Model-based reinforcement learning for predictive control and optimisation

Filippo Airaldi

Model-based reinforcement learning for predictive control and optimisation

Model-based reinforcement learning for predictive control and optimisation

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof. dr. ir. H. Bijl,
chair of the Board for Doctorates
to be defended publicly on
Monday 11, May 2026 at 15:00

by

Filippo AIRALDI

This dissertation has been approved by the promotor and the copromotor.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. B. De Schutter,	Delft University of Technology, *promotor*
Dr. A. Dabiri,	Delft University of Technology, *copromotor*

Independent members:

Prof. dr. R. Babuška	Delft University of Technology
Prof. dr. T. Keviczky	Delft University of Technology
Prof. dr. S. N. Gros	NTNU, Norway
Prof. dr. M. N. Zeilinger	ETH Zürich, Switzerland
Dr. ir. S. Haesaert	Eindhoven University of Technology

The work in this thesis has been partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101018826 - CLariNet)

Keywords: Model-based Reinforcement Learning, Model Predictive Control, Safe Control, Global Optimisation
Printed by: Proefschriftenprinten.nl
Cover by: Filippo Airaldi

We live after the conquering of time, but before its resurrection.
In the meantime, make some time for time to control you. Be bored, miss out, fall behind,
feel time passing, lose track of time, while experiencing only what your body can alone.
You might not have the time of your life, but you'll have more time of your life.

Michael Stevens

Contents

Summary

In the current age of emerging autonomous and artificial-intelligence-driven machines, sequential decision making constitutes one of the theoretical fundaments at the core of intelligent agency. As these systems are increasingly deployed in real-world engineering applications (e.g., autonomous vehicles and drones, as well as smart energy grids and greenhouses), there is a growing need for the control architectures governing these agents to meet, aside from traditional performance requirements, also interpretability and safety criteria, while encouraging adaptability and scalability. Classical model-based methodologies, such as Model Predictive Control (MPC), can in general provide rigorous frameworks that integrate a priori knowledge (e.g., via explicit, though often approximate, prediction models) and can handle constraints to enforce safety, yet their performance is tightly coupled with the accuracy of the underlying model and expert manual tuning of its parameters. Conversely, purely model-free approaches, such as deep Reinforcement Learning (RL), offer remarkable data-driven adaptation, but often lack interpretability and reliable constraint handling required to provide formal guarantees.

This work expands on the current state-of-the-art results that combine these two distinct approaches into a single framework. While not always straightforward, it is well known that endowing these decision-making processes with model-based knowledge can not only enhance their performance but also benefit their interpretability and analysis: model-based RL, also known as Approximate Dynamic Programming (ADP), is perhaps the most renowned machine learning paradigm to craft these intelligent predictive agents. This dissertation aims to look at RL from a different perspective. Instead of as an alternative to model-based control, RL is used as a performance-enhancing mechanism operating within rigorously defined safety requirements. Concurrently, this thesis establishes MPC as a unifying and scalable foundation block for learning-based control and optimisation for constrained, uncertain, and distributed decision-making systems.

Part I of this dissertation explores how optimal control can benefit from model-based RL. At its core is the combination of RL with MPC, where the latter is used as function approximation scheme of the former. In Chapters 2 and 3, new ways to promote safety in MPC-based RL are proposed, where safety is intended as the ability of a control policy to satisfy given constraints, usually on state variables. The underlying issue is that guaranteeing safe behaviour during training and at deployment is generally challenging, if possible at all. In these chapters, probabilistic guarantees on safety are achieved in two different ways: in Chapter 2, the RL search space of available MPC policies is restricted to those that, according to collected observations, are likely to be safe; in Chapter 3, probabilistic Control Barrier Functions (CBFs) are leveraged to create an MPC policy that is by construction safe up to some probability, and then RL is applied on top of this safe

controller to improve its closed-loop performance. Chapter 4 extends the application of MPC-RL to the multi-agent setting. To avoid the need for a centralised unit, MPC computations are distributed in a computationally efficient way onto each local agent, while ensuring that inter-agent communication is handled privately. Q-learning, a widely adopted RL algorithm, is cast into this scheme and, under some assumptions, it is shown that the proposed distributed learning scheme converges to the centralised version. Lastly, Chapters 5 and 6 broaden the range of real-world applications where the MPC-RL framework has been applied. Chapter 5 tackles the greenhouse climate control problem of adjusting the climate inside a greenhouse to maximise the crop economic yield and, at the same time, minimise control effort and satisfy as much as possible system constraints despite the presence of weather prediction uncertainty. Chapter 6 focuses instead on highway traffic management in the form of ramp metering control. The problem is similarly defined: the controller must regulate the inflow of vehicles at a highway ramp so as to avoid congestion on the highway as well as long queues at the ramp, while minimising control effort. Both chapters show promising results, with MPC-based RL ranking as the best performing methodology under several metrics in comparison to other state-of-the-art traditional and learning-based approaches.

In Part II, also gradient-free global optimisation is cast as a sequential decision making process, and is enhanced via model-based RL techniques. Chapter 7 proposes to adapt two well-known ADP methodologies, rollout and multi-step scenario-based optimisation, to the case in which the global optimisation surrogate model is deterministic and either based on inverse distance weighting or radial basis functions. The benefits of the proposed approaches combine the computational efficiency of these deterministic models with the nonmyopic lookahead behaviour of ADP techniques.

In summary, this dissertation attempts to answer the overarching question of how model-based RL can benefit predictive control and optimisation. It corroborates that safety and learning in RL are not necessarily incompatible, and that by leveraging MPC it is possible to promote safe control policies. It confirms that an appropriate choice of the parametrisation of the MPC controller, though not always intuitive, is key to effective and safe learning. MPC-RL is also shown to consistently outperform both pure model-based and pure learning-based approaches across different applied, uncertain, and multi-agent problems, and it can also be effectively extended to other domains, e.g., global optimisation. Despite these advances, the proposed methodologies face certain limitations. Generally, high computation loads and sensitivity to hyperparameter selection (e.g., the choice of parametrisation, which is problem-specific and not universally applicable without suitable modifications) are challenges that hinder reliability and wider adoption. For these reasons, applicability of these methodologies to real-world problems remains limited. Future work may mitigate these issues by 1) a more thorough integration of system identification and/or CBFs within MPC-RL to achieve harder, broader guarantees on safety; 2) expansion to wider classes of nonlinear multi-agent settings, coupled with more advanced RL methodologies, e.g., policy-based methods; 3) testing the proposed adaptive control frameworks in high-fidelity simulators with larger, more complex scenarios that better mimic real-world conditions; 4) and a formal investigation of convergence properties and extensions (e.g., to also handle discrete optimisation variables) for the proposed nonmyopic global optimisation methods.

Samenvatting

In het huidige tijdperk van opkomende autonome agenten en door kunstmatige intelligentie aangestuurde machines vormt sequentiële besluitvorming een van de belangrijkste theoretische fundamenten van intelligente systemen. Naarmate deze systemen steeds vaker worden ingezet in praktische technische toepassingen (zoals autonome voertuigen en drones, maar ook slimme energienetten en kassen), groeit de behoefte aan besturingsarchitecturen die, naast traditionele prestatie-eisen, ook voldoen aan interpretabiliteits- en veiligheidscriteria, terwijl ze aanpasbaarheid en schaalbaarheid bevorderen. Klassieke modelgebaseerde methodologieën, zoals Model Predictive Control (MPC), kunnen over het algemeen rigoureuze raamwerken bieden die a priori kennis integreren (via expliciete, zij het vaak benaderende, voorspellingsmodellen) en beperkingen kunnen hanteren om de veiligheid te waarborgen. Hun prestaties zijn echter sterk gekoppeld aan de nauwkeurigheid van het onderliggende model en de handmatige afstemming van de parameters door experts. Omgekeerd bieden puur modelvrije benaderingen, zoals deep Reinforcement Learning (RL), opmerkelijke datagestuurde aanpasbaarheid, maar missen ze vaak de interpreteerbaarheid en betrouwbare omgang met beperkingen die nodig zijn voor formele garanties.

Dit werk bouwt voort op de huidige stand van de techniek door deze twee verschillende benaderingen te combineren in één raamwerk. Hoewel dit niet altijd eenvoudig is, is het algemeen bekend dat het voorzien van deze besluitvormingsprocessen van modelgebaseerde kennis niet alleen hun prestaties kan verbeteren, maar ook hun interpreteerbaarheid en formele analyse vergemakkelijkt. Modelgebaseerd Reinforcement Learning (RL), ook wel benaderende Dynamische Programmering (ADP) genoemd, is wellicht het bekendste machine learning-paradigma om dergelijke intelligente voorspellende agenten te ontwikkelen en te trainen. Dit proefschrift onderzoekt RL vanuit een ander perspectief. In plaats van als alternatief voor modelgebaseerde regeltechniek, wordt RL ingezet als prestatieverbeterend mechanisme dat opereert binnen strikt gedefinieerde veiligheidseisen. Tegelijkertijd vestigt dit proefschrift MPC als een verenigend en schaalbaar fundament voor leergebaseerde regeltechniek en optimalisatie voor beperkte, onzekere en gedistribueerde besluitvormingssystemen.

Deel I van dit proefschrift onderzoekt hoe optimale regeltechniek kan profiteren van modelgebaseerde RL. Het hoofdonderwerp draait met name om de combinatie van RL met MPC, waarbij MPC wordt ingezet als functiebenaderingsmethode voor RL. In Hoofdstuken 2 en 3 worden nieuwe methoden voorgesteld om de veiligheid in MPC-gebaseerde RL te bevorderen, waarbij veiligheid wordt opgevat als het vermogen van een regelbeleid om aan bepaalde beperkingen te voldoen, meestal opgelegd aan toestandsvariabelen. Het onderliggende probleem is dat het waarborgen van veilig gedrag tijdens training en implementatie doorgaans een uitdaging vormt, als het al mogelijk is. In deze hoofdstukken worden probabilistische garanties op veiligheid op

twee verschillende manieren bereikt: in Hoofdstuk 2 wordt de RL-zoekruimte van beschikbare MPC-beleidsregels beperkt tot die beleidsregels die, op basis van verzamelde observaties, waarschijnlijk veilig zijn; in Hoofdstuk 3 worden probabilistische Control Barrier Functions (CBFs) gebruikt om een MPC-beleid te ontwerpen dat, tot op zekere hoogte, probabilistisch veilig is, waarna RL wordt toegepast op deze veilige controller om de gesloten-lusprestaties verder te verbeteren. Hoofdstuk 4 breidt de toepassing van dit MPC-gebaseerde RL-schema uit naar de multi-agentomgeving. Om de noodzaak van een gecentraliseerde eenheid te vermijden, worden MPC-berekeningen op een computationeel efficiënte manier verdeeld over de lokale agenten, terwijl de communicatie tussen agenten privé wordt gehouden. Q-learning, een veelgebruikt RL-algoritme, wordt binnen dit kader toegepast, en onder bepaalde aannames wordt aangetoond dat het voorgestelde gedistribueerde leerschema convergeert naar de gecentraliseerde oplossing. Ten slotte breiden Hoofdstuken 5 en 6 het toepassingsgebied van dit MPC-gebaseerde RL-raamwerk uit naar praktijkgerichte toepassingen. Hoofdstuk 5 richt zich op klimaatregeling in kassen: de regelaar past het binnenklimaat aan om de economische opbrengst van het gewas te maximaliseren en tegelijkertijd de regelinspanning te minimaliseren en zoveel mogelijk te voldoen aan systeembeperkingen, ondanks voorspellingsonzekerheden. Hoofdstuk 6 richt zich daarentegen op snelwegverkeersbeheer, in de vorm van ramp metering control. Het probleem wordt op vergelijkbare wijze gedefinieerd: de regelaar moet de instroom van voertuigen via de oprit reguleren om congestie op de snelweg en wachtrijen op de oprit te vermijden, terwijl de regelinspanning tot een minimum wordt beperkt. Beide hoofdstukken laten veelbelovende resultaten zien, waarbij MPC-gebaseerde RL volgens verschillende maatstaven beter presteert dan andere geavanceerde traditionele en leeralgoritmische benaderingen.

In Deel II wordt ook gradiëntvrije globale optimalisatie voorgesteld als een sequentieel besluitvormingsproces dat wordt versterkt met modelgebaseerde RL-technieken. Meer specifiek stelt Hoofdstuk 7 voor om twee bekende methoden uit de benaderende dynamische programmering — rollout en meerstaps scenario-gebaseerde optimalisatie — aan te passen aan het geval waarin het surrogaatmodel voor globale optimalisatie deterministisch is en gebaseerd op inverse afstandsweging of radiale basisfuncties. De voorgestelde methoden combineren de computationele efficiëntie van deze deterministische modellen met het langetermijn-vooruitziende gedrag van rollout- en meerstapsoptimalisatietechnieken.

Samenvattend probeert dit proefschrift een antwoord te geven op de overkoepelende vraag hoe modelgebaseerde RL voorspellende regeltechniek en optimalisatie kan versterken. In Deel I biedt het, vanuit het perspectief van MPC, oplossingen om uitdagingen rond veiligheid en schaalbaarheid aan te pakken, en breidt het de toepassingsmogelijkheden van MPC-gebaseerde RL uit. In de gradiëntvrije globale optimalisatiecontext van Deel II onderzoekt het hoe computationeel efficiënte, vooruitkijkende acquisitiestrategieën kunnen worden ontworpen. Het bevestigt dat veiligheid en leren in RL niet per se onverenigbaar zijn, en dat het door gebruik te maken van MPC mogelijk is om veilige regelstrategieën te bevorderen. Het bevestigt eveneens dat een geschikte keuze van de parametrisering van de MPC-regelaar, hoewel niet altijd intuïtief, cruciaal is voor effectief en veilig leren. MPC-RL presteert consistent beter dan zowel puur modelgebaseerde als puur leergebaseerde benaderingen bij verschillende toegepaste, onzekere en

multi-agentproblemen, en kan ook effectief worden uitgebreid naar andere domeinen, zoals globale optimalisatie. Ondanks deze vooruitgang kennen de voorgestelde methodologieën bepaalde beperkingen. Over het algemeen vormen hoge rekenlasten en gevoeligheid voor de keuze van hyperparameters (zoals de keuze van de parametrisering, die probleemspecifiek is en niet zonder geschikte aanpassingen universeel toepasbaar) uitdagingen die de betrouwbaarheid en bredere toepassing belemmeren. Om deze redenen blijft de toepasbaarheid van deze methodologieën op praktijkproblemen beperkt. Toekomstig onderzoek kan deze problemen verhelpen door 1) een grondigere integratie van systeemidentificatie en/of CBF's binnen MPC-RL om sterkere en bredere veiligheidsgaranties te bereiken; 2) uitbreiding naar bredere klassen van niet-lineaire multi-agentomgevingen, in combinatie met meer geavanceerde RL-methodologieën, zoals op beleid gebaseerde methoden; 3) het testen van de voorgestelde adaptieve regelkaders in hoogwaardige simulatoren met grotere, complexere scenario's die de omstandigheden in de praktijk beter nabootsen; 4) en een formeel onderzoek naar convergentie-eigenschappen en uitbreidingen (bijvoorbeeld om ook discrete optimisatievariabelen te kunnen verwerken) voor de voorgestelde niet-myopische globale optimalisatiemethoden.

1

Introduction

1.1. Background

Systems theory defines a dynamical system as a set of input-output trajectories unfolding along a time axis. By introducing states, auxiliary variables that split the past and future of the system's behaviour and capture its internal memory, we obtain a mathematical model that can predict how future control inputs will steer the system [223]. In other words, at every continuous or discrete time instant, given the current state of a system, the (possibly infinite) history of inputs and outputs can be forgotten and the evolution of future states can be predicted (deterministically or probabilistically) from the current state and impending inputs alone.

Many engineering systems, such as autonomous vehicles, drones, or greenhouses, fit naturally into this description, as they can be characterised as dynamical processes. Optimal control then asks the question [118]: *how can a sequence of control inputs be synthesised so as to realise a desired objective as well as possible while satisfying constraints imposed by, e.g., physical limitations or safety requirements?* Autonomous driving [107, 231], flight control [103, 187], or climate regulation [78, 147, 215] are but a few examples of optimal control tasks that differ only in the plant's dynamics, performance objective, and constraints. As these applications gain widespread adoption, the focus in the state of the art has been shifting towards control architectures that are not only high-performing but also interpretable, safe, and capable of adapting to changing conditions or of scaling to complex, multi-agent networks.

To tackle these optimal control problems with such stringent requirements, among other approaches, there exist two methodological (and somewhat antithetical) pillars that dominate the current state of the art, namely, Model Predictive Control and Reinforcement Learning. The former provides a rigorous framework for predictive decision-making with explicit constraint handling, yet the performance of its policies remains fundamentally coupled with the quality of a priori prediction models, and the necessity for expert manual tuning of its model parameters and objective function. Conversely, the latter, specifically in its model-free variant, achieves remarkable data-driven capabilities that allow it to learn optimal control behaviours just from interactions with the real system, but its black-box nature and lack of inherent formal guarantees hinder its applicability to real-world safety-critical scenarios. This dissertation sits at the intersec-

tion of these two methodologies, where new integrated approaches are developed that leverage the benefits of one to mitigate the shortcoming of the other. In particular, this work uses Reinforcement Learning as a tool to automatically enhance the operational performance of a model-based controller, while at the same time Model Predictive Control is leveraged as a foundation framework to formulate safe, scalable, learning-based decision-making systems for control and optimisation.

In what follows, background information on the current state of the art is provided. Then, the current challenges and the open research questions addressed by this dissertation are discussed.

1.1.1. Model predictive control

Model Predictive Control (MPC) [138, 163] is a well-known and widely adopted control methodology, both in literature and in industry. It is a predictive model-based approach, i.e., it leverages a prediction model (which can either match the real system's dynamics, or be an approximation of it) to predict over a finite-horizon window the evolution of states starting from the current system's state, and chooses an optimal sequence of actions along the same horizon that optimises the predefined objective and satisfy the given constraints. MPC does so by solving an optimisation problem, either analytically or, much more often, numerically. Typically, the MPC policy is carried out in a closed-loop, receding horizon fashion: only the first optimal action is actually applied to the real system while the rest is discarded, and the optimisation is repeated again at the next time step given the new system's state. This algorithm is repeated until the end of the control task, if any.

The appeal of the MPC framework stems from its three core components. 1) Because it explicitly propagates the dynamics through a prediction model of the plant, it results in a nonmyopic control policy that acts proactively rather than reactively; this aspect also allows for the injection of first-principles or domain-expert knowledge on the behaviour of the system into the prediction model in a seamless way. 2) The receding-horizon optimisation-based nature of the framework also guarantees, by construction, that the first control action belongs to a sequence that optimises the prescribed cost so that the policy is intrinsically optimal, at least over the chosen horizon window and in the absence of model errors and disturbances. 3) Moreover, MPC allows to explicitly and systematically handle constraints on inputs and/or states along its prediction horizon, which can even be further expressed into robust or probabilistic formulations that can account for disturbances or mismatches in the prediction model [17, 140]. These constraints are often used to represent and impose, among others, physical limitations of the actuation systems (e.g., the torque should not exceed the maximum value an engine can deliver), operational requirements (e.g., the maximum temperature in a chemical reaction should not be higher than a threshold to maintain good yield), and safety conditions (e.g., the position of an autonomous vehicle should be within the given lane).

On the other hand, the MPC methodology comes with its own drawbacks. Similarly to other model-based approaches, the closed-loop performance and safety guarantees of an MPC controller are highly dependent on the accuracy of the prediction model. Mismatches between the prediction model and the real plant's dynamics are commonly

present because real-world phenomena are often too complex to capture in a mathematical model in their entirety. In most cases, engineers and experts prefer to settle for simplified but manageable representations of the systems to be controlled. Furthermore, while more detailed models can lead to better prediction accuracy, they are often more numerically complex and lead to a significant increase in computational complexity. As aforementioned, when the prediction accuracy is limited or there are dynamical effects that have not been accounted for in the model, one can turn to robust or stochastic formulations that try to strengthen safety conditions against the presence of unknown disturbances in the dynamics. However, again, the numerical complexity of these approaches can be limited and their tractability poor, often requiring heavy approximations when deployed to realistic systems. Lastly, the objective function of the MPC optimisation scheme also plays a vital role in shaping the closed-loop performance of the controller. When the process is complex and the overarching control objectives are many and difficult to express in mathematical terms, the specification of the MPC objective function can become a non-trivial, daunting task.

1.1.2. Model-free reinforcement learning

The control community has been recently investigating ways to craft and tune controllers directly from data, thus limiting the amount of a-priori expert knowledge required and enabling automatic adaptivity [92, 165, 170, 177]. Reinforcement Learning (RL) [197] is perhaps one of the most well-known approaches, popularised by its successes in playing Atari games, chess, and Go at a super-human level [144, 179, 181]. Together with supervised and unsupervised learning, it forms the basic paradigms in Machine Learning (ML).

Similarly to MPC, the goal of RL is to find a control policy that minimises the desired cost function.[1] In its model-free variant (where no a priori knowledge about the system model is provided to the learning algorithm), an RL agent learns how to behave solely by interacting with the environment and receiving feedback on its performance via a reward signal. This signal has the crucial role of evaluating the quality of the agent's action in the current state, relative to the control objective. In other words, just like MPC, at each time step the RL agent is given the current state of the system and must provide the action it deems to be optimal to be applied to the system. The action is then implemented, and the reward feedback is provided to the agent. From these transitional data points, RL algorithms have been devised to extrapolate the optimal action to take in any given state.

Modern RL algorithms almost invariably rely on function approximation schemes in order to scale beyond small systems with tabular state spaces. The idea is to approximate the policy with a generic function from the state space to the action space, and then to adapt this approximation to the observed data. Linear parametrisations were an early choice [21, 37] but have nowadays been outclassed by nonlinear approximators, most prominently deep neural networks, giving rise to the field of Deep Reinforcement Learning (DRL) [12]. Compared to classic MPC, the advantages of such data-driven

[1] More often than not, instead of "costs", the RL community prefers the term "rewards" (i.e., the opposite of cost), which must be then maximised instead of minimised. The two terms are used interchangeably in this dissertation.

approaches are several. The absence of an explicit prediction model often makes the policy architecture comparatively simple (e.g., a multilayer perceptron whose topology exclusively consists of stacked layers of neurons), requiring no or little a priori information or knowledge on the plant to be controlled. In principle, the only hand-crafted component is the reward signal that encodes high-level performance objectives; once this is specified, the optimisation is handled automatically by the learning algorithm. RL also naturally enjoys online adaptivity capabilities, since the policy is learnt from interactions with the true system. This is in contrast with more traditional control approaches (including MPC), where the controller is designed offline once and then deployed to the system. Without any further automatic adaptation scheme in place, in case the plant dynamics were to change (e.g., due to a fault or an unexpected event), the controller would often be unable to adapt to the new conditions without manual intervention. Moreover, the same RL algorithm is readily applicable to a wide spectrum of control problems with minimum changes, an aspect that has also been empowered by the generalisation capabilities of deep neural networks.

Nevertheless, the model-free approximation-based stance of DRL suffers from some shortcomings. Policies represented by function approximation schemes, possibly neural network-based, lack interpretability due to their black box nature, making it difficult to predict their behaviour outside training scenarios or to obtain insights on the inner workings [74]. Likewise, standard RL offers few guarantees on constraint satisfaction at deployment (only in the limit of an infinite amount of training data), and none during the learning phase, making it less suitable for safety-critical problems [32, 70]. Convergence proofs are also challenging to formulate, often being tractable only for the tabular case. Additionally, RL algorithms are often sample-inefficient, requiring millions of interaction steps to learn a satisfactory policy, which is prohibitive whenever real-world data is expensive, slow, or risky to obtain. This is an intuitive drawback of any model-free algorithm, since the algorithm is given no prior knowledge and must learn everything from data.

1.1.3. Model-based reinforcement learning

Recent literature has seen the development of hybrid methodologies that aim to combat the shortcomings of MPC and RL, respectively [32, 92]. Among others, adaptive MPC formulations have been proposed that can adjust part of the controller parametrisation in response to the observed data and closed-loop behaviour [33, 56, 108, 191]. These adaptive capabilities have also been extended to a probabilistic setting where, e.g., the dynamics of the prediction model are adjusted online via Gaussian process regression [91, 109, 211]. MPC has also been coupled with global optimisation techniques (in particular, Bayesian optimisation) in an effort to automate the tuning process of the controller's parameters [65, 127, 161, 188]. On the RL side, model-based approaches are broadly defined as the integration of dynamics model learning together with planning, within a single framework or algorithm [145]. While model-free RL is typically a more popular choice, the advantages of model-based RL include higher sample efficiency and, depending on the choice of model representation, interpretability of the learnt policy.

Model predictive control for function approximation in reinforcement learning

In the context of model-based RL, recently [75] proposed and justified the use of MPC as function approximation scheme in RL. Figure 1.1 depicts a schematic overview of said approach. In such a scheme, a parametric MPC controller acts both as value function approximator (estimating "how good" it is to be in a given state) and policy provider (picking an action based on the current state). At the same time, an RL algorithm is tasked with adjusting the controller's parametrisation to maximise closed-loop performance based on observed data from interactions with the real environment. Algorithms that are typically employed are Q-learning [60, 75, 221] and policy gradient methods [41, 180]; to enable the corresponding gradient-based updates, nonlinear sensitivity analysis of the MPC optimisation problem can be leveraged [35].

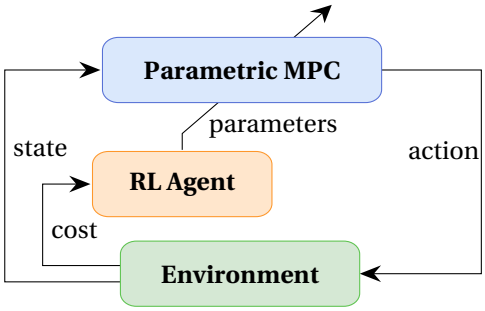

Figure 1.1.: Diagram of the MPC-based RL architecture

The ensuing MPC-based RL scheme constitutes an adaptive, model-based architecture that enjoys the benefits of both MPC and RL. Despite the presence of mismatches between the prediction model and the real system and of other approximately crafted and tuned components, the MPC control scheme is able to learn and deliver at convergence the optimal policy and value functions of the underlying RL task, assuming successful convergence during learning and a sufficiently expressive and rich parametrisation of the MPC problem. In contrast to DRL with deep neural networks as function approximators, this approach exhibits the following key advantages:

- The MPC problem at its core allows for an easy and seamless integration of a priori expert-based knowledge into the control scheme, usually in the form of an approximate, imperfect prediction model (e.g., based on first principles).
- MPC-based agents are generally more amenable to analysis and certification in terms of stability and constraint satisfaction. This claim is supported by the vast body of literature on MPC [28, 163], and the lack of interpretability for the majority of neural network-based schemes [74]. Although challenging, it is possible to ensure that theoretical properties of the MPC controller, such as stability, recursive feasibility and constraint guarantees, are preserved during the RL learning process [75, 233].
- Constraints on inputs and/or states can also be explicitly and systematically taken care of by the MPC policy, something that neural networks struggle with.

Compared to traditional non-learning MPC formulations, the MPC-based RL approach shows the following benefits:

- The need for an extensive open-loop data collection and system identification phase is eliminated, as the MPC parametrisation is automatically adjusted based on closed-loop data to improve performance, and not necessarily the prediction accuracy. This bypasses the issue of model inaccuracies and endows the scheme with performance-oriented online adaptation capabilities. In other words, the RL algorithm seeks the optimal model that, when cast into the MPC problem, yields the best closed-loop performance; it does not aim at finding the best model in terms of fidelity and accuracy. This concept is at other times referred to as *identification for control* [161].[2]

- By appropriately penalising constraint violations and with a rich enough MPC parametrisation, the RL algorithm can guide the learning process to produce an MPC policy that has learnt to satisfy system constraints. This may be achieved despite the prediction model being inaccurate (see above) and unable to predict state trajectories exactly. In fact, the performance-driven model and the other MPC components will get adjusted in such a way as to compensate for prediction errors that would otherwise result in constraint violations.

Beyond control: global optimisation as a sequential decision problem

While the discussion above has implicitly focused on closed-loop control of physical systems, the same RL formalism naturally extends to control tasks where the states are not a representation of some physical or real-world quantities. Global optimisation problems fall under this umbrella [16, 19, 72, 99]. The goal of global optimisation is to find the global optimum of an expensive-to-evaluate, black-box function, often without gradient information, with the fewest iterations possible. In this setting, the "state" can be regarded as the information gathered so far (used to build a surrogate model of the function), and the "action" as the next experiment to evaluate, e.g., a simulation or hyper-parameter configuration to test out. Classical global optimisation algorithms typically commit to a single myopic decision at each iteration, selecting the next candidate point according only to the information currently available, and disregard its impact on the evolution of the information and surrogate model at subsequent iterations (i.e., how the state will evolve after taking an action).

Casting the optimisation search itself as a sequential decision process reveals a dynamical nature: every evaluation changes the surrogate model's state, and its cost/reward is felt not only immediately but also through how it influences future evaluations. This viewpoint runs perfectly in parallel with the control of physical plants that was tackled in previous sections. In this context, approximate dynamic programming (an alternative term to model-based RL) methodologies have been proposed to optimally solve this decision process [96, 97, 115–117]. A model-based RL agent applied to this context can

[2] Note that the two identification procedures, i.e., the standard system identification and performance-driven identification, are not mutually exclusive and can be appropriately merged into a single approach (see, e.g., [136]).

then plan several evaluations ahead by rolling the evolution of the information/surrogate model's state forward along the iteration axis, analogously to how an MPC controller rolls its prediction model forward in time. The ensuing policy is labelled nonmyopic, because it is able to automatically and inherently balance the exploration-exploitation trade-off over the prediction horizon by casting it into an optimisation problem.

1.2. Challenges and research questions

The previous section highlighted how model-based RL can offer impressive benefits, in particular when paired with MPC to formulate a learning-based predictive control strategy and when integrated into global optimisation to combat the myopia of standard approaches. However, the literature is still undoubtedly plagued with open questions and further challenges. Amongst these, in this dissertation the following issues are tackled. In the context of RL-enhanced predictive control:

1. Safety guarantees on constraint satisfaction are still difficult to achieve for RL agents using an MPC controller as function approximator. Relevant works are [76, 233], which formalise how safety and stability may be preserved during learning and conceive a robust MPC formulation coupled with an online set-membership system identification approach for a linear system, guaranteeing safety robustly against the estimated disturbance set. Despite these advances, the application of the same methodologies to more complex nonlinear cases is still a challenge due to tractability and numerical complexity. On top of that, despite the availability of said robust approach in the linear case, safety can inherently only be guaranteed up to a probability less than one, due to the lack of perfect knowledge of the underlying system and limited data.

   Therefore, there is a strong need to seek RL-enhanced MPC policies that are safe for a wider range of systems and applications, from linear to nonlinear, both at the trajectory level (i.e., given the current parametrisation, the corresponding MPC controller shall induce safe state trajectories with high probability) as well as the learning level (i.e., during learning the RL agent shall only explore parametrisations that yield safe controllers with high probability).

2. The MPC-based RL framework tackled so far constitutes a learning approach for a single, centralised agent. This paradigm becomes in general prohibitive and intractable when extended to multi-agent systems, where centralisation requires either a specific topology, with all agents connected to the central agent, or communication across intermediate agents. Either of these features may be lacking or infeasible to implement. Concurrently, from the MPC point of view, centralised computation can become numerically too complex, particularly as the number of subsystems grows, and requires the sharing of sensitive information, such as objective functions, with the central agent.

   Thus, in the event that centralised training is unattainable (e.g., because of privacy concerns, or to enable continuous online learning, or simply due to the lack of a centralised unit or proper communication to and from it), the MPC-based RL

framework may offer an alternative approach to handle the multi-agent setting in a computationally efficient and privacy-aware way.

3. The extent to which model-based RL with MPC has been adopted in real-world applications is still limited. Though it has shown success in the control of, e.g., smart power grids [39, 41] and autonomous surface vehicles [40, 110, 137], its impact on other applications is yet to be investigated.

For what concerns global optimisation:

4. While nonmyopic strategies, rooted in model-based RL, tend to outperform their myopic counterparts in terms of performance and convergence, they are often computationally much more intensive. This is mainly due to their probabilistic Bayesian nature and its associated complexity in predicting how the stochastic surrogate model evolves as new experiments are designed. Recently, alternatives to probabilistic myopic approaches have been based on deterministic prediction models instead [100, 139], and have been shown to be competitive with respect to their Bayesian counterparts [16, 19, 50]. Since no probabilistic computations are needed, the ensuing algorithms are typically more lightweight and computationally efficient.

   However, there is a gap in the literature in investigating how to integrate nonmyopic RL in these deterministic formulations, thus taking advantage of both the nonmyopia offered by RL and the efficiency of deterministic models.

In particular, given these open challenges from the literature, in this dissertation the following research questions are addressed:

RQ-SA. *How can safety be enforced with high probability and in a computationally efficient and tractable way during the training process of a reinforcement learning agent with model predictive control as function approximator?*

RQ-SC. *How can reinforcement learning with model predictive control as function approximator be scaled to a multi-agent setting in a distributed and privacy-preserving way?*

RQ-AP. *Can the range of applications in which reinforcement learning with model predictive control as function approximator has been successfully applied be extended to other relevant, challenging control problems not yet tackled in the state of the art?*

RQ-GO. *How can approximate dynamic programming techniques be employed to formulate nonmyopic global optimisation approaches with deterministic surrogate models?*

1.3. Contributions and outline

This dissertation addresses the aforementioned open questions in its following chapters. Here, an overview of the key contributions and their connection to these questions are

given. The structure of the dissertation is also outlined in Figure 1.2. Briefly, Chapters 2 and 3 tackle the issue of encouraging probabilistically safe policies in MPC-based RL; Chapter 4 proposes an adaptation of the framework to the multi-agent RL setting; Chapters 5 and 6 extend the application of this framework to the greenhouse climate and traffic control problems; and Chapter 7 leverages RL to devise nonmyopic, multi-step lookahead global optimisation strategies with deterministic surrogate models. Lastly, concluding remarks and recommendations for future work can be found in Chapter 8.

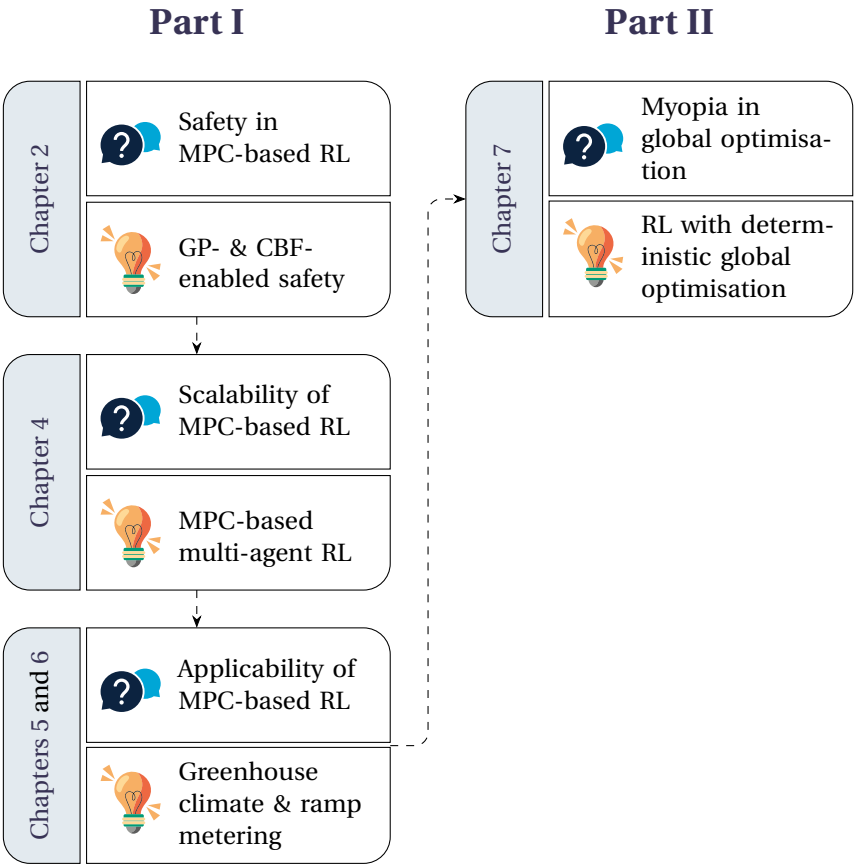

Figure 1.2.: Structure of this dissertation

Grouped by the corresponding research question, the contributions read as follows. Part I deals with MPC-based RL:

RQ-SA. Chapters 2 and 3 delve into the issue of safety in RL with MPC as function approximation scheme. Both chapters tackle the probabilistic case (in which constraint satisfaction, thus safety, can only be guaranteed with some probability), but do so at two different levels. Chapter 2 implements a safety-promoting mechanism at the update level, i.e., it ensures with high probability and under

some assumptions that, each time after the MPC scheme gets updated by the RL algorithm, the ensuing policy yields safe trajectories. The proposed mechanism leverages Gaussian process regression [162] to stochastically model the unknown relationship between MPC parametrisation and constraint satisfaction and to infer which parametrisations are likely to yield safe control policies. On the other hand, Chapter 3 guarantees safety at the control level, i.e., the MPC controller is devised in such a way as to always behave in a probabilistically safe manner, no matter what the parametrisation provided by RL is. This is possible thanks to the use of probabilistic control barrier functions [10] embedded into the MPC scheme, guaranteeing up to some probability that the controlled trajectories will satisfy the constraint. This is advantageous since, while the controller is safe by construction, the RL algorithm can tune it purely for performance without worrying about preserving safety in its updates.

RQ-SC. The multi-agent case is addressed in Chapter 4, where a distributed MPC scheme is proposed as function approximation scheme of the RL problem [130]. The MPC problem is solved distributively via the alternating direction method of multipliers (ADMM) [30], while the communication between agents prior to RL updates is handled privately through consensus [152]. The ensuing approach thus enables decentralised learning with efficient, privacy-preserving computations. On top of that, in the limit of ADMM and consensus iterations, the proposed distributed learning scheme is proven to converge to the centralised solution.

RQ-AP. Chapters 5 and 6 extend the use of MPC-based RL to two new real-world applications. In both of these applications, the dynamical model is an approximation of the corresponding real system, with several parameters to be identified correctly in order for the MPC controller to achieve satisfactory closed-loop performance. This challenge prompts the use of learning-based control methodologies, such as MPC-based RL, that can automatically take care of this tuning issue. In Chapter 5, the greenhouse climate control problem is tackled [27, 204]. This task requires the controller to adjust the climate inside a greenhouse to maximise the crop economic yield and, at the same time, minimise control effort and satisfy as much as possible system constraints despite the presence of prediction uncertainty. Chapter 6 focuses on highway traffic management instead, in the form of ramp metering control [155, 182, 209]. The problem is similarly defined: the controller must regulate the inflow of vehicles at a highway ramp so as to avoid congestion in the highway as well as long queues at the ramp, while minimising control effort. Both chapters show promising results, with MPC-based RL being the most performing methodology in several aspects.

Conversely, Part II tackles the application of model-based RL to global optimisation:

RQ-GO. Chapter 7 adapts two well-known approximate dynamic programming methodologies, rollout [24] and multi-step scenario-based optimisation [97], to the case in which the surrogate model is deterministic and either based on

inverse distance [100] or radial basis functions [139]. The benefits of the proposed approaches combine the computational efficiency of these deterministic models with the nonmyopic lookahead behaviour of rollout and multi-step optimisation techniques.

I

Model-based reinforcement learning for predictive control

2

Learning safety in model-based reinforcement learning using MPC and Gaussian processes

We propose a method to encourage safety in Model Predictive Control (MPC)-based Reinforcement Learning (RL) via Gaussian Process (GP) regression. This framework consists of 1) a parametric MPC scheme that is employed as model-based controller with approximate knowledge on the real system's dynamics, 2) an episodic RL algorithm tasked with adjusting the MPC parametrisation in order to increase its performance, and lastly, 3) GP regressors used to estimate, directly from data, constraints on the MPC parameters capable of predicting, up to some probability, whether the parametrisation is likely to yield a safe or unsafe policy. These constraints are then enforced onto the RL updates in an effort to enhance the learning method with a probabilistic safety mechanism. Compared to other recent publications combining safe RL with MPC, our method does not require further assumptions on, e.g., the prediction model in order to retain computational tractability. We illustrate the results of our method in a numerical example on the control of a quadrotor drone in a safety-critical environment.

This chapter is based on [4].

2.1. Introduction

The modern proliferation of advances in Machine Learning, combined with the increasing availability of computational and sensing capabilities in modern control systems, has led to a growing interest in learning-based control methodologies that strive to learn different elements of the controller scheme from data in order to, e.g., reduce conservativeness and improve closed-loop performance, or encourage safety and robustness [32]. In particular, Model Predictive Control (MPC) is the subject of a large amount of the current research in this field thanks to its wide variety of applications and high versatility, especially in regard to multivariate and constrained systems [92]. While robust and stochastic MPC schemes allow for a systematic handling of different sources of uncertainties affecting the MPC performance, they often follow a strict separation between the offline design phase and the closed-loop application, where in most cases the controller remains unresponsive to changes in, e.g., the system or task. Adaptive and learning-based methodologies aim at overcoming this paradigm by automating the controller design and adaptation based on data collected during operation [141].

To this end, several tools from ML have been successfully coupled with MPC in order to enhance its performance or augment its capabilities. Among all, we can refer to Gaussian Process (GP) regression, which has been widely employed to, e.g., automatically learn the unmodelled (often nonlinear) dynamics in the prediction model [91] and to construct provably accurate confidence intervals on predicted trajectories [109]; and black-box Bayesian optimisation, which is a widespread tool for automatic tuning and optimisation of the MPC controller hyperparameters [161], thus relieving some burden on the design phase. Equally interesting is the combination of MPC with Reinforcement Learning (RL), another paradigm of ML gaining more and more attention thanks to its recent developments, in which an agent learn how to interact with the environment and take actions so as to maximise/minimise a reward/cost signal [197]. The majority of RL algorithms that are nowadays popular in the ML community are model-free and rely purely on observed state transitions and realisations of stage cost to increase the performance of the control policy. Deep Neural Networks (DNN) are predominantly used as function approximators and, while they have proven effective, they lack interpretability, and their behaviour is difficult to analyse, certify, and trust in regard to stability and constraint satisfaction.

Recently, [75] proposed and justified the use of MPC as the function approximation of the optimal policy in model-based RL. In such a scheme, the MPC controller yields the policy and value functions via its optimisation problem, while, at the same time, the learning algorithm is tasked with adjusting the parametrisation of the controller in an effort to implicitly or explicitly discover the optimal policy, thus improving closed-loop performance in a data-driven fashion. Via sensitivity analysis [35], it is possible to apply RL algorithms such as Q-learning and stochastic/deterministic policy gradient by differentiating the MPC optimisation problem with respect to its parameters and updating these accordingly. In contrast to DNN-based RL, MPC-based RL agents can ideally exploit the underlying optimisation problem to provide stability requirements by construction and guarantee safety, which defines the ability of a control policy to yield state-action trajectories that do not violate system constraints. However, even in this novel paradigm, certificates on safety and stability remain an arduous challenge. If

proper care is not taken, the RL updates can still potentially jeopardise the MPC scheme by updating its parameters to an undesirable configuration, resulting in a controller with unsatisfactory behaviour, e.g., that does violate constraints and hence is unsafe.

In this chapter, we propose and explore a learning method in the context of safe MPC-based RL where the estimation of the safety set (i.e., the set of MPC parameters that yields a controller able to attain constraint satisfaction) is carried out in a data-driven fashion. In particular, we borrow a technique for estimating unknown constraints from the fields of constrained Bayesian optimisation [188] and safe exploration [173], which offer tools to tackle black-box optimisation problems that are constrained by some unknown functions in the decision variables. As a consequence, we are able to learn in a probabilistic way which configurations of the MPC controller are safe or not directly from data.

The rest of this article is organised as follows. Background and motivations both for safe RL in combination with MPC and for unknown constraint modelling via GP are given in Section 2.2. The proposed algorithm and details of its implementation are discussed in Section 2.3. A numerical experiment showing the effectiveness of the proposed method in a safe-critical quadrotor control application is reported and analysed in Section 2.4. Lastly, conclusions and directions for future research are presented in Section 2.5.

2.2. Background

2.2.1. MPC as function approximation in safe RL

As in [37, 197], we describe discrete-time system dynamics as a Markov Decision Process with continuous state $s \in \mathbb{R}^{n_s}$ and continuous action $a \in \mathbb{R}^{n_a}$, and state transitions $s, a \to s_+$ with the underlying conditional probability density

$$\mathbb{P}(s_+ \mid s, a) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to [0, 1]. \tag{2.1}$$

Considering (2.1) a deterministic policy $\pi_\theta(s) : \mathbb{R}^{n_s} \to \mathbb{R}^{n_a}$ parametrised in $\theta \in \mathbb{R}^{n_\theta}$ and resulting in the state distribution τ_{π_θ}. The performance of such policy is defined as

$$J(\pi_\theta) := \mathbb{E}_{\tau_{\pi_\theta}}\left[\sum_{k=0}^{\infty} \gamma^k L\big(s_k, \pi_\theta(s_k)\big)\right], \tag{2.2}$$

where s_k is the state at time step k, $L(s, a) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to \mathbb{R}$ is the stage cost and $\gamma \in (0, 1]$ is the discount factor. The RL problem is then to find the optimal policy $\pi_\theta^\star$ as

$$\pi_\theta^\star = \arg\min_\theta \quad J(\pi_\theta). \tag{2.3}$$

While DNNs are possibly the most common choice for representing such policy, an MPC scheme can be exploited as function approximation as proposed in [75]. Consider the

following MPC problem approximating the value function $V_\theta : \mathbb{R}^{n_s} \to \mathbb{R}$ as

$$V_\theta(s) = \min_{u,x,\sigma} \quad \lambda_\theta(x_0) + \sum_{k=0}^{N-1} \gamma^k \left(L_\theta(x_k, u_k) + w^\top \sigma_k\right) + \gamma^N \left(V_\theta^{\mathrm{f}}(x_N) + w_{\mathrm{f}}^\top \sigma_N\right) \tag{2.4a}$$

$$\text{s.t.} \quad x_{k+1} = f_\theta(x_k, u_k), \quad k = 0, \dots, N-1, \tag{2.4b}$$

$$h_\theta(x_k, u_k) \le \sigma_k, \quad k = 0, \dots, N-1, \tag{2.4c}$$

$$h_\theta^{\mathrm{f}}(x_N) \le \sigma_N, \tag{2.4d}$$

$$x_0 = s, \tag{2.4e}$$

where vectors x, u and σ respectively collect states, actions and slack variables over the horizon N. In (2.4a), $\lambda_\theta(x_0)$ is an initial cost term, $L_\theta(x,u)$ is the stage cost, and $V_\theta^{\mathrm{f}}(x)$ is a terminal cost approximation. $f_\theta(x,u)$ is the model approximation, and $h_\theta(x,u)$, $h_\theta^{\mathrm{f}}(x)$ are inequality constraints. Lastly, w and w_{f} are the weights of the slack variable in the objective. The value function (2.4) satisfies the fundamental equalities of the Bellman equations:

$$Q_\theta(s,a) = \min_{u,x,\sigma} \quad (2.4a) \tag{2.5a}$$

$$\text{s.t.} \quad (2.4b) - (2.4e) \tag{2.5b}$$

$$u_0 = a, \tag{2.5c}$$

$$\pi_\theta(s) = \arg\min_a \quad Q_\theta(s,a). \tag{2.6}$$

Therefore, in RL terms, the MPC parametric scheme acts as policy provider for the learning agent, whose goal is to modify the parameters θ of the controller in an effort to minimise (2.3). Various forms of RL exist that solve this problem directly or indirectly via iterative updates

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{k=0}^{m} \psi(s_k, a_k, s_{k+1}, \theta), \tag{2.7}$$

where $\alpha \in \mathbb{R}_+$ is the learning rate, m denotes the number of observations used in the update (i.e., a batch of observations), and ψ captures the controller's performance and varies with the specific algorithm. For example, in recursive Q-learning we have that $m = 1$ and

$$\psi(s_k, a_k, s_{k+1}, \theta) = \left(L(s_k, a_k) + \gamma V_{\theta'}(s_{k+1}) - Q_\theta(s_k, a_k)\right)^2, \tag{2.8}$$

where $V_{\theta'}$ is evaluated for a frozen copy of the parameters $\theta' = \theta$ to prevent value estimates from contributing to the gradient update. In contrast, policy gradient methods compute

$$\mathbb{E}_{\tau_{\pi_\theta}}[\psi(s_k, a_k, s_{k+1}, \theta)] = J(\pi_\theta). \tag{2.9}$$

Nevertheless, while in most learning algorithms a safe policy is usually implicitly achieved at convergence assuming violations are appropriately penalised, naively performing updates (2.7) does not guarantee such property during the learning process itself. Thus, we would like to restrict the search of θ to a set $\mathscr{S}$ that ensures the safety of the corresponding parametric policy. More specifically, consider a generic episodic task of duration T with n_{c} safety-critical constraints, gathered as the function

$h : \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_c}$ (e.g., bounds on the temperature of a chemical reactor, or obstacles in autonomous navigation). By deploying the MPC policy π_θ to accomplish the task, safety amounts to satisfying

$$h\big(x_t, \pi_\theta(x_t)\big) \leq 0, \quad t = 1, \dots, T. \tag{2.10}$$

Thus, we can define the safe set $\mathscr{S}$ as

$$\mathscr{S} := \Big\{\theta \;\Big|\; (2.10) \text{ holds}\Big\}, \tag{2.11}$$

leading to the following reformulation of (2.7) as a constrained update

$$\min_{\theta_+} \quad \frac{1}{2}\|\theta_+ - \theta\|_2^2 + \alpha \nabla_\theta \sum_{k=0}^{m} \psi(s_k, a_k, s_{k+1}, \theta)^\top (\theta_+ - \theta) \tag{2.12a}$$

$$\text{s.t.} \quad \theta_+ \in \mathscr{S}, \tag{2.12b}$$

by then setting $\theta \leftarrow \theta_+^\star$, where $\theta_+^\star$ is the optimal point of (2.12). Nonetheless, the characterisation of $\mathscr{S}$ is in general difficult, even more so in the case of function approximators with vast parametrisations. Fortunately, an MPC approximator can be leveraged here: as shown in [233] and further analysed in terms of feasibility and stability in [76], a robust MPC formulation coupled with an online set-membership system identification approach allows to guarantee that the MPC-RL algorithm is robustly safe with respect to the estimated disturbance set throughout the whole training, despite the fact that the dynamics of the real system are unknown to the learning agent. However, even though an adaptive robust scheme is deployed, safety is only guaranteed up to a probability < 1, due to the lack of perfect knowledge of the underlying system (2.1). In this context, safety can be viewed in the light of Bayesian inference, where it is probabilistically conditioned on our limited knowledge of the system, i.e., prior information and actual data, labelled as $\mathscr{D}$, with probability β. Henceforth, we will refer to the notion of safety in a probabilistic term and define $\mathscr{S}_\mathscr{D}$ as

$$\mathscr{S}_\mathscr{D} := \Big\{\theta \;\Big|\; \mathbb{P}((2.10) \,|\, \mathscr{D}) \geq \beta\Big\}. \tag{2.13}$$

A limitation in the approach proposed by [233] is that, in order to retain computational tractability in the algorithm, the MPC prediction model f_θ has to be linear and kept fixed during learning. As acknowledged in [233], while linear MPC applied to nonlinear systems can still be successful, strong nonlinearities are likely to deteriorate its performance, compromise safety, and prevent the agent from tuning the prediction model, which might severely hinder learning and lead to convergence to suboptimal policies.

In this chapter, we propose a method that estimates the safe set $\mathscr{S}_\mathscr{D}$ directly from episodic data in a way that bypasses the aforementioned limitation and that conforms with the probabilistic nature of the issue of safety itself. As detailed next, we propose to leverage GP regression tools that deliver surrogate models that are able to provide uncertainty estimates whether a specific θ belongs to $\mathscr{S}_\mathscr{D}$ or not. This will then allow us to formulate an RL algorithm that can learn while being safe, up to some probability.

2.2.2. Estimation of unknown constraints

In essence, the matter of the estimation of set $\mathscr{S}_{\mathscr{D}}$ in order to impose safety coincides with estimating whether a given state-action trajectory, produced by an MPC controller parametrised in θ, will satisfy constraints $h(x_t, u_t) \leq 0,\ t = 1, \dots, T$, with $u_t = \pi_\theta(x_t)$. However, while on the one hand the relationship $h(x, u)$ is given by the task, on the other hand we cannot predict beforehand whether $\pi_\theta(x)$ will yield safe control actions and hence safe trajectories. Therefore, we introduce the following safety constraint function $z : \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_c}$

$$z(\theta) = \max_{t=1,\dots,T} \quad h\big(x_t, \pi_\theta(x_t)\big), \tag{2.14}$$

that directly maps parametrisations to positive values, in case θ resulted in trajectories with violations, or to negative values, in case of safe trajectories. Analytically, this function is unknown to us, but each time a state-action trajectory for the given task is computed, it can be evaluated for the current θ. Therefore, by leveraging past observed data, an approximation of (2.14) can be learnt and used to predict safety at any previously unobserved query point θ^*. Thus, we could use this surrogate model of $z(\theta)$ to constrain learning only to those parameters that are estimated to be safe.

Assumption 2.1. *We assume the trajectory to be dependent only on θ, and, consequently, that $z(\theta)$ wholly captures safety (i.e., no other factor has an impact).*

Remark 2.1. *In general, this is not the case as also the initial and final conditions of the task, as well as the initialisation of the optimisation solver in case of nonlinear MPC, will affect constraint satisfaction. To mitigate these dependencies, we assume the episodic task to be repetitive, e.g., to have the same initial and final conditions, and the MPC problem to be convex. Analogously, we could also assume these factors to be readily included into the MPC parametrisation θ so that, in principle, all parameters affecting safety may be included in $z(\theta)$ and made learnable.*

Remark 2.2. *In* (2.14)*, the maximum over the time horizon T is taken in order to reduce violations along a trajectory to a point-wise maximum violation, but this does not invalidate* (2.13) *since $z(\theta) \leq 0 \Leftrightarrow h\big(x_t, \pi_\theta(x_t)\big) \leq 0,\ t = 1, \dots, T$. Likewise, functions other than* max *could be crafted to quantify safety of a trajectory while still encoding* (2.13).

Unknown constraint estimation has been already tackled in literature. In constrained Bayesian optimisation, [188] propose a method for black-box optimisation with unknown constraints that can only be evaluated at specific query points, where GP regression is exploited to achieve surrogate models of both the objective and the constraint functions, the latter being then used to weigh the exploration strategy by the probability of constraint satisfaction; conversely, [111] include the GP models as barrier penalty terms. Safe exploration algorithms also make extensive use of GPs, either for classification [173] or in combination with regularity tools [203]. While GPs are a popular choice because they naturally allow for a quantification of the prediction uncertainty, [238] suggest modelling the unknown functions via Inverse Distance Weighting interpolation, which are employed to directly predict the probability of being feasible.

In this chapter, GPs are used to model the safety-critical constraints (2.14) since these processes allow to inherently address the probabilistic nature of (2.13). Training data for

the GPs are collected each time the given task is solved, yielding the tuple $\langle \theta, \tilde{z} \rangle$, where θ is the current MPC parametrisation and $\tilde{z}$ is a noisy observation of (2.14). Observations are generally regarded as noisy due to, e.g., measurement noise or disturbances in the dynamics. Therefore, the basic idea is that, given the dataset $\mathscr{D}_n = \{(\theta^i, \tilde{z}^i)\}_{i=1}^n$, one independent GP regressor per constraint is used to model the corresponding safety constraint function z_j, $j = 1, \dots, n_c$. In particular, we assume the function to have a GP prior with mean $\mu_0 : \mathbb{R}^{n_\theta} \to \mathbb{R}$ and covariance kernel $\nu : \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_\theta} \to \mathbb{R}$. Under the GP prior and assumed i.i.d. Gaussian noise, observations $\tilde{z}_j^i$ are jointly Gaussian distributed, so that, at any query point θ^*, the corresponding constraint violation $z_j(\theta^*)$ must be jointly Gaussian too, i.e., $z_j(\theta^*) \sim \mathcal{N}\big(\mu_j^n(\theta^*), \sigma_j^{n2}(\theta^*)\big)$, where $\mu_j^n(\theta^*)$ is the estimate mean and $\sigma_j^n(\theta^*)$ its uncertainty [162, Algorithm 2.1]. In this context, (2.13) can be approximated from data as

$$\mathscr{S}_{\mathscr{D}_n} = \left\{ \theta \;\middle|\; \mathbb{P}\Big(z_j(\theta) \le 0 \;\Big|\; \mathscr{D}_n\Big) \ge \beta,\ j = 1, \dots, n_c \right\}, \tag{2.15}$$

where z_j is represented by a GP trained on data in $\mathscr{D}_n$.

In the next section, we illustrate how to combine MPC-based RL with GP constraint modelling into a safety-aware algorithm.

2.3. Data-driven safe MPC-based RL

Algorithm 1 illustrates the procedure to promote safety, up to some probability β, upon an MPC-based RL agent. Starting from an initial parametrisation of the MPC controller and a (possibly empty) dataset of initial observations, experiment i is carried out by completing the given task with an MPC scheme parametrised in θ^i. At the end of such experiment, the full state-action trajectory can be collected and the maximum constraint violation $\tilde{z}^i$ computed. Thanks to the collected data, constraints $z_j(\theta)$, $j = 1, \dots, n_c$ are then approximated via GP regressors. The RL update is then enhanced with the trained GP-based constraints, which help the algorithm in selecting the next parameters θ_+ that satisfy these constraints and thus are likely to yield a safe policy.

Remark 2.3. *Gaussianity of the GPs can be conveniently leveraged to express constraint* (2.16b) *deterministically as*

$$\mu_j^i(\theta_+) + \Phi^{-1}(\beta)\sigma_j^i(\theta_+) \le 0, \quad j = 1, \dots, n_c, \tag{2.17}$$

where Φ^{-1} is the normal distribution quantile function. However, since the GP terms $\mu_j^i(\theta_+)$ and $\sigma_j^i(\theta_+)$ are in practice always nonlinear, the overall RL update optimisation problem (2.16) *becomes nonlinear too. Moreover, there exist no guarantees on the feasibility of* (2.16)*, i.e., the update might get stuck as it is unable to find a safe $\theta_+^\star$. This is especially relevant at the beginning of the simulation in case $\mathscr{D}_0$ is empty, i.e., data is scarce and the GP approximation is poor. Currently, one can mitigate this issue by backtracking (i.e., iteratively reducing) the requested safety probability β to smaller and smaller values till feasibility is recovered, as shown in lines 6-9. Of course, this trades off safety with feasibility, which might be undesirable in cases where the former must be guaranteed with high enough certainty. However, one could easily envision methods for alleviating*

Algorithm 1: Data-driven safe MPC-based RL

Input: Initial MPC parameters θ^0; initial observations $\mathscr{D}_0$; $\varrho \in (0,1)$
Output: Learnt parametrisation $\theta^{n_{\max}}$

for $i = 0, \dots, n_{\max} - 1$ **do**
  Perform MPC closed-loop task with θ^i by solving (2.4) at each time step $t = 1, \dots, T$
  Observe $\tilde{z}_j^i$, $j = 1, \dots, n_c$, as per (2.14)
  Augment dataset $\mathscr{D}_i \leftarrow \mathscr{D}_{i-1} \cup \{(\theta^i, \tilde{z}^i)\}$
  Train GPs for $z_j(\theta)$, $j = 1, \dots, n_c$, on $\mathscr{D}_i$
  do
    Try performing safe RL update

$$\theta_+^\star \leftarrow \arg\min_{\theta_+} \quad (2.12a) \tag{2.16a}$$

$$\text{s.t.} \quad \theta_+ \in \mathscr{S}_{\mathscr{D}_i} \tag{2.16b}$$

    Reduce safety probability $\beta \leftarrow \varrho\beta$
  while (2.16) *is* ***not*** *feasible*
  Update parametrisation $\theta^{i+1} \leftarrow \theta_+^\star$
end

this issue by, e.g., injecting prior knowledge via the GP prior mean or the initial dataset $\mathscr{D}_0$, or, alternatively, assuming the initial condition θ_0 to be safe with unitary probability (thus acting as a sort of backup configuration).

Remark 2.4. *While GPs are convenient because they allow a deterministic formulation of* (2.16b) *that naturally takes estimation uncertainty into account, they are not per se a mandatory component of the proposed approach. One could indeed envision the use of different regressor types with different advantages and disadvantages, such as, to mention few, Inverse Distance Weighting interpolants [238], whose output in range* $[0,1]$ *can be directly interpreted as probability, Support Vector Machines [184], that perform well with small/medium-size training sets but do not quantify uncertainty, or Bayesian Neural Networks [101], which are powerful tools in approximating complex distributions but require much larger amounts of training data. Moreover, different types of Gaussian Process regression are available that address the weaknesses of vanilla GPs that may emerge during training, such as the sparse [51] and warped variants [185], which are suitable for training on larger datasets or allow for a non-Gaussian process by warping variables to a latent space, respectively.*

Remark 2.5. *While update* (2.16) *is performed at every iteration of Algorithm 1, it can be in principle performed at lower frequencies. This is especially relevant in those RL tasks in which convergence is difficult to achieve and can benefit from the employment of experience replay buffers to stabilise learning.*

2.4. Numerical experiment

To put our proposed algorithm to the test, it was implemented and simulated in a numerical experiment of a quadrotor drone application adapted from [211]. In the following sections, we detail the system, the controller and the safe learning algorithm, and present the final results.

2.4.1. System description

The aim of the task is for the drone to reach a final destination in the fastest and most accurate way, while avoiding violations of safe-critical boxing constraints on the states and control actions. Wind disturbances at different altitudes influence the drone, which add nonlinear effects to the dynamics. The goal of the RL agent is then double: on one side, it tunes the parameters of the MPC controller to achieve stable flight and better performance in targeting the final position; on the other side, it learns to avoid constraint violations as much as possible, despite wind disturbances and while still accomplishing the task.

The discrete-time dynamics of the drone are of the form

$$x_{t+1} = Ax_t + Bu_t + C\Psi(x_t)w_t + G, \tag{2.18}$$

where the state $x \in \mathbb{R}^{10}$ contains the pose (position, velocity, attitude, and its rate) of the quadrotor, the control action $u \in \mathbb{R}^3$ dictates the desired pitch/roll attitude and vertical acceleration, and system matrices A and B describe the dynamics of the quadrotor around the hovering state, and vector G represents the constant gravity pull. See Appendix 2.A and [29] for a more detailed insight on the model. The additional term $C\Psi(x_t)$ models nonlinear wind disturbances at time step t: $\Psi(x) \in \mathbb{R}^3$ is a state-dependent vector simulating three different wind currents at different altitudes, each implemented as a squared exponential basis function with unknown hyperparameters, whereas matrix C modulates the influence of each gust on each state, and $w_t \sim \mathscr{U}(0,1)$ acts as a source of randomness. Box constraints h are imposed on the position, pitch, roll, as well as on the three control actions. The RL stage cost takes into account, at each step, the error from the target position s_f, the control action usage, as well as constraint violations

$$L(s,a) = \|s - s_\mathrm{f}\|_2^2 + c_1\|a\|_2^2 + c_2^\top \max\{0, h(s,a)\}, \tag{2.19}$$

where scalar c_1 and vector c_2 weigh the different contributions.

2.4.2. MPC function approximation

The controller chosen for this task consists of an MPC scheme based on (2.4): the stage/final costs are tracking costs (in both states and actions) but their cost matrices are fixed/non-learnable, and no initial cost is used, i.e., $\lambda_\theta(x_0) = 0$; the parametric prediction model is $f_\theta(x,u) = A_\theta x + B_\theta u + G_\theta$; the constraints are parametrised by means of a backoff parameter $h_\theta(x,u) = (1+\delta)h(x,u)$ and are made soft in the MPC to avoid infeasibilities. Eight different system parameters in A_θ, B_θ and G_θ affect the dynamics of the quadrotor drone but, out of these, the learning is restricted exclusively to the gravitational pull constant g and the vertical thruster coefficient K_z. Additionally, the constraint backoff

δ is made learnable, i.e., $\theta = \begin{bmatrix} g & K_z & \delta \end{bmatrix}^\top$, while all the other parameters in the drone model and optimisation are kept fixed during learning. To better test the viability of the proposed approach, all eight dynamics parameters in the MPC scheme are initialised wrongly (with an error $\approx \pm 20\%$), so as to lead to a controller with initial poor performance and constraint violations. Therefore, the agent has to achieve better performance and safety by only tuning θ, a small subset of all the dynamics parameters. It is worth highlighting the notion that this kind of parameter tuning differs in essence from the classical system identification, where parameters are chosen in such a way as to match the system's behaviour with observed data (e.g., in the sense of the prediction mean-squared-error), and a controller is only subsequently designed, in a possibly iterative procedure. On the contrary, here the adjustments of the learnable parameters are directly carried out with the goal of optimising performance, regardless of the model's prediction accuracy. This approach is sometimes called Identification for Control, according to which the best model for control needs not be the one providing the best predictions, but the best performance on the true system [161]. The two identification procedures are not mutually exclusive and can be appropriately incorporated in a unique approach, as done in [136].

2.4.3. Safe RL algorithm

For learning safety constraints (2.14), one independent GP is employed to model each constraint. A zero prior mean function was assumed, whereas the kernel function was selected as the sum of the squared exponential kernel and a white noise kernel:

$$\nu(\theta, \theta^*) = \sigma_1^2 \exp\left(-\frac{1}{2\ell^2} \|\theta - \theta^*\|_2^2\right) + \sigma_2^2 I_{n_\theta} \delta_{\theta\theta^*}, \tag{2.20}$$

where ℓ is the length scale and σ_1 is the multiplier of the exponential, σ_2 denotes the white noise level, and $\delta_{\theta\theta^*}$ is the Kronecker delta. As per Algorithm 1, the GP regressors are trained on past observed data before each update of the MPC parameters. For the RL updates, a second-order LSTD Q-learning algorithm [114] is employed to find the parameter vector θ that minimises closed-loop performance under the policy $\pi_\theta(s)$. Q-learning is a classical RL algorithm that tries to find the parametrisation which best fits the action-value function to the observed data, thus indirectly finding the optimal policy, and has shown promising results also with MPC [60]. The second-order Newton's method ensures faster convergence, for which gradient p and approximate Hessian H are found as

$$\delta_i = L(s_i, a_i) + \gamma V_\theta(s_{i+1}) - Q_\theta(s_i, a_i), \tag{2.21}$$

$$p = -\sum_{i=1}^{m} \delta_i \nabla_\theta Q_\theta(s_i, a_i), \tag{2.22}$$

$$H = \sum_{i=1}^{m} \nabla_\theta Q_\theta(s_i, a_i) \nabla_\theta Q_\theta^\top(s_i, a_i) - \delta_i \nabla_\theta^2 Q_\theta(s_i, a_i), \tag{2.23}$$

where H is modified when needed to be positive definite [151, Algorithm 3.3], and the gradient $\nabla_\theta Q_\theta$ is computed via nonlinear programming sensitivity analysis [35]. As in [233], exploratory behaviour is ensured during learning by adding to the MPC objective

(2.4a) the perturbation term $q^\top u_0$, with q randomly chosen. Lastly, in the constrained update (2.16), the target safety level β is set to 0.9, but backtracking on this level is enabled in case of infeasibility.

2.4.4. Results

The experiment was implemented in Python leveraging the symbolic framework CasADi [11] and its interface to the IPOPT solver [212]. The source code and simulation results can be found in the following repository: `https://github.com/FilippoAiraldi/learning-safety-in-mpc-based-rl`.

To investigate the effectiveness of the proposed algorithm, simulations were carried out for the LSTD Q-learning algorithm both with and without our GP-based safety enhancement. The safe variant of the algorithm was also tested with zero initial knowledge (i.e., $\mathscr{D}_0$ empty), and with limited prior knowledge in the form of 5 datapoints picked randomly from previous simulations. Where relevant, we also compare the algorithm with a baseline, in which the MPC controller is initialised with perfect knowledge on the system model. Each algorithm is averaged over 100 different simulations for 50 learning episodes. In the figures, average results are plotted with one standard deviation.

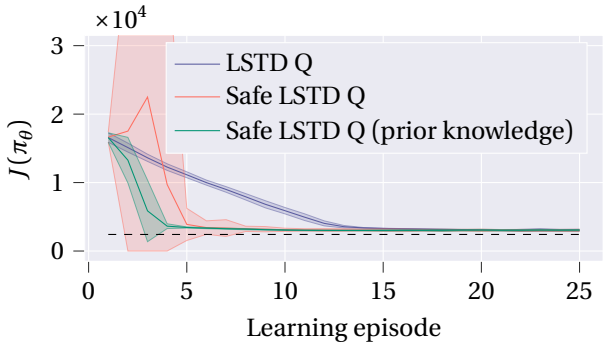

Figure 2.1.: Comparison in performance between the original LSTD Q-learning algorithm and its GP-based safe variant (with and without prior knowledge). In dashed black, the baseline cumulative cost when exact knowledge of the system is given to the MPC controller.

As shown in Figure 2.1, despite starting from an unfavourable initial configuration θ that yields large constraint violations, both safe and unsafe algorithms are able to recover and converge to a performance that is just $\approx 20\%$ worse than the baseline. Note that this recovery is achieved through adjusting only a limited subset of the system dynamics parameters. However, the convergence speed of the two substantially differs, the safe variants being much faster. This is explained by the fact that already in the first few iterations the safety mechanism is able to approximate the safe set $\mathscr{S}_{\mathscr{D}}$, even if only locally, and to steer the learning towards parameters θ that are likely to yield safer policies. However, we note that at the very beginning of the learning process the zero-knowledge safe algorithm suffers from much higher variance in the results

compared to its counterpart initialised with some prior information. This confirms that at the beginning the ability to predict safety is extremely poor and can even damage performance.

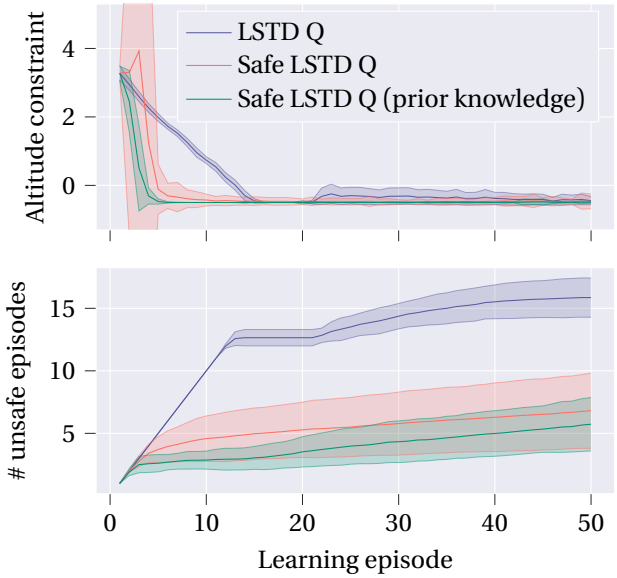

Figure 2.2.: Comparison in (top) violations of the altitude constraint, where positive values imply violation, and (bottom) the cumulative number of unsafe episodes

Figure 2.2 shows the effectiveness of our method in encouraging safety. At the end of the learning process, we witnessed a 57% and 64% reduction in the total number of unsafe episodes (episodes in which constraint violation occurs) for the two safe algorithms compared to the unsafe LSTD Q-learning variant. These results are achieved in a purely data-driven fashion and with no or little previous information on safety. Furthermore, constraint violations are recovered much faster, especially in the quadrotor altitude, which is the state variable most prone to violations. This explains why the performance of the safe algorithms shows faster convergence to better suboptimal policies.

Lastly, Figure 2.3 shows the impact of the backtracking on β, as explained in Remark 2.3. In particular, it shows on average during learning at which probability Algorithm 1 had to decrease β in order to obtain the feasibility of the constrained update (2.16). In the context of zero initial information, at the beginning β has to be substantially decreased to achieve feasible updates, while at convergence, when enough data has been collected to properly train the GPs, the required probability of 0.9 is achieved. Conversely, the simulations with initial knowledge seems to confirm that providing a small $\mathscr{D}_0$ to pre-train the GP-based safety constraints (2.16b) is an effective way to avoid backtracking on β to recover feasible updates.

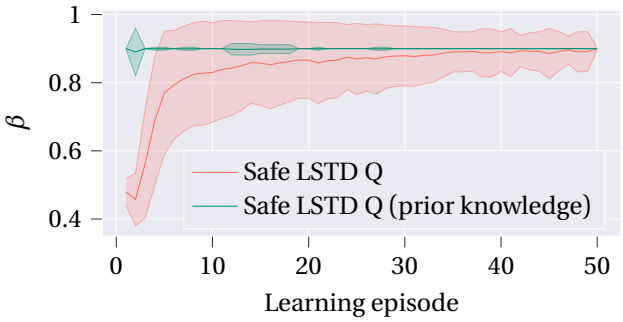

Figure 2.3.: Backtracked safety probability β during learning

2.5. Conclusions

In this chapter, we propose a method that enables safety in the context of MPC-based RL via GP regression. It learns and encourages safety exclusively from observed past trajectories, without the need of, e.g., a priori information on the system or estimation of disturbances affecting its dynamics. The algorithm is straightforward and can be implemented without extensive customisation on top of several RL agents, such as Q-learning and policy gradient methods. The numerical experiment for the quadrotor control application demonstrated that the safety mechanism successfully reduces the number of unsafe episodes during learning and also promotes faster convergence by avoiding unnecessary constraint violations.

Nonetheless, there are still open points regarding the proposed method. The necessity of backtracking β is somewhat undesirable, since it deteriorates safety. To avoid that, simulations where initial knowledge on safety is provided showed how this can alleviate the need for reducing β. However, even so, the algorithm may still get stuck and become infeasible for any β. An open question is how to avoid such situations. Additionally, it would be desirable to integrate this algorithm with more complex MPC schemes (e.g., robust or stochastic) where also other components (e.g., the disturbance set) may be learnt online in a probabilistic way. Finally, readers acquainted with Bayesian optimisation may have noticed a natural similarity between Algorithm 1 and its sampling-based counterparts, which we took inspiration from. An interesting point would be therefore to compare the efficacy, both in terms of performance and safety, of our gradient-based algorithm with safe BO algorithms.

2.A. Quadrotor's system dynamics

As described in Section 2.4, the discrete-time dynamics of the quadrotor drone are

$$x_{t+1} = Ax_t + Bu_t + C\Psi(x_t)w_t + G,$$

where the state consists of the 3D positions and velocities, as well as roll and pitch attitudes and rates, i.e.,

$$x = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & a_p & a_r & r_p & r_r \end{bmatrix}^\top,$$

and the control action is desired pitch and roll and vertical acceleration, i.e.,

$$u = \begin{bmatrix} u_p & u_r & u_z \end{bmatrix}^\top.$$

In particular, the dynamics can be described with the following matrices:

$$A = I_{10} + T_\mathrm{s} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -d_p & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_r & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -d_{\dot{p}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{\dot{r}} & 0 & 0 \end{bmatrix},$$

$$B = T_\mathrm{s} \begin{bmatrix} & 0_{5\times3} & \\ 0 & 0 & K_z \\ & 0_{2\times3} & \\ K_p & 0 & 0 \\ 0 & K_r & 0 \end{bmatrix},$$

$$G = -T_\mathrm{s} \begin{bmatrix} 0_{1\times5} & g & 0_{1\times4} \end{bmatrix}^\top,$$

where the various parameters can be found in Table 2.1. The nonlinear term $C\Psi(x)$ is

Table 2.1.: Quadrotor's Parameters

Parameter	Symbol
sampling time	T_s
gravitational constant	g
pitch/roll attitude coefficients	$d_{\{p,r\}}$
pitch/roll rate coefficients	$d_{\{\dot{p},\dot{r}\}}$
control gains	$K_{\{p,r,z\}}$

used to model three wind currents disturbing the dynamics:

$$C = T_\mathrm{s} \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ & 0_{3\times3} & \\ c_{71} & c_{72} & c_{73} \\ c_{81} & c_{82} & c_{83} \\ & 0_{2\times3} & \end{bmatrix},$$

$$\Psi(x) = \begin{bmatrix} \psi_1(p_z) & \psi_2(p_z) & \psi_3(p_z) \end{bmatrix}^\top,$$

where $\psi_i(p_z) = \exp\left(-(p_z - b_i)^2\right)$ represents the effect of the i-th gust at altitude b_i given the current vertical position p_z of the drone.

3

Probabilistically safe and efficient model-based reinforcement learning

This chapter proposes tackling safety-critical stochastic Reinforcement Learning (RL) tasks with a sample-based, model-based approach. At the core of the method lies a Model Predictive Control (MPC) scheme that acts as function approximation, providing a model-based predictive control policy. To ensure safety, a probabilistic Control Barrier Function (CBF) is integrated into the MPC controller. To approximate the effects of stochasticies in the optimal control formulation and to fulfil the probabilistic CBF condition, a sample-based approach with guarantees is employed. Furthermore, to counterbalance the additional computational burden due to sampling, a learnable terminal cost formulation is included in the MPC objective. An RL algorithm is deployed to learn both the terminal cost and the CBF constraint. Results from a numerical experiment on a constrained LTI problem corroborate the effectiveness of the proposed methodology in reducing computation time while preserving control performance and safety.

This chapter is based on [6].

3.1. Introduction

Reinforcement Learning (RL) has emerged as a successful methodology for solving complex optimal control problems, including when dealing with systems subject to uncertainty and stochastic disturbances [197]. However, employing RL in safety-critical scenarios remains in general challenging due to the inherent trial-and-error nature of the learning process and the difficulties in explicitly ensuring constraint satisfaction throughout training, even if probabilistically.

Control Barrier Functions (CBFs) have gained significant traction as an effective tool for handling safety constraints in control problems [10]. CBFs can enforce forward invariance of a safe set, thus guaranteeing safety conditions over the controlled trajectories, via an energy-based argumentation rather than relying on explicit set computations. Robust and stochastic extensions have also spun off, e.g., [8, 48], which account for uncertainties and/or disturbances affecting the system. At the same time, CBFs have been successfully integrated with various control architectures, including optimisation-based control schemes such as Model Predictive Control (MPC) [149, 171, 216, 234]. While CBFs offer guarantees on safety, their usage introduces some challenges. One of these lies in properly calibrating the CBF parameters, particularly its class $\mathcal{K}$ function. Poor tuning can severely impact the feasibility of the control problem and its closed-loop performance. Selecting an appropriate class $\mathcal{K}$ function thus involves a non-trivial trade-off between conservativeness (safety) and control performance. In this regard, adaptive formulations have been proposed in the literature that, e.g., employ auxiliary constructions [227] or leverage intelligent decision-making [171, 228] to adjust the barrier parameters autonomously.

Recently, MPC has been proposed as a promising function approximation strategy for RL algorithms, where the predictive controller acts both as policy provider and value function approximation for the underlying RL task [75, 165]. In contrast to model-free approaches, this method often results in higher sample efficiency, better interpretability and certifiability, since the MPC controller can explicitly incorporate system dynamics and systematically handle constraints. Importantly, variants of the nominal MPC formulation can also address robust and/or stochastic control problems characterised by uncertainties and/or disturbances [43, 140, 172]. Despite its benefits, the application of stochastic MPC, particularly in its sample-based forms, is often computationally demanding. One common approach to mitigate this computational complexity is to shorten the MPC prediction horizon. However, doing so can adversely affect control performance and safety due to the induced myopia of the controller. This issue is commonly addressed by introducing a terminal cost approximation, crafted to appropriately approximate the true (generally unknown) cost-to-go beyond the shortened horizon [1, 87, 102, 174]. Nonetheless, similar to the class $\mathcal{K}$ functions in CBFs, manually selecting or designing an effective terminal cost approximation introduces another trade-off between computational complexity, safety, and control performance.

In this chapter, we propose a novel approach that leverages MPC-based RL combined with probabilistic CBFs and terminal cost approximation to automatically learn from data a model-based policy that ensures probabilistic safety while maintaining computational efficiency. Our methodology integrates probabilistic CBF constraints into the MPC formulation to enforce safety despite stochastic disturbances with arbitrary

probability. A sample-based approximation is introduced to render the optimisation control problem tractable. This is distinct from, e.g., MPPI [224], where sampling is exploited to compute a Monte Carlo approximation of the optimal action sequence of the control problem. To address the computational complexity introduced by the sample-based approach and additional CBF constraint, we employ a shortened MPC prediction horizon alongside a learnable terminal cost approximation, which is automatically tuned via RL. Furthermore, the class $\mathcal{K}$ function within the CBF is also learnt from interaction data, eliminating manual tuning and enabling adaptivity. Algorithm 2 summarises the proposed approach in a compact scheme.

The main contributions of this chapter can thus be summarised as follows:

1. We introduce a stochastic MPC formulation with integrated probabilistic CBF constraints, explicitly designed to handle stochasticity in safety-critical tasks.

2. We provide a computationally efficient sample-based approximation of this formulation and propose to leverage RL to automatically learn both the terminal cost approximation and the CBF class $\mathcal{K}$ function.

3. We provide probabilistic safety guarantees and illustrate the effectiveness and computational advantages of our proposed method on a numerical example.

The remainder of this chapter is structured as follows. In Section 3.2, we review relevant background on safe RL, MPC as function approximation, and terminal cost approximations. Section 3.3 describes and analyses the proposed method. Simulation results validating our approach are presented in Section 3.4, followed by conclusions in Section 3.5.

Notation: vector and matrix quantities are in bold. Inequalities on vectors are applied element-wise. Operation $\|\boldsymbol{y}\|_{\boldsymbol{A}}$ indicates $\sqrt{\boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{y}}$, and $\boldsymbol{y} \odot \boldsymbol{z}$ the Hadamard product between $\boldsymbol{y}$ and $\boldsymbol{z}$.

3.2. Background

3.2.1. Safe reinforcement learning

Consider the discrete-time, possibly nonlinear, stochastic system

$$\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\omega}_t), \tag{3.1}$$

where, at each time step $t \in \mathbb{N}$, $\boldsymbol{s}_t \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ denotes its state, $\boldsymbol{a}_t \in \mathcal{A} \subset \mathbb{R}^{n_a}$ the control action, and $\boldsymbol{\omega}_t \in \Omega \subseteq \mathbb{R}^{n_d}$ the disturbance affecting the system. Dynamics $f : \mathcal{S} \times \mathcal{A} \times \Omega \to \mathcal{S}$ are assumed to be known and Lipschitz continuous w.r.t. $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ with constant L_f over the domain $\mathcal{S} \times \mathcal{A}$.

Assumption 3.1 (Uncertainty)**.** *Sequences $\{\boldsymbol{\omega}_\tau\}_{\tau=t}^{t+N} \sim \mathcal{W}$, for $N > 0$, are independent and identically distributed (i.i.d.) random variables with support Ω^N. Further, a sufficient number of i.i.d. samples of these sequences can be drawn from $\mathcal{W}$ or is available (e.g., from historical data).*

Algorithm 2: Stochastic safe MPC-based RL. See Section 3.3 for further details on each step below.

Input: Initial MPC parameters $\boldsymbol{\theta}^0$; violation prob. ε; full and short horizons N, $\overline{N}$; number of training episodes $n_{\max}$

Output: Learnt parametrisation $\boldsymbol{\theta}^{n_{\max}}$

1 Select number of samples M from ε, N, $\overline{N}$ empirically or via conservative bounds (3.24), (3.26)

2 Create sampled-based MPC controller (3.20) with learnable parametric CBF constraints (to ensure safety) and terminal cost approximation (to compensate for $\overline{N}$)

3 **for** $i = 0, \dots, n_{\max} - 1$ **do**

4 Perform MPC closed-loop task with current parametrisation $\boldsymbol{\theta}^i$

5 Update parametrisation to $\boldsymbol{\theta}^{i+1}$ via RL

6 **end**

For a deterministic policy $\pi_{\boldsymbol{\theta}} : \mathscr{S} \to \mathscr{A}$, parametrised in $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^{n_{mathbf\theta}}$, we define its performance as[1]

$$J(\pi_{\boldsymbol{\theta}}) := \mathbb{E}_{\chi_{\pi_{\boldsymbol{\theta}}}}\left[\sum_{t=0}^{T} \ell\big(\boldsymbol{s}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t)\big)\right], \tag{3.2}$$

where $T \in \mathbb{N}$ is the horizon, $\ell : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ is a stage cost function, and $\chi_{\pi_{\boldsymbol{\theta}}}$ the state distribution the policy induces. In safe RL, we are primarily interested in finding a policy that optimises the performance while providing safe trajectories with high probability, i.e.,

$$\pi^\star_{\boldsymbol{\theta}} \in \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\Theta} \quad \left\{ J(\pi_{\boldsymbol{\theta}}) : \mathbb{P}\left(\bigcap_{t=0}^{T} \boldsymbol{s}_t \in \mathscr{C}\right) \geq 1 - \varepsilon \right\}, \tag{3.3}$$

where $\mathscr{C} = \{\boldsymbol{s} \in \mathscr{S} \mid h(\boldsymbol{s}) \geq 0\}$ denotes the desired safe set, defined by $h : \mathscr{S} \to \mathbb{R}$, a Lipschitz continuous function in $\mathscr{S}$ with constant L_h, and $\varepsilon \in (0,1)$ the confidence level for the joint chance constraint. Note that, due to the presence of stochastic disturbances, also the state becomes a random variable and generally cannot be constrained to satisfy h with unit probability without further assumptions (e.g., boundedness of the support of $\boldsymbol{\omega}_t$).

The familiar notions of state- and action-value functions [197] apply here as well:

$$V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) = \mathbb{E}_{\chi_{\pi_{\boldsymbol{\theta}}}}\left[\sum_{\tau=t}^{T} \ell\big(\boldsymbol{s}_\tau, \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_\tau)\big)\right], \tag{3.4}$$

$$Q_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \ell(\boldsymbol{s}_t, \boldsymbol{a}_t) + \mathbb{E}_{\boldsymbol{\omega}_t}\big[V_{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1})\big]. \tag{3.5}$$

3.2.2. MPC as function approximation in RL

To parametrise the policy $\pi_{\boldsymbol{\theta}}$ and deploy an RL agent, (deep) neural networks are oftentimes the most common choice [12]. However, model-free approaches generally suffer

[1] For simplicity, we address in this chapter the finite-horizon undiscounted setting, but our results can be easily extended to the infinite-horizon discounted case.

from several drawbacks, as discussed in Section 3.1. In this chapter, a model-based solution to (3.3), which leverages MPC as the function approximation scheme, is instead pursued.

Given the current state $\boldsymbol{s}_t$, consider the MPC scheme

$$\begin{aligned}
\min_{\{\boldsymbol{u}_k\}_{k=0}^{N-1}} \quad & \lambda_{\boldsymbol{\theta}}(\boldsymbol{x}_0) + \mathbb{E}\left[\sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\boldsymbol{x}_N)\right] && \text{(3.6a)}\\
\text{s.t.} \quad & \boldsymbol{u}_k \in \mathscr{A}, && k = 0, \dots, N-1, && \text{(3.6b)}\\
& \boldsymbol{x}_0 = \boldsymbol{s}_t, && && \text{(3.6c)}\\
& \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k), && k = 0, \dots, N-1, && \text{(3.6d)}\\
& \{\boldsymbol{\omega}_k\}_{k=0}^{N-1} \sim \mathscr{W}, && && \text{(3.6e)}\\
& \mathbb{P}\left(\bigcap_{k=1}^{N} \boldsymbol{x}_k \in \mathscr{C}\right) \geq 1 - \varepsilon, && && \text{(3.6f)}
\end{aligned}$$

where $N \geq 0$ is the prediction horizon, $\ell_{\boldsymbol{\theta}} : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ and $\lambda_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}^{\mathrm{f}} : \mathscr{S} \to \mathbb{R}$ the stage, initial, and final cost approximations respectively. This scheme serves as the approximation of the value function as

$$V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) = \min_{\{\boldsymbol{u}_k\}_{k=0}^{N-1}} \{(3.6a) : (3.6b) - (3.6f)\}, \tag{3.7}$$

and it satisfies the Bellman equations so that

$$Q_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \min_{\{\boldsymbol{u}_k\}_{k=0}^{N-1}} \{(3.6a) : (3.6b) - (3.6f),\ \boldsymbol{u}_0 = \boldsymbol{a}_t\}, \tag{3.8}$$

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t) = \boldsymbol{u}_0^\star = \arg\min_{\{\boldsymbol{u}_k\}_{k=0}^{N-1}} \{(3.6a) : (3.6b) - (3.6f)\}. \tag{3.9}$$

It was first shown in [75] that the solution to an MPC optimisation problem can approximate the optimal value function. This is especially intuitive if the MPC horizon N were to approach the task horizon T and we would take $\ell_{\boldsymbol{\theta}} = \ell$ and $\lambda_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}^{\mathrm{f}} = 0$. However, in general, long prediction horizons and the stochastic arguments in (3.6) massively hinder the tractability of the MPC problem. In Section 3.3, we present our approach to circumvent both issues in the context of safe RL. While in general, in addition to $V_{\boldsymbol{\theta}}^{\mathrm{f}}$, it is beneficial to have both $\ell_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\theta}}$ parametrised to increase the number of degrees of freedom of the approximation scheme [75], in what follows we propose to only focus on $V_{\boldsymbol{\theta}}^{\mathrm{f}}$ for computational relief and control performance.

3.2.3. Cost-to-go approximation

A proper choice of terminal cost $V_{\boldsymbol{\theta}}^{\mathrm{f}}$ in (3.6a) is essential in capturing the cost-to-go for the terminal state $\boldsymbol{x}_N$. In general, analytical forms of the true cost-to-go are often unavailable, and approximations must be used instead. As discussed in Section 3.1, the control literature offers various solutions to this challenge. In particular, in this chapter, we highlight the following approaches from the literature.

Nonconvex case

for a nonconvex system and stage cost, the cost-to-go approximation can be parametrised as in [1]:

$$\boldsymbol{P}_{\boldsymbol{\theta}}(\boldsymbol{c}) = L_{\boldsymbol{\theta}}(\boldsymbol{c})L_{\boldsymbol{\theta}}^{\top}(\boldsymbol{c}), \tag{3.10}$$

$$V_{\boldsymbol{\theta}}^{\mathrm{f,psd}}(\boldsymbol{x},\boldsymbol{c}) = \left\|\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\theta}}^{\mathrm{f}}(\boldsymbol{c})\right\|_{\boldsymbol{P}_{\boldsymbol{\theta}}(\boldsymbol{c})}^{2}, \tag{3.11}$$

3

where $\boldsymbol{c} \in \mathbb{R}^{n_c}$ is the task-relevant context available at the current time step (which can include any information, e.g., state $\boldsymbol{x}$, previous actions, references, etc.), and both $\boldsymbol{x}_{\boldsymbol{\theta}}^{\mathrm{f}} : \mathbb{R}^{n_c} \to \mathbb{R}^{n_s}$ and $L_{\boldsymbol{\theta}} : \mathbb{R}^{n_c} \to \mathbb{R}^{n_s \times n_s}$ are represented by two neural networks (NNs), whose parameters are meant as included in $\boldsymbol{\theta}$. In particular, $L_{\boldsymbol{\theta}}(\boldsymbol{c})$ from (3.10) is a lower triangular matrix with only $\frac{1}{2}n_s(n_s+1)$ free entries. This Cholesky decomposition-like form allows the approximate terminal cost to be positive semidefinite (PSD) w.r.t. $\boldsymbol{x}$ by construction. For a fixed $\boldsymbol{c}$, this makes optimising over the ensuing quadratic form relatively easy and cheap. At the same time, the quadratic form is context-dependent, meaning its value and gradient information will change from time step to time step, making the approximation also time-dependent. Lastly, the approach is quite malleable as the two NNs can be seamlessly scaled down or up as needed.

Convex case

in the case of constrained linear time-invariant systems with quadratic regulation cost, it is well-known that the optimal value function is convex piecewise quadratic (PWQ) [18]. This result can also be extended to the stochastic setting with zero-mean, time-uncorrelated Gaussian disturbances [121]. In such a case where the value function is known to have (exactly or even approximately) a convex PWQ shape, [87] suggests the use of the approximation

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \mathrm{ReLU}\left(\boldsymbol{W}_{\boldsymbol{\theta}}\boldsymbol{x} + \boldsymbol{b}_{\boldsymbol{\theta}}\right), \tag{3.12}$$

$$V_{\boldsymbol{\theta}}^{\mathrm{f,pwq}}(\boldsymbol{x}) = \boldsymbol{w}_{\boldsymbol{\theta}}^{\top}\left(\boldsymbol{\varphi}(\boldsymbol{x}) \odot \boldsymbol{\varphi}(\boldsymbol{x})\right), \tag{3.13}$$

where $\boldsymbol{W}_{\boldsymbol{\theta}} \in \mathbb{R}^{m \times n_s}$, $\boldsymbol{b}_{\boldsymbol{\theta}} \in \mathbb{R}^{m}_{<0}$ and $\boldsymbol{w}_{\boldsymbol{\theta}} \in \mathbb{R}^{m}_{\geq 0}$ are the adjustable weights and biases of the NN, and $\boldsymbol{\varphi} \in \mathbb{R}^{m}$ its hidden features. By enforcing $\boldsymbol{b}_{\boldsymbol{\theta}} < 0$ and $\boldsymbol{w}_{\boldsymbol{\theta}} \geq 0$, it is shown in [87] that this function is PWQ and convex w.r.t. $\boldsymbol{x}$. The advantage of this approximation lies in its scalability (by appropriately selecting the hidden size m) and ability to represent any PWQ convex functions by construction.

3.3. Methodology

This section introduces a sample-based approximation to the stochastic MPC problem. Similarly to, e.g., [1, 87], we propose to employ a learnable terminal cost formulation, coupled with a short prediction horizon, to mitigate the computational complexity induced by the sampling approach. At the same time, to preserve the probabilistic safety of the closed-loop state trajectories despite the increased myopia of the controller, we leverage the notion of CBF to guarantee step-wise forward invariance of the safe set with high probability. We adopt RL to perform training of both the terminal cost and the CBF class $\mathcal{K}$ function in an end-to-end fashion.

3.3.1. Probabilistic control barrier function

In this chapter, we propose to leverage the CBF framework to guarantee safety. Again, it is essential to remark that, due to the stochasticity affecting the system, in general safety cannot be guaranteed with unit probability. Rather, we will take a probabilistic approach.

Definition 3.1 (N-Step ε-Control Invariant Set [69])**.** *A set $\mathscr{Q} \subseteq \mathbb{R}^{n_s}$ is N-step ε-control invariant for system (3.1) if, for any $\boldsymbol{s}_t \in \mathscr{Q}$, there exists a control policy such that*

$$\mathbb{P}\left(\bigcap_{\tau=1}^{N} \boldsymbol{s}_{t+\tau} \in \mathscr{Q}\right) \geq 1-\varepsilon. \tag{3.14}$$

Definition 3.2 (Probabilistic Control Barrier Function [216])**.** *For system (3.1) and safe set $\mathscr{C} \subseteq \mathscr{S}$, the continuous function $h : \mathscr{S} \to \mathbb{R}$ is a discrete-time probabilistic CBF if there exist a class $\mathscr{K}$ function $\alpha : [0, a) \to [0, \infty)$, $\alpha(y) \leq y$, $\forall y \geq 0$, and a control action $\boldsymbol{a}_t \in \mathscr{A}$ such that, with $\xi \in [0, 1)$, it holds that*

$$\mathbb{P}\big(h(\boldsymbol{s}_{t+1}) - h(\boldsymbol{s}_t) \geq -\alpha\big(h(\boldsymbol{s}_t)\big)\big) \geq 1-\xi, \quad \forall t \in \mathbb{N}. \tag{3.15}$$

Note that the above follows straightforwardly from the standard CBF definition, on top of which the probability operator $\mathbb{P}$ has been applied since the state is now a random variable. Now, we can state a result on the step-wise probabilistic invariance guarantee for the set $\mathscr{C}$ thanks to the CBF condition.

Theorem 3.1. *Given a safe set $\mathscr{C} \subseteq \mathscr{S}$ defined by the continuous function $h : \mathscr{S} \to \mathbb{R}$ and current state $\boldsymbol{s}_t \in \mathscr{C}$, if h is a discrete-time probabilistic CBF, any control policy satisfying (3.15) with $\xi \leq \frac{\varepsilon}{N}$ renders the set $\mathscr{C}$ N-step ε-control invariant.*

Proof. The proof is similar to that of [216, Theorem 2]. By complement, the joint safety condition along an N-step trajectory is satisfied as long as

$$\mathbb{P}\left(\bigcup_{\tau=1}^{N} \boldsymbol{s}_{t+\tau} \notin \mathscr{C}\right) \leq \varepsilon. \tag{3.16}$$

Applying the union bound, we get that

$$\mathbb{P}\left(\bigcup_{\tau=1}^{N} \boldsymbol{s}_{t+\tau} \notin \mathscr{C}\right) \leq \sum_{\tau=1}^{N} \mathbb{P}(\boldsymbol{s}_{t+\tau} \notin \mathscr{C}). \tag{3.17}$$

To ensure the joint probability of violation is at most ε, it is thus sufficient to require $\sum_{\tau=1}^{N} \mathbb{P}(\boldsymbol{s}_{t+\tau} \notin \mathscr{C}) \leq \varepsilon$. The simplest choice is to allocate the risk uniformly per time step, i.e., we need to satisfy

$$\mathbb{P}(\boldsymbol{s}_{t+\tau} \in \mathscr{C}) \geq 1 - \frac{\varepsilon}{N}, \quad \tau = 1, \dots, N. \tag{3.18}$$

To achieve this, we select $\xi \leq \frac{\varepsilon}{N}$ and compute the action at each time step $t+\tau$ according to (3.15), so that

$$\begin{aligned} \mathbb{P}(h(\boldsymbol{s}_{t+\tau+1}) \geq 0) &\geq \mathbb{P}\big(h(\boldsymbol{s}_{t+\tau+1}) \geq h(\boldsymbol{s}_{t+\tau}) - \alpha\big(h(\boldsymbol{s}_{t+\tau})\big)\big) \\ &\geq 1-\xi \geq 1 - \frac{\varepsilon}{N}, \quad \tau = 1, \dots, N-1, \end{aligned} \tag{3.19}$$

where the first inequality leverages the fact that $\boldsymbol{s}_t \in \mathscr{C} \Rightarrow h(\boldsymbol{s}_t) \geq 0$, and the property $\alpha(y) \leq y$, $\forall y \geq 0$. □

This result implies that, if the control policy acts accordingly to (3.15) with ξ properly selected as $\frac{\varepsilon}{N}$, the state trajectory can occasionally leave the safe set $\mathscr{C}$ but the chance of doing so is bounded by ε. Note that, while leveraging the CBF condition is beneficial to safety, there are still some open issues. In particular, finding a control input directly via (3.15) is in general challenging due to the probability operator: a distributional characterisation of its argument may be challenging due to the possible nonlinear nature of f, h, and/or α, and would also require exact knowledge of the distribution $\mathscr{W}$. Additionally, it is well-known that properties of the ensuing control policy (such as performance) are dependent on the selection of a proper class $\mathscr{K}$ function.

3.3.2. Sample-based MPC approximation

We can now introduce the proposed sample-based approximation of (3.6). Let us introduce a shortened horizon $\overline{N} \ll N$. At time step t, assume M samples $\{\boldsymbol{\omega}_\tau^{(i)}\}_{\tau=t}^{t+\overline{N}-1}$, $i = 1,\dots,M$, are available (see Assumption 3.1).[2] Then, we can replace the original intractable formulation (3.6) with the following scheme:

$$\min_{\{\boldsymbol{u}_k\}_{k=0}^{\overline{N}-1}} \quad \lambda_{\boldsymbol{\theta}}(\boldsymbol{s}_t) + \frac{1}{M}\sum_{i=1}^{M}\left[\sum_{k=0}^{\overline{N}-1} \ell_{\boldsymbol{\theta}}\big(\boldsymbol{x}_k^{(i)}, \boldsymbol{u}_k\big) + V_{\boldsymbol{\theta}}^{\mathrm{f}}\big(\boldsymbol{x}_{\overline{N}}^{(i)}\big)\right] \tag{3.20a}$$

$$\text{s.t.} \quad \boldsymbol{u}_k \in \mathscr{A}, \qquad k = 0,\dots,\overline{N}-1, \tag{3.20b}$$

$$\boldsymbol{x}_0^{(i)} = \boldsymbol{s}_t, \qquad i = 1,\dots,M, \tag{3.20c}$$

$$\boldsymbol{x}_{k+1}^{(i)} = f\big(\boldsymbol{x}_k^{(i)}, \boldsymbol{u}_k, \boldsymbol{\omega}_k^{(i)}\big), \qquad i = 1,\dots,M,\ k = 0,\dots,\overline{N}-1, \tag{3.20d}$$

$$h\big(\boldsymbol{x}_{k+1}^{(i)}\big) - h\big(\boldsymbol{x}_k^{(i)}\big) \geq \zeta - \alpha_{\boldsymbol{\theta}}\Big(h\big(\boldsymbol{x}_k^{(i)}\big)\Big), \qquad i = 1,\dots,M,\ k = 0,\dots,\overline{N}-1. \tag{3.20e}$$

Major differences lie in the safety condition (3.6f) being replaced by the proposed probabilistic CBF formulation (3.20e), and the probabilistic operators, e.g., the expectation in (3.6a), by sample approximation. By selecting a (much) shorter horizon, we are able to counterbalance the increased size of the optimisation problem. However, myopic policies tend to be less safe. For this reason, we leverage the CBF to ensure control invariance. Still, to avoid jeopardising safety due to the sample-based approximation, we stress that the number of samples M must be selected in such a way to guarantee that the CBF condition is satisfied with probability $1 - \frac{\varepsilon}{N}$. In what follows, we discuss how to select M in order to achieve this, thus preserving safety with confidence ε. Note that $\zeta \geq 0$ in (3.20e) is a (usually small) scalar required to ensure the probabilistic guarantees discussed below. It is trivial to check that, being nonnegative, its presence does not jeopardise the CBF validity. If the problem (3.20) is convex, ζ can be freely set to zero; for the generic nonconvex case, $\zeta \neq 0$ (see proof of Theorem 3.2).

[2] Since disturbances could be time-correlated, these samples must be drawn by sampling whole sequences from $\mathscr{W}$ and then considering only the first $\overline{N}-1$ elements.

Assumption 3.2 (Recursive Feasibility)**.** *Under the ensuing control policy, the sampled-based MPC scheme* (3.20) *admits a feasible solution at every time step $t \in \mathbb{N}$ almost surely.*

This assumption is a requirement for the following result, and is standard in other works, e.g., [43, 172]. At first, it might appear restrictive but in practice hard constraints are often replaced by soft constraints in stochastic/learning settings. This choice is corroborated by the probabilistic nature of the control problem, i.e., violations cannot be avoided with unit probability (without further assumptions). Furthermore, this choice is also helpful in the context of RL: during learning, it is beneficial for the RL agent to violate constraints occasionally and receive appropriate penalties so as to learn better to discern safe and unsafe behaviours [75].

Theorem 3.2. *Given a confidence parameter $\beta \in (0,1)$, there exists a minimum number of samples M for which the solution $\{\boldsymbol{u}_k^\star\}_{k=0}^{\overline{N}-1}$ to the sample-based optimisation problem* (3.20) *satisfies*

$$\mathbb{P}\left(\bigcap_{k=0}^{\overline{N}-1} h\left(\boldsymbol{x}_{k+1}^\star\right) - h\left(\boldsymbol{x}_k^\star\right) \geq \zeta - \alpha\left(h\left(\boldsymbol{x}_k^\star\right)\right)\right) \geq 1 - \frac{\varepsilon}{N} \tag{3.21}$$

with probability no smaller than $1-\beta$, where $\boldsymbol{x}_0^\star = \boldsymbol{s}_t$ and $\boldsymbol{x}_{k+1}^\star = f\left(\boldsymbol{x}_k^\star, \boldsymbol{u}_k^\star, \boldsymbol{\omega}_k\right)$.

Proof. For sake of brevity, for any feasible solution to (3.20) we drop the implicit dependency of $\{\boldsymbol{x}_k\}_{k=0}^{\overline{N}}$ on $\{\boldsymbol{u}_k\}_{k=0}^{\overline{N}-1}$. Given $\boldsymbol{s}_t$, define the violation probability of a solution as

$$V_{\boldsymbol{u}} = \mathbb{P}\left(\bigcup_{k=0}^{\overline{N}-1} h\left(\boldsymbol{x}_{k+1}\right) - h\left(\boldsymbol{x}_k\right) < \zeta - \alpha\left(h\left(\boldsymbol{x}_k\right)\right)\right). \tag{3.22}$$

Take $\xi = \frac{\varepsilon}{N}$. Analogously to Section 3.2.3, we distinguish between two cases.

In case (3.20) is nonconvex, let $d_{\mathscr{A}} = \sup_{\boldsymbol{a},\boldsymbol{a}' \in \mathscr{A}} \|\boldsymbol{a} - \boldsymbol{a}'\|_\infty$ be the diameter of $\mathscr{A}$. Because $\alpha(y) \leq y$, $\forall y \geq 0$, and α is strictly increasing, α is Lipschitz continuous with constant 1. Given the Lipschitz constants of f and h, the CBF constraint (3.20e) is also Lipschitz continuous with constant at most $L_{\text{CBF}} = L_h L_f + L_h + L_h$. By [128, Theorem 10] we have

$$\mathbb{P}(V_{u^\star} > \xi) \leq \left\lceil \frac{2}{\xi} \right\rceil \left\lceil \frac{2 d_{\mathscr{A}} L_{\text{CBF}}}{\zeta} \right\rceil^{\overline{N} n_a} e^{-\frac{1}{2} M \xi^2}. \tag{3.23}$$

By requiring that the right-hand side be $\leq \beta$, we get

$$M \geq \frac{2}{\xi^2}\left(\ln \beta^{-1} + \overline{N} n_a \ln \left\lceil \frac{2 d_{\mathscr{A}} L_{\text{CBF}}}{\zeta} \right\rceil + \ln \left\lceil \frac{2}{\xi} \right\rceil\right). \tag{3.24}$$

Let us tackle the special case in which (3.20) is convex w.r.t. its decision variables (it needs not be convex w.r.t. the disturbance). Contrarily to the previous case, here we can consider $\zeta = 0$ as it is not required. The scenario approach theory [42, 43, 172] shows that the probability of violation at the optimal solution of (3.20) is (possibly tightly) bounded by [42, Theorem 1]

$$\mathbb{P}(V_{\boldsymbol{u}^\star} > \xi) \leq \sum_{j=0}^{\overline{N} n_a - 1} \binom{M}{j} \xi^j (1-\xi)^{M-j}. \tag{3.25}$$

By requiring that the right-hand side be $\leq \beta$, we obtain

$$M \geq \frac{2}{\xi}\left(\ln \beta^{-1} + \overline{N} n_a\right). \tag{3.26}$$

□

Despite of arguably limited applicability, this theorem importantly confirms the intuition that, as the sample size M increases, the confidence at which the safety condition is satisfied increases. Also, note that, while the horizon $\overline{N}$ has been shrunk to combat the computational complexity due to the sampling scheme, the probabilistic safety condition has been left untouched and is still imposed over the original N-step trajectory (see right-hand side of (3.21) where the risk of violation is allocated over N steps instead of $\overline{N}$).

3.3.3. RL algorithm

Note that most of the major components in (3.20) are parametrised in $\boldsymbol{\theta}$, including the class $\mathcal{K}$ function $\alpha_{\boldsymbol{\theta}}$ and the terminal cost function $V^{\mathrm{f}}_{\boldsymbol{\theta}}$. We propose to adjust this parametrisation in closed loop via an MPC-based RL algorithm [75]. This approach solves the original safe RL problem (3.3) as the safety constraint is taken into account into the MPC function approximation while the performance cost (3.2) is minimised by a gradient-based RL method. Among the advantages of this approach is the fact that it bypasses the need to manually craft and select the parametrised components, which are instead adjusted by RL via interactions with the environment. This encompasses the ability also to learn $\alpha_{\boldsymbol{\theta}}$, yielding an intrinsically adaptive CBF that can automatically balance the trade-off between trajectory safety and control performance.

Because we explicitly include a learnable terminal cost term in the objective, a value-based method is leveraged here. In particular, we propose the use of Q-learning [221]. Briefly, Q-learning indirectly finds the optimal policy by solving

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}\big[\|\ell(\boldsymbol{s},\boldsymbol{a}) + V_{\boldsymbol{\theta}'}(\boldsymbol{s}_+) - Q_{\boldsymbol{\theta}}(\boldsymbol{s},\boldsymbol{a})\|^2\big], \tag{3.27}$$

where $V_{\boldsymbol{\theta}}$ and $Q_{\boldsymbol{\theta}}$ are defined in (3.7) and (3.8). The problem can be minimised via, e.g., gradient descent, with $V_{\boldsymbol{\theta}'}$ evaluated for a frozen copy of the parameters $\boldsymbol{\theta}' = \boldsymbol{\theta}$ to prevent value estimates from contributing to the gradient update. The update thus reads as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\delta\nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s},\boldsymbol{a}), \tag{3.28}$$

with $\eta > 0$ a properly selected learning rate and δ the temporal difference error. For the computation of $\nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$, while not straightforward, nonlinear sensitivity analysis of the MPC scheme (3.6) shows that this sensitivity coincides with the partial derivative of the Lagrangian w.r.t. $\boldsymbol{\theta}$ at the optimal primal-dual solution [35]. Details on the implementation can be found in, e.g., [4, 233].

3.4. Numerical experiment

In this section, we test the proposed methodology on a numerical case. The experiment was implemented in Python 3.12.6 and conducted on a server with 16 AMD EPYC 7252

(3.1 GHz) processors and 252GB RAM. The optimisation problems were formulated with CasADi [11], and solved via Gurobi [82]. Source code and results are available in the following repository: `https://github.com/FilippoAiraldi/mpcrl-cbf`.

3.4.1. Problem description

Consider the stochastic LTI system $f(\boldsymbol{s}_t, \boldsymbol{a}_t, \omega_t) = \boldsymbol{A}\boldsymbol{s}_t + \boldsymbol{B}\boldsymbol{a}_t + \boldsymbol{E}\omega_t$ with

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0.4 \\ -0.1 & 1 \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} 1 & 0.05 \\ 0.5 & 1 \end{bmatrix}, \ \boldsymbol{E} = \begin{bmatrix} 0.03 \\ 0.01 \end{bmatrix}, \tag{3.29}$$

where the disturbances are time-uncorrelated zero-mean normally distributed, i.e., $\mathbb{E}[\omega_t] = 0$ and $\mathbb{E}\left[\omega_i \omega_j\right] \propto \delta(i-j)$. The control space is $\mathscr{A} = \left\{\boldsymbol{a} \in \mathbb{R}^2 : \|\boldsymbol{a}\|_\infty \leq 0.5\right\}$. The safe set is defined as $\mathscr{C} = \left\{\boldsymbol{s} \in \mathbb{R}^2 : \|\boldsymbol{s}\|_\infty \leq 3\right\}$, where the infinity-norm state constraint is turned into four separate CBFs $h_j : \mathbb{R}^2 \to \mathbb{R}$, $j = 1, \dots, 4$, i.e., $h_{2i-1}(\boldsymbol{s}) = 3 - s_i$ and $h_{2i}(\boldsymbol{s}) = 3 + s_i$, $i = 1, 2$. The RL stage cost includes quadratic terms alongside penalties for the violation of the safety condition:

$$\ell(\boldsymbol{s}, \boldsymbol{a}) = \|\boldsymbol{s}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}\|_{\boldsymbol{R}}^2 - c \sum_{j=1}^{4} \min\left\{0, h_j(\boldsymbol{s})\right\}, \tag{3.30}$$

with $\boldsymbol{Q} = \boldsymbol{I}_{2\times 2}$, $\boldsymbol{R} = 0.1\boldsymbol{I}_{2\times 2}$, and $c = 10^3$. The length of a single episode is set to $T = 30$ time steps.

3.4.2. MPC and RL implementation

Given the current state $\boldsymbol{s}_t$, the following unit-horizon MPC scheme derived from (3.6) and (3.20) is employed as function approximation with $M = 32$ samples[3]:

$$\min_{\boldsymbol{u}_0, \Sigma} \quad \ell(\boldsymbol{s}_t, \boldsymbol{u}_0) + \frac{1}{M} \sum_{i=1}^{M} \left[c \sum_{j=1}^{4} \sigma_j^{(i)} + V_{\boldsymbol{\theta}}^{\mathrm{f,pwq}}\left(\boldsymbol{x}_1^{(i)}\right) \right] \tag{3.31a}$$

$$\text{s.t.} \quad -0.5\boldsymbol{u}_0 \leq 0.5, \tag{3.31b}$$

$$\boldsymbol{x}_1^{(i)} = f\left(\boldsymbol{s}_t, \boldsymbol{u}_0, \boldsymbol{\omega}_0^{(i)}\right), \qquad i = 1, \dots, M, \tag{3.31c}$$

$$h_j\left(\boldsymbol{x}_1^{(i)}\right) - \left(1 - \gamma_{\boldsymbol{\theta},j}\right) h_j(\boldsymbol{s}_t) + \sigma_j^{(i)} \geq 0, \quad j = 1, \dots, 4, \ i = 1, \dots, M, \tag{3.31d}$$

$$\sigma_j^{(i)} \geq 0, \qquad j = 1, \dots, 4, \ i = 1, \dots, M, \tag{3.31e}$$

is employed as function approximation with $M = 32$ samples. As terminal cost approximation in (3.31a), the convex PWQ function $V_{\boldsymbol{\theta}}^{\mathrm{f,pwq}}$ is employed with a hidden size of 16 neurons. For each CBF constraint (3.31d), the corresponding class $\mathscr{K}$ function is parametrised linearly, i.e., $\alpha_{\boldsymbol{\theta},j}(y) = \gamma_{\boldsymbol{\theta},j} y$, $j = 1, \dots, 4$, where $\gamma_{\boldsymbol{\theta},j} \in [0, 1]$ is an adjustable scalar value. The whole MPC learnable parametrisation is therefore

$$\boldsymbol{\theta} = \left(\boldsymbol{W}_{\boldsymbol{\theta}}, \boldsymbol{b}_{\boldsymbol{\theta}}, \boldsymbol{w}_{\boldsymbol{\theta}}, \gamma_{\boldsymbol{\theta},1}, \dots, \gamma_{\boldsymbol{\theta},4}\right), \tag{3.32}$$

[3] Although not selected according to (3.26) (which in principle provides a conservative bound on the number of samples required), this M already leads to a sufficiently safe and computationally not-too-expensive policy in our test environment.

where $\boldsymbol{W}_{\boldsymbol{\theta}}$, $\boldsymbol{b}_{\boldsymbol{\theta}}$ and $\boldsymbol{w}_{\boldsymbol{\theta}}$ are defined per Section 3.2.3. Note that the CBF constraints (3.31d) have been relaxed via slack variables $\Sigma = \{\sigma_j^{(i)}, i = 1,\dots,M, j = 1,\dots,4\}$ to preserve feasibility (see Assumption 3.2), and, since the problem is convex, we set $\zeta = 0$.

The MPC parametrisation is initialised to uniformly random values for the PWQ NN, and to $\gamma_{\boldsymbol{\theta},j} = 0.7$ for all the CBF parameters. A Q-learning agent is trained for 1000 episodes with a learning rate of 0.005 via *rmsprop* [200]. The parametrisation is updated at the end of each episode, based on the experiences observed in the last episode. Since $\gamma_{\boldsymbol{\theta},j}$ and part of the parametrisation of the PWQ NN must be constrained, a constrained step update of $\boldsymbol{\theta}$ is performed [4]. To induce exploration in an epsilon-greedy fashion, a term $\boldsymbol{q}^\top \boldsymbol{u}_0$ is added to the objective (3.31a), where $\boldsymbol{q} \sim \mathcal{N}\left(\boldsymbol{0}_{2\times 2}, \rho_{\boldsymbol{q}} \boldsymbol{I}_{2\times 2}\right)$. The exploration scale $\rho_{\boldsymbol{q}}$ and probability both start at 1 but decay by a factor of 0.997 after each episode. Lastly, the training procedure is repeated for 10 differently randomly seeded agents to account for randomness.

3.4.3. Results

Numerical results corroborate the capability of the proposed framework in appropriately learning the terminal cost MPC component via RL, as well as the effectiveness of the learnt policy compared to a full-length horizon MPC controller. Figure 3.1 shows the evolution of the PWQ approximation w.r.t. the explicit optimal solution, computed in accordance to [121]. Convergence of both the normalised error and the R^2 coefficient during training provides empirical evidence that the Q-learning algorithm is able to steer the $V_{\boldsymbol{\theta}}^{\mathrm{f}}$ term towards the real optimal one. Figure 3.2 reports the evolution of the CBF

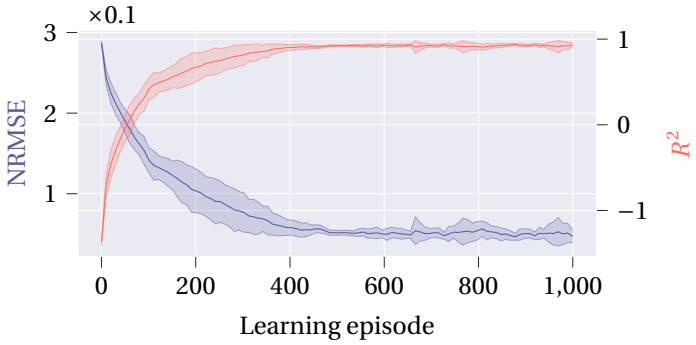

Figure 3.1.: Evolution of the learnt terminal cost approximation in terms of normalised RMSE and coefficient of determination w.r.t. the optimal cost-to-go function for the constrained stochastic LTI experiment. Average results ± one standard deviation over 10 different seeds are reported.

parameters $\gamma_{\boldsymbol{\theta},1}$ and $\gamma_{\boldsymbol{\theta},3}$, which correspond to the lower and upper bounds on the first state. These are of more interest because, due to the dynamics, most of the violations tend to occur in these two constraints (the other two parameters $\gamma_{\boldsymbol{\theta},2}$ and $\gamma_{\boldsymbol{\theta},4}$ are omitted as they do not change as much during learning). It is important to stress again that these CBF parameters (as well as all other parameters included in $\boldsymbol{\theta}$) are adjusted by

Q-learning to enhance closed-loop performance. Because constraint violations are included in the cost (3.30) as penalty term, safety is only indirectly taken into account by the RL algorithm. Nonetheless, since the parameters $\gamma_{\boldsymbol{\theta},j}$ are constrained to the interval $[0,1]$ in each update, the CBFs remain valid throughout the learning process. As a matter of fact, during training, the MPC-based RL policy achieves a small empirical probability ($0.0942 \pm 0.00483\%$) of violating any constraint.

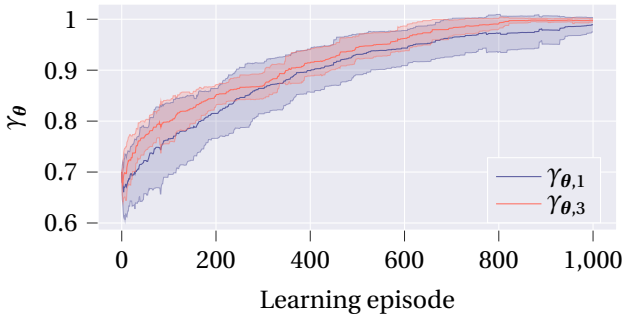

Figure 3.2.: Evolution of two of the linear class $\mathscr{K}$ function learnable coefficients. Average results ± one standard deviation over 10 different seeds are reported.

After the training phase, the learnt MPC-based RL policy is evaluated against a full-length horizon stochastic MPC policy. The latter is similar to (3.31) but is fixed (i.e., it contains no learnable terms) and has a horizon of 12 (instead of 1), which was found to be sufficient to achieve the lowest closed-loop cost (see, e.g., [47], for a more thorough discussion on how to find such a horizon). The other hyperparameters, e.g., the number of samples M, are the same in both policies. For each evaluation episode, the initial conditions are drawn from the boundary of the maximal invariant set. Figure 3.3 shows the outcomes of this evaluation comparison. Unsurprisingly, CPU time spent online in solving the MPC-RL policy is almost two orders of magnitude shorter than that for the fixed MPC controller, thanks to the corresponding optimisation problem being considerably smaller. However, from the point of view of costs, both policies achieve remarkably similar closed-loop performance despite the difference in horizon lengths. This finding is further validated in Figure 3.4, which reports ten state trajectories that highlight how both control policies behave rather similarly. Moreover, both policies exhibit comparable empirical constraint violation probabilities at evaluation ($0.0839 \pm 0.0138\%$ and $0.1 \pm 0.014\%$, respectively). These probabilities are also in line with the constraint violation probability recorded during training.

3.5. Conclusions

We have proposed a control methodology for stochastic safety-critical systems that merges MPC, CBFs and RL. The parametric MPC controller acts as the backbone, providing the control policy and value function approximation for the RL task. A probabilistic CBF formulation, integrated in the MPC scheme, is put in place to ensure safety of state trajectories with arbitrary probability. To retain tractability of the optimisation

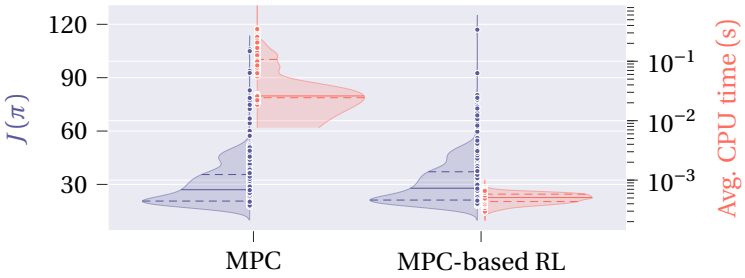

Figure 3.3.: Comparison between the non-learning MPC policy (horizon of 12) and the learnt MPC-based RL policy (unit horizon) in terms of the total incurred cost and average solver time over different 1000 episode trials. Lines represent the second (solid) and first and third (dashed) quartiles.

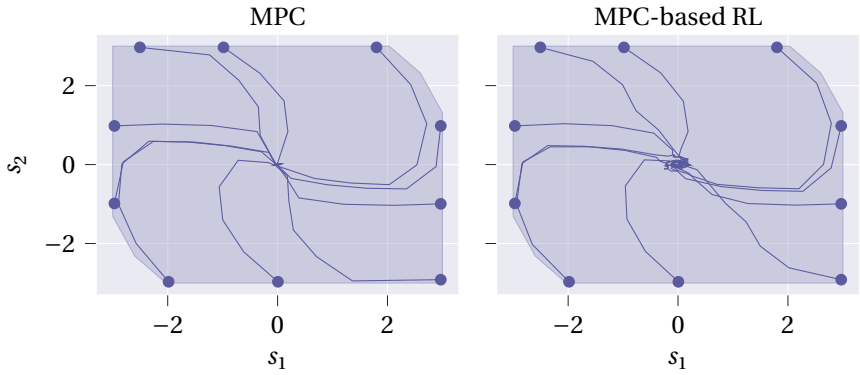

Figure 3.4.: Ten examples of state trajectories recorded during the evaluation of the non-learning MPC policy (horizon of 12) against the learnt MPC-based RL policy (unit horizon). Also reported is the maximal invariant set.

problem, the MPC horizon is (substantially) shrunk and a learnable terminal cost is introduced to combat performance drops. RL is then used to adjust the parametrisation of this learnable cost as well as the class $\mathcal{K}$ function, automatically tuning the MPC parametrisation to achieve higher closed-loop performance. A numerical example on a constrained LTI environment showcases the proposed method. Future work will investigate the use of more complex CBF parametrisations (e.g., neural network-based), as well as applications of the proposed methodology to nonlinear systems.

4

Multi-agent reinforcement learning via distributed MPC as a function approximator

This chapter presents a novel approach to multi-agent reinforcement learning (RL) for linear systems with convex polytopic constraints. Existing work on RL has demonstrated the use of model predictive control (MPC) as a function approximator for the policy and value functions. The current chapter is the first work to extend this idea to the multi-agent setting. We propose the use of a distributed MPC scheme as a function approximator, with a structure allowing for distributed learning and deployment. We then show that Q-learning updates can be performed distributively without introducing nonstationarity, by reconstructing a centralised learning update. The effectiveness of the approach is demonstrated on a numerical example.

This chapter is based on [134].

4.1. Introduction

Reinforcement learning (RL) [197] has proven to be a popular approach for control of complex processes. For large or continuous state and action spaces, function approximators are commonly used to learn representations of the policy. Deep neural networks (DNNs) [12] are a prevalent choice in this context; however, they often lack interpretability and are not conducive to safety verification, resulting in traditional DNN-based RL not yet widely being accepted in the control community. Alternatively, model predictive control (MPC) is an extremely successful control paradigm [28], involving solving a finite-horizon optimal control problem in a receding horizon fashion. Extensive results on the stability and performance of MPC exist [138]. However, MPC is entirely model-based, with its performance depending on an accurate system model.

The integration of MPC and RL is a promising direction for achieving safe and interpretable learning-based control [92, 169]. In particular, MPC has been proposed as a replacement for DNN function approximators in RL [75]. In this context, the optimal control action and cost of the MPC optimisation problem represent the policy and value function respectively. An MPC-based policy facilitates the use of the rich theory underlying MPC for obtaining insights into the policy, and allows to deliver certificates on the resulting behaviour. The state of the art [4, 75, 76], however, relies on a centralised approach with a single learning agent. This is in general prohibitive for multi-agent systems, where centralisation requires either a specific topology, with all agents connected to the central agent, or multi-hop communication across intermediate agents, where the number of hops grows with the network size. Additionally, centralised computation can become too complex, and requires the sharing of sensitive information, such as objective functions, with the central agent.

Addressing these challenges, distributed control of multi-agent systems offers computational scalability and privacy, with only neighbour-to-neighbour communication. Many existing works have adapted the MPC methodology to the distributed setting with distributed MPC [130] and, likewise, RL to the multi-agent setting RL (MARL) setting, in which multiple learning agents act in a common environment. A central challenge in MARL is that the interaction of simultaneously learning agents renders the learning target of each agent nonstationary, degrading the performance and convergence properties of single-agent algorithms [36]. Several works have tried to circumvent this problem using a centralised training and decentralised execution paradigm [9, 126]. However, centralised training is often either unrealistic or unavailable. Some approaches address the nonstationarity issue through communication across the network of agents during learning [196, 213, 235]. These works provide theoretical convergence guarantees, but focus on linear function approximation. MARL has also been addressed with DNN function approximators [64, 81]; however, these approaches do not emphasise information exchange between agents, and suffer from the same drawbacks as DNN-based single-agent RL. Addressing the nonstationarity of learning targets in MARL remains an open challenge.

This chapter proposes the following contributions. The use of MPC as a function approximator in RL is extended to the multi-agent setting. To this end we propose a structured convex distributed MPC scheme as an approximator for the policy and value functions, introducing a novel, model-based MARL approach for linear systems, free

from nonstationary. The method is distributed in training *and* deployment, with data sharing only between neighbours, irrespective of the network size and topology, thus avoiding centralised computation and multi-hop data communication. Furthermore, privacy of sensitive information in local parameters and functions is preserved, with only state trajectories being shared, in contrast to a centralised approach where local functions are shared with the central agent. Thanks to the MPC-based approximation, insights into the policy can be gained from the learnt components, e.g., the prediction model and constraints. Additionally, in contrast to DNN-based approaches, it is possible to inject *a priori* information, e.g., model approximations. Furthermore, we prove a result for consensus optimisation; relating the dual variables recovered distributively through the alternating direction method of multipliers (ADMM) to the optimal dual variables of the original problem, that enables the distributed learning. Nevertheless, it is noteworthy to remark that, unlike for the centralised case in [75], the proposed MPC approximation scheme lacks the theoretical guarantee of capturing the optimal multi-agent solution. This is because our function approximation, regardless of its expressiveness, is not explicitly equipped to capture the additional inter-agent couplings that may arise in the optimal solution to the MARL problem beyond those induced by the system topology.

The chapter is structured as follows. Section 4.2 provides the problem description and background theory. In Section 4.3 we present a result on the dual variables in ADMM, which will be used later on in the chapter. In Section 4.4 we introduce the structured distributed MPC function approximator. In Section 4.5 we propose Q-learning as the learning algorithm, and show how the parameter updates can be performed distributively. Section 4.6 gives an illustrative example.

Notation: define the index sets $\mathscr{M} = \{1, ..., M\}$ and $\mathscr{K} = \{0, ..., N-1\}$. The symbols t, k, and τ represent time steps in an RL context, an MPC context, and iterations of an algorithm, respectively. The symbols s and a refer to states and actions in an RL context, while x and u are used for MPC. We use bold variables to gather variables over a prediction window, e.g., $\boldsymbol{U} = \begin{bmatrix} u^\top(0) & \dots & u^\top(N-1) \end{bmatrix}^\top$ and $\boldsymbol{X} = \begin{bmatrix} x^\top(0) & \dots & x^\top(N) \end{bmatrix}^\top$. A vector stacking the vectors x_i, $i \in \mathscr{M}$, in one column is denoted $\mathrm{col}_{i\in\mathscr{M}}(x_i)$. The term *local* describes components known only by the corresponding agent. For simplicity we write $(x^\tau)^\top$ as $x^{\tau,\top}$. Inequalities on vectors are applied element-wise. Operation $\|\boldsymbol{y}\|_{\boldsymbol{A}}$ indicates $\sqrt{\boldsymbol{y}^\top \boldsymbol{A} \boldsymbol{y}}$.

4.2. Preliminaries and background

4.2.1. Problem description

We define a multi-agent Markov decision process (MDP) for M agents as the tuple $\left(\mathscr{S}, \{\mathscr{A}_i\}_{i\in\mathscr{M}}, P, \{L_i\}_{i\in\mathscr{M}}, \mathscr{G}\right)$. The set $\mathscr{S}$ is the global state set, composed of the local state sets for each agent $\mathscr{S} = \mathscr{S}_1 \times \ldots \times \mathscr{S}_M$. Moreover, $\mathscr{A}_i$ is the local action set and L_i is the local cost function for agent i, while P describes the state transition dynamics for the whole system. The graph $\mathscr{G} = (\mathscr{M}, \mathscr{E})$ defines a coupling topology between agents in the network, where edges $\mathscr{E}$ are ordered pairs (i, j) indicating that agent i may affect the cost and state transition of agent j. Define the neighbourhood of agent i as $\mathscr{N}_i = \{j \in M \mid (j, i) \in \mathscr{E}, i \neq j\}$. Note that an agent is not in its own neighbourhood,

i.e., $(i,i) \notin \mathscr{E}$. We assume the graph $\mathscr{G}$ is connected. Additionally, agents i and j can communicate if $i \in \mathscr{N}_j$ or $j \in \mathscr{N}_i$.

We consider agents to be linear dynamical systems with state $s_i \in \mathscr{S}_i \subseteq \mathbb{R}^n$ and control input $a_i \in \mathscr{A}_i \subseteq \mathbb{R}^m$. The true dynamics P of the network are assumed to be unknown, and we introduce an approximation of the dynamics for agent i, parametrised by a local parameter θ_i, as

$$s_i(t+1) = f_{\theta_i}\left(s_i(t), a_i(t), \{s_j(t)\}_{j\in\mathscr{N}_i}\right) = A_{i,\theta_i} s_i(t) + B_{i,\theta_i} a_i(t) + \sum_{j\in\mathscr{N}_i} A_{ij,\theta_i} s_j(t) + b_{\theta_i}, \tag{4.1}$$

with $b_{\theta_i} \in \mathbb{R}^n$ a constant offset allowing (1) to capture affine relationships.

We consider a co-operative RL setting. At time step t, agent i observes its own state $s_{i,t} \in \mathscr{S}_i$ and takes an action with a local policy parametrised by the local parameter θ_i; $\pi_{\theta_i}(s_{i,t}) = a_{i,t}$, observing the incurred local cost $L_{i,t}$ and next state $s_{i,t+1}$. The cooperative goal is to minimise, by modifying the parameters θ_i, the discounted cost over an infinite horizon

$$J\left(\{\pi_{\theta_i}\}_{i\in\mathscr{M}}\right) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t L_t\right], \tag{4.2}$$

with $L_t = \frac{1}{M}\sum_{i\in\mathscr{M}} L_{i,t}$ the average of the agents' local costs, and $\gamma \in (0,1]$. We define the global parametrisation as $\theta = \begin{bmatrix}\theta_1^\top & \dots & \theta_M^\top\end{bmatrix}^\top$. The joint policy, parametrised by θ, is then $\pi_\theta(s) = \{\pi_{\theta_i}(s_i)\}_{i\in\mathscr{M}}$. The joint action $a = \{a_i\}_{i\in\mathscr{M}}$ is generated by the joint policy; $\pi_\theta(s) = a$, where s is the joint state $s = \{s_i\}_{i\in\mathscr{M}}$. Note that the local policy π_{θ_i} is parametrised with the same parameter θ_i as the dynamics f_{θ_i} because the policy generates an action via an optimisation problem in which these dynamics form equality constraints (see Section 4.4).

4.2.2. Consensus optimisation

Section 4.4 shows that evaluation of the proposed MPC-based policy and value functions can be posed as a consensus optimisation problem and solved using the ADMM and global average consensus (GAC) algorithms. This section provides the relevant background.

Alternating direction method of multipliers

ADMM solves problems of the form

$$\min_{x\in\mathscr{X}, z\in\mathscr{Z}} \quad \{f_{\text{ADMM}}(x) + g_{\text{ADMM}}(z) \,:\, Ax + Bz = c\} \tag{4.3}$$

by alternating between minimisation of the augmented Lagrangian, split over x and z, and maximisation of the result with respect to the multipliers y as

$$\begin{aligned} x^{\tau+1} &= \arg\min_{x\in\mathscr{X}} \quad \mathscr{L}(x, z^\tau, y^\tau), \\ z^{\tau+1} &= \arg\min_{z\in\mathscr{Z}} \quad \mathscr{L}\left(x^{\tau+1}, z, y^\tau\right), \\ y^{\tau+1} &= y^\tau + \rho\left(Ax^{\tau+1} + Bz^{\tau+1} - c\right), \end{aligned} \tag{4.4}$$

with y the Lagrange multipliers, $\rho > 0$, and $\mathscr{L}(x,z,y) = f_{\text{ADMM}}(x) + g_{\text{ADMM}}(z) + y^\top (Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$ the augmented Lagrangian. We have the following convergence result:

Proposition 4.1 ([148])**.** *Assume the optimal solution set of* (4.3) *is nonempty and has optimal objective $P^\star$, the functions f_{ADMM} and g_{ADMM} are convex, $\mathscr{X}$ and $\mathscr{Z}$ are convex polytopic sets, and A and B are full column rank. Then, $f_{ADMM}(x^\tau) + g_{ADMM}(z^\tau) \to P^\star$ as $\tau \to \infty$. Additionally, $\{(x^\tau, z^\tau)\}_{\tau=1}^\infty$ has a single limit point $(x^\star, z^\star)$, which solves* (4.3).

Consider the following optimisation problem defined over the graph $\mathscr{G}$:

$$\begin{aligned} V_\theta(s) = \min_{x_1,\dots,x_M} \quad & \sum_{i\in\mathscr{M}} F_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right) \\ \text{s.t.} \quad & h_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right) \le 0, \\ & g_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right) = 0, \end{aligned} \tag{4.5}$$

where F_i, h_i, and g_i are convex and local to agent i, along with its state $x_i \in \mathbb{R}^n$. To solve this problem distributively we introduce the augmented state $\tilde{x}_i = \begin{bmatrix} x_i^\top & \text{col}_{j\in\mathscr{N}_i}^\top(x_j^{(i)}) \end{bmatrix}^\top$ for agent i, where $x_j^{(i)}$ is a local copy of agent j's state. We then introduce a *global* copy of each state $z = \begin{bmatrix} z_1^\top & z_2^\top & \dots & z_M^\top \end{bmatrix}^\top$, with z_i corresponding to x_i. The relevant portion of the global copies for agent i is $\tilde{z}_i = \begin{bmatrix} z_i^\top & \text{col}_{j\in\mathscr{N}_i}^\top(z_j) \end{bmatrix}^\top$. Define the local feasible sets for the augmented states as

$$\widetilde{\mathscr{X}}_i = \left\{ \tilde{x}_i \;\middle|\; h_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathscr{N}_i}\right) \le 0,\ g_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathscr{N}_i}\right) = 0 \right\}. \tag{4.6}$$

Problem (4.5) can then be reformulated with the addition of a redundant constraint

$$\begin{aligned} \min_{\{\tilde{x}_i\in\widetilde{\mathscr{X}}_i\}_{i\in\mathscr{M}}, z} \quad & \sum_{i\in\mathscr{M}} F_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right) \\ \text{s.t.} \quad & \tilde{x}_i - \tilde{z}_i = 0, \quad i \in \mathscr{M}, \end{aligned} \tag{4.7}$$

which is a particular instance of (4.3) satisfying the assumptions in Proposition 4.1 (see Section 4.A.1). The steps in (4.4), when applied to (4.7), reduce to [30, Section 7.2]:

$$\tilde{x}_i^{\tau+1} = \underset{\tilde{x}_i\in\widetilde{\mathscr{X}}_i}{\arg\min} \quad F_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathscr{N}_i}\right) + y_i^{\tau,\top}\tilde{x}_i + \frac{\rho}{2}\left\|\tilde{x}_i - \tilde{z}_i^\tau\right\|_2^2, \qquad i \in \mathscr{M}, \tag{4.8a}$$

$$z_i^{\tau+1} = \frac{1}{|\mathscr{N}_i|+1}\left(x_i^{\tau+1} + \sum_{j\in\mathscr{N}_i} x_i^{(j),\tau+1}\right), \qquad i \in \mathscr{M}, \tag{4.8b}$$

$$y_i^{\tau+1} = y_i^\tau + \rho\left(\tilde{x}_i^{\tau+1} - \tilde{z}_i^{\tau+1}\right), \qquad i \in \mathscr{M}. \tag{4.8c}$$

This is a distributed procedure as each step decouples over the agents, and uses only local and neighbouring information. By Proposition 4.1, at convergence of ADMM the local variable $\tilde{x}_i$ for each agent will contain the minimisers $x_i^\star$ of the original problem (4.5), and copies of the minimisers for neighbouring agents $\{x_j^{(i),\star}\}_{j\in\mathscr{N}_i}$.

Global average consensus

The GAC algorithm allows a network of agents to agree on the average value of local variables $v_{i,0} \in \mathbb{R}$, $i \in \mathscr{M}$, communicating over the graph $\mathscr{G}$. For each agent, the algorithm updates values as $v_i^{\tau+1} = \mathbf{P}(i,i)v_i^\tau + \sum_{j \mid (i,j)\in\mathscr{E}} \mathbf{P}(i,j)\ v_j^\tau$, where $\mathbf{P} \in \mathbb{R}_+^{M\times M}$ is a doubly stochastic matrix, i.e., entries in each row and column sum to 1. The iterates converge as $\lim_{\tau\to\infty} v_i^\tau = M^{-1}\sum_{i\in\mathscr{M}} v_{i,0}$, $i \in \mathscr{M}$, with M the number of agents [152].

4.3. Local recovery of optimal dual variables from ADMM

4

In this section we provide a result linking the dual variables from the local minimisation step (4.8a) to a subset of the dual variables in the original problem (4.5). This will be used later to construct the distributed learning update. The Lagrangian of (4.5) is

$$\mathscr{L}\left(\{x_i\}_{i\in\mathscr{M}}, \{\lambda_i\}_{i\in\mathscr{M}}, \{\mu_i\}_{i\in\mathscr{M}}\right) = \sum_{i\in\mathscr{M}}\Big(f_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right) + \lambda_i^\top h_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right) + \mu_i^\top g_i\left(x_i, \{x_j\}_{j\in\mathscr{N}_i}\right)\Big), \quad (4.9)$$

where λ_i and μ_i are the multipliers associated with the inequality and equality constraints, for each agent. We stress that these multipliers are related to the local constraints, and differ from the consensus multipliers $\{y_i\}_{i\in\mathscr{M}}$ used in the ADMM iterations. Denote the optimal multipliers as $\left(\{\lambda_i^\star\}_{i\in\mathscr{M}}, \{\mu_i^\star\}_{i\in\mathscr{M}}\right)$. At iteration τ of ADMM, the Lagrangian of the local minimisation step (4.8a) for agent i is

$$\mathscr{L}_i^\tau\left(x_i, \lambda_i^\tau, \mu_i^\tau\right) = f_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathscr{N}_i}\right) + \lambda_i^{\tau,\top} h_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathscr{N}_i}\right) + \mu_i^{\tau,\top} g_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathscr{N}_i}\right) + y_i^{\tau,\top}\tilde{x}_i + \frac{\rho}{2}\|\tilde{x}_i - \tilde{z}_i^\tau\|_2^2. \quad (4.10)$$

Denote the optimal multipliers for iteration τ as $(\lambda_i^{\star,\tau}, \mu_i^{\star,\tau})$.

Proposition 4.2. *Assume that the assumptions in Proposition 4.1 hold for problem* (4.5)*. Additionally assume that the functions F_i are strictly convex. Then, at convergence of ADMM, the optimal multipliers from the local minimisations* (4.8a) *converge to the corresponding subset of optimal multipliers for the original problem* (4.5)*, i.e.,*

$$\left(\lambda_i^{\star,\tau}, \mu_i^{\star,\tau}\right) \to \left(\lambda_i^\star, \mu_i^\star\right),\ i \in \mathscr{M}, \quad \text{as} \quad \tau \to \infty. \quad (4.11)$$

Proof. See Section 4.A.2. □

4.4. Distributed MPC as a function approximator

In [75] it was shown how an MPC scheme can capture the RL policy, state-value, and action-value functions. This section introduces a distributed counterpart, parametrised in $\theta = \begin{bmatrix}\theta_1^\top & \dots & \theta_M^\top\end{bmatrix}^\top$, approximating these quantities for a multi-agent system. However, unlike [75] and as already remarked in Section 4.1, guarantees of these approximations on capturing the multi-agent MDP solution (even with a rich enough parametrisation θ) are not provided.

4.4.1. Parametrised distributed MPC scheme

Consider the distributed MPC-based action-value function approximation

$$
\begin{aligned}
Q_\theta(s,a) = \min_{\substack{\{x_i\}_{i\in\mathcal{M}} \\ \{u_i\}_{i\in\mathcal{M}} \\ \{\Sigma_i\}_{i\in\mathcal{M}}}} \quad & \sum_{i\in\mathcal{M}} \Big(\beta_{\theta_i}(x_i(0)) + \sum_{k\in\mathcal{K}} \Big(\gamma^k l_{\theta_i}\left(x_i(k), u_i(k), \{x_j(k)\}_{j\in\mathcal{N}_i}\right) \\
& + \gamma^k w^\top \sigma_i(k) \Big) + \gamma^N V_{\mathrm{f},\theta_i}(x_i(N)) \Big) && \text{(4.12a)} \\
\text{s.t.} \quad & u_i(0) = a_i, && \forall i \in \mathcal{M}, \quad \text{(4.12b)} \\
& x_i(0) = s_i, && \forall i \in \mathcal{M}, \quad \text{(4.12c)} \\
& x_i(k+1) = f_{\theta_i}\left(x_i(k), u_i(k), \{x_j(k)\}_{j\in\mathcal{N}_i}\right), && \forall i \in \mathcal{M}, \forall k \in \mathcal{K}, \quad \text{(4.12d)} \\
& h_{\theta_i}\left(x_i(k), u_i(k), \{x_j(k)\}_{j\in\mathcal{N}_i}\right) \le \sigma_i(k), && \forall i \in \mathcal{M}, \forall k \in \mathcal{K}, \quad \text{(4.12e)}
\end{aligned}
$$

The functions β_{θ_i}, l_{θ_i}, and V_{f,θ_i} are the initial, stage, and terminal cost approximations respectively, f_{θ_i} is a model approximation, and h_{θ_i} is an inequality constraint function. Each is parametrised by a local parameter θ_i and *known only to the corresponding agent*, such that knowledge of the components of the MPC scheme is distributed across the agents. Parametrised terminal constraints could be included, but are omitted here for simplicity. For conciseness, we summarise (4.12) as

$$
\begin{aligned}
Q_\theta(s,a) = \min \quad & \sum_{i\in\mathcal{M}} F_{\theta_i}\left(\boldsymbol{X}_i, \{\boldsymbol{X}_j\}_{j\in\mathcal{N}_i}, \boldsymbol{U}_i, \boldsymbol{\Sigma}_i\right) && \text{(4.13a)} \\
\text{s.t.} \quad & G_{\theta_i}\left(s_i, a_i, \boldsymbol{X}_i, \{\boldsymbol{X}_j\}_{j\in\mathcal{N}_i}, \boldsymbol{u}_i\right) = 0, && \forall i \in \mathcal{M}, \quad \text{(4.13b)} \\
& H_{\theta_i}\left(\boldsymbol{X}_i, \{\boldsymbol{X}_j\}_{j\in\mathcal{N}_i}, \boldsymbol{u}_i, \boldsymbol{\Sigma}_i\right) \le 0, && \forall i \in \mathcal{M}. \quad \text{(4.13c)}
\end{aligned}
$$

The joint policy is then computed as

$$
\begin{aligned}
\pi_\theta(s) = \ & \arg\min_{\{u(0)_i\}_{i\in\mathcal{M}}} \quad \text{(4.13a)} \\
& \text{s.t.} \quad \text{(4.12c)} - \text{(4.12e)},
\end{aligned}
\tag{4.14}
$$

and the global value function $V_\theta(s)$ can be obtained as the optimal value of (4.14), satisfying the fundamental Bellman equations [75].

Intuitively, this structured MPC scheme approximates the local cost of each agent through the local cost functions. Interactions between coupled agents enter the scheme via the parametrised stage costs F_{θ_i}, the dynamics in G_{θ_i}, and the inequality constraints in H_{θ_i}. Inexact knowledge of agents' dynamics, constraints, inter-agent interactions, and local costs can be encoded in an initial guess of the local parameters θ_i. Through learning, the cost and model representations will be modified to improve the global performance. The constraints in (4.13) are chosen to be affine and the functions F_{θ_i} to

be strictly convex in their arguments $\boldsymbol{X}_i$, $\{\boldsymbol{X}_j\}_{j\in\mathcal{N}_i}$, $\boldsymbol{U}_i$, and $\boldsymbol{\Sigma}_i$, such that (4.13) satisfies the assumptions for Propositions 4.1 and 4.2. This is not a strong assumption for linear systems, as the true optimal value function is convex when the stage costs are convex, as is common in the literature [28]. Indeed, for quadratic stage costs, with a sufficiently long prediction horizon, a convex optimisation problem can capture the true infinite horizon cost [47].

4

4.4.2. Distributed evaluation

The ADMM and GAC algorithms can be used to solve (4.12) distributively. Each agent stores a local copy of the predicted states of neighbouring agents over the prediction horizon, and constructs the augmented state $\widetilde{\boldsymbol{X}}_i = \begin{bmatrix}\boldsymbol{X}_i^\top & \mathrm{col}^\top_{j\in\mathcal{N}_i}(\boldsymbol{X}_j^{(i)})\end{bmatrix}^\top$, where $\boldsymbol{X}_j^{(i)}$ is agent i's local copy of agent j's predicted state over the prediction horizon. Global copies are introduced of each state prediction $\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{Z}_1^\top & \dots & \boldsymbol{Z}_M^\top\end{bmatrix}^\top$, with the relevant components of $\boldsymbol{Z}$ for agent i denoted $\widetilde{\boldsymbol{Z}}_i = \begin{bmatrix}\boldsymbol{Z}_i^\top & \mathrm{col}^\top_{j\in\mathcal{N}_i}(\boldsymbol{Z}_j)\end{bmatrix}^\top$. As in Section 4.2.2, ADMM solves (4.13) by the following iterations:

$$\begin{aligned}\widetilde{\boldsymbol{X}}_i^{\tau+1}, \boldsymbol{U}_i^{\tau+1}, \boldsymbol{\Sigma}_i^{\tau+1} = \quad & \arg\min \quad F_{\theta_i}\left(\boldsymbol{X}_i, \{\boldsymbol{X}_j^{(i)}\}_{j\in\mathcal{N}_i}, \boldsymbol{U}_i, \boldsymbol{\Sigma}_i\right) \\ & \qquad\quad + \textstyle\sum_{k\in\mathcal{K}} y_i^{\tau,\top}(k)\tilde{x}_i(k) + \frac{\rho}{2}\|\tilde{x}_i(k) - \tilde{z}_i(k)^\tau\|_2^2 \\ & \text{s.t.} \qquad G_{\theta_i}\left(s_i, a_i, \boldsymbol{X}_i, \{\boldsymbol{X}_j^{(i)}\}_{j\in\mathcal{N}_i}, \boldsymbol{u}_i\right) = 0, \\ & \qquad\quad H_{\theta_i}\left(\boldsymbol{X}_i, \{\boldsymbol{X}_j^{(i)}\}_{j\in\mathcal{N}_i}, \boldsymbol{u}_i, \boldsymbol{\Sigma}_i\right) \le 0,\end{aligned} \tag{4.15a}$$

$$\boldsymbol{z}_i^{\tau+1} = \frac{1}{|\mathcal{N}_i|+1}\left(\boldsymbol{X}_i^{\tau+1} + \sum_{j\in\mathcal{N}_i} \boldsymbol{X}_i^{(j),\tau+1}\right), \tag{4.15b}$$

$$\boldsymbol{Y}_i^{\tau+1} = \boldsymbol{Y}_i^{\tau} + \rho\left(\widetilde{\boldsymbol{X}}_i^{\tau+1} - \widetilde{\boldsymbol{Z}}_i^{\tau+1}\right). \tag{4.15c}$$

By Proposition 4.1, as $\tau\to\infty$, the outputs of the local minimisation (4.15a) converge to the minimisers of the original problem (4.13), i.e., $\widetilde{\boldsymbol{X}}_i^{\tau+1}, \boldsymbol{U}_i^{\tau+1}, \boldsymbol{\Sigma}_i^{\tau+1} \to \widetilde{\boldsymbol{X}}_i^\star, \boldsymbol{U}_i^\star, \boldsymbol{\Sigma}_i^\star$. Agents then evaluate their local objective component $F_{\theta_i}^\star = F_{\theta_i}\left(\boldsymbol{X}_i^\star, \{\boldsymbol{X}_j^{(i),\star}\}_{j\in\mathcal{N}_i}, \boldsymbol{U}_i^\star, \boldsymbol{\Sigma}_i^\star\right)$. The global action-value $Q_\theta(s,a) = \sum_{i\in\mathcal{M}} F_{\theta_i}^\star$ can then be agreed upon across the network via the GAC algorithm. Each agent makes the naive initial guess $Q_\theta(s,a) = MF_{\theta_i}^\star$, i.e., each agent assumes that all other agents have the same local cost. The GAC algorithm gives convergence of these values, as per Section 4.2.2, to the average $\sum_{i\in\mathcal{M}} F_{\theta_i}^\star$, which is the global action-value.

Evaluation of the global value $V_\theta(s)$ and the joint policy follows from applying the same steps to (4.14). From the optimal control sequence $\boldsymbol{U}_i^\star$ agents evaluate their local component $a_i = \pi_{\theta_i}(s_i) = u_i^\star(0)$ of the joint policy $\pi_\theta(s)$. We highlight that, while the joint policy is a function of the global state s, the local policies are functions only of the local state s_i, i.e., $\pi_{\theta_i}(s_i)$, as (4.15a) requires knowledge only of s_i.

4.4.3. Summary of distributed MPC as function approximator

We briefly summarise the key points of the distributed-MPC-based approximator. Each agent stores local knowledge of its learnable parameter θ_i, and its local parametrised functions β_{θ_i}, l_{θ_i}, V_{f,θ_i}, f_{θ_i}, and h_{θ_i}. In addition, it maintains its own state prediction $\boldsymbol{X}_i$, copies of predictions for its neighbours' states $\{\boldsymbol{X}_j^{(i)}\}_{j\in\mathcal{N}_i}$, and an additional set of copies required for agreement. Finally, each agent stores the local Lagrange multipliers $\boldsymbol{Y}_i$ used in ADMM. Evaluating Q_θ (or V_θ), from the perspective of agent i, is summarised in Algorithm 3. For simplicity, both ADMM and GAC use a fixed number of iterations in the algorithm, T_A and T_C respectively, with the consequences discussed in Section 4.5.2.

4

Algorithm 3: Evaluation of $Q_\theta(s,a)$ for agent i

Input: s_i and a_i (not a_i if evaluating (4.14))
Output: Global state-value $Q_\theta(s,a)$ (local action $a_i \leftarrow u_i^{T_\mathrm{A}}(0)$ if evaluating (4.14))

for $\tau = 0,\dots,T_A$ **do**
  Get $\tilde{\boldsymbol{x}}_i^{\tau+1}, \boldsymbol{U}_i^{\tau+1}, \boldsymbol{\Sigma}_i^{\tau+1}$ via (4.15a)
  Send $\boldsymbol{X}_j^{(i),\tau+1}$ and receive $\boldsymbol{X}_i^{(j),\tau+1}$, $\forall j \in \mathcal{N}_i$
  Perform averaging step (4.15b)
  Send $\boldsymbol{Z}^{\tau+1}$ and receive $\boldsymbol{Z}_j^{\tau+1}$, $\forall j \in \mathcal{N}_i$
  Perform local multipliers update (4.15c)
end
$\bar{Q}_\theta(s,a) \leftarrow MF_{\theta_i}\left(\boldsymbol{X}_i^{T_\mathrm{A}}, \{\boldsymbol{X}_j^{(i),T_\mathrm{A}}\}_{j\in\mathcal{N}_i}, \boldsymbol{U}_i^{T_\mathrm{A}}, \boldsymbol{\Sigma}_i^{T_\mathrm{A}}\right)$
Perform T_C iterations GAC, with initial guess $\bar{Q}_\theta(s,a)$, to agree on $Q_\theta(s,a)$

4.5. Distributed Q-learning

In this section we show that using Q-learning as the RL algorithm to learn the local parameters θ_i enables a distributed learning update that avoids nonstationarity. Q-learning [197] adjusts, at each time step t, the global parameters $\theta = \begin{bmatrix}\theta_1^\top & \dots & \theta_M^\top\end{bmatrix}^\top$ as

$$\begin{aligned}\delta_t &= L_t + \gamma V_\theta(s_{t+1}) - Q_\theta(s_t,a_t),\\ \theta &\leftarrow \theta + \alpha\delta_t \nabla_\theta Q_\theta(s_t,a_t),\end{aligned} \tag{4.16}$$

where $\alpha \in \mathbb{R}$ is a learning rate and $\delta_t \in \mathbb{R}$ is the temporal-difference (TD) error at time step t.

4.5.1. Separable Q-learning updates

The update (4.16) appears to be centralised due to the global components a_t, L_t, s_t, s_{t+1}, and θ. However, the structure of (4.12) allows decomposition into local updates of θ_i for each agent. Addressing first the TD error δ_t, we have shown in Section 4.4 that $V_\theta(s)$ and $Q_\theta(s,a)$ can be evaluated distributively. The globally averaged cost L_t is the average

of all agents' locally incurred costs $L_{i,t}$, and can be shared across the network using the GAC algorithm. This can be incorporated into line 8 of Algorithm 3, with no additional communication overhead.

We now address the gradient term $\nabla_\theta Q_\theta(s_t, a_t)$. The Lagrangian of (4.12) is

$$\mathscr{L}_\theta(s,a,p) = \sum_{i\in\mathscr{M}} \Big(F_{\theta_i}\left(\boldsymbol{X}_i, \{\boldsymbol{X}_j\}_{j\in\mathscr{N}_i}, \boldsymbol{U}_i, \boldsymbol{\Sigma}_i\right) + \boldsymbol{\lambda}_i^\top G_{\theta_i}\left(s_i, a_i, \boldsymbol{X}_i, \{\boldsymbol{X}_j\}_{j\in\mathscr{N}_i}, \boldsymbol{U}_i\right) + \boldsymbol{\mu}_i^\top H_{\theta_i}\left(\boldsymbol{X}_i, \{\boldsymbol{X}_j\}_{j\in\mathscr{N}_i}, \boldsymbol{U}_i, \boldsymbol{\Sigma}_i\right)\Big), \quad (4.17)$$

where $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ are the multiplier vectors associated with the equality and inequality constraints, respectively, and the primal and dual variables are grouped as

$$p = \left(\{\boldsymbol{X}_i\}, \{\boldsymbol{U}_i\}, \{\boldsymbol{\Sigma}_i\}, \{\boldsymbol{\lambda}_i\}, \{\boldsymbol{\mu}_i\}\right)_{i\in\mathscr{M}}. \quad (4.18)$$

We stress again that the multipliers in p are associated with the constraints of (4.12), and are unrelated to the multipliers $\mathbf{y}_i$ used in the ADMM procedure. Via sensitivity analysis [35], the gradient $\nabla_\theta Q_\theta(s,a)$ coincides with the gradient of the Lagrangian at the optimal primal and dual variables $p^\star$:

$$\nabla_\theta Q_\theta(s,a) = \frac{\partial \mathscr{L}_\theta\left(s,a,p^\star\right)}{\partial\theta}. \quad (4.19)$$

Observe that (4.17) is separable over parameters θ_i, states s_i, actions a_i, and subsets of the primal and dual variables p_i, i.e., $\mathscr{L}_\theta(s,a,p) = \sum_{i\in\mathscr{M}} \mathscr{L}_{\theta_i}\left(s,a,p^\star\right)$, where $\mathscr{L}_{\theta_i}$ is the i-th term within the sum (4.17), and $p_i = \left(\boldsymbol{X}_i, \boldsymbol{U}_i, \boldsymbol{\Sigma}_i, \{\boldsymbol{X}_j\}_{j\in\mathscr{N}_i}, \boldsymbol{\lambda}_i, \boldsymbol{\mu}_i\right)$. We then express (4.19) as

$$\frac{\partial \sum_{i\in\mathscr{M}} \mathscr{L}_{\theta_i}(s_i,a_i,p_i^\star)}{\partial\theta} = \begin{bmatrix} \frac{\partial \mathscr{L}_{\theta_1}(s_1,a_1,p_1^\star)}{\partial\theta_1} \\ \vdots \\ \frac{\partial \mathscr{L}_{\theta_M}(s_M,a_M,p_M^\star)}{\partial\theta_M} \end{bmatrix}, \quad (4.20)$$

and the centralised parameter update (4.16) can be hence unpacked into M local updates:

$$\theta_i \leftarrow \theta_i + \alpha\delta_t \frac{\partial \mathscr{L}_{\theta_i}\left(s_{i,t}, a_{i,t}, p_i^\star\right)}{\partial\theta_i}, \quad i\in\mathscr{M}. \quad (4.21)$$

What then remains to be shown is that the subset of optimal primal and dual variables $p_i^\star$ for (4.12) is available locally to agent i. By Proposition 4.1, at convergence of ADMM, the local minimisation (4.15a) returns the minimisers $\left(\boldsymbol{X}_i^\star, \boldsymbol{U}_i^\star, \boldsymbol{\Sigma}_i^\star\right)$ and local copies of the minimisers of neighbouring agents $\{\boldsymbol{X}_j^{(i),\star}\}_{j\in\mathscr{N}_i} = \{\boldsymbol{X}_j^\star\}_{j\in\mathscr{N}_i}$. Each agent hence has local knowledge of the optimal *primal* values in $p_i^\star$. For the optimal *dual* values, by Proposition 4.2, at convergence of ADMM, the optimal dual variables of the local minimisation (4.15a) for agent i are equal to the relevant subset of optimal dual variables for the original problem, i.e., $\left(\boldsymbol{\lambda}_i^\star, \boldsymbol{\mu}_i^\star\right)$. Each agent hence has local knowledge of the optimal *dual* values in $p_i^\star$ as well. Agents can therefore perform the local update (4.21). Nonstationarity is avoided as the local updates reconstruct the centralised update of the whole network (4.16).

4.5.2. Implementation details

In this section we discuss some auxiliary details in the implementation of our proposed approach.

- Exploration can be added to the approach in, e.g., an epsilon-greedy fashion, in which case the agent adds a random perturbation to the objective (4.12a) with some probability ϵ_i, where ϵ_i decreases as training progresses [4, 233]. This causes a perturbation in the joint policy. It is assumed that the system, with exploration injected, is sufficiently persistently exciting.
- The finite termination of ADMM and the GAC algorithm introduces errors in the evaluation of V_θ and Q_θ, π_θ, and $p^\star$, and could lead to instability in the learning. To counter this, large numbers of ADMM and GAC iterations may be used in a simulated learning phase, when there are no constraints on computation time between actions. This is desirable as primal and dual variables with high precision are needed to reliably compute the sensitivity (4.19). On the contrary, at deployment, sensitivities are not required and only the policy must be evaluated. Thus, the iteration numbers can be reduced, as ADMM often converges to modest accuracy within few iterations [30]. Additionally, experience replay (ER) [4, 122] can improve learning stability by using an average of past observations when calculating the gradient and the TD error. In our method ER requires no extra mechanism as agents can maintain a local history of values for δ and $\nabla_{\theta_i}\mathscr{L}_{\theta_i}\left(s, p_i^\star\right)$. In our numerical experiments, we found the learning to succeed with ER and modest numbers of ADMM and GAC iterations.
- The variables $\alpha > 0$ and $\gamma \in (0, 1]$ are hyperparameters; as the distributed update fully reconstructs the centralised update, existing methods for selecting these parameters in centralised learning apply directly, e.g., see [197, Chapter 9.6].

4.6. Numerical examples

This section presents two numerical example. Source code and simulation results can be found at `https://github.com/SamuelMallick/dmpcrl-concept`.

4.6.1. Academic example

We modify the system from [75], forming a three-agent system with state coupling in a chain, i.e., $1 \leftrightarrow 2 \leftrightarrow 3$, with real (unknown) dynamics $s_i(t+1) = A_i s_i(t) + B_i a_i(t) + \sum_{j \in \mathcal{N}_i} A_{ij} s_j(t) + \begin{bmatrix} e_i(t) & 0 \end{bmatrix}^\top$ where

$$A_i = \begin{bmatrix} 0.9 & 0.35 \\ 0 & 1.1 \end{bmatrix}, \quad \{A_{ij}\}_{j \in \mathcal{N}_i} = \begin{bmatrix} 0 & 0 \\ 0 & -0.1 \end{bmatrix}, \tag{4.22}$$

with $B = \begin{bmatrix} 0.0813 & 0.2 \end{bmatrix}^\top$ and $e_i(t)$ uniformly distributed over the interval $[-0.1, 0]$. The RL task is to drive the states and control towards the origin while avoiding violations of the state constraints $\underline{s} \leq s_i \leq \overline{s}$, with local costs as

$$L_{i,t}(s_i, a_i) = \|s_i\|_2^2 + \frac{1}{2}\|u_i\|_2^2 + \max\left\{0, \omega^\top\left(\underline{s} - s_i\right)\right\} + \max\left\{0, \omega^\top\left(s_i - \overline{s}\right)\right\}, \tag{4.23}$$

with $\omega = \begin{bmatrix}10^2 & 10^2\end{bmatrix}^\top$, $\underline{s} = \begin{bmatrix}0 & -1\end{bmatrix}^\top$, and $\overline{s} = \begin{bmatrix}1 & 1\end{bmatrix}^\top$. Non-positive noise on the first state biases that term towards violating the lower bound of zero. This renders the task challenging, as the agents must regulate the state to zero, and yet driving the first dimension to zero will result in constraint violations due to the noise. The true model is unknown, with agents instead knowing a uniform distribution of models, containing the true model:

$$\widehat{A}_i = \begin{bmatrix}1+a_{1,i} & 0.25+a_{2,i}\\ 0 & 1+a_{3,i}\end{bmatrix}, \quad \{\widehat{A}_{ij}\}_{j\in\mathcal{N}_i} = \begin{bmatrix}0 & 0\\ 0 & c_i\end{bmatrix}, \tag{4.24}$$

with $\widehat{B}_i = \begin{bmatrix}0.0312+b_{1,i} & 0.25+b_{2,i}\end{bmatrix}^\top$ and, for all i, the random variables distributed uniformly as $a_{1,i}, a_{2,i}, a_{3,i} \in [-0.1, 0.1]$, $b_{1,i} \in [0, 0.075]$, $b_{2,i} \in [-0.075, 0]$ and $c_i \in [-0.1, 0]$. We implement the following distributed MPC scheme

$$\begin{aligned}
\min_{\{(\boldsymbol{X}_i,\boldsymbol{U}_i,\boldsymbol{\Sigma}_i)\}_{i\in\mathcal{M}}} \quad & \sum_{i\in\mathcal{M}}\left(V_{i,0} + \sum_{k\in\mathcal{K}} f_i^\top \begin{bmatrix}x_i(k)\\ u_i(k)\end{bmatrix} + \frac{1}{2}\gamma^k\left(\|x_i(k)\|_2^2 + \frac{1}{2}\|u_i(k)\|_2^2 + \omega^\top\sigma_i(k)\right)\right)\\
\text{s.t.} \quad & -1 \le u_i(k) \le 1, && \forall i\in\mathcal{M}, \forall k\in\mathcal{K},\\
& x_i(0) = s_i, && \forall i\in\mathcal{M},\\
& x_i(k+1) = A_i x_i(k) + B_i u_i(k) + \sum_{j\in\mathcal{N}_i} A_{ij}x_j(k) + b_i, && \forall i\in\mathcal{M}, \forall k\in\mathcal{K},\\
& \underline{s} + \underline{x}_i - \sigma_i(k) \le x_i(k) \le \overline{s} + \overline{x}_i + \sigma_i(k), && \forall i\in\mathcal{M}, \forall k\in\mathcal{K},
\end{aligned} \tag{4.25}$$

where $M = 3$, $\mathcal{K} = \{0, \dots, 9\}$, and $\gamma = 0.9$. The learnable parameters for each agents are then $\theta_i = \left(V_{i,0}, \underline{x}_i, \overline{x}_i, b_i, f_i, A_i, B_i, \{A_{ij}\}_{j\in\mathcal{N}_i}\right)$. The initial values for $A_i, B_i, \{A_{ij}\}_{j\in\mathcal{N}_i}$ are the inaccurate model (4.24) with the random variables set to zero. All other learnable parameters are initialised to zero.

To illustrate Proposition 4.2, for a given global state s, Figure 4.1 shows the error between the true optimal dual variables and those recovered from evaluating the MPC scheme with ADMM, as a function of the ADMM iteration index τ. The locally recovered dual variables are close to the true values, with the error initially decreasing with the iteration index.

We now compare the learning performance of our distributed approach with the centralised approach inspired from [75], using the same MPC scheme, and the same learning hyperparameters for both. We use an exploration probability of $\epsilon_i = 0.7$, exponentially decaying with rate 0.99, with local costs perturbed uniformly over the interval $[-1, 1]$. We use a learning rate of $\alpha = 6\cdot 10^{-5}$, exponentially decaying with rate 0.9996, and ER with an average over 15 past samples at an update rate of every 2 time steps. For the distributed approach we use 50 iterations in ADMM and 100 iterations in GAC. These values were tuned to keep the iterations low without introducing significant approximation errors. Figure 4.2 shows the state and input trajectories for the three agents during training. Figure 4.3 shows the evolution of the global TD error and the collective cost. Figure 4.4 shows the learnable parameters of the second agent during training (similar convergence profiles are observed for the other agents). It is seen that the behaviour of the distributed approach is similar to that of the centralised approach, reducing the TD error and costs incurred, with the centralised approach converging slightly faster.

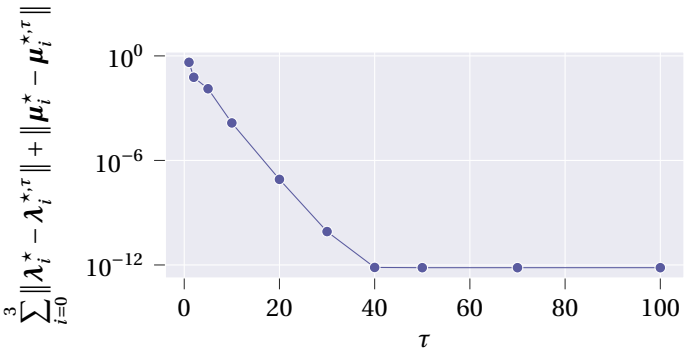

Figure 4.1.: Accuracy of the dual variables recovered by Proposition 4.2 as a function of the ADMM iteration index τ

The costs are reduced by maintaining the first state of each agent above zero to prevent expensive violations of the state constraint due to coupling and noise.

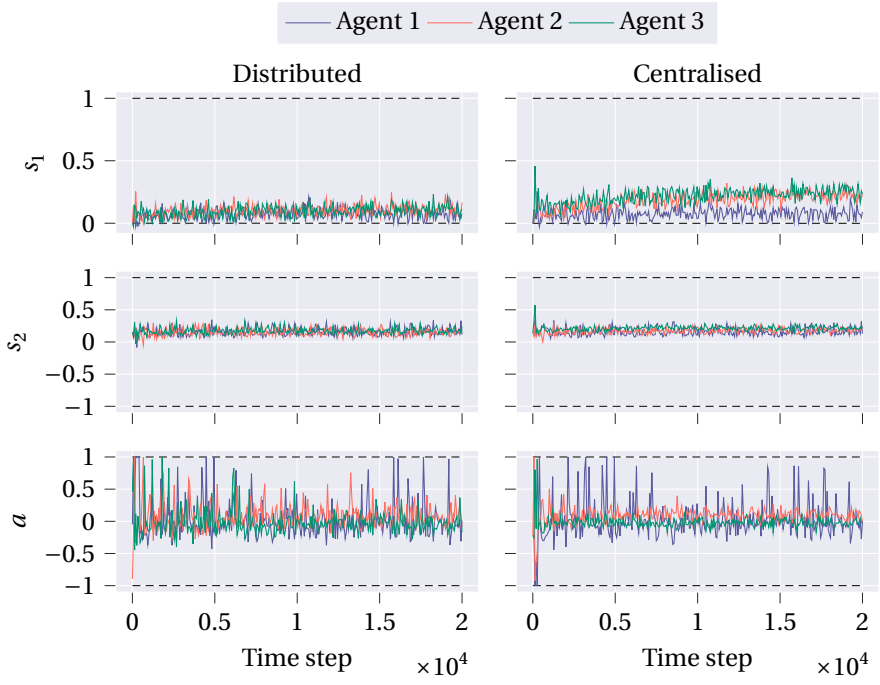

Figure 4.2.: Evolution of the states and inputs during training for the distributed and centralised approaches for different agents

We compare the performance of the distributed policy against both a distributed

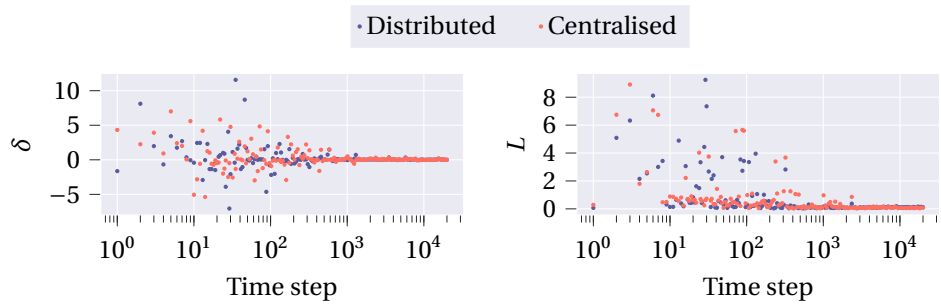

Figure 4.3.: Evolution of TD errors and stage costs during training

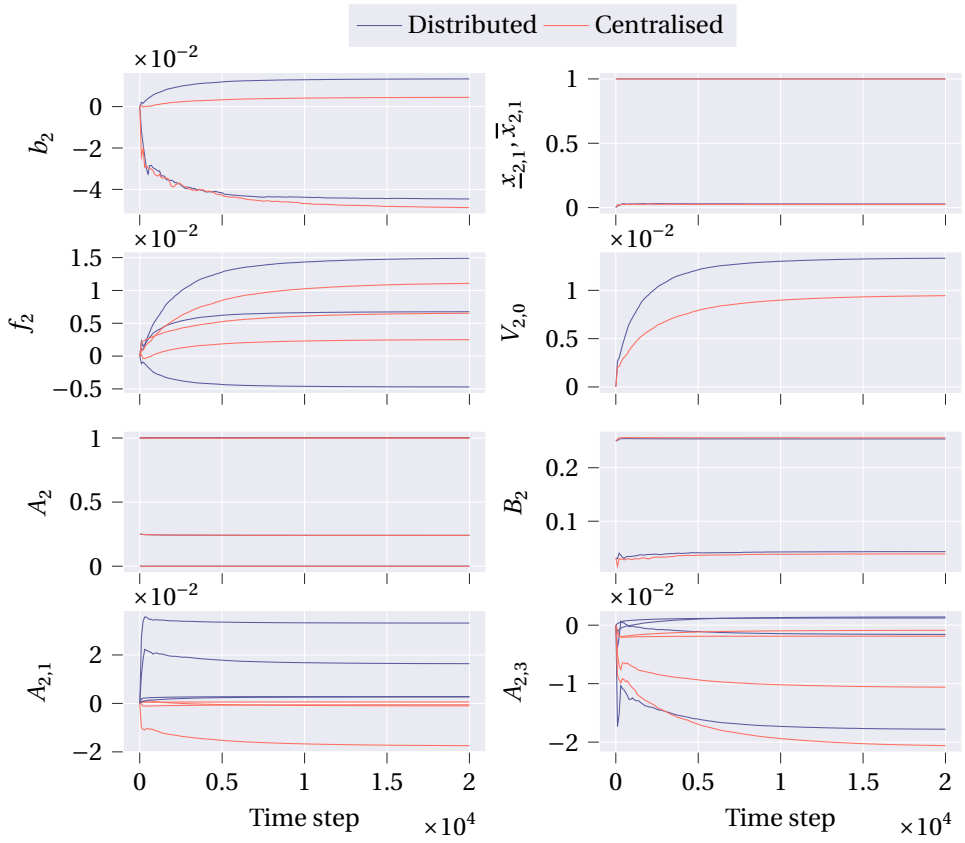

Figure 4.4.: Evolution of learnable parameters for agent 2 during training for the distributed and centralised approaches

nominal MPC controller (NMPC) and a distributed stochastic MPC controller (SMPC) based on the scenario approach [172]. The nominal MPC controller uses the inexact model (4.24), with all random variables set to 0. For the SMPC controller we consider the case where the true model is unknown and $N = 25$ samples of the model distribution (4.24) and the noise distribution are used to account for the uncertainty. The number of samples was manually tuned to balance conservativeness and robustness [172]. We also consider the case where the exact model (4.22) is known, and only the noise distribution is sampled. The controllers are compared in Figure 5, showing the closed-loop cost accumulated over 100 time steps. The learnt policy can be seen to improve from a performance initially comparable to NMPC, learning to outperform SMPC, and approaching the performance of SMPC with a perfect model. We highlight that our approach retains the computational complexity of NMPC, while SMPC is significantly more complex, optimising over N copies of the state trajectories.

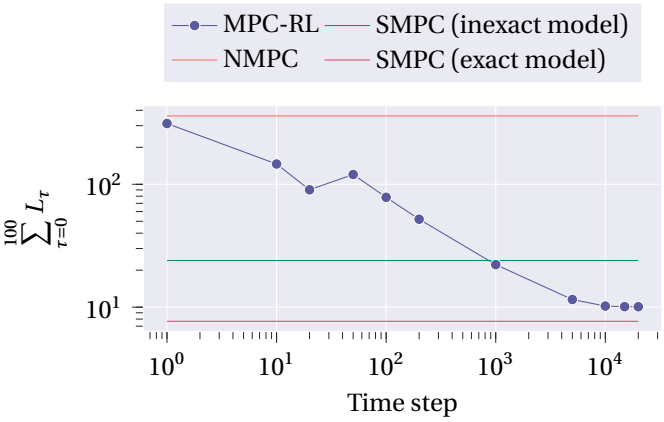

Figure 4.5.: NMPC and SMPC compared against the learning-based MPC-RL policy at different training time steps

4.6.2. Power systems example

Automatic generation control is a popular case study for distributed control [63]. We consider the benchmark system presented, along with test scenarios, in [167]. The network is a four-area power system whose continuous-time linearised dynamics for sub-system i are $\dot{x}_i = A_i\left(\{P_{ij}\}_{j\in\mathcal{N}_i}\right)x_i + B_i u_i + L_i \Delta P_{L_i} + \sum_{j\in\mathcal{N}_i} A_{ij}\left(\{P_{ij}\}_{j\in\mathcal{N}_i}\right)x_j$, where $x_i = \begin{bmatrix}\Delta\phi_i & \Delta\omega_i & \Delta P_{m_i} & \Delta P_{v_i}\end{bmatrix}^\top$ contains the rotor's angular displacement $\Delta\phi_i$, the rotating mass' speed $\Delta\omega_i$, the mechanical power ΔP_{m_i}, and the steam valve's position ΔP_{v_i}. Moreover, ΔP_{L_i} is the local power load, the control action u_i is local power generation, and $\mathcal{N}_i$ is the set of neighbouring sub-systems, i.e., those connected directly through tie-lines. The agents are again coupled as a chain. The P_{ij} term, which enters the matrices linearly, is a synchronising coefficient for the tie-line between sub-systems i and j. For the sake of space, we refer to [167] for the full definitions of the dynamics matrices. The

local continuous dynamics are discretised, treating the coupling and loads as exogenous inputs, with the discrete-time matrices denoted $\widehat{A}_i$, $\widehat{B}_i$, $\widehat{A}_{ij}$, and $\widehat{L}_i$. In our experiments we use the discrete-time model in the MPC controllers, and the continuous-time model for the underlying system.

We consider a scenario where the step changes $(+0.15, -0.15, +0.12, -0.24, 0.28)$ occur in the local loads $(\Delta P_{L_1}, \Delta P_{L_2}, \Delta P_{L_3}, \Delta P_{L_3}, \Delta P_{L_4})$ at times $(5, 15, 20, 40, 40)$, with the initial load values all zero. As in [63], we take the case where local power production is insufficient to balance the local loads, requiring power transfer between areas. The control constraints are then $(\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4) = (0.2, 0.1, 0.3, 0.1)$, where $|u_i| \leq \bar{u}_i$. Additionally the angular displacements are constrained as $|\Delta\phi_i| \leq 0.1$. Each sub-system has a learning agent. Agents know the nominal value of ΔP_{L_i}; however, the real loads are subject to uniform additive noise in the range $[-0.1, 0.1]$. Exact knowledge of P_{ij} is unknown, with agents having initial approximations of the true values $\widehat{P}_{ij} = P_{ij} + d_{ij}$, where $d_{ij} \sim \mathcal{U}(-2, 2)$. For the RL task, we take the performance criteria from [167], regulating the states and control input to the time-varying equilibria $\bar{x}_i = \begin{bmatrix} 0 & 0 & \Delta P_{L_i} & \Delta P_{L_i} \end{bmatrix}^\top$ and $\bar{u}_i = \Delta P_{L_i}$, while minimising power transfer between sub-systems and avoiding expensive constraint violations. We implement the following distributed MPC scheme

$$
\begin{aligned}
\min_{\{(\boldsymbol{X}_i, \boldsymbol{U}_i, \boldsymbol{\Sigma}_i)\}_{i\in\mathcal{M}}} \quad & \sum_{i\in\mathcal{M}} \Bigg(V_{i,0} + \sum_{k\in\mathcal{K}} f_i^\top \begin{bmatrix} x_i(k) \\ u_i(k) \end{bmatrix} \\
& + \gamma^k \left(\left\| x_i(k) - \overline{x}_i \right\|_{Q_{x_i}}^2 + \left\| u_i(k) - \overline{u}_i \right\|_{Q_{u_i}}^2 + \omega^\top \sigma_i(k) \right) \Bigg) \\
\text{s.t.} \quad & |u_i(k)| \leq \overline{u}_i, && \forall i \in \mathcal{M}, \forall k \in \mathcal{K}, \\
& x_i(0) = s_i, && \forall i \in \mathcal{M}, \\
& x_i(k+1) = \widehat{A}_i \left(\left\{ \widehat{P}_{ij} \right\}_{j\in\mathcal{N}_i} \right) x_i(k) + \widehat{B}_i u_i(k) + \widehat{L}_i \Delta P_{L_i} \\
& \qquad + \sum_{j\in\mathcal{N}_i} \widehat{A}_{ij} \left(\left\{ \widehat{P}_{ij} \right\}_{j\in\mathcal{N}_i} \right) x_j(k) + 0.1 b_i, && \forall i \in \mathcal{M}, \forall k \in \mathcal{K}, \\
& -0.1 + \underline{\Delta\phi}_i - \sigma_i(k) \leq x_i(k) \leq 0.1 + \overline{\Delta\phi}_i + \sigma_i(k), && \forall i \in \mathcal{M}, \forall k \in \mathcal{K},
\end{aligned}
$$

where $M = 4$, $\mathcal{K} = \{0, \ldots, 4\}$, $\omega = \begin{bmatrix} 50^2 & 50^2 \end{bmatrix}^\top$, and $\gamma = 0.9$. The learnable parameters for each agents are $\theta_i = \left(V_{i,0}, \underline{\Delta\phi}_i, \overline{\Delta\phi}_i, b_i, f_i, Q_{x_i}, Q_{u_i}, \left\{ \widehat{P}_{ij} \right\}_{j\in\mathcal{N}_i} \right)$. The $\widehat{P}_{ij}$ values are initialised with the initial approximated values known by the agents, while Q_{x_i} and Q_{u_i} are initialised with the values given in the MPC controller in [167]. All other parameters are initialised to zero. The agents are trained episodically. We use an exploration probability of $\epsilon = 0.5$, exponentially decaying with rate 0.8, with local costs perturbed uniformly over the interval $[-0.04, 0.04]$. We use a learning rate of $\alpha = 7 \cdot 10^{-7}$, exponentially decaying with rate 0.98, and experience replay over 3 episodes at an update rate of once per episode. Again, we use the tuned number of iterations: 50 iterations in ADMM and 100 iterations in GAC.

Figure 4.6 shows the average TD error $\bar{\delta}$ and return $\sum_{t=0}^{100} L_t$ per episode. The distributed learning succeeds in reducing the TD error and improving the overall performance of the network. Figure 4.7 shows $\Delta\phi_4$ and the deviation in power flow from area 3 to area 4, i.e., $\Delta P_{\text{tie},3,4} = P_{34}(\Delta\phi_3 - \Delta\phi_4)$, for the first and last episodes of training. The local load

for area 4 is not satisfiable by local generation, requiring power flow from area 3, and driving $\Delta\phi_4$ towards the bound. Initially, due to noise and an incorrect model, this leads to constraint violations. After training, it can be seen that agent 3 learnt to provide a larger power surge to agent 4, attenuating the decrease in $\Delta\phi_4$, and avoiding constraint violations. We also compare the performance of the resulting policy against a distributed nominal MPC controller (NMPC) and a distributed stochastic MPC controller (SMPC), based on the Scenario approach [172], to account for the noise on the load. Both use the true model and stage costs tuned as in [167]. Figure 4.8 demonstrates the performance and number of constraint violations over 100 episodes of the policy, the stochastic MPC controller, and the nominal MPC controller. Stochastic MPC improves on the performance of nominal MPC, as it better avoids violations. The proposed approach, despite starting from an incorrect model of the dynamics, distributively learnt a policy that achieves zero violations and improves the performance over the stochastic and nominal MPC controllers.

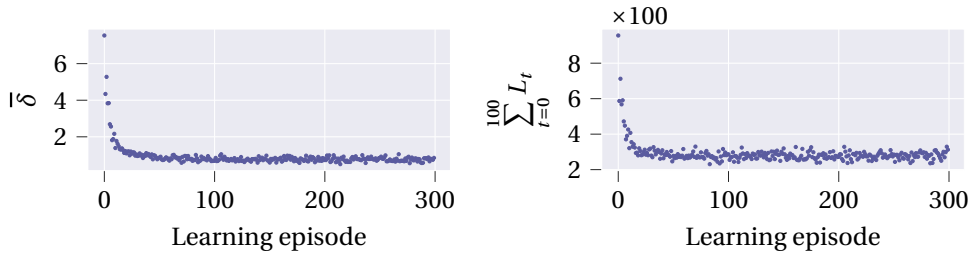

Figure 4.6.: Average TD error and the return per episode during training

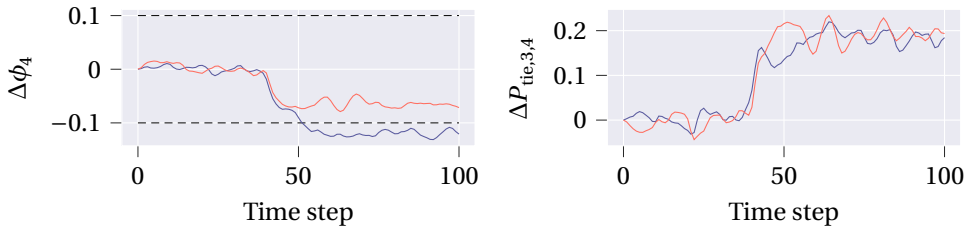

Figure 4.7.: States $\Delta\phi_4$ and $\Delta P_{\text{tie},3,4}$ for the first episode (purple) and the last episode (orange) of training

4.7. Conclusions

This chapter has extended the idea of using MPC-based RL to the multi-agent setting. We have proposed a novel approach to MARL via the use of distributed MPC as a distributed function approximator within Q-learning. A result on the optimal dual variables in

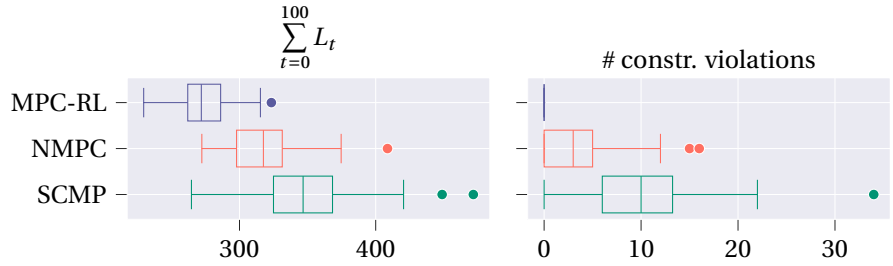

Figure 4.8.: Performance and number of constraint violations per episode, over 100 episodes, of the learnt policy, NMPC, and SMPC

ADMM is first presented. The structure of the distributed MPC function approximator is then detailed, and is shown to enable a distributed evaluation of the global value functions and the local policies. Finally, by using Q-learning to update the parametrisation of the MPC scheme, the parameter updates can also be performed fully distributively. The effectiveness of the newly proposed method is demonstrated on a numerical example.

A limitation of the proposed approach is the convexity requirement on the MPC scheme, restricting the class of systems for whom the optimal policy can be captured to linear systems with convex cost functions. Additionally, as both the ADMM and GAC algorithms use iterative communication between neighbours, the approach is less suited for applications with high communication costs. Future work will look at extending the idea to other RL methods, such as policy-based approaches, and a formal analysis on the propagation of errors from the finite termination of the ADMM and GAC algorithms.

4.A. Appendix

4.A.1. Equivalence of (4.3) and (4.7)

Define the aggregate of the local augmented states as $\bar{x} = \text{col}_{i\in\mathcal{M}}(\tilde{x}_i)$ and the feasible set as $\overline{\mathcal{X}} = \widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_M$. Problem (4.7) can then be written in the form of (4.3) as

$$\min_{\bar{x}\in\overline{\mathcal{X}},z} \quad \{f_{\text{ADMM}}(\bar{x}) + g_{\text{ADMM}}(z) \,:\, A\bar{x} + Bz = c\}, \tag{4.26}$$

where $f_{\text{ADMM}}(\bar{x}) = \sum_{i\in\mathcal{M}} F_i(\tilde{x}_i)$, $g_{\text{ADMM}}(\cdot)$ is the zero map, and $c = \mathbf{0}$. Furthermore, $A = \boldsymbol{I}_{q\times q}$, $q = \sum_{i\in\mathcal{M}} n(|\mathcal{N}_i| + 1)$, and $B \in \mathbb{R}^{q\times nM}$ is a block matrix that contains negative identity matrix entries picking out the corresponding global states, with all other entries being zero matrices. Clearly, matrix A is full column rank. For B, we note that every column has at least one identity entry, and that by rearranging its rows we can express it as $\begin{bmatrix} -\mathbf{I}_{nM\times nM} & B_+^\top \end{bmatrix}^\top$, where B_+ contains the remaining rows. Therefore, B is full column rank. The assumptions in Proposition 4.1 are hence satisfied.

4.A.2. Proof of Proposition 4.2

The proof involves showing that the Karush-Kuhn-Tucker (KKT) conditions for the local minimisation problems (4.8a), when aggregated for all agents, are equivalent to the

KKT conditions of the original problem (4.5). Then, due to strict convexity of both (4.8a) and (4.5), the primal and dual solutions are unique, and the equivalence between the optimal dual variables follows. We write the optimal primal and dual variables of (4.5) as $\left(\{x_i^\star\}_{i\in\mathcal{M}}, \{\lambda_i^\star\}_{i\in\mathcal{M}}, \{\mu_i^\star\}_{i\in\mathcal{M}}\right)$, and the optimal primal and dual variables of (4.8a) for agent i, *at convergence of ADMM*, as $\left(x_i^\star, \{x_j^{(i),\star}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i^\star, \hat{\mu}_i^\star\right)$. Primal variable equivalence is given by Proposition 4.1, i.e., the $x_i^\star$'s are equivalent for both problems. Also, optimal local copies are equal to their true values

$$\{x_j^{(i),\star}\}_{j\in\mathcal{N}_i} = \{x_j^\star\}_{j\in\mathcal{N}_i} \quad i\in\mathcal{M}. \tag{4.27}$$

4

First, we state the KKT conditions of (4.5). For ease of exposition, let use define

$$\mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda_i, \mu_i\right) = F_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) + \lambda_i^\top h_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) + \mu_i^\top g_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right), \tag{4.28}$$

The KKT conditions are

$$h_i\left(x_i^\star, \{x_j^\star\}_{j\in\mathcal{N}_i}\right) \le 0, \quad i\in\mathcal{M}, \tag{4.29}$$

$$g_i\left(x_i^\star, \{x_j^\star\}_{j\in\mathcal{N}_i}\right) \le 0, \quad i\in\mathcal{M}, \tag{4.30}$$

$$\lambda_i^\star \ge 0, \quad i\in\mathcal{M}, \tag{4.31}$$

$$\nabla_x \sum_{i\in\mathcal{M}} \mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda_i, \mu_i\right)\Big|_{\{x_i,\lambda_i,\mu_i\}_{i\in\mathcal{M}}=\left\{x_i^\star,\lambda_i^\star,\mu_i^\star\right\}_{i\in\mathcal{M}}} = 0, \tag{4.32}$$

where $x = \begin{bmatrix} x_1^\top & \dots & x_M^\top \end{bmatrix}^\top$. Now consider the composition of the KKT conditions of each agent's local minimisation (4.8a). First, primal feasibility, dual feasibility, and complementary slackness for agent i:

$$\begin{aligned} h_i\left(x_i^\star, \{x_j^{(i),\star}\}_{j\in\mathcal{N}_i}\right) &\le 0, \\ g_i\left(x_i^\star, \{x_j^{(i),\star}\}_{j\in\mathcal{N}_i}\right) &= 0, \\ \hat{\lambda}_i^\star &\ge 0, \\ \hat{\lambda}_i^{\star,\top} h_i\left(x_i^\star, \{x_j^{(i),\star}\}_{j\in\mathcal{N}_i}\right) &= 0. \end{aligned} \tag{4.33}$$

Inserting the equivalence (4.27) into (4.33), we find that the composition of these conditions for all agents are the original problem's conditions (4.29)-(4.31). Let us bundle the primal and dual variables as $p = \left(\{x_i\}_{i\in\mathcal{M}}, \{\lambda_i\}_{i\in\mathcal{M}}, \{\mu_i\}_{i\in\mathcal{M}}\right)$, such that the stationarity condition (4.32) reads

$$\nabla_x \sum_{i\in\mathcal{M}} \mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda_i, \mu_i\right)\Big|_{p=p^\star} = 0. \tag{4.34}$$

By decomposing the derivative operator and considering each row ℓ of (4.34) we have

$$\frac{\partial}{\partial x_\ell} \sum_{i\in\mathcal{M}} \mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda_i, \mu_i\right)\Big|_{p=p^\star} = 0. \tag{4.35}$$

Observing that $\frac{\partial}{\partial x_\ell}\mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda_i, \mu_i\right)$ if $i\notin\mathcal{N}_\ell\cup\ell$, from (4.35) we have

$$\frac{\partial}{\partial x_\ell}\mathscr{F}_\ell\left(x_\ell, \{x_j\}_{j\in\mathcal{N}_\ell}, \lambda_\ell, \mu_\ell\right)\Big|_{p=p^\star} + \sum_{i\in\mathcal{N}_\ell} \frac{\partial}{\partial x_\ell}\mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda_i, \mu_i\right)\Big|_{p=p^\star} = 0. \tag{4.36}$$

We highlight that the partial derivative terms within the sum are not zero, as x_ℓ is one of the elements of $\{x_j\}_{j\in\mathcal{N}_i}$. We will now show that the condition (4.36) for each ℓ can be reconstructed by a unique subset of the KKT conditions arising in the agents' local minimisations. Consider the stationarity conditions of the agents' local minimisations. The primal and dual variables for agent i are grouped as $p_i = \left(\tilde{x}_i, \hat{\lambda}_i, \hat{\mu}_i\right)$. The stationarity condition for agent i's local minimisation can be written as

$$\nabla_{\tilde{x}_i}\left(\mathscr{F}_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right) + y_i^{\tau,\top}\tilde{x}_i + \frac{\rho}{2}\|\tilde{x}_i - \tilde{z}_i^\tau\|_2^2\right)\Big|_{p_i=p_i^\star} = 0. \tag{4.37}$$

Evaluating the derivative of the second term, we have

$$\nabla_{\tilde{x}_i}\mathscr{F}_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right)\Big|_{p_i=p_i^\star} + y_i^\tau + \rho\left(\tilde{x}_i^\star - \tilde{z}_i^\tau\right) = 0. \tag{4.38}$$

From Proposition 4.1, we have that at convergence of ADMM, $\tilde{x}_i^\star - \tilde{z}_i^\star = 0$. Decomposing the derivative operator we have

$$\begin{bmatrix} \frac{\partial}{\partial x_i} \\ \mathrm{col}_{j\in\mathcal{N}_i}\left(\frac{\partial}{\partial x_j^{(i)}}\right)\end{bmatrix}\mathscr{F}_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right)\Big|_{p_i=p_i^\star} + \begin{bmatrix} y_i^\tau \\ \mathrm{col}_{j\in\mathcal{N}_i}(y_j^{(i),\tau})\end{bmatrix} = 0. \tag{4.39}$$

We denote this as $\Psi_i = \begin{bmatrix}\Psi_i^{(i),\top} & \mathrm{col}_{j\in\mathcal{N}_i}^\top\left(\Psi_j^{(i)}\right)\end{bmatrix}^\top$, where $\Psi_j^{(i)}$ is the row of (4.39) corresponding to the partial derivative with respect to $x_j^{(i)}$. Let us sum all rows of the agents' local stationarity conditions which correspond to the partial derivative with respect to x_ℓ, or a copy of x_ℓ, to get

$$\Psi_\ell^{(\ell)} + \sum_{i\in\mathcal{N}_l}\Psi_i^{(\ell)} = 0. \tag{4.40}$$

The result is equal to zero as each term in the sum is zero. The sum expression reads

$$\frac{\partial}{\partial x_\ell}\mathscr{F}_\ell\left(x_\ell, \{x_j^{(\ell)}\}_{j\in\mathcal{N}_\ell}, \hat{\lambda}_\ell, \hat{\mu}_\ell\right)\Big|_{p_\ell=p_\ell^\star} + \sum_{i\in\mathcal{N}_\ell}\frac{\partial}{\partial x_\ell^{(i)}}\mathscr{F}_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right)\Big|_{p_i=p_i^\star} + y_\ell^\tau + \sum_{i\in\mathcal{N}_l} y_\ell^{(i),\tau} = 0, \tag{4.41}$$

where again we highlight that the partial derivative terms in the sum are not zero, as $\ell \in \mathcal{N}_i$. It is shown in [30] that the sum of the dual ADMM variables corresponding to the same global variables is zero after the first iteration of the procedure, i.e., $y_\ell^\tau + \sum_{i\in\mathcal{N}_l} y_\ell^{(i),\tau} = 0$, $\tau > 1$. Hence, at convergence, we have

$$\frac{\partial}{\partial x_\ell}\mathscr{F}_\ell\left(x_\ell, \{x_j^{(\ell)}\}_{j\in\mathcal{N}_\ell}, \hat{\lambda}_\ell, \hat{\mu}_\ell\right)\Big|_{p_\ell=p_\ell^\star} + \sum_{i\in\mathcal{N}_\ell}\frac{\partial}{\partial x_\ell^{(i)}}\mathscr{F}_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right)\Big|_{p_i=p_i^\star} = 0. \tag{4.42}$$

Substituting the equivalence of the local copies (4.27) and observing that

$$\frac{\partial}{\partial x_\ell^{(i)}}\mathscr{F}_i\left(x_i, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right)\Big|_{p_\ell=p_\ell^\star} = \frac{\partial}{\partial x_\ell}\mathscr{F}_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}, \hat{\lambda}_i, \hat{\mu}_i\right)\Big|_{p_\ell=p_\ell^\star}, \tag{4.43}$$

we obtain the equivalent stationarity condition from the ℓ-th row of (4.34). Repeating the sum in (4.40) for each agent, we reconstruct the equivalent rows of (4.34). We then have equivalence between the stationarity conditions on $\left(\{\hat{\lambda}_i^\star\}_{i\in\mathcal{M}}, \{\hat{\mu}_i^\star\}_{i\in\mathcal{M}}\right)$ and on $\left(\{\lambda_i^\star\}_{i\in\mathcal{M}}, \{\mu_i^\star\}_{i\in\mathcal{M}}\right)$. This concludes the proof.

5

Reinforcement learning-based model predictive control for greenhouse climate control

Greenhouse climate control is concerned with maximising performance in terms of crop yield and resource efficiency. One promising approach is model predictive control (MPC), which leverages a model of the system to optimise the control inputs, while enforcing physical constraints. However, prediction models for greenhouse systems are inherently inaccurate due to the complexity of the real system and the uncertainty in predicted weather profiles. For model-based control approaches such as MPC, this can degrade performance and lead to constraint violations. Existing approaches address uncertainty in the prediction model with robust or stochastic MPC methodology; however, these necessarily reduce crop yield due to conservatism and often bear higher computational loads. In contrast, learning-based control approaches, such as reinforcement learning (RL), can handle uncertainty naturally by leveraging data to improve performance. This chapter proposes an MPC-based RL control framework to optimise the climate control performance in the presence of prediction uncertainty. The approach employs a parametrised MPC scheme that learns directly from data, in an online fashion, the parametrisation of the constraints, prediction model, and optimisation cost that minimises constraint violations and maximises climate control performance. Simulations show that the approach can learn an MPC controller that significantly outperforms the current state-of-the-art in terms of constraint violations and efficient crop growth.

This chapter is based on [135].

5.1. Introduction

Greenhouse climate control presents a key opportunity to address the growing world population's food production requirements in a changing climate, and the grand societal challenge of efficient energy consumption. With modern greenhouses equipped with actuation systems such as heating, ventilation, and CO_2 injection, effective control approaches can lead to high crop yield in an energy-efficient manner. However, the control challenge is difficult as the process dynamics are highly nonlinear and complex [204], and the climate variables, such as temperature and humidity, must be effectively constrained to avoid damage to crops due to, e.g., spread of diseases [46, 80, 229]. While traditional methods, e.g., on-off and PID control, have been used for low-level regulation, these strategies are not rooted in optimal control and are in general unable to deliver optimal performance and to systematically handle complex state and/or input constraints [84, 113].

Model predictive control (MPC) is an optimisation-based control methodology that naturally handles multi-input-multi-output systems with state, input, and output constraints [28]. It has seen huge theoretical success and application in process control, and has been proposed to solve greenhouse control challenges [26, 78, 146]. However, MPC relies heavily on an accurate prediction model, while in a greenhouse prediction model there always exist uncertainties due to, e.g., modelling error and inaccurate weather forecasting. For model-based control, an incorrect model can lead the controller to drive the system to an undesired point of operation, possibly violating constraints. Furthermore, as constraints often represent the validity range for the accuracy of the prediction model, violations of the constraints can lead to degraded performance when applied to the real system [229].

Some existing works have addressed prediction uncertainty only in the context of external disturbances, such as weather predictions [46, 112]. In [71], the uncertainty in market prices is addressed with a scenario-based stochastic MPC controller; however, the controller is based on mixed-integer optimisation, requiring significant computational efforts. Alternatively, the following existing works have explored uncertainty stemming from an incorrect physical model of the system. In [84], the robustness of predictive control for a greenhouse in the presence of model uncertainties is considered empirically, however no mechanism to compensate for the prediction error is introduced. In [131, 132], a neural network is used as a prediction model, with a robust MPC controller addressing the prediction uncertainty; however, a min-max robust MPC approach is used, which is inherently conservative. In [229], an approach to mitigating the negative effects of model uncertainty is presented using online parameter estimation. However, only a small subset of the (possibly) uncertain model parameters are estimated. Of particular note, in [27], parametric uncertainty in all model parameters is addressed. A robust sample-based MPC controller is proposed to incorporate the uncertainty into the control approach. However, the resulting control scheme is unable to adequately reduce constraint violations, and results in less crop yield due to conservativeness. More recently, [198] has proposed a chance-constrained stochastic MPC formulation to address parametric uncertainties in greenhouse production systems. However, this approach relies on linearisation of the prediction model, further increasing prediction uncertainty. Additionally, the chance constrained formulation leads to a

computational load that is higher than traditional MPC schemes.

In contrast to MPC, reinforcement learning (RL) is a model-free control methodology where a control policy is learnt from data observed from the system [197]. RL controllers naturally handle uncertainty and adapt to changing conditions with no additional mechanisms, as they are learnt through direct interaction with the real system. For complex systems with large continuous state and action spaces, the state-of-the-art for RL uses deep neural networks (DNNs) as function approximators to represent the controller. With the power of DNNs as general function approximators, this idea has seen unprecedented success on previously unsolved problems, e.g., the games of chess and Go [179]. The power and inherent uncertainty handling of RL has been identified as useful for greenhouse climate control, with [215] proposing a DNN-based RL approach based on deep deterministic policy gradient (DDPG) [120], and [147] drawing a comparison between MPC and DDPG. However, an inherent drawback for RL controllers is the absence of theoretical guarantees on the satisfaction of constraints and a lack of interpretability due to the black-box DNN function approximation. In the context of greenhouse control, this means growers have no guarantee that the automated controller will effectively constrain climate variables to safe ranges, with potential negative implications on crop health and profit.

Recently, [75] proposed and justified an integrated MPC and RL control paradigm, where the MPC controller serves as a function approximator for the optimal policy in model-based RL. In such a scheme, the MPC controller's optimisation problem acts both as policy provider, picking actions based on a state, and value function approximator, estimating "how good" it is to be in a given state. The learning algorithm, e.g., Q-learning [220], is tasked with adjusting the parametrisation of the MPC controller in an effort to discover the optimal control policy, thus improving closed-loop performance in a data-driven fashion. In this way, despite the presence of mismatches between the prediction model and the real system, the MPC control scheme is able to learn and deliver, at convergence, the optimal policy and value functions of the underlying RL task, granted the MPC parametrisation is rich enough. In contrast to DNN-based RL, the MPC scheme at the core of this approach provides the option of integrating prior information that may be known on the system in the form of, e.g., an expert-based, possibly imperfect, prediction model. Moreover, MPC-based agents are in general more amenable to analysis and certification in terms of stability and constraint satisfaction [28]. Finally, MPC-based controllers can take constraints into account in an explicit and structured way, which DNNs are generally incapable of doing. The above benefits render this methodology suitable for greenhouse climate control, where an inaccurate prediction model is known, and climate variables must be constrained.

Therefore, in this chapter we propose an integrated MPC and RL framework to address the problem of greenhouse climate control under parametric uncertainty stemming from uncertain weather predictions and modelling mismatches. Specifically:

1. To the best of the authors' knowledge, a combined MPC and RL approach for greenhouse climate control in the presence of uncertainty is proposed for the first time. A parametrised MPC scheme, inspired by [75], is crafted to serve as policy provider and value function approximator in an RL formulation of the greenhouse climate control problem. A second-order Q-learning algorithm is leveraged to

adjust the parametrisation of the MPC scheme online, automatically learning a control policy. The approach provides an adaptive and high-performing climate controller that minimises potentially dangerous constraint violations, without negatively affecting crop growth due to robustness conservatism, and without relying on an expensive-to-acquire accurate prediction model. Additionally, in contrast to DNN-based learning approaches, the behaviour of the resulting controller can be interpreted by analysing the learnt parametrisation of the constraints, prediction model, and cost function.

2. The approach is then validated in simulation, and compared against both model-based MPC and model-free RL state-of-the-art controllers. The results demonstrate the effectiveness of the approach, with the proposed methodology outperforming existing controllers from the literature in both constraint satisfaction and efficient crop growth.

Compared to the state-of-the-art MPC formulations [27, 71, 131, 132, 198], instead of addressing uncertainty in the parametrisation in a robust or stochastic fashion, the proposed methodology adapts its policy via RL in order to improve closed-loop performance. This has the distinct advantage of yielding less conservative control schemes while retaining low computational complexity. In comparison with DNN-based approaches such as [147, 216], our method integrates MPC as function approximator in the RL algorithm, fostering a model-based approach that is more suitable for learning high-performance, constraint-abiding policies.

The chapter is structured as follows. In Section 5.2 relevant background is provided on the greenhouse model used, and on the combined MPC and RL paradigm from [75]. The problem we address is formally defined in Section 5.3. Section 5.4 presents the proposed methodology, which is then applied and assessed extensively in simulation in Section 5.5. Finally, conclusions and future work directions are given in Section 5.6.

5.2. Background

This section describes the greenhouse model considered in this chapter. Additionally, background theory on combining MPC and RL is provided, upon which the methodology proposed in this chapter is built.

5.2.1. Lettuce greenhouse model

We consider a greenhouse for lettuce growing, with the continuous-time model, presented in [204], given in Section 5.A.2. While the continuous-time model is used in all simulations, for control purposes a discrete-time model is considered

$$\begin{aligned} x(k+1) &= f\left(x(k), u(k), d(k), p\right), \\ y(k) &= g\left(x(k), p\right), \end{aligned} \tag{5.1}$$

with $x \in \mathbb{R}^4$ the state, $u \in \mathbb{R}^3$ the control input, $d \in \mathbb{R}^4$ the weather disturbance, and $y \in \mathbb{R}^4$ the output. Furthermore, $p \in \mathbb{R}^{28}$ is a set of model parameters, and $k \in \mathbb{Z}^+$ is the discrete-time counter for discrete time steps of $\Delta t = 900\,\mathrm{s}$ (15 minutes). The nonlinear

functions f and g, and the model parameters p, are given in the Appendix. The physical meaning of the states, outputs, inputs, and disturbances is summarised in Table 5.1. Estimation is out the scope of this chapter, and it is assumed, as in [27], that at each time step k a perfect estimate of the state $x(k)$ is available.

Table 5.1.: Physical meaning of the state x, output y, input u, and disturbance d

x_1	dry-weight ($\mathrm{kg\,m^{-2}}$)	y_1	dry-weight ($\mathrm{g\,m^{-2}}$)	d_1	radiation ($\mathrm{W\,m^{-2}}$)
x_2	indoor CO_2 ($\mathrm{kg\,m^{-3}}$)	y_2	indoor CO_2 (‰)	d_2	outdoor CO_2 ($\mathrm{kg\,m^{-3}}$)
x_3	indoor temperature (°C)	y_3	indoor temperature (°C)	d_3	outdoor temperature (°C)
x_4	indoor humidity ($\mathrm{kg\,m^{-3}}$)	y_4	indoor humidity (%)	d_4	outdoor humidity ($\mathrm{kg\,m^{-3}}$)
u_1	CO_2 injection ($\mathrm{mg\,m^{-2}\,s^{-1}}$)	u_2	ventilation ($\mathrm{mm\,s^{-1}}$)	u_3	heating ($\mathrm{W\,m^{-2}}$)

5.2.2. MPC as a function approximator in RL

Consider discrete-time system dynamics as a Markov Decision Process (MDP) [197] with continuous state $s \in \mathbb{R}^n$, continuous action $a \in \mathbb{R}^m$, and state transitions $s, a \to s_+$ with the underlying conditional probability density

$$\mathbb{P}(s_+ \mid s, a) : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to [0, 1]. \tag{5.2}$$

Consider a deterministic policy $\pi_\theta : \mathbb{R}^n \to \mathbb{R}^m$ parametrised by $\theta \in \mathbb{R}^l$. Selecting actions based on this policy will cause the system to visit the MDP's states with a given distribution, denoted η_{π_θ}. The performance of such a policy is defined as [197]

$$J(\pi_\theta) = \mathbb{E}_{\eta_{\pi_\theta}} \left[\sum_{k=0}^{N_\mathrm{s}} \gamma^k L(s_k, \pi_\theta(s_k)) \right], \tag{5.3}$$

where s_k is the state at time step k, $L(s, a) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ the stage cost, $\gamma \in (0, 1]$ the discount factor, and N_s the number of time steps considered in a task. The RL task is then to find the optimal policy $\pi_\theta^\star$ as

$$\pi_\theta^\star = \arg\max_\theta \quad J(\pi_\theta). \tag{5.4}$$

The familiar notions of state- and action-value functions [197] are defined respectively as

$$Q_\theta(s_k, a_k) = L(s_k, a_k) + \mathbb{E}_{\eta_{\pi_\theta}} \left[\sum_{\tau=k+1}^{N_\mathrm{s}} \gamma^{\tau-k} L(s_k, \pi_\theta(s_k)) \right], \tag{5.5}$$

and $V_\theta(s_k) = Q_\theta\big(s_k, \pi_\theta(s_k)\big)$. While DNNs are the most common choice for representing the policy and value functions [120], their black-box nature does not facilitate the injection of prior information, e.g., approximate prediction models, nor is it conducive to an interpretable policy and the addition of constraints. To account for these drawbacks, [75] proposed the use of an MPC scheme in place of a DNN.

Consider the following MPC problem approximating the value function, parametrised by θ, $V_\theta : \mathbb{R}^n \to \mathbb{R}$ as

$$V_\theta(s) = \min_{\mathbf{X},\mathbf{U},\boldsymbol{\sigma}} \quad \lambda_\theta(x(0)) + \sum_{k=0}^{N-1} \gamma^k \left(L_\theta(x(k),u(k)) + \omega^\top \sigma(k)\right) + \gamma^N \left(V_\theta^{\mathrm{f}}(x(N)) + w_{\mathrm{f}}^\top \sigma(N)\right) \tag{5.6a}$$

$$\text{s.t.} \quad x(0) = s, \tag{5.6b}$$

$$x(k+1) = f_\theta(x(k),u(k)), \quad k = 0,\dots,N-1, \tag{5.6c}$$

$$h_\theta(x(k),u(k)) \le \sigma(k), \quad k = 0,\dots,N-1, \tag{5.6d}$$

$$h_\theta^{\mathrm{f}}(x(N)) \le \sigma(N), \tag{5.6e}$$

$$\sigma(k) \ge 0, \quad k = 0,\dots,N, \tag{5.6f}$$

where slack variable $\sigma(k)$ softens the inequality constraint for time step k, and the vectors $\mathbf{X}$, $\mathbf{U}$ and $\boldsymbol{\Sigma}$ respectively collect the states, actions, and slack variables over the horizon N. Problem (5.6) is solved numerically online to generate the value $V_\theta(s)$. In (5.6), λ_θ is an initial cost term, L_θ is the stage cost, and V_θ^{f} is a terminal cost approximation, all of which are parametrised by θ. Furthermore, f_θ is the model approximation, and h_θ, h_θ^{f} are inequality constraints. Lastly, w and w_{f} are the weights of the slack variable in the objective. Note that the formulation (5.6) is general, i.e., the dimension of θ, and how it enters into the respective functions in (5.6), is not made explicit. In Section 5.4 we will introduce a concrete realisation, designed for the greenhouse climate control problem.

Given (5.6), the action value function Q_θ and the policy π_θ, satisfying the fundamental equalities of the Bellman equations [75], are defined as follows:

$$\begin{aligned} Q_\theta(s,a) = \min_{X,U,\Sigma} \quad & (5.6a) \\ \text{s.t.} \quad & (5.6b) - (5.6f), \\ & u(0) = a, \end{aligned} \tag{5.7}$$

$$\pi_\theta(s) = \arg\min_a \quad Q_\theta(s,a). \tag{5.8}$$

Therefore, in RL terms, the parametric MPC scheme acts as policy provider for the learning agent, whose goal is to modify the parameters θ of the controller in order to minimise (5.4). Various forms of RL [197] exist that solve this problem directly or indirectly via iterative updates

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{i=1}^{m} \psi(s_i, a_i, s_{i+1}, \theta), \tag{5.9}$$

where $\alpha \in \mathbb{R}_+$ is the learning rate, z denotes the number of observations used in the update (i.e., a batch of observations), and ψ captures the controller's performance and varies with the specific algorithm. For example, in recursive Q-learning we have that $m = 1$ and $\psi = \delta_i^2$, where the temporal-difference (TD) error

$$\delta_i = L(s_i, a_i) + \gamma V_{\theta'}(s_{i+1}) - Q_\theta(s_i, a_i) \tag{5.10}$$

captures the estimation error of the value functions [197]. Note that the value function estimate is evaluated for a frozen copy of the parametrisation $\theta' = \theta$ to prevent it from contributing to the gradient update.

5.3. Problem formulation

We address the problem of optimal greenhouse climate control for crop yield and resource efficiency. As in [27], we consider the predictive uncertainty, stemming from uncertain weather predictions and modelling errors, to be captured by parametric uncertainty[1] in the model parameters p. Specifically, the true values for p are assumed to be unknown.

Growth cycles of 40 days are considered, where control inputs are computed at 15 minute time steps, i.e., growth cycles of $N_s = 3840$ time steps. This duration is in line with the literature [27, 147], and represents a standard lettuce growth cycle, at the end of which the lettuce is harvested and sold, generating economic profit. During each growth cycle, we wish to maximise the yield and minimise the violations of constraints on the system outputs, whilst minimising the cost associated with control signals. The system outputs are constrained as $y_{\min}(k) \leq y(k) \leq y_{\max}(k)$, with [27, 147]

$$\begin{aligned} y_{\min}(k) &= \begin{bmatrix} 0 & 0 & y_{3,\min}(d_1(k)) & 0 \end{bmatrix}^\top, \\ y_{\max}(k) &= \begin{bmatrix} \infty & 1.6 & y_{3,\max}(d_1(k)) & 70 \end{bmatrix}^\top, \end{aligned} \tag{5.11}$$

and with the time-varying third element defined as

$$\begin{aligned} y_{3,\min}(d) &= \begin{cases} 10, & \text{if } d < 10, \\ 15, & \text{if } d \geq 10, \end{cases} \\ y_{3,\max}(d) &= \begin{cases} 15, & \text{if } d < 10, \\ 20, & \text{if } d \geq 10. \end{cases} \end{aligned} \tag{5.12}$$

These time varying constraints are common in the literature; they reflect that inside the greenhouse it is colder during the night than during the day [27, 147, 175]. The inputs are constrained as $u_{\min} \leq u(k) \leq u_{\max}$, with [27, 147]

$$\begin{aligned} u_{\min} &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, \\ u_{\max} &= \begin{bmatrix} 1.2 & 7.5 & 150 \end{bmatrix}^\top. \end{aligned} \tag{5.13}$$

Finally, the input rate is constrained as [27, 147]

$$|u(k+1) - u(k)| \leq \frac{1}{10} u_{\max}. \tag{5.14}$$

Define the following performance indicators:

[1] The choice to represent modelling error and weather prediction inaccuracy via the uncertainty in p results in a prediction model in which even the parameters that may represent known physical quantities, e.g., the ideal gas constant p_{23}, being unknown.

- Final yield y_1 (N_s)
- Constraint violations Ψ
- Economic profit indicator P.

The final yield y_1 (N_s) is the dry lettuce weight at the end of a growth cycle of N_s time steps. The constraint violations indicator Ψ is defined as

$$\Psi = \sum_{k=0}^{N_s} \sum_{i=1}^{4} \left(\max\left\{0, \frac{y_i(k) - y_{i,\max}(k)}{y_{i,\max}(k) - y_{i,\min}(k)}\right\} + \max\left\{0, \frac{y_{i,\min}(k) - y_i(k)}{y_{i,\max}(k) - y_{i,\min}(k)}\right\}\right). \quad (5.15)$$

This performance indicator captures the magnitude of violations on the output constraints over a growth cycle. Finally, the economic profit indicator[2] P is defined as [147, 205]

$$P = c_{\text{price},1} + c_{\text{price},2}\, y_1(N_s) - \sum_{k=0}^{N_s} \left(c_{\text{CO}_2} u_1(k) + c_q u_3(k)\right) \Delta t, \quad (5.16)$$

where the relation between auction price and harvest weight of lettuce is modelled linearly with coefficients $c_{\text{price},1}$ and $c_{\text{price},2}$, and the financial cost of the climate conditioning equipment is linearly related to the amount of energy and carbon dioxide put into the system, weighted by prices c_q and c_{CO_2}, respectively. Note that in P no cost is associated to natural ventilation used for cooling and dehumidification, i.e., u_2 [71]. The values of the coefficients of this economic model are given in Table 5.2, and more details are available in, e.g., [205, Section 2.1]. This performance indicator represents the monetary value of a growth cycle, and captures the efficiency of the control with respect to the use of costly actuators.

Table 5.2.: Coefficient values for the economic profit indicator P (the former Dutch currency Hfl, the Guilder, is used for adherence to the literature [27, 147, 198, 204, 205])

Symbol	Value	Unit
$c_{\text{price},1}$	1.8	$\text{Hfl}\,\text{m}^{-2}$
$c_{\text{price},2}$	1.6	$\text{Hfl}\,\text{kg}^{-1}$
c_q	$6.35 \cdot 10^{-9}$	$\text{Hfl}\,\text{J}^{-1}$
c_{CO_2}	$42 \cdot 10^{-2}$	$\text{Hfl}\,\text{kg}^{-1}$
Δt	900	s

[2]Note that P represents a profit indicator per square meter, as the lettuce model in this chapter is normalised by the surface area (see Section 5.2.1 and [204, 205]).

5.4. Methodology

To address the issues caused by inaccurate knowledge of the true prediction model parameters p for model-based MPC controllers, we propose a novel parametrised MPC scheme for the greenhouse climate control problem. Then, leveraging closed loop data, we show how the parameter values can be learnt using RL techniques [75], compensating for the performance loss introduced by an inaccurate prediction model.

5.4.1. Greenhouse climate control as an RL problem

In order to apply an RL methodology to learn the MPC parametrisation, the greenhouse climate control must be modeled as an RL task, i.e., as an MDP defined as in Section 5.2.2. Trivially, the greenhouse input variable u corresponds directly to the actions a in the RL context. Define the concatenation of the current state, and the current and previous outputs, as the RL state:

$$s(k) = \begin{bmatrix} x^\top(k) & y^\top(k) & y^\top(k-1) \end{bmatrix}^\top. \tag{5.17}$$

Furthermore, the state transitions are determined by the true system model (5.1), where the exact probabilities are conditioned on the weather disturbance d. Note that, to enforce the control rate constraint (5.14) in the MDP, the input variable is clipped prior to being applied to the system.

The stage cost L requires particular attention, as it implicitly defines the optimisation problem (5.4) that the RL agent is tasked with solving. Crafting the stage cost L such that the resulting RL policy performs well with respect to the performance indicators defined in Section 5.3 is a design challenge. Consider the stage cost

$$L(s(k), a(k)) = L_{y_1}(s(k)) + L_u(a(k)) + L_\Psi(s(k)). \tag{5.18}$$

The function

$$L_{y_1}(s(k)) = -c_{\delta,y_1}(y_1(k) - y_1(k-1)), \tag{5.19}$$

with $c_{\delta,y_1} > 0$, rewards the step-wise increase in dry lettuce weight. The function

$$L_u(a(k)) = c_u^\top u(k), \tag{5.20}$$

with $c_u \in \mathbb{R}^3 > 0$, penalises the control inputs, while the function

$$L_\Psi(s(k)) = \omega^\top \left(\max\left\{0, \frac{y(k) - y_{\max}(k)}{y_{\max}(k) - y_{\min}(k)}\right\} + \max\left\{0, \frac{y_{\min}(k) - y(k)}{y_{\max}(k) - y_{\min}(k)}\right\} \right) \tag{5.21}$$

penalises constraint violations, with $\omega \in \mathbb{R}^4 > 0$, and with the max operator and vector division performed element-wise. The constants c_{δ,y_1}, c_u, and ω are then hyperparameters that are tuned such that the RL policy performs well on the performance indicators in Section 5.3. Lastly, the RL policy π_θ, and value functions V_θ and Q_θ, are addressed via the parametrised MPC scheme, which is introduced next.

5.4.2. Parametrised MPC scheme

In this section we introduce the parametrised MPC scheme that, following the theory outlined in Section 5.2.2, serves as policy provider and value function approximator for the RL task outlined in Section 5.4.1.

Consider the following parametrised MPC scheme, a concrete realisation of (5.6), representing the value function, with prediction horizon $N \in \mathbb{Z}^+$ and discount factor $\gamma \in (0,1]$:

$$
\begin{aligned}
V_\theta(s) = \min_{\mathbf{Y},\mathbf{X},\mathbf{U},\boldsymbol{\Sigma}} \quad & \theta_0 - \theta_{\delta,y_1} \sum_{k=1}^{N} \gamma^k \left(y_1(k) - y_1(k-1)\right) + \theta_u^\top \sum_{k=0}^{N-1} \gamma^k u(k) \\
& + \theta_\omega^\top \sum_{k=0}^{N} \gamma^k \frac{\sigma(k)}{y_{\text{range}}} + \theta_{y_1,\text{f}} \gamma^N V_{\theta,\text{f}}(y_1(N)) && \text{(5.22a)} \\
\text{s.t.} \quad & x(0) = s, && \text{(5.22b)} \\
& x(k+1) = f\left(x(k), u(k), d(k), \theta_p\right), && k = 0,\dots,N-1, \quad \text{(5.22c)} \\
& u_{\min} \le u(k) \le u_{\max}, && k = 0,\dots,N-1, \quad \text{(5.22d)} \\
& -\delta u \le u(k) - u(k-1) \le \delta u, && k = 1,\dots,N-1, \quad \text{(5.22e)} \\
& y(k) = g\left(x(k), \theta_p\right), && k = 0,\dots,N, \quad \text{(5.22f)} \\
& y_{\min}(k) - \sigma(k) \le y(k) \le y_{\max}(k) + \sigma(k), && k = 0,\dots,N, \quad \text{(5.22g)} \\
& \sigma(k) \ge 0, && k = 0,\dots,N, \quad \text{(5.22h)}
\end{aligned}
$$

where[3] $y_{\text{range}} = \begin{bmatrix} \infty & 1.6 & 5 & 70 \end{bmatrix}^\top$. The cost (5.22a) rewards step-wise lettuce weight increase, and penalises constraint violations and control inputs. The first and last terms, θ_0 and $V_{\theta,\text{f}}$, are additional initial offset and terminal costs respectively, enriching the parametrisation to help the MPC scheme capture the true RL value functions. In particular, θ_0 is required in order to learn the (possibly non-zero) offset in the value function due to its economic nature (see [75] for more theoretical details), while $V_{\theta,\text{f}}$ encodes the cost-to-go. The parameters $\theta_{\delta_{y_1}}, \theta_u, \theta_\omega$, and $\theta_{y_{1,\text{f}}}$ weight the contributions of each cost term. Since in general $\theta_p \neq p$, the constraints for the dynamics and output equations, (5.22c) and (5.22f), are parametrised by θ_p in place of the true model parameters p. Hence, the predicted state and output trajectories from solving (5.22) do not necessarily respect the dynamics and output functions of the true system. The parameter θ_p then provides a degree of freedom to optimise the predicted trajectories for closed-loop performance, using closed-loop data.

Remark 5.1. *Note that the MPC parameters $\theta_{\delta,y_1}, \theta_u, \theta_\omega$, and θ_p are distinct from the parameters in the RL stage cost $c_{\delta,y_1}, c_u, \omega$, and the true model parameters p. While they represent equivalent notions, the MPC parameters will be adjusted to optimise closed-loop performance and will not, in general, converge to the corresponding values in the RL stage-cost and the true model. Indeed, in [75] it is stressed that the prediction model yielding the best closed-loop performance for an MPC controller is not necessarily the*

[3] The ∞ in y_{range} causes the first component of the violation penalty to be always zero, as the first output is unbounded.

one with the smallest prediction error. Hence, even if instantiated with accurate model parameters $\theta_p = p$, it is likely the RL agent would adjust θ_p regardless, optimising for closed-loop performance.

The terminal cost $V_{\theta,\mathrm{f}}$ is added to capture the future reward from lettuce weight increase for the remainder of the growth cycle, occurring after the prediction horizon of the MPC controller. The step-wise reward for lettuce growth for the remainder of the cycle can be simplified as

$$\sum_{k=N+1}^{N_\mathrm{s}} \gamma^k L_{y_1}(s(k)) = -c_{\delta,y_1} \sum_{k=N+1}^{N_\mathrm{s}} \gamma^k \left(y_1(k) - y_1(k-1)\right), \tag{5.23}$$

$$= -c_{\delta,y_1} \left(\gamma^{N_\mathrm{s}} y_1(N_\mathrm{s}) - \gamma^{N+1} y_1(N) + \sum_{k=N+1}^{N_\mathrm{s}-1} \left(\gamma^k - \gamma^{k+1}\right) y_1(k) \right), \tag{5.24}$$

$$\approx -c_{\delta,y_1} \left(y_1(N_\mathrm{s}) - y_1(N)\right), \tag{5.25}$$

where, for the sake of simplicity, it is assumed that $\gamma \approx 1$, which is common when a non-myopic policy is desired [197]. Of course, (5.25) cannot be used as terminal cost directly, as the final yield $y_1(N_\mathrm{s})$ is not known and varies depending on the weather disturbance and the control inputs. As the value functions attempt to capture the expected cost under the current policy (see (5.5)), we introduce the learnable parameter θ_{y_N} in order to learn the expected value $\mathbb{E}_{\eta_{\pi_\theta}}[y_1(N_\mathrm{s})]$, where, again, η_{π_θ} is the state distribution induced by the policy π_θ and the weather disturbance d. To capture (5.25), the terminal cost is then defined as

$$V_{\theta,\mathrm{f}}(y_1(N)) = -\theta_{\delta,y_1} \left(\theta_{y_N} - y_1(N)\right). \tag{5.26}$$

The full parametrisation is then

$$\theta = \begin{bmatrix} \theta_0 & \theta_{\delta,y_1} & \theta_u^\top & \theta_\omega^\top & \theta_{y_1,\mathrm{f}} & \theta_{y_N} & \theta_p^\top \end{bmatrix}^\top. \tag{5.27}$$

The allowable range for the parameters θ can be bounded during learning, incorporating prior knowledge on viable ranges, or enforcing realistic values on parameters that have physical significance. Table 5.3 gives a summary of the parametrisation, while the bounds and initialisation of the parameters in our experiments is given in Section 5.5.

As outlined in Section 5.2.2, the MPC scheme (5.22) for the value function approximation $V_\theta(s)$ satisfies the Bellman equalities [75], such that the action-value function and policy are delivered by the same scheme as

$$\begin{aligned} Q_\theta(s,a) = \min_{\mathbf{Y},\mathbf{X},\mathbf{U},\boldsymbol{\Sigma}} \quad & (5.22a) \\ \text{s.t.} \quad & (5.22b) - (5.22h), \\ & u(0) = a, \end{aligned} \tag{5.28}$$

$$\pi_\theta(s) = \arg\min_u \quad Q_\theta(s,u). \tag{5.29}$$

Instead of (5.29), the equivalent form

$$\begin{aligned} \pi_\theta(s) = \arg\min_{u(0)} \quad & (5.22a) \\ \text{s.t.} \quad & (5.22b) - (5.22h), \end{aligned} \tag{5.30}$$

Table 5.3.: Summary of MPC parametrisation in (5.22)

Symbol	Scope	Space
θ_0	cost - offset	$\mathbb{R}$
θ_{δ,y_1}	cost - weight reward	$\mathbb{R}$
θ_u	cost - control penalty	$\mathbb{R}^3$
θ_ω	cost - violations penalty	$\mathbb{R}^4$
$\theta_{y_1,\mathrm{f}}$	cost - terminal cost weight	$\mathbb{R}$
θ_{y_N}	cost - terminal weight estimate	$\mathbb{R}$
θ_p	model parameters	$\mathbb{R}^{28}$

is used in practice such that both $V_\theta(s)$ and $\pi_\theta(s)$ are found by solving the one optimisation problem (5.22).

5.4.3. Second-order LSTD Q-learning

We apply a second-order least-squares temporal difference (LSTD) Q-learning algorithm [114] to learn the parametrisation θ of (5.22), such that the policy π_θ optimises the greenhouse climate control RL task, as defined in Section 5.4.1. Q-learning attempts to learn a parametrisation θ such that the action-value function Q_θ fits the observed data. The policy is then inferred from the action-value function as $\pi_\theta(s) = \arg\min_a Q_\theta(s,a)$. In this chapter the Q-learning algorithm is applied online, i.e., training data is generated via interaction with the real system, as the policy is trained. Reformulating the fitting of Q_θ as a least-squares problem, in combination with a second-order Newton's method and an experience replay buffer of the past observed transitions [122], provides faster convergence and better sample efficiency with respect to traditional first-order methods. This approach has been effectively applied in the context of MPC [7, 60].

As in [7], we impose lower and upper bounds on the parameters with the following constrained optimisation problem

$$\begin{aligned} \Delta\theta^* = \arg\min_{\Delta\theta} \quad & \frac{1}{2}\Delta\theta^\top \bar{H}\Delta\theta + \alpha\bar{g}^\top\Delta\theta \\ \text{s.t.} \quad & \theta_{\mathrm{lb}} \le \theta + \Delta\theta \le \theta_{\mathrm{ub}}, \\ & \Delta\theta_{\mathrm{lb}} \le \Delta\theta \le \Delta\theta_{\mathrm{ub}}, \end{aligned} \tag{5.31}$$

with $\alpha > 0$ the learning rate, and $\bar{g}$ and $\bar{H}$ the gradient and Hessian of the Q-learning fitting problem averaged over m samples drawn from the replay buffer, i.e.,

$$\bar{g} = -\sum_{i=1}^{m} \delta_i \nabla_\theta Q_\theta(s_i,a_i), \tag{5.32}$$

$$\bar{H} = \sum_{i=1}^{m} \nabla_\theta Q_\theta(s_i,a_i)\nabla_\theta Q_\theta^\top(s_i,a_i) - \delta_i \nabla_\theta^2 Q_\theta(s_i,a_i). \tag{5.33}$$

Note that the sensitivities $\nabla_\theta Q_\theta$ and $\nabla^2_\theta Q_\theta$ are available automatically upon solving (5.28); see [4, 60] for details. This formulation additionally allows to limit the rate of change of each parameter with $\Delta\theta_{\text{lb}}$ and $\Delta\theta_{\text{ub}}$. Finally, the parametrisation θ is then updated as $\theta \leftarrow \theta + \Delta\theta^\star$.

Note that during training of the policy two optimisation problems are solved at each time step: (5.22) for V_θ and π_θ, and (5.28) for Q_θ, in order to be able to compute and store in buffer the TD error and sensitivities necessary for an update. Furthermore, the quadratic program (5.31) must be solved only once per parameter update. In contrast, when the policy is not being trained, only (5.22) must be solved for π_θ.

Remark 5.2. *In general, problem* (5.4) *is nonlinear and nonconvex. This implies that the RL algorithm is likely to converge to local suboptimal solutions. To alleviate this issue, exploratory behaviour can be injected into the learning policy in an effort to escape such local minima and to converge to a better (possibly global) solution. One method is to perturb the gradient of the objective of Q_θ in an, e.g., ε-greedy fashion [7]. In the current chapter, exploration is instead induced during the learning process by the weather forecast disturbances which, as explained in more detail in Section 5.5.1, are stochastically generated at the beginning of each training episode. As later shown in Section 5.5.3, this is enough to provide satisfactory convergence of the TD error.*

5.5. Numerical experiments

In this section the methodology proposed in Section 5.4 is demonstrated in simulation, with the resulting performance compared against existing state-of-art approaches. In the following, all MPC simulations are run on a Linux machine using one AMD EPYC 7252 core, 1.38GHz clock speed, and 251Gb of RAM. All simulations for training DNN-based RL methods, i.e., DDPG, are run on the same Linux machine using four NVIDIA RTX 2090 GPUs. Python source code and simulation results can be found at `https://github.com/SamuelMallick/mpcrl-greenhouse`. All optimisation problems are solved using the CasADi framework [11] and the IPOPT solver [212].

Assume that the values of all model parameters p are unknown. What is known is an uncertainty range $[p - 0.2|p|,\ p + 0.2|p|]$ that contains the true values, where $|\cdot|$ is the element-wise absolute value, and defines a uniform distribution spanning the uncertainty range as

$$\mathscr{R} = \mathscr{U}\left(p - 0.2|p|,\ p + 0.2|p|\right). \tag{5.34}$$

The initial values for the learnable model parameters are then a random sample from $\mathscr{R}$, and they are bounded to an enlargement of the uncertainty range. To initialise the cost parameters, the values from the RL stage cost are used. Further, we bound all cost terms to be non-negative, such that the notion of rewarded and penalised behaviour does not invert. The initial values and bounds for the full MPC parametrisation is given in Table 5.4. The RL stage cost coefficients used in our experiments are given in Table 5.5.

5.5.1. Weather disturbances

The weather disturbance profile d used in the experiments is real weather data presented in [105], collected from "the Venlow Energy greenhouse", located in Bleiswijk, The

Table 5.4.: Initial values and bounds on parametrisation in (5.22)

Symbol	Initial value	Bounds
θ_0	0	$(-\infty, \infty)$
$\theta_{\delta y_1}$	100	$[0, \infty)$
θ_u	$\begin{bmatrix}10 & 1 & 1\end{bmatrix}^\top$	$[0, \infty)$
θ_ω	$\begin{bmatrix}10^5 & 10^5 & 10^5 & 10^5\end{bmatrix}^\top$	$[0, \infty)$
$\theta_{y_1,\mathrm{f}}$	1	$[0, \infty)$
θ_{y_N}	135	$[0, \infty)$
θ_p	$\sim \mathscr{R}$	$[0.5\lvert p\rvert, 1.5\lvert p\rvert]$

5

Table 5.5.: Coefficients in the RL stage cost.

Symbol	Value
$c_{\delta y_1}$	100
c_u	$\begin{bmatrix}10 & 1 & 1\end{bmatrix}^\top$
ω	10^5

Netherlands. The data covers a full growth cycle of 40 days and, originally sampled at 5 minute intervals, has been resampled using an FIR anti-aliasing low pass filter at the sampling time used in this chapter, i.e., 15 minutes [27].

As RL approaches train on repeated growth cycle episodes, to avoid over-fitting of the learnt policy to a specific instance of weather data and to foster generalisation over a wider range of weather disturbances, a stochastic process is added to the real weather data in order to generate a new perturbed profile for each episodic growth cycle. In particular, Brownian noise, a time-correlated stochastic process, is added to the original weather profile. The Brownian noise is generated as the cumulative sum of white noise as

$$B(k) = \sum_{\tau=0}^{k} W(\tau), \quad W(\tau) \sim \mathscr{U}(-\rho, \rho), \tag{5.35}$$

where $\rho = \begin{bmatrix}0.01 & 0.005 & 0.01 & 0.005\end{bmatrix}^\top$, and $B(k)$ is the noise value at time step k. For the outdoor CO_2 level (d_2) and the outdoor humidity (d_4), this noise is added directly to the weather data. For the radiation (d_1) and the outdoor temperature (d_3), special care is required to ensure the perturbed weather profiles still follow a reasonable day/night cycle. To achieve this, Brownian excursions [208], stochastic processes that have constrained initial and final values, are used such that the Brownian noise only takes effect during the day cycle, and arrives at the relative night-time values at the start of the night cycle. Figure 5.1 demonstrates the original, and five perturbed, weather disturbance profiles for the last two days of the growth cycle.

5.5.2. Comparison approaches

We compare our approach, referred to as MPC-RL, against the robust-sample based MPC controller from [27], and a model-free RL controller trained with DDPG [147,

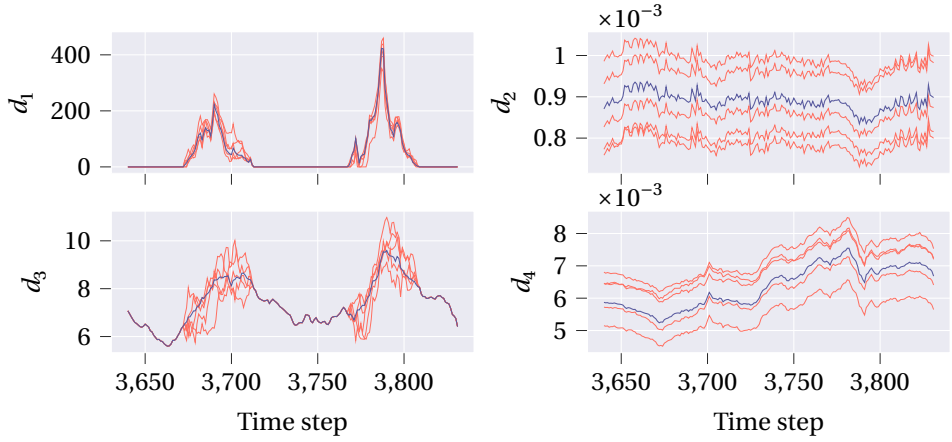

Figure 5.1.: Last two days of weather profiles. The purple profile is the original, and the orange lines are five example perturbed profiles.

215]. Furthermore, we include two nominal MPC controllers for comparison. One is ideal, with perfect (but unrealistic) model knowledge, while the other is a nominal MPC controller using an incorrect prediction model. All MPC controllers in the following use a horizon of $N = 24$. This value is selected to balance performance and computation time [27, 147]. In particular, the comparison approaches are:

- **Ideal MPC controller (I-MPC)**: The ideal MPC controller is a standard MPC controller with knowledge of the true dynamics via the real values of p.
- **Nominal MPC controller (N-MPC)**: The nominal MPC controller is a standard MPC controller using an incorrect prediction model with parameters $\hat{p} \sim \mathscr{R}$.
- **Robust MPC controller (R-MPC-n) [27]**: The robust sample-based MPC controller is essentially an implementation of a stochastic MPC controller based on the scenario approach [172]. The parametric model uncertainty is addressed by sampling the probability distribution of model parameters, in this case the uniform distribution $\mathscr{R}$, and optimising one state trajectory for each sample in the MPC optimisation. In [27], 20 samples were used. In our experiments we test a range of numbers of samples n, indicating the number of samples in the shorthand name, e.g., 5 samples is denoted R-MPC-5.
- **DDPG RL controller (DDPG) [147, 215]**: DDPG is a model-free off-policy RL algorithm that leverages DNNs as function approximators. Conversely to the proposed Q-learning algorithm, this method belongs to the policy gradient family, which optimises the parametrisation via estimates of the gradient of the policy w.r.t. θ [120]. The learning hyper-parameters are taken from [147] and are reported in Table 5.6.

Note that both learning-based and non-learning-based controllers described above are deployed in simulations on the same greenhouse environment. Obviously, this

environment employs the correct model to simulate the evolution of the real dynamics and to generate training data.

5.5.3. Results

Table 5.6.: Hyper-parameters for the DDPG RL Controller

Parameter	Value
learning rate	10^{-5}
gradient threshold	1
L_2 regularisation factor	10^{-5}
experience buffer size	10^4
experience mini-batch size	64

We run 100 growth cycles to learn the parametrisation of the MPC scheme (5.22), with the initial parametrisation in Table 5.4. We simulate online learning, where the data generated while interacting with the system is used to learn the policy. The learning is concluded after 100 episodes as, for this case study, this is sufficient to observe satisfactory convergence of the TD error and the learnt MPC parametrisation, alongside improvements to both the economic profit and constraint satisfaction. The hyper-parameters of the learning process are tuned as follows. The discount factor is selected as $\gamma = 0.99$, while the learning rate is $\alpha = 0.1$. The parameters are updated once at the end of each growth cycle, with the maximum update being constrained to 5% of their current values, i.e., $\Delta\theta_{\text{lb}} = -0.05|\theta|$ and $\Delta\theta_{\text{ub}} = 0.05|\theta|$. A replay buffer stores the observations from the last three cycles, and every update considers two cycles worth of observations, with all observations from the most recent cycle guaranteed to be used. Three different parametrisations are learnt for both RL-based approaches, where, for each, the random seed for generating the weather profiles is different. The initial state for each cycle is $x(0) = \begin{bmatrix}0.0035 & 0.001 & 15 & 0.008\end{bmatrix}^T$.

Figure 5.2 shows the episode-wise stage cost $L_{\text{ep}} = \sum_{k=0}^{N_{\text{s}}} L(s(k), u(k))$, the TD error $\delta_{\text{ep}} = 1/N_{\text{s}} \sum_{k=0}^{N_{\text{s}}} \delta(k)$, and the constraint violations Ψ for each growth cycle during training. It can be seen that the incurred stage cost and TD error are significantly reduced during training. This is further represented in the constraint violations, which, starting from a large value, approach zero. Figure 5.3 shows the constrained outputs during the first and last growth cycles of training. It can be seen that in the first growth cycle, the indoor CO_2 upper bound (y_2) is violated several times towards the end of the cycle, while the indoor humidity upper bound (y_4) is violated for practically the entire cycle. After 100 cycles of learning, it can be seen that the indoor CO_2 level never violates the bounds, and the indoor humidity level exceeds the upper bound only a few times, quickly dropping back to allowable levels. Figure 5.4 shows the parameter evolution over the training for a subset of the learnable parameters; the four parameters whose values changed the most. Finally, in Section 5.A.1 the control inputs during the first and last growth cycles of training are included for completeness.

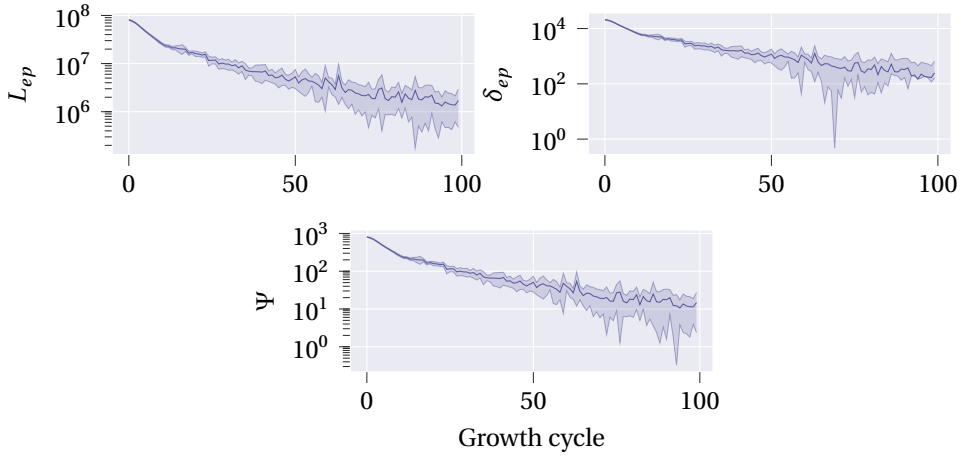

Figure 5.2.: Stage cost, TD error, and constraint violations over 100 growth cycles of training

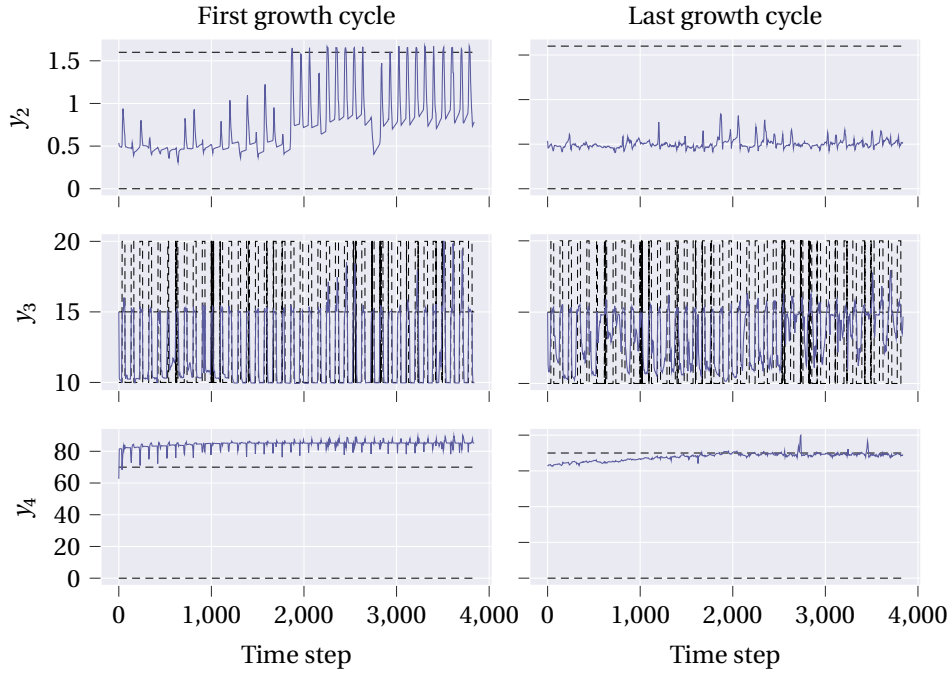

Figure 5.3.: Outputs for the last 10 days of the first and last growth cycle of training with upper and lower bounds (black dashed lines)

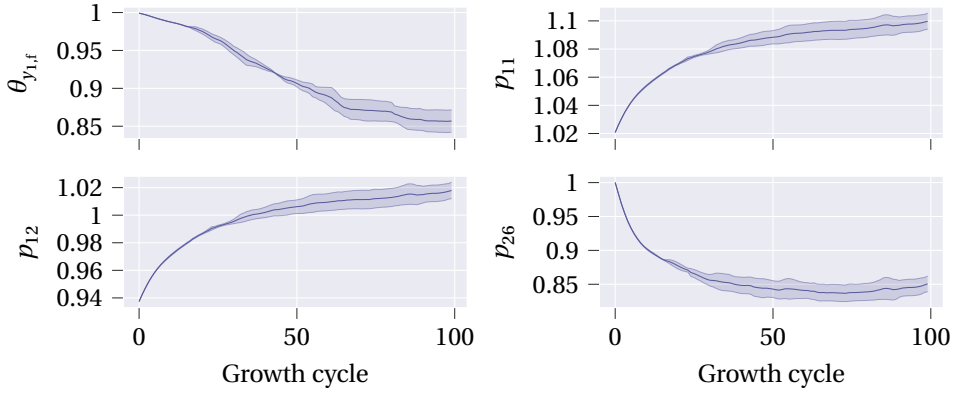

Figure 5.4.: Parameter evolution, for the 4 most changed parameters, over 100 growth cycles of training. Note that model parameters θ_p have been normalised with respect to their true values.

In an evaluation phase, i.e., with the policy fixed, we compare our final trained RL-based MPC controller against the approaches outlined in Section 5.5.2. Each approach is evaluated over 100 growth cycles, with Figure 5.5 demonstrating the results with respect to the performance indicators given in Section 5.3. The episode-wise stage cost is included for interest, especially for comparison between the two RL-based approaches, that during training learn policies to minimise this cost, as in (5.4). For completeness, the state, output, and input trajectories for the final 10 days of an evaluation growth cycle are included in Section 5.A.1.

The constraint violations Ψ show that our approach has reduced the constraint violations the most, approaching the level of the ideal MPC controller. The robust MPC controller does improve the constraint violations over the nominal MPC; however, they remain at an unsatisfactory level. Finally, the DDPG-based controller improves the violations over both the nominal and robust MPC controllers, but does not reach that of our approach.

The crop yield $y_1(N_s)$ shows that the controllers with very high constraint violations, i.e., the nominal MPC controller and robust MPC controller with 5 samples, generate a very high crop yield. This is in line with observations in the literature [229], and is due to the unrealistic growth that occurs in the model when variables such as CO_2 levels and humidity are dangerously high. Notably, for higher numbers of samples, the robust MPC controllers significantly reduce the crop yield due to conservatism. Our approach has reduced the drop in crop yield, even while vastly improving the constraint violations. The DDPG controller gives superior crop yield even with respect to the ideal MPC controller. This is again in line with observations in the literature [147], and is due to the DDPG approach's tendency to optimise the crop yield over efficiency and constraint violations.

The economic profit indicator P shows similar trends to the crop yield for all MPC-based controllers. Again, we highlight that the high economic profit of the nominal

MPC controller, and the robust MPC controller with 5 samples, is due to the unrealistic growth occurring in the model when the output constraints are significantly violated. A clear difference from the crop yield trend is for the DDPG controller, which now under-performs in comparison to both the ideal MPC controller and our approach. This demonstrates how the DDPG controller maximises the yield in an inefficient way with respect to the control inputs, resulting in an overall reduced profit.

Finally, with respect to the stage cost L_{ep}, our approach outperforms the DDPG controller. This is notable as, during training, both RL-based approaches attempt to solve an optimisation problem that minimises L_{ep}, i.e., the performance (5.3).

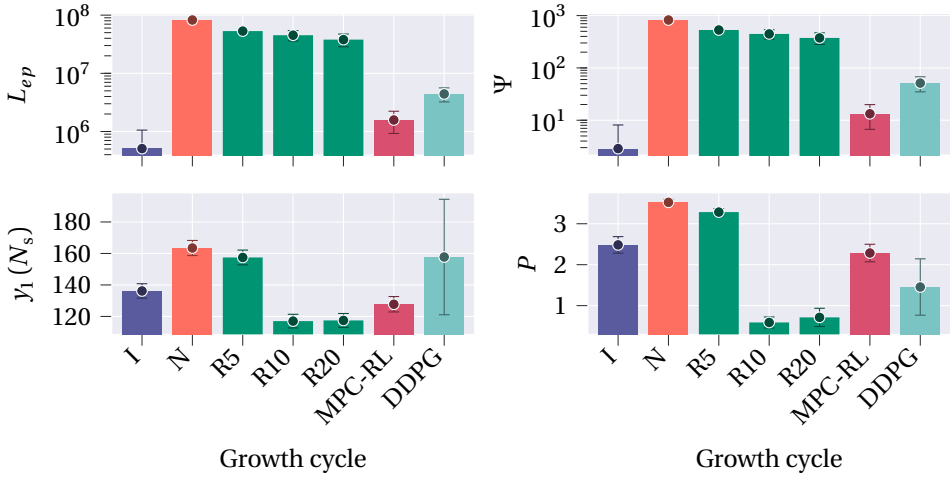

Figure 5.5.: Comparison of all approaches over 100 growth cycles. The learning-based strategies are tested after convergence. Bars show the mean value, with error bars showing a standard deviation.

Table 5.7 shows the average computation time needed per time step for each of the approaches. Naturally, the DDPG approach is the fastest as no optimisation problem is solved when computing its actions. Comparing the MPC-based approaches, our approach requires similar computation times to those of the I-MPC and N-MPC approaches during evaluation, and is only slightly slower during training. In contrast, the R-MPC approaches introduce additional optimisation variables with each additional sample, and the required computation time increases as the samples increase.

Table 5.7.: Average computation time required by each approach

Approach	I	N	R5	R10	R20	MPC-RL (train.)	MPC-RL (eval.)	DDPG
Time (s)	0.0378	0.0390	0.229	0.570	1.234	0.0793	0.0430	0.000155

5.6. Conclusions

In this chapter, we propose an RL-based MPC controller for greenhouse climate control. At the core of the approach is a parametrised MPC scheme that can serve as policy provider and value-function approximator for an RL task. The greenhouse climate control problem has been formulated as an RL task, such that the MPC scheme can learn a parametrisation online using closed-loop data. Second-order Q-learning has been proposed as the RL algorithm for learning the parametrisation. In simulations, the proposed approach has been shown to learn an MPC scheme that significantly reduces the violations of output constraints in closed-loop operation. The final learnt controller has then been compared against state-of-the-art MPC- and RL-based controllers from the literature, showing the best performance in terms of constraint violations and efficient crop growth.

Future work directions include the application of this methodology to alternative greenhouse models, and experimental validation in real-world tests. Additionally, alternative learning algorithms, such as policy gradient approaches, could be explored to learn the parametrisation of the MPC scheme.

5.A. Appendix

5.A.1. Extra input, output, and state trajectories

Figure 5.6 shows the control inputs of our approach during the first and last episodes of training. The trajectories are clearly different, as the approach has learnt a policy that optimises for closed-loop performance.

Figures 5.7 to 5.9 show the output, state, and input trajectories, respectively, for the final 10 days of a growth cycle during evaluation of the compared controllers. Notably, the additional constraint violations in the outputs, introduced by the comparison controllers, are evident. Further, the control inputs demonstrate the inefficiency of the DDPG controller's actuation.

5.A.2. Greenhouse model

The continuous-time lettuce growing greenhouse model is described by the following set of equations [204]

$$
\begin{aligned}
\dot{x}_1(t) &= p_1 \phi_{\text{phot,c}}(t) - p_2 x_1(t) 2^{x_3(t)/10-5/2}, \\
\dot{x}_2(t) &= \frac{1}{p_9}\left(-\phi_{\text{phot,c}}(t) + p_{10} x_1(t) 2^{x_3(t)/10-5/2} + 10^{-6} u_1(t) - \phi_{\text{vent,c}}(t)\right), \\
\dot{x}_3(t) &= \frac{1}{p_{16}}\left(u_3(t) - \left(10^{-3} p_{17} u_2(t) + p_{18}\right)\left(x_3(t) - d_3(t)\right) + p_{19} d_1(t)\right), \\
\dot{x}_4(t) &= \frac{1}{p_{20}}\left(\phi_{\text{transp,h}}(t) - \phi_{\text{vent,h}}(t)\right), \\
y_1(t) &= 10^3 x_1(t), \\
y_2(t) &= 10^3 x_2(t) \frac{p_{12}\left(x_3(t) + p_{13}\right)}{p_{14} p_{15}},
\end{aligned}
$$

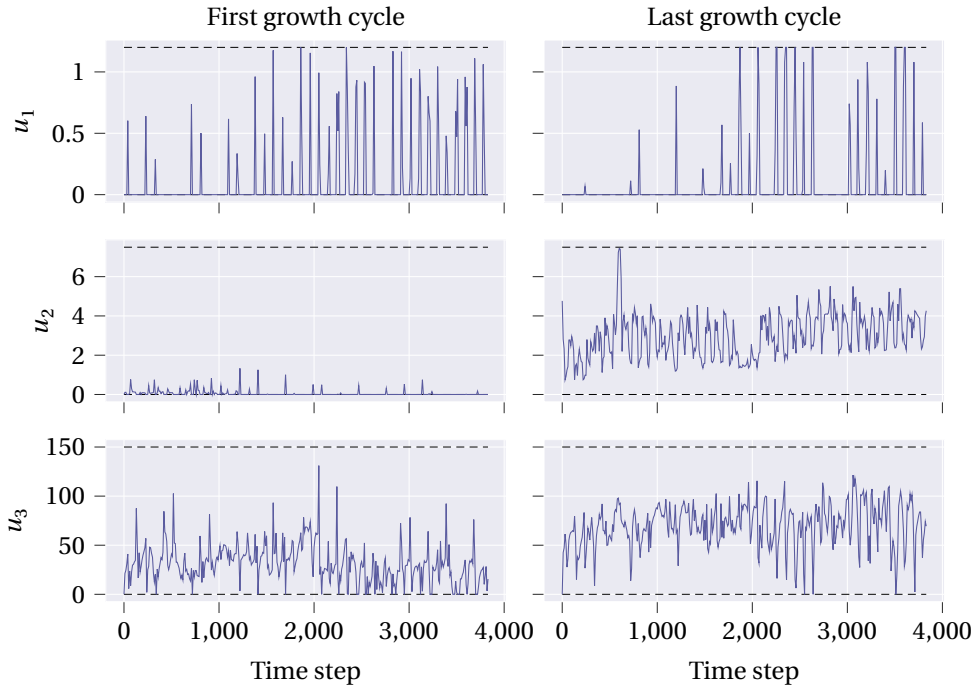

Figure 5.6.: Control inputs for the first and last growth cycle of training with upper and lower bounds (black dashed lines)

$$y_3(t) = x_3(t),$$
$$y_4(t) = \frac{10^2}{11} x_4(t) \frac{p_{12}\left(x_3(t) + p_{13}\right)}{\exp\left(\frac{p_{27} x_3(t)}{x_3(t) + p_{28}}\right)}.$$

where $t \geq 0$ is continuous time. Further, define the following functions:

$$\phi_{\text{phot,c}}(t) = \frac{1}{\varphi(t)} \left(1 - \exp\left(-p_3 x_1(t)\right)\right) \left(p_4 d_1(t) \left(-p_5 x_3(t)^2 + p_6 x_3(t) - p_7\right) \left(x_2(t) - p_8\right)\right),$$
$$\varphi(t) = p_4 d_1(t) + \left(-p_5 x_3(t)^2 + p_6 x_3(t) - p_7\right) \left(x_2(t) - p_8\right),$$
$$\phi_{\text{vent,c}}(t) = \left(u_2(t) 10^{-3} + p_{11}\right) \left(x_2(t) - d_2(t)\right),$$
$$\phi_{\text{transp,h}}(t) = p_{21} \left(1 - \exp\left(-p_3 x_1(t)\right)\right) \left(\frac{p_{22}}{p_{23}(x_3(t) + p_{24})} \exp\left(\frac{p_{25} x_3(t)}{x_3(t) + p_{26}}\right) - x_4(t)\right),$$
$$\phi_{\text{vent,h}}(t) = \left(u_2(t) 10^{-3} + p_{11}\right) \left(x_4(t) - d_4(t)\right),$$

where $\phi_{\text{phot,c}}(t), \phi_{\text{vent,c}}(t), \phi_{\text{transp,h}}(t)$ and $\phi_{\text{vent,h}}(t)$ are the gross canopy photosynthesis rate, mass exchange of CO_2 through the vents, canopy transpiration and mass exchange of H_2O through the vents, respectively.

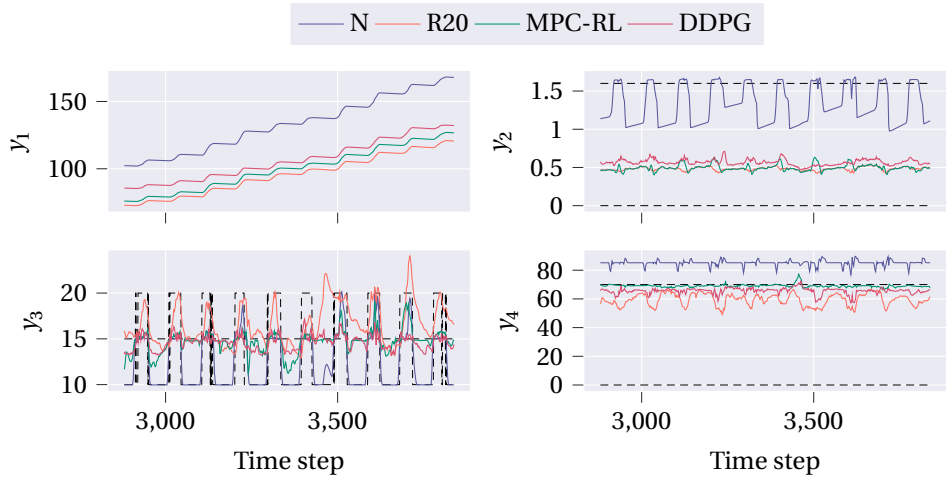

Figure 5.7.: Output trajectories for the last 10 days of a growth cycle during evaluation, with upper and lower bounds (black dashed lines)

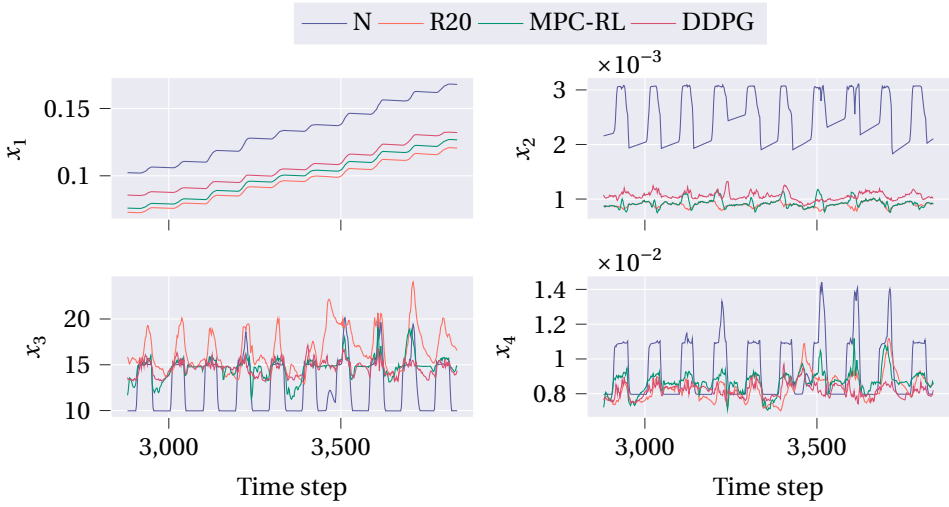

Figure 5.8.: State trajectories for the last 10 days of a growth cycle during evaluation

To generate the discrete-time prediction mode (5.1), discretisation is performed using the explicit fourth order Runge-Kutta method with a time step of 15 minutes, as in [27]. Finally, the values for the model parameters p are given in Table 5.8.

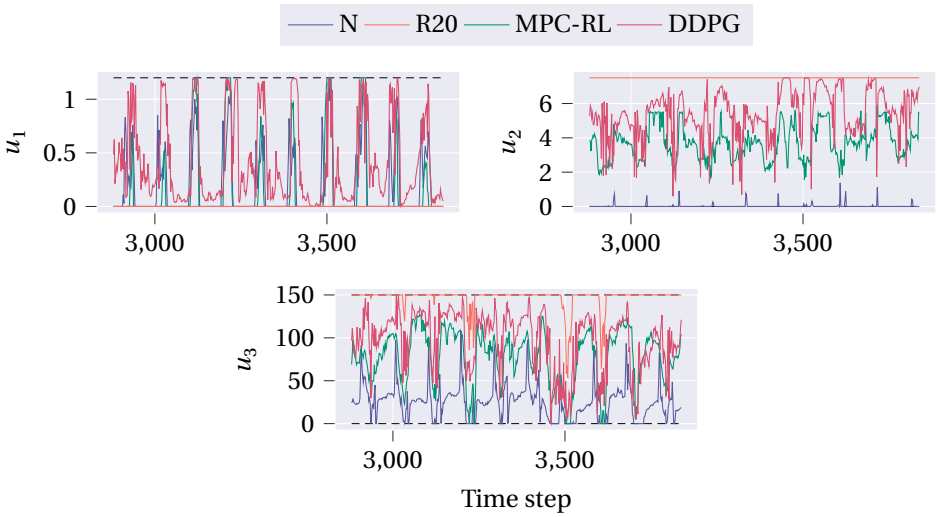

Figure 5.9.: Input trajectories for the last 10 days of a growth cycle during evaluation, with upper and lower bounds (black dashed lines)

Table 5.8.: Model parameters

Symbol	Value	Unit	Explanation from [204, 205]
p_1	0.544	-	Yield factor; aggregation of c_α (CO_2-glucose stoichiometric conversion factor) and c_β (yield of carbohydrates conversion to structural material)
p_2	$2.65 \cdot 10^{-7}$	s^{-1}	Aggregation of $c_{\text{resp,s}}$ and $c_{\text{resp,r}}$ (maintenance respiration rates for shoot and root at 25 °C) weighted by c_β and c_τ (ratio of root dry weight to total crop dry weight)
p_3	53	$m^2\,kg^{-1}$	Effective canopy surface; aggregation of $c_{\text{lar,d}}$ (shoot leaf area ratio) with c_{k} (canopy extinction coefficient) and c_τ
p_4	$3.55 \cdot 10^{-9}$	$kg\,J^{-1}$	Aggregation of c_{par} (photosynthetically activate radiation to total solar radiation ratio) and $c_{\text{rad,rf}}$ (roof solar radiation transmission coefficient)
p_5	$5.11 \cdot 10^{-6}$	$m\,s^{-1}\,°C^{-2}$	Second-order term in polynomial approximation of temperature effect on CO_2 leaf diffusion
p_6	$2.3 \cdot 10^{-4}$	$m\,s^{-1}\,°C^{-1}$	First-order term in polynomial approximation of temperature effect on CO_2 leaf diffusion
p_7	$6.29 \cdot 10^{-4}$	$m\,s^{-1}$	Zeroth-order term in polynomial approximation of temperature effect on CO_2 leaf diffusion
p_8	$5.2 \cdot 10^{-5}$	$kg\,m^{-3}$	CO_2 compensation point at 25 °C
p_9	4.1	m	Volumetric capacity of greenhouse air for CO_2
p_{10}	$4.87 \cdot 10^{-7}$	s^{-1}	Aggregation of $c_{\text{resp,s}}$ and $c_{\text{resp,r}}$ weighted by c_α and c_τ
p_{11}	$7.5 \cdot 10^{-6}$	$m\,s^{-1}$	Leakage air exchange through greenhouse cover
p_{12}	8.314	$J\,K^{-1}\,mol^{-1}$	Gas constant
p_{13}	273.15	K	Absolute temperature
p_{14}	101325	Pa	Sea level standard atmospheric pressure
p_{15}	0.044	$kg\,mol^{-1}$	CO_2 molar mass
p_{16}	$3 \cdot 10^4$	$J\,m^{-2}\,°C^{-1}$	Heat capacity of greenhouse air
p_{17}	1290	$J\,m^{-3}\,°C^{-1}$	Heat capacity per volume unit of greenhouse air
p_{18}	6.1	$W\,m^{-2}\,°C^{-1}$	Heat transfer coefficient through greenhouse cover
p_{19}	0.2	-	Sun heat load coefficient accounting for roof transmission, solar radiation interception by structural components, and conversion from absorbed solar energy to sensible heat load by canopy
p_{20}	4.1	m	Volumetric capacity of greenhouse air for humidity
p_{21}	0.0036	$m\,s^{-1}$	Canopy transpiration mass transfer coefficient
p_{22}	9348	$J\,kg\,m^{-3}\,kmol^{-1}$	Aggregation of $c_{\text{H}_2\text{O}}$ (water molar mass), $c_{\text{v,0}}$ (calibration parameter) and $c_{\text{v,1}}$ (saturation water vapor pressure parameter for canopy transpiration)
p_{23}	8314	$J\,K^{-1}\,kmol^{-1}$	Gas constant
p_{24}	273.15	K	Absolute temperature
p_{25}	17.4	-	Saturation water vapor pressure parameter for canopy transpiration
p_{26}	239	°C	Saturation water vapor pressure parameter for canopy transpiration
p_{27}	17.269	-	Saturation water vapor pressure parameter for humidity
p_{28}	238.3	°C	Saturation water vapor pressure parameter for humidity

6

Reinforcement learning with model predictive control for highway ramp metering

In the backdrop of an increasingly pressing need for effective urban and highway transportation systems, this work explores the synergy between model-based and learning-based strategies to enhance traffic flow management by use of an innovative approach to the problem of ramp metering control that embeds Reinforcement Learning (RL) techniques within the Model Predictive Control (MPC) framework. The control problem is formulated as an RL task by crafting a suitable stage cost function that is representative of the traffic conditions, variability in the control action, and violations of the constraint on the maximum number of vehicles in queue. An MPC-based RL approach, which leverages the MPC optimal problem as a function approximation for the RL algorithm, is proposed to learn to efficiently control an on-ramp and satisfy its constraints despite uncertainties in the system model and variable demands. Simulations are performed on a benchmark small-scale highway network to compare the proposed methodology against other state-of-the-art control approaches. Results show that, starting from an MPC controller that has an imprecise model and is poorly tuned, the proposed methodology is able to effectively learn to improve the control policy such that congestion in the network is reduced and constraints are satisfied, yielding an improved performance that is superior to the other controllers.

This chapter is based on [7].

6.1. Introduction

Over the past decades, the need for urban and highway transportation has become significantly relevant to our society. Extended travel times (primarily spent in traffic jams), impact on stress levels and pollution are but a few examples of the negative effects due to high demands for mobility that traffic engineers are facing nowadays [98, 153]. In particular, the necessity for advanced intelligent traffic control strategies is arising as the construction of new infrastructure and upgrades to the existing one become increasingly unsustainable, both in economical, environmental, and societal terms [44, 182].

Ramp metering (RM) control has been proven to be an effective tool in regulating the traffic flow entering the network at on-ramps [158]. It consists in modulating the flow at an on-ramp via traffic lights with appropriate timings of red, green, and amber lights. Earlier approaches were fixed-time, i.e., the timings were calculated offline with the help of historic traffic data [222]. More efficient strategies involve real-time measurements of the traffic conditions, enabling online computation of the timings in a traffic-responsive way. In these cases, the underlying algorithms range from simpler feedback strategies [68, 157, 217] to more complex architectures, such as Model Predictive Control (MPC) [88–90] and Reinforcement Learning (RL) [86, 214]. One of the primary research challenges in RM today is the design of strategies that can control the entering flow in a way that effectively balances congestion in the freeway and the queue length at the on-ramp, two often conflicting goals [176, 209]. Most strategies deployed on actual ramps are either feedback-based local policies, which are inherently myopic, or model-based policies and, as such, are vulnerable to model inaccuracies and thus require precise identification of traffic parameters, as well as an attentive assessment of their efficacy [129]. Conversely, data-driven approaches, which are one of the main focuses of current research, typically necessitate extensive datasets of real or synthetic traffic data for training.

In modern control literature, MPC is an established control paradigm whose appeal can be attributed to its strong theoretical foundations, a diverse array of applications, as well as its remarkable capability to manage multivariate and constrained systems [163]. Its versatility further allows for alternative formulations that can systematically handle disturbances affecting the system, e.g., in a robust [17] or a stochastic [140] manner. Unlike other black-box or purely data-driven approaches, MPC-based control strategies often maintain a high degree of explainability and interpretability, which is especially favourable in settings where constraint satisfaction is critical. These properties are in part thanks to the prediction model that the MPC scheme contains, which allows it to forecast the dynamical evolution of state trajectories along the time horizon, and which is often designed by domain experts.

Nonetheless, it is well known that the performance of such predictive controllers is heavily bound to the quality of the prediction model, which calls for a high level of expertise during the system identification phase. In an attempt to counter this limitation, driven by the recent surge in advancements in Machine Learning (ML) algorithms and facilitated by the growing computational and sensing capabilities of digital systems, there is currently a substantial body of research dedicated to learning-based control methodologies that offer ways of leveraging open- or closed-loop data to, e.g., synthesise

models and controllers, or enhance existing ones.

Amongst these methodologies, one of the most notable approaches is Reinforcement Learning (RL), a paradigm of ML that has gained substantial attention due to its advancements and successes. It provides a framework that allows to train an agent to interact with an unknown environment and to make decisions to maximise rewards or minimise costs [197]. The popularity in the contemporary ML community tends to favour its model-free variants that rely solely on observed state transitions and stage cost realisations to increase the performance of the control policy. Early RL focused on tabular approaches that enjoy simplicity, some degree of interpretability, and convergence guarantees, but they are only applicable to small and simple environments [220], and are often limited in handling larger state spaces, exhibit slow convergence, and struggle to generalise. As the field progressed, researchers recognised the potential of function approximators and, in particular, of Neural Networks (NNs) [37], leading to the development of Deep RL (DRL). However, while DRL has demonstrated great efficacy [143], its NN-based policies generally lack interpretability, and the integration of prior knowledge (e.g., from domain experts) is still difficult. Thus, providing formal guarantees on properties relevant to a controller, such as stability and constraint satisfaction, in the context of NN-enhanced control architectures is one of today's challenges in the field. Additionally, existing model-free approaches are usually plagued by sample inefficiency and poor real-world performance, since they are often trained offline in approximate simulators and require large amounts of data for meaningful converge [57].

At the same time, on the other side of the coin, the control community is also dedicating efforts in integrating data-driven techniques with control systems in more structured ways, fostering methodologies that aim to learn various components of a controller directly from data [32, 92]. The goals include, e.g., reducing model uncertainties and the impact of disturbances, improving closed-loop performance and battling conservativeness, or promoting robust control policies with guarantees on constraint satisfaction. Notably, learning-based MPC methodologies [141] have been devised to, e.g., leverage Gaussian Process regression to learn unmodelled nonlinear dynamics in the prediction model via state transition observations [91], or use Bayesian Optimisation to automatically tune and optimise the controller hyperparameters [161]. In the literature, MPC has been successfully combined also with RL, e.g., as a safety filter for the learning-based agents (also called *shielding*) [92, 210], as a model reference and baseline control provider [168, 236], and to solve mixed-logical control problems more efficiently [52].

Recently, [75] proposed and justified the use of MPC as function approximation of the optimal policy in model-based RL. In such a scheme, the MPC controller's optimisation problem acts both as policy provider and value function approximation of the RL task. Concurrently, the learning algorithm (such as Q-learning and stochastic/deterministic policy gradient) is tasked with adjusting the parametrisation of the controller in an effort to directly or indirectly discover the optimal policy, thus improving closed-loop performance in a data-driven fashion. In this way, despite the presence of mismatches between the prediction model and the real system, the MPC control scheme is able to learn and deliver at convergence the optimal policy and value functions of the underlying RL task, granted the MPC parametrisation is rich enough. Unlike the more traditional pipeline, where a system identification phase (which selects model parameters that

match the system's behaviour with observed data by, e.g., minimising the predicted mean-squared-error) is followed by the controller design (possibly, iteratively), this approach directly tunes the parametrisation to maximise performance, irrespective of the controller's predictive accuracy. Such a rationale is often called Identification for Control, and advocates that the best parametrisation for control shall not yield the best predictions, but rather the best performance on the true system [75, 161, 233]. In contrast to NN-based RL, the key advantages of the approach adopted in this chapter are multiple:

- The MPC scheme at its core allows to easily integrate prior information on the system in the form of an expert-based, possibly imperfect, prediction model.
- MPC-based agents are in general more amenable to analysis and certification in terms of stability and constraint satisfaction, a benefit supported by the extensive body of literature on MPC [163]. Although challenging and beyond the scope of the proposed approach, it is possible to ensure that theoretical properties, such as stability, recursive feasibility and constraint guarantees, are preserved during the RL learning process [76, 233]. On the other hand, NNs are notoriously black-box, lack interpretability, and often provide no guarantees prior to convergence.
- MPC controllers can also take constraints on states and actions into account in an explicit and structured way, which NNs are generally incapable of or can do poorly.

Compared to traditional non-learning MPC formulations, the proposed approach enjoys the following benefits:

- The need for extensive open-loop data collection and system identification phase is eliminated, as the MPC parametrisation is automatically adjusted based on closed-loop data to improve performance, rather than prediction accuracy. This bypasses the issue of model inaccuracies and endows the scheme with online adaptivity capabilities.
- By appropriately penalising constraint violations, the RL algorithm can guide the learning process to produce an MPC policy that satisfies system constraints.

All combined, these advantages can foster a safer training where the agent may enjoy a more stable and constraint-abiding learning process, opening up in the future the possibility of training directly in the real world [57], and contributing to the suitability of this framework for the RM control problem. In particular, the proposed MPC-RL method automatically learns from data a policy that optimally balances congestion and queue lengths, without requiring further manual tuning. This bypasses the issue of model inaccuracies in traffic predictions, which are often the primary concern in non-myopic control strategies. It also embeds adaptivity in the RM strategy in the face of stochasticies (e.g., demand forecasts or the behaviour of drivers can vary significantly across different cities or countries, meaning that system identification performed in one region may not generalise well to another). Besides, we stress again that improving the accuracy of the prediction model may not necessarily result in a better closed-loop

performance. Hence, this framework allows for improvements to the control policy without focusing on nor needing an accurate prediction model or identification as a separate step in the design process. Additionally, the approach strengthens constraint satisfaction, which is crucial in real-world applications like RM control to avoid critical congested scenarios.

This chapter contributes to the state of the art by proposing a novel approach, which embeds the MPC framework within an RL algorithm, to tackle the highway RM control problem. Specifically:

1. To the best of the authors' knowledge, an MPC-based RL method inspired by [75] is proposed for the first time to solve the RM control problem. The method combines a differentiable parametrised MPC scheme, which acts as policy provider and function approximation, with an online second-order Q-learning algorithm. The Q-learning algorithm adjusts the MPC parametrisation via gradient updates based only on observed closed-loop data, allowing to automatically learn a congestion-free and constraint-abiding control policy. As explained earlier, this approach overcomes the issue of model inaccuracies and uncertainties in the MPC and in the context of RM.

2. The proposed RM strategy is then implemented and tested on a benchmark highway stretch, where it is trained in an online fashion to simultaneously avoid bottlenecks and excessive queue lengths at the on-ramp, incorporating penalties for constraint violations directly into the MPC objective and the RL reward function. Simulation results underscore the effectiveness of the MPC-RL approach. Notably, starting from a poorly tuned controller with an imprecise prediction model, the proposed approach demonstrates quick learning in reducing congestion and satisfying the queue constraint. These results are further validated with a comparison with other state-of-the-art traffic controllers, both model-based and learning-based. Our method empirically outperforms them in performance and sample-efficiency.

The chapter is organised as follows. Section 6.2 summarises relevant works in MPC, RL, and their combination in RM applications. Background on modelling and controlling traffic networks via MPC is provided in Section 6.3. Section 6.4 presents the proposed methodology, showing how MPC can be combined with RL, as well as explaining the MPC-based RL algorithm itself. The method is then applied to a numerical case study in Section 6.5. Finally, Section 6.6 concludes the chapter with some final remarks and future directions.

6.2. Related work

Several methods have been proposed to address the RM problem [182]. In this section, we report and discuss recent developments in model-based and model-free approaches, as well as the combination of the two, in comparison to our proposed methodology.

6.2.1. MPC approaches for ramp metering control

A well-studied approach to tackle RM is MPC. First formalised in [55], it was deployed for RM [14] and coordinated traffic control [88–90], and is still subject of research, e.g., [85, 201]. As explained in Section 6.1, its predictive capabilities make MPC highly desirable, but at the same time leave it susceptible to model mismatches and uncertainties. When these disturbances are assumed to be modelled, one can resort to robust or stochastic formulations to mitigate their impact, as proposed in, e.g., [123]; however, such assumptions are in general hard to satisfy in practice, and the additional computational burden can be taxing. For these reasons, MPC techniques for traffic control that can adapt to and circumvent model mismatches and uncertainties are still a vibrant research area.

6.2.2. Model-free approaches for ramp metering control

Perhaps the most established RM approach is ALINEA [157], a local feedback strategy that seeks to maximise freeway flow in a merging area by keeping the bottleneck density near critical value. Many extensions have been proposed to enhance its basic formulation, such as PI-ALINEA, which deals with distant downstream bottlenecks [217], and FF-ALINEA, which anticipates the evolution of a bottleneck density [68]. However, these closed-loop controllers are fixed since they have no means to adapt their gains to changing conditions, and thus suffer from, e.g., uncertainties in traffic parameters and poor tuning. Ways have been devised to automatically account for it, e.g., via Kalman filtering to estimate the traffic parameters [183] or based on historic data [45]. Unfortunately, these methods are still plagued with a fundamentally myopic control strategy, and either assume that parameters evolve according to a known observable model, or that a huge amount of offline recorded data is available and is representative of both current and future traffic scenarios. Moreover, these feedback methods, alongside MPC, often need to be paired with a mechanism for traffic state estimation, enabling them to respond to real-time traffic behaviour [218].

More recently, RL has gained popularity thanks to its ability to automatically learn an optimal policy solely by interacting with the environment, addressing the issue of uncertain and unknown dynamics. Earlier works adopted tabular Q-learning for the RM problem [54, 62, 94]. However, with function approximations becoming more and more predominant, especially NNs, DRL algorithms have emerged as a relevant alternative to model-based controllers. In [214], different policy gradient DRL algorithms are deployed in freeway control and compared. While results are promising, the issue of state constraints is not addressed, especially on the queue length, which should be contained to avoid spillback [182]. This is a fundamental issue in RL, since these algorithms can often be made aware of the presence of constraints only when violations occur and are appropriately penalised in the reward, whereas MPC can handle constraints explicitly in its formulation. In [86] it is proposed to use a combination of offline historic data and online synthetic data in an iterative way to avoid the RL model from being trapped in an inaccurate training environment. This work underscores the severe impact of high-quality data on the RL learning process, and raises the question of whether research efforts might be more effectively directed towards the development of online model-based learning algorithms that can learn directly from closed-loop interactions

with the target environment while addressing constraints more explicitly. DRL has also facilitated the advancement of multi-agent solutions [199], which are especially useful in large-scale highway control problems. However, in addition to the aforementioned issues, multi-agent RL must also address the nonstationarity of its learning targets.

6.2.3. Combination of MPC and RL for ramp metering

The use of learning-based MPC schemes for traffic management has been investigated in the recent literature. For instance, [2] shows how recurrent NNs can be used to predict traffic behaviour, and [79] proposes to adopt such a network as prediction model in their MPC scheme.

Nonetheless, to the best of the authors' knowledge, the combination of MPC and RL in traffic control has received scarce attention, with [166, 194] the only relevant works. Therein, an MPC-DRL architecture is proposed that hierarchically combines the two controllers: the high-level MPC controller, which runs at a lower frequency, provides initial optimality while explicitly incorporating constraints, whereas the low-level DRL agent operates at a higher frequency and aims to compensate for model mismatches in the MPC. The MPC and RL control actions are then linearly combined together and fed to the freeway environment as a unique control input.

Given that the final control signal is a summation of an optimisation-based one and a learning-based one, it is anticipated that the resulting policy may not always be able to prevent constraint violations, particularly early in the training process. However, a benefit of [194] compared to the proposed methodology is the higher control frequency at which the DRL policy is run, granting quicker control response. This is especially advantageous in applications with fast dynamics, although traffic systems are typically slow, with sampling times up to tens of seconds. Note that the slower control frequency of our method is not inherently limiting as, with the improving hardware capabilities and faster solvers in recent years, increasing this frequency is feasible. The two methodologies also differ in computational complexity. While our approach is more demanding during the online training phase due to the need to solve the MPC problem twice per time step, it becomes more efficient at deployment, as it requires only one MPC calculation per time step [75]. In contrast, [194] runs both the MPC and DRL policies at every step, leading to higher computational costs at inference. Moreover, note that if training is performed offline, e.g., via an off-policy algorithm, then the additional computational burden of our method becomes irrelevant to online performance.

6.2.4. Comparison with proposed methodology

Compared to feedback strategies such as ALINEA, the MPC-based RL framework [75] we propose to use for RM leverages as its core an MPC controller, which, thanks to its predictive capabilities, can yield non-myopic policies. At the same time, its RL module allows to automatically adjust the MPC controller, and thus, it circumvents some of the challenges that conventional MPC typically struggles to address, i.e., model misspecifications and poor identification and tuning. This framework also boasts a higher degree of interpretability, better handling of constraints and integration of prior knowledge compared to other model-free RL methods. Furthermore, in contrast to

[194], only one policy provider (the differentiable MPC controller itself) is present. For this reason, as remarked in Section 6.1, this approach has the potential to result in fewer or no constraint violations at all depending on, e.g., the complexity of the underlying model and objective and the presence or lack of uncertainties [76, 233]. Likewise, at deployment, the proposed method relies only on the learnt MPC problem, whereas the method from [194] must additionally compute the output from the NN actor.

6.3. Background

Three preliminary components are necessary to illustrate the proposed MPC-RL control method. First, a modelling framework must be employed to build a representation of the dynamical behaviour of the highway network, and how RM can influence it to prevent, e.g., bottlenecks and spill-backs. Then, we report a classical formulation of MPC for this task that makes use of such a model, and propose some modifications to it that are relevant to this work. Lastly, the essential concepts of RL are briefly introduced.

Notation: the set of indices $\{1, \dots, N\} \subseteq \mathbb{N}$ is denoted as $\mathcal{N}$. Vector quantities are in bold. Inequalities on vectors are applied element-wise. To avoid confusion when other subscripts are present, dynamic quantities (e.g., ρ) at time step k can be also referred to as, e.g., $\rho_j(k)$. The i-th index of predicted quantities based on the information available at time step k are denoted as, e.g., $\hat{y}_{i|k}$.

6.3.1. METANET modelling framework

Based on the traffic phenomena to investigate, different approaches to modelling a highway network exist [202]. In this chapter, the macroscopic second-order METANET framework is employed to obtain a discrete-time dynamical representation of the behaviour of highway traffic under RM control. Proposed as a simulation tool in [142, 156], it made one of its first appearances in combination with MPC in [14, 90], and has since been extended to cover several phenomena and traffic measures [53, 159].

In the remainder of this section, the basics of METANET are introduced. In this framework, a network is represented as a directed graph where each segment (a continuous stretch of highway with homogeneous properties) is modelled as an arc, and origins are represented as nodes. Virtual nodes are also employed to separate segments with heterogeneous properties (e.g., due to a lane drop). We assume the network consists of $|\mathcal{S}|$ segments and $|\mathcal{O}|$ origins, where $\mathcal{S}$ and $\mathcal{O}$ are the sets of indices of segments and origins, respectively. At time step k, segment $j \in \mathcal{S}$ is characterised by its traffic density $\rho_j(k)$ and a mean speed $v_j(k)$, which form its state. Conversely, origin $o \in \mathcal{O}$ is characterised by the queue length $w_o(k)$. In the destination-independent variant, the

evolution in time of segment states is described by the following equations:

$$\rho_j(k+1) = \rho_j(k) + \frac{T}{L_j \lambda_j}\left(q_{j-1}(k) - q_j(k) + r_o(k)\right), \tag{6.1}$$

$$\begin{aligned} v_j(k+1) = v_j(k) + \frac{T}{\tau}\left(V_e\left(\rho_j(k)\right) - v_j(k)\right) + \frac{T}{L_j} v_j(k)\left(v_{j-1}(k) - v_j(k)\right) \\ - \frac{\eta T}{\tau L_j}\frac{\rho_{j+1}(k) - \rho_j(k)}{\rho_j(k) + \kappa} - \frac{\mu T r_o(k) v_j(k)}{L_j \lambda_j \left(\rho_j(k) + \kappa\right)}, \end{aligned} \tag{6.2}$$

$$q_j(k) = \lambda_j \rho_j(k) v_j(k), \tag{6.3}$$

$$V_e(\rho(k)) := v_{\text{free}} \exp\left(-\frac{1}{a}\left(\frac{\rho(k)}{\rho_{\text{crit}}}\right)^a\right), \tag{6.4}$$

where T is the sampling time, L_j and λ_j are respectively the length of the segment and its number of lanes, (6.3) describes the traffic flow $q_j(k)$ of segment j at time step k, (6.4) is the well-known equilibrium speed formula, and $\tau, \eta, \kappa, \mu, a, \rho_{\text{crit}}, v_{\text{free}}$ are model parameters describing different quantities and phenomena. In particular, ρ_{crit} indicates the traffic density at which traffic flow transitions from free-flow conditions to congested conditions. At this density, the flow rate of vehicles through the network is at its maximum, and any increase in the number of vehicles beyond this point results in reduced speeds and increased congestion. The free-flow speed v_{free} describes instead the speed at which vehicles can travel when the road is not congested or disrupted, i.e., it is the maximum speed that vehicles can achieve on a particular segment under low traffic volumes. Lastly, $r_o(k)$ is the incoming flow generated by origin o connected to segment j; if none is connected, it is null. In these equations, the index $j-1$ refers to the segment upstream of j, and $j+1$ to the segment downstream. For origins, the queue length evolves as

$$w_o(k+1) = w_o(k) + T\left(d_o^w(k) - r_o(k)\right), \tag{6.5}$$

where the origin's demand $d_o^w(k)$ acts as an uncontrollable external input affecting the queue dynamics. This work is mainly concerned with on-ramps; however, different types of models for origins exist, and interested readers are referred to [159]. Having defined the queue length as in (6.5), the on-ramp flow can be computed as

$$r_o(k) = \tilde{r}_o(k) \min\left\{d_o^w(k) + \frac{w_o(k)}{T}, C_o, C_o\left(\frac{\rho_{\max} - \rho_{j_o}(k)}{\rho_{\max} - \rho_{\text{crit}}}\right)\right\}, \tag{6.6}$$

where C_o is the capacity of the origin, $\rho_{j_0}(k)$ is the density of the segment j_o that is connected to this ramp, and $\rho_{\max}$ is the maximum density segments can withstand (assumed here independent of index j_o, for sake of simplicity). Lastly, $\tilde{r}_o(k) \in [0,1]$ is the metering rate control action, which allows modulating the entering ramp flow to prevent, e.g., congestion and spillback. However, to avoid numerical issues in the MPC gradient-based solvers due to the min operator in (6.6), which can cause the gradient to be zero over a vast region of the state-action space, it is more efficient to control the ramp flow r_o directly. This requires imposing the additional terms in (6.6) as constraints on the control action, as shown in the next subsection.

Finally, the downstream boundary conditions are here considered to be affected by congestion, i.e., the highway segment j_i that ends in destination $i \in \mathscr{D}$ is subject to the (virtual) downstream density modelled in [88] as

$$\rho_{j_i+1}(k) = \max\left\{\min\left\{\rho_{j_i}(k), \rho_{\text{crit}}\right\}, d_i^{\rho}(k)\right\}, \tag{6.7}$$

where $\mathscr{D}$ is the set of destination indices, $d_i^{\rho}(k)$ models the congestion density of destination i, and is regarded as another external disturbance.

In a more concise formulation, the states, actions and external factors at time k can be lumped in vectors as

$$\boldsymbol{x}_k = \begin{bmatrix} rho_1(k) & \dots & \rho_{|\mathscr{S}|}(k) & v_1(k) & \dots & v_{|\mathscr{S}|}(k) & w_1(k) & \dots & w_{|\mathscr{O}|}(k) \end{bmatrix}^\top, \tag{6.8}$$

$$\boldsymbol{u}_k = \begin{bmatrix} r_1(k) & \dots & r_{|\mathscr{O}|}(k) \end{bmatrix}^\top, \tag{6.9}$$

$$\boldsymbol{d}_k = \begin{bmatrix} d_1^w(k) & \dots & d_{|\mathscr{O}|}^w(k) & d_1^{\rho}(k) & \dots & d_{|\mathscr{D}|}^{\rho}(k) \end{bmatrix}^\top, \tag{6.10}$$

respectively. Then, the METANET framework can be exploited to build the nonlinear dynamics

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k), \tag{6.11}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$, with $n_x = 2|\mathscr{S}| + |\mathscr{O}|$, $n_u = |\mathscr{O}|$, $n_d = |\mathscr{O}| + |\mathscr{D}|$. The next section details how these dynamics can be used in designing a model-based controller.

6.3.2. MPC for ramp metering control

In what follows, the classical MPC-based paradigm for the RM control problem is presented and summarised.

MPC [163] is a well-known control framework for dynamical systems that is based on the formulation of a finite-horizon optimal control problem, consisting of three fundamental pillars: the prediction of the evolution of the system's dynamics over a finite window, a suitable objective function to be minimised along this prediction window, and the inclusion of constraints on state and/or action variables. The first is typically achieved by means of a (often approximate) model of the system's behaviour, which is used to simulate the evolution of the states along the window. The second is a function that quantifies the performance of the controller and is often a trade-off between conflicting objectives. Thirdly, MPC allows to systematically integrate into its structure constraints on the states and/or actions, thus yielding policies that are able to take these constraints into account in an optimal way. On the whole, the framework offers quite a degree of flexibility, and it is unsurprising that it lends itself also to RM control problems.

We define the MPC prediction horizon N_{p} and the control horizon $N_{\text{c}} \leq \frac{N_{\text{p}}}{M}$, where M is a natural number to distinguish the process timescale T from the controller timescale $T_{\text{c}} = \frac{T}{M}$. In other words, the process runs M times faster than the controller, with the MPC being solved only every M time steps (instead of at each time step). Once solved, the first optimal action is then applied for the next M time steps, in a receding horizon fashion. Therefore, in the MPC controller timescale, control action indices are indicated as i_{c} and

can be related to indices i of the dynamics timescale as $i_c(i) = \min\left\{\left\lfloor \frac{i}{M} M \right\rfloor, N_c - 1\right\}$.[1] Additionally, we define a constrained RM control scenario by imposing at time step k, for the on-ramps in the network, the constraint

$$w_o(k) \leq w_{\max}, \quad \forall o \in \mathscr{O}, \tag{6.12}$$

where $w_{\max}$ represents the maximum queue length the ramps should experience. For the sake of simplicity, it is assumed here that $w_{\max}$ is independent of the index o, and that all queue lengths are subject to this constraint, though it could be readily applied to only a subset of them. As mentioned in Section 6.2, this is usually done to prevent spillback, and similarly constrained scenarios can be found in, e.g., [90]. This constraint is not to be considered hard, but we favour control strategies that are able to satisfy it, if not always, at least for most time steps.

Altogether, given the current state x_k of the network, the MPC controller is given by

$$\begin{aligned}
\min_{\widehat{\boldsymbol{X}}_k, \widehat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & \sum_{i=0}^{N_p} \left(L_T(\hat{\boldsymbol{x}}_{i|k}) + \boldsymbol{\zeta}_i^\top \boldsymbol{\sigma}_i\right) + \xi \sum_{i=0}^{N_c-1} L_V(\hat{\boldsymbol{u}}_{i|k}) && && \text{(6.13a)}\\
\text{s.t.} \quad & \hat{\boldsymbol{x}}_{0|k} = \boldsymbol{x}_k, && && \text{(6.13b)}\\
& \hat{\boldsymbol{x}}_{i+1|k} = f(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_c(i)|k}, \boldsymbol{d}_k), && i = 0, \dots, N_p - 1, && \text{(6.13c)}\\
& h(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_c(i)|k}) \leq 0, && i = 0, \dots, N_p, && \text{(6.13d)}\\
& g(\hat{\boldsymbol{x}}_{i|k}, \boldsymbol{\sigma}_i) \leq 0, && i = 0, \dots, N_p, && \text{(6.13e)}\\
& \boldsymbol{\sigma}_i \geq 0, && i = 0, \dots, N_p, && \text{(6.13f)}
\end{aligned}$$

where the optimisation variables are the actions and states along the MPC control and prediction horizons, and the slack variables (whose purpose is explained later in the section), respectively in order

$$\widehat{\boldsymbol{X}}_k = \begin{bmatrix} \hat{\boldsymbol{x}}_{0|k}^\top & \dots & \hat{\boldsymbol{x}}_{N_p|k}^\top \end{bmatrix}^\top \in \mathbb{R}^{n_x\left(N_p+1\right)}, \tag{6.14}$$

$$\widehat{\boldsymbol{U}}_k = \begin{bmatrix} \hat{\boldsymbol{u}}_{0|k}^\top & \dots & \hat{\boldsymbol{u}}_{N_c-1|k}^\top \end{bmatrix}^\top \in \mathbb{R}^{n_u N_c}, \tag{6.15}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\sigma}_0^\top & \dots & \boldsymbol{\sigma}_{N_p}^\top \end{bmatrix}^\top \in \mathbb{R}^{|\mathscr{O}|\left(N_p+1\right)}, \tag{6.16}$$

where $\widehat{\boldsymbol{X}}_k$ and $\widehat{\boldsymbol{U}}_k$ are defined in analogy to (6.8) and (6.9) respectively, and $\boldsymbol{\sigma}_i$ collects the slack variables at the predicted step i corresponding to all on-ramps, i.e., $\boldsymbol{\sigma}_i = \begin{bmatrix} \sigma_i^1 & \dots & \sigma_i^{|\mathscr{O}|} \end{bmatrix}^\top$. As aforementioned, once this optimisation problem is solved, we apply the optimal control action $\hat{\boldsymbol{u}}_{0|k}^\star$ from time step k to $k + M - 1$, as per the receding horizon approach.

The objective (6.13a) comprises three terms. The first is the total time spent (TTS), a metric frequently used to quantify efficiency of traffic control, defined as $L_T : \mathbb{R}^{n_x} \to \mathbb{R}$

$$L_T(\boldsymbol{x}_k) \coloneqq T\left(\sum_{j \in \mathscr{S}} L_j \lambda_j \rho_j(k) + \sum_{o \in \mathscr{O}} w_o(k)\right). \tag{6.17}$$

[1] This definition of $i_c(i)$ entails that the last action is kept constant for the remainder of the prediction horizon, a common choice in literature, though more complex solutions could be employed.

The second term of (6.13a) is a cost proportional to the variability of the inputs along the control horizon, where $L_{\mathrm{V}} : \mathbb{R}^{n_u} \to \mathbb{R}$ is

$$L_{\mathrm{V}}(\boldsymbol{u}_k) \coloneqq \sum_{o \in \mathscr{O}} \left(\frac{r_o(k) - r_o(k-1)}{C_o} \right)^2. \tag{6.18}$$

Lastly, the third term in (6.13a) acts as a penalty for the slack variables. The nonnegative weights $\xi \in \mathbb{R}$ and $\boldsymbol{\zeta}_i \in \mathbb{R}^{|\mathscr{O}|}$, $i = 0, \dots, N_{\mathrm{p}}$, govern the trade-off between these three terms and are selected by the designer so as to encode the desired behaviour of the controller [89]. Section 6.4 will show how these parameters (under the new symbols θ_{V} and $\boldsymbol{\Theta}_{\mathrm{C}}$), together with others, can be adjusted automatically via RL instead of having to be specified manually.

In (6.13d) and (6.13e), $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{4|\mathscr{O}|}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{|\mathscr{O}|} \to \mathbb{R}^{|\mathscr{O}|}$ correspond to all the inequality constraints to be imposed in the MPC problem. In particular, they are defined as

$$h(\boldsymbol{x}_i, \boldsymbol{u}_{i_{\mathrm{c}}}) \coloneqq \begin{bmatrix} h_1(\boldsymbol{x}_i, \boldsymbol{u}_{i_{\mathrm{c}}})^\top & \dots & h_{|\mathscr{O}|}(\boldsymbol{x}_i, \boldsymbol{u}_{i_{\mathrm{c}}})^\top \end{bmatrix}^\top, \tag{6.19}$$

$$g(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \coloneqq \begin{bmatrix} g_1(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) & \dots & g_{|\mathscr{O}|}(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \end{bmatrix}^\top, \tag{6.20}$$

where

$$h_o(\boldsymbol{x}_i, \boldsymbol{u}_{i_{\mathrm{c}}}) \coloneqq \begin{bmatrix} -r_o(i_{\mathrm{c}}) \\ r_o(i_{\mathrm{c}}) - C_o \\ r_o(i_{\mathrm{c}}) - d_o^w(i) - \frac{w_o(i)}{T} \\ r_o(i_{\mathrm{c}}) - C_o \left(\frac{\rho_{\max} - \rho_{j_o}(i)}{\rho_{\max} - \rho_{\mathrm{crit}}} \right) \end{bmatrix}, \qquad \forall o \in \mathscr{O}, \tag{6.21}$$

$$g_o(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \coloneqq w_o(i) - w_{\max} - \sigma_i^o, \qquad \forall o \in \mathscr{O}, \tag{6.22}$$

and we have dropped the dependency on i in i_{c} for readability. The constraints $h_o(\boldsymbol{x}_i, \boldsymbol{u}_{i_{\mathrm{c}}}) \leq 0$ emulate the behaviour of (6.6) and prevent the control action from assuming invalid values. The constraints $g_o(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \leq 0$ instead incorporate the queue requirements (6.12) into the MPC controller as soft constraints, where $\boldsymbol{\sigma}_k \geq 0$ is a slack variable whose purpose is to relax the constraint in order to avoid infeasibility issues, and so it must be appropriately penalised in the objective (6.13a). We opt for a soft constraint because 1) it is assumed that small violations to this constraint, although undesired, are tolerable, and 2) we need a mechanism to both relax constraints during the learning process and penalise violations in order to inform the agent about policies that violate constraints and ones that do not.

Note that in (6.13b) we assume that the current process state $\boldsymbol{x}_k$ is fully measurable. If this is not the case, an observer is required to provide an estimate of it. Likewise, in (6.13c) we assume the external inputs $\boldsymbol{d}_k$ to be fully known. If this is not the case, a forecast of the evolution of these quantities along the MPC horizon is required.

6.3.3. Reinforcement learning

To better understand how MPC can be integrated with RL, a brief introduction to the latter is here provided. In what follows, we abide by the canonical RL approach [197].

Given the continuous state $\boldsymbol{s}$ and action $\boldsymbol{a}$, a discrete-time Markov Decision Process with state transitions $\boldsymbol{s} \xrightarrow{\boldsymbol{a}} \boldsymbol{s}_+$ and underlying conditional probability density

$$\mathbb{P}(\boldsymbol{s}_+ \mid \boldsymbol{s}, \boldsymbol{a}) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to [0,1] \tag{6.23}$$

is used to model the discrete-time system dynamics. The performance of a given deterministic policy $\pi_{\boldsymbol{\theta}} : \mathbb{R}^{n_s} \to \mathbb{R}^{n_a}$ parametrised in $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ is defined as

$$J(\pi_{\boldsymbol{\theta}}) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k L(\boldsymbol{s}_k, \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_k))\right], \tag{6.24}$$

where $L : \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to \mathbb{R}$ is a stage cost function and $\gamma \in (0,1]$ the discount factor. The goal of the RL algorithm is then to find the optimal policy $\pi_{\boldsymbol{\theta}}^\star$ as

$$\pi_{\boldsymbol{\theta}}^\star = \arg\min_{\boldsymbol{\theta}} \quad J(\pi_{\boldsymbol{\theta}}). \tag{6.25}$$

Since it is in general impossible to find and characterise the true unknown optimal value functions and policy $V_\star, Q_\star, \pi_\star$, function approximation techniques (such as NN and, as in this chapter, MPC) have been employed as a powerful alternative for tackling problem (6.25) [37]. Depending on the algorithm, these allow formulating the approximations $V_{\boldsymbol{\theta}}$, $Q_{\boldsymbol{\theta}}$, and $\pi_{\boldsymbol{\theta}}$, which are then used to solve (6.25) either directly or indirectly via iterative gradient updates of the parametrisation

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \psi(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_{i+1}, \boldsymbol{\theta}), \tag{6.26}$$

where $\alpha \in \mathbb{R}_+$ is the learning rate, m is the size of the batch of observations used in the update, and ψ captures the controller's performance and varies from algorithm to algorithm. Direct (or policy-based) methods explicitly parametrise and learn the approximation $\pi_{\boldsymbol{\theta}}$ of the optimal policy itself. For instance, policy gradient methods attempt to solve (6.25) directly by moving along the gradient of the policy, i.e.,

$$\mathbb{E}[\psi(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_{i+1}, \boldsymbol{\theta})] = J(\pi_{\boldsymbol{\theta}}). \tag{6.27}$$

On the other hand, indirect (or value-based) methods opt to indirectly derive the optimal policy by estimating and learning the value functions $V_{\boldsymbol{\theta}}, Q_{\boldsymbol{\theta}}$ from the underlying RL task, and implicitly derive the policy from these. A member of this category, Q-learning learns to approximate the action-value by solving

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}\big[\|Q_\star(s,a) - Q_{\boldsymbol{\theta}}(s,a)\|^2\big], \tag{6.28}$$

with the hope of indirectly recovering the optimal policy from it. In its first-order recursive (i.e., with $m = 1$) formulation, $\boldsymbol{\theta}$ is updated via (6.26) with $\psi(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_{i+1}, \boldsymbol{\theta}) = \delta_i^2$, where δ_i is the well-known temporal difference (TD) error:

$$\delta_i = L(\boldsymbol{s}_i, \boldsymbol{a}_i) + \gamma V_{\boldsymbol{\theta}'}(\boldsymbol{s}_{i+1}) - Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i). \tag{6.29}$$

Note that the value function estimate in (6.29) is usually evaluated for a frozen copy of the parametrisation $\theta' = \theta$ so as to prevent it from contributing to the gradient update.

6.4. MPC-based RL for ramp metering

As discussed in Section 6.1, the closed-loop performance of the MPC controller (6.13) is dependent on its components, especially the prediction model. Mismatches in the dynamics can undermine the ability of MPC to reliably predict the system behaviour, thus impacting the controller performance and its ability to avoid constraint violations. The highly nonlinear nature of the METANET model contributes to making this issue even more relevant, since the model calibration requires the application of some nonlinear technique, e.g., nonlinear least-squares [150]. In what follows, we show how these shortcomings can be addressed by leveraging online closed-loop data to learn most of the MPC parameters automatically via RL.

6.4.1. Ramp metering as an RL task

To appropriately apply an RL algorithm to the RM task, we first need to define a proper stage cost $L(x_k, u_k)$ to quantify the performance of a policy in controlling the metering, as per (6.24). Also referred to as reward in literature (i.e., the negative of the cost), it defines the overall return that the RL algorithm must minimise and it encodes the objective that the designer wants to see minimised by the MPC control policy. Consider

$$L(\boldsymbol{x}_k, \boldsymbol{u}_k) := c_{\mathrm{T}} L_{\mathrm{T}}(\boldsymbol{x}_k) + c_{\mathrm{V}} L_{\mathrm{V}}(\boldsymbol{u}_k) + c_{\mathrm{C}} L_{\mathrm{C}}(\boldsymbol{x}_k), \tag{6.30}$$

where $L_{\mathrm{C}} : \mathbb{R}^{n_x} \to \mathbb{R}$ is

$$L_{\mathrm{C}}(\boldsymbol{x}_k) := \sum_{o \in \bar{\mathcal{O}}} \max\{0, w_o(k) - w_{\max}\}. \tag{6.31}$$

The first term in (6.30) penalises the time spent travelling through the network and waiting in queues, at the current time step k. This term is representative of the TTS criterion, which is often the primary target of RM control strategies in literature. The second term penalises variations of the current control action $r_o(k)$ compared to the previous instant $r_o(k-1)$, and attempts to punish policies that are too jerky in the input signal. Lastly, the third contribution acts as the RL counterpart to the MPC soft constraint (6.22) and penalisation of the corresponding slack variables in the MPC objective (6.13a): it penalises violations of the queue constraint so that the agent is encouraged to come up with policies that are able to satisfy this constraint. Scalars c_{T}, c_{V}, and c_{C} are weights that balance each term differently, and it is the designer's task to choose these in such a way to properly embed the learning objective in the RL stage cost.[2] Next, we delve into how MPC can be used as function approximation to solve the ramp metering RL task.

6.4.2. MPC as function approximation in RL

While the RL stage cost (6.30) defines and guides the learning process, a function approximation is needed to estimate the underlying optimal value and policy functions. A recurrent candidate for this rule in the literature is NNs. However, in Section 6.1 it was discussed how [75] suggests the employment of a parametric MPC scheme as function approximation in the RL problem, and lies the foundations of its theory. In this chapter,

[2] In our work, the values for c_{T}, c_{V}, and c_{C} were manually adjusted to achieve a reasonable learning process.

we follow this paradigm. In particular, we choose to parameterise the cost function, the prediction model as well as some of the constraints in (6.13). Then, the RL algorithm is applied to adjust the parametrisation based on observed state transitions and rewards, in order to achieve better closed-loop performance in terms of (6.24).

Consider the MPC controller with parametrisation $\boldsymbol{\theta}$

$$\begin{aligned}
\min_{\hat{\boldsymbol{X}}_k, \hat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & \sum_{i=0}^{N_\mathrm{p}} \gamma^i \left(\theta_\mathrm{T} L_\mathrm{T}(\hat{\boldsymbol{x}}_{i|k}) + \boldsymbol{\theta}_\mathrm{C}^{i\,\top} \boldsymbol{\sigma}_i \right) + \theta_\mathrm{V} \sum_{i=0}^{N_\mathrm{c}-1} \gamma^{iM} L_\mathrm{V}(\hat{\boldsymbol{u}}_{i|k}) + \lambda_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{0|k}) \\
& + \sum_{i=1}^{N_\mathrm{p}-1} \gamma^i \ell_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{i|k}) + \gamma^{N_\mathrm{p}} \Gamma_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{N_\mathrm{p}|k}) && \text{(6.32a)} \\
\text{s.t.} \quad & \hat{\boldsymbol{x}}_{0|k} = \boldsymbol{x}_k, && \text{(6.32b)} \\
& \hat{\boldsymbol{x}}_{i+1|k} = f_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_\mathrm{c}(i)|k}, \boldsymbol{d}_k), \qquad i = 0, \dots, N_\mathrm{p} - 1, && \text{(6.32c)} \\
& h_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_\mathrm{c}(i)|k}) \leq 0, \qquad i = 0, \dots, N_\mathrm{p}, && \text{(6.32d)} \\
& g(\hat{\boldsymbol{x}}_{i|k}, \boldsymbol{\sigma}_i) \leq 0, \qquad i = 0, \dots, N_\mathrm{p}, && \text{(6.32e)} \\
& \boldsymbol{\sigma}_i \geq 0, \qquad i = 0, \dots, N_\mathrm{p}, && \text{(6.32f)}
\end{aligned}$$

with $\boldsymbol{\Theta}_\mathrm{C} = \begin{bmatrix} \boldsymbol{\theta}_\mathrm{C}^{0\,\top} & \dots & \boldsymbol{\theta}_\mathrm{C}^{N_\mathrm{p}\,\top} \end{bmatrix}^\top \in \mathbb{R}^{|\mathscr{O}|\left(N_\mathrm{p}+1\right)}$. Because this controller acts as function approximation and must be able to accurately predict the realisations of the RL stage cost along the prediction horizon, the objective (6.32a) is devised so as to mimic (6.30). Furthermore, note the additional parameterised terms $\lambda_{\boldsymbol{\theta}}, \ell_{\boldsymbol{\theta}}, \Gamma_{\boldsymbol{\theta}} : \mathbb{R}^{n_x} \to \mathbb{R}$. These are the learnable initial, stage, and terminal costs respectively, and serve the purpose of enriching the parametrisation and facilitating generalisation across the state-action space. The initial cost is a linear combination of the state

$$\lambda_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \coloneqq \sum_{j \in \mathscr{S}} \left(\theta^\rho_{\lambda,j} \frac{\rho_j(k)}{\rho_\mathrm{max}} + \theta^v_{\lambda,j} \frac{v_j(k)}{v_\mathrm{max}} \right) + \sum_{o \in \mathscr{O}} \theta^w_{\lambda,o} \frac{w_o(k)}{w_\mathrm{max}}, \tag{6.33}$$

where ρ_max, v_max, w_max are here used as fixed, non-learnable normalisation constants in order to improve the stability of the learning process. As explained in [75], since our objective is economic, this initial cost is required in order for the MPC scheme to recover the optimal policy. Conversely, stage and terminal costs are chosen to be quadratic

$$\ell_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \coloneqq \sum_{j \in \mathscr{S}} \theta^\rho_{\ell,j} \left(\frac{\rho_j(k) - \rho_\mathrm{sp}}{\rho_\mathrm{max}} \right)^2 + \sum_{j \in \mathscr{S}} \theta^v_{\ell,j} \left(\frac{v_j(k) - v_\mathrm{sp}}{v_\mathrm{max}} \right)^2 + \sum_{o \in \mathscr{O}} \theta^w_{\ell,o} \left(\frac{w_o(k)}{w_\mathrm{max}} \right)^2, \tag{6.34}$$

$$\Gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \coloneqq \sum_{j \in \mathscr{S}} \theta^\rho_{\Gamma,j} \left(\frac{\rho_j(k) - \rho_\mathrm{sp}}{\rho_\mathrm{max}} \right)^2 + \sum_{j \in \mathscr{S}} \theta^v_{\Gamma,j} \left(\frac{v_j(k) - v_\mathrm{sp}}{v_\mathrm{max}} \right)^2 + \sum_{o \in \mathscr{O}} \theta^w_{\Gamma,o} \left(\frac{w_o(k)}{w_\mathrm{max}} \right)^2, \tag{6.35}$$

where densities and speeds are encouraged to track the fixed non-learnable setpoints ρ_sp and v_sp respectively, and the queues to be as small as possible. Regarding the METANET model $f_{\boldsymbol{\theta}}$ in (6.32c), we allow the RL algorithm to adjust two fundamental parameters in the dynamics, i.e., $\tilde{\rho}_\mathrm{crit}$ and $\tilde{a}$ (which in general can differ, during the learning process, from their counterparts ρ_crit, a of the real dynamics). In this way, the agent is able to partially tune the prediction model based on performance (6.24) rather than on

system identification, implying that these learning parameters might grow to different values compared to the ones belonging to the true underlying dynamics. Finally, also constraints $h_{\boldsymbol{\theta}}$ (6.32d) get parameterised, since $\tilde{\rho}_{\text{crit}}$ appears in it; see (6.13d) and (6.21).

Remark 6.1. *In general, a parametrisation that makes use of quadratic terms with fixed setpoints, as in* (6.34) *and* (6.35)*, can lead to suboptimal MPC policies in low-demand, free-flow regimes because it penalises low densities and low speeds as much as it penalises high densities and high speeds. At the same time, the theory at the foundation of MPC-based RL assumes the controller's parametrisation to be rich enough to adequately capture the real value function [75]. For this reason, in this work, we opt for the aforementioned adjustable quadratic cost terms. That being said, one could envision more involved and more performing learnable cost terms, especially in more complex and difficult traffic settings. For instance, asymmetric penalty functions are a valid choice, e.g., a left-sided Huber loss that quadratically penalises high traffic quantities, but penalises low ones only linearly.*

Remark 6.2. *The MPC parameters $\theta_T, \theta_V, \boldsymbol{\Theta}_C$ are disjoint from the RL stage cost weights c_T, c_V, c_C of* (6.30)*. While they encode similar notions, there is in general no reason to expect the values of the former to converge to the values of the latter during learning. Rather, they will converge to those values that maximise the controller's closed-loop performance.*

The whole parametrisation is

$$\boldsymbol{\theta} = \begin{bmatrix} \tilde{\rho}_{\text{crit}} & \tilde{a} & \theta_{\{\text{T,V}\}} & \boldsymbol{\Theta}_{\text{C}}^{\top} & \boldsymbol{\theta}_{\{\lambda,\ell,\Gamma\}}^{\{\rho,v,w\}\top} \end{bmatrix}^{\top}. \tag{6.36}$$

Table 6.1 shows a summary of the parametrisation, reporting the scope and the space of each term. As reported in Table 6.2 in the next section, these real spaces are often bounded in order to avoid undesired consequences, e.g., to preserve convexity of the parametric cost terms, or to prevent parameters that have some physical meaning from taking unrealistic values.

Table 6.1.: Parametrisation $\boldsymbol{\theta}$ of the MPC function approximation (6.32)

Symbol	Scope	Space
$\tilde{\rho}_{\text{crit}}$	model, cost, constraint	$\mathbb{R}$
$\tilde{a}$	model	$\mathbb{R}$
θ_{T}	cost - TTS weight	$\mathbb{R}$
θ_{V}	cost - control variability weight	$\mathbb{R}$
$\boldsymbol{\Theta}_{\text{C}}$	cost - slack weights	$\mathbb{R}^{\lvert\mathscr{O}\rvert\left(N_{\text{p}}+1\right)}$
$\boldsymbol{\theta}_{\{\lambda,\ell,\Gamma\}}^{\{\rho,v\}}$	{init., stage, term.} cost - $\{\rho, v\}$ weights	$\mathbb{R}^{\lvert\mathscr{S}\rvert}$
$\boldsymbol{\theta}_{\{\lambda,\ell,\Gamma\}}^{w}$	{init., stage, term.} cost - w weights	$\mathbb{R}^{\lvert\mathscr{O}\rvert}$

Scheme (6.32) yields the approximation $V_{\boldsymbol{\theta}} : \mathbb{R}^{n_x} \to \mathbb{R}$ of the value function as

$$\begin{aligned} V_{\boldsymbol{\theta}}(\boldsymbol{x}_k) = \min_{\widehat{\boldsymbol{X}}_k, \widehat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & \text{(6.32a)} \\ \text{s.t.} \quad & \text{(6.32b)} - \text{(6.32f)}. \end{aligned} \tag{6.37}$$

The value function (6.37) satisfies the fundamental equalities of the Bellman equations [75], so that

$$\begin{aligned} Q_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \min_{\hat{\boldsymbol{X}}_k, \widehat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & (6.32a) \\ \text{s.t.} \quad & (6.32b) - (6.32f), \\ & \hat{\boldsymbol{u}}_{0|k} = \boldsymbol{u}_k. \end{aligned} \tag{6.38}$$

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_k) = \arg\min_{\boldsymbol{u}} \quad Q_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}). \tag{6.39}$$

In practice, policy (6.39) is found as the first optimal action $\hat{\boldsymbol{u}}^{\star}_{0|k}$ from the arg min of (6.37). The goal of this parametrisation is to give the MPC scheme (6.32) enough degrees of freedom to sufficiently adapt to the RL task at hand. The cardinal point is that $\boldsymbol{\theta}$ must be rich enough to allow the scheme to capture the optimal policy $\pi^{\star}$ [75]. Of course, the choice of parametrisation is not unique, and other solutions may be viable: similarly to NNs, where the topology of the network and the number of neurons and activation functions to used in each layer are arbitrary, here also different choices are possible. In the context of RM, the inclusion of some parameters in $\boldsymbol{\theta}$ comes from the knowledge of the system and its control, e.g., the system's high sensitivity to the dynamics parameters a, ρ_{crit} and the cost weights θ_{T}, θ_{V}, $\boldsymbol{\theta}_{\text{C}}$. Other elements of the parametrisation, such as the additional quadratic terms, are instead dictated by the framework [75] to ensure a rich approximation scheme. However, unlike NNs, the benefit of this model-based approach is that knowledge about the system can be incorporated in the parametrisation rather trivially. Further discussion on different parametrisations and their impact on the learning outcome can be found in the appendix.

Beyond a proper selection of $\boldsymbol{\theta}$, other factors influence the suboptimality of the MPC-based RL solution. During training, the RL algorithm can only process a finite amount of data, whereas theoretical optimality is achieved only as this amount approaches infinity [197]. The MPC approximation scheme is also inherently suboptimal due to its nonlinearity, converging to global optimality only in the asymptotic limit, though the issue can be alleviated via, e.g., multi-start [163].

6.4.3. Second-order LSTD Q-learning

In order to adjust the parametrisation $\boldsymbol{\theta}$ of (6.32), a second-order least-squares temporal difference (LSTD) Q-learning algorithm [114] is employed to find the policy $\pi_{\boldsymbol{\theta}}$ that minimises the closed-loop performance. Q-learning is one of the most well-known indirect, temporal difference algorithms available in RL. In essence, it searches for the parametrisation that best fits the action-value function $Q_{\boldsymbol{\theta}}$ to the observed data, with the aim of recovering via this approximation the unknown optimal $Q_{\star}$ and, indirectly from that, the optimal policy $\pi_{\star}$. It has also shown very promising results in the context of MPC [60]. The second-order Newton's method, coupled with an experience replay buffer of the past observed transitions [122] and a re-formulation of the Q-fitting problem as a least-squares, ensures faster convergence and higher sample efficiency compared to traditional first-order methods. For more details, see [60].

More concretely, given the approximation $Q_{\boldsymbol{\theta}}$ (irrespective of its nature, e.g., MPC, NN, etc.), Q-learning solves the least-squares Bellman residual problem (6.28) via the

second-order gradient update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{H}^{-1} \boldsymbol{p}, \tag{6.40}$$

where $\alpha \geq 0$ is the learning rate, and the gradient $\boldsymbol{p}$ and Hessian $\boldsymbol{H}$ are found as

$$\boldsymbol{p} = -\sum_{i=1}^{m} \delta_i \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i), \tag{6.41}$$

$$\boldsymbol{H} = \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i) \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}^{\top}(\boldsymbol{s}_i, \boldsymbol{a}_i) - \delta_i \nabla_{\boldsymbol{\theta}}^2 Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i), \tag{6.42}$$

with m denoting the experience sample batch size and δ_i the TD error (6.29). The update rule (6.40) requires differentiation of the MPC scheme (6.38) with respect to $\boldsymbol{\theta}$ to find $\nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$ and $\nabla_{\boldsymbol{\theta}}^2 Q_{\boldsymbol{\theta}}$. A nonlinear programming sensitivity analysis demonstrates how these quantities can be computed through differentiation of the Lagrangian of (6.38) [35]. Prior to any update, the computed Hessian matrix is modified to be positive-definite by the addition of a multiple of the identity matrix [151]. Furthermore, to impose lower and upper bounds on the parameters (as reported in Table 6.2), (6.40) is cast as the following optimisation problem

$$\begin{aligned} \Delta\boldsymbol{\theta}^{\star} = \arg\min_{\Delta\boldsymbol{\theta}} \quad & \frac{1}{2} \Delta\boldsymbol{\theta}^{\top} \boldsymbol{H} \Delta\boldsymbol{\theta} + \alpha \boldsymbol{p}^{\top} \Delta\boldsymbol{\theta} \\ \text{s.t.} \quad & \boldsymbol{\theta}_{\text{lb}} \leq \boldsymbol{\theta} + \Delta\boldsymbol{\theta} \leq \boldsymbol{\theta}_{\text{ub}}, \\ & \Delta\boldsymbol{\theta}_{\text{lb}} \leq \Delta\boldsymbol{\theta} \leq \Delta\boldsymbol{\theta}_{\text{ub}}. \end{aligned} \tag{6.43}$$

This formulation allows to limit the rate of change of each parameter with $\Delta\boldsymbol{\theta}_{\text{lb}}$ and $\Delta\boldsymbol{\theta}_{\text{ub}}$. The parametrisation is then updated as $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}^{\star}$. Note that, compared to (6.32), (6.43) adds negligible computational complexity as it is a convex quadratic problem and is usually solved at a lower, batch-update frequency (as explained in Section 6.5.1).

Lastly, as in [233], exploratory behaviour is ensured during learning by adding to the MPC objective (6.32a) the perturbation term $\boldsymbol{q}^{\top} \boldsymbol{u}_0$, with $\boldsymbol{q}$ randomly chosen from, e.g., a normal distribution. When properly calibrated, this perturbation on the first action ensures that enough exploration is added on top of the policy in order to prevent the Q-learning agent from getting stuck in very suboptimal local minima.

6.5. Numerical case study

To examine the performance of the proposed method for RM control, we implement and simulate the algorithm when applied to a highway network benchmark.

6.5.1. Configuration

Network environment

Let us consider a simple highway traffic network taken from [88] and pictured in Figure 6.1, which will serve as the RL environment for the task at hand. The network consists of three segments, each 1 km in length and with two lanes. The first segment S_1 is supplied by the uncontrolled mainstream origin O_1, which simulates incoming traffic from the unmodelled upstream region of the network, and is characterised by its capacity C_1.

Between segments S_2 and S_3, additional traffic can enter the network via the on-ramp O_2. This origin has capacity C_2 and offers the only control measure in order to avoid congestion in the network. However, we would also like to keep the queue length on this on-ramp to 50 vehicles or fewer. Lastly, traffic exits the network via the only available destination D_1 with congested outflow.

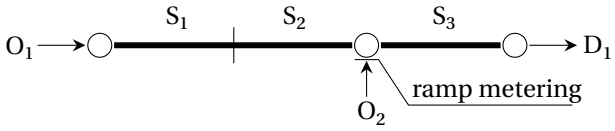

Figure 6.1.: Structure of the three-segment highway network

As described in Section 6.3.1, the network environment is subject to one external input per origin and destination. Figure 6.2 depicts the demands at both origins, starting low and then increasing to near capacity within 20 minutes. With some delay, also congestion at the destination rapidly appears. These profiles are intended to simulate a peak-hour-like traffic (e.g., morning or evening rush), and are devised in such a way to induce a variable degree of congestion in the network, if the on-ramp is not properly controlled. At the start of each training episode, these two-peak profiles are generated randomly to introduce a degree of variability in the task, aimed at enhancing the agent's generalisation to a wider variety of congestion scenarios. Nevertheless, generalisation in RL is challenging [73], and a performance drop is anticipated were the controller to face a new unforeseen scenario, e.g., with a different number of peaks. At the same time, our approach exhibits an inherent level of robustness: it incorporates an online learning process that can seamlessly integrate novel training data on the fly, and at its core is a model-based controller, which generally extrapolates better to new situations in comparison to its model-free counterparts.

Model predictive control

The controller (6.32) is deployed to control the on-ramp flow with the aim of avoiding congestion in the traffic network environment. In line with other similar benchmarks in the literature, e.g., [88–90], $M = 6$, meaning that, while the process dynamics are stepped every 10 seconds, the controller is run only once per minute. The prediction horizon is set to $N_\mathrm{p} = 24$ (which is in the order of the average travel time through the network), and the control horizon to $N_\mathrm{c} = 3$.

To showcase the ability of RL to automatically improve the performance of the MPC controller, the parametrisation $\boldsymbol{\theta}$ of the controller is poorly initialised on purpose. In particular, in order to replicate the effects of a poor system identification phase, the parameters of the prediction model are initialised with a 30% error with respect to their true values, i.e., $\tilde{\rho}_\mathrm{crit} = 0.7\rho_\mathrm{crit}$, $\tilde{a} = 1.3a$ (both learnable), and $1.3v_\mathrm{free}$ (fixed). Furthermore, the remaining parametrisation of the objective function of the MPC optimisation problem is left untuned, imitating a suboptimal, low-expertise design process of the controller. Table 6.2 reports the initial values of each learnable parameter. The other non-learnable parameters in (6.32) are set to the same values as in the real process. The queue threshold

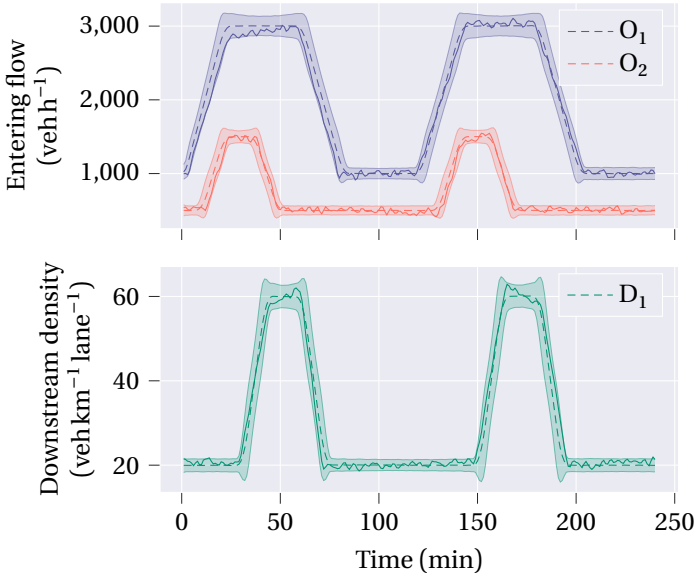

Figure 6.2.: External inputs affecting the network's dynamics, in the form of the demands at the origins (upper) and the congestion scenario at the destination (lower). The shaded areas represent the 2-standard deviation ranges from which the random demands are sampled, whereas the solid lines represent one random sample for each.

on O_2 is set to $w_{\max} = 50\,\text{veh}$. Lastly, the setpoints are set to $\rho_{\text{sp}} = 0.7\rho_{\text{crit}}$, $v_{\text{sp}} = 1.3v_{\text{free}}$, and the normalisation coefficients to $\rho_{\max} = 180\,\text{veh/km/lane}$, $w_{\max} = 50\,\text{veh}$, and $v_{\max} = 1.3v_{\text{free}}$.

Table 6.2.: Initial values, bounds and dimensions of the parametrisation $\boldsymbol{\theta}$ of (6.32)

Symbol	Initial value	Bounds	Dimension
$\tilde{\rho}_{\text{crit}}$	23.45	$[10, 162]$	$\text{veh}\,\text{km}^{-1}\,\text{lane}^{-1}$
$\tilde{a}$	2.4271	$[1.1, 3]$	-
θ_{T}	1	$[10^{-3}, \infty)$	$\text{veh}^{-1}\,\text{h}^{-1}$
θ_{V}	160000	$[10^{-3}, \infty)$	$\text{veh}^2\,\text{h}^{-2}$
$\boldsymbol{\Theta}_{\text{C}}$	5	$[10^{-3}, \infty)$	veh^{-1}
$\boldsymbol{\theta}_{\lambda}^{\{\rho,v,w\}}$	1	$(-\infty, \infty)$	-
$\boldsymbol{\theta}_{\{\ell,\Gamma\}}^{\{\rho,v,w\}}$	1	$[10^{-6}, \infty)$	-

Reinforcement learning

The agent is trained in an episodic fashion, with each episode lasting $T_{\text{fin}} = 4\,\text{h}$, i.e., each episode features two peaks in demands and congestion in a row (for reference, see Figure 6.2). At the end of each episode, the demands are generated anew randomly, and the state of the network is reset to steady-state in order to avoid unreasonable accumulation of vehicles in queues from past episodes. With the same frequency, the parametrisation $\boldsymbol{\theta}$ is updated, and a new episode is started, till a termination condition of the learning process is met, typically in the form of a maximum number of iterations. Here, we train our agent for 80 episodes. With the aforementioned update frequency, the proposed learning setup is off-policy, since θ remains fixed throughout each episode, ensuring stable data collection.

The stage cost function $L(\boldsymbol{x}_k, \boldsymbol{u}_k)$, with which the traffic network feeds a cost signal back to the agent according to (6.30), has coefficients $c_{\text{T}} = 5$, $c_{\text{V}} = 1600$, and $c_{\text{C}} = 5$.

Based on results obtained from initial simulations, the algorithm's hyperparameters are set as follows. The discount factor is set to $\gamma = 0.98$. The learning rate is initially set to $\alpha = 0.925$, and decayed by a multiplicative factor of 0.925 at the end of each update. The maximum update change of each parameter is set to 30% of its current value, i.e., $\Delta\boldsymbol{\theta}_{\text{ub}} = -\Delta\boldsymbol{\theta}_{\text{lb}} = 0.3\boldsymbol{\theta}$. As aforementioned, a replay buffer is used to store past observations in memory and re-use them. In particular, the buffer size is set up to store transitions from the 10 past episodes and, before performing an update, a batched sample half its size is drawn from this buffer. In turn, half of this batch is dedicated to containing information from the latest two and a half episodes, whereas the remaining half is sampled uniformly at random from even older transitions. All together, these measures contribute to stabilising the learning process.

Lastly, exploration is induced as described in Section 6.4.3 by sampling the cost perturbation from a normal distribution in an ε-greedy fashion, i.e., $\boldsymbol{q} \sim \mathcal{N}(0, \sigma_q)$ with probability ε; otherwise, $\boldsymbol{q} = 0$. The exploration strength is $\sigma_q = 0.025$, and the exploration probability $\varepsilon = 0.5$. Both are decayed by half at the end of each episode.

Table 6.3 summarises all the hyper-parameters and non-learnable parameters of the traffic environment and the MPC controller, as found in [88], as well as of the RL algorithm.

Setup

The simulations were run on a Ubuntu 20.04.6 server equipped with 16 AMD EPYC 7252 (3.1 GHz) processors and 252GB of RAM, and implemented in Python 3.11.4. The nonlinear optimisation problems were formulated and solved with the symbolic framework CasADi [11] and its interface to the IPOPT solver [212]. The source code and simulation results are open and available in the following repository: `https://github.com/FilippoAiraldi/mpcrl-for-ramp-metering`.

6.5.2. Comparison

Alongside the proposed MPC-based RL controller, we implement and simulate on the same setup three other policies. These include another learning-based approach based on DDPG, a DRL algorithm that has been proven effective in freeway control [194, 214],

Table 6.3.: Hyper- and other non-learnable parameters

METANET	Symbol	T	τ	η	κ	μ
	Value	10	18	60	40	0.0122
	Dimension	s	s	$\mathrm{km^2\,lane^{-1}}$	$\mathrm{veh\,km^{-1}\,lane^{-1}}$	-
	Symbol	$\rho_{\max}$	ρ_{crit}	v_{free}	a	$C_{\{1,2\}}$
	Value	180	33.5	102	1.867	{3500, 2000}
	Dimension	$\mathrm{veh\,km^{-1}\,lane^{-1}}$	$\mathrm{veh\,km^{-1}\,lane^{-1}}$	$\mathrm{km\,h^{-1}}$	-	$\mathrm{veh\,h^{-1}}$
MPC	Symbol	M	N_{p}	N_{c}	$w_{\max}$	$v_{\max}$
	Value	6	24	3	50	132.6
	Dimension	-	-	-	veh	$\mathrm{km\,h^{-1}}$
	Symbol	ρ_{sp}	v_{sp}			
	Value	23.45	132.6			
	Dimension	$\mathrm{veh\,km^{-1}\,lane^{-1}}$	$\mathrm{km\,h^{-1}}$			
RL	Symbol	c_{T}	c_{V}	c_{C}	batch size	γ
	Value	5	1600	5	5 episodes	0.98
	Symbol	α	α decay	$\Delta\boldsymbol{\theta}_{\{\mathrm{lb,ub}\}}$	$\sigma_{\boldsymbol{q}}$	ε
	Value	0.925	0.925	$\{-0.3\boldsymbol{\theta}, 0.3\boldsymbol{\theta}\}$	0.025	0.5

and two other non-learning solutions: the classical non-learning MPC formulation (6.13) [88], as well as the well-known local ramp metering PI-ALINEA [157, 217]. To promote a fair comparison, all of the tested controllers suffer from the same inexact knowledge of the traffic parameters.

The relevant hyper-parameters for DDPG are taken from [194]. The PI-ALINEA controller is equipped with a queue management strategy to take into account the queue length constraint [189], and its gains are fine-tuned to this specific task via Bayesian optimisation.

6.5.3. Results

We evaluate the performance of all the aforementioned methods deployed on the traffic network environment and average the results over 15 simulations with different seeds to iron out randomness due to, e.g., exploration. Figures that are shown and discussed next report the average results; however, 95% confidence intervals are shown only for our proposed method to reduce visual clutter.

Figure 6.3 shows, for each method, the total RL cost (6.30) incurred in each episode, i.e., $\sum_{k=1}^{T_{\text{fin}}/T} L(\boldsymbol{x}_k, \boldsymbol{u}_k)$, subdivided in its three contributions, and how these evolve during learning. Due to the untuned objective weights and the 30% mismatches in the predictive model's parameters, the initial MPC-based RL controller is poorly performing and cannot reliably avoid congestion and constraint violations, as can be seen from the costs in the first episodes. However, despite these initial shortcomings, it can be noticed that in the next 20 episodes, by exclusively leveraging the observed transitions via Q-learning, the controller is able to achieve a substantial improvement of the Total-Time-Spent cost, i.e., the time spent by vehicles in the network on average, from around 1100 - 700 veh h (or a 35% reduction). At the same time, as constraint violations of $w_{\max}$ in O_2 induce the most severe cost realisation by several orders of magnitude, one can see that the agent is even quicker in learning to avoid such a constraint-violating behaviour within the first 5 episodes, despite the fact it has no knowledge of the exact traffic dynamics parameters.

This phenomenon can be further appreciated in Figure 6.4, which shows the progression of the average queue in the on-ramp O_2 with respect to its constraint, as the parametrised MPC scheme gets better and better at yielding a policy with less or no constraint violation. It must be noted here that, despite the non-stationary nature of the traffic environment, the agent is able to learn a policy that is robust to the variability of the randomly generated demand and congestion scenarios and, therefore, is capable of avoiding all constraint violations.

Further analysis of the results indicates that, while learning to satisfy the constraint on the on-ramp O_2 is directly beneficial to the reduction of cost related to violations, it also indirectly fosters better TTS, since longer queues in O_2 proportionally relate to a higher TTS. Nonetheless, improvements to the violation and TTS contributions seem to occur at the expense of the third contribution, i.e., variability of the control action, which clearly is on the rise during learning for the MPC-based RL agent. However, in the overall context of learning, one must notice that at convergence the sacrifice in control variability (of around 60 points) allows the agent to gain a larger improvement in TTS and to achieve zero constraint violations (which are indeed the largest contributors to the cost of the RL agent). Furthermore, as the learning proceeds past the 16^{th} episode

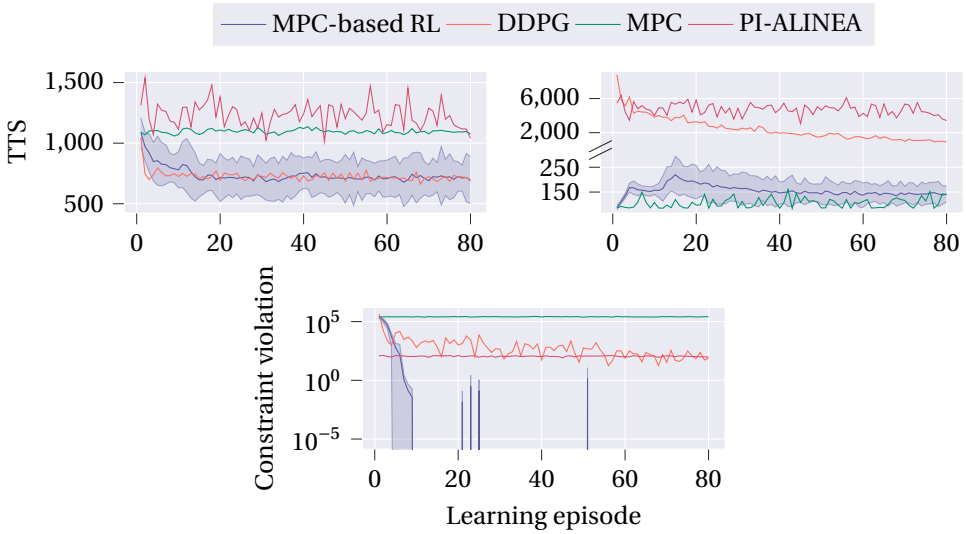

Figure 6.3.: Evolution of the three contributions to the RL cost (6.30) during the learning process, namely, in clockwise order, the Total-Time-Spent (TTS), variability of the control action, and violation of the maximum queue constraint on O_2

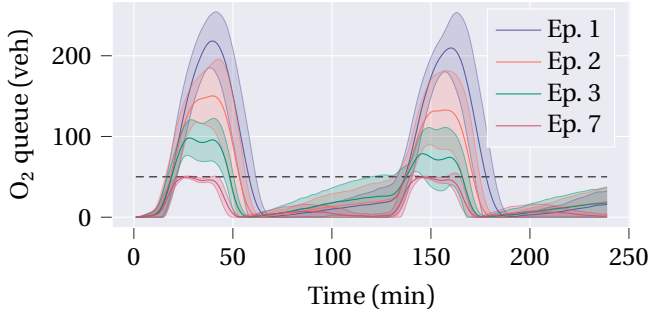

Figure 6.4.: On average, the policy $\pi_{\boldsymbol{\theta}}$ gets better at avoiding constraint violations as it learns (the dashed line represents the threshold $w_{\max}$ imposed on the queue on the on-ramp O_2).

and the other two costs have settled, the agent is also able to achieve further reduction of the cost associated to the control action variability.

The results from the other methods appear to corroborate the usefulness of data-driven adaptivity in such a context. The non-learning MPC formulation is stuck at the initial poor performance of its RL-enhanced counterpart, with high TTS and constraint violations. On the contrary, thanks to its queue management strategy, PI-ALINEA is able to curb violations, but at the expense of a much higher control variability (since

the strategy frequently requires to switch control action abruptly) and higher TTS. The DDPG agent follows a similar trend to our method in quickly learning to reduce TTS. However, variability and violations are decreasing at a substantially lower rate. This is not unexpected, as the NN approximation of this agent has significantly more learnable parameters (namely, 293 380 vs. 53) and is thus less data-efficient and requires more episodes to establish convergence.

As discussed in Section 6.1, an advantage of the proposed MPC-RL approach over DRL is that its learning process can be more readily inspected and explained via the evolution of its parametrisation $\boldsymbol{\theta}$, reported in Figure 6.5. In particular

- the evolution of θ_{T} and θ_{V} agrees with the observations made in fig. 6.3, that is, the agent learns to favour the TTS cost over the control action variability, and thus increases the weight of the former at the expense of the latter
- the parameter $\tilde{a}$, which the METANET model is known to be very sensitive to, develops at first towards its true value a, but then convergences to a slightly lower value, resulting in a less pronounced, less aggressive equilibrium speed V_{e} (6.4), i.e., favouring predictions of lower speeds at lower densities, and higher speeds at higher densities
- $\tilde{\rho}_{\mathrm{crit}}$ grows in the opposite direction of its true value ρ_{crit}, thus settling for a prediction model that is pessimistic towards congestion, i.e., it tends to overestimate congestion scenarios, as well as resulting in a tighter constraint h_o^1 (6.21), which in turn restricts the control action further.

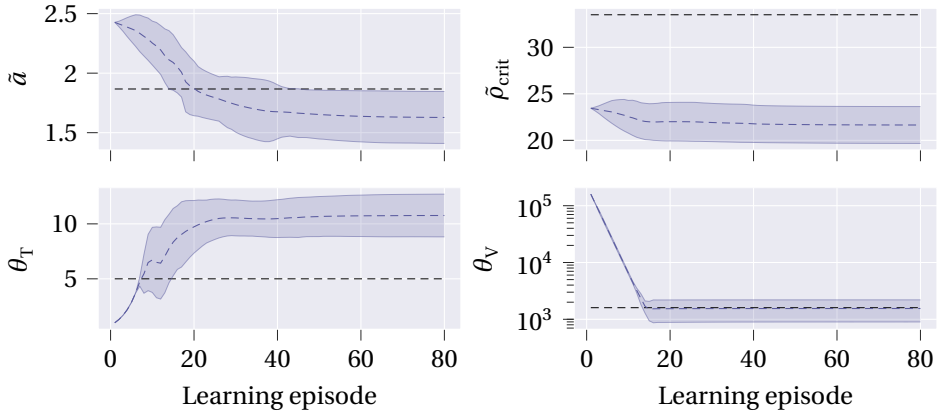

Figure 6.5.: Evolution of a subset of $\boldsymbol{\theta}$ during the learning process (the dashed grey lines represent, in the top plots, the true values of the parameters a and ρ_{crit}, and, in the bottom ones, the constants c_{T} and c_{V})

Figure 6.6 depicts the TD error during the learning process, whose moving average converges to smaller and smaller values as the episodes increase. This is a good indication that the MPC scheme (6.32), with its parametrisation $\boldsymbol{\theta}$, is able to provide a reliable

approximation $Q_{\boldsymbol{\theta}}$ of the true unknown action-value function (and, indirectly, of the optimal policy). Yet, it is important to note that the convergence of the TD error, while consistently diminishing, does not reach an absolute zero. This observation can be attributed to factors such as imperfect parametrisation and the inherent stochasticity present in the environment, which fluctuates in its demands, and thus renders the prediction of the value functions more challenging. Besides these issues, another challenge related to the TD error is its sparsity. Specifically, the instantaneous TD signal remains low and largely uninformative for most of the episode, only to spike significantly during congestion events. This sparsity can lead to inefficient updates if not properly addressed. Therefore, we advocate that the use of experience replay and batch updates is essential to smooth the learning process and mitigate the risk of destabilisation due to abrupt parameter adjustments.

Figure 6.6.: Moving average of the TD error during the learning process (window length = 239, i.e., number of transitions per episode)

Finally, Figure 6.7 reports the evolution of the traffic quantities of the three segments (i.e., density ρ, speed v, and flow q) that make up the network. Interestingly, it can be noticed how, compared to the initial episode, the final controller is able to better prevent the congestion in the last segment from propagating backwards through the network, and from causing speed drops in the first segment.

6.6. Conclusions

In this chapter, we have proposed a novel learning-based and model-based approach to the ramp metering problem that combines MPC and RL. The two frameworks are integrated in such a way as to promote the strengths of each while countering the disadvantages by exploiting the parametrised MPC scheme as a function approximation of the action-value function and leveraging RL to adjust the parametrisation based on observed data to improve closed-loop performance. Even with wrong model parameters and a poorly tuned initial controller, the proposed methodology shows a remarkable ability in learning to improve traffic control performance and satisfy constraints in an automatic, data-driven fashion.

Future work will focus on 1) the validation of the proposed methodology with microscopic traffic model simulators, 2) the study of formal guarantees on stability and

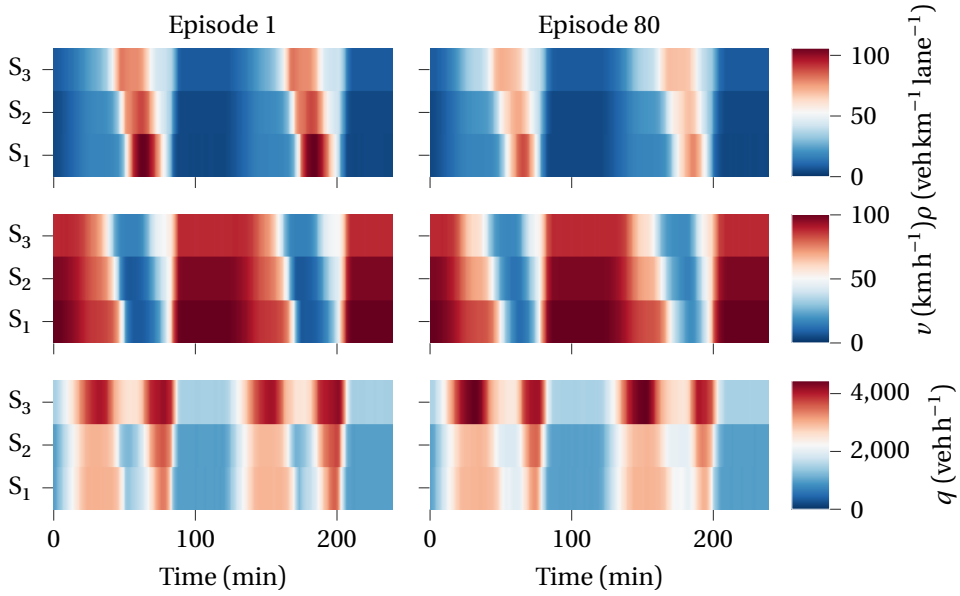

Figure 6.7.: Differences in traffic quantities of the network's three segments between the first episode and the last episode of the learning process, averaged across the 15 simulation runs

recursive feasibility for the proposed approach during learning and at convergence, 3) more complex parametrisations of the MPC scheme, e.g., with neural networks, in order to better address the nonlinear nature of the METANET framework and to better capture the shape of the true action-value function, as well as 4) extending the current approach by integrating variable speed limits (VSLs) with ramp metering, a coordinated control strategy that has been shown to achieve better results than ramp metering alone, especially in high demand/density regimes.

6.A. Appendix

We provide here additional insights on our proposed methodology and the numerical results. First, we provide a short investigation on the effects of different parametrisations of the MPC scheme (6.32). Then, we report the evolution of all the parameters $\boldsymbol{\theta}$ during the learning process implemented in Section 6.5.

6.A.1. Different MPC parametrisations

Different parametrisations $\boldsymbol{\theta}$ of the MPC scheme (6.32) have been tested in simulations. In particular, we have considered the following variants:

- learning a combination of the dynamics parameters $\tilde{a}$, $\tilde{v}_{\text{free}}$, and $\tilde{\rho}_{\text{crit}}$ (including the empty set, i.e., learning none of these parameters)
- employing the learnable dynamics parameters $\tilde{v}_{\text{free}}$ and $\tilde{\rho}_{\text{crit}}$ also as tracking setpoints of the stage cost (6.34) and terminal cost (6.35), i.e., $v_{\text{sp}} = \tilde{v}_{\text{free}}$, $\rho_{\text{sp}} = \tilde{\rho}_{\text{crit}}$.

The rationale behind trying out these various combinations is that, while having a richer parametrisation can potentially help in fitting more complex value functions, a drawback is that the learning task becomes much more complex and more likely to converge to a very suboptimal local minimum. Moreover, the sensitivity of the RL solution to the parametrisation can vary significantly from parameter to parameter, so learning some of them may end up in a less stable process than others. As is the case with most function approximators, the choice of parametrisation relies mostly on prior and/or expert knowledge (such as, in our case, the sensitivity analysis of the METANET model w.r.t. its parameters) and is mostly an iterative procedure (both for MPC and other function approximators), whose difficulty is attributable to the RL algorithm itself rather than to the proposed control scheme. However, meta-learning methods can be found in the literature to optimise over the selection of this and other hyper-parameters via, e.g., Bayesian Optimisation [225] and bilevel optimisation [66].

Figure 6.8 shows the influence on the performance of some of the combinations we tested. These tests were carried out with a lower variability of the randomly generated profiles in an effort to filter out the impact of randomness on the performance. One can notice that, when neither the dynamics parameters nor the tracking setpoints are learnt, the resulting controller converges to a solid performance. Adding $\tilde{a}$ to the set of learnable parameters seems to boost the performance even further, which can be attributed to the fact that, among the various parameters, $\tilde{a}$ is the one the METANET model is most sensitive to. Further testing indicates that learning also $\tilde{\rho}_{\text{crit}}$ achieves an additional performance improvement, while learning $\tilde{v}_{\text{free}}$ does not help. The last empirical finding is that making the tracking setpoints learnable, despite increasing the number of degrees of freedom of the parametrisation, does not bring any improvement.

6.A.2. Evolution of parametrisation

Interested readers can find in Figure 6.9 the evolution, during the whole learning process, of the whole parametrisation $\boldsymbol{\theta}$ of the MPC scheme (6.32), as detailed in Section 6.4.2 and simulated in Section 6.5.

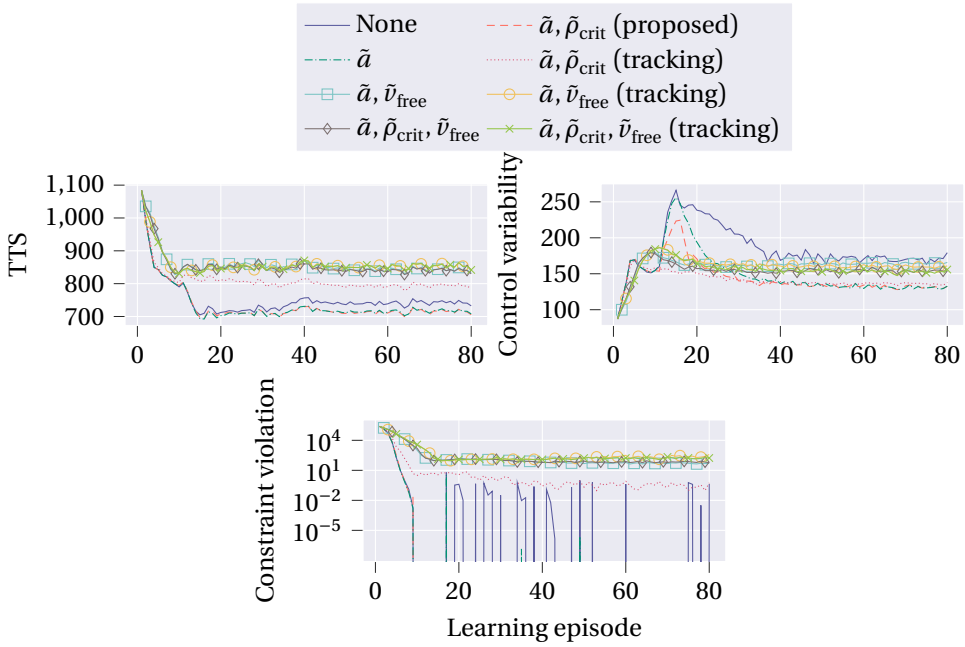

Figure 6.8.: Evolution of the RL cost contributions for various different parametrisations $\boldsymbol{\theta}$. The legend lists for each entry the dynamics parameters that were allowed to be learnt in the corresponding simulation, including also the parametrisation that we proposed in this work, i.e., *(proposed)*. If either $\tilde{\rho}_{\text{crit}}$, $\tilde{v}_{\text{free}}$ or both were also employed as tracking setpoints, the entry is marked accordingly, i.e., *(tracking)*. To avoid visual clutter, only the average over the 15 simulation runs is shown for each parametrisation.

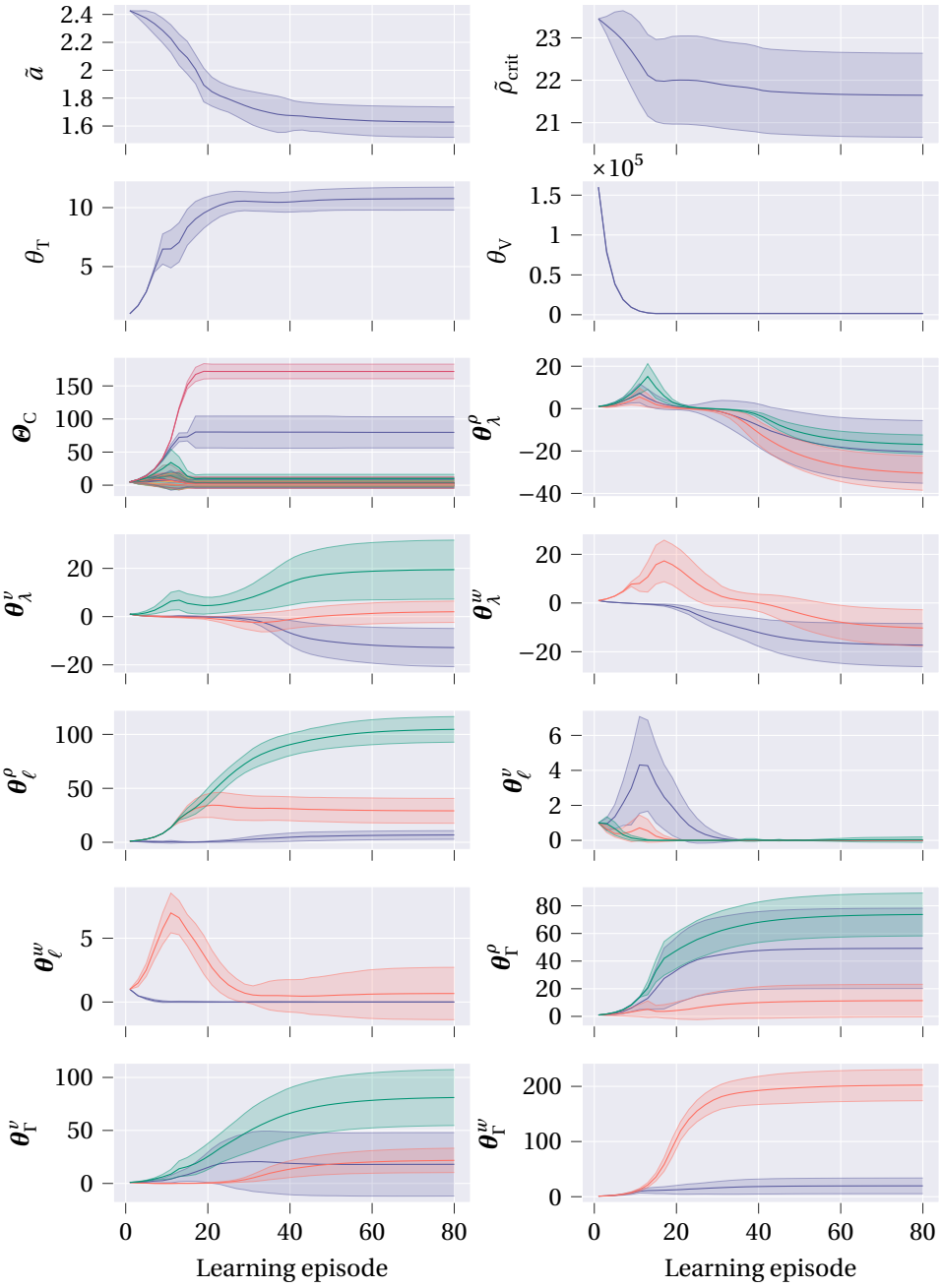

Figure 6.9.: Average evolution of the parametrisation $\boldsymbol{\theta}$ during the learning process across the 15 simulation runs. For those parameters that are vector quantities, one colour is associated with each entry.

II

Model-based reinforcement learning for optimisation

7

Nonmyopic global optimisation via approximate dynamic programming

Global optimisation techniques can be used to optimise expensive-to-evaluate black-box functions without gradient information. Bayesian optimisation, one of the most well-known techniques, typically employs Gaussian processes as surrogate models, leveraging their probabilistic nature to balance exploration and exploitation. However, Gaussian processes become computationally prohibitive in high-dimensional spaces. Recent alternatives, based on inverse distance weighting (IDW) and radial basis functions (RBFs), offer competitive, computationally lighter solutions. Despite their efficiency, both traditional global and Bayesian optimisation strategies suffer from the myopic nature of their acquisition functions, which focus solely on immediate improvement neglecting future implications of the sequential decision making process. Nonmyopic acquisition functions devised for the Bayesian setting have shown promise in improving long-term performance. Yet, their use in deterministic strategies with IDW and RBF remains unexplored. In this chapter, we introduce novel nonmyopic acquisition strategies tailored to IDW- and RBF-based global optimisation. Specifically, we develop dynamic programming-based paradigms, including rollout and multi-step scenario-based optimisation schemes, to enable lookahead acquisition. These methods optimise a sequence of query points over a horizon (instead of only at the next step) by predicting the evolution of the surrogate model, inherently managing the exploration-exploitation trade-off in a systematic way via optimisation techniques. The proposed approach represents a significant advance in extending nonmyopic acquisition principles, previously confined to Bayesian optimisation, to the deterministic framework. Empirical results on synthetic and hyperparameter tuning benchmark problems, as well as on a data-driven predictive control application, demonstrate that these nonmyopic methods outperform conventional myopic approaches, leading to faster and more robust convergence.

This chapter is based on [5].

7.1. Introduction

Global optimisation (GO) is a class of optimisation techniques designed for finding the global minimum (or maximum) of an objective function. These techniques are especially relevant when the objective is, e.g., a black-box and expensive-to-evaluate function whose gradient information cannot be computed, or it is highly irregular and nonconvex [16, 19]. These aspects make GO strategies appealing and useful in domains where derivative-based methods struggle or fail, e.g., when the objective function is non-smooth or its gradient too costly to evaluate. Examples of applications include hyperparameter selection in machine learning algorithms [186] and automatic tuning of closed-loop controllers [188]. The primary goal of GO is to minimise the number of function evaluations while ensuring that the algorithm escapes local optima and converges to a global solution or, at least, to a near-optimal minimum. To this end, the majority of GO strategies consist of two main ingredients: a surrogate model that approximates the underlying unknown objective function and an acquisition function, usually in the form of a cheaper-to-solve optimisation problem, that guides the sampling of new points to be observed and governs the fundamental trade-off between exploration (searching for better solutions in unexplored regions) and exploitation (focusing on convergence in the current most promising region). An iterative algorithm integrates these two components by first updating the surrogate model with the latest information and then solving the acquisition for the next query point, and it runs until a stopping criterion is met, e.g., a maximum number of iterations or satisfactory convergence to a local or global minimum.

One of the most prominent techniques in GO is Bayesian optimisation (BO), which takes a probabilistic approach to the problem [72]. In BO, evaluations of the unknown function at any query point are assumed to be random variables, and the uncertainty of the objective over the unexplored input space is modelled as a probability distribution. The key to BO is the construction of a prior belief in the form of a probabilistic model, typically based on Gaussian processes (GPs) [162], which provides a stochastic estimate of the function at any point in the input space. As the noisy objective evaluations are iteratively collected, the prior is updated into a posterior which, in turn, is used to construct the acquisition function whose solution determines the next query point. While BO has been widely successful, the reliance on GP models comes with a computational burden, particularly in high-dimensional spaces.

Recently, alternative optimisation strategies that employ inverse distance weighting (IDW) [100] and radial basis functions (RBFs) [139] have emerged as competitive approaches. Compared to BO, surrogate models based on IDW and RBF functions offer deterministic approximations of the black-box function and of the uncertainty between the surrogate and the unknown objective. Since no posterior computation is needed, an immediate benefit of this approach is that the resulting algorithms are lightweight and computationally efficient while still retaining a performance that is comparable to BO [16, 50, 100]. Previous work has laid the theoretical foundations and shown their efficacy [83, 164]; more recently, their use has been successfully extended to other domains, such as preference learning [19] and active learning [15].

However, despite the success of these traditional GO and BO strategies, much of the research has highlighted a crucial weak point: the myopic nature of its acquisition

functions. Most standard acquisition functions, such as expected improvement [99] and upper confidence bound [190], make decisions based solely on the current state of the surrogate model without considering the long-term implications of future query-evaluation data. Even a more sophisticated stratagem like the knowledge-gradient policy suffers from this drawback, as it considers the expected increment in the value of information for only one step ahead [67]. Such short-sighted acquisition functions are likely to lead to suboptimal solutions, especially in problems where the exploration-exploitation balance is critical for good convergence or the number of evaluations is limited.

Addressing the myopia of acquisition functions is thus a parallel and vibrant thread of the current research in the field. In the context of BO, nonmyopic acquisition functions have been proposed that leverage concepts from the field of dynamic programming (DP) [22]. In particular, [115] was among the first to propose the use of rollout, a well-known approximate DP algorithm [20, 24], to compute a querying strategy that can predict across a horizon of multiple iterations the evolution of the surrogate model as new evaluations of the unknown objective are observed. Briefly, it employs a base greedy acquisition function (e.g., expected improvement) as reward, and then uses a rollout scheme to find the sequence of query points that maximises the accumulation of these rewards along the prediction horizon. Whereas the base function is greedy, i.e., it selects the next query point to maximise the immediate reward without regard to the remaining successive iteration, the resulting rollout strategy is able to take into consideration how hypothetical future information affects the sequential decision making process, thus acting nonmyopically. This ultimately results in *multi-step lookahead* strategies that are often more performing and converge faster, and that can inherently balance the exploration-exploitation trade-off over the horizon in a systematic and structured way via optimisation techniques (though the extent of these benefits can vary from one application to another).

More recent literature has sprouted from the work of [115]. Advanced nonmyopic strategies for BO have been devised that, e.g., improve the convergence of statistical estimates [116], employ more complex and scalable optimisation formulations [96, 97], adjust the length of the prediction horizon automatically [232], or tackle cost-constrained optimisation problems [117]. Also the formal problem statement and theory behind these methodologies have been enhanced [23]. However, while it is apparent that this line of research is gaining momentum in the Bayesian framework, its application to deterministic GO techniques, especially the ones based on IDW and RBF, remains relatively unexplored.

In this chapter, we propose to address this gap in the current literature by leveraging the successes of IDW- and RBF-based optimisation strategies and of nonmyopic BO strategies. We propose to employ DP methodologies to craft multi-step lookahead acquisition problems for deterministic GO strategies using IDW and RBF. Specifically:

1. We adapt the current IDW- and RBF-based GO optimisation procedure to the DP setting. This includes formulating the dynamics of these surrogate models, which describe how these approximation models evolve when new data is made available, and devising a proper reward signal. Furthermore, since (unlike GPs) these surrogate models do not generate a posterior, we propose a heuristic posterior

that allows for sampling in the DP algorithms.

2. We propose two well-known algorithms, rollout and multi-step scenario-based optimisation, to solve our nonmyopic acquisition problems based on IDW and RBF via different sampling strategies.

3. We validate the proposed nonmyopic schemes on several numerical benchmark cases of various dimensionality and difficulty, including synthetic and real-world hyperparameter tuning problems, as well as on a data-driven tuning experiment of a predictive controller based on closed-loop performance. The simulation results empirically confirm that the proposed methods can outperform the vanilla myopic GO strategies.

The rest of the chapter is organised as follows. Section 7.2 provides background information on myopic GO strategies and DP algorithms. The proposed nonmyopic methods are then presented and explained in Section 7.3. Section 7.4 provides two numerical experiments to validate these algorithms. Finally, Section 7.5 concludes the chapter with some final remarks.

7.2. Background

In this section, the foundations of the two main components of this chapter, i.e., global optimisation and dynamic programming, are presented.

Notation: In the context of nonmyopic/predictive algorithms, given the information available at time step (or iteration) k, quantities predicted at time step (or iteration) $i \geq k$ are denoted as, e.g., $\hat{y}_{i|k}$.

7.2.1. Global optimisation

Given the function $f : \mathbb{R}^n \to \mathbb{R}$, we consider the following optimisation problem

$$x^\star \in \arg\min_{x \in \mathscr{X}} \quad f(x), \tag{7.1}$$

where x is an n-dimensional vector over the compact set $\mathscr{X} \subseteq \mathbb{R}^n$. In particular, we assume that $f(x)$ is expensive to evaluate and no gradient information is available, while $\mathscr{X}$ encodes an arbitrarily constrained but known space. The goal is to estimate a solution $x^\star$ of (7.1) using an optimal sequential selection of observations based on feedback from preceding observations. As mentioned in the Introduction, in GO this problem is tackled by modelling the objective function f with a surrogate model. In line with [16], this chapter focuses on IDW [100] and RBF [139] functions as surrogates of the black-box objective. To construct these surrogate models of the unknown function f, which will be referred to as $\hat{f}_k$, let us consider iteration $k \in \{N_0, \dots, N\}$ of the sequential estimation, with N the maximum number of iterations and $N_0 \in [1, N)$ the number of initial warm-up iterations.[1] Assume $F_k = \begin{bmatrix} f_1 & \dots & f_k \end{bmatrix}^\top \in \mathbb{R}^k$ gathers the noiseless measurements

[1] These N_0 iterations are often used to prime the regressor on an initial dataset of N_0 data points before starting the optimisation algorithm. This dataset can include a priori knowledge on $f(x)$, but more often is generated via randomly sampled evaluations of the objective. For the sake of simplicity, assume here $N_0 = 1$.

$f_i = f(x_i)$ collected so far at the corresponding points $X_k = \begin{bmatrix} x_1 & \dots & x_k \end{bmatrix}^\top \in \mathbb{R}^{k \times n}$, with $x_i \neq x_j$, $\forall i, j \in \{1, \dots, k\}$, $i \neq j$. These query-observation pairs constitute the dataset $\mathscr{D}_k = \{(x_i, f_i) \mid i = 1, \dots, k\}$ at the current iteration k. Naturally, these can also be defined recursively:

$$\mathscr{D}_0 = \emptyset, \quad \mathscr{D}_{k+1} = \mathscr{D}_k \cup \{(x_{k+1}, f_{k+1})\}. \tag{7.2}$$

Given a generic query point $x \in \mathbb{R}^n$, IDW interpolation leverages the observed data to formulate, at current iteration k, the non-parametric approximation $\hat{f}_k : \mathbb{R}^n \to \mathbb{R}$

$$\hat{f}_k(x) = \sum_{i=1}^{k} f_i v_{k,i}(x) = F_k^\top V_k(x), \tag{7.3}$$

where

$$v_{k,i}(x) = \frac{w_i(x)}{\sum_{j=1}^{k} w_j(x)}, \qquad i = 1, \dots, k, \tag{7.4}$$

$$w_i(x) = \frac{1}{\max\{d^2(x, x_i), \delta\}}, \qquad i = 1, \dots, k, \tag{7.5}$$

and

$$V_k(x) = \begin{bmatrix} v_{k,1}(x) & \dots & v_{k,k}(x) \end{bmatrix}^\top \in \mathbb{R}^k. \tag{7.6}$$

We use $v_{k,i}$ to refer to the i-th element of V_k at iteration k. In (7.5), we indicate with $d^2 : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ the squared Euclidean distance $d^2(x, x_i) = \|x - x_i\|_2^2$ and $\delta > 0$ is a small scalar value that avoids division by excessively small numbers when $x \to x_i$. By inspection of (7.3), $v_{k,i}(x)$ can be interpreted as weighting the contribution of each observation f_i based on the distance of the query point x from x_i, hence the name IDW. Such formulation enjoys some favourable properties [16, Lemma 1], but other choices of w_i are available, e.g., $w_i(x) = \frac{e^{-d^2(x,x_i)}}{d^2(x,x_i)}$ is proposed in [100] to intensify the rate of decay of each contribution with increasing distances.

Likewise, RBFs can be used to build a parametric surrogate function as

$$\hat{f}_k(x) = \sum_{i=1}^{k} \beta_{k,i} \phi\big(\varepsilon d(x, x_i)\big) = B_k^\top \Phi_k(x), \tag{7.7}$$

with

$$B_k = \begin{bmatrix} \beta_{k,1} & \dots & \beta_{k,k} \end{bmatrix}^\top \in \mathbb{R}^k, \tag{7.8}$$

$$\Phi_k(x) = \begin{bmatrix} \phi\big(\varepsilon d(x, x_1)\big) & \dots & \phi\big(\varepsilon d(x, x_k)\big) \end{bmatrix}^\top \in \mathbb{R}^k, \tag{7.9}$$

where $\varepsilon > 0$ is a scalar parameter, $\phi : \mathbb{R} \to \mathbb{R}$ is an RBF, and $\beta_{k,i}$ is the i-th coefficient to be determined by solving the linear system

$$F_k = M_k B_k, \tag{7.10}$$

with the symmetric interpolation matrix

$$M_k = \begin{bmatrix} \Phi_k(x_1) & \dots & \Phi_k(x_k) \end{bmatrix} \in \mathbb{R}^{k \times k}. \tag{7.11}$$

For the sake of conciseness, here we assume M_k to be non-singular, so that a solution to (7.10) always exists. Note that methods to circumvent this issue are available, e.g., the singular value decomposition is commonly employed to invert M_k while discarding singular values below a small threshold [16]. RBFs have widely been used in function approximation schemes, e.g., in neural networks [125], since Φ_k offers a suitable basis for the function space from $\mathbb{R}^n$ to $\mathbb{R}$. In particular, in this chapter, the inverse quadratic basis function $\phi(y) = \frac{1}{1+y^2}$ is employed (see [16] for a collection of some of the other most popular choices for ϕ).

The ensuing GO algorithm is carried out as follows. It starts at $k = N_0$ with the initial dataset $\mathscr{D}_{N_0}$ of query and observation pairs. At each iteration k, regression is performed on the current dataset $\mathscr{D}_k$ to obtain a surrogate model $\hat{f}_k$, which allows for the prediction of the unknown objective f at any query point $x \in \mathscr{X}$. Next, the algorithm selects a new target point x_{k+1} for evaluation, leading to observation f_{k+1}. The new pair (x_{k+1}, f_{k+1}) is then added to the dataset $\mathscr{D}_k$, and a new iteration begins. The algorithm continues till the maximum number of iterations N is reached (or another termination condition is met), at which point the approximate solution $x^\star \approx \max_{i=1,\dots,N+1} x_i$ to the original problem (7.1) is returned.

The selection of the next query point is encoded by the auxiliary problem

$$x_{k+1} \in \arg\min_{x \in \mathscr{X}} \quad \Lambda(\mathscr{D}_k, x), \tag{7.12}$$

where $\Lambda : (\mathbb{R}^n \times \mathbb{R})^k \times \mathbb{R}^n \to \mathbb{R}$ is often referred to as the *acquisition function*. Compared to (7.1), problem (7.12) is assumed to be much cheaper and faster to solve, and does not involve evaluating the expensive objective function f but rather only depends on the current dataset $\mathscr{D}_k$ and surrogate model $\hat{f}_k$. Thus, the acquisition function is assigned to the paramount task of guiding the optimisation towards an approximate global optimum while balancing the trade-off between exploration and exploitation.

Commonly, acquisition functions in BO exploit the posterior covariance associated with the GP approximator to take exploration into account naturally. However, since this chapter focuses on deterministic predictors, we borrow the acquisition function suggested and described in [16, 100] that is crafted ad hoc for IDWs and RBFs to induce the exploration-exploitation trade-off. For this purpose, let us define two notions, namely the *variance function* and *distance function*. In any given iteration k, define the *variance function* $\sigma_k : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ as

$$\sigma_k(x) = \sqrt{\sum_{i=1}^{k} v_{k,i}(x) \left(\hat{f}_k(x) - f_i\right)^2}. \tag{7.13}$$

This function is able to induce exploratory behaviour by providing data-driven confidence intervals around the surrogate model. As a matter of fact, $\lim_{\delta \to 0} \sigma_k(x_i) = 0$, $i = 1, \dots, k$ as no uncertainty affects points x_i where f has been evaluated exactly. Additionally, the *distance function* $\zeta_k : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, defined by

$$\zeta_k(x) = \frac{2}{\pi} \arctan\left(\frac{1}{\sum_{i=1}^{k} w_i(x)}\right), \tag{7.14}$$

approaches zero at sampled points (similarly to σ_k) but grows in between, pushing the GO algorithm towards unexplored regions. Given scalars $\lambda, \mu \geq 0$, the current dataset $\mathscr{D}_k$ and surrogate model $\hat{f}$, the overall acquisition function Λ is then

$$\Lambda(\mathscr{D}_k, x) = \hat{f}_k(x) - \lambda\sigma_k(x) - \mu\Delta_{F_k}\zeta_k(x), \tag{7.15}$$

where $\Delta_{F_k} = \max_{i=1,\dots,k} f_i - \min_{i=1,\dots,k} f_i$ is the range of observed samples F_k. As discussed in more detail in [16], the term $\hat{f}_k$ accounts for the exploitation of the samples F_k collected so far, whereas the other two terms encourage exploration: σ_k promotes high-uncertainty regions, and ζ_k unexplored ones.

However, similarly to most of the other classical acquisition functions in BO, the aforementioned acquisition function suffers from myopia, i.e., it is oblivious to the number of objective function evaluations left and does not take into account the evolution of the surrogate model due to future observations. Therefore, being essentially a one-step lookahead function, this acquisition function leads to greedy optimisation strategies. Fortunately, next is discussed dynamic programming, a tool that makes it possible to devise multi-step lookahead strategies that perform better and can balance the exploration-exploitation trade-off in an optimisation setting.

7.2.2. Dynamic programming and its approximated schemes

Dynamic programming [22] is a well-known method that provides us with a mathematical formulation to tackle optimal decision-making and control problems, in which we seek a policy (i.e., a sequence of control decisions) to control a (possibly stochastic) dynamical system in such a way as to minimise a given objective function. While not immediately straightforward, global optimal sequential estimation problems can also be cast as such, allowing us to leverage DP to address them formally in a nonmyopic way. For the sake of compactness, we limit ourselves to an introduction to DP and some of its approximated schemes.

Dynamic programming

We consider a stochastic discrete-time system described by

$$s_{k+1} = \mathscr{F}_k(s_k, a_k, w_k), \tag{7.16}$$

where $s_k \in \mathscr{S}_k$ is the state at time step $k \in \{1, \dots, N\}$, $a_k \in \mathscr{A}_k$ the control action, and w_k is a random process disturbance with probability distribution $\mathbb{P}_{w_k}$ with support $\mathscr{W}_k$. The function $\mathscr{F}_k : \mathscr{S}_k \times \mathscr{A}_k \times \mathscr{W}_k \to \mathscr{S}_{k+1}$ represents the (possibly nonlinear) dynamics that govern the underlying state transitions. A decision rule $\pi_k : \mathscr{S}_k \to \mathscr{A}_k$ is a mapping from states to actions, and a policy $\pi = \pi_1, \dots, \pi_N$ is a sequence of rules. Moreover, let a stage-reward function $R_k : \mathscr{S}_k \times \mathscr{A}_k \to \mathbb{R}$ quantify the benefit of applying control action a_k when the system is in state s_k.

Given these building blocks, the expected total return (or value function) over a finite

horizon N starting from time step $k = 1$ with state s_1 and policy π is given by[2]

$$J_1^{\pi}(s_1) = \mathbb{E}_{w_1,\dots,w_{N-1}}\left[\sum_{k=1}^{N} R_k\big(s_k, \pi_k(s_k)\big)\right], \tag{7.17}$$

and the goal of DP is to find the optimal policy $\pi^\star$ in the space of admissible policies Π that maximises this long-term return

$$J_1^\star(s_1) = J_1^{\pi^\star}(s_1) = \max_{\pi\in\Pi} \quad J_1^{\pi}(s_1), \tag{7.18}$$

where, for simplicity, we assume the initial state s_1 to be exactly measurable and $\mathscr{S}_1 = \{s_1\}$. Following Bellman's principle of optimality, the problem can be solved using the DP recursive algorithm [22]

$$J_k^\star(s_k) = \max_{a_k\in\mathscr{A}_k} R_k(s_k, a_k) + \mathbb{E}_{w_k}\Big[J_{k+1}^\star\big(\mathscr{F}_k(s_k, a_k, w_k)\big)\Big], \quad \forall s_k \in \mathscr{S}_k, \tag{7.19}$$

starting with $J_{N+1}^\star(s_{N+1}) = T(s_{N+1})$, and by moving backwards from $k = N$ to $k = 1$. The optimal return is obtained at the last step as $J_1^\star(s_1)$, while the optimal policy can be found going forward, from $k = 1$ to $k = N$, as one of the maximisers $a_k^\star = \pi^\star(s_k)$ of (7.19). The function $T : \mathscr{S}_{N+1} \to \mathbb{R}$ represents the terminal reward and, without loss of generality, is assumed to be zero in the rest of the chapter.

Unfortunately, solving the DP problem exactly as described above is often only viable for problems with finite state and action spaces, as DP algorithms are known to suffer from the curse of dimensionality, especially as N grows. The policy space Π may also be infinite-dimensional, contributing to rendering the problem intractable. To address this issue, one needs to resort to approximate dynamic programming (ADP). Next, we briefly introduce two ADP techniques from the literature.

Rollout

Among the ADP methodologies, rollout stands out as a simple yet reliable sequential decision strategy to tackle the optimal control problem [20, 24]. In short, starting from a base policy (e.g., a heuristic decided a priori by the user), rollout attempts to improve it via a one-step lookahead minimisation, thus emulating a single iteration of the policy iteration method. In mathematical terms, given the base policy $\mu = \mu_1, \dots, \mu_N$, for the current state s_k, the one-step lookahead rollout policy $\tilde{\mu}(s_k)$ can be computed as

$$\tilde{\mu}(s_k) \in \arg\max_{a_k\in\mathscr{A}_k} \quad R_k(s_k, a_k) + \mathbb{E}_{w_k}\Big[J_\mu\big(\mathscr{F}_k(s_k, a_k, w_k)\big)\Big]. \tag{7.20}$$

Though similar to (7.19) at first sight, (7.20) is meant to be solved online only for the current state s_k and does not involve backwards computations over the entire state space. Instead, it under-approximates the optimal cost-to-go $J_{k+1}^\star$ with the heuristic base policy cost J_μ, which is assumed much easier to compute since the base policy

[2] Note that a discount factor γ is often included in (7.17) to discount future rewards in comparison to immediate ones. Here, however, it is omitted for simplicity as its effect on performance is not clear in the context of nonmyopic optimisation [115].

is known. Under some assumptions, it can be shown that the rollout policy enjoys an improvement property, i.e., $J_{\tilde{\mu}}(s_k) \geq J_{\mu}(s_k)$, $\forall s_k \in \mathscr{S}_k$ [22, Section 6.4]. The expectation operator in (7.20) is often the most expensive term to compute and, unless it can be formulated in an analytical expression, it is often addressed via approximations. There exist different ways of doing so [22, 24], and some of the most popular include Certainty Equivalence approaches (in which case, under additional assumptions, the expectation is removed from the objective), Monte Carlo sampling-based strategies (one of which is discussed next in Section 7.2.2), and other cost-to-go approximations.

In this chapter, we employ a receding horizon control scheme, a variation of rollout formulation that uses a one-step minimisation base heuristic. Given a shorter horizon $H \leq N$, the scheme only takes into account the next $h = \min\{H, N-k+1\}$ future sequential decisions, i.e., from time step k to $k+h$, by solving the following stochastic optimal control problem:

$$\begin{aligned} \max_{\hat{a}_{k|k},\dots,\hat{a}_{k+h-1|k}} \quad & R_t\left(\hat{s}_{k|k}, \hat{a}_{k|k}\right) + \mathbb{E}_{w_k,\dots,w_{k+h-2}}\left[\sum_{t=k+1}^{k+h-1} R_t\left(\hat{s}_{t|k}, \hat{a}_{t|k}\right)\right] \\ \text{s.t.} \quad & \hat{s}_{k|k} = s_k, \\ & \hat{s}_{t+1|k} = \mathscr{F}_t\left(\hat{s}_{t|k}, \hat{a}_{t|k}, w_t\right), \quad t = k,\dots,k+h-1, \\ & w_t \sim \mathbb{P}_{w_t}, \qquad\qquad\qquad\quad t = k,\dots,k+h-1, \end{aligned} \tag{7.21}$$

where $\hat{s}_{k|k},\dots,\hat{s}_{k+h|k}$ and $\hat{a}_{k|k},\dots,\hat{a}_{k+h-1|k}$ are the predictive state and action trajectories based on the information available at k. Then, only the first optimal control action $a_k = \hat{a}^{\star}_{k|k}$ is applied to the real system, leading to a new state s_{k+1} at the next time step, at which point the procedure is repeated. In case the dynamics $\mathscr{F}_k$ are deterministic, computations for (7.20) and (7.21) simplify substantially as the evolution of the system is no longer aleatory (thus, there is no need to sample w_t), and the expectation can be elided from the objective. This formulation is heavily reminiscent of a single-shooting stochastic model predictive control (MPC) scheme, and interested readers are referred to [140, 163] for more details. In the numerical experiments in this chapter, to make the rollout problem (7.21) tractable and solve it, the stochastic dynamics and expectation operators are approximated by replacing the disturbance signal w_t with a sample drawn from P_{w_t} at each predicted time step t. However, as further detailed in the next section, a more general and holistic sampling-based technique can be formulated to solve the control problem despite the presence of stochasticities.

Remark 7.1. *Note that perfect knowledge of the dynamics $\mathscr{F}_k$ is cardinal to guarantee the improvement property so as to predict exactly how the model evolves at the next time step. However, in practice, benefits in employing rollout can still be observed when only approximate models are available, as is the case in GO and BO [23, 232].*

Multi-step tree-based optimisation

A paramount question is how to deal with the stochastic quantities in (7.21) in a non-deterministic case. A viable and tractable answer to it is to use sample-based techniques, which have been recently popularised in BO as well [97].

Let $m_t \in \mathbb{N}_{>0}$, $t = k, \dots, k+h-2$, be the number of samples of the disturbance to be drawn at the prediction time step t. The core idea is then to replace the stochasticities in (7.21) with their sampled counterparts. Starting from the given initial state s_k, m_k samples of the disturbance $\left\{w_k^{(1)}, \dots, w_k^{(m_k)}\right\}$ are taken from $\mathbb{P}_{w_k}$ and, with the application of the first action $\hat{a}_{k|k}$, equally many next predicted states $\left\{\hat{s}_{k+1|k}^{(1)}, \dots, \hat{s}_{k+1|k}^{(m_k)}\right\}$ are obtained. For this first time step, the stage cost requires no approximation as it is deterministically found as $R_k\left(s_k, \hat{a}_{k|k}\right)$. In the next time step, for each state $\hat{s}_{k+1|k}^{(i_k)}$, $i_k = 1, \dots, m_k$, a new set of m_{k+1} disturbances $\left\{w_{k+1}^{(i_k,1)}, \dots, w_{k+1}^{(i_k,m_{k+1})}\right\}$ is sampled according to $\mathbb{P}_{w_{k+1}}$, leading to approximating the next state evolution with the samples $\left\{\hat{s}_{k+2|k}^{(i_k,1)}, \dots, \hat{s}_{k+2|k}^{(i_k,m_{k+1})}\right\}$ after applying $\hat{a}_{k+1|k}$. The stage cost for $k+1$ is then approximated as $\frac{1}{m_k}\sum_{i_k=1}^{m_k} R_{k+1}\left(\hat{s}_{k+1|k}^{(i_k)}, \hat{a}_{k+1|k}\right)$. The procedure is so repeated along the predicted horizon, creating a scenario tree-based approximation of the original problem (7.21). The approach is visually summarised in Figure 7.1. The resulting optimisation scheme is

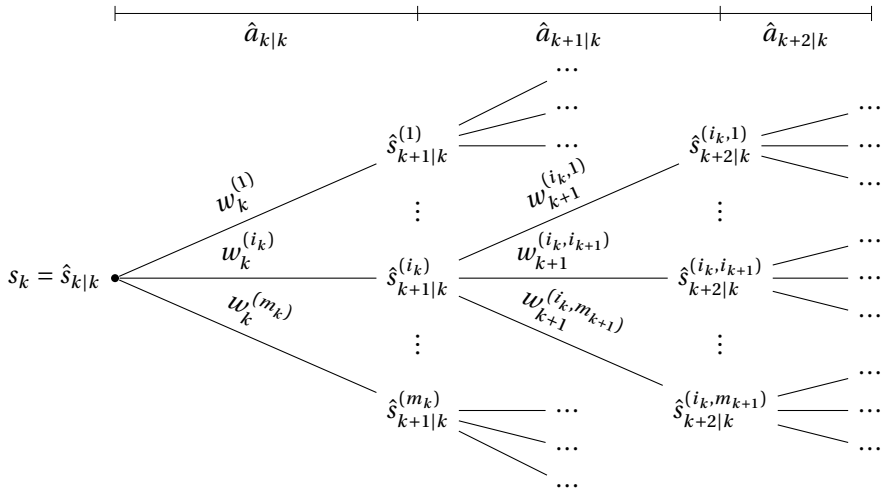

Figure 7.1.: Depiction of the tree structure induced by the multi-step sampling of the disturbances, whereas the actions to be optimised remain constant within the same stage.

$$
\begin{aligned}
\max_{\hat{a}_{k|k},\dots,\hat{a}_{k+h-1|k}} \quad & R_k\left(\hat{s}_{k|k}, \hat{a}_{k|k}\right) + \frac{1}{m_k}\sum_{i_k=1}^{m_k} R_{k+1}\left(\hat{s}_{k+1|k}^{(i_k)}, \hat{a}_{k+1|k}\right) + \dots \\
& + \frac{1}{\prod_{\ell=k}^{k+h-2} m_\ell}\sum_{i_k=1}^{m_k} \cdots \sum_{i_{k+h-2}=1}^{m_{k+h-2}} R_{k+h-1}\left(\hat{s}_{k+h-1|k}^{(i_k,\dots,i_{k+h-2})}, \hat{a}_{k+h-1|k}\right) \qquad (7.22) \\
\text{s.t.} \quad & \hat{s}_{k|k} = s_k, \\
& \hat{s}_{t+1|k}^{(i_k,\dots,i_{t-1},i_t)} = \mathscr{F}_t\left(\hat{s}_{t|k}^{(i_k,\dots,i_{t-1})}, \hat{a}_{t|k}, w_t^{(i_k,\dots,i_{t-1},i_t)}\right), \\
& \qquad\qquad i_t = 1, \dots, m_t,\ t = k, \dots, k+h-1.
\end{aligned}
$$

This formulation is advantageous because it circumvents the issue of stochastic dynamics and computations of expected values, and leaves us with a one-shot optimisation problem that does not involve nested maximisations and expectations, in contrast with (7.19). Under weak conditions, it can also be shown to lead to improvements with respect to a greedy approach [97]. On the other side, it is straightforward to notice that the approach lacks in scalability, especially when $m_t \gg 1$, since the number of scenarios in the tree grows exponentially with the horizon h, despite the number of primal variables growing only linearly.[3]

Similarly to the rollout approach (7.20), analogies to this sample-based approach can be found within the control community in the context of the MPC literature, e.g., [25, 172]. Moreover, notice that (7.22) can be seen as a generalisation of (7.20). In particular, when $m_t = 1$ and $w_t^{(1,\dots,1)} = \mathbb{E}[w_t]$, (7.22) regresses to a single-branch certainty equivalent solution to (7.21).

Remark 7.2. *Again, it is important to stress that, also in this formulation, the performance of the resulting control policy is dependent on the quality of the prediction model $\mathscr{F}_k$, in case this is not fully known. Likewise, the disturbance sampling strategy affects this performance, and different methods with different properties can be employed, e.g., Monte Carlo (MC), quasi-MC, or Gauss-Hermite (GH) sampling [97, 116].*

7.3. Nonmyopic global optimisation

In this section, we show how the GO sequential optimisation problem based on IDW and RBF regression can be formulated and tackled via the ADP techniques previously discussed, with the goal of crafting optimisation strategies that are systematically and structurally able to take future decisions into account and, thus, are nonmyopic.

Given a surrogate model approximating the unknown objective function at unexplored query points, the additional necessary ingredients are 1) a formulation of the dynamics describing how the surrogate model evolves when new information is made available and allowing to sample probable trajectories of such evolution, and 2) a long-term objective goal to be minimised in the form of a reward function. With these components in place, the GO sequential estimation problem can indeed be posed as a DP problem. By considering the iterations as the time axis, we can assign a state at a given iteration k to the regression model to fully characterise it, and we can define the control action at time step k as the next query point. By applying this control action, i.e., by observing the value of the unknown function at the query point, the state of the model is advanced to the next iteration $k+1$, i.e., the regression is updated to include the new information. With a cost associated with this state transition, DP and ADP algorithms can be deployed to optimise the selection of the next query points so as to minimise the cumulative costs incurred during the GO iterations. Therefore, this approach qualifies as nonmyopic as it takes into account the evolution of the underlying surrogate model and the cumulative costs incurred when choosing the sequence of points to be queried. Conversely, greedy GO strategies (e.g., see Section 7.2.1) do not in general take into

[3] In this regard, [97] shows how GPU-based automatic differentiation tools, such as `PyTorch` [160], can efficiently attenuate this drawback by leveraging massive parallelisation of the computations.

account the evolution of the underlying surrogate model for more than one iteration, and are considered myopic for this reason.

Whereas surrogate modelling has already been introduced and discussed in Section 7.2.1 in the form of IDW and RBF regression, the other two necessary ingredients, the surrogate models' dynamics and the reward function, are tackled in Sections 7.3.1 and 7.3.2, respectively.

7.3.1. Dynamics of surrogate models

The first ingredient to prepare in order to tackle GO problems via DP is a dynamical description of the two surrogate modelling approaches, namely IDW and RBF regression. The aim is to approximately predict and simulate, using only the information available at iteration k, how these models would evolve if new hypothetical observations were queried. This will then enable the use of DP techniques to improve upon the base GO sequential optimisation strategy.

Let us first tackle the case of IDW regression, which turns out to be the most straightforward. By leveraging the non-parametric nature of IDW, the model's state s_k at each iteration k is encoded solely by the corresponding dataset $\mathscr{D}_k$, which fully characterises the surrogate model $\hat{f}_k$, i.e., starting from any $\mathscr{D}_k$ we can use (7.3) to fully construct $\hat{f}_k$. The control action a_k is the next query point x_{k+1} to be selected for evaluation. Thus, we have $\mathscr{S}_k = (\mathscr{X} \times \mathbb{R})^k$ and $\mathscr{A}_k = \mathscr{X}$.

In formal terms, consider the initial state $\hat{s}_{k|k} = \mathscr{D}_k = \widehat{\mathscr{D}}_{k|k}$ and assume a sequence of control actions $a_t = x_{t+1}$, $t = k, \dots, k+h$, is given over a horizon of arbitrary length $h \geq 1$. It is possible to predict the evolution of the IDW regression as new data is added by approximating the value of the unknown objective function at the new query points (which cannot be directly observed as only information up to the current iteration k can be used) as $f(x_{t+1}) \approx \hat{f}_{t|k}(x_{t+1})$. Therefore, as per (7.2), we define the next predicted state as $\hat{s}_{t+1|k} = \widehat{\mathscr{D}}_{t+1|k} = \hat{s}_{t|k} \cup \left\{\left(a_t, \hat{f}_{t|k}(a_t)\right)\right\}$, and the overall dynamics of the IDW model are represented as

$$\mathscr{F}_t(\hat{s}_{t|k}, a_t) = \widehat{\mathscr{D}}_{t|k} \cup \left\{\left(x_{t+1}, \hat{f}_{t|k}(x_{t+1})\right)\right\}, \quad t = k, \dots, k+h. \tag{7.23}$$

Obviously, this iterative procedure can only yield an approximate evolution of the regression model as additional hypothetical query-observation pairs are added during the sequential optimisation strategy. The reason is that it only exploits information available up to k, and in general $f(x_{t+1}) \neq \hat{f}_{t|k}(x_{t+1})$, especially in the earlier iterations when the dataset is small and less informative.

Via Radial Basis Functions

In case RBFs are used as surrogate model, an efficient representation of the regression evolution as new data is collected is less intuitive. Recall that the RBF regressor at iteration k has coefficients B_k that, together with points X_k, allow us to compute the estimate of objective f at any $x \in \mathscr{X}$ according to (7.7). Like IDW, the state of such a surrogate model can be defined solely as $s_k = \mathscr{D}_k$, but it is inconvenient to do so. In fact, this definition entails that each time new information is added as (7.23), the linear

system (7.10) has to be solved anew. It is clear then that as the iteration count k increases each linear system becomes more and more computationally expensive to set up and solve.

To formulate an alternative, let us focus on how the regression coefficients change when a new query point is made available, i.e., the relationship between B_k and B_{k+1} when the new data point (x_{k+1}, f_{k+1}) is added to the dataset. Assume $B_k = M_k^{-1} F_k$ has been previously computed, e.g., as per Section 7.2.1. It is then possible to write M_{k+1}^{-1} in an incremental and recursive way as the blockwise inverse [49, Section 28.2]

$$M_{k+1}^{-1} = \begin{bmatrix} M_k & \Phi_k(x_{k+1}) \\ \Phi_k(x_{k+1})^\top & \phi(\varepsilon d(x_{k+1}, x_{k+1})) \end{bmatrix}^{-1} = \frac{1}{c} \begin{bmatrix} cM_k^{-1} + LL^\top & -L \\ -L^\top & 1 \end{bmatrix}, \tag{7.24}$$

where

$$L = M_k^{-1} \Phi_k(x_{k+1}), \tag{7.25}$$

$$c = \phi(\varepsilon d(x_{k+1}, x_{k+1})) - \Phi_k(x_{k+1})^\top M_k^{-1} \Phi_k(x_{k+1}). \tag{7.26}$$

In this context, $c \in \mathbb{R}$ is also known as the Schur complement. It is assumed here that $c \neq 0$.[4] Hence, the blockwise inverse (7.24) allows us to efficiently get the new coefficients as $B_{k+1} = M_{k+1}^{-1} \begin{bmatrix} F_k \\ f_{k+1} \end{bmatrix}$, without having to solve the full system of equations (7.10). For convenience of notation, let us define the function $\mathscr{M}_k : \mathbb{R}^{k \times k} \times \mathbb{R}^n \to \mathbb{R}^{k+1 \times k+1}$ as a compact shorthand for this procedure, i.e.,

$$M_{k+1}^{-1} = \mathscr{M}_k \left(M_k^{-1}, x_{k+1} \right). \tag{7.27}$$

Therefore, in order to exploit the recursive blockwise inversion scheme, in comparison to the case of IDWs, the state of the RBF regression at iteration k must be enhanced with the current inverse of the interpolation matrix, i.e., $\mathscr{S}_k = (\mathscr{X} \times \mathbb{R})^k \times \mathbb{R}^{k \times k}$, while the action space remains unchanged, i.e., $\mathscr{A}_k = \mathscr{X}$. In summary, given the initial state $\hat{s}_{k|k} = \left(\widehat{\mathscr{D}}_{k|k}, \widehat{M}_{k|k}^{-1} \right)$ and the sequence of control actions $a_t = x_{t+1}$, $t = k, \dots, k+h$, the dynamics of the RBF regression model are given by

$$\mathscr{F}_t \left(\hat{s}_{t|k}, a_t \right) = \left(\widehat{\mathscr{D}}_{t|k} \cup \left\{ \left(x_{t+1}, \hat{f}_{t|k}(x_{t+1}) \right) \right\}, \mathscr{M}_t \left(\widehat{M}_{t|k}^{-1}, x_{t+1} \right) \right), \quad t = k, \dots, k+h. \tag{7.28}$$

As before, we remark these dynamics are an approximation due to the fact that, as we move along the prediction horizon from k to $k+h$, we are using the surrogate model to simulate the value of the objective function at the new query points instead of measuring its real value, and in general we have $f(x_{t+1}) \neq \hat{f}_{t|k}(x_{t+1})$.

[4] For simplicity, in those cases where this assumption fails, we fall back to solving the full linear system (7.10) instead. However, more refined incremental strategies that do not need such an assumption could be employed, e.g., incremental singular value decomposition [31]. Alternatively, the sequential decision algorithm may be appropriately penalised for or constrained from selecting any x_{k+1} for which $|c| < \xi$, for some small $\xi > 0$.

Sampling

So far, the dynamics for IDW and RBF regression have been discussed. These surrogate models evolve deterministically because their estimates $\hat{f}_k$ are intrinsically so. However, in order to apply the multi-step tree-based approach (7.22), we need to be able to sample possible trajectories of the regressor dynamics when the objective function is approximated at new query points. Note that, contrarily to IDW- and RBF-based GO, in BO such a property is readily available thanks to the employment of GP models. In fact, a GP regressor offers a probabilistic prediction of the unknown objective function at an unexplored point in the form of a Gaussian posterior distribution. This distribution can then be sampled, thus allowing the simulation of different possible evolutions of the regressor.

In this chapter, we opt to enhance our IDW and RBF predictors with similar behaviour as GPs by inducing Gaussian posterior distributions via the variance function $\sigma_k(x)$ in (7.13). This function was introduced in Section 7.2.1, in the context of the myopic acquisition function, as a heuristic able to provide confidence bounds to induce exploration towards regions of the space $\mathscr{X}$ where the regression prediction is less trustworthy. However, the same function can also be exploited to sample likely estimates of the unknown objective based on IDW/RBF surrogate models, thus rendering these more amenable to the formulation in (7.22).

Consider an IDW or RBF regression model fitted on $\mathscr{D}_k$, at iteration k. The stochastic estimate of the unknown objective f at x_{k+1} is now the random variable $\hat{f}_k^{\mathrm{s}}$ such that

$$\hat{f}_k^{\mathrm{s}}(x_{k+1}) \sim \mathcal{N}\left(\hat{f}_k(x_{k+1}), \sigma_k^2(x_{k+1})\right), \tag{7.29}$$

where $\hat{f}_k(x_{k+1})$ follows from either (7.3) or (7.7), and σ_k has been introduced in (7.13). Thanks to this definition of $\hat{f}_k^{\mathrm{s}}$, we can then sample its distribution to get different sample-based state trajectories of the regressor. We stress here that this estimated distribution is heuristic, contrarily to GPs where typically suitable assumptions are put in place to guarantee that such posterior distributions are theoretically sound.

7.3.2. Reward function

The second ingredient before exploiting the ADP formulation (7.22) to craft a nonmyopic GO strategy is to define a suitable reward function. In this chapter, we choose to define such reward (both for the IDW and RBF setting) as

$$\begin{aligned} -R_k(s_k, a_k) &= \Lambda^{\mathrm{s}}(\mathscr{D}_k, x_{k+1}) \\ &:= \mathbb{E}_{\hat{f}_k^{\mathrm{s}}}\left[\Lambda(\mathscr{D}_k, x_{k+1})\right] \\ &= \hat{f}_k(x_{k+1}) - \lambda \mathbb{E}_{\hat{f}_k^{\mathrm{s}}}\left[\sigma_k(x_{k+1})\right] - \mu \Delta_{F_k} \zeta_k(x_{k+1}), \end{aligned} \tag{7.30}$$

where, instead of using the deterministic prediction $\hat{f}_k$, we advocate to the use of the stochastic posterior $\hat{f}_k^{\mathrm{s}}$ and the computation of the corresponding reward expectation. Since ζ_k does not depend on the prediction and $\mathbb{E}_{\hat{f}_k^{\mathrm{s}}}\left[\hat{f}_k^{\mathrm{s}}(x_{k+1})\right] = \hat{f}_k(x_{k+1})$ by definition, we are left with just the computation of the expected value of the variance function.

No simple closed-form solution to this exists but, since the expectation is over a normally distributed variable, one can approximate it up to an arbitrary precision via GH quadrature [3], i.e.,

$$\mathbb{E}_{\hat{f}_k^{\mathrm{s}}}\left[\sigma_k(x_{k+1})\right] \approx \sum_{j=1}^{q} \xi_j \sqrt{\sum_{i=1}^{k} v_{k,i}(x)\left(z_j - f_i\right)^2}, \tag{7.31}$$

where $q \in \mathbb{N}_{>0}$ is the number of sample points, $z_j \in \mathbb{R}$, $j = 1, \dots, q$, are appropriately spaced (depending on q) samples of $\hat{f}_k^{\mathrm{s}}(x_{k+1})$ and $\xi_j \in \mathbb{R}$ are the associated weights found as

$$\xi_j = \frac{2^{q-1} q! \sqrt{\pi}}{q^2 H_{q-1}^2(\varrho_j)^2}, \tag{7.32}$$

with ϱ_j, $j = 1, \dots, q$, the roots of the Hermite polynomial $H_q : \mathbb{R} \to \mathbb{R}$. Lastly, note that a negative sign has been included in (7.30) to flip the optimisation direction because, whereas R_k is maximised by convention, Λ must be minimised.

7.3.3. Nonmyopic acquisition problems and global optimisation

In summary, we have shown how the surrogate model (be it IDW or RBF) can be regarded as a dynamical system and how it is possible to approximately predict its evolution during the optimisation strategy when an action is taken, i.e., a new query point is selected. Furthermore, we have enabled sampling this evolution of such models by defining a heuristic distribution of the objective estimate (so as to be able to generate different possible state trajectories), as well as modified the vanilla GO myopic acquisition function to take into account this new probabilistic approach.

Inspired by the multi-step tree-based approach (7.22), we propose the following nonmyopic acquisition methods that combine and take advantage of the aforementioned modifications. Compared to the original greedy acquisition (7.12), the proposed ones are predictive, and thus they are able to carry out nonmyopic decisions that foresee and take into consideration the evolution of the surrogate model.

For the case of IDW regression, we define at iteration k the following nonmyopic acquisition problem:

$$\begin{aligned}
\min_{\hat{x}_{k+1|k},\dots,\hat{x}_{k+h|k}} \quad & \Lambda^{\mathrm{s}}\left(\widehat{\mathscr{D}}_{k|k}, \hat{x}_{k+1|k}\right) + \frac{1}{m_k} \sum_{i_k=1}^{m_k} \Lambda^{\mathrm{s}}\left(\widehat{\mathscr{D}}_{k+1|k}^{(i_k)}, \hat{x}_{k+2|k}\right) + \dots \\
& + \frac{1}{\prod_{\ell=k}^{k+h-2} m_\ell} \sum_{i_k=1}^{m_k} \dots \sum_{i_{h-1}=1}^{m_{k+h-2}} \Lambda^{\mathrm{s}}\left(\widehat{\mathscr{D}}_{k+h-1|k}^{(i_k,\dots,i_{k+h-2})}, \hat{x}_{k+h|k}\right) && \text{(7.33a)} \\
\text{s.t.} \quad & \widehat{\mathscr{D}}_{k|k} = \mathscr{D}_k, && \text{(7.33b)} \\
& \widehat{\mathscr{D}}_{t+1|k}^{(i_k,\dots,i_{t-1},i_t)} = \widehat{\mathscr{D}}_{t|k}^{(i_k,\dots,i_{t-1})} \cup \left\{\left(\hat{x}_{t+1|k}, \hat{f}_{t|k}^{(i_k,\dots,i_{t-1},i_t)}\right)\right\}, && \text{(7.33c)} \\
& \qquad\qquad i_t = 1, \dots, m_t, \; t = k, \dots, k+h-1, \\
& \hat{x}_{t+1|k} \in \mathscr{X}, \qquad t = k, \dots, k+h-1, && \text{(7.33d)}
\end{aligned}$$

and take as the next query point its first optimal action, i.e., $x_{k+1} = \hat{x}_{k+1|k}^{\star}$. Since (7.33) is nonlinear, we break ties equally likely in case of multiple equivalent local minima.

Note that the samples of the dynamics evolution $\hat{f}_t^{(i_k,\dots,i_t)}$, $i_t = 1,\dots,m_t$, are drawn from $\hat{f}_t^{\mathrm{s}}\left(\hat{x}_{t+1|k}\right) \sim \mathcal{N}\left(\hat{f}_t\left(\hat{x}_{t+1|k}\right), \sigma_t^2\left(\hat{x}_{t+1|k}\right)\right)$. To preserve differentiability through this sampling step for the sake of computing the sensitivity of the objective (7.33a) w.r.t. the primal variables, the well-known re-parametrisation trick is used [106], i.e., samples are drawn as $\hat{f}_t^{\mathrm{s}}\left(\hat{x}_{t+1|k}\right) = \hat{f}_t\left(\hat{x}_{t+1|k}\right) + \sigma_t\left(\hat{x}_{t+1|k}\right) z$, with $z \sim \mathcal{N}(0,1)$.

Slightly more convoluted is the case for RBF regression, where the additional state entry leads to another set of constraints to be included in the optimisation, leading us to the following problem:

$$\begin{aligned} \min_{\hat{x}_{k+1|k},\dots,\hat{x}_{k+h|k}} \quad & (7.33a) \\ \text{s.t.} \quad & (7.33b) - (7.33d), \\ & \widehat{M}_{k|k}^{-1} = M_k^{-1}, \\ & \widehat{M}_{t+1|k}^{-1} = \mathscr{M}_t\left(\widehat{M}_{t|k}^{-1}, \hat{x}_{t+1|k}\right), \quad t = k,\dots,k+h-1. \end{aligned} \tag{7.34}$$

One can see that, as they do not depend on the samples $\hat{f}_t^{(i_k,\dots,i_t)}$, the predicted interpolation matrices $\widehat{M}_{t|k}^{-1}$, $t = k,\dots,k+h$ need not be sampled, as opposed to $\widehat{\mathscr{D}}_{t|k}^{(i_k,\dots,i_t)}$. All other considerations from (7.33) apply also to (7.34).

Algorithm 4 summarises the proposed nonmyopic methodology. Compared to its myopic counterpart, two major improvements are introduced. In particular, the vanilla myopic acquisition problem (7.12) is replaced with the nonmyopic one in order to combat the greediness of the standard GO formulation and to improve upon its performance. Note that for $h = 1$, i.e., the last iteration, we fall back to solving the (stochastic) myopic problem as there is no need to have a predictive strategy because the algorithm is about to terminate; equivalently, one could notice that for $h = 1$ problems (7.33) and (7.34) degenerate to the minimisation in line 11. On top of this first modification, instead of training in each iteration k the IDW or RBF regression on the whole dataset $\mathscr{D}_k$ from scratch, we also propose to take again advantage of the dynamical formulation of these surrogate models to partially fit them only on the new data (x_{k+1}, f_{k+1}). This is, in general, more computationally efficient, especially in the case of RBF regression where determining solutions to the linear system (7.10) can become computationally expensive for large k.

7.4. Numerical experiments

In this section, we implement the proposed nonmyopic GO optimisation strategies (7.33)-(7.34) and assess the resulting performance on two different experiments, namely, on a collection of several benchmark synthetic and real-world functions from the optimisation and machine learning literature, and on the tuning of a controller for a chemical control task. Our implementation builds on top of `BoTorch` [13], a Python framework for GO/BO that leverages `PyTorch` [160] for massive GPU-based parallelisation and differentiation of its computations. Source code and simulation results are made available in the following repository: `https://github.com/FilippoAiraldi/global-optimization`.

Algorithm 4: Nonmyopic GO algorithm

Input: initial dataset $\mathscr{D}_{N_0}$; search space $\mathscr{X}$; maximum number of iterations N; horizon H; number of samples m_k, $k = N_0, \dots, N$ ($m_k = 1$, if rollout)
Output: $\max_{i=1,\dots,N+1} x_i$

for $k = N_0 \to N$ **do**
  if $k = N_0$ **then**
    Fit initial regression model $\hat{f}_k$ to $\mathscr{D}_{N_0}$ via (7.3) or (7.7)
  else
    Partially fit regression model $\hat{f}_k$ to new data via (7.2) and, if using RBF, (7.27)
  end
  $h \leftarrow \min\{H, N-k+1\}$ `/* compute current horizon */`
  if $h > 1$ **then**
    Get new query point $x_{k+1} = \hat{x}^\star_{k+1|k}$ by solving (7.33) or (7.34)
  else
    Get new query point by $\min_{x \in \mathscr{X}} \Lambda^s(\mathscr{D}_N, x)$ `/* last iteration */`
  end
  $f_{k+1} \leftarrow f(x_{k+1})$ `/* observe value at query point */`
end

In all numerical experiments, the search space $\mathscr{X}$ is a hyperrectangle, i.e., $\mathscr{X} = \left\{x \in \mathbb{R}^n \,\middle|\, \underline{x}_i \le x_i \le \overline{x}_i,\ i = 1, \dots, n\right\}$ for some problem-specific lower and upper bounds $\underline{x}, \overline{x} \in \mathbb{R}^n$. Due to their high nonlinearity, the nonmyopic optimisation problems are solved via L-BFGS-B [38, 237], a quasi-Newton method designed to tackle problems with box-constrained variables, and with multistart to avoid convergence to very suboptimal local minima. As suggested in [16], the coefficients of the acquisition function are scaled with the size of the search space as $\lambda = n^{-1}$ and $\mu = 0.5n^{-1}$. For those problems that are regressed via RBF, we also set $\varepsilon = n^{-1}$. Moreover, in (7.5) $\delta = 10^{-12}$, and a threshold of 10^{-8} on the smallest singular values is imposed (see Section 7.2.1 and [16, Section 3.2] for more details on this). Lastly, the number of samples to approximate the expectation in (7.30) via GH quadrature is fixed as $q = 16$. This value was tuned to strike a balanced trade-off between approximation accuracy and computational performance.

Each numerical experiment is run on a server with Python 3.11.3 and equipped with 16 AMD EPYC 7252 (3.1 GHz) processors, 252GB RAM, and 4 NVIDIA RTX 3090 GPUs.

7.4.1. Synthetic and real-world problems

We test the proposed nonmyopic approaches on a collection of standard synthetic benchmark functions [95, 195] as well as on a set of real-world hyperparameter tuning problems [97, 133, 186, 219]. Table 7.1 reports all the functions, including their search space dimension and bounds. Each is solved in two stages: first, $N_0 = 2n$ initial random points are drawn at random to warm the surrogate models up; then, the main sequential algorithm is run to pursue the function's optimum. To combat the influence of random-

ness, each experiment is repeated 30 times with different seeds, while the rest of the other hyperparameters is kept the same.

Table 7.1.: Synthetic benchmarks and real-world hyperparameter tuning problems

Group	Name	n	$\underline{x}$	$\overline{x}$
Synthetic	Ackely	2	-32.767	32.767
	Adjiman	2	-1	$[2, 1]$
	Bohachevsky	2	-100	100
	Branin	2	$[-5, 0]$	$[10, 15]$
	Bukin	2	$[-15, -3]$	$[-5, 3]$
	Drop-wave	2	-5.12	5.12
	Egg holder	2	-512	512
	Hartman (3d)	3	0	1
	Hartman (6d)	6	0	1
	Rastrigin	2	-5.12	5.12
	Rosenbrock	8	-5	10
	Step 2	5	-100	100
	Styblinski-Tang	5	-5	5
Real-world	LDA	3	$[0.5, 0, 0]$	$[1, 10, 14]$
	LogReg	4	$[0, 0, 4.32, 2.32]$	$[10, 1, 11.32, 11.32]$
	NN Boston	4	0	1
	NN Cancer	4	0	1
	Robot pushing (3d)	3	0	1
	Robot pushing (4d)	4	0	1
	SVM	3	$[-3.32, -3.32, -13.29]$	$[19.93, 2.32, -3.32]$

The rollout and the multi-step approach are both tested with different horizon lengths H. The former is tested with a lookahead of up to five steps, while the latter is tested only up to three due to computational complexity. For the multi-step approach, the numbers of samples at iteration k are set to $m_k = 10$, $m_{k+1} = 5$, and are kept constant across iterations. For rollout, obviously we have that $m_t = 1$, $t = k, \dots, k + h - 2$, with $h \in \{2, \dots, 5\}$. The sampling of uncertain observations is carried out according to the distribution (7.29) for both methodologies. This is achieved, in line with the state of the art [97], either randomly via MC sampling of the normal distribution (in particular, via the Quasi-MC Box-Muller method [154]) or deterministically by approximation via GH quadrature samples.

To quantitatively appreciate the performance of the algorithms, the optimality gap G, defined as

$$G = \frac{\min\limits_{i=1,\dots,N_0} f_i - \min\limits_{i=1,\dots,N+1} f_i}{\min\limits_{i=1,\dots,N_0} f_i - f(x^\star)}, \tag{7.35}$$

is employed as a metric that measures the improvement in the unknown objective function from the initial iteration N_0 to the final iteration N, normalised by the maximum

achievable reduction. Obviously, the closer the gap is to 1, the less suboptimal the convergence point is[5]. The results on the mean and median gap, obtained with the aforementioned settings, are reported in Table 7.2. The functions' groups and names follow from Table 7.1, while methods' names are abbreviated to include strategy, horizon, and sampling method, e.g., *MS-3 (GH)* corresponds to a multi-step tree-based approach with horizon $H = 3$ and GH sampling. For the sake of comparison, we include the results for the myopic strategy from [16], upon which we have built our proposed approach. Finally, the best mean/median result in each problem is highlighted in blue bold, whereas results that are not significantly worse than the best (under the one-sided Wilcoxon signed-rank test with $p = 5\%$) are reported in purple italics.

One can easily notice that all nonmyopic combinations outperform the myopic vanilla strategy, which is comparable in a statistically significant way only for a small subset of the problems. This empirically validates the efficacy of exploiting a lookahead strategy to overcome the drawbacks of single-state acquisition functions. Similarly, since the multi-step approach generalises rollout, as highlighted in Section 7.2.2, it is unsurprising to witness it being the most performing method (or not significantly worse than the best) for the majority of tests. On the other side, longer horizon windows do not necessarily relate to better performance. This observation is in accordance with the rest of the BO and control state of the art [178, 232]: while predicting further into the future is generally desirable, it may be counterproductive due to model mismatches.

All the aforementioned points can also be appreciated in Figure 7.2, which summarises the average performance (in terms of the gap) versus the mean computation time per iteration for each method. From the figure, it is again clear how the nonmyopic methods outperform the myopic. It also highlights one of the immediate drawbacks of employing a multi-step approach or long horizons, i.e., computational complexity. In fact, despite harnessing the GPU parallelisation capabilities, nonmyopic algorithms are innately more complex to execute and often need to be run with a larger number of multistart trials to obtain convergence to an acceptable suboptimal point.

In Figure 7.2, we note also that rollout with MC sampling in general appears to be associated with slightly better empirical performance than rollout with GH quadrature; conversely, GH is comparable to or outperforms MC in the multi-step algorithm. This outcome is also observed in [97] in the context of non-myopic BO. However, our observation, while consistent with the literature, remains empirical and necessitates further theoretical investigation. It can be additionally noticed in all configurations that GH sampling requires less computation time: the reason is probably attributable to the sampling of the Sobol quasi-MC engine, which empirically appears to be slower than employing the fixed GH quadrature samples.

Lastly, we also report in Figure 7.3 the evolution of the optimality gap during the sequential optimisation iterations. For clarity, here we include only the base myopic strategy and the best nonmyopic one. It can be seen that, in most cases, not only does the nonmyopic algorithm converge to a better gap, but it also does so often faster than its myopic counterpart. This property is especially desirable in contexts in which the budget, e.g., the maximum number of iterations N, is limited [115, 117].

[5] This holds true except in the extremely unlikely case in which the global optimal point is already found in the first N_0 random iterations, i.e., $\exists i \in \{1, \dots, N_0\} : f_i = f(x^\star)$. If so, this metric is no longer well-defined.

Table 7.2.: Mean and median final optimality gap on the synthetic and real functions for rollout (R) and multi-step (MS) methods with different horizon lengths and sampling schemes ([a]an RBF surrogate model was used instead of IDW)

Name		Myopic	R-2 (GH)	R-2 (MC)	R-3 (GH)	R-3 (MC)	R-4 (GH)	R-4 (MC)	R-5 (GH)	R-5 (MC)	MS-2 (GH)	MS-2 (MC)	MS-3 (GH)	MS-3 (MC)
Ackely[a]	mean	0.397	*0.466*	*0.443*	*0.452*	*0.502*	*0.443*	*0.444*	*0.446*	**0.509**	*0.442*	*0.437*	*0.499*	*0.459*
	median	0.409	*0.503*	*0.419*	*0.514*	*0.528*	*0.478*	*0.473*	*0.457*	*0.528*	*0.466*	*0.465*	**0.532**	*0.478*
Adjiman	mean	0.713	0.887	*0.953*	0.799	*0.943*	0.801	*0.933*	0.810	0.893	*0.987*	*0.969*	*0.981*	**0.991**
	median	0.731	0.997	1.000	0.895	*0.999*	0.840	0.999	0.973	0.994	**1.000**	*1.000*	1.000	*1.000*
Bohachevsky[a]	mean	0.171	*0.405*	0.354	*0.411*	0.328	*0.460*	*0.418*	**0.472**	*0.351*	0.334	0.221	*0.424*	0.359
	median	0.000	*0.186*	0.200	*0.375*	0.203	*0.518*	*0.399*	**0.518**	*0.106*	0.167	0.000	*0.464*	0.237
Branin	mean	0.841	0.849	0.842	*0.892*	*0.880*	*0.887*	0.875	**0.907**	*0.852*	0.847	0.849	0.802	0.853
	median	0.881	0.927	0.899	*0.931*	*0.958*	*0.950*	0.940	**0.960**	*0.919*	0.883	0.894	0.902	0.919
Bukin	mean	0.886	**0.959**	0.887	0.884	0.875	0.819	0.832	0.794	0.773	*0.941*	*0.944*	0.906	0.914
	median	0.969	**0.977**	0.944	0.947	0.936	0.893	0.900	0.851	0.855	*0.967*	*0.962*	0.944	0.943
Drop-wave	mean	0.417	0.501	0.510	*0.533*	0.467	0.496	*0.574*	*0.589*	**0.645**	*0.537*	*0.585*	*0.626*	*0.625*
	median	0.365	0.537	0.546	*0.631*	0.476	0.447	*0.563*	*0.616*	**0.732**	*0.518*	*0.704*	*0.721*	*0.701*
Egg holder	mean	0.397	0.480	0.425	0.444	*0.480*	*0.500*	0.448	*0.502*	*0.487*	0.498	0.452	*0.493*	**0.550**
	median	0.364	0.427	0.385	0.383	*0.435*	*0.474*	0.414	*0.442*	*0.485*	0.445	0.389	*0.467*	**0.524**
Hartman (3d)	mean	0.576	0.647	0.744	0.725	0.758	0.637	0.726	0.628	0.690	0.802	0.791	*0.890*	**0.902**
	median	0.701	0.730	0.821	0.822	0.848	0.693	0.794	0.697	0.806	0.924	0.905	*0.952*	**0.956**
Hartman (6d)[a]	mean	*0.646*	**0.714**	0.346	0.651	0.493	*0.655*	0.510	*0.676*	0.538	0.292	0.288	0.517	0.494
	median	*0.724*	*0.746*	0.336	*0.699*	0.506	**0.752**	0.553	*0.719*	0.603	0.292	0.256	0.540	0.520
Rastrigin	mean	0.267	0.374	*0.410*	0.336	*0.402*	0.342	0.322	0.366	*0.428*	*0.432*	*0.392*	**0.468**	*0.399*
	median	0.227	0.365	*0.404*	0.298	*0.398*	0.304	0.313	0.340	*0.390*	*0.390*	*0.375*	**0.440**	*0.413*
Rosenbrock[a]	mean	0.873	0.892	0.861	0.901	0.963	0.923	**0.967**	0.938	*0.940*	0.831	0.859	*0.928*	*0.949*
	median	0.933	0.941	0.932	0.931	0.973	0.949	**0.982**	0.951	*0.969*	0.890	0.933	*0.977*	*0.977*
Step 2	mean	0.872	0.923	0.868	0.879	0.839	0.851	0.840	0.836	0.823	*0.936*	*0.935*	*0.948*	**0.952**
	median	0.888	0.933	0.905	0.894	0.836	0.866	0.864	0.848	0.870	*0.954*	*0.954*	**0.964**	*0.954*
Styblinski-Tang	mean	0.390	0.490	0.580	0.430	0.553	0.438	0.503	0.432	0.494	**0.632**	0.593	0.618	*0.624*
	median	0.385	0.495	0.570	0.445	0.528	0.431	0.478	0.430	0.456	**0.648**	0.573	0.643	*0.621*
LDA	mean	*0.875*	*0.868*	*0.846*	*0.864*	*0.884*	*0.882*	*0.875*	*0.879*	0.876	**0.890**	*0.857*	0.812	*0.875*
	median	*0.922*	*0.915*	*0.930*	*0.919*	*0.921*	*0.926*	*0.929*	*0.919*	*0.916*	*0.929*	*0.921*	*0.917*	**0.931**
LogReg[a]	mean	0.906	*0.921*	0.803	*0.920*	0.737	*0.934*	0.732	**0.939**	0.704	0.899	0.811	0.789	0.706
	median	0.982	*0.988*	0.959	*0.984*	0.911	*0.990*	0.915	**0.992**	0.893	0.971	0.938	0.935	0.887
NN Boston	mean	0.143	0.232	0.203	0.198	*0.245*	0.232	0.255	0.212	0.268	*0.268*	0.260	*0.258*	**0.304**
	median	0.066	0.210	0.125	0.101	*0.206*	0.226	0.249	0.138	0.276	*0.265*	0.258	*0.266*	**0.333**
NN Cancer	mean	0.168	0.190	0.154	0.215	*0.325*	0.260	*0.313*	0.219	**0.354**	0.214	*0.242*	0.201	*0.286*
	median	0.000	0.000	0.000	0.000	*0.077*	0.000	*0.000*	0.000	**0.304**	0.000	*0.000*	0.000	*0.000*
Robot pushing (3d)	mean	0.593	0.730	0.839	0.720	0.818	0.690	0.806	0.675	0.797	*0.877*	*0.892*	*0.891*	**0.893**
	median	0.583	0.786	0.886	0.771	0.880	0.758	0.862	0.714	0.868	*0.900*	*0.903*	*0.905*	**0.916**
Robot pushing (4d)	mean	0.358	0.497	0.536	0.432	0.526	0.416	0.476	0.428	0.472	0.674	0.652	*0.686*	**0.702**
	median	0.356	0.528	0.530	0.418	0.522	0.441	0.456	0.429	0.461	0.682	0.676	*0.710*	**0.718**
SVM	mean	*0.929*	0.956	0.948	0.948	0.943	*0.959*	0.947	0.950	0.903	*0.961*	*0.956*	*0.957*	**0.963**
	median	*0.977*	*0.977*	*0.977*	0.975	*0.972*	*0.971*	0.973	0.976	0.966	**0.980**	*0.978*	*0.977*	*0.977*

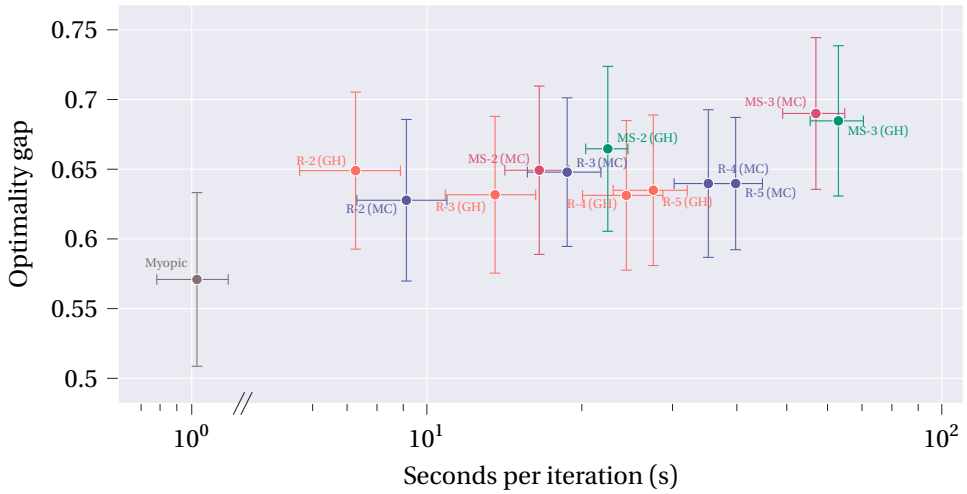

Figure 7.2.: Average gap (aggregated over all problems) versus the mean time per iteration ± the corresponding standard error of the mean for rollout (R) and multi-step (MS) methods with different horizon lengths and sampling schemes. Note that the time axis is in logarithmic scale and has a discontinuity

7

7.4.2. Data-driven tuning of an MPC controller

The second numerical experiment consists of utilising the proposed methodology to automatically tune an MPC controller for a continuously stirred tank reactor system. This system, adapted from [188, 193], undergoes the set of reactions

$$\mathrm{A} \rightarrow \mathrm{B} \rightarrow \mathrm{C}, \quad 2\mathrm{A} \rightarrow \mathrm{D}. \tag{7.36}$$

where A, B, C and D identify different chemical reagents. Its nonlinear dynamics are given by the following differential equations:

$$\dot{c}_\mathrm{A} = F\left(c_{\mathrm{A},0} - c_\mathrm{A}\right) - k_1 c_\mathrm{A} - k_3 c_\mathrm{A}^2, \tag{7.37}$$

$$\dot{c}_\mathrm{B} = -F c_\mathrm{B} + k_1 c_\mathrm{A} - k_2 c_\mathrm{B}, \tag{7.38}$$

$$\begin{aligned}\dot{T}_\mathrm{R} = {} & F\left(T_\mathrm{in} - T_\mathrm{R}\right) + \frac{k_\mathrm{W}\mathrm{A}}{\rho c_\mathrm{p} V_\mathrm{R}}\left(T_\mathrm{K} - T_\mathrm{R}\right) \\ & - \frac{k_1 c_\mathrm{A}\Delta H_\mathrm{AB} + k_2 c_\mathrm{B}\Delta H_\mathrm{BC} + k_3 c_\mathrm{A}^2 \Delta H_\mathrm{AD}}{\rho c_\mathrm{p}},\end{aligned} \tag{7.39}$$

$$\dot{T}_\mathrm{K} = \frac{\dot{Q}_\mathrm{K} + k_\mathrm{W} A\left(T_\mathrm{R} - T_\mathrm{K}\right)}{m_\mathrm{K} c_{\mathrm{pK}}}, \tag{7.40}$$

with

$$k_i = k_{0,i} \exp\left(-\frac{E_{A,i}}{R\left(T_\mathrm{R} + 273.15\right)}\right), \quad i = 1,2,3, \tag{7.41}$$

where the feed flowrate $F \in \mathbb{R}$ is the control input, and the concentrations $c_A, c_B \in \mathbb{R}$ of the corresponding components and the reactor and coolant temperatures $T_R, T_K \in \mathbb{R}$ form the state (please refer to [193] for more details on the dynamics and its coefficients). At any sampled time t, the whole state is unknown and only the concentration of species B and the reactor temperature are assumed to be measurable with some additive noise, so that

$$y_t = \begin{bmatrix} c_B \\ T_R \end{bmatrix} + w_t, \quad w_t \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.2^2 & 0 \\ 0 & 10^2 \end{bmatrix} \right). \tag{7.42}$$

The goal is to control the system in such a way as to maximise the production of component B, expressed by the instantaneous production of moles of this component $\dot{n}_B = V_R c_B F$. At the same time, the controller should strive to constrain the temperature of the reactor within the range $[100, 150]$ °C. Therefore, we discretise time with the sampling interval $t_0 = 0.005\,\text{h}$ and deploy a piecewise-constant control policy. At sampled time t, this policy is provided by the following parametric discrete-time MPC controller with horizon $h_c = 10$:

$$\min_{\substack{\hat{y}_{t|t},\dots,\hat{y}_{t+h_c|t} \\ \hat{F}_{t|t},\dots,\hat{F}_{t+h_c-1|t} \\ \underline{\sigma}_t,\dots,\underline{\sigma}_{t+h_c} \\ \overline{\sigma}_t,\dots,\overline{\sigma}_{t+h_c}}} \quad \sum_{j=t}^{t+h_c} w\left(\underline{\sigma}_j + \overline{\sigma}_j\right) - \sum_{j=t}^{t+h_c-1} V_R \hat{c}_{B,j|t} \hat{F}_{j|t} \tag{7.43a}$$

$$\text{s.t.} \quad \hat{y}_{t|t} = y_t, \tag{7.43b}$$

$$\hat{y}_{j+1|t} = \mathscr{F}_{\theta_m}\left(\hat{y}_{j|t}, \hat{F}_{j|t}\right), \quad j = t, \dots, t+h_c-1, \tag{7.43c}$$

$$5 \le \hat{F}_{j|t} \le 35, \quad j = t, \dots, t+h_c-1, \tag{7.43d}$$

$$\hat{T}_{R,j|t} \ge 100 + \theta_b + \underline{\sigma}_j, \quad j = t, \dots, t+h_c, \tag{7.43e}$$

$$\hat{T}_{R,j|t} \le 150 - \theta_b + \overline{\sigma}_j, \quad j = t, \dots, t+h_c, \tag{7.43f}$$

$$\overline{\sigma}_j \ge 0, \ \underline{\sigma}_j \ge 0, \quad j = t, \dots, t+h_c. \tag{7.43g}$$

The constraints on the temperature (7.43e)-(7.43f) are made soft via the use of the slack variables $\underline{\sigma}_t, \overline{\sigma}_t$, which are appropriately penalised in the objective (7.43a) with a sufficiently high weight $w = 4 \cdot 10^3$. In this way, the controller is encouraged to satisfy the constraints, but the optimisation problem will not become infeasible in case of an unavoidable violation of these. Moreover, a backoff parameter $\theta_b \in \mathbb{R}$ is included in the temperature constraints. When appropriately tuned, this parameter allows the controller to avoid violations more robustly. As in [188], we assume the dynamics (7.37)-(7.40) are black-box, i.e., they cannot be used in the proposed model-based control approach. Thus, as prediction model in (7.43c), we use the unitary-lag NARX model $\mathscr{F}_{\theta_m} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ defined as

$$\mathscr{F}_{\theta_m}(y, F) = \theta_{m,o} + \begin{bmatrix} \theta_{m,y,1}^\top \\ \theta_{m,y,2}^\top \end{bmatrix} y + \theta_{m,F} F + \begin{bmatrix} \theta_{m,y^2,1}^\top \\ \theta_{m,y^2,2}^\top \end{bmatrix} y^2 + \theta_{m,F^2} F^2, \tag{7.44}$$

with $\theta_m = \begin{bmatrix} \theta_{m,o}^\top & \theta_{m,y,1}^\top & \theta_{m,y,2}^\top & \theta_{m,F}^\top & \theta_{m,y^2m,1}^\top & \theta_{m,y^2,2}^\top & \theta_{m,F^2}^\top \end{bmatrix}^\top \in \mathbb{R}^{14}$. Alongside θ_b, these forms the parameter vector $\theta = \begin{bmatrix} \theta_m^\top & \theta_b \end{bmatrix}^\top$, totaling to $14 + 1 = 15$. As usual, we

adopt a receding horizon approach: once (7.43) is solved for t, only the first optimal action $\widehat{F}^{\star}_{t|t}$ is applied to the real system, and at the next time step $t+1$ the procedure is iterated again.

To maximise the production of component B and to minimise constraint violations, a valid numerical value must be assigned to the parameters θ. However, doing so manually can be challenging due to, e.g., the nonlinearities in the system and the high number of parameters to be tuned. Therefore, we propose to exploit the proposed GO strategies to automatically tune these parameters. Consider the objective function

$$J(\theta) = \int_0^{t_\mathrm{f}} \big(w \max\{0, 100 - T_\mathrm{R}\} + w \max\{0, T_\mathrm{R} - 150\} - n_\mathrm{B}\big) dt, \tag{7.45}$$

with $t_\mathrm{f} = 0.2\,\mathrm{h}$. This function is easy to evaluate via simulation once a fixed value of θ is given, but is otherwise difficult to formulate or analyse in closed form. Therefore, mirroring (7.1), we can pose the tuning problem as the optimisation problem

$$\theta^{\star} \in \underset{\theta \in \Theta}{\arg\min} \quad J(\theta), \tag{7.46}$$

where Θ bounds the search space to the hyperrectangle $[-2,2]^{14} \times [0,10]$.

To solve it, we deploy a similar algorithm to Section 7.4.1. First, a random selection of $N_0 = 5$ parameters and their corresponding evaluation of the performance of the parametric MPC policy, i.e., $\big\{\big(\theta_i, J(\theta_i)\big) \mid i = 1, \dots, 5\big\}$, are used to warm the surrogate model up (in this case, IDW). Then, different GO strategies are given 50 iterations to seek the optimal parametrisation of the MPC controller. When evaluating $J(\theta_i)$, each episode is limited to $\frac{t_f}{t_0} = 40$ time steps, and the initial conditions are fixed to $c_{\mathrm{A},0} = c_{\mathrm{B},0} = 1\,\mathrm{mol\,L^{-1}}$, $T_{\mathrm{R},0} = T_{\mathrm{K},0} = 100\,^{\circ}\mathrm{C}$ for every simulation. Since the aforementioned NARX model is nonlinear, to solve (7.43) we employ the IPOPT solver [212] alongside a multistart approach with 10 run to mitigate convergence to poor solutions. The evolution of the real reactor dynamics under the MPC control policy is instead computed via numerical integration of (7.37)-(7.40) with CVODES [93]. Since the state observations y_t and the selection of warm-up parameters are stochastic, the experiment is repeated 30 times with different random seeds.

We tested the myopic algorithm [16] and the proposed nonmyopic ones as in the previous experiment. In particular, two different nonmyopic approaches were selected as best: the rollout approach with horizon 4 and the multi-step approach with horizon 3, both making use of the GH sampling scheme. The results on the convergence of the optimality gap are reported in Figure 7.4a, where it is confirmed again that the two nonmyopic methods outperform the myopic one, despite the much higher number of decision variables compared to the benchmarks in Section 7.4.1. However, it is surprising to see the multi-step method performing worse than the rollout. One explanation could be that, given the high dimensionality of the search space Θ (which causes the multi-step tree-based optimisation problem to be more complex and difficult to solve accurately in comparison to the rollout problem) as well as the fact that the surrogate model's approximate dynamics are especially poor in such space, it empirically appears to be more efficient to rely on less predicted trajectories (e.g., one as in rollout) rather than multiple (as in the multi-step approach).

Figure 7.4b shows how the reactor temperature T_{R} evolves in time at different optimisation iterations of the best performing rollout method. From this figure, two aspects of notice can be seen. The first is that it confirms that the method is efficient in learning at convergence a control policy that is safe, i.e., yielding temperatures that do not violate the constraints. This is thanks to these violations being adequately penalised in the objective (7.45). Although this is a naive way of addressing constraints, we note that the proposed nonmyopic method can be easily extended to the setting of constrained black-box optimisation [4, 188], a topic that investigates algorithms that are able to systematically handle and learn at the same time unknown constraint functions on the decision variable $\theta \in \Theta$. The second point of interest is that the proposed method is also successful in discovering MPC policies that can sustain higher reactor temperatures more consistently. This is advantageous because a higher T_{R} will lead to higher reaction rates k_i, $i = 1, 2, 3$, leading to a higher conversion of component A to component B. This will in turn maximise the objective (7.45).

7.5. Conclusions

In this chapter, we have demonstrated how deterministic GO approaches can benefit from the incorporation of multi-step lookahead formulations. Building on top of solid theoretic foundations, from dynamic programming to optimal control, we introduced novel nonmyopic acquisition strategies tailored to IDW and RBF surrogate modelling, extending the lookahead paradigm from the Bayesian setting to the deterministic as well. The proposed methods are better suited for handling the complex and ever-present exploration-exploitation dilemma compared to the traditional acquisition functions, and, by acting less greedily, are often able to converge to better solutions in a smaller amount of iterations. Numerical experiments showcased these advantages even in high-dimension, highly nonlinear optimisation benchmarks and problems. As a downside, the proposed methodologies are hindered by the increased computational burden due to the larger and more complex optimisation problems. However, these issues are mitigated by leveraging GPUs and their massive parallelisation capabilities.

Future work will tackle the issue of constrained global optimisation where the constraint functions are also unknown and need to be represented by IDW or RBF regression models, and investigate how the proposed nonmyopic deterministic strategies can be extended to such scenarios with unknown constraints.

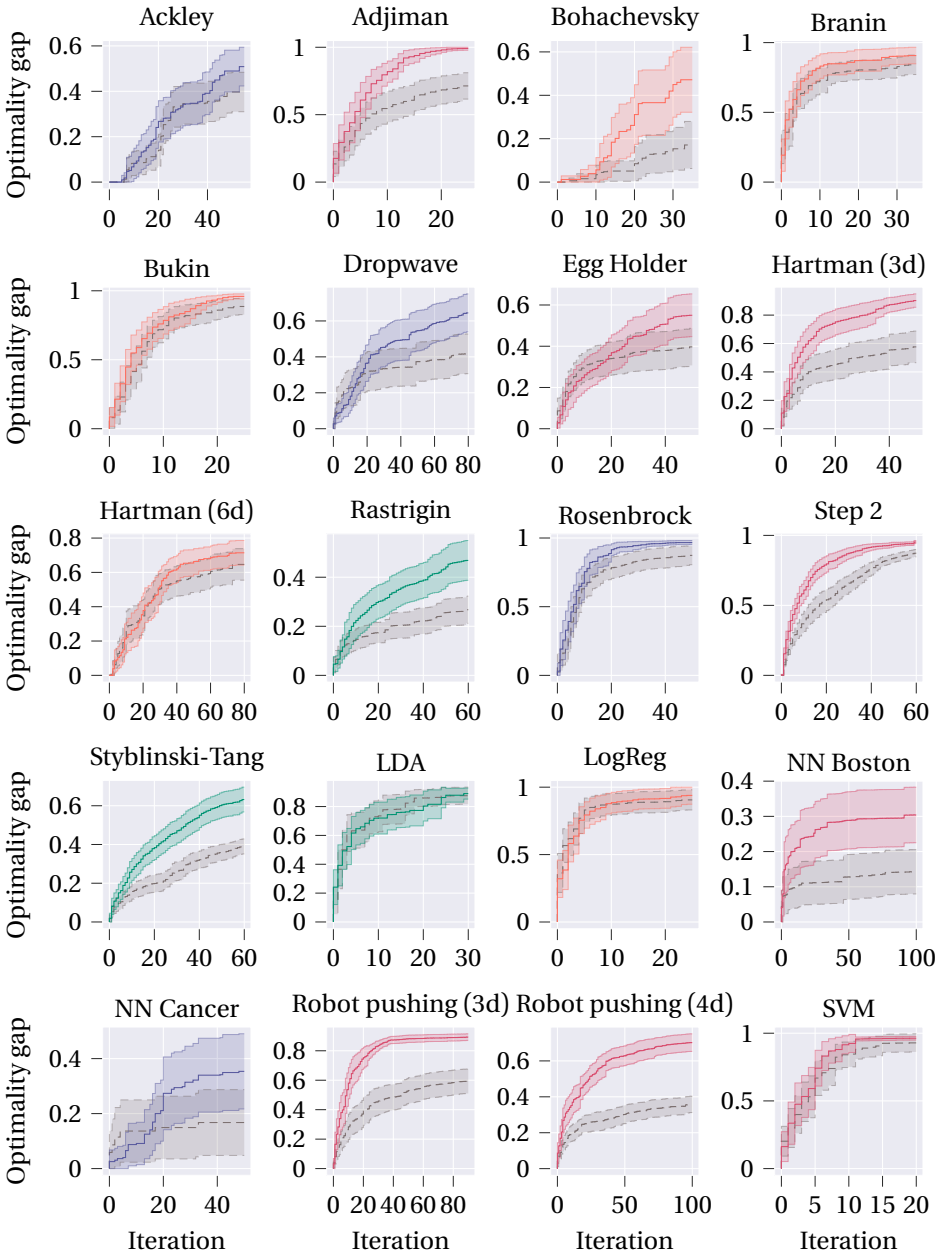

Figure 7.3.: Comparison of the evolution of the optimality gap during optimisation (excluding the N_0 warm-up iterations) of each problem for the myopic method (dashed) and the best mean-performing nonmyopic method (solid), as reported in Table 7.2. The colour scheme follows from Figure 7.2, and averages and 95% confidence intervals are computed over the 30 repetitions

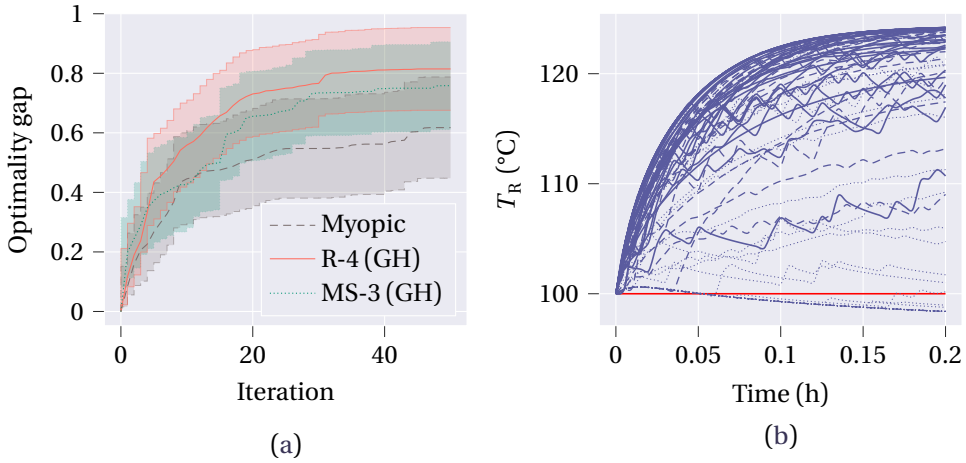

Figure 7.4.: (a) Comparison of the evolution of the optimality gap during optimisation (excluding the N_0 warm-up iterations) of the MPC tuning problem for the rollout (R) and multi-step (MS) method. The colour scheme follows from Figure 7.2, and averages and 95% confidence intervals are computed over the 30 repetitions. (b) Evolution of the reactor temperature T_R, under the MPC policy, during initial (dotted), middle (dashed) and final (solid) iterations of the *R-4 (GH)* method, for each of the 30 repetitions of the experiment. In red, its lower bounds

8

Conclusions

This chapter concludes this thesis with a summary of the contributions, addressing the research questions brought forth in Section 1.2, and with indications for future research work in the field of model-based reinforcement learning.

8.1. Summary

The aim of this dissertation is to provide answers to the overarching research question on how model-based RL can benefit predictive control and optimisation.

At a high level, this dissertation offers new methodologies and further numerical evidence corroborating that guarantees on safety during RL training process can indeed be provided, if not deterministically, at least in a probabilistic sense. Unsurprisingly, achieving such results in general necessitates to give up on purely model-free approaches and to integrate some degree of model-based knowledge in the design. This could be seen as a disadvantage in some applications, e.g., where modelling the system to be controlled is impossible. However, leveraging MPC as function approximation does not preclude its model-based components (i.e., prediction model, objective, and constraints) from being trainable, so even approximate representations (such as a neural network as terminal cost function) can often yield a better-behaved, more interpretable policy than its model-free counterparts. The benefits of this model-based philosophy are observed throughout the dissertation in different application-based, uncertain and multi-agent systems, and also readily extend to other domains, such as global optimisation based on deterministic surrogate models.

At the same time, it is worth noting that, while standard MPC requires the practitioner to build the prediction model and objective function from first principles and manually tune their parameters, it may be argued that the introduction of RL shifts this burden rather than eliminating it. In particular, the focus falls now on the hyperparameters of the learning process, whose selection carries its own non-trivial challenges. Indeed, the quality of outcomes in MPC-RL was found to be strongly correlated with these learning hyperparameters, most notably the choice of the controller's parametrisation. This choice is rarely straightforward and generally still requires a degree of domain expertise in the control task at hand (though, once a parametrisation family is selected, MPC-RL conveniently delegates the tuning of its values to an automatic gradient-based

algorithm). Note that this issue is analogous to the problem of selecting a network topology for neural function approximators in deep RL, a problem for which automated tools, such as neural architecture search [58], are being investigated in the literature. However, this remains a challenge in MPC-RL.

Another drawback of relying on MPC to inject model-based knowledge into the decision-making process is the increased computational burden. This is due to the necessity of repeatedly solving one or more optimal control problems at every time step or iteration of the process. This issue affects both control and optimisation applications, as well as both deployment and training phases (if any), and becomes more prominent as the task at hand increases in dimension and complexity (e.g., due to nonlinear behaviours). Interested practitioners are advised to carefully consider the software/hardware computational capacity at their disposal before integrating predictive algorithms into their own architectures. As examples, we highlight that training an MPC-RL agent via a value-based method, such as Q-learning, requires solving two optimisation problems per visited state, doubling computation times compared to traditional MPC. Even at deployment after training, MPC control policies were usually found to be much slower than deep RL policies (i.e., based on neural networks); see Chapter 5, Table 5.7, where the MPC computation time was recorded on average to be two orders of magnitude larger than the neural network-based counterpart. Likewise, the very same issue affects our contribution on gradient-free optimisation in Chapter 7: as evidenced in Figure 7.2, our lookahead acquisition policies are at least seven times slower than the non-predictive, myopic baseline. For these reasons, especially in the case of real-time decision making, the adoption of the advanced solutions discussed in this dissertation should be also guided by the trade-off between the additional computational cost and the performance gains they introduce.

Delving in more details into each part of this thesis, from the perspective of MPC in Part I, the dissertation offers insights on how safety and scalability can be tackled, and extends the realm of applications of MPC-based RL; in the gradient-free global optimisation setting in Part II, the dissertation investigates how to craft computationally efficient lookahead acquisition strategies. In relation to each of the research questions posed in the Introduction chapter, the contributions are summarised below.

RQ-SA. *How can safety be enforced with high probability and in a computationally efficient and tractable way during the training process of a reinforcement learning agent with model predictive control as function approximation?*

Chapter 2 answers this question by leveraging GP regression to learn a probabilistic representation of the set of MPC parameters yielding a safe policy. This is achieved by collecting trajectory data and corresponding constraint violations. When updated by the RL algorithm, the parametrisation is then constrained to lie inside this set via a chance constraint. The method thus encourages the discovery of safe policies while retaining computational tractability of said chance constraint thanks to the normality assumption.

Chapter 3 proposes as solution an MPC controller that is always probabilistically safe by construction, no matter what its parametrisation is, thanks to the use of CBFs. To address disturbances in the dynamics, it employs a stochastic

MPC formulation. A short horizon length, coupled with a learnable terminal cost function, guarantees that the MPC controller retains low computational demands.

RQ-SC. *How can reinforcement learning with model predictive control as function approximation be scaled to a multi-agent setting in a distributed and privacy-preserving way?*

This question is addressed in Chapter 4, where a decentralised Q-learning algorithm is adapted to a topology with multiple agents, each equipped with an MPC function approximator. ADMM is employed to solve the centralised control problem in a distributed way. It is shown that, under the assumption of convexity of the control problem and in the limit of ADMM and consensus iterations, the set of MPC sensitivities required for distributed updates is locally available. With this proof in place, the distributed Q-learning update is shown to be equivalent to a centralised one, without any centralisation. This is achieved in a privacy-preserving manner by leveraging global average consensus, allowing global agreement on values necessary for the distributed updates only via neighbour-to-neighbour communication.

RQ-AP. *Can the range of applications in which reinforcement learning with model predictive control as function approximation has been successfully applied be extended to other relevant, challenging control problems not yet tackled in the state of the art?*

Chapters 5 and 6 extend the applications of MPC-based RL to climate control in greenhouses and ramp metering for highway traffic control. Chapter 5 empirically demonstrates the competitiveness of MPC-based RL, compared to other non-learning and learning-based approaches from the literature, in boosting crop production and overall economic profit while reducing violations of operational constraints (on, e.g., indoor humidity and temperature). Likewise, Chapter 6 is successful in deploying MPC-based RL to a highway benchmark stretch to control its on-ramp flows. When compared to traditional and learning-based metering strategies, the proposed method shows the best trade-off between average travel time, control variability and violation of the length constraint on its on-ramp queue. After RL training, the proposed MPC controller is able to deliver, on average, higher vehicle speeds and lower traffic densities in the highway stretch.

RQ-GO. *How can approximate dynamic programming techniques be employed to formulate nonmyopic global optimisation approaches with deterministic surrogate models?*

Chapter 7 replies to this question by adopting rollout and multi-step lookahead optimisation, two well-known model-based RL techniques in the field of approximate dynamic programming, to the case of RBF- and IDW-based deterministic surrogate models. These models have been recently shown to demonstrate performance comparable to the conventional Bayesian approaches, but at a lower computational cost. By reformulating the dynamical evolution of

these models as new information is observed and crafting a heuristic sampling scheme, new nonmyopic acquisition strategies based on the aforementioned lookahead strategies are constructed. Their efficacy is then demonstrated on synthetic benchmarks (e.g., Rosenbrock, Ackley, Branin functions) and real-world problems (e.g., tuning of ML methods such as neural networks and support vector machines), as well as on a data-driven tuning application of a nonlinear MPC controller.

8.2. Recommendations for future research

While this thesis addresses some of the gaps in the literature, there are still several avenues to explore to advance the state of the art of RL in predictive control and optimisation. Here, suggestions for plausible future work and extensions to the research questions tackled within this thesis are mentioned.

- **Safety.** Achieving safety during learning with RL, particularly in the presence of mismatches between the prediction model and the system dynamics, is intrinsically challenging. This is because RL algorithms seek to adjust the MPC parametrisation towards maximum performance, with no care for the convergence of the prediction model to the true dynamics. In turn, constraint satisfaction becomes more difficult to certify when prediction errors occur. To mitigate this issue, Chapter 2 proposes an approach that relies on the collection of trajectory data on constraint violations to probabilistically promote, but not enforce, safe control policies after each update. However, by bringing GPs into the loop, this approach introduces additional assumptions on the Gaussianity of the regressed data and adds an extra layer of design complexity. On the other hand, Chapter 3 employs a CBF to guarantee safety at the controller level, but requires perfect knowledge of the system dynamics to ensure the CBF condition is always valid. One way for this approach to become more practical would then be to extend it to systems where a discrepancy between nominal model and real dynamics is present, and to robustify the CBF condition against this modelling error.

  Future research on safety-critical MPC-based RL should focus on the integration of system identification procedures into the performance-seeking RL algorithms: in this way, the former can be used to infer reliable prediction models to enable safe control, while the latter can modify the rest of the MPC components for the sake of performance. To a limited degree, this idea was already put forth in [136, 137], but more effort should be dedicated to investigating how the two could work seamlessly at unison. For instance, one could rely on GP regression to obtain a stochastic prediction model that approaches the real dynamics, and then use RL to optimise not only the rest of the MPC parameters, but also the GP regression hyperparameters. Another avenue of research is to investigate how CBFs can be leveraged to impose safety at the update level rather than at the trajectory level. Briefly, the RL update could be constrained to a subset of the MPC parametrisation space for which the ensuing control policy always satisfies the CBF condition, therefore guaranteeing safety at all time steps. With some level of approximations

in place, this idea could be applied to the likely case in which the true dynamics are unknown and coupled with robust or probabilistic system identification to tighten the CBF constraint correspondingly.

- **Scalability.** The method proposed in Chapter 4 has two major limitations. From the point of view of the MPC computations, we assume the distributed optimisation problem to be convex. This is crucial for establishing convergence of ADMM to a KKT-optimal point, and in turn enabling the computations of the sensitivity. Unfortunately, this assumption confines the application of this methodology to a limited set of systems, e.g., with linear dynamics and convex constraints. On the side of learning, we only adapt Q-learning to our distributed setting, leaving unexplored how policy gradient methods could be employed instead. These methods are often preferable to their value-based counterparts, since they search directly for the policy that achieves maximum performance. Contrarily, value-based methods first attempt to fit the unknown optimal value function, and only then recover the optimal policy indirectly from it.

  Future efforts shall investigate how to solve these two issues. Extending the methodology to the broader class of nonconvex and nonlinear systems requires a more involved distributed optimisation strategy that still guarantees convergence to a local KKT point while preserving the distribution of the computations across a network of agents. See [192] for a good candidate solution that combines ADMM with sequential quadratic programming. Adapting policy-based methodologies to the multi-agent distributed MPC setting involves devising (possibly approximated) formulations of the policy gradients that can be computed in a way that does not require a central unit and preserves privacy, as achieved for Q-learning in Chapter 4. This is no easy feat, as usually there are inter-agent gradient dependencies, introducing further couplings that need to be disentangled [119, 230].

- **Applications.** The approaches provided in Chapters 5 and 6 need further improvement to foster their adoption in the plausible future. First of all, both methods lack validation in higher-fidelity simulation environments such as *SUMO* [124] and *GreenLight* [206, 207], respectively. The scope of proposed methodologies should also be expanded; in reality, traffic and climate control systems are much more complex and enjoy an array of other actuation means that were not included in these chapters. For instance, ramp metering is often accompanied by Variable Speed Limits (VSLs), displays that inform drivers on the suggested or mandatory (depending on the country) driving speed. Since these VSLs can only pick these speeds from a discrete set of available choices, the ensuing MPC optimisation becomes a mixed-integer problem. This kind of problem is notoriously more expensive to solve, and the sensitivity computations require further considerations due to the presence of integer variables [77]. Extensions to a larger pool of actuators and more complex traffic and crop models should be considered in the future. Lastly, both methods assume perfect knowledge of the current state and of disturbance/demand forecasts. Obviously, this is a strong assumption in real-world scenarios, and the controller must be enhanced with some estimation

algorithm to take care of these forecasts. The application of RL to such estimation-control architectures becomes inherently more challenging [59, 61], but is a key step to apply the proposed methodologies to real-life settings.

- **Global optimisation.** While Chapter 7 lays the foundations for model-based RL in global optimisation with deterministic surrogate models, the formal analysis of the methodology is left for future work. This encompasses proofs on convergence and, possibly, guarantees on convergence rates, in line with what has been achieved for other algorithms [34, 104, 190]. Another improvement to the proposed methodology would be to exploit confidence bounds on the quality of the regression estimates (e.g., according to [226]) to characterise the uncertainty around the predicted function values at specific query points. This would improve predictions of the stochastic evolution of the surrogate object, which currently rely merely on a heuristic modelling of the uncertainty. Besides, extending the methodology to also handle discrete action spaces would surely render it more viable as, e.g., a hyperparameter optimisation tool, where often a large number of hyperparameters, such as number of hidden layers or neurons per layer in a neural network, number of clusters in K-means clustering, etc., are of a discrete nature.

Bibliography

[1] S. Abdufattokhov, M. Zanon and A. Bemporad. "Learning Lyapunov terminal costs from data for complexity reduction in nonlinear model predictive control". In: *International Journal of Robust and Nonlinear Control* 34.13 (2024), pp. 8676–8691.

[2] R. L. Abduljabbar, H. Dia and P.-W. Tsai. "Development and evaluation of bidirectional LSTM freeway traffic forecasting models using simulation data". In: *Scientific Reports* 11.1 (Dec. 2021), p. 23899.

[3] M. Abramowitz and I. A. Stegun, eds. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. 9th ed. New York, USA: Dover Publications, 1972, p. 890.

[4] F. Airaldi, B. De Schutter and A. Dabiri. "Learning safety in model-based reinforcement learning using MPC and Gaussian processes". In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 5759–5764.

[5] F. Airaldi, B. De Schutter and A. Dabiri. "Nonmyopic global optimisation via approximate dynamic programming". In preparation for submission to journal. 2025.

[6] F. Airaldi, B. De Schutter and A. Dabiri. "Probabilistically safe and efficient model-based reinforcement learning". Accepted to the 64th Conference on Decision and Control. 2025.

[7] F. Airaldi, B. De Schutter and A. Dabiri. "Reinforcement learning with model predictive control for highway ramp metering". In: *IEEE Transactions on Intelligent Transportation Systems* 26.5 (2025), pp. 5988–6004.

[8] A. Alan, T. G. Molnar, A. D. Ames and G. Orosz. "Parameterized barrier functions to guarantee safety under uncertainty". In: *IEEE Control Systems Letters* 7 (2023), pp. 2077–2082.

[9] M. Alqahtani, M. J. Scott and M. Hu. "Dynamic energy scheduling and routing of a large fleet of electric vehicles using multi-agent reinforcement learning". In: *Computers & Industrial Engineering* 169 (2022), p. 108180.

[10] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath and P. Tabuada. "Control barrier functions: Theory and applications". In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3420–3431.

[11] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl. "CasADi: A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[12] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath. “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[13] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson and E. Bakshy. “BoTorch: A framework for efficient Monte-Carlo Bayesian optimization”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. Vol. 33. Virtual: Curran Associates, Inc., 2020, pp. 21524–21538.

[14] T. Bellemans, B. De Schutter and B. De Moor. “Model predictive control for ramp metering of motorway traffic: A case study”. In: *Control Engineering Practice* 14.7 (2006), pp. 757–767.

[15] A. Bemporad. “Active learning for regression by inverse distance weighting”. In: *Information Sciences* 626 (2023), pp. 275–292.

[16] A. Bemporad. “Global optimization via inverse distance weighting and radial basis functions”. In: *Computational Optimization and Applications* 77.2 (2020), pp. 571–595.

[17] A. Bemporad and M. Morari. “Robust model predictive control: A survey”. In: *Robustness in identification and control*. Ed. by A. Garulli and A. Tesi. London: Springer, 1999, pp. 207–226.

[18] A. Bemporad, M. Morari, V. Dua and E. N. Pistikopoulos. “The explicit linear quadratic regulator for constrained systems”. In: *Automatica* 38.1 (2002), pp. 3–20.

[19] A. Bemporad and D. Piga. “Global optimization based on active preference learning with radial basis functions”. In: *Machine Learning* 110.2 (2021), pp. 417–448.

[20] D. P. Bertsekas. “Multiagent reinforcement learning: Rollout and policy iteration”. In: *IEEE/CAA Journal of Automatica Sinica* 8.2 (2021), pp. 249–272.

[21] D. P. Bertsekas and J. N. Tsitsiklis. “Neuro-dynamic programming: An overview”. In: *Proceedings of 1995 34th IEEE Conference on Decision and Control*. Vol. 1. 1995, pp. 560–564.

[22] D. Bertsekas. *Dynamic Programming and Optimal Control*. 4th ed. Vol. 1. Belmont, USA: Athena Scientific, 2017.

[23] D. Bertsekas. “Rollout algorithms and approximate dynamic programming for Bayesian optimization and sequential estimation”. In: *arXiv preprint arXiv:2212.07998* (2022).

[24] D. Bertsekas. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Belmont, USA: Athena Scientific, 2020.

[25] L. Blackmore, M. Ono, A. Bektassov and B. C. Williams. “A probabilistic particle-control approximation of chance-constrained stochastic predictive control”. In: *IEEE Transactions on Robotics* 26.3 (2010), pp. 502–517.

[26] X. Blascom, M. Martínez, J. M. Herrero, C. Ramos and J. Sanchis. “Model-based predictive control of greenhouse climate for reducing energy and water consumption”. In: *Computers and Electronics in Agriculture* 55.1 (2007), pp. 49–70.

[27] S. Boersma, C. Sun and S. van Mourik. “Robust sample-based model predictive control of a greenhouse system with parametric uncertainty”. In: *IFAC-PapersOnLine* 55.32 (2022), pp. 177–182.

[28] F. Borrelli, A. Bemporad and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge, UK: Cambridge University Press, 2017.

[29] P. Bouffard. “On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments”. MA thesis. EECS Department, University of California, Berkeley, Dec. 2012.

[30] S. Boyd. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends in Machine Learning* 3.1 (2010), pp. 1–122.

[31] M. Brand. “Fast low-rank modifications of the thin singular value decomposition”. In: *Linear Algebra and its Applications* 415.1 (2006). Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems, pp. 20–30.

[32] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati and A. P. Schoellig. “Safe learning in robotics: From learning-based control to safe reinforcement learning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), pp. 411–444.

[33] M. Bujarbaruah, X. Zhang, M. Tanaskovic and F. Borrelli. “Adaptive stochastic MPC under time-varying uncertainty”. In: *IEEE Transactions on Automatic Control* 66.6 (2021), pp. 2840–2845.

[34] A. D. Bull. “Convergence rates of efficient global optimization algorithms”. In: *Journal of Machine Learning Research* 12.88 (2011), pp. 2879–2904.

[35] C. Büskens and H. Maurer. “Sensitivity analysis and real-time optimization of parametric nonlinear programming problems”. In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke and J. Rambau. Berlin, Heidelberg: Springer, 2001, pp. 3–16.

[36] L. Busoniu, R. Babuska and B. De Schutter. “A comprehensive survey of multiagent reinforcement learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2 (2008), pp. 156–172.

[37] L. Busoniu, R. Babuska, B. De Schutter and D. Ernst. *Reinforcement Learning and Dynamic Programming using Function Approximators*. Boca Raton, USA: CRC press, 2017.

[38] R. H. Byrd, P. Lu, J. Nocedal and C. Zhu. “A limited memory algorithm for bound constrained optimization”. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[39] W. Cai, H. N. Esfahani, A. B. Kordabad and S. Gros. “Optimal management of the peak power penalty for smart grids using MPC-based reinforcement learning”. In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 6365–6370.

[40] W. Cai, A. B. Kordabad, H. N. Esfahani, A. M. Lekkas and S. Gros. “MPC-based reinforcement learning for a simplified freight mission of autonomous surface vehicles”. In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 2990–2995.

[41] W. Cai, A. B. Kordabad and S. Gros. “Energy management in residential microgrid using model predictive control-based reinforcement learning and Shapley value”. In: *Engineering Applications of Artificial Intelligence* 119 (2023), p. 105793.

[42] M. C. Campi and S. Garatti. “The exact feasibility of randomized solutions of uncertain convex programs”. In: *SIAM Journal on Optimization* 19.3 (2008), pp. 1211–1230.

[43] M. C. Campi, S. Garatti and M. Prandini. “Scenario optimization for MPC”. In: *Handbook of Model Predictive Control*. Ed. by S. V. Raković and W. S. Levine. Cham, Switzerland: Springer International Publishing, 2019, pp. 445–463.

[44] K. Castañeda, O. Sánchez, R. F. Herrera and G. Mejía. “Highway planning trends: A bibliometric analysis”. In: *Sustainability* 14.9 (2022), p. 5544.

[45] J. Chen, W. Lin, Z. Yang, J. Li and P. Cheng. “Adaptive ramp metering control for urban freeway using large-scale data”. In: *IEEE Transactions on Vehicular Technology* 68.10 (2019), pp. 9507–9518.

[46] W. H. Chen and F. You. “Data-driven robust optimization for greenhouse temperature control using model predictive control”. In: *CET Journal-Chemical Engineering Transactions* 81 (2020).

[47] D. Chmielewski and V. Manousiouthakis. “On constrained infinite-time linear quadratic optimal control”. In: *Systems & Control Letters* 29.3 (1996), pp. 121–129.

[48] A. Clark. “Control barrier functions for complete and incomplete information stochastic systems”. In: *2019 American Control Conference (ACC)*. 2019, pp. 2928–2935.

[49] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*. Third. Cambridge, USA: MIT Press, 2009.

[50] A. Costa and G. Nannicini. “RBFOpt: An open-source library for black-box optimization with costly function evaluations”. In: *Mathematical Programming Computation* 10.4 (2018), pp. 597–629.

[51] L. Csató and M. Opper. “Sparse on-line Gaussian processes”. In: *Neural Computation* 14.3 (2002), pp. 641–668.

[52] C. F. O. da Silva, A. Dabiri and B. De Schutter. “Integrating reinforcement learning and model predictive control with applications to microgrids”. In: *arXiv preprint arXiv:2409.11267* (2024).

[53] A. Dabiri and B. Kulcsár. “Distributed ramp metering—A constrained discharge flow maximization approach”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.9 (2017), pp. 2525–2538.

[54] M. Davarynejad, A. Hegyi, J. Vrancken and J. van den Berg. “Motorway ramp-metering control with queuing consideration using Q-learning”. In: *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1652–1658.

[55] B. De Schutter and B. De Moor. “Optimal traffic light control for a single intersection”. In: *European Journal of Control* 4.3 (1998), pp. 260–276.

[56] M. Degner, R. Soloperto, M. N. Zeilinger, J. Lygeros and J. Köhler. “Adaptive economic model predictive control for linear systems with performance guarantees”. In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. 2024, pp. 7490–7496.

[57] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal and T. Hester. “Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis”. In: *Machine Learning* 110.9 (Sept. 2021), pp. 2419–2468.

[58] T. Elsken, J. H. Metzen and F. Hutter. “Neural Architecture Search: A Survey”. In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21.

[59] H. N. Esfahani, A. B. Kordabad, W. Cai and S. Gros. “Learning-based state estimation and control using MHE and MPC schemes with imperfect models”. In: *European Journal of Control* 73 (2023), p. 100880.

[60] H. N. Esfahani, A. B. Kordabad and S. Gros. “Approximate robust NMPC using reinforcement learning”. In: *2021 European Control Conference (ECC)*. 2021, pp. 132–137.

[61] H. N. Esfahani, A. B. Kordabad and S. Gros. “Reinforcement learning based on MPC/MHE for unmodeled and partially observable dynamics”. In: *2021 American Control Conference (ACC)*. 2021, pp. 2121–2126.

[62] A. Fares and W. Gomaa. “Freeway ramp-metering control based on reinforcement learning”. In: *11th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2014, pp. 1226–1231.

[63] F. Fele, E. Debada, J. M. Maestre and E. F. Camacho. “Coalitional control for self-organizing agents”. In: *IEEE Transactions on Automatic Control* 63.9 (2018), pp. 2883–2897.

[64] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli and S. Whiteson. “Stabilising experience replay for deep multi-agent reinforcement learning”. In: *International Conference on Machine Learning. PMLR*. 2017, pp. 1146–1155.

[65] M. Forgione, D. Piga and A. Bemporad. “Efficient calibration of embedded MPC”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 5189–5194.

[66] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi and M. Pontil. “Bilevel programming for hyperparameter optimization and meta-learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 1568–1577.

[67] P. I. Frazier, W. B. Powell and S. Dayanik. “A knowledge-gradient policy for sequential information collection”. In: *SIAM Journal on Control and Optimization* 47.5 (2008), pp. 2410–2439.

[68] J. R. D. Frejo and B. De Schutter. “Feed-forward ALINEA: A ramp metering control algorithm for nearby and distant bottlenecks”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.7 (2019), pp. 2448–2458.

[69] Y. Gao, K. H. Johansson and L. Xie. “Computing probabilistic controlled invariant sets”. In: *IEEE Transactions on Automatic Control* 66.7 (2021), pp. 3138–3151.

[70] J. García and F. Fernández. “A comprehensive survey on safe reinforcement learning”. In: *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480.

[71] F. García-Mañas, F. Rodríguez, M. Berenguel and J. M. Maestre. “Multi-scenario model predictive control for greenhouse crop production considering market price uncertainty”. In: *IEEE Transactions on Automation Science and Engineering* 21.3 (2024), pp. 2936–2948.

[72] R. Garnett. *Bayesian Optimization*. Cambridge, UK: Cambridge University Press, 2023.

[73] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams and S. Levine. “Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 25502–25515.

[74] C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao and W. Liu. “A survey on interpretable reinforcement learning”. In: *Machine Learning* 113.8 (Aug. 2024), pp. 5847–5890.

[75] S. Gros and M. Zanon. “Data-driven economic NMPC using reinforcement learning”. In: *IEEE Transactions on Automatic Control* 65.2 (2020), pp. 636–648.

[76] S. Gros and M. Zanon. “Learning for MPC with stability & safety guarantees”. In: *Automatica* 146 (2022), p. 110598.

[77] S. Gros and M. Zanon. “Reinforcement learning for mixed-integer problems based on MPC”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 5219–5224.

[78] J. K. Gruber, J. L. Guzmán, F. Rodríguez, C. Bordons, M. Berenguel and J. A. Sánchez. “Nonlinear MPC based on a Volterra series model for greenhouse temperature control using natural ventilation”. In: *Control Engineering Practice* 19.4 (2011), pp. 354–366.

[79] C. Gu, T. Zhou and C. Wu. “Deep Koopman traffic modeling for freeway ramp metering”. In: *IEEE Transactions on Intelligent Transportation Systems* 24.6 (2023), pp. 6001–6013.

[80] M. L. Gullino, R. Albajes and P. C. Nicot, eds. *Integrated Pest and Disease Management in Greenhouse Crops*. 2nd. Vol. 9. Plant Pathology in the 21st Century. Cham, Switzerland: Springer, 2020.

[81] J. K. Gupta, M. Egorov and M. Kochenderfer. “Cooperative multi-agent control using deep reinforcement learning”. In: *Autonomous Agents and Multiagent Systems* (2017), pp. 66–83.

[82] L. Gurobi Optimization. *Gurobi optimizer reference manual*. 2025. URL: https://www.gurobi.com.

[83] H.-M. Gutmann. “A radial basis function method for global optimization”. In: *Journal of Global Optimization* 19.3 (2001), pp. 201–227.

[84] A. Hamza, M. Ramdani and W. Bougheloum. “Robust T-S fuzzy constrained predictive control design for greenhouse micro-climate”. In: *International Journal of Scientific Research Engineering Technology* 13 (2019), pp. 1–4.

[85] Y. Han, M. Ramezani, A. Hegyi, Y. Yuan and S. Hoogendoorn. “Hierarchical ramp metering in freeways: An aggregated modeling and control approach”. In: *Transportation Research Part C: Emerging Technologies* 110 (2020), pp. 1–19.

[86] Y. Han, M. Wang, L. Li, C. Roncoli, J. Gao and P. Liu. “A physics-informed reinforcement learning-based strategy for local and coordinated ramp metering”. In: *Transportation Research Part C: Emerging Technologies* 137 (2022), p. 103584.

[87] K. He, S. Shi, T. van den Boom and B. De Schutter. “Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach”. In: *Automatica* 160 (2024), p. 111456.

[88] A. Hegyi. “Model predictive control for integrating traffic control measures”. PhD thesis. Delft, The Netherlands: Delft University of Technology, 2004.

[89] A. Hegyi, B. De Schutter, H. Hellendoorn and T. van den Boom. “Optimal coordination of ramp metering and variable speed control-An MPC approach”. In: *2002 American Control Conference (ACC)*. Vol. 5. 2002, pp. 3600–3605.

[90] A. Hegyi, B. De Schutter and H. Hellendoorn. “Model predictive control for optimal coordination of ramp metering and variable speed limits”. In: *Transportation Research Part C: Emerging Technologies* 13.3 (2005), pp. 185–209.

[91] L. Hewing, J. Kabzan and M. N. Zeilinger. “Cautious model predictive control using Gaussian process regression”. In: *IEEE Transactions on Control Systems Technology* 28.6 (2020), pp. 2736–2743.

[92] L. Hewing, K. P. Wabersich, M. Menner and M. N. Zeilinger. “Learning-based model predictive control: Toward safe learning in control”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (2020).

[93] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker and C. S. Woodward. “SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers”. In: *ACM Transactions on Mathematical Software* 31.3 (2005), pp. 363–396.

[94] E. Ivanjko, D. Koltovska Nečoska, M. Gregurić, M. Vujić, G. Jurković and S. Mandžuka. “Ramp metering control based on the Q-learning algorithm”. In: *Cybernetics and Information Technologies* 15.5 (2015), pp. 88–97.

[95] M. Jamil and X. S. Yang. “A literature survey of benchmark functions for global optimisation problems”. In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194.

[96] S. Jiang, H. Chai, J. Gonzalez and R. Garnett. “BINOCULARS for efficient, nonmyopic sequential experimental design”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. Daumé III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. Virtual: PMLR, 2020, pp. 4794–4803.

[97] S. Jiang, D. Jiang, M. Balandat, B. Karrer, J. Gardner and R. Garnett. “Efficient nonmyopic Bayesian optimization via one-shot multi-step trees”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. Vol. 33. Virtual: Curran Associates, Inc., 2020, pp. 18039–18049.

[98] O. Johansson, D. Pearce and D. Maddison. *Blueprint 5: True Costs of Road Transport*. London, UK: Routledge, 2014.

[99] D. R. Jones, M. Schonlau and W. J. Welch. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[100] V. R. Joseph and L. Kang. “Regression-based inverse distance weighting with applications to computer experiments”. In: *Technometrics* 53.3 (2011), pp. 254–265.

[101] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine and M. Bennamoun. “Hands-on Bayesian neural networks—A tutorial for deep learning users”. In: *IEEE Computational Intelligence Magazine* 17.2 (2022), pp. 29–48.

[102] N. Karnchanachari, M. de la Iglesia Valls, D. Hoeller and M. Hutter. “Practical reinforcement learning for MPC: Learning from sparse objectives in under an hour on a real robot”. In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Ed. by A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin and M. Zeilinger. Vol. 120. Proceedings of Machine Learning Research. PMLR, 2020, pp. 211–224.

[103] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun and D. Scaramuzza. “Champion-level drone racing using deep reinforcement learning”. In: *Nature* 620.7976 (Aug. 2023), pp. 982–987.

[104] K. Kawaguchi, L. P. Kaelbling and T. Lozano-Pérez. “Bayesian optimization with exponential convergence”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[105] F. L. K. Kempkes, J. Janse and S. Hemming. “Greenhouse concept with high insulating double glass with coatings and new climate control strategies; from design to results from tomato experiments”. In: *International Symposium on New Technologies for Environment Control, Energy-Saving and Crop Production in Greenhouse and Plant*. 2013, pp. 83–92.

[106] D. P. Kingma and M. Welling. “Auto-encoding variational Bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[107] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani and P. Pérez. “Deep reinforcement learning for autonomous driving: A survey”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926.

[108] J. Köhler, E. Andina, R. Soloperto, M. A. Müller and F. Allgöwer. “Linear robust adaptive model predictive control: Computational complexity and conservatism”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 1383–1388.

[109] T. Koller, F. Berkenkamp, M. Turchetta and A. Krause. “Learning-based model predictive control for safe exploration”. In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 6059–6066.

[110] A. B. Kordabad, H. N. Esfahani, A. M. Lekkas and S. Gros. “Reinforcement learning based on scenario-tree MPC for ASVs”. In: *2021 American Control Conference (ACC)*. 2021, pp. 1985–1990.

[111] D. Krishnamoorthy and F. J. Doyle. “Safe Bayesian optimization using interior-point methods—Applied to personalized insulin dose guidance”. In: *IEEE Control Systems Letters* 6 (2022), pp. 2834–2839.

[112] W. J. P. Kuijpers, D. J. Antunes, S. van Mourik, E. J. van Henten and M. van de Molengraft. “Weather forecast error modelling and performance analysis of automatic greenhouse climate control”. In: *Biosystems Engineering* 214 (2022), pp. 207–229.

[113] F. Lafont, N. Pessel, J. F. Balmat and M. Fliess. “On the model-free control of an experimental greenhouse”. In: *International Conference on Modeling, Simulation and Control*. San Francisco, United States, 2013.

[114] M. G. Lagoudakis, R. Parr and M. L. Littman. “Least-squares methods in reinforcement learning for control”. In: *Methods and Applications of Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 249–260.

[115] R. Lam, K. Willcox and D. H. Wolpert. “Bayesian optimization with a finite budget: An approximate dynamic programming approach”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett. Vol. 29. Barcelona, Spain: Curran Associates, Inc., 2016, pp. 883–891.

[116] E. Lee, D. Eriksson, D. Bindel, B. Cheng and M. Mccourt. “Efficient rollout strategies for Bayesian optimization”. In: *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*. Ed. by J. Peters and D. Sontag. Vol. 124. Proceedings of Machine Learning Research. Virtual: PMLR, 2020, pp. 260–269.

[117] E. H. Lee, D. Eriksson, V. Perrone and M. Seeger. “A nonmyopic approach to cost-constrained Bayesian optimization”. In: *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence*. Ed. by C. de Campos and M. H. Maathuis. Vol. 161. Proceedings of Machine Learning Research. Virtual: PMLR, 2021, pp. 568–577.

[118] F. L. Lewis, D. Vrabie and V. L. Syrmos. *Optimal Control*. New York, USA: John Wiley & Sons, 2012.

[119] Y. Li, Y. Tang, R. Zhang and N. Li. "Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach". In: *IEEE Transactions on Automatic Control* 67.12 (2022), pp. 6429–6444.

[120] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[121] A. E. B. Lim, J. B. Moore and L. Faybusovich. "Separation theorem for linearly constrained LQG optimal control". In: *Systems & Control Letters* 28.4 (1996), pp. 227–235.

[122] L. J. Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine Learning* 8.3 (1992), pp. 293–321.

[123] S. Liu, A. Sadowska and B. De Schutter. "A scenario-based distributed model predictive control approach for freeway networks". In: *Transportation Research Part C: Emerging Technologies* 136 (2022), p. 103261.

[124] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner and E. Wiessner. "Microscopic traffic simulation using SUMO". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2575–2582.

[125] D. Lowe and D. Broomhead. "Multivariable functional interpolation and adaptive networks". In: *Complex Systems* 2.3 (1988), pp. 321–355.

[126] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel and I. Mordatch. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in Neural Information Processing Systems* 30 (2017).

[127] Q. Lu, R. Kumar and V. M. Zavala. "MPC controller tuning using Bayesian optimization techniques". In: *arXiv preprint arXiv:2009.14175* (2021).

[128] J. Luedtke and S. Ahmed. "A sample approximation approach for optimization with probabilistic constraints". In: *SIAM Journal on Optimization* 19.2 (2008), pp. 674–699.

[129] X. Ma, A. Karimpour and Y.-J. Wu. "Statistical evaluation of data requirement for ramp metering performance assessment". In: *Transportation Research Part A: Policy and Practice* 141 (2020), pp. 248–261.

[130] J. M. Maestre and R. R. Negenborn, eds. *Distributed Model Predictive Control made easy*. Dordrecht, The Netherlands: Springer, 2014.

[131] F. Mahmood, R. Govindan, A. Bermak, D. Yang and T. Al-Ansari. "Data-driven robust model predictive control for greenhouse temperature control and energy utilisation assessment". In: *Applied Energy* 343 (2023), p. 121190.

[132] F. Mahmood, R. Govindan, A. Bermak, D. Yang and T. Al-Ansari. “Greenhouse temperature regulation in the presence of uncertainties using data-driven robust model predictive control”. In: *33rd European Symposium on Computer Aided Process Engineering*. Ed. by A. C. Kokossis, M. C. Georgiadis and E. Pistikopoulos. Vol. 52. Computer Aided Chemical Engineering. Elsevier, 2023, pp. 1591–1596.

[133] G. Malkomes and R. Garnett. “Automating Bayesian optimization with Bayesian optimization”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett. Vol. 31. Montréal, Canada: Curran Associates, Inc., 2018.

[134] S. Mallick, F. Airaldi, A. Dabiri and B. De Schutter. “Multi-agent reinforcement learning via distributed MPC as a function approximator”. In: *Automatica* 167 (2024), p. 111803.

[135] S. Mallick, F. Airaldi, A. Dabiri, C. Sun and B. De Schutter. “Reinforcement learning-based model predictive control for greenhouse climate control”. In: *Smart Agricultural Technology* 10 (2025), p. 100751.

[136] A. B. Martinsen, A. M. Lekkas and S. Gros. “Combining system identification with reinforcement learning-based MPC”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 8130–8135.

[137] A. B. Martinsen, A. M. Lekkas and S. Gros. “Reinforcement learning-based NMPC for tracking control of ASVs: Theory and experiments”. In: *Control Engineering Practice* 120 (2022), p. 105024.

[138] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814.

[139] D. B. McDonald, W. J. Grantham, W. L. Tabor and M. J. Murphy. “Global and local optimization using radial basis function response surface models”. In: *Applied Mathematical Modelling* 31.10 (2007), pp. 2095–2110.

[140] A. Mesbah. “Stochastic model predictive control: An overview and perspectives for future research”. In: *IEEE Control Systems Magazine* 36.6 (2016), pp. 30–44.

[141] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell and J. A. Paulson. “Fusion of machine learning and MPC under uncertainty: What advances are on the horizon?” In: *2022 American Control Conference (ACC)*. 2022, pp. 342–357.

[142] A. Messner and M. Papageorgiou. “METANET: A macroscopic simulation program for motorway networks”. In: *Traffic Engineering & Control* 31.8-9 (1990), pp. 466–470.

[143] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller. “Playing Atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[144] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[145] T. M. Moerland, J. Broekens, A. Plaat and C. M. Jonker. "Model-based reinforcement learning: A survey". In: *Foundations and Trends in Machine Learning* 16.1 (2023), pp. 1–118.

[146] A. P. Montoya, J. L. Guzmán, F. Rodríguez and J. A. Sánchez-Molina. "A hybrid-controlled approach for maintaining nocturnal greenhouse temperature: Simulation study". In: *Computers and Electronics in Agriculture* 123 (2016), pp. 116–124.

[147] B. Morcego, W. Yin, S. Boersma, E. J. van Henten, V. Puig and C. Sun. "Reinforcement learning versus model predictive control on greenhouse climate control". In: *Computers and Electronics in Agriculture* 215 (2023), p. 108372.

[148] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar and M. Püschel. "A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions". In: *arXiv:1112.2295* (2018).

[149] A. A. D. Nascimento, A. Papachristodoulou and K. Margellos. "Probabilistically safe controllers based on control barrier functions and scenario model predictive control". In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. 2024, pp. 1814–1819.

[150] D. Ngoduy and S. P. Hoogendoorn. "An automated calibration procedure for macroscopic traffic flow models". In: *IFAC Proceedings Volumes* 36.14 (2003), pp. 263–268.

[151] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. NewYork, USA: Springer, 2006.

[152] R. Olfati-Saber, J. A. Fax and R. M. Murray. "Consensus and cooperation in networked multi-agent systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 215–233.

[153] W. H. Organization, ed. *Global Status Report on Road Safety*. Geneva, Switzerland: World Health Organization, 2018.

[154] G. Pagès. *Numerical Probability: An Introduction with Applications to Finance*. Cham, Switzerland: Springer, 2018.

[155] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos and Y. Wang. "Review of road traffic control strategies". In: *Proceedings of the IEEE* 91.12 (2003), pp. 2043–2067.

[156] M. Papageorgiou, J.-M. Blosseville and H. Hadj-Salem. "Macroscopic modelling of traffic flow on the Boulevard Périphérique in Paris". In: *Transportation Research Part B: Methodological* 23.1 (1989), pp. 29–47.

[157] M. Papageorgiou, H. Hadj-Salem and J.-M. Blosseville. “ALINEA: A local feedback control law for on-ramp metering”. In: *Transportation Research Record* 1320.1 (1991), pp. 58–64.

[158] M. Papageorgiou and A. Kotsialos. “Freeway ramp metering: An overview”. In: *IEEE Transactions on Intelligent Transportation Systems* 3.4 (2002), pp. 271–281.

[159] M. Papageorgiou, I. Papamichail, A. Messmer and Y. Wang. “Traffic simulation with METANET”. In: *Fundamentals of Traffic Simulation*. Ed. by J. Barceló. New York: Springer, 2010, pp. 399–430.

[160] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox and R. Garnett. Vol. 32. Long Beach, USA: Curran Associates, Inc., 2019, pp. 8026–8037.

[161] D. Piga, M. Forgione, S. Formentin and A. Bemporad. “Performance-oriented model learning for data-driven MPC design”. In: *IEEE Control Systems Letters* 3.3 (2019), pp. 577–582.

[162] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Cambridge, USA: MIT Press, Nov. 2005.

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

[164] R. G. Regis and C. A. Shoemaker. “Constrained global optimization of expensive black box functions using radial basis functions”. In: *Journal of Global Optimization* 31.1 (2005), pp. 153–171.

[165] R. Reiter, J. Hoffmann, D. Reinhardt, F. Messerer, K. Baumgärtner, S. Sawant, J. Boedecker, M. Diehl and S. Gros. “Synthesis of model predictive control and reinforcement learning: Survey and classification”. In: *arXiv:1207.2000* (2025).

[166] W. Remmerswaal, D. Sun, A. Jamshidnejad and B. De Schutter. “Combined MPC and reinforcement learning for traffic signal control in urban traffic networks”. In: *2022 26th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2022, pp. 432–439.

[167] S. Riverso and G. Ferrari-Trecate. “Hycon2 benchmark: Power network system”. In: *arXiv:1207.2000* (2012).

[168] A. Romero, Y. Song and D. Scaramuzza. “Actor-critic model predictive control”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 14777–14784.

[169] U. Rosolia and F. Borrelli. “Learning model predictive control for iterative tasks. A data-driven control framework”. In: *IEEE Transactions on Automatic Control* 63.7 (2018), pp. 1883–1896.

[170] U. Rosolia, X. Zhang and F. Borrelli. “Data-driven predictive control for autonomous systems”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 259–286.

[171] E. Sabouni, H. M. Ahmad, V. Giammarino, C. G. Cassandras, I. C. Paschalidis and W. Li. “Reinforcement learning-based receding horizon control using adaptive control barrier functions for safety-critical systems”. In: *2024 IEEE Conference on Decision and Control (CDC)*. 2024, pp. 401–406.

[172] G. Schildbach, L. Fagiano, C. Frei and M. Morari. “The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations”. In: *Automatica* 50.12 (2014), pp. 3009–3018.

[173] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert and M. Toussaint. “Safe exploration for active learning with Gaussian processes”. In: *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2015, pp. 133–149.

[174] K. Seel, A. B. Kordabad, S. Gros and J. T. Gravdahl. “Convex neural network-based cost modifications for learning model predictive control”. In: *IEEE Open Journal of Control Systems* 1 (2022), pp. 366–379.

[175] I. Seginer, C. Gary and M. Tchamitchian. “Optimal temperature regimes for a greenhouse crop with a carbohydrate pool: A modelling study”. In: *Scientia Horticulturae* 60.1-2 (1994), pp. 55–80.

[176] K. Shaaban, M. A. Khan and R. Hamila. “Literature review of advancements in adaptive ramp metering”. In: *Procedia Computer Science* 83 (2016), pp. 203–211.

[177] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. de Freitas. “Taking the human out of the loop: A review of Bayesian optimization”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[178] S. Shi, A. Tsiamis and B. De Schutter. “Suboptimality analysis of receding horizon quadratic control with unknown linear systems and its applications in learning-based control”. In: *arXiv preprint arXiv:2301.07876* (2023).

[179] D. Silver, T. Hubert, J. Schrittwieser, L. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[180] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra and M. Riedmiller. “Deterministic policy gradient algorithms”. In: *International Conference on Machine Learning*. *PMLR*. 2014, pp. 387–395.

[181] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel and D. Hassabis. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359.

[182] S. Siri, C. Pasquale, S. Sacone and A. Ferrara. “Freeway traffic control: A survey”. In: *Automatica* 130 (2021), p. 109655.

[183] E. Smaragdis, M. Papageorgiou and E. Kosmatopoulos. “A flow-maximizing adaptive local ramp metering strategy”. In: *Transportation Research Part B: Methodological* 38.3 (2004), pp. 251–270.

[184] A. J. Smola and B. Schölkopf. “A tutorial on support vector regression”. In: *Statistics and Computing* 14.3 (2004), pp. 199–222.

[185] E. Snelson, Z. Ghahramani and C. Rasmussen. “Warped Gaussian processes”. In: *Advances in Neural Information Processing Systems*. Vol. 16. MIT Press, 2003.

[186] J. Snoek, H. Larochelle and R. P. Adams. “Practical Bayesian optimization of machine learning algorithms”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou and K. Weinberger. Vol. 25. Lake Tahoe, USA: Curran Associates, Inc., 2012, pp. 2951–2959.

[187] Y. Song and D. Scaramuzza. “Policy search for model predictive control with application to agile drone flight”. In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2114–2130.

[188] F. Sorourifar, G. Makrygirgos, A. Mesbah and J. A. Paulson. “A data-driven automatic tuning method for MPC under uncertainty using constrained Bayesian optimization”. In: *IFAC-PapersOnLine* 54.3 (2021). 16th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2021, pp. 243–250.

[189] A. D. Spiliopoulou, D. Manolis, I. Papamichail, M. Papageorgiou, J. Wu and X. Jin. “Queue management techniques for metered freeway on-ramps”. In: *Transportation Research Record* 2178.1 (2010), pp. 40–48.

[190] N. Srinivas, A. Krause, S. Kakade and M. Seeger. “Gaussian process optimization in the bandit setting: No regret and experimental design”. In: *Proceedings of the 27th International Conference on Machine Learning*. Ed. by J. Fürnkranz and T. Joachims. Proceedings of Machine Learning Research. Haifa, Israel: Omnipress, 2010, pp. 1015–1022.

[191] C. Stamouli, A. Tsiamis, M. Morari and G. J. Pappas. “Adaptive stochastic MPC under unknown noise distribution”. In: *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*. Ed. by R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager and M. Kochenderfer. Vol. 168. Proceedings of Machine Learning Research. PMLR, June 2022, pp. 596–607.

[192] G. Stomberg, A. Engelmann and T. Faulwasser. “Decentralized non-convex optimization via bi-level SQP and ADMM”. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022, pp. 273–278.

[193] S. Subramanian, S. Lucia and S. Engell. “Handling structural plant-model mismatch via multi-stage nonlinear model predictive control”. In: *2015 European Control Conference (ECC)*. Linz, Austria, 2015, pp. 1602–1607.

[194] D. Sun, A. Jamshidnejad and B. De Schutter. “A novel framework combining MPC and deep reinforcement learning with application to freeway traffic control”. In: *IEEE Transactions on Intelligent Transportation Systems* 25.7 (2024), pp. 6756–6769.

[195] S. Surjanovic and D. Bingham. *Virtual library of simulation experiments: Test functions and datasets*. Retrieved July 19, 2025, from http://www.sfu.ca/~ssurjano.

[196] W. Suttle, Z. Yang, K. Zhang, Z. Wang, T. Başar and J. Liu. "A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning". In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 1549–1554.

[197] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, USA: MIT Press, 2018.

[198] J. L. Svensen, X. Cheng, S. Boersma and C. Sun. "Chance-constrained stochastic MPC of greenhouse production systems with parametric uncertainty". In: *Computers and Electronics in Agriculture* 217 (2024), p. 108578.

[199] S. El-Tantawy, B. Abdulhai and H. Abdelgawad. "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto". In: *IEEE Transactions on Intelligent Transportation Systems* 14.3 (2013), pp. 1140–1150.

[200] T. Tieleman. *rmsprop: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural networks for machine learning. 2012.

[201] U. Todorović, J. R. D. Frejo and B. De Schutter. "Distributed MPC for large freeway networks using alternating optimization". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022), pp. 1875–1884.

[202] M. Treiber and A. Kesting. *Traffic Flow Dynamics: Data, Models and Simulation*. Berlin, Heidelberg: Springer, 2013.

[203] M. Turchetta, F. Berkenkamp and A. Krause. "Safe exploration for interactive machine learning". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[204] E. J. van Henten. "Greenhouse climate management: An optimal control approach". PhD thesis. Wageningen, The Netherlands: Landbouwuniversiteit Wageningen, 1994.

[205] E. J. van Henten. "Sensitivity analysis of an optimal control problem in greenhouse climate management". In: *Biosystems Engineering* 85.3 (2003), pp. 355–364.

[206] B. van Laatum, E. J. van Henten and S. Boersma. "GreenLight-Gym: Reinforcement learning benchmark environment for control of greenhouse production systems". In: *arXiv preprint arXiv:2410.05336* (2025).

[207] B. H. E. Vanthoor. "A model-based greenhouse design method". PhD thesis. Wageningen, The Netherlands: Wageningen University, 2011.

[208] W. Vervaat. "A relation between Brownian bridge and Brownian excursion". In: *The Annals of Probability* (1979), pp. 143–149.

[209] F. Vrbanić, E. Ivanjko, K. Kušić and D. Čakija. "Variable speed limit and ramp metering for mixed traffic flows: A review and open questions". In: *Applied Sciences* 11.6 (2021).

[210] K. P. Wabersich, L. Hewing, A. Carron and M. N. Zeilinger. “Probabilistic model predictive safety certification for learning-based control”. In: *IEEE Trans. Autom. Control* 67.1 (2022), pp. 176–188.

[211] K. P. Wabersich and M. N. Zeilinger. “Cautious Bayesian MPC: Regret analysis and bounds on the number of unsafe learning episodes”. In: *IEEE Transactions on Automatic Control* 68.8 (2023), pp. 4896–4903.

[212] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106.1 (Mar. 2006), pp. 25–57.

[213] H. Wai, Z. Yang, Z. Wang and M. Hong. “Multi-agent reinforcement learning via double averaging primal-dual optimization”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[214] C. Wang, Y. Xu, J. Zhang and B. Ran. “Integrated traffic control for freeway recurrent bottleneck based on deep reinforcement learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.9 (2022), pp. 15522–15535.

[215] L. Wang, X. He and D. Luo. “Deep reinforcement learning for greenhouse climate control”. In: *2020 IEEE International Conference on Knowledge Graph (ICKG)*. 2020, pp. 474–480.

[216] Y. Wang, X. Shen and H. Qian. “Stochastic model predictive control with probabilistic control barrier functions and smooth sample-based approximation”. In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. 2024, pp. 4798–4803.

[217] Y. Wang, E. B. Kosmatopoulos, M. Papageorgiou and I. Papamichail. “Local ramp metering in the presence of a distant downstream bottleneck: Theoretical analysis and simulation study”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.5 (2014), pp. 2024–2039.

[218] Y. Wang and M. Papageorgiou. “Real-time freeway traffic state estimation based on extended Kalman filter: A general approach”. In: *Transportation Research Part B: Methodological* 39.2 (2005), pp. 141–167.

[219] Z. Wang and S. Jegelka. “Max-value entropy search for efficient Bayesian optimization”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. Sydney, Australia: PMLR, 2017, pp. 3627–3635.

[220] C. J. C. H. Watkins and P. Dayan. “Q-learning”. In: *Machine learning* 8 (1992), pp. 279–292.

[221] C. J. C. H. Watkins. “Learning from delayed rewards”. PhD thesis. Cambridge, UK: King’s College, 1989.

[222] J. A. Wattleworth. “Peak-period analysis and control of a freeway system”. In: *Highway Research Record* 157 (1965), pp. 1–21.

[223] J. C. Willems and J. W. Polderman. *Introduction to Mathematical Systems Theory: A Behavioral Approach*. Vol. 26. New York, USA: Springer, 1997.

[224] G. Williams, P. Drews, B. Goldfain, J. M. Rehg and E. A. Theodorou. "Aggressive driving with model predictive path integral control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1433–1440.

[225] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei and S.-H. Deng. "Hyperparameter optimization for machine learning models based on Bayesian optimization". In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40.

[226] Z.-m. Wu and R. Schaback. "Local error estimates for radial basis function interpolation of scattered data". In: *IMA Journal of Numerical Analysis* 13.1 (1993), pp. 13–27.

[227] W. Xiao, C. Belta and C. G. Cassandras. "Adaptive control barrier functions". In: *IEEE Transactions on Automatic Control* 67.5 (2022), pp. 2267–2281.

[228] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li and D. Rus. "BarrierNet: Differentiable control barrier functions for learning of safe robot control". In: *IEEE Transactions on Robotics* 39.3 (2023), pp. 2289–2307.

[229] D. Xu, S. Du and G. van Willigenburg. "Adaptive two time-scale receding horizon optimal control for greenhouse lettuce cultivation". In: *Computers and Electronics in Agriculture* 146 (2018), pp. 93–103.

[230] Y. Yan and Y. Shen. "Distributed policy gradient for linear quadratic networked control with limited communication range". In: *IEEE Transactions on Signal Processing* 72 (2024), pp. 2087–2100.

[231] S. Yu, M. Hirche, Y. Huang, H. Chen and F. Allgöwer. "Model predictive control for autonomous ground vehicles: a review". In: *Autonomous Intelligent Systems* 1.1 (Aug. 2021), p. 4.

[232] X. Yue and R. A. Kontar. "Why non-myopic Bayesian optimization is promising and how far should we look-ahead? A study via rollout". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. Virtual: PMLR, 2020, pp. 2808–2818.

[233] M. Zanon and S. Gros. "Safe reinforcement learning using robust MPC". In: *IEEE Transactions on Automatic Control* 66.8 (2021), pp. 3638–3652.

[234] J. Zeng, B. Zhang and K. Sreenath. "Safety-critical model predictive control with discrete-time control barrier function". In: *2021 American Control Conference (ACC)*. 2021, pp. 3882–3889.

[235] K. Zhang, Z. Yang, H. Liu, T. Zhang and T. Basar. "Fully decentralized multi-agent reinforcement learning with networked agents". In: *International Conference on Machine Learning. PMLR*. 2018, pp. 5872–5881.

[236] Q. Zhang, W. Pan and V. Reppa. "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2022), pp. 8770–8781.

[237] C. Zhu, R. H. Byrd, P. Lu and J. Nocedal. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on Mathematical Software* 23.4 (1997), pp. 550–560.

[238] M. Zhu, D. Piga and A. Bemporad. “C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration”. In: *IEEE Transactions on Control Systems Technology* 30.5 (2022), pp. 2176–2187.

Curriculum Vitæ

Filippo Airaldi

17-05-1995 Born in Savigliano, Italy

Education

2014–2017 BEng in Biomedical Engineering
Politecnico di Torino, Italy

2017–2019 MSc in Mechatronic Engineering
Politecnico di Torino, Italy
Thesis: Development of an operation library and proofs-of-concept for plug-in software within the framework of a new generation BIW assembly engineering tool
Promotor: Prof. M. Indri

2021–2025 PhD
Delft University of Technology
Thesis: Model-based reinforcement learning for predictive control and optimisation
Promotor: Dr. A. Dabiri & Prof. dr. ir. B. De Schutter

Work

2017 Biomedical Engineering Intern

Comau Innovation Center, Comau, Italy
2019–2020 R&D Software Development Intern

Engineering Enhancement Initiative, Comau, Italy
2020–2021 R&D Software Development Intern

Formula 1 Wind Tunnel, Ferrari, Italy
2021 Research Fellow

Wearable Robotics Lab, Scuola Superiore Sant'Anna, Italy

List of Publications

Journal Papers

4. **F. Airaldi**, B. De Schutter and A. Dabiri. “Nonmyopic global optimisation via approximate dynamic programming”. In preparation for submission to journal. 2025

3. **F. Airaldi**, B. De Schutter and A. Dabiri. “Reinforcement learning with model predictive control for highway ramp metering”. In: *IEEE Transactions on Intelligent Transportation Systems* 26.5 (2025), pp. 5988–6004

2. S. Mallick, **F. Airaldi**, A. Dabiri, C. Sun and B. De Schutter. “Reinforcement learning-based model predictive control for greenhouse climate control”. In: *Smart Agricultural Technology* 10 (2025), p. 100751

1. S. Mallick, **F. Airaldi**, A. Dabiri and B. De Schutter. “Multi-agent reinforcement learning via distributed MPC as a function approximator”. In: *Automatica* 167 (2024), p. 111803

Conference Proceedings

2. **F. Airaldi**, B. De Schutter and A. Dabiri. “Probabilistically safe and efficient model-based reinforcement learning”. Accepted to the 64th Conference on Decision and Control. 2025

1. **F. Airaldi**, B. De Schutter and A. Dabiri. “Learning safety in model-based reinforcement learning using MPC and Gaussian processes”. In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 5759–5764