BIOLOGICAL INSPIRED CONTROL AND MACHINE LEARNING FOR CLINICAL REHABILITATION AND ENGINEERING SYSTEMS

Thomas Doublein

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

JAMES WATT SCHOOL OF ENGINEERING

COLLEGE OF SCIENCE AND ENGINEERING

University of Glasgow

OCTOBER 2024

To my loving wife and family.

Abstract

Human quiet standing has been studied over the years in order to model controllers able to replicate variability and intermittency present in human control mechanisms. The Intermittent Control (IC) framework was proposed as a computational model and can be described by a serie of open-loop trajectories with close-loop triggering. However, the original implementation is based on a deterministic approach which requires knowledge of the underlying system's dynamics. This thesis explores the capabilities of a data-driven stochastic Intermitent Controller, using Gaussian Processes (GP), applied mainly to a Single Inverted Pendulum (SIP).

Throughout this thesis, the Intermitent Controller framework has been adapted to move toward a data-driven intermittent controller using Reinforcement Learning (RL) and Data Informativity (DI) to estimate the state feedback gains. Simulations show the benefit of the open-loop trajectories created by IC in improving the overall estimation of the parameters, compared to Continuous Controllers. These results are the initial basis for a deterministic data-driven Intermittent Controller. In addition, the integration of GPs within this IC framework is able to introduce varibility in the generated control input by using probabilistic open-loop trajectories. Both these implementations have been combined to create the first data-driven stochastic intermittent controller. Results presented in this work are showing the capability of this newly controller approach to handle adaptation as well as switching between deterministic like behavior to fully probabilistic characteristics based on IC and GP parameters.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my deepest appreciation to Dr. Henrik Gollee who generously provided expertise and support on multiple subjects, especially Intermittent Control. His invaluable feedback has been a great help to ensure completion of this project. Words cannot express my gratitude to Dr. J. Alberto Álvarez Martín for his amazing support and knowledge about control. He was always available to brainstorm ideas, debugging code or dealing with Git issues, and he always kept me motivated during rough time. I also wanted to thank all members of the Center of Rehabilitation Engineering.

To Elizabeth, my heartfelt thanks go out to you for your unwavering support and constant encouragement. Your dedication and belief in me have been instrumental in helping me complete my PhD. A special thank you to my parents and siblings for all the support and kind words when needed. I also would like to recognize Wallace and our newly added family member, Leonard, for their endless entertainment and emotional support.

Finally, I am grateful to the Engineering and Physical Sciences Research Council (EPSRC) for their financial support, which was crucial in enabling the completion of this project.

Declaration

I declare that, except where explicit reference is made to the contribution of others, that this dissertation is the result of my own work and has not been submitted for any other degree at the University of Glasgow or any other institution

Thomas Doublein

Chapter 1

Introduction

In control theory, open-loop control fully relies on the known dynamics of the systems. As the output of the system is not assessed, it is not possible to compensate for any internal or external disturbances. In contrast, closed-loop control, also known as feedback control, compares the system's response to the reference to control the system accordingly and compensate for any disturbance.

In contrast to the open and closed-loop approach, Intermittent Control (IC) uses a sequence of open-loop trajectories with intermittent feedback. This implementation can be widely applied to different systems such as biomedical applications. The trigger mechanism present in IC closing the feedback loop allows to switch between an exploration and an exploitation phase. In the IC implementation by Gawthrop and Wang 2009, the open-loop trajectories are generated by a deterministic closed-loop representation of the plant.

In the idea of designing a controller to minimize the error between the set point and the actual output of the system, the term ”optimal control” was introduced in the late 1950s. One approach to this problem has been developed by Richard Bellman called *dynamic programming* and is based on a nineteenth-century theory of Hamilton and Jacobi (Peng 1992). This approach has been applied to Continuous Controllers as well as the Intermittent Controllers.

Intermittent Control has been used to answer some of the questions in human motor control such as balancing tasks and quiet standing (Craik 1947; Gawthrop 2009; Gawthrop and Wang 2009; Gawthrop et al. 2011; Loram et al. 2011; Gawthrop et al. 2013, 2014a; Zgonnikov and Lubashevsky 2015; Michimoto et al. 2016). Studies on human balancing have shown that similar quiet standing dynamics behavior can be found in an inverted pendulum system (Winter et al. 1998; Gatev et al. 1999). Previous research (Loram et al. 2011) has proven the robustness of IC in the case of pendulum balancing using tapping control. It is now used to understand and learn how human control works, for example after a Spinal Cord Injury.

Following Spinal Cord Injury (SCI), the rehabilitation can be quite long and the human body has to relearn basic tasks such as walking. However, 80% of patients can become independent ambulators after this kind of injury (Burns et al. 1997). Since rehabilitation is a process of relearning, it provides a great base for understanding the human learning process and designing novel rehabilitation approaches following SCI. Learning, relearning, or adaptation to new physical properties has been studied in the context of control (Martín 2018; Martín et al. 2021) using recent approaches such as Machine Learning and Reinforcement Learning (Renaudo et al. 2015).

