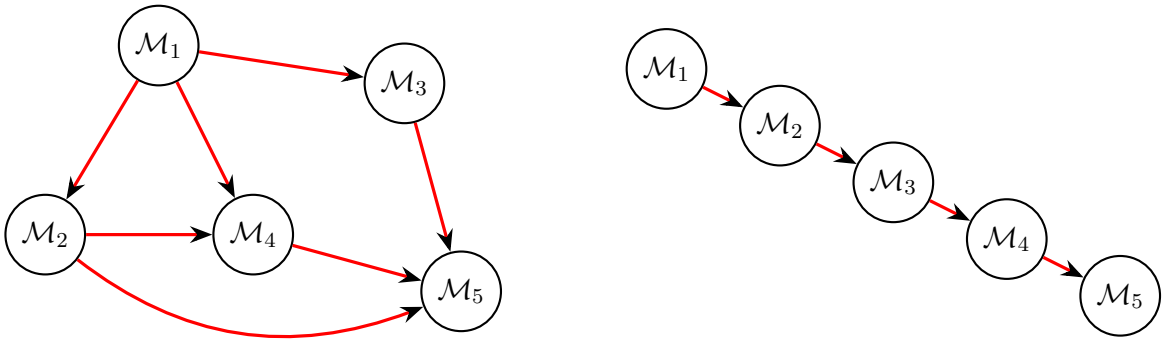


Figure 4.1: In the general case, nodes might sink in multiple neighbors as multiple paths might exist towards the final goal. The restriction used in this work is motivated by its simplicity and by the use of prior knowledge about WEST operations.

This directed path is a straightforward way to link the present formulation with the scenarios studied in this PhD. The implicit curriculum we mentioned earlier exhibits this exact property, as each scenario fixes a plasma configuration for each timestep, with one transition sinking in only one of the others. When it comes to the explicit formulation, it follows the same conclusion and avoids any possibilities for branching, i.e. considering that a successful task might lead to several new ones. This is especially true in the case

of automatic curriculum generation, where an algorithm chooses the ordering online and might opt for different paths according to various metrics [91]. However, the uncertainties around plasma dynamics make the choice for a handcrafted curriculum more suitable. Considering the importance of human intuition to define simple tasks [13], domain experts could efficiently distinguish between objectives that are neither too simple nor too complex. Prior control experience on the device precisely informs which tasks should form the curriculum, and most importantly, which sequencing should be considered easier than others. This work relies entirely on human experts for both determining $\mathcal{T}$, as well as the resulting ordering based on $\mathcal{V}$ and ε, and we will now refer only to the explicit curriculum formulation.

4.1.3 Connections to the state-of-the-art

It is worth noticing that [118] addressed the initial drawbacks of the method from the state-of-the-art we attempt to reproduce. In the initial research, training speed and steady-state performance could also be improved. Their approach resembles curriculum learning by borrowing similar codes. Target scenarios are made of snapshots between which transitions are interpolated. These sequences are partitioned into smaller *chunks* $\{H_i\}_{i>1}$, each related to one part of the general task. For example, let us define a transition from a limiter plasma to an X-point one, built using two intermediate reference snapshots. The entire scenario would then be made of three sections, and the proposed scheme would use each of the latter as a chunk. Let us recall that PILOT uses many NICE environments running in parallel. These distributed environments are divided into subsets of various cardinalities, each linked to one of the said chunks. Reusing our example, we could have 70% of the actor nodes run on the entire scenario, while the remaining 30% would be split equally among the three chunks. Trajectories are accumulated from MDPs that differ in their underlying dynamics, namely in P and $\mathcal{S}$. The samples become more diverse right at the beginning of training and implicitly combine several levels of difficulty inside the same training procedure. This method has already reduced training time by a factor of 4, but how does it exactly differ from our paradigm of interest? Despite multiple initial state distributions, the explicit reward definition remains the same between chunks, and neither curriculum, knowledge transfer, nor task ordering are specifically mentioned between each partition (Figure 4.2). Hence, the chunks procedure builds up parallel MDPs sharing action space and reward function, only letting transition dynamics and state space vary. Oppositely, the curriculum strategy builds a reward hierarchy among the available MDPs sharing action and state spaces, with transition dynamics and reward definition evolving through the procedure. An interesting outcome shows that the two methods are not orthogonal and could be combined easily. This is done simply by:

1. splitting in chunks according to the related approach;
2. for each task in the curriculum, all chunks would refer to the same $\mathcal{R}_i$.

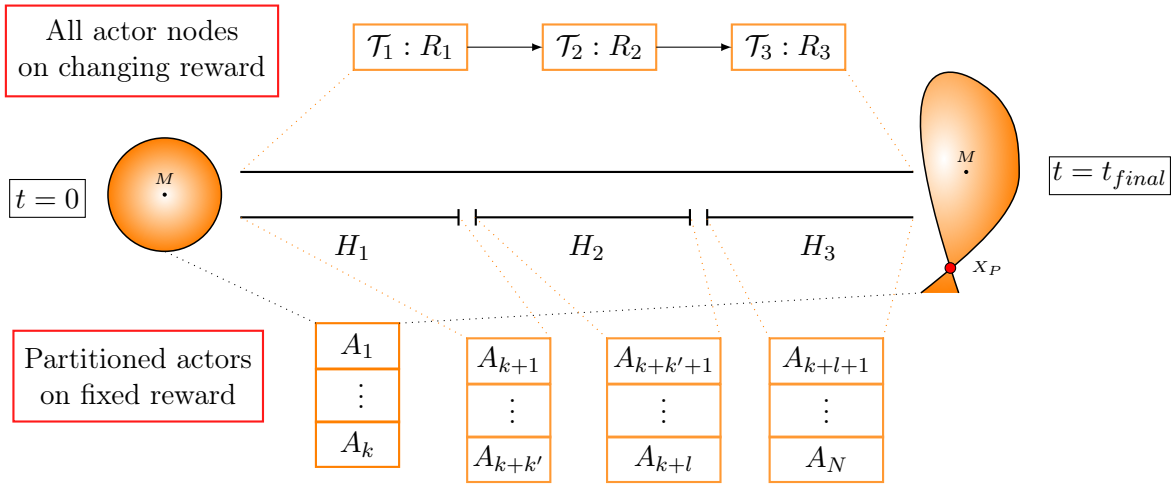


Figure 4.2: In the present setup, curriculum learning (top) uses an evolving sequence of rewards on which all actors are trained. In the *chunks* procedure (bottom), subsets of actors work either on the entire scenario or partitions of the latter, while the reward definition is kept fixed between chunks.

We will use the combined procedure to benefit from both approaches. Since we restricted the curriculum to a specific setup, we must now describe the actual tasks and justify the handcrafted ordering.

Tasks generation and scheduling

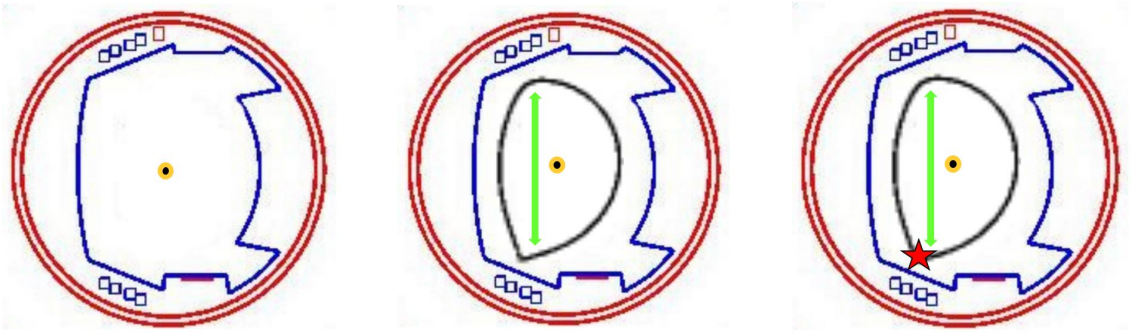
We start from the scenarios defined for the reference generator in Chapter 3 and consider only one of the two possibilities mentioned earlier regarding curriculum definition. The chosen curriculum is entirely conditioned by a set of fixed reward definitions $\mathcal{R}_i$. More precisely, the curriculum has been built from intuition around several key control challenges studied for all tokamaks (Figure 4.3):

1. control of the plasma centroid while tracking plasma current is a well-known challenge. Starting from it, it directly relates to the vertical stabilization problem. In most cases, PID controllers are used to stabilize the plasma's centroid (r_m, z_m) and plasma current I_p. Their relative simplicity is not far from a naive RL-based approach, as an agent can be seen as a proportional-integral controller. The initial reward function then considers targets for the two mentioned components. Handling such a classical problem is a good start to reinforce the foundations of our agent before more complex tasks;

2. tracking the LCFS becomes more difficult, as methods from classical control usually rely on advanced methods to synthesize controllers. Since the difficulty reaches a new step, we add the plasma boundary and the elongation to the desired references. In the limiter case, this would be the final task. In the case of an X-point transition,

this guides the agent towards an elongated shape, correctly positioning it before the final curriculum step;

3. the final objective looks at setting up and maintaining the X-point configuration. We extend the reward by including targets related to the X-point characteristics (distance to the reference location, magnetic flux at the current location if the X-point is observed, etc.). This might be seen as a fine-tuning exploration since the agent must have already learned to position the plasma boundary accordingly in the previous step.



$\mathcal{R}_1 : (r_M, z_M), I_p$ $\mathcal{R}_2 : R_1, \kappa, \text{LCFS}$ $\mathcal{R}_3 : R_2, X_p$ targets

Figure 4.3: We start from a simple vertical control stabilization problem with a free plasma current to a complex one involving shape and X-point.

As a side note, a question arises when we focus on X-point scenarios. In this context, ending the curriculum by extending the reward with various X-point targets is understandable: the new objectives aim to form an X-point, knowing that the agent already learned to position the plasma boundary as well as possible. Let us now consider a situation starting from the X-point configuration and transitioning to a limiter one. In this case, stopping at the second step of the curriculum should be preferable, removing the related targets from the procedure. This idea assumes that the lack of reward components discriminating between configurations does not penalize the agent while going to a limiter shape. Indeed, we will see in the next chapter that when asked to stabilize a plasma without X-point targets or heavy weight put on the elongation, the agent tends to reduce the latter. This is an expected result since limiter plasmas are inherently stable, and the agent exploits this underlying fact [29]. Hence, the latter reduces κ and removes X-point(s) at the same time. All scenarios of interest can then use this curriculum, but how would we perform skills transfer between tasks?

Knowledge transfer and evaluation

At each task, let us specify that target networks are copied from the online networks in the same way as the standard procedure. We transfer the policy and the action-value function

by common transfer learning. The parameters of $Q^{\pi_\theta}_{\lambda_i}$ learned during an intermediate task serve as initialization for the weights of the next action-value function $Q^{\pi_\theta}_{\lambda_j}$, without any freezing procedure which could cause negative knowledge transfer [132]. Doing so bias voluntarily the agent towards more guided exploration in the next MDP. The policy's weights are also retained to initialize the parameters of the new one, again without any freezing procedure. This seems counter-intuitive as transfer learning in a supervised setting is performed as such almost exclusively. One could have incrementally frozen layers between tasks to save learned representations implicitly handled by the neural network. However, we empirically observed that it is not necessary for the curriculum learning to work well in practice. Firstly, it limits the number of tasks in the curriculum, as the number of layers is bounded. Secondly, the learning dynamics in reinforcement learning are noisy enough, so an agent with frozen layers might not have enough expressive power, i.e. number of learnable parameters, to account for strong changes in the state representation one task after the other. On a side note, one could have transferred only one of each component because of the actor-critic structure, but it is observed that transferring both works better in practice [132]. Earlier in this manuscript, we mentioned how exploration is a core concept in reinforcement learning. It happens to be handled by several parameters of MPO, namely the parameters of Gaussian policies and the hard constraints in the policy improvement step. To enhance the overall process, especially as training starts, we could find a way to guide the agent, starting from states that were valuable at the previous task and overlap with the new one. One could keep transitions in the replay buffer, but this idea could be harmful since the information is not guaranteed to be insightful. We could, however, add a bonus to the reward, shaping it with respect to our current intent. Instead of looking for a physically related bonus, we leverage the use of the previous critic through *potential-based advice* reward shaping (PBARS) [83]. In this setup, the rewards sampled for MPO are extended thanks to the formula:

$$\begin{aligned} R'_j(s_t, a_t) =& R_i(s_t, a_t) + F(s_t, a_t) + R_j(s_t, a_t) \\ & \text{with} \quad F(s_t, a_t, s_{t+1}, a_{t+1}) = \gamma[Q^{\pi_\theta}_{\lambda'_i}(s_{t+1}, a_{t+1}) - Q^{\pi_\theta}_{\lambda'_i}(s_t, a_t)] \end{aligned}$$

with γ discount factor set to 0.95 to balance between the different terms. Following our previous statement, R_i relates to the knowledge gained from the source task, and F computes the distances between Q-values of consecutive states from the old target critic. It encourages exploration from valuable states that overlap with the target j. Indeed, starting from states with high Q-values computed from the previous critic will emphasize going into ones that still increase the Q-values, i.e. maximize the related distance. Hence, a valuable state for the previous task overlaps with the new set of proper ones. This idea forms the potential-based bonus with guarantees that it will not change the optimal policy [50]. Even if final performance on the target task matters, our main interest lies in evaluating how curriculum-based RL could rapidly produce magnetic controllers for routine use on WEST. For this reason, we must choose metrics accordingly to measure how CL enhances training times, compared with the vanilla method in which the agent immediately learns the final task. This choice is intrinsically linked to knowledge transfer,

as the latter impacts convergence speed every time a new task is launched. In this PhD, we will use two tools:

- the *jumpstart*, which measures the performance increase at the beginning of each new task, as a result of transfer;
- the *Time To Threshold* (TTT) which checks how fast the overall performance reaches a threshold on the episodic return. This threshold is set for the final task, as intermediate ones will see their duration caped so that they do not exceed 24 hours of execution at worst. Indeed, we do not look necessarily for strong performance on intermediate tasks, only for enough gained knowledge to boost performance after each transfer.

Many other metrics could have been defined [137], but the present tools are well suited to measure convergence speed. By doing so, we do not look at how the reward evolves in a detailed way, but only concentrate on how it starts and ends within each curriculum step and for the entire training duration.

4.1.4 Limitations of the current approach

The procedure at stake is now fully described, with justified task generation and curriculum ordering. The metrics are in line with transfer learning, as the latter is the key to transitioning between tasks and answering the second problematic. It is worth noting that the method displays several limitations, which must be discussed in the context of what we try to achieve.

The curriculum is restricted to a fixed sequence of reward functions, assuming that the latter can be generalizable to many scenarios. However, one should prove formally that this curriculum is at least part of a set of sequences known to produce optimal policies, which might be complex. Moreover, even if the present method gives interesting outcomes, it still requires fine-tuning on each reward component. Let us recall that the second objective of this PhD is to look for ways to make the entire training procedure easier for routine use in tokamaks. However, the relative inflexibility of the curriculum could lead to a lack of adaptability. Thus, ablation studies should be performed to prove that the chosen sequence is useful for any WEST scenario. In the formalism section, it is stated that the action space is kept fixed between tasks. Nevertheless, several parts of the defined scenarios might not require the entirety of the control coils. Extending the curriculum by manually varying the number of available actions considered for each task might be beneficial, to look for decisions in lower dimensions and hypothetically optimize training speed [138].

One phenomenon yet to be mentioned is known as *catastrophic forgetting* [40]. It is encountered with neural networks, which tend to have difficulties learning tasks sequentially. Even if the curriculum procedure is well documented, catastrophic forgetting occurs when the network abruptly forgets previously gained knowledge and starts to perform poorly on new examples. In reinforcement learning, the agent's capacity to handle old objectives

is reduced, and its performance drops sharply after transitioning to a new goal, with no real possible recovery, i.e. no comeback to a better behavior. The root causes of this serious problem might come from many possibilities, ranging from the neural network architecture, to gradient-related hyperparameters (learning rate, etc), through exploration mechanisms. Most importantly, it could be related to how the curriculum has been designed. We must proceed cautiously to avoid this infamous phenomenon since we found several observations in favor of it happening during many of our trainings. Such clues come from a simple analysis of the reward jumpstart without an in-depth exploration of the mentioned potential sources. This discussion would have had many benefits in performing a full ablation study to identify which part of the curriculum could have caused it. Even so, it is kept as a prospective work for future research.

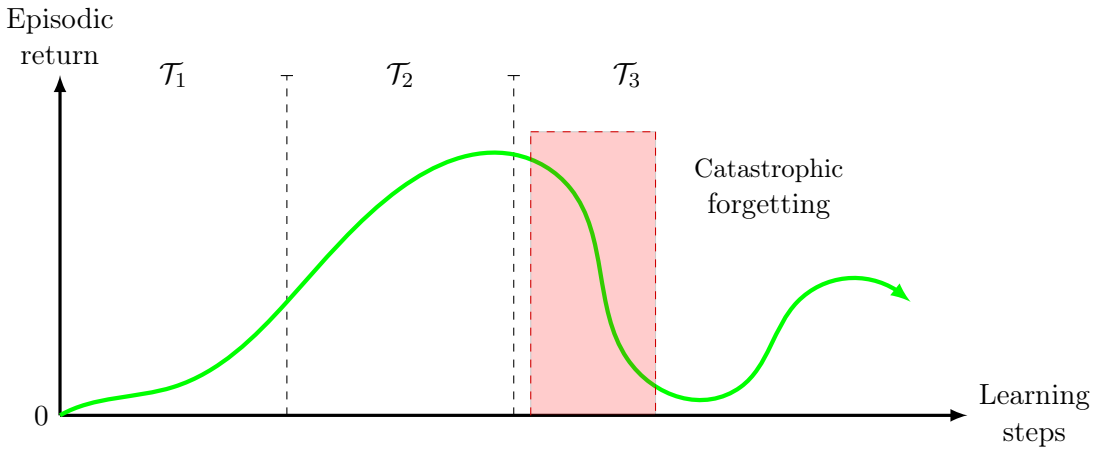


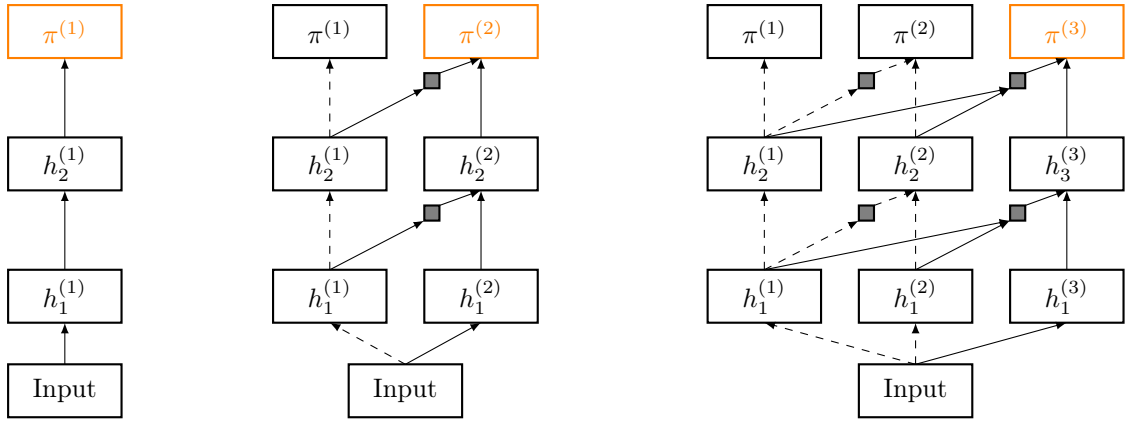
Figure 4.4: Catastrophic forgetting might occur while transferring to a new task, with a drop of performance difficult to recover.

The remaining section presents an attempt to mitigate catastrophic forgetting, thanks to a dedicated architecture. This neural network was recently implemented in PILOT but has not been extensively benchmarked on the set of available scenarios. Hence, the following discussion acts as a short-term enhancement of the present procedure.

4.2 A structural view against catastrophic forgetting

During transfer learning, catastrophic forgetting often acts in the shadows. Because of it, many researches were conducted to identify a common cause or ideas to mitigate it [137]. A straightforward idea emerges from *Progressive neural networks* (PNN) [98], which can counteract catastrophic forgetting with ease. Instead of looking for the right layers to freeze, they leverage the entire set of learned parameters at each transition to a new task. When convergence is observed at each curriculum step, the policy network is frozen and

connected to a new network randomly instantiated (Figure 4.5).



Step 1 : $\mathcal{T}_1$ Step 2 : $\mathcal{T}_2$ Step 3 : $\mathcal{T}_3$

Figure 4.5: At task 1, it is a classical training. For more tasks, the neural network displays several *columns* at the end with lateral connections between hidden layers. Frozen weights (dashed) keep knowledge from previous tasks, and the remaining ones (plain) are optimized. Small gray boxes represent dimensionality reduction. For task $\mathcal{T}_i$, the output of column i gives $\pi^{(i)}(a|s)$.

Let us consider a series of $|\mathcal{T}|$ fully connected neural networks with L layers each. We will denote $h_i^{(k)}$, the activated output of layer $1 \leq i \leq L$ for network $k \geq 1$, with dimension $n_i^{(k)}$ for the number of neurons at layer i of network k. For example, $h_1^{(1)}$ corresponds to the hidden activations from layer 1 of the first network, i.e. the network related to the first task in the curriculum. The forward pass follows the standard NN formula:

$$h_i^{(1)} = f\left(\mathbf{W}_i^{(1)} h_{i-1}^{(1)}\right)$$

with activation function f, $\mathbf{W}_i^{(1)}$ the weight matrix from layer $i-1$ to layer i of the first network with resulting dimension $n_i^{(k)} \times n_{i-1}^{(k)}$, and $h_{i-1}^{(1)}$ the activations from its previous layer. As a precision, h_0 refers to the input layer. After training, the first network is completely frozen and is connected to a second network randomly initialized, which will be optimized. The forward pass becomes:

$$h_i^{(2)} = f\left(\mathbf{W}_i^{(2)} h_{i-1}^{(2)} + \mathbf{U}_i^{(2:1)} h_{i-1}^{(1)}\right)$$

with $\mathbf{U}_i^{(2:1)}$ lateral connections from layer $i-1$ of the first network to layer i of the second, with dimension $n_i^{(2)} \times n_{i-1}^{(1)}$. This is repeated for K tasks, and the forward pass generalizes to:

$$h_i^{(k)} = f\left(\mathbf{W}_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} \mathbf{U}_i^{(k:j)} h_{i-1}^{(j)}\right)$$

By keeping previous networks intact throughout the curriculum, it is structurally impossible for catastrophic forgetting to happen, as all features learned from previous tasks are kept intact. Reusing RL notation convention, we end up with $\pi^{(k)}(a|s) = h_L^{(k)}$. It can be extended to the critic, but we consider only a progressive policy for the sake of simplicity. In practice, the lateral connections implemented with the $\mathbf{U}$ matrices are modified to integrate a one fully connected layer neural network. It performs dimensionality reduction to avoid an actual explosion in the number of parameters. Indeed, lateral connections consider all previous networks, not just the latest one. This leads to a final modification of the forward pass:

$$h_i^{(k)} = f\left(\mathbf{W}_i^{(k)} h_{i-1}^{(k)} + \mathbf{U}_i^{(k)} g\left(\mathbf{V}_i^{(k:1)} \alpha_i^{(k:1)} h_{i-1}^{(k:1)}\right)\right)$$

One can see that the summation was replaced by a matrix operation using the activation function g. Moreover, $h_{i-1}^{(k:1)}$ is the concatenation of all $h_{i-1}^{(l)}$, $\forall l < k$. It is important to notice that $\alpha_{i-1}^{(k:1)}$ is a learnable scalar that scales the content of $h_{i-1}^{(k:1)}$, i.e. the scalar is repeated in a vector to respect matrix operations. The one hidden layer NN is represented by $\mathbf{V}_i^{(k:1)}$ of dimensions $n_{i-1}^{(k)} \times (n_{i-1}^{(1)} + \cdots + n_{i-1}^{(k-1)})$, confirming the stated reduction of dimensions. The notation for $\mathbf{U}_i^{(k)}$ has changed to represent the remaining lateral connections from the output of the corresponding one layer NN to layer i of network k, with dimension $n_i^{(k)} \times n_{i-1}^{(k)}$.

The PNNs are then capable of sequentially learning multiple tasks, structurally protected from catastrophic forgetting. Despite this simple scheme based on traditional MLPs, the parameter complexity increases as the number of tasks grows. As it is, this does not impact training but poses a serious issue regarding magnetic control applications. Indeed, deploying the policy on the actual device requires fast inference, which might not be achieved if the network becomes too big. A promising approach resolves around *distilling* the knowledge of this PNN expert into a smaller network. By gathering interactions between a control policy parameterized by the strong PNN controller and the NICE environment, we could train a student policy in a supervised and offline manner without too much loss of accuracy. This student would be parameterized by a smaller neural network similar to the one currently used in PILOT, fast enough to respect real-time constraints. This forms a promising path, which could also get extended to multi-scenario control with many experts on different scenarios, gathering data for a student policy learning to follow multiple trajectories.

4.3 A procedure ready for benchmark

This chapter described a curriculum strategy to reduce training time and increase performance. Because of the training timescales, this approach could be an essential milestone

to answer the second problematic of this PhD, and put RL under the spotlight within the routine operation of WEST. The chosen curriculum is defined as a sequence of fixed rewards, with tasks generation and ordering advised by prior control experience on the device, without relying on an automatic curriculum generation. A parallel has been made between the strategy developed in this manuscript and the literature, which differs from the paradigm of interest. Combining the two is straightforward, and the results will be discussed in the next chapter. Transfer learning between each task is done through both the policy and the action-value networks without any freezing procedure. This setup might seem counter-intuitive compared to supervised learning, but we empirically see that it works without needing an incremental freeze of layers. However, it might undergo catastrophic forgetting, a serious problem related to neural network training. A hyperparameter search and a study of the reward definition under the latter mitigate it. A more complete analysis is part of the prospective works to assess the viability and generality of the method, as well as the potential sources responsible for the phenomenon. One promising idea comes from *Progressive Neural Networks*, which are structurally impervious to catastrophic forgetting at the cost of increasing model complexity, i.e. a high number of learnable parameters. Even if their implementation has been recently added to PILOT, future works must rigorously benchmark them against the classical transfer approach. Finally, we briefly mentioned *Policy Distillation* as a way to distill knowledge from an expert progressive policy into a smaller network in line with the real-time constraints of the WEST plasma control system. This manuscript now describes a complete environment for RL-based control, an agent ready for training, and a curriculum procedure capable of reducing convergence time. The next chapter will present the results following MPO's trainings, and the application of curriculum learning.

Performance of RL-based magnetic control

Contents

5.1 Evaluation on the scenarios of interest 110
5.1.1 Plasma centroid, elongation and minor radius 110
5.1.2 Careful calibration of the reward hyperparameters 115
5.2 Issues regarding the LCFS and the plasma current 117
5.2.1 Myopic exploration 117
5.2.2 An issue regarding plasma current 119
5.3 The Good, the Bad and the Ugly of Curriculum learning . . 123
5.4 A paradigm under careful calibration 126

In this chapter, we present the performance and robustness of RL-based control on the four scenarios currently used in PILOT. The path towards the best performance obtained so far will be discussed from a qualitative and quantitative point of view, using an example of obtained control trajectories for each scenario. A discussion will follow on which reward threshold could be sufficient to achieve efficient magnetic control. This question relates to the second problematic of this study and is directly linked to the curriculum formulation. All evaluated trajectories are performed on episodes lasting for 300 timesteps. One must note that the transport equation is not part of all the experiments, as it was not yet added to the PILOT framework at the time. In this case, we rely only on the NICE resistive diffusion mode. As stated previously, the simulation could not last without observing unknown crashes due to the solver's non-convergence, even if the initialization procedure stabilized the overall scheme significantly. Once transport was enabled, we tested X-point scenarios for 400 timesteps, as crashes would not occur as quickly and often. A few examples perform control over 500ms to account for longer plasma discharges. It is worth noticing that we display results over plasma current, position, elongation, and boundary, as well as on its minor radius. While not part of the objectives stated in the introduction, we include the latter to complete our views on this evaluation. All targets are displayed in dashed blue lines, and observations in orange, except if stated differently. Throughout the chapter, one can refer to Appendix B for a complete description of the reward components for all scenarios.

5.1 Evaluation on the scenarios of interest

In this section, performance relates to tracking the reference plasma's boundary (LCFS), its elongation, minor radius, and plasma current, as generated by PILOT's reference generator. Results are analyzed for each scenario, always looking at the steady-state tracking error throughout the discharge. In some cases, a discussion is made on how fast the agent bridges the gap between the observations and the reference value. This is helpful to see how far the agent deviated from the sequence of snapshots given by the reference generator. One must also recall that the control networks used in this section are Multi-Layered Perceptrons, whose architecture can be found in Figure 3.20. More details about their implementation can be found in Appendix E.

5.1.1 Plasma centroid, elongation and minor radius

Limiter maintain The present scenario is simple: we only seek to maintain the location and properties of a fixed limited plasma. This is a baseline experiment that classical PID control can perform without much effort. The results are easy to analyze, as the plasma should stay close to the fixed targets. This is indeed the case, as radial and vertical coordinates of the plasma centroid (r_m, z_m) do not diverge from the initial configuration (Figure 5.1). Similarly, the minor radius stays within range without any substantial evolution far from the reference. We see that the elongation gets stabilized but tends towards 1. The

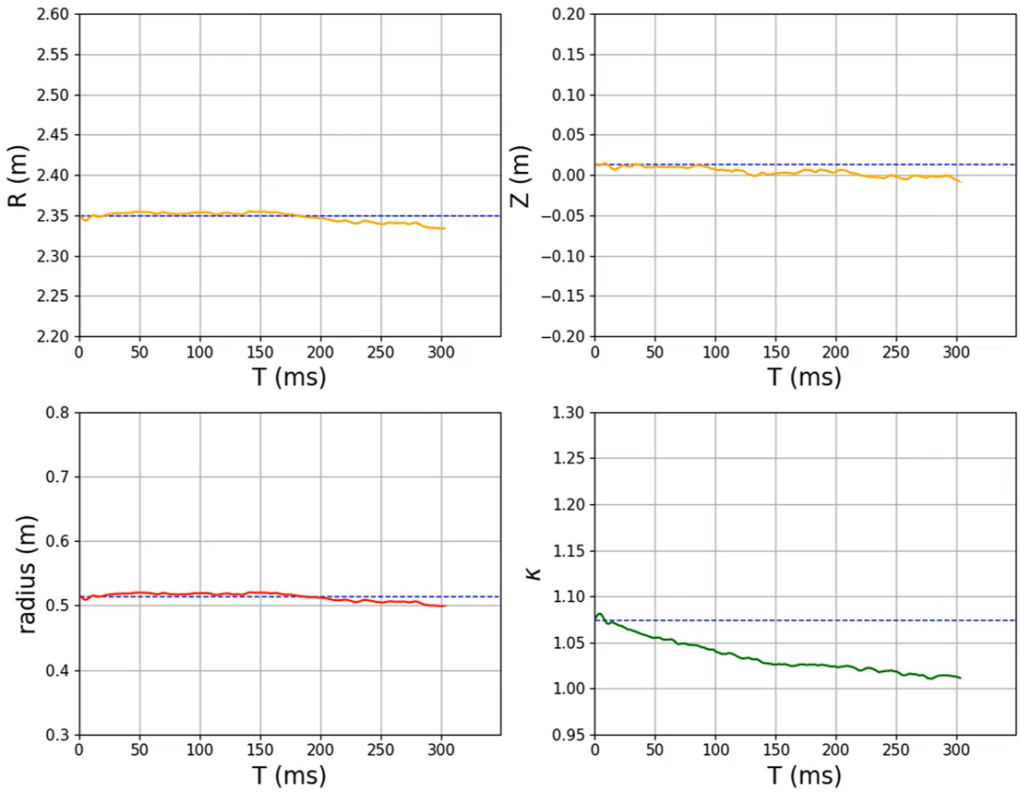


Figure 5.1: The controller stabilizes the plasma close to the targets. The elongation strays from its reference, which might be explained by the agent's tendency to exploit trajectories with lower κ because of a reward component minimizing the coil currents.

starting plasma has low elongation, and the control seems to reduce it. The difference between the target and the observation is outside the bounds of the good and bad parameters for the Softplus formulation, indicating that the corresponding signal saturates towards a low reward value. One potential explanation comes from the implicit connection shared with a reward component on the poloidal field coils. The latter aims to minimize their current, and one way to perform it would be to reduce the elongation. We know that the closest κ is from 1, the more vertically stable the plasma becomes. From this idea, the controller found a way of exploiting the underlying dynamics of the plasma to stabilize it, undermining κ for the benefit of other components. This is an equivalent result of the one presented in [29], where a simple reward only meant to stabilize the plasma results in a controller trying to reduce the elongation. Through this example, we found a way to reproduce a small part of TCV's experiments.

Limiter evolve Let us recall that this second scenario looks for the displacement of the plasma magnetic center and boundary, slightly varying elongation while keeping all other information constant. We do not display the minor radius, as it shows the same behavior

visible in the limiter maintain task. This scenario exhibits an evolution of the references to evaluate how the controller might track moving targets. Despite minor steady-state errors on the centroid position, the control is performed swiftly and rapidly (Figure 5.2). Within the reward, the components on the radial and vertical locations of the plasma consider a wide interval between good and bad parameters. Choosing stricter intervals could have been more effective in reducing the steady-state errors, but despite several tests, a bias is still present. This could be explained by the lack of integral control over the said coordinates, as we will see later in this chapter. For elongation, it is carried out without producing an excessive stabilization bias. It is worth mentioning that the scenario was first defined transitioning slowly to the final plasma configuration. Indeed, the time intervals specified in the reference generator were wider than showcased here. This update is conceived to check how fast the response of an RL-based controller could become, hence the fast ramp towards the desired references. Also, it validates the fact that the agent can stay relatively close to the reference trajectory, even with a relatively permissive reward definition. However, this idea should not be applied to the X-point scenarios, mainly to respect the transition times performed on WEST while dealing with such configurations. Going under this threshold might give unexpected or unrealistic results, which could harm a potential transfer of the policy to the real domain.

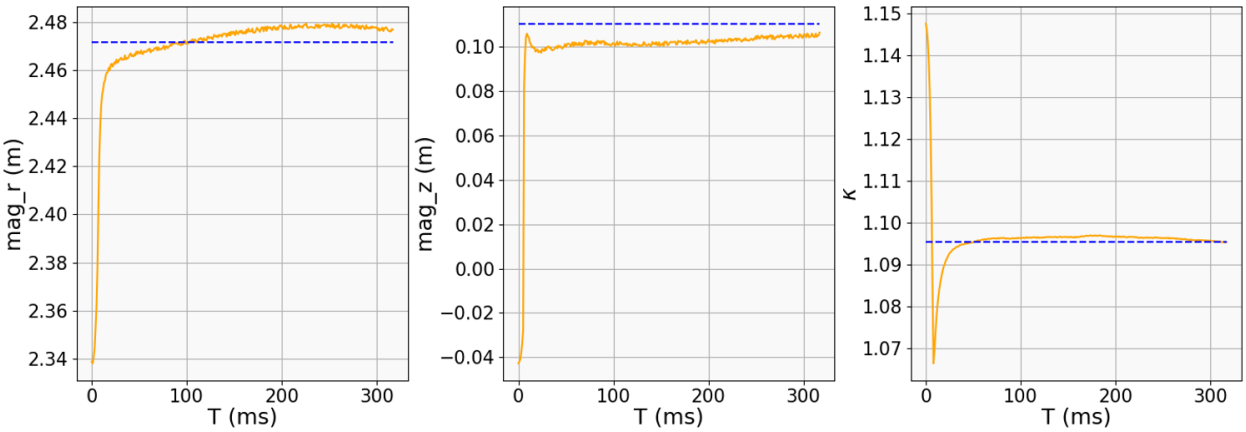


Figure 5.2: Control is performed rather precisely on moving targets, with others kept constant, like in the limiter maintain scenario. The transition duration is chosen to verify RL's ability to perform faster response. Only the final setpoint is displayed for each controlled variable.

X-point maintain This scenario is designed to observe the performance of RL-based control facing an unstable elongated plasma in an X-point configuration kept constant over time. Numerous difficulties appear, as staying close to the references involves controlling an unstable vertical position that can move abruptly. Moreover, the X-point information creates new objectives along with existing ones (X-point location, etc). Again, the observed

quantities do not drift far from the targets, and fine-tuning would only be performed to reduce the steady-state errors on each component, depending on our needs (Figure 5.3). The minor radius and κ logically follow what is observed for the limiter case since they are not subjected to strong instabilities. Despite this efficient control, one must notice that the trajectories seem to end at way less than the expected 300 timesteps. This means that the episode encountered an issue related to NICE convergence or the plasma's state. The second option becomes more probable since NICE convergence was checked and ensured. Indeed, we will see in the following sections that RL-based controllers initially have difficulties controlling the boundary location and the plasma current. This issue is even more important when X-point configurations are at stake. The LCFS is implicitly related to the centroid, the elongation, and the minor radius, which are all stable. We can then discard the boundary from the set of potential threats. The X-point maintain scenario has been appropriately handled throughout many attempts regarding the plasma's centroid, LCFS, minor radius, and elongation. Still, the short episode duration foresees an issue with the plasma current.

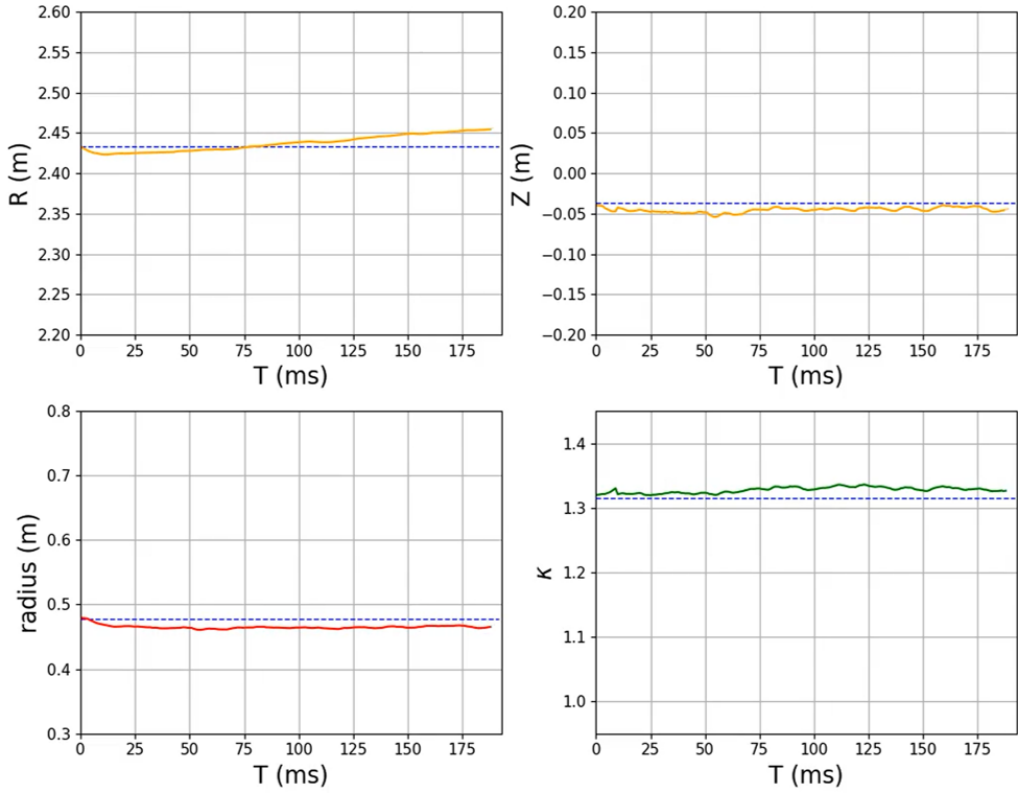


Figure 5.3: Control of the quantities is performed well again, but the short episode duration presages an unstable behavior of the plasma current.

X-point evolve This final scenario studies a classical transition between a limiter configuration and an X-point one. It serves as a final proof of concept because of the unstable moving targets. Here, all stated quantities vary over time, which might interfere with each other. The agent must find the right balance between them, knowing that the reward definition voluntarily focuses on exploration at the cost of precise control. The example trajectories align with this idea (Figure 5.4), and controlled quantities stay within range of the references, considering the good and bad parameters in each of their Softplus transformations. The exception comes from the minor radius at almost 10cm from the setpoint. Its weighting has been set to a low value with a wide Softplus, which might explain why the component stabilizes far from its references. On the contrary, it is important to notice that more weight is put on the vertical centroid coordinate than the other components, which is verified by looking at how the z trajectory is sensibly closer to its targets.

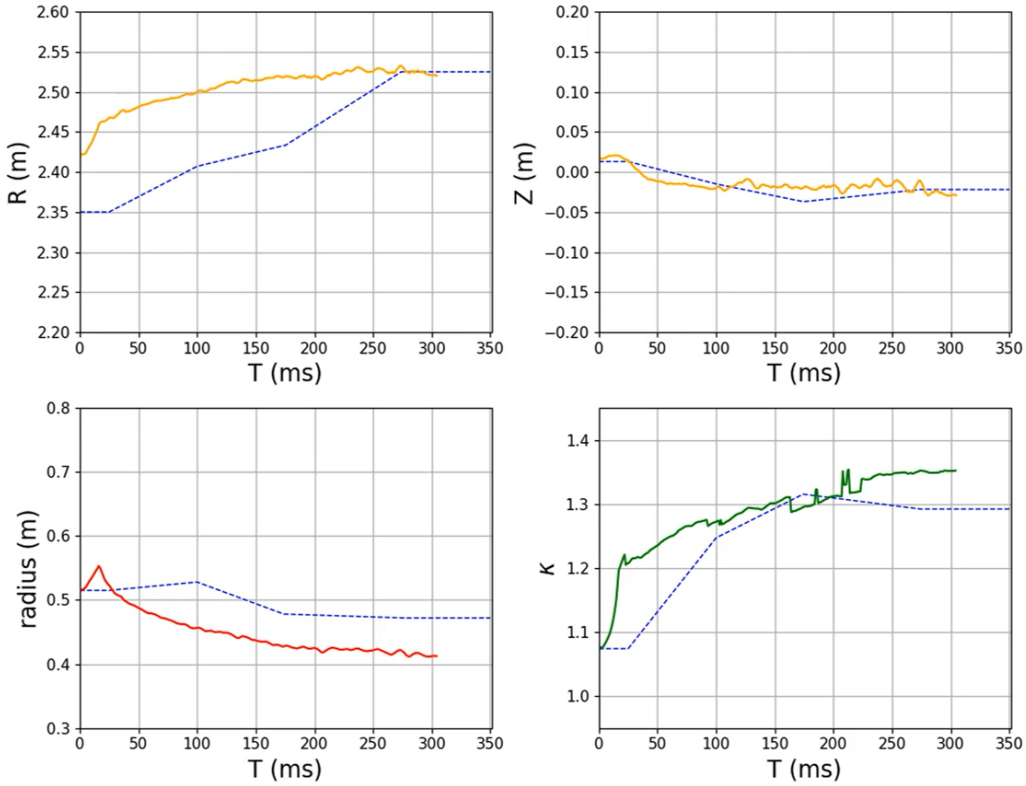


Figure 5.4: The unstable moving targets are handled well, with the relaxed reward definition allowing more gap between the observations and the references.

Furthermore, we can see that the initial radial location is far from its initial reference. This is a voluntary attempt to evaluate how RL can perform if strong disturbances, like changes in the initial state distribution, impact the behavioral policy during inference. In real conditions, this could happen after an exceptional update of the initial plasma

location between two plasma discharges. The initial displacement is not too high, so learning performs relatively better than expected. If we increase the distance in question, control can not be maintained anymore, and the radial position will move sub-optimally. These relaxed constraints over several objectives serve as an example to highlight the careful calibration of the reward function, especially in the context of moving references. One must proceed cautiously to avoid situations where one component would take the lead, leaving others poorly controlled.

5.1.2 Careful calibration of the reward hyperparameters

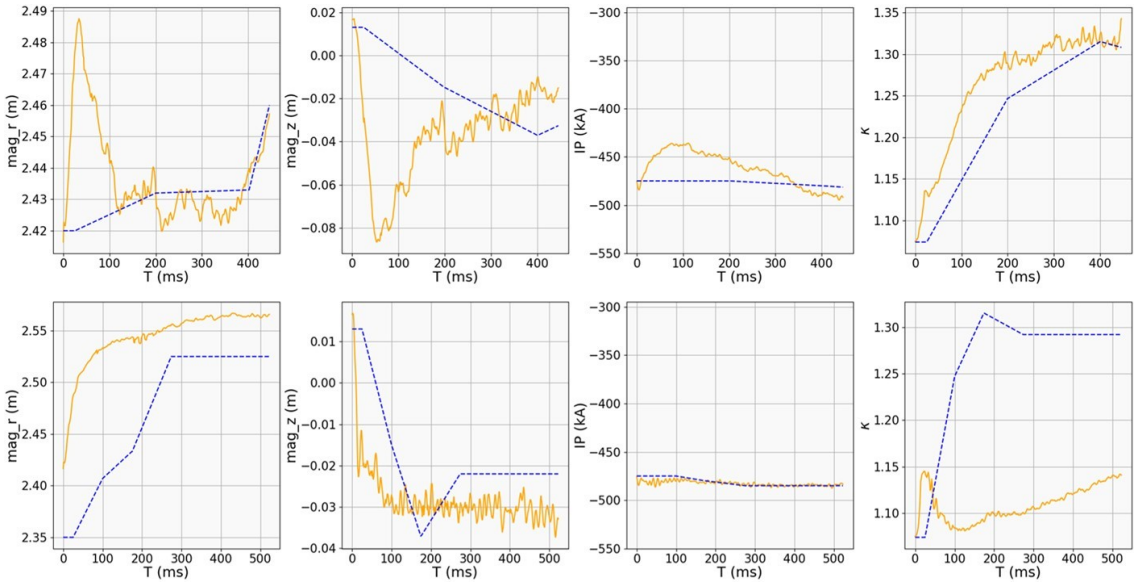


Figure 5.5: In the upper example, twice more weight is put on the elongation. As a result, the control of the other components tends to drift quickly from their targets, not always returning to the desired trajectory. In the lower example, three times more weight is put on the plasma current and z components, resulting in low efficiency in the remaining control objectives.

There are many ways of describing the reward, such that the policy displays good behavior. However, it is difficult to define in the first place and costly to fine-tune. For this PhD, the simulation was enhanced with resistive diffusion and transport, enabling the agent to use more realistic data. Thanks to this, we could start to train on more extended scenarios, kept at 400ms. We observed that the stated problems were even more present throughout these longer episodes, even when only 100ms are added to the overall scenarios. Without doubt, finding the right balance between the reward components is complex. Apart from several heuristics from physics, it is an art on its own that requires careful exploration of the reward landscape. In the following example (Figure 5.6), the component on the LCFS has much more weight than the two others. Looking at the

immediate bad performance on the elongation, the divergence from the target boundary overshadows the learning dynamics, and even the radius term decreases over time.

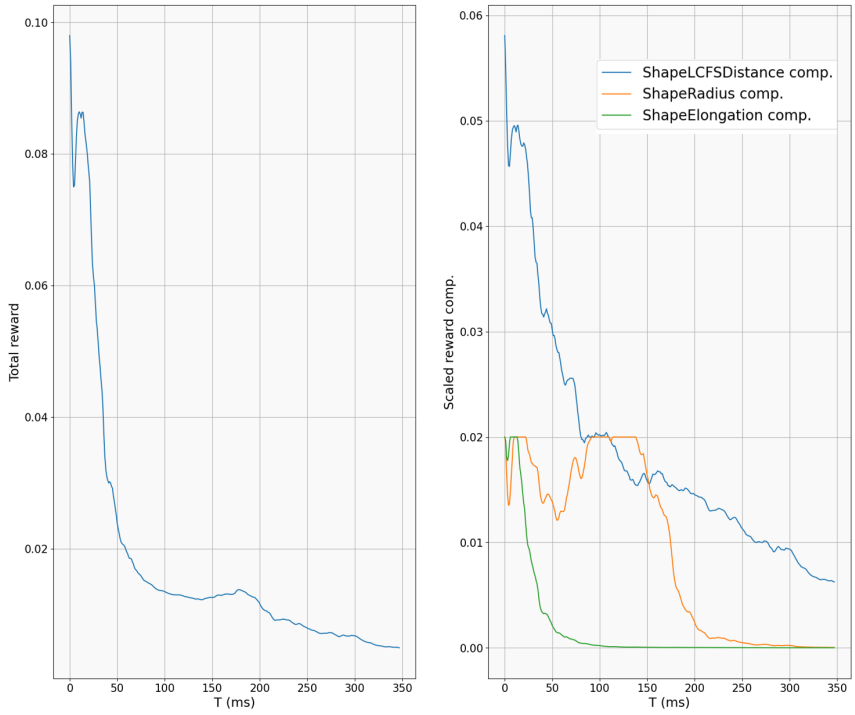


Figure 5.6: The reward includes many terms, which are combined in a non-linear manner. They can interfere with each other, with the underlying dynamics of the function not entirely known.

One must be careful while analyzing the entire reward dynamics in such a way. The latter can be misguiding, as the number of information can quickly push towards over-interpretation. The performance drops significantly in many situations because of reward misspecifications (Figure 5.5). Indeed, if the reward specifies too much weight on one component, it could achieve strong performance, but only for the element of interest. The others, such as the plasma centroid or the elongation, are then impossible to handle correctly, with poor performing policies. One of the main problems of the RL-based magnetic control approach lies in the multiplicity of reward components set to guide the controller efficiently. The usual purpose of RL would be to find a generalized policy which could work on many scenarios. Considering the uncertainties of plasma dynamics and the state-of-the-art in RL, nothing guarantees that complete generalization is practically feasible, and we have to resort to domain knowledge to design the reward, overspecializing it within a subset of potential behaviors. This idea is reinforced in the case of longer scenarios when we increased the number of timesteps to 400, or 500 once NICE was augmented with resistive diffusion and transport. Indeed, a wrong reward specification impacts the behavior in

the entirety of the scenario without any possibility of recovery. As a side note, the current trajectories exhibited by the RL-based controllers present an inversion (Figure 5.7) in the E and F coils (Figure 3.2). This observation seems specific to WEST, as the naming convention of the coils is only related to the device of interest. Interpreting this outcome is not straightforward, as it is opposed to what is generally performed using PID control. One short-term perspective is linked to using these currents as feedforward trajectories for the classical controllers used in actual experiments. This will help gain insights into what the RL-based solution exploited to perform the X-point transition, but not only. Indeed, we observed this phenomenon for all four control scenarios in this manuscript. The analysis of such behavior is a first step towards a better understanding of the learned behavior, to minimize the effort needed for deployment of the full control policy on the machine.

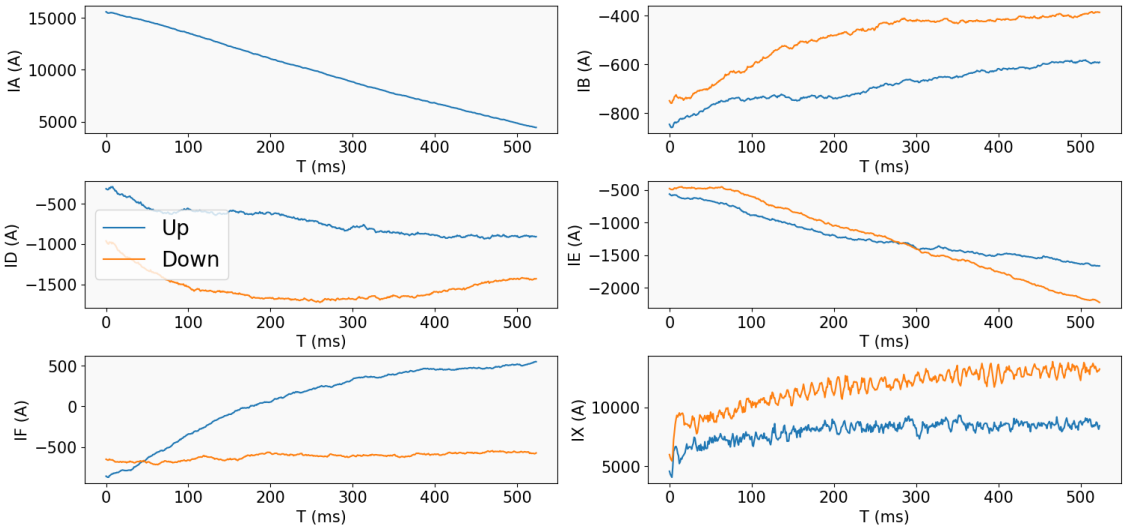


Figure 5.7: Through its actions, the agent tends to influence the evolution of the poloidal field coils currents. In the context of RL, the obtained trajectories differ from classical ones and could help optimize the landscape of configurations through the feedforward control system of WEST.

5.2 Issues regarding the LCFS and the plasma current

5.2.1 Myopic exploration

Until now, we have only referred to plasma's magnetic center and general quantities related to its geometry. Including the LCFS and the X-point would allow a complete analysis of RL-based magnetic control. In limiter scenarios, the LCFS stays stable without too much trouble. This was also the case of X-point maintain scenarios (Figure 5.8), where the LCFS tracks its reference well. However, one can realize that the plasma boundary is in

the middle of the vacuum chamber, and the X-point is close to the lower divertor. In fact, many experiments on WEST locate the X-point slightly higher, and the LCFS closer to the lower field side of the torus, i.e. the outer edge of the vessel, with lower magnetic field than in the inner side of the torus. While trying to master this final configuration, the X-point evolve scenario starts to pose more challenges. Let us look at the evolution of the LCFS and the X-point location under this realistic case. We clearly see that the boundary does not match its targets entirely, with a subset of reference points not tracked correctly. It becomes even worse in the case of the X-point, as the plasma goes back in limiter configuration periodically. The root mean squared error over all the points of interest decreases when considering the final target shape (Table 5.1), but remains higher than what was seen in TCV's experiments. In the WEST use-case, the weight placed on the LCFS reward component is greater than those placed on the other reward terms, but the agent still fails to match the boundary completely (Figure 5.9).

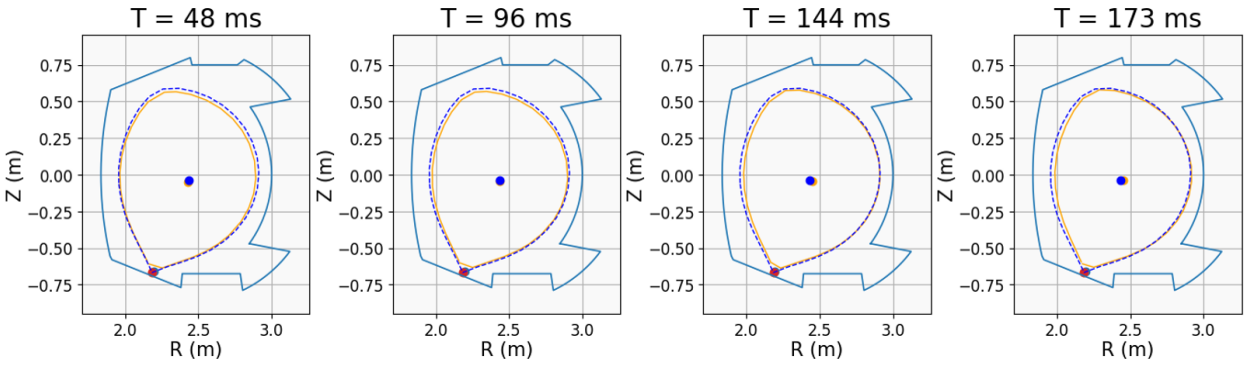


Figure 5.8: While maintaining an X-point configuration, the location of the desired X-point and LCFS is crucial on WEST. The observed (orange) and desired (dashed blue) elements match almost exactly, with the X-point (red) exactly on its target location.

After several hyperparameter searches and observations on the behavior of the obtained policies, we identified two major outcomes related to the present issue (Figure 5.10). In the first case, the plasma follows the X-point location carefully but does not match the LCFS on the entirety of the lower field side of the torus. The significant mismatch increases the mean squared error over the LCFS, even if the X-point configuration is maintained somehow. In the second case, the agent sends the plasma to match the LCFS on the side in question, but it turns into a limiter configuration, as the X-point can not be formed anymore. This myopic behavior can be explained by how the agent explores the possibilities but struggles to stabilize the plasma in the area of interest: it is easier to exploit current knowledge than explore possibilities in this subset of the state space. It is even more interesting to compare it to the X-point maintain scenario where the target LCFS is far from the outer side of the torus and easier to keep at the center of the vacuum chamber. No proper general solution is found now, except after careful fine-tuning of the reward to end up in the first case. The mismatch with the reference boundary is still

present but mitigated as much as possible.

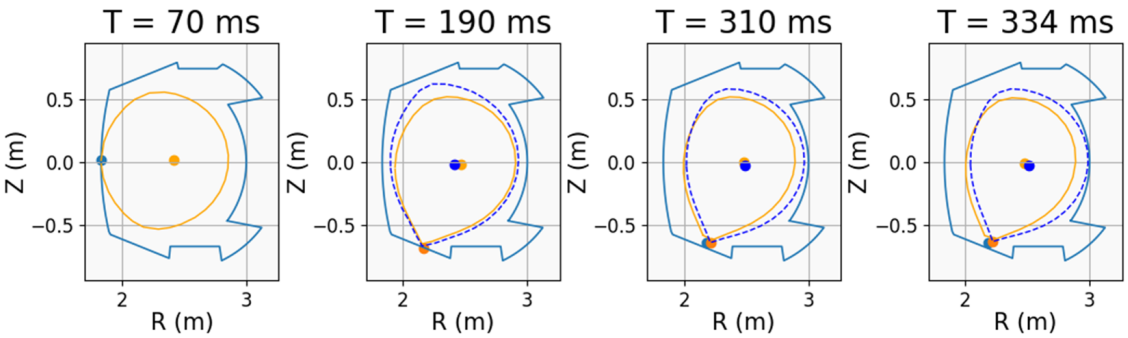


Figure 5.9: While performing an X-point transition with observations (blue) and targets (orange), the configuration oscillates between leaning on the wall materials in a limiter fashion, or put the x-point quite far from its desired location.

An important observation comes from the need for *discriminant* features. Reward components should help differentiate between states appropriate for each configuration of interest. While transitioning from a limiter plasma to an X-point configuration, there should be a way to clearly see the difference between states involving an X-point and those not. The distance to the X-point itself does not suffice, as the exploration process does not find strong intensities to reach the desired target from the moment it is first encountered. Adding a component on the *limit point* is then mandatory: it is set as the interaction point with the material walls in the limiter configuration or the actual X-point in the aforementioned situation. During training, the reward is boosted as a bonus appears in the X-point-related states. Future works will focus on finding more efficient ways of augmenting the reward signal to perform better in this final scenario. One idea comes from *gap control*, which is actively used in the WEST plasma control system (Figure 5.10 - red arrows). Developing a reward component on the three studied locations (upper, equatorial, and lower ones) could prevent the policy from collapsing in a myopic mode of operation, throwing the plasma on the outer side of the torus.

5.2.2 An issue regarding plasma current

Many trials were attempted to get a stable control of the plasma current. However, the same outcome always appeared: I_p dropped significantly far from the target, avoiding proper considerations of the associated good and bad hyperparameters. A first explanation comes from using the P1 formulation in NICE without resistive diffusion nor transport. Within this basic environment configuration, we observed sudden spikes of plasma current up to several kA [63], which is not physically possible. This concerned all scenarios of interest, no matter how much the reward component on I_p was modified. (Figure 5.11a). The agent then learns representations which are not only biased, but also susceptible to harm learning. Indeed, it observes data linked to incoherent information in shortened data sequences and acts based on these wrong observations: the resulting policy is then

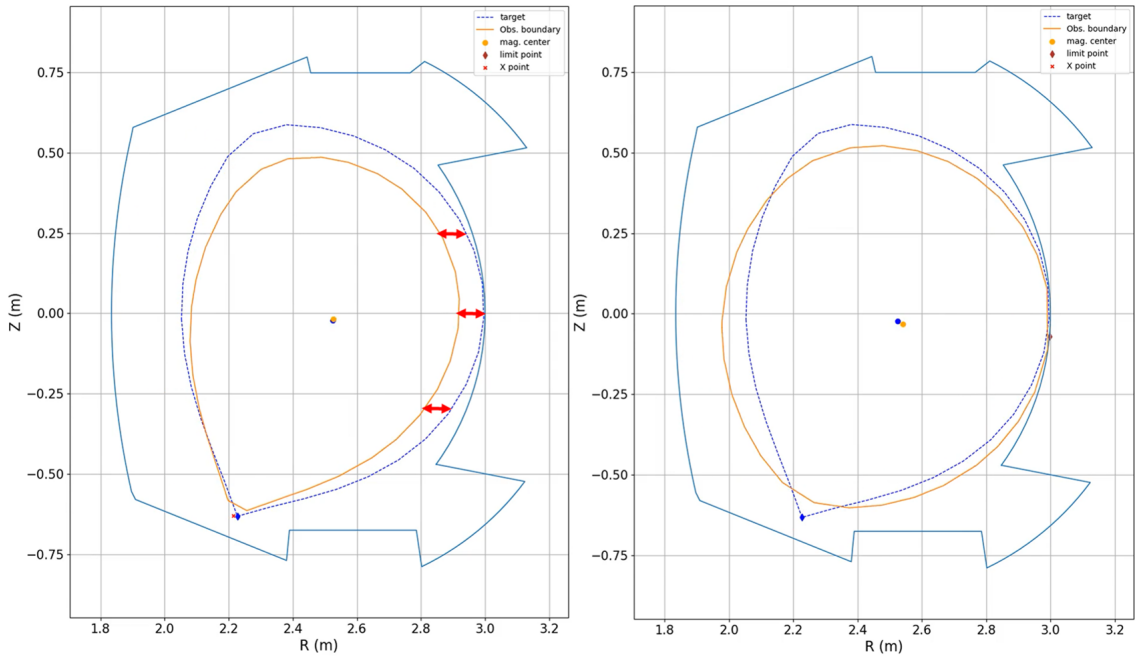


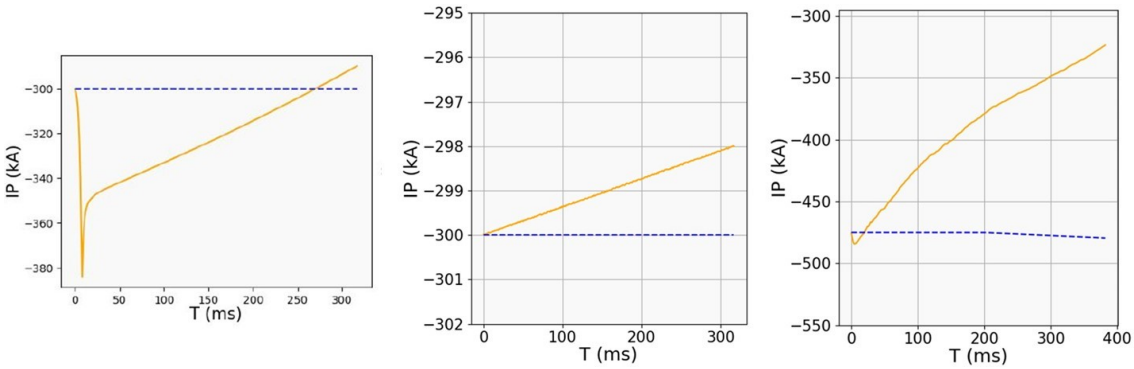
Figure 5.10: When performing an X-point transition, the agent exploits a sub-optimal way of positioning the plasma. The X-point configuration is either lost or does not match an important part of the targets.

sub-optimal and biased regarding the real dynamics of the plasma. Even after extensions of NICE to the resistive diffusion and the electron energy transport, supported by the improved initialization procedure, the RL-based controllers would fail the control of the plasma current. A formal way exists to explain that this phenomenon will always happen on WEST whenever integral control is lacking. Considering mutual inductance M between the plasma and the coils, we obtain a circuit diagram of the WEST tokamak with the central solenoid CS, the latter's current I_{CS}, and the plasma one I_p (Figure 5.12).

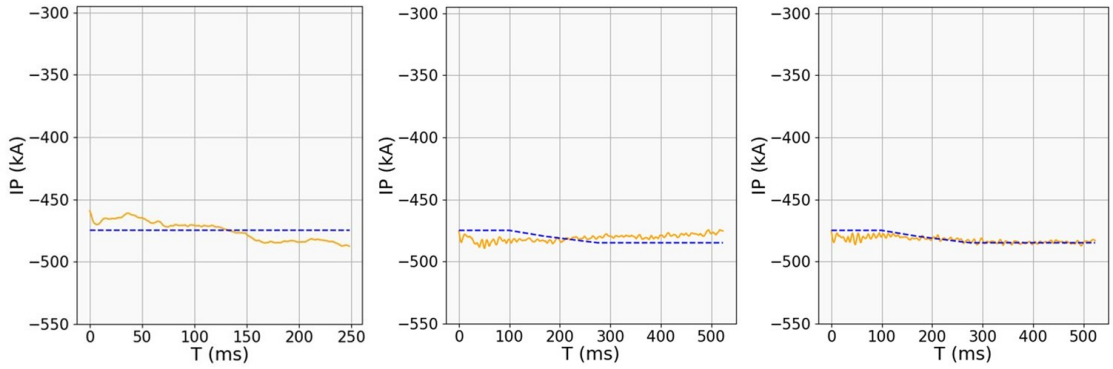
Notations follow with the plasma resistance R_p and self-inductance L_p, the central solenoid resistance R_{CS} and self-inductance L_{CS}, and the source voltage V_CS. We can develop the circuit equations related to this lumped system through the *Kirchhoff circuit law* stating that voltage around any circuit is equal to 0:

$$\begin{cases} L_p\dot{I}_p + M\dot{I}_{CS} = -R_pI_p \\ M\dot{I}_p + L_{CS}\dot{I}_{CS} - R_{CS}I_{CS} = V_{CS} \end{cases}$$

We retrieve the idea stated in the introduction that a varying current in the central solenoid will induce a voltage in the second circuit related to the plasma, which leads to the plasma current. In steady-state regimes, the plasma current I_p is constant, and its first-order derivative is then equal to zero. As a side note, inductance terms could be considered constant. Then, by integration, we get:



(a) No matter the scenario, I_p diverged either because of many numerical instabilities in the NICE environment, or potentially because of another explanation related to the conception of WEST.



(b) After integration of integral effects within the PILOT framework, the control of the plasma current becomes more efficient with examples using limiter maintain and X-point evolve scenarios, as well as the latter with a combination of integral input and reward components (from left to right).

Figure 5.11: The two sides of the plasma current control

$$\begin{cases} MI_{CS} = -R_p \int_t I_p \, dt \\ L_{CS}\dot{I}_{CS} = -R_{CS}I_{CS} + V_{CS} \end{cases}$$

We quickly see that the integral over I_p will increase over time, and only I_{CS} can compensate through this phenomenon. This current is directly linked to the source voltage V_{CS}, which varies over time. Hence, the steady-state error related to the plasma current will increase without the latter. Formally, it is shown that one must add an integral effect, trying to minimize the term in question. This is explicitly done by a controller varying the appropriate voltage source in the solenoid, i.e. the A coil.

It can be defined as part of the reward definition, with a target looking to minimize this integral error over time, or as part of the input like state observers in classical control. This second path was notably implemented in [118], as an average error term computed

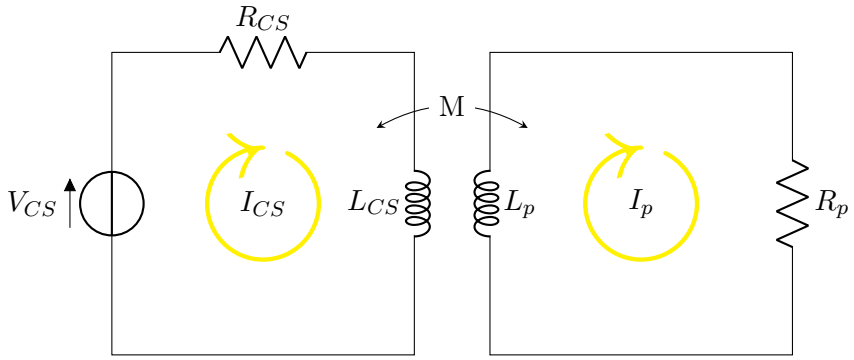


Figure 5.12: Circuit equations for the central solenoid and the plasma. We would refer to the A coil to perform variations of the plasma current.

at each timestep and added to the input observations. The results align with our previous statements since the steady-state error on the plasma current is improved compared to earlier works [29]. Applying such an idea in our context leads to the same conclusion: plasma current control is now enabled in our framework (Figure 5.11b - left and center). The variations observed in the limiter maintain example come from the fact that the plasma current component is not weighted as much as in the other scenarios. We also tested the approach using an exponential decay of the accumulated error d, in the form of $d_t = \alpha e_t + (1 - \alpha)d_{t-1}$, with $\alpha = 0.9$. Nevertheless, we did not empirically identify any advantage given to one of the two approaches; only the overall addition of the integral effect matters. Future studies would need to carefully determine if the initial failure in I_p control came from WEST specifications in the simulation or if it could have benefited more hyperparameter searches. The second idea remains challenging because of the available computing resources, so the proposed approach completely fits our needs. A question arises about what could happen if we couple this observer-like error measurement with an actual new reward component performing the same objective. This combination helped reduce the steady-state error even more (Figure 5.11b - right). The integral reward component has good and bad parameters, respectively set to 1500kA and 15000kA, and does not induce much more complexity into the reward calculation. If we remove the observer input, with or without the integral reward component, the poor control performance appears again, and no proper plasma current control can be performed. This example shows how classical control can interact with reinforcement learning, and the idea is also applicable to the plasma centroid coordinates. It is worth noting that this approach requires the computation of the I_p error over time in the WEST control system so that it can be fed to the neural policy. So, adding the integral input on other controllable parameters would need reconstruction codes or any tool to infer I_p in real-time. This would take out one of the advantages of RL since a few plasma characteristics would not be observed directly from the diagnostics.

	LCFS	κ	minor radius	I_p	X-point distance
Limiter maintain	2.1cm	0.057	0.87cm	4.3kA	X
Limiter evolve	2.8cm	0.023	0.99cm	3.01kA	X
X-point maintain	3.04cm	0.013	1.3cm	2.03kA	0.63cm
X-point evolve	12.3cm	0.053	7.4cm	1.7kA	0.84cm

Table 5.1: The root mean squared error of components of interest is averaged for three different initial conditions (core profiles) in each scenario, after fine-tuning at best. The integral effect is used in all of them for I_p control.

5.3 The Good, the Bad and the Ugly of Curriculum learning

In initial works [29, 63], training time could go up to several days for proper convergence of MPO. This is not in line with the requirements of the WEST control system, which needs adaptable and flexible controllers. To answer the second problematic of this manuscript, curriculum learning is introduced as well as the chunks procedure described by [118]. The combined approach is denoted *CLC*, and the comparison is performed against the vanilla procedure, which does not rely on chunks or curriculum learning. Training results are averaged with three different seeds incremented over all distributed actors used to fill the replay buffer. We analyze only the X-point evolve scenario, but the results generalize directly to the other scenarios. Figures are displayed using episodes as time units. This is because computing speed was variable during training (and between tasks within the latter), as the shared resources could not be entirely available. That is why we compute average simulation step durations depending on the situation and use this approximation with respect to the number of considered episodes. The reward set for the Time To Threshold (TTT) is 25, as control starts to perform sufficiently well with regards to the duration of 400 ms. For intermediate tasks in the curriculum, their duration is caped to 60 evaluator node episodes so that they do not exceed 30 hours each of execution, at worst. This threshold was identified after benchmarking MPO's start of learning, PILOT's communication, and NICE speed on dedicated servers.

First, an environment's step lasts for 15 seconds on average during exploration (Chapter 3) since the plasma moves to locations of the vacuum chamber where convergence of NICE is more difficult to achieve. In this way, these complex situations where poor reward signals are standard exhibit tedious and lengthy exploration. Consequently, an episode has its computing time increased to almost 3 hours when it reaches its entire duration of 400 timesteps. Thereby, the monitored convergence time, when trained from scratch, reaches the symbolic length of a whole week. Furthermore, the reward never surpasses 20 inside the 60 episodes limit and struggles to attain 25 afterwards. This is largely under our expectations regarding TTT (Figure 5.14 - upper). One could mention that further hyperparameters search could have been performed on the reward definition. However, we kept it similar to its previous definition to efficiently compare the performance of one procedure against the other in the same conditions from which we obtained the results presented in the last sections. Furthermore, this search over the hyperparameter space

could have helped in finding a better policy with the vanilla method, but convergence would have still been slow, which is not in line with our primary goal. Indeed, our interest is more related to how much we can speed up training, and we look for reward performance only through the prism of *minimal required performance*. By that, we mean that to answer the second problematic of this PhD, one must find a way to build efficient controllers as fast as possible. If one controller achieves good performance without reaching the full reward, we still consider it a step towards faster production of magnetic controllers. Future works will then focus on optimizing the procedure as stated in the approach's limitations.

Method	Jumpstart on the final task	TTT
Vanilla	4.3	180h
CL	-10.2	24h

(a) Comparing transfer metrics between the vanilla and CLC methods.

	Average reward	Error margin
Vanilla	5.2	±3.65
CLC	18.4	±4.23

(b) Average reward over tasks.

Figure 5.13: Study of the control policy learned from scratch and the CLC method, across three seeds.

Returning to our analysis, the CLC procedure implicitly pushes the exploration to reachable states that are considered easier at the beginning of the initial task. Thus, the duration of an environment step becomes shorter on average, and NICE converges to an equilibrium in about 2 seconds. Good results from the initial task condition next ones, keeping this state assumption valid, i.e. each new task will start from a better-than-averaged behavior. Moreover, the chunks partitioning enables efficient diversification of the situations filling the replay buffer, meaning that the state space coverage is greatly improved right at the beginning of training. These two elements lead to 2.3 seconds on average for the remaining parts of the curriculum. Finally, we obtain episodes computed at worst in 0.26 hours if the maximum episode duration is reached, which is already an auspicious outcome. This way, the reward threshold is reached in about 120 episodes, and the TTT is reduced to approximately 24 hours. Previous attempts[1] reduced the TTT to 60 hours, which was before optimization of the reward components displayed in Appendix D. Even so, we observe an apparent reduction in convergence time towards a reward sufficient to perform magnetic control in each configuration of interest (Figure 5.13a). The training was stopped before 60 evaluation episodes for the final curriculum task, with a stable return of about 25. The TTT of 24 hours is higher than the convergence time of 10 hours presented in the work introducing the chunks procedure [118]. Nonetheless, the training timescales depend on the machine and the environment, and only the procedure matters to increase training speed drastically.

[1]Publications list available at the beginning of this manuscript

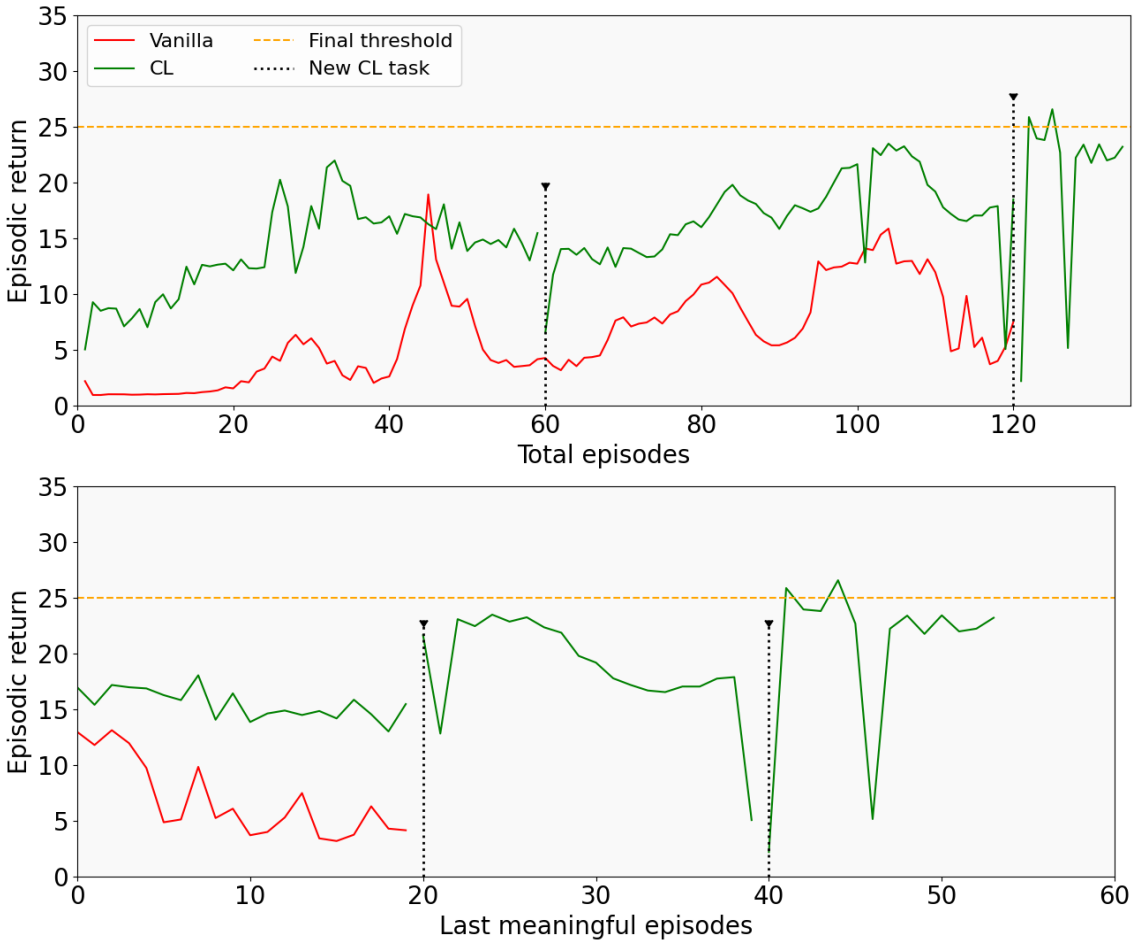


Figure 5.14: Episodic return for both methods (vanilla - red, CL - green). Since MPO goes through a warm-up phase, we consider the last meaningful episodes regarding reward convergence.

One important thing to notice comes from the jumpstart using the total number of episodes. The curriculum strategy sometimes performs worse than the best return for the vanilla approach at each new task (Figure 5.14 - upper). A concise explanation comes from the fact that the new reward definition has an increased complexity, which inevitably drops the return at start. Another explanation could arise from catastrophic forgetting, but a more rigorous evaluation must confirm this hypothesis. This phenomenon could occur for many reasons, while this PhD only displays a practical application of the method. This means careful analysis is required to conclude which part of the curriculum is responsible for this punctual performance loss. After those sudden reductions, the agent fails its first attempt but recovers at the end. Let us remind that we are not stopping previous tasks based on control performance, but rather to constrain the entire training time. Again, we do not necessarily look for strong performance on intermediate tasks; we only look

for enough knowledge transfer to boost performance after each curriculum step. So, this observation is not entirely surprising since no optimal behavior is guaranteed nor expected at the end of each intermediate task. Furthermore, MPO needs several initial exploratory episodes for training to start concretely with informative data at hand. The curriculum strategy could also be evaluated without these warm-up phases. Hence, we restrict our findings to the last 20 meaningful episodes (Figure 5.14 - lower). Consequently, both metrics give better results, as only improved behaviors are considered: the jumpstart is significantly better, despite the last drop for the last transition, and the time to threshold is reduced. Curriculum learning improves the final average performance(Figure 5.13b), as it displays higher reward than the vanilla policy (Figures 5.14 - both). It shows that the approach does not induce training instabilities, apart from the hypothetical catastrophic forgetting.

5.4 A paradigm under careful calibration

These examples provided a qualitative and quantitative assessment of RL-based controllers over the four scenarios studied in this work. More than just performance, they highlighted the capacity for reinforcement learning to handle relatively small disturbances, fast transitions, and unstable moving targets. The related results answer the first problematic of this PhD, as RL-based solutions exhibit interesting behaviors in line with what was observed in the literature. Nevertheless, this is done by properly calibrating the reward function, which makes the overall setup prone to overfitting. This could be considered the underlying objective of these trainings: we want a controller that tracks the reference as closely as possible, overspecializing on the related scenario without intending to generalize effectively to others. In a sense, we look to "overfit" to a minimal working example, robust enough in front of the uncertainties of plasma dynamics. Finally, curriculum learning is coupled with the chunks procedure from the state-of-the-art and allows for fast training and increased general performance while transitioning to an X-point. This first proof of concept requires ablation studies to identify which part of the curriculum could be optimized and which specification could cause the said catastrophic forgetting. For now, we performed a simple statistical study with a qualitative description of the potential and existing difficulties. Nonetheless, RL lacks a deterministic tool to assess its robustness in the same way classical control offers. Future works are directed towards finding ways to assess the robustness and stability of the approach in a more precise way.

Conclusion and perspectives

This PhD aimed at defining the first building bricks enabling the use of deep reinforcement learning for tokamak control. The PILOT framework was developed from scratch to allow the training of magnetic controllers with enough flexibility to address various problems. It incorporates an agent chosen as the Maximum a posteriori Policy Optimization algorithm, and an environment based on the NICE software. Both were built to answer the first problematic of this PhD, namely the design of core components required to use reinforcement learning for magnetic control on WEST:

- the NICE environment acts as a numerical twin of the tokamak, taking into account power supply, diagnostics and noise models. This correctly mimics the uncertainties and specifications of the device. It is worth realizing that NICE required rigorous configuration to stabilize the solver and obtain helpful plasma information for training;
- the MPO algorithm is an alternative interpretation of reinforcement learning from the point of view of probabilistic inference. This distributed actor-critic was implemented after carefully understanding its dynamics and properly fine-tuning its hyperparameters. Its advantages come from its low sensitivity to small changes of the latter, depending on the use-case, and its capability to run numerous parallel actor instances to gather more data in a replay buffer. This makes its setup straightforward for magnetic control within the landscape of potential applications;

The plasma scenarios considered for this work are inspired by the ones used in the state-of-the-art [29, 118]. The presented study is complementary to the original one performed on TCV, strengthening the idea of RL as an alternative to classical control design in tokamak operation. As such, the reward signal is computed from an observed plasma state and a plasma target interpolated by the reference generator of PILOT, starting from an ordered set of desired plasma snapshots. These rewards are designed in an interpretable way so that post-training analysis rapidly informs of potential updates in the reward definition. This reasoning gives the final answer to the first problematic as trained controllers perform well on the said scenarios, comparably to what is observed in the aforementioned literature.

It is essential to realize that this entire research requires much initial engineering effort to develop the core elements needed for RL-based magnetic control, including exhaustive hyperparameter searches. Nonetheless, it usually comes down to fine-tuning a few reward components once the general set of hyperparameters has been optimized.

However, the paradigm of interest has multiple pitfalls, making its adoption for routine use in tokamak operations unlikely to happen on a short timescale. As it is, it still needs careful reward design to find the right combination of components, and is restricted to only one scenario per training. Most importantly, training times easily trespass the symbolic week threshold: the method uses a flexible framework, but is not flexible itself to produce magnetic controllers rapidly. This is particularly problematic if we compare it to classical controllers, which can be obtained much more quickly. This comes mainly from the NICE environment, which can take hours to advance the plasma equilibrium in time for the entirety of an episode. Instead of an exhaustive modification of NICE, we leverage the use of curriculum learning, a simple procedure that sequentially increases the difficulty of control objectives. By that, we mean that a sequence of tasks of increasing complexity is produced, and the agent is trained on each of them sequentially. It is known to improve performance and reduce convergence time towards an optimal policy by several orders of magnitude. Previous works [118] also tried to speed up the production of magnetic controllers, and their method can be combined trivially to curriculum learning. The results show the potential of the mutual approach, as training duration goes from more than a week to 24 hours on average. Nevertheless, the method presented in this research has several limitations which make difficult its generalization to any case:

- the curriculum is handcrafted, and despite being generalizable to many scenarios, it is not formally proven to be the best one;
- the action and state spaces remain the same between tasks, but nothing guarantees that it is required, i.e. we could have restricted the action space to a subset of the control coils in initial tasks, to speed up training even more;
- curriculum might be subject to catastrophic forgetting which causes divergence of the policy in several occasions.

A discussion is conducted to counteract these issues, alleviating progressive neural networks as promising architectures that can not suffer from catastrophic forgetting, while being similar to regular fully connected neural networks. Their complexity is not in line with the real-time constraints of WEST magnetic control, and only future works on policy distillation could make such networks part of the training procedure. It is worth mentioning that the latter could enable multi-scenario control by involving multiple expert controllers. Furthermore, reinforcement learning does not exhibit tools to assess the stability and robustness of an agent in the same way it is done in classical control. Because of that, it is impossible to evaluate consistently the involved algorithm apart from rigorous statistical analysis. Nonetheless, the second problematic is answered as curriculum learning limits training duration to what is expected from the field. It is a first step towards making RL a routine part of tokamak operations.

The results show that RL-based controllers can follow complex trajectories if the balance between reward components has been carefully designed. Indeed, without doing so, we can end up with low-performance controllers incapable of handling disturbances and uncertainties in plasma dynamics. Since NICE is augmented with more realistic modes, the agent benefits from more accurate data, and training becomes more stable as sudden changes in how the agent pictures the state space occur less frequently. It is essential to consider that RL-based controllers are trained on reference trajectories inspired by experimental ones. This means they refer to trajectories that are close if not similar to PID control extensively used on WEST. Arguably, we only reproduce more robustly what classical controllers are already capable of. One could take the idea even further by stating that maximizing the reward in our context comes down to overfitting the neural network to a specific scenario already explored on the device. It would be beneficial to look for ways to learn representations that did not occur in the set of existing control experiments, but this is outside the scope of what was presented in this manuscript. Ultimately, deep RL displayed a strong potential in producing magnetic controllers for plasma control on WEST, from limiter to X-point configurations. All developments are now focused on a practical deployment on WEST, by fine-tuning the X-point transition to remove any steady-state errors on the LCFS before producing a binary ready for execution on the WEST control system.

Deep reinforcement learning is a paradigm displaying many hopes for the future. From control of many configurations to multi-scenario control, there is room for improvement at each level of the developed framework. Indeed, we identified training speed as a major bottleneck and tried to counteract this issue through a simple yet effective curriculum learning procedure. This was justified as a path contained within the neighbouring scope of this PhD, with other potential approaches being too complex or too far from the applicative character of this research. However, prospective works could definitely focus on them, not only to enhance the capabilities of PILOT, but also to optimize the overall performance of RL-based controllers. Typical prospects include:

- Enhancement of the NICE environment:
  - *Surrogate models*: With computing times reaching hours to simulate several seconds of plasma, the NICE software might have to undergo a few modifications to increase its execution speed. One idea would be to replace expensive steps within the evolution mode, with neural networks trained in a supervised manner to output an approximate yet precise solution of these computations. One could go even further by replacing NICE with a neural network directly solving the plasma evolution equation [61, 60]. However, this would come at the cost of losing many functionalities that are more interpretable than neural networks. It would be worth studying the balance offered by these concepts since a fast simulator would allow the application of RL-based solutions to long plasma pulses operation.
  - *Coupling codes*: In fusion applications, the coupling of different codes is standard to refine the initial conditions of a simulation, or even its intermediate

calculations, with better defined parameters. Even if NICE includes resistive diffusion and electron energy transport, it could be beneficial to augment it with a complete transport solver like METIS [10]. This would not only offer the previous advantages regarding stability but also enable more controls within WEST (density control, etc), with an extended set of actuators (heating systems, etc). It is worth noticing that coupled codes are also concerned by the surrogate models approach.

- Improvements over the agent:
  - *Comparing algorithms*: The Maximum a posteriori Policy Optimization is the sole algorithm used in this PhD. Despite correct performance on various control scenarios and different tokamaks (TCV and WEST), it is not the only existing solution. It would be beneficial to benchmark multiple actor-critic methods [105, 103, 48] to conclude which algorithm suits magnetic control best.
  - *Exploring neural architectures*: In this work, one of the primary intents was reproducing the state-of-the-art. It then reuses a similar setup where the policy is parameterized by a regular fully connected neural network and the critic by an LSTM recurrent one. Nonetheless, many architectures could be particularly efficient not only during training but also for inference. It includes *recurrent state-space* models [49, 31], which integrate classical control ideas to the recurrent approach. The obtained networks could benefit from increased interpretability and proper performance since they efficiently model long-range sequences. Their relatively low complexity makes them capable of being deployed for the critic, and the policy that has to function under real-time constraints. This is different from what we observe from LSTMs or even transformer networks[72], which model long-term dependencies at high complexity, making their use impractical for the control policy. These options must be benchmarked to enhance the horizon of events that the RL-based controller could foresee.
  - *Improving exploration*: The Maximum a posteriori Policy Optimization already exhibits interesting properties in the dilemma between exploitation and exploration. Through careful definition of the hard constraints over policy search, exploration is influenced in policy space, benefiting from probabilistic inference principles. Exploration could be augmented in many ways, from exploration bonuses [51] to a restyle of the reward design integrating new transformations.
- Routine training:
  - *Curriculum learning*: We displayed curriculum learning as a way to speed up the convergence of the algorithm and general performance on complex tasks. This has been shown to constitute a powerful tool that could be used in the future to produce magnetic controllers for each new experimental campaign rapidly. However, we applied this methodology in a restricted setup, which would require more ablation studies. This concerns an analysis of the learning

dynamics, within which unwanted phenomena such as catastrophic forgetting can happen. Ultimately, it would help to better transfer the policy between tasks. An automatic curriculum should be compared with the handcrafted one to discover new learning trajectories and accelerate training even more.

- *Progressive networks and distillation*: A clear limitation of the current curriculum approach comes from the risk of catastrophic forgetting. Even if the mentioned ablation study is performed, this would still endanger the adoption of this routine as part of PILOT's main features. *Progressive Neural Networks* [98] were presented and are implemented in PILOT. They will be tested accordingly, with proven guarantees regarding catastrophic forgetting. However, these architectures have high complexity, making them difficult to use in real-time control systems. One could prune the network to overcome this bottleneck since several of its weights are not required after transfer [99]. A more elegant solution might come from *Policy Distillation* [97, 26]. By training a PNN with curriculum learning, an expert policy could be learned quickly and distilled in a supervised manner into a smaller network, which aligns with the plasma control system requirements.
- *Continual learning and multi-scenario control*: A discussion was made on the use of policy distillation not only to distillate the knowledge of one expert policy into a smaller network but also to perform the same procedure using many experts trained on different scenarios. By doing that, the small network could handle multiple control trajectories and pave the way towards multi-scenario control. This idea is part of a general concept known as *Continual learning* [126], in which we try to optimize agents under requirements changing over time, i.e. agents that have to learn many new tasks often, while keeping intact their acquired knowledge. This would constitute a final goal where RL-based controllers are adapted swiftly to changes in the overall control objectives across devices. This is intrinsically close to curriculum learning while focusing even more on preventing catastrophic forgetting no matter the neural architecture employed [34, 93].
- *Experimental data*: In this work, training was performed entirely with simulations. This choice was motivated by models of plasma evolution that were readily accessible from the start of this journey, while relying on complete datasets of experimental data was not evident. The latter could be used to perform offline reinforcement learning [110] or imitation learning [73], offering new perspectives for comparison with the model-free off-policy approach presented in this work. An excellent idea could come from using these experimental data to pre-fill the replay buffer. Easy to implement, it would enhance cold start of training with information guiding the controller at the beginning of exploration.

Formal comparison between RL and OC

Let us consider a simple offline optimization scheme, that can be exhibited while assuming a general case where an optimal trajectory exists with implicit state and control constraints. Consider a dynamical system f, a control law u, and g an observation function over the full state:

$$\forall t \in [0, \infty[\quad \begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = g(x(t)) \end{cases} \quad \text{with} \quad x(t) \in \mathbb{R}^n, \quad u(t) \in \mathbb{R}^m$$

We look for trajectories x and u that minimizes the total cost $L(x, u)$:

$$x^*, u^* = \arg\min_{x,u} L(x, u) = \arg\min_{x,u} \int_0^\infty l(x(t), u(t))dt \tag{A.1}$$
$$\text{with} \begin{cases} x(0) = x_0 \\ \forall t, \quad l(x(t), u(t)) \geq 0 \end{cases}$$

One element remains important regarding its computation: the integral over an infinite interval. While infinite horizon optimal control is defined formally, it is somehow ill-posed since it could result in an infinite cost even in the case of optimal trajectories. We introduce a discount factor of the form $\mathcal{O}(\beta^t)$ with $0 < \beta < 1$ which decays towards 0 as $t \to \infty$. This will ensure that the cost will be bounded without turning the problem into a proper finite horizon one. We could also add a terminal cost that penalizes the state attained at the final state, but the discounted formulation allows for a more straightforward connection with RL definitions. So, we end up rewriting (A.1) into:

$$x^*, u^* = \arg\min_{x,u} L(x, u) = \arg\min_{x,u} \int_0^\infty \beta^t l(x(t), u(t))dt$$
$$\text{with} \begin{cases} x(0) = x_0 \\ l(x(t), u(t)) \geq 0 \\ 0 < \beta < 1 \end{cases}$$

Discretization is often needed as a necessary step before actual computation, and we will use such representation from now on. Optimal trajectories become discrete as $\{x(i)\}_{i\geq 0}$ and $\{u(i)\}_{i\geq 0}$, with discrete dynamics f_d, and minimizes a discounted cumulative cost:

$$x^*, u^* = \arg\min_{x,u} L(x, u) = \arg\min_{x,u} \sum_{k=0}^{\infty} \beta^k l(x_k, u_k) \tag{A.2}$$
$$\text{with} \begin{cases} x_{k+1} = f_d(x_k, u_k) \\ u_{k+1} = u(x_k) \qquad \forall k \geq 0 \\ x(k) = x_k \\ l(x_k, u_k) \geq 0 \end{cases}$$

We can, of course, restrain the optimization from any starting state, and by development, we turn (A.2) into:

$$x_t^*, u_t^* = \arg\min_{x,u} L(x_t, u_t) = \arg\min_{x,u} \sum_{k=0}^{\infty} \beta^k l(x_{t+k}, u_{t+k})$$

with discrete time $t = i \times T$, with timestep size T and $i \geq 0$. Precisely, we can rearrange $L(x_t, u_t)$ in order to obtain equations comparable to what has been seen in the previous section:

$$\begin{aligned} L(x_t, u_t) &= \sum_{k=0}^{\infty} \beta^k l(x_{t+k}, u_{t+k}) \\ &= l(x_t, u_t) + \sum_{k=1}^{\infty} \beta^k l(x_{t+k}, u_{t+k}) \\ &= l(x_t, u_t) + \beta L(x_{t+1}, u_{t+1}) \end{aligned}$$

Let us recall that RL solves an infinite horizon problem by learning an optimal policy π_*, which maximizes a discounted cumulative reward over time. Notably, it relies on the value function from starting state s_t:

$$V_\pi(s) = \sum_a \pi(a|s_t) \sum_{s'} P(s'|s,a)[\mathcal{R}(s,a) + \gamma V_\pi(s')]$$

If we consider a deterministic MDP and policy, with the value function definition in mind, we find a straightforward connection between each paradigm in terms of cost/reward-to-go:

$$\begin{aligned} V_\pi(s_t) &= \mathcal{R}(s_t, a_t) + \gamma V_\pi(s_{t+1}) \\ L(x_t, u_t) &= l(x_t, u_t) + \beta L(x_{t+1}, u_{t+1}) \end{aligned}$$

Furthermore, looking at optimality, a similar pattern is observable:

$$\begin{aligned} \pi_* &= \arg\max_\pi V_\pi(s) = \arg\max_\pi \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_k | s_t = s] \\ x^*, u^* &= \arg\min_{x,u} L(x,u) = \arg\min_{x,u} \sum_{k=0}^{\infty} \beta^k l(x_k, u_k) \end{aligned}$$

More than a difference in notation conventions, an interesting take on their dissimilarities could come from the question of stochasticity. Indeed, MDPs are a discrete stochastic formulation of optimal control problems [113]. This equivalence could also be obtained by referring, for example, to the injection of noise both in the dynamics and observations so that $x_{t+1} = f(x, u, \mathcal{N}_x)$ and $y_{t+1} = g(x, u, \mathcal{N}_y)$. This would still initially be formulated without any feedback mechanism in the optimization process.

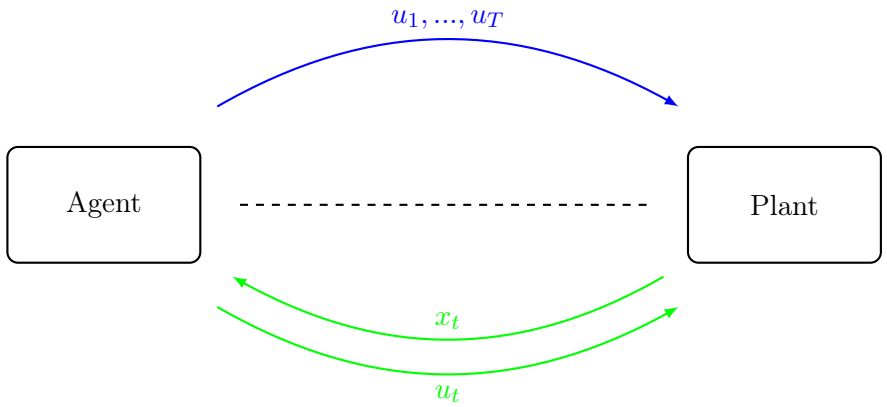


Figure A.1: Open (blue) and closed loop (green) interaction between an agent/controller and a system. The closed loop enhances adaptability and allows the use of real-time information at each step.

Moreover, whereas RL is mainly seen as a *closed loop* system where the policy is learned by feedback from the environment, optimal control is primarily seen as an *open-loop* planning formulation where the entire optimal control trajectory is optimized only at first timestep (Figure A.1):

$$u_1, ..., u_T = \underset{u_1,...,u_T}{\arg\min}\, L(x, u) \quad \text{with} \quad \begin{cases} x_{t+1} = f(x_t, u_t) \\ u_{t+1} = u(x_t) \end{cases} \tag{A.3}$$

Even if we consider a stochastic version of the open loop, the disadvantages of the approach remain the same, as it can not benefit from real-time information, inducing potential inefficiency and risks of instability:

$$u_1, ..., u_T = \underset{u_1,...,u_T}{\arg\min}\, \mathbb{E}_u[L(x, u)|u_1, ..., u_T] \quad \text{with} \quad \begin{cases} x_{t+1} = f(x_t, u_t, w_t) \\ u_{t+1} = u(x_t) \\ w_t \sim \mathcal{N}_x(\mu_x, \sigma_x) \end{cases}$$

Most importantly, a major concern regarding the open-loop approach is that the dynamics model is different from reality. This could cause the obtained trajectory to diverge from the optimal behavior, as no feedback could be used to correct modeling errors. The gap between the latter and the former can be closed by *Model Predictive Control* (MPC). Starting from state x_0^*, we would simulate the closed loop for $t \in [0, T-1[$:

$$u_t^* = h_t^*(x_t^*)$$
$$x_{t+1}^* = f(x_t^*, u_t^*)$$

MPC traditionally finds the optimal control trajectory $(u_0^*, ..., u_{T-1}^*)$ by optimizing through a sliding window for reference $\overline{y}$. Figure A.2 displays a closed loop where the

output of the system is given by $g = Id(.)$. Indeed, we only design u_0^* and start by solving (A.3) on discrete times spanned by the initial horizon Δ_0. The first element of the control law h_0^* is kept as u_1^*, and shifting of the time window to Δ_1 is performed. Once again, (A.3) is solved on the new window: the first value of h_1^* is kept as u_2^*, shifting to Δ_2. We repeat this online process sequentially, yielding optimal trajectories x^* and u^*. Formally, MPC is an optimal control method which does not assume any hypothesis on linearity, and is capable of handling strong disturbances because of its efficient account for the receding horizon. Following the discrete formulation, and since we bound the sum to a time window of length N, this turns the whole setup into a tractable finite horizon problem:

$$x^*, u^* = \arg\min_{x,u} \sum_{k=t}^{t+N} l(x_k, u_k), \quad \forall t \geq 0,$$
$$\text{with} \begin{cases} x_{k+1} = f_d(x_k, u_k) \\ u_{k+1} = u(x_k) \qquad \forall k \in \{1, \cdots N\} \\ x(k) = x_k \\ l(x_k, u_k) \geq 0 \\ N > 0 \end{cases}$$

Consequently, optimization is still at the heart of MPC. Indeed, even if a closed-loop form is accessible, we optimize for an entire trajectory at each timestep. Because of that, fast hardware is required to recompute every timestep for the updated horizon and obtain a refined control strategy. Sufficient computing power enables direct optimization of those systems in reasonable times, without an extensive need for linearization and *Linear Quadratic Regulator* (LQR) control. MPC and LQR are both OC methods, but the second assumes linearity and quadratic costs. With MPC, such a hypothesis does not hold, as it can deviate from equilibrium states without becoming weak in its control objectives or needing quadratic costs. However, there are no guarantees of an optimal solution for such non-linear systems, which makes MPC a more efficient solution in terms of performance and robustness but a hazardous option when it comes to global stability [68]. Therefore, we can say that MPC allows a closed-loop formulation of OC where trajectories are optimized continuously in an online manner. This formulation allows a new comparison with RL which better contextualizes the need for closed-loop control, while general optimal control helped realize the stochastic properties of RL. One question arises then: *Which general theory bridges the gap between RL and MPC ?*

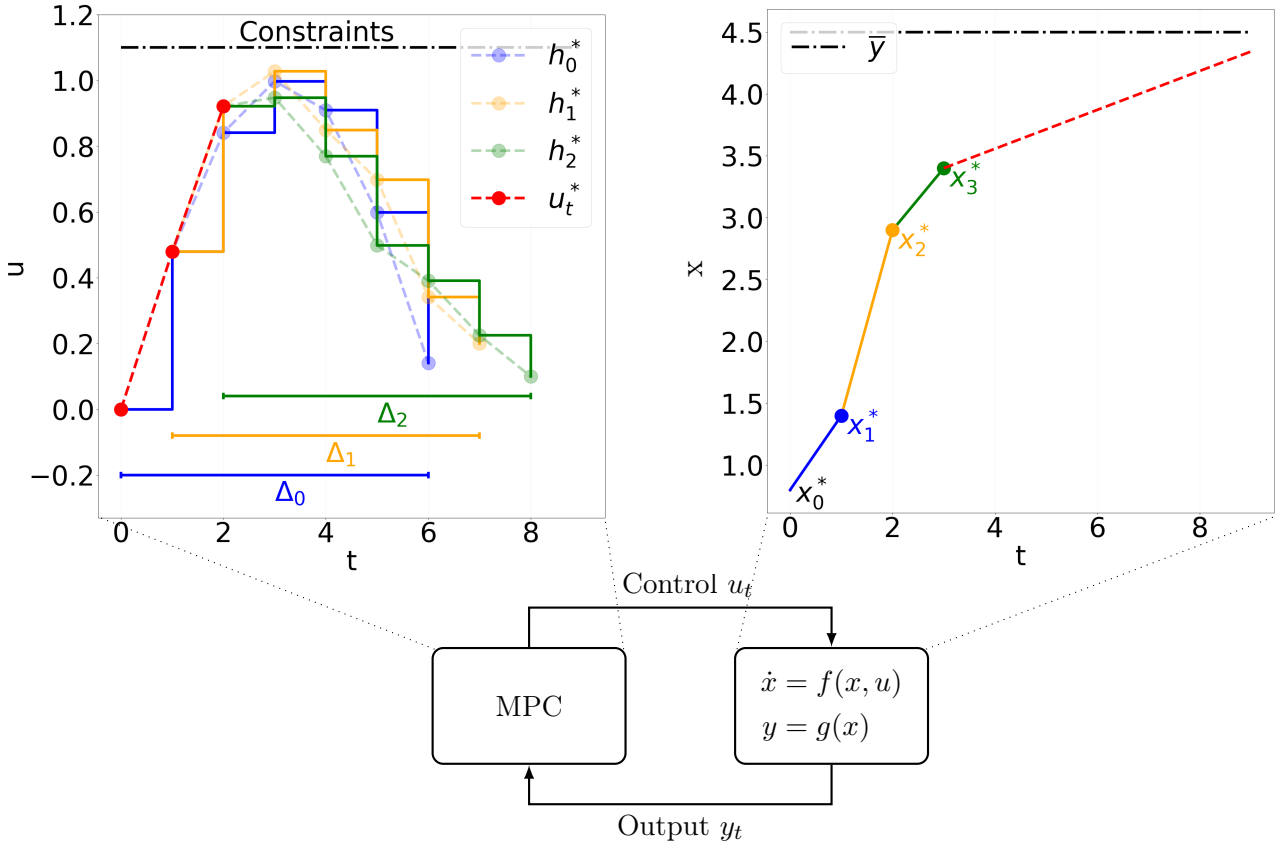


Figure A.2: Online MPC where optimal control is computed iteratively.

In Chapter 2, policy and value iteration were mentioned as algorithms used to find an optimal policy in the presence of a perfect MDP. They are derived from *Dynamic Programming* (DP)[12], which is at the basis of RL and initially a way of solving MPC using the said Bellman equations. Starting from *Bellman's principle of Optimality*, an optimal cost trajectory has the property that any subsequent partition is optimal (Figure A.3).

Definition. Bellman's Principle of optimality An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Based on this idea, we get a recursive definition of the optimal cost for each interval $[k, k+1]$:

$$L_k^*(x_k, u_k) = \min_{x,u} l(x_k, u_k) + \beta L_{k+1}^*(x_{k+1}, u_{k+1})$$

Here, we must notice that with its subscript, L^* is non-stationary as it depends both on time and state. In such a case, the optimal cost-to-go could evolve at each timestep,

which is not a situation covered in this brief analysis. This forward recursion, called exact dynamic programming, allows to compute an optimal feedback control law $u_k^* = h^*(x_k) = \arg\min_{x,u} l(x_k, u_k) + \beta L^*(x_{k+1}, u_{k+1})$, as well as an optimal state trajectory x^* by the closed-loop system $x_{k+1}^* = f(x_k^*, u_k^*)$. Requiring a stationary property for DP recursion, i.e $L_k^* = L_{k+1}^*$ and generalizing to any state x, the optimal cost known as the *stationary Bellman equation*, and the corresponding control law, can finally get simplified to:

$$\begin{aligned} L^*(x) = L^*(x,.) &= \min_u \underbrace{l(x,u) + \beta L^*(f(x,u))}_{Q(x,u)} \\ h^*(x) = h^*(x,.) &= \arg\min_u Q(x,u) \end{aligned}$$

Trivially, the action-value function from RL is recovered by rearranging notations. Looking only at the action-value has its importance, as discussed in Chapter 2. The missing stochasticity of RL could easily be added as an expectancy over disturbances put on both f and h, but we rely more on the recursive properties of DP to close the gap between MPC and RL. Consequently, the *Bellman operator* $\mathcal{T}$ can be introduced as:

$$\begin{aligned} \mathcal{T}[L](x) &:= \min_u l(x,u) + \beta L^*(f(x,u)) \\ &\iff \forall s \in \mathbb{R}^n, L \geq L' \Rightarrow \mathcal{T}[L] \geq \mathcal{T}[L'] \end{aligned}$$

Again, we quickly recover the Bellman optimality equations. This material conditional is historically known as *monotonicity* in DP, which is directly translatable to policies and rewards from RL. This shows that RL (and DP more generally) can be seen as a fixed-point problem of the form $\mathcal{T}L^* = L^*$. Consequently, the solutions of the Bellman Optimality equations are the fixed points of the Bellman operator. Since the Bellman operator is

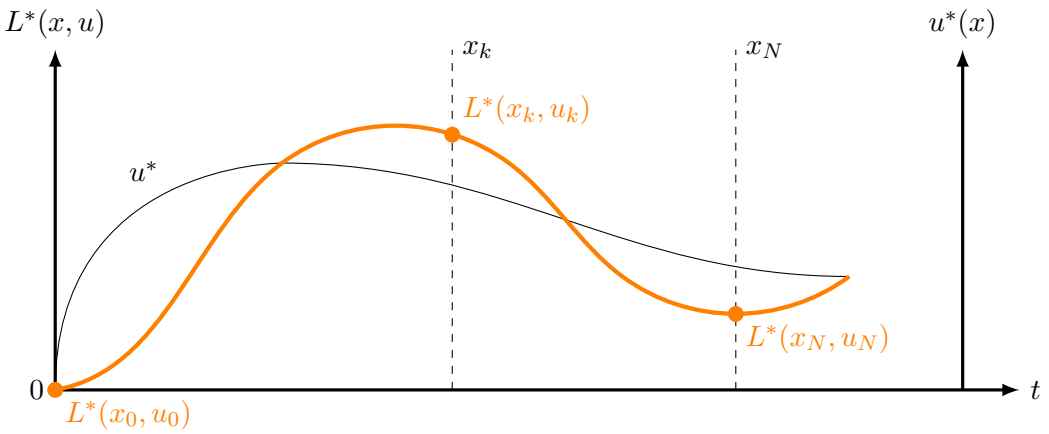


Figure A.3: From optimal trajectories L^* and u^*, any portion [k,N] remains optimal for initial state x_k.

a contraction mapping in infinity norm, one can prove through the Banach fixed-point theorem that it has a unique fixed point [113]. Hence, any policy based on the Bellman optimality equations has values equal to that solution, i.e the optimal policy formally exists.

One could point out that we face a minimization problem, while RL defines a maximization one. However, this difference is negligible, as the Bellman operator can be defined similarly from both perspectives, leading to proper optimality equations. So, DP is an elegant way of solving MPC, performing an exhaustive search of all possible actions for all possible states. Nevertheless, its computational complexity increases exponentially with the number of states, and actions to a lesser extent. For this reason, the so-called *Curse of Dimensionality* becomes a substantial issue. On the one hand, classical MPC restricts computation to only start at the current state s_0. Applying such a method to DP recursion does not remove all problems regarding dimensionality, as the mentioned search remains infeasible for all actions. On the other hand, discretization and reduction of state and action spaces could be considered, at the cost of losing tremendous precision in the process. Overall, both paths would lead to tabular representations of L^*, which can not be exactly implemented in most real applications where the state space becomes too large. Most importantly, a significant bottleneck of DP (including MPC as a general consequence) is the need for system identification. By that, we mean that they require proper knowledge of the system's dynamics, as well as the stochastic processes that can be added to the whole framework (noise, disturbances). If we rely on our second option, we could destroy such an assumption and create an ill-posed problem. That is why dynamic programming and what comes straight from it in reinforcement learning (policy and value iteration algorithms) are restricted to tractable problems, in terms of dimensions and dynamics modelization.

Precisions on value and gradient-based methods

This appendix presents more details about value-based and policy-based methods, notably how the bias-variance trade-off inherent to machine learning helps analyze the outcomes of each approach.

B.1 A focus on value learning

Let us assume a fixed policy π. The purpose of value learning can be illustrated at a high level by first evaluating Q^π and then setting the improved policy using the best values. In the context of actor-critic algorithms, only the first evaluation is performed, and gradient-based optimization is conducted to improve the policy. This initially resolves around *Monte-Carlo* reinforcement learning. Let us define a trajectory τ over state and action spaces, built using π. Let us recall that $\forall s \in \mathcal{S}, Q^\pi(s,a) = \mathbb{E}_\pi[G_\tau | s_t = s, a_t = a]$ with $G_\tau = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. Using τ, Monte-Carlo methods compute the empirical average return over each visited state-action pair for this episode instead of the expected return, starting at step T. More precisely, we update $Q^\pi(s,a)$ using a running average of the return for all occurrences of (s,a) registered from previous trajectories. Through these definitions, we can see that Monte-Carlo approximations definitely work in model-free settings because experiences are assumed to contain enough knowledge to fit the action-value function, without knowing anything about the MDP's dynamics. Nonetheless, this approach is problematic since we need entire episodes to perform training, assuming that they effectively reach a defined terminal state. To avoid this issue, we must use a method that does not rely on complete sequences. *Temporal-Difference* (TD) learning steps in without the need for entire episodes. Its pinnacle principle looks after updating the estimate of Q^π using actual estimates. In a sense, this *bootstrapping* tries to guess the action-value out of previously guessed estimates. Despite our focus on Q^π, we will resolve on V^π to explain TD learning, as keeping our focus on the action-value function would result in a specific case within the TD domain. Hence, the TD update is expressed as:

$$V^\pi(s_t) \longleftarrow V^\pi(s_t) + \alpha \underbrace{[\underbrace{r_{t+1} + \gamma V^\pi(s_{t+1})}_{\text{TD target}} - V^\pi(s_t)]}_{\text{TD error}}$$

with learning rate $\alpha \in [0,1]$ and discount factor $\gamma \in [0,1]$. We look to minimize the TD error, which is the reward from transitioning from state s_t to s_{t+1} augmented with the difference between the discounted value estimate for s_{t+1} and the value estimate for s_t. So, we do not update the value based on the entire return but only on the immediate reward and an estimate of the next state. As learning steps go on, the TD error will become more stable than the actual rewards received during an episode, converging to the optimal value function defined by the Bellman equation. A simple understanding of TD learning could be stated using an intuitive example. While planning a car trip on a weekend, one could try to predict its duration. Moreover, a road traffic model with ongoing works could be available, considering everything that happened during the week. A naive approach would be to wait until the weekend to update the model based on all outcomes of previous days.

However, you could already have enough information on Friday to confirm your itinerary and plan your trip accordingly. That is why the update does not need to wait until the weekend since it can be done in an online manner. TD learning using the Q^π defines the $SARSA$[1] approach [113], which updates the action-value representation with the following rule:

$$Q^\pi(s_t, a_t) \longleftarrow Q^\pi(s_t, a_t) + \alpha[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)]$$

in which the γ parameter weights by how much we want to balance between previous estimates and the target value. This differs from the famous *Q-learning* algorithm, which is an off-policy variation of the idea:

$$\begin{aligned} Q^\pi(s_t, a_t) &\longleftarrow Q^\pi(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a \in \mathcal{A}} Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t)] \\ &= (1 - \alpha)\, Q^\pi(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a \in \mathcal{A}} Q^\pi(s_{t+1}, a)] \end{aligned}$$

This is off-policy, mainly because it disregards the actual trajectory, and updates based on what the result of the next greedy state-action pair could be, i.e. the best action using the maximum. The learned policy acts greedily, while the concrete policy used to interact with the environment and gather the actual trajectory could have been anything exploratory. Thanks to its smaller event horizon, the TD target displayed in both regular TD-learning and Q-learning has a lower variance than the return. Unfortunately, it is inherently biased because of the bootstrapping idea. *N-step bootstrap* is a way to unify Monte-Carlo methods with TD learning, by looking at an update based on n intermediate rewards. We can compare the actual n-step return computation with the two previous approaches. Now, the action-value function has time indexes that relate to when it was updated:

- Monte-Carlo: $G_\tau = r_{t+1} + \cdots + \gamma^{T-t-1} r_T$
- TD: $G_{t:t+1} = r_{t+1} + \gamma V_{t+1}(s_{t+1})$
- N-step: $G_{t:t+n} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n V_{t+n-1}(s_{t+n})$

N-step returns then function as estimates of the full return using bootstrapped values. In this case, 1-step returns are similar to TD learning, and ∞-step returns align with Monte-Carlo methods. In the end, it helps overcome the time constraints that exist in other designs. We can easily extend the update rule to this specific generated target and obtain the following:

$$V^\pi_{t+n}(s_t) \longleftarrow V^\pi_{t+n-1}(s_t) + \alpha[G_{t:t+n} - V^\pi_{t+n-1}(s_t)]$$

with $V^\pi_{t+n-1}(s) = V^\pi_{t+n}(s_t), \forall s \neq s_t$. Deriving an actual formula from SARSA is straightforward and follows the same pattern as the previous equation. However, performing the same derivation for Q-learning is not trivial, as it would no longer respect

[1] Acronym standing for *State-Action-Reward-State-Action*

several assumptions of off-policy learning. Indeed, we would have to re-establish the off-policy nature of the initial algorithm [113], mostly using importance sampling [117], for example. This overall definition would need an actual thorough discussion on the $TD(\lambda)$ paradigm, in which $G_t = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$. Previous methods and other ones can be derived from this expression. However, we avoid displaying the entirety of available possibilities since we practically implemented our target computation following the formula:

$$G_t = r_{t+1} + \gamma\,[\,r_{t+2} + \gamma\,[\,\cdots\,[\,r_{t+n} + \gamma\,Q^{\pi}_{t+n}(s_t, a_t)]] \tag{B.1}$$

and used these bootstrap estimates to fit a parameterized representation of Q^{π} in a supervised manner, as it works sufficiently well with MPO, the algorithm used in this thesis. Now that we have described the value estimate in the actor-critic setup, we need to understand the whole point of policy-based approaches better.

B.2 Looking at policy learning

Let us remind that the optimal policy is defined as $\pi_* = \arg\max_{\pi}\mathbb{E}_{\tau\sim P(.)}\left[\sum_t r(s_t, a_t)\right]$. We define a cost $\mathcal{J}$ with respect to π so that:

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim P(.)}\left[\sum_t r(s_t, a_t)\right] \approx \frac{1}{N}\sum_i\sum_t r(s_{i,t}, a_{i,t})$$

with N the number of trajectory samples. A first approach comes from the *REINFORCE*[2] algorithm [131] which uses the fact that $P(\tau)\nabla_{\pi}\,log\,P(\tau) = P(\tau)\frac{\nabla_{\pi}P(\tau)}{P(\tau)} = \nabla_{\pi}P(\tau)$, in order to express:

$$\begin{aligned}\mathcal{J}(\pi) &= \mathbb{E}_{\tau\sim P(.)}\left[\left(\sum_t \nabla_{\pi}\,log\,\pi(a_t|s_t)\right)\left(\sum_t \gamma\,r(s_t, a_t)\right)\right] \\ &\approx \frac{1}{N}\sum_i\left(\sum_t \nabla_{\pi}\,log\,\pi(a_t|s_t)\right)\left(\sum_t \gamma r(s_t, a_t)\right)\end{aligned}$$

with N entire trajectories sampled from P, following Monte-Carlo procedures depicted in the previous section. Finally, the policy is improved by gradient ascent with the update rule $\pi \longleftarrow \pi + \alpha\nabla_{\pi}\mathcal{J}(\pi)$. Actually, the cost $\mathcal{J}$ is close to a maximum likelihood formulation. Thanks to it, we can interpret the approach as a procedure to increase the probabilities of good trajectories and decrease the probabilities of bad ones. Nevertheless, this method shows high variance, leading to overestimating probabilities related to actions with low rewards. Said differently, for the same policy and state-action pair, the reward can be different. This is solved by the fact that π at time t' can not affect the reward at

[2]Common shorthand written in capital letters

time t if $t < t'$. Indeed, the future logically does not impact the past. So we can rewrite the gradient and approximate it by:

$$\nabla_\pi \mathcal{J}(\pi) \approx \frac{1}{N} \left[\sum_t \nabla_\pi \, log \, \pi(a_t|s_t) \left(\sum_{t' \geq t} \gamma \, r(s_{t'}, a_{t'}) - b \right) \right]$$

with b an unbiased estimator, such as the average reward. We subtract to the current reward $r(s_t, a_t)$, the mean return starting from state s_t, keeping the advantage taken out the chosen action a_t [77]. This overall scheme is on-policy because of the expectation over the entire trajectory τ. We can make it off-policy [30], but it does not solve entirely our main concerns regarding convergence. Indeed, $\nabla_\pi \mathcal{J}(\pi)$ has a low bias, but a high variance because of entire trajectories which might be long. Moreover, this signal can be really noisy, which would raise difficulties in defining proper learning rates. These clarifications have provided insights into the gradient-based methods, which could be useful for potential comparison with the actor-critic setup.

Precisions on WEST

C.1 A more precise representation of WEST geometry

In Chapter 2, a schematic view of a tokamak poloidal plane is presented. However, this general representation does not account for WEST particularities. Below, the real geometry of WEST is displayed (Figure C.1), as configured in the NICE environment. This more realistic figure includes all defined domains for NICE computation. The domains are similar, except that the vacuum vessel is treated differently:

- the iron structures Ω_f highlighted in green;
- the passive structures Ω_{ps} in burgundy, including the divertor structures, with the circular chamber walls hosting the vacuum chamber, and surrounded by toroidal coils (solid black lines). It differs from the D-shaped scheme in the general representation;
- the PF coils regions Ω_c are placed around the circular chamber (blue squares) to control plasma's shape, position, and current;
- the limiter region Γ_L occupies the entirety of the vacuum chamber;
- the plasma domain Ω_p, here displayed in light yellow.

C.2 A noisy description of delays

In chapter 3, we stated that PILOT incorporates noise injection in the observations, as well as delays both in the diagnostics and power supply models. The following table concisely lists these elements.

Power supply related Each of the bounds presented below is multiplied by 0.9 to ensure safe transfer to reality once deployment is performed.

	Delay (ms)	Min. current (A)	Max. Current (A)	Voltage limit (V)
A	0.002	-30000	35000	1400
B	0.003	-4300	5600	1400
D	0.003	-2000	2300	2500
E	0.003	-4000	1300	2500
F	0.003	-4000	1000	2500
X	0.003	-2000	20000	370

Diagnostics related For each diagnostic i, noise is sampled from a normal law $\mathcal{N}(0, \sigma_i)$. The following table presents the standard deviations for all observations of interest.

	σ	Delay (ms)
Coils currents	25	0.5
Loop voltage	0.3	
Flux loops	10^{-4}	
Magnetic probes	10^{-4}	
Temporal derivatives	0.05	
Coils currents	0.003	

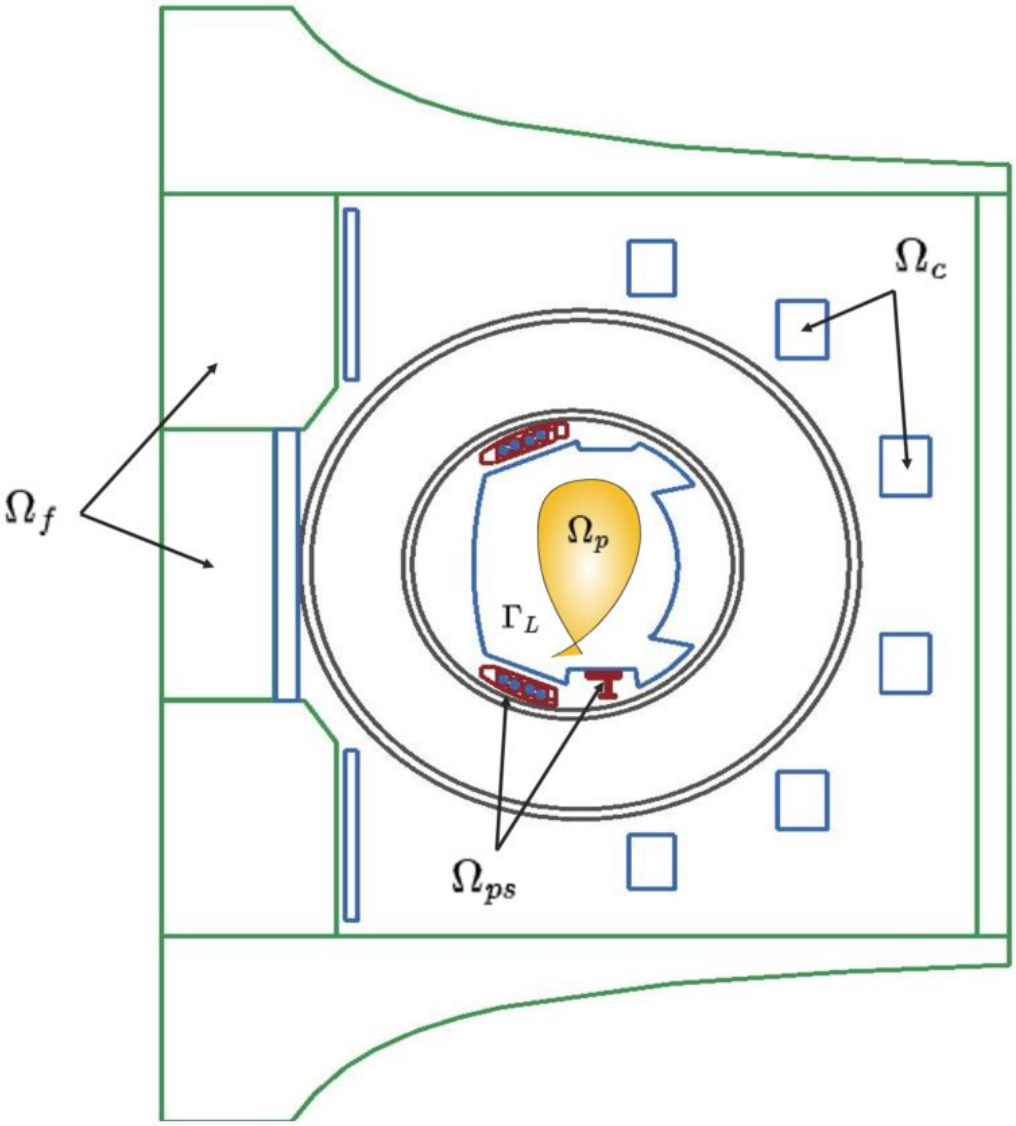


Figure C.1: Schematic view of the WEST poloidal plane, as opposed to the general representation presented in a previous chapter.

C.3 May the snapshot be with you

We present several snapshots used to build control scenarios. The shapes range from a plasma in limiter configuration, to an X-point one stabilized in the long run, through a plasma which just started to stretch. Any combination of the latter is possible, as well as the addition of new shapes to PILOT, under the condition that new transitions are physically accurate.

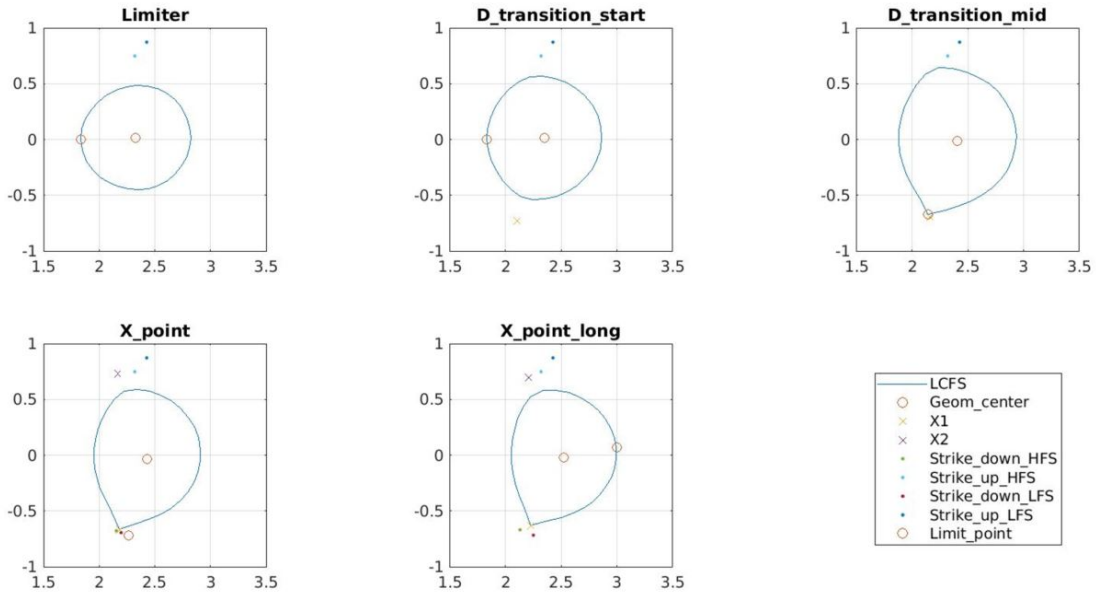


Figure C.2: Several snapshots are at the basis of our control scenarios. The *D_transition_start* increases elongation, and *D_transition_mid* forms an intermediate shape with an X-point. Hence, we can configure a vast number of potential scenarios.

Reward definitions

In Chapter 3, the reward computation is described, but no proper definition of its components is performed. This appendix presents all the reward components used in this PhD, as well as the combination performed for each of the four scenarios of interest. As a side note, the reward penalty related to terminal conditions is set to -5, which is equal to -0.5 once scaled properly.

D.1 Reward components

General purpose

LCFS Distance between the observed LCFS upsampled to 128 points, and the target boundary composed of 32 reference points. For each of the latter, its distance to the nearest segment made of observed LCFS points is computed, and expressed in meters. The 32 values are normalized through the procedure depicted in Chapter 3, leading to one scalar value.

R Distance to the reference radial coordinate of the plasma centroid, expressed in meters. This is computed from the real R which is interpolated for increased precision. Indeed, it would have followed nodes of the simulation mesh otherwise.

Z Distance to the vertical coordinate of the plasma centroid, expressed in meters. Interpolation is performed again.

I_p Plasma current expressed in amperes.

κ Plasma elongation, which is unitless. This is expressed by the height of the plasma, divided by its width.

a Minor radius, expressed in meters.

IC_{min} Minimization of the poloidal coils currents, expressed in amperes. We consider general good and bad parameters since a specific pair per coil showed no significant improvement over the current implementation.

X-points related

X_l Distance to the desired X-point location, expressed in meters. This is computed from the real x-point location, which is interpolated for increased precision. Indeed, it would have followed nodes of the simulation mesh otherwise.

X_n Normalized flux at the desired X-point location. Target set to 1. It forces the X-point to be on the LCFS, forming the configuration of interest.

X_f Gradient of the flux at the desired X-point location. Target set to 0, and expressed in $W.rad^{-1}.s^{-1}$. It helps form an X-point in the neighborhood of the target.

X_{ns} Normalized flux at the desired left and right strike points, located on the lower divertor. Target set to 1.

LP Coordinates of the actual limit point, either the limiter in corresponding scenarios or the X-point in the other case, expressed in meters.

D.2 Reward definition

The reward definition with all hyperparameters is described for each of the control scenarios used in this PhD in the following tables.

Limiter maintain

Component	Softplus		Smoothmax	Weight
	good	bad	α	
LCFS	0.001	0.025	-1	3
R	0.001	0.025	x	1
Z	0.001	0.025	x	1
a	0.002	0.02	x	1
I_p	1000	10000	x	1
κ	0.002	0.02	x	1
a	0.002	0.02	x	1
IC_{min}	100	10000	-0.5	1
Combined with $\alpha = -0.5$				

Limiter evolve

Component	Softplus		Smoothmax	Weight
	good	bad	α	
LCFS	0.001	0.03	-1	3
R	0.005	0.05	x	1
Z	0.005	0.05	x	1
a	0.002	0.025	x	1
I_p	1000	15000	x	2
κ	0.002	0.15	x	1
a	0.002	0.03	x	1
IC_{min}	100	10000	-0.5	1
Combined with $\alpha = -5$				

X-point maintain

Component	Softplus		Smoothmax	Weight
	good	bad	α	
LCFS	0.005	0.035	-1	3
R	0.002	0.025	x	1
Z	0.002	0.025	x	1
I_p	500	10000	x	2
a	0.002	0.03	x	1
κ	0.002	0.025	x	1
a	0.002	0.03	x	1
IC_{min}	100	10000	-0.5	1
X_l	0.005	0.03	x	1
X_n	0	0.08	x	1
X_f	0	3.5	x	0.5
LP	0.002	0.025	x	1
Combined with $\alpha = -0.5$				

X-point evolve

Component	Softplus		Smoothmax	Weight
	good	bad	α	
LCFS	0.004	0.04	-1	3
R	0.002	0.02	x	1
Z	0.002	0.015	x	2
a	0.02	0.5	x	0.5
I_p	500	20000	x	3
κ	0.005	0.2	x	1
a	0.002	0.035	x	1
IC_{min}	100	12000	-5	1
X_l	0.01	0.15	x	3
X_n	0	1	x	2
X_f	0	3.5	x	1
X_{ns}	0.1	0.8	-5	2
LP	0.1	0.25	x	1
Combined with $\alpha = -5$				

D.3 Curriculum definition

The curriculum is set to be a sequence of fixed rewards. It sequentially starts by controlling the plasma centroid and its current. Then, it adds the elongation and the LCFS. All defined X-point targets are included for the final task in X-point scenarios. At each step of the curriculum and for each scenario, the chosen hyperparameters are the same as those presented in the previous tables.

Training hyperparameters

This section describes all MPO hyperparameters and their related values. They follow the hyperparameter search described in Chapter 3.

E.1 The role of each hyperparameter

E.1.1 Generally for PILOT

A The number of actor nodes running in parallel.

SL Length of the sequences gathered in the replay buffer.

BL The stored sequences will be of length SL + |Burn-in length|. The added part is used to initialize the critic LSTM core.

RP Replay period. This allows the overlap of $SL - RP$ timesteps in consecutive sequences.

$\min_{RS}$ Minimum size of the replay buffer, which impacts the pace at which data is processed. Training starts when $\min_{RS}$ are in the replay buffer.

$\max_{RS}$ Maximum size of the replay buffer, with the same kind of impact stated right before.

SPI Samples per insert. If the environments are faster than the learner, data might be replaced too fast, and training might become unstable as the distribution of samples evolves too rapidly. If the learner is faster than the environments, we would have more samplings than writings. It is more stable but can lead to instabilities if we encounter several samples that are completely different from the average distribution. This hyperparameter then regulates the ratio between the number of examples sampled by the learner node and the number of writings happening in the replay buffer.

V The period after which actor nodes retrieve policy weights from the learner node. This can be set automatically to the maximum duration of an episode. If an episode ends before its full length, the counter keeps going until the update threshold is reached.

I Type of integral effect used either in the policy's input, or as a reward component. It can be the empirical average (Emp), a classic integral representation (Int), or an exponential decay (Exp).

E.1.2 Specifically for MPO

General

B Batch size.

γ Discount factor.

n N-step bootstrapping horizon.

T_π Number of online updates before target policy network update.

T_Q Number of online updates before target critic network update.

l_c Learning rate for the critic update in policy evaluation.

l_d Learning rate for the dual problem related to η in the E-step.

l_p Learning rate for the policy update in the M-step.

π_{Σ_0} Initial standard deviation of the Gaussian policies.

Policy improvement

m Number of sampled actions in the E-step.

ϵ Hard constraint for the E-step.

ϵ_μ Hard constraint for the M-step while fixing the mean of the policy.

ϵ_Σ Hard constraint for the M-step while fixing the covariance matrix of the policy.

$\log \eta_0$ Initial temperature in the E-step in log-space.

$\log \alpha_{\mu_0}$ Initial alpha in log-space for the inner optimization of the objective to fit Σ_θ in the M-step.

$\log \alpha_{\Sigma_0}$ Initial alpha in log-space for the inner optimization of the objective to fit μ_θ in the M-step.

ϵ_a Hard constraint for the action penalty summed to the overall policy.

E.2 Current configuration

PILOT		MPO			
		General parameter		Specific parameters	
A	100	B	256	m	20
SL	64	γ	0.99	ϵ	5×10^{-1}
BL	20	n	5	ϵ_μ	9.09
RP	10	T_π	50	ϵ_Σ	0.966
$\min_{RS}$	512	T_Q	50	$\log \eta_0$	10
$\max_{RS}$	10^6	l_p	3×10^{-4}	$\log \alpha_{\mu_0}$	10
SPI	256	l_d	3×10^{-4}	$\log \alpha_{\Sigma_0}$	1000
V	Adaptative	l_c	10^{-2}	ϵ_a	10^{-3}
I	Emp	π_{Σ_0}	0.5		

E.3 Neural architectures

Finally, we thoroughly describe the policy's fully connected structure and the recurrent critic. All weights are initialized randomly from a truncated normal distribution with a zero bias and scaled with the number of inputs. The last layer of the policy follows the same specification, but the weights are scaled with 10^{-4}. This is similar to [29], as [7] showed that the last policy layer should be initialized with weights a hundred times smaller than remaining layers, i.e. considering the Tensorflow implementation TensorFlow [1], it corresponds to a product by 10^{-4}. The policy network uses distributions for the 11 control coils used to operate WEST. Both architectures follow the state-of-the-art [29], as hyperparameter searches over the properties of the network lead to structures performing worse on average. Moreover, MPO tends to work best with networks at least as wide and deep as the ones presented in [38].

```
policy_network = Sequential([
        Linear(256),
        LayerNorm(),
        tf.nn.tanh,
        Linear(256),
        tf.nn.elu,
        Linear(256),
        tf.nn.elu,
        Linear(256),
        tf.nn.elu,
        MultivariateNormalDiagHead(11,init_scale=pi_Sigma_0)
        ])

critic_network = snt.Sequential([
        ClipActionToSpec(),
        tf.nn.tanh,
        ConcatObsAndAction(),
        LSTM(256),
        ResidualConcat() -> Output of LSTM concatened to its inputs
        Linear(256),
        tf.nn.elu,
        Linear(256),
        tf.nn.elu,
        Linear(1)
        ])
```

References

[1] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] A. Abdolmaleki, S. Huang, L. Hasenclever, et al. A distributional view on multi-objective policy optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11–22. PMLR, 2020.

[3] A. Abdolmaleki, B. Price, N. Lau, et al. Deriving and improving CMA-ES with information geometric trust regions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 657–664. Association for Computing Machinery, 2017.

[4] A. Abdolmaleki, J. T. Springenberg, J. Degrave, et al. Relative entropy regularized policy iteration. *CoRR*, abs/1812.02256, 2018.

[5] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, et al. Maximum a Posteriori Policy Optimisation. In *International Conference on Learning Representations*, 2018.

[6] R. Albanese, J. Blum, and O. Barbieri. On the solution of the magnetic flux equation in an infinite domain. In *EPS. 8th Europhysics Conference on Computing in Plasma Physics*, pages 41–44, 1986.

[7] M. Andrychowicz, A. Raichuk, P. Stańczyk, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021.

[8] K. H. Ang, G. Chong, and Y. Li. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.

[9] M. Ariola and A. Pironti. *Magnetic Control of Tokamak Plasmas*. Springer London, 2008.

[10] J.F. Artaud, F. Imbeaux, J. Garcia, et al. Metis: a fast integrated tokamak modelling tool for scenario design. *Nuclear Fusion*, 58(10):105001, aug 2018.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[12] R. Bellman. *Dynamic Programming*. Dover Publications, 1957.

[13] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 41–48. Association for Computing Machinery, 2009.

[14] D. Bertsekas. *A Course in Reinforcement Learning: 2nd Edition*. Athena Scientific, 2024.

[15] J. Blum. *Numerical simulation and optimal control in plasma physics*. New York, NY; John Wiley and Sons Inc., 1989.

[16] S. Borman. The expectation maximization algorithm a short tutorial, 2006.

[17] C. Bourdelle, J.F. Artaud, V. Basiuk, et al. WEST Physics Basis. *Nuclear Fusion*, 55(6), 2015.

[18] J. Bucalossi, J. Achard, O. Agullo, et al. Operating a full tungsten actively cooled tokamak: overview of WEST first phase of operation. *Nuclear Fusion*, 62(4):042007, 2022.

[19] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and Ivana Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018.

[20] F. Carpanese. *Development of free-boundary equilibrium and transport solvers for simulation and real-time interpretation of tokamak experiments*. PhD thesis, EPFL, 2021.

[21] A. Cassirer, G. Barth-Maron, E. Brevdo, et al. Reverb: A framework for experience replay, 2021.

[22] I. Char, J. Abbate, L. Bardoczi, et al. Offline model-based reinforcement learning for tokamak control. In Nikolai Matni, Manfred Morari, and George J. Pappas, editors, *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, pages 1357–1372. PMLR, 2023.

[23] P. Christiano, J. Leike, T. B. Brown, et al. Deep reinforcement learning from human preferences, 2023.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[25] G. Cunningham. High performance plasma vertical position control system for upgraded MAST. *Fusion Engineering and Design*, 88(12):3238–3247, 2013.

[26] W. M. Czarnecki, R. Pascanu, S. Osindero, et al. Distilling policy distillation, 2019.

[27] G. De Tommasi, S. Dubbioso, Y. Huang, et al. A RL-based vertical stabilization system for the EAST tokamak. In *2022 American Control Conference (ACC)*, 2022.

[28] P.C. de Vries and Y. Gribov. ITER breakdown and plasma initiation revisited. *Nuclear Fusion*, 59(9):096043, 2019.

[29] J. Degrave, F. Felici, J. Buchli, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602:414–419, 2022.

[30] T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, page 179–186, Madison, WI, USA, 2012. Omnipress.

[31] A. Doerr, C. Daniel, M. Schiegg, et al. Probabilistic recurrent state-space models, 2018.

[32] S. Dubbioso, G. De Tommasi, and A. Mele et al. A deep reinforcement learning approach for vertical stabilization of tokamak plasmas. *Fusion Engineering and Design*, 194, 2023.

[33] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark, 2022.

[34] B. Ehret, C. Henning, M. Cervera, et al. Continual learning in recurrent neural networks. In *International Conference on Learning Representations*, 2021.

[35] L. C. Evans. An introduction to mathematical optimal control theory version 0.2, 2013.

[36] B. Faugeras. An overview of the numerical methods for tokamak plasma equilibrium computation implemented in the nice code. *Fusion Engineering and Design*, 160:112020, 2020.

[37] W. Fedus, P. Ramachandran, R. Agarwal, et al. Revisiting fundamentals of experience replay. In *International conference on machine learning*, pages 3061–3071. PMLR, 2020.

[38] H. Furuta, T. Kozuno, T. Matsushima, Y. Matsuo, and S. (Shane) Gu. Co-adaptation of algorithmic and implementational innovations in inference-based deep reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9828–9842. Curran Associates, Inc., 2021.

[39] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8(5–6):359–483, 2015.

[40] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015.

[41] G. J. Gordon. Reinforcement learning with function approximation converges to a region. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[42] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks, 2017.

[43] Y. Gribov, A. Kavin, V. Lukash, et al. Plasma vertical stabilisation in ITER. *Nuclear Fusion*, 55(7), 2015.

[44] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[45] G. Gros, B. Faugeras, C. Boulbe, et al. Numerical simulation of tokamak plasma equilibrium evolution. Technical report, INRIA, 2024.

[46] D. Görges. Relations between model predictive control and reinforcement learning. *IFAC-PapersOnLine*, 50(1):4920–4928, 2017. 20th IFAC World Congress.

[47] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[48] T. Haarnoja, A. Zhou, P. Abbeel, and S. 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.

[49] D. Hafner, T. Lillicrap, I. Fischer, et al. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[50] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowe. Expressing arbitrary reward functions as potential-based advice. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), 2015.

[51] M. Henaff, M. Jiang, and R. Raileanu. A study of global and episodic bonuses for exploration in contextual MDPs. In *International Conference on Machine Learning*, pages 12972–12999. PMLR, 2023.

[52] H. Heumann. A Galerkin method for the weak formulation of current diffusion and force balance in tokamak plasmas. *Journal of Computational Physics*, 442, 2021.

[53] H. Heumann, J. Blum, C. Boulbe, et al. Quasi-static free-boundary equilibrium of toroidal plasma with CEDRES++: Computational methods and applications. *Journal of Plasma Physics*, 81(3), 2015.

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

[55] M. W. Hoffman, B. Shahriari, J. Aslanides, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2022.

[56] J. Hollenstein, S. Auddy, M. Saveriano, E. Renaudo, and J. Piater. Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *Transactions on Machine Learning Research*, 2022. Survey Certification.

[57] D. Humphreys, A. Kupresanin, M.D. Boyer, et al. Advancing fusion with machine learning research needs workshop report. *J Fusion Energ*, 39:123–155, 2020.

[58] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[59] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.

[60] S. Joung, Y.-C. Ghim, J. Kim, et al. GS-DeepNet: mastering tokamak plasma equilibria with deep neural networks and the Grad–Shafranov equation. *Scientific Reports*, 2023.

[61] S. Joung, J. Kim, S. Kwak, et al. Deep neural network Grad–Shafranov solver constrained with measured magnetic signals. *Nuclear Fusion*, 60(1):016034, December 2019.

[62] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, and R. Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

[63] S. Kerboua-Benlarbi, R. Nouailletas, B. Faugeras, E. Nardon, and P. Moreau. Magnetic control of west plasmas through deep reinforcement learning. *IEEE Transactions on Plasma Science*, 2024.

[64] D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] J. Kober, J. A. Bagnell, and J. Peters. *Reinforcement Learning in Robotics: A Survey*, pages 9–67. Springer International Publishing, 2014.

[66] V. Konda and J. Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[67] E.A. Lazarus, J.B. Lister, and G.H. Neilson. Control of the vertical instability in tokamaks. *Nuclear Fusion*, 30(1):111, 1990.

[68] N. Lehtomaki, N. Sandell, and M. Athans. Robustness results in linear-quadratic gaussian based multivariable control designs. *IEEE Transactions on Automatic Control*, 26(1):75–93, 1981.

[69] S. Levine. *Motor Skill Learning with Local Trajectory Methods*. PhD thesis, Stanford University, Stanford, CA, USA, 2014.

[70] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018.

[71] F. L. Lewis, D. L. Vrabie, and K. G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems*, 32:76–105, 2012.

[72] T. Lin, Y. Wang, X. Liu, and X. Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022.

[73] M. Liu, H. Zhao, Z. Yang, et al. Curriculum offline imitating learning. *Advances in Neural Information Processing Systems*, 34:6266–6277, 2021.

[74] Y. Lu and J. Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3094–3105. Curran Associates, Inc., 2020.

[75] P. MacAlpine and P. Stone. Overlapping layered learning. *Artificial Intelligence*, 254:21–43, 2018.

[76] W. Mcculloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[77] J. Mei, W. Chung, V. Thomas, et al. The role of baselines in policy gradient optimization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[78] A. Mele, R. Albanese, R. Ambrosino, et al. MIMO shape control at the EAST tokamak: Simulations and experiments. *Fusion Engineering and Design*, 146:1282–1285, 2019.

[79] V. Mnih, A. P. Badia, M. Mirza, et al. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[80] V. Mnih, K. Kavukcuoglu, D. Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[81] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. *CoRR*, abs/1606.02647, 2016.

[82] S. Narvekar, B. Peng, M. Leonetti, et al. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[83] S. Narvekar and P. Stone. Learning curriculum policies for reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 25–33, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

[84] T. Ni, B. Eysenbach, and R. Salakhutdinov. Recurrent model-free RL is a strong baseline for many pomdps. *CoRR*, abs/2110.05038, 2021.

[85] R. Nouailletas, P. Moreau, B. Santraine, et al. West plasma control system status. *Fusion Engineering and Design*, 192:113582, 2023.

[86] J. Oh, Y. Guo, S. Singh, and H. Lee. Self-imitation learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3878–3887. PMLR, 2018.

[87] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[88] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[89] F. Pesamosca, F. Felici, S. Coda, C. Galperti, and the TCV Team. Improved plasma vertical position control on TCV using model-based optimized controller synthesis. *Fusion Science and Technology*, 78(6):427–448, 2022.

[90] P. Piovesan, J. M. Hanson, P. Martin, et al. Tokamak operation with safety factor q(95) < 2 via control of mhd stability. *Physical review letters*, 113:045003, 2014.

[91] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey, 2020.

[92] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[93] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. Experience replay for continual learning. *CoRR*, abs/1811.11682, 2018.

[94] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[95] W. Rui, Y. Wang, H. Song, et al. Using LQR controller for vertical position control on EAST. *Nuclear Fusion*, 64(6):066040, 2024.

[96] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[97] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, et al. Policy distillation, 2016.

[98] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, et al. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[99] A. A. Rusu, M. Večerík, T. Rothörl, et al. Sim-to-real robot learning from pixels with progressive nets. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 262–270. PMLR, 2017.

[100] O. Sauter, C. Angioni, and Y. R. Lin-Liu. Neoclassical conductivity and bootstrap current formulas for general axisymmetric equilibria and arbitrary collisionality regime. *Physics of Plasmas*, 6:2834–2839, 1999.

[101] O. Sauter and S. Yu. Medvedev. Tokamak coordinate conventions: *COCOS. Comput. Phys. Commun.*, 184(2):293–302, 2013.

[102] A. M. Schäfer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In S. D. Kollias, A. Stafylopatis, W. Duch, and E. Oja, editors, *Artificial Neural Networks – ICANN 2006*, pages 632–640, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[103] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 2015. PMLR.

[104] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[105] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[106] J. Seo, S. Kim, A. Jalalvand, et al. Avoiding fusion plasma tearing instability with deep reinforcement learning. *Nature*, 626:746–751, 2024.

[107] J. Seo, Y.-S. Na, B. Kim, et al. Feedforward beta control in the KSTAR tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10), 2021.

[108] A. Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.

[109] D. Silver, A. Huang, C. J. Maddison, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[110] S.Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

[111] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

[112] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time evolution in the NERO video game (winner of CIG 2005 best paper award). In *IEEE Conference on Computational Intelligence and Games*, 2005.

[113] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[114] V. Tangkaratt, A. Abdolmaleki, and M. Sugiyama. Guide actor-critic for continuous control. *arXiv preprint arXiv:1705.07606*, 2017.

[115] Y. Tassa, Y. Doron, A. Muldal, et al. DeepMind control suite, 2018.

[116] A. A. Taïga, W. Fedus, M. C. Machado, A. Courville, and M. G. Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment, 2021.

[117] S. T. Tokdar and R. E. Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 2010.

[118] B. D. Tracey, A. Michi, Y. Chervonyi, et al. Towards practical reinforcement learning for tokamak magnetic control. *ArXiv*, abs/2307.11546, 2023.

[119] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter. Survey on reinforcement learning for language processing. *Artificial Intelligence Review*, 56(2):1543–1575, 2023.

[120] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017.

[121] O. Vinyals, I. Babuschkin, W. M. Czarnecki, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019.

[122] T. Wakatsuki, T. Suzuki, N. Hayashi, N. Oyama, and S. Ide. Safety factor profile control with reduced central solenoid flux consumption during plasma current ramp-up phase using a reinforcement learning technique. *Nuclear Fusion*, 59(6):066022, 2019.

[123] T. Wakatsuki, T. Suzuki, N. Oyama, and N. Hayashi. Ion temperature gradient control using reinforcement learning technique. *Nuclear Fusion*, 61(4):046036, 2021.

[124] M. L. Walker, P. de Vries, F. Felici, and E. Schuster. Introduction to tokamak plasma control. In *2020 American Control Conference (ACC)*, pages 2901–2918, 2020.

[125] M. L. Walker and D. A. Humphreys. On feedback stabilization of the tokamak plasma vertical instability. *Automatica*, 45(3):665–674, 2009.

[126] L. Wang, X. Zhang, H. Su, and J. Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

[127] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer. Recent advances in bayesian optimization. *ACM Computing Surveys*, 55(13s):1–36, 2023.

[128] Z. Wang, A. Novikov, K. Zolna, et al. Critic regularized regression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7768–7778. Curran Associates, Inc., 2020.

[129] G. I. Webb. *Bayes Rule*, pages 74–75. Springer US, Boston, MA, 2010.

[130] J. Wesson. Tokamaks 3rd edition. *Journal of Plasma Physics*, 71(3):377–377, 2004.

[131] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, 1992.

[132] M. Wolczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś. Disentangling transfer in continual reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[133] Y. Wu and Y. Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2017.

[134] F. Yang, G. Barth-Maron, P. Stańczyk, et al. Launchpad: A programming model for distributed machine learning research, 2021.

[135] Z. Yang and H. Nguyen. Recurrent off-policy baselines for memory-based continuous control. *CoRR*, abs/2110.12628, 2021.

[136] J. Zhang, T. He, S. Sra, and A. Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[137] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13344–13362, 2023.

[138] M. Zimmer, Y. Boniface, and A. Dutech. Developmental reinforcement learning through sensorimotor space enlargement. In *2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2018.