Artificial Intelligence (AI), Machine Learning (ML), and more specifically Reinforcement Learning (RL) have been used in the context of control theory to map a specific state to an action. Moreover, this mapping is learned by RL algorithms and can adapt itself to deal with situations where the system's dynamics evolve through time. RL is based on the idea of dynamical systems theory, such as *dynamic programming*, specifically in the context of optimal control of incompletely-known Markov decision processes (Strehl and Littman 2008; Van Otterlo and Wiering 2012; Ghavamzadeh et al. 2015).

1.1 Aims and Objectives

The main aim of this research was to investigate the potential of data-driven techniques applied to the IC framework for rehabilitation and engineering systems purposes.

The detailed aims are:

- (1) to evaluate the robustness of IC with system identification techniques applied to systems with unknown dynamics. As the trigger mechanism allows us to explore the data space when desired, while exploiting feedback when needed, it is expected that the switching between open and closed-loop regimes would improve learning overall, leading to better control performances.
- (2) to explore the potential of probabilistic modeling integrated within the Intermittent Control framework relying on Gaussian Processes. The expectation was to achieve a more accurate representation of human control, focusing on incremental relearning behavior while keeping variability.

The first objective was to adapt the intermittent control framework to be used with any system identification techniques, hence being able to test multiple algorithms using the same conditions, as well as the capabilities of integrating any type of generalized hold.

The second objective was to adapt and implement algorithms taking into consideration the presence of events in the Intermittent Control while still performing accurately. The implementation was based on occlusion techniques, a time-varying window for data collection, and a triggering mechanism in IC.

The last objective was to evaluate and validate the robustness of a data-driven Intermittent Control on systems with unknown dynamics in simulation, to then apply it within a physics engine, with limited knowledge of the plant.

1.1.1 Approach

In this section, the approach used to reach the aims and objectives is described. In its original form, Intermittent Control relies on the knowledge of the plant's dynamics in order to be designed appropriately. When dealing with systems where the dynamics are partially or completely unknown, a data-driven approach can be considered more appropriate. This is explored in this work, with implementations tested against a Single Inverted Pendulum as well as the Cartpole system.

The first step is to implement and/or adapt available data-driven algorithms to fit within the Intermittent Control framework. Two algorithms, Reinforcement Learning (RL), with policy iteration algorithm and Data Informativity (DI) have been adapted and compared to assess accuracy and limitations in estimating the current plant. Reinforcement Learning being a direct method, is estimating the feedback gain directly without the need to estimate the system matrices first, as is the case with Data Informativity.

The second step is to evaluate the potential of Gaussian Processes (GP) within the Intermittent Controller to replace the traditional System-Matched Hold (SMH) which respresents and internal model of the controlled systems. This allows the implementation of a non-linear probabilistic controller.

By combining GP and RL/DI, it is possible to get an accurate model representing the system's dynamics as well as accurate state feedback which includes non-linear stochastic and probabilistic components. This fully data-driven approach allows using the Intermittent Control with time-varying plants.

1.1.2 Thesis contributions

The main contribution is that, for the time, a fully data-driven and stochastic Intermittent Control system has been implemented. This implementation is divided into two parts, using two different algorithms:

- Integration of a probabilistic hold (Gaussian processes) to replace the traditional deterministic hold (SMH). The results show the capability of Gaussian Processes to model system dynamics with data generated from an Intermittent Controller. In addition, by tweaking the number of points used by the GP or the time between each retraining, experiments have shown the capability of varying variability without the need for external disturbance such as noise.
- Integration of Reinforcement Learning and Data Informativity frameworks to estimate the optimal state feedback to accurately update online the Intermittent Controller. Even though RL is showing some limitations in the sampling of the data, Data Informativity benefits from the open-loop behavior in estimating accurately the underlying system.

1.1.3 Publications

Conference proceedings

J.A. Álvarez-Martín, T. Doublein, H. Gollee, J. Müller, R. Murray-Smith. ”Understanding the variability of pointing tasks with event-driven intermittent control”, *In proceedings of the 25th International Symposium on Mathematical Theory of Networks and Systems MTNS 2022. Bayreuth, Germany, 2022*

In preparation

- T. Doublein, J.A. Álvarez-Martín, H. Gollee. ”Stochastic Intermittent Control for engineering systems”. This paper will cover the integration of GP within the Intermittent Control framework. Experiments will be applied to the Single Inverted Pendulum and compared to the traditional SMH response under different conditions.
- T. Doublein, J.A. Álvarez-Martín, H. Gollee, R. Murray-Smith. ”Stochastic Data-driven Intermittent Control for adaptation”. This paper will extend the previous stated paper by addind additional algorithm with the IC framework to create a fully data-driven IC that can be used for adaptation.

1.2 Overview and structure of the thesis

This thesis is separated into eight chapters as follows:

In Chapter One, a short introduction presenting the high-level motivation for using the Intermittent Control framework is described as well as the motivation and aims of this project, followed by the overview of the thesis.

Chapter Two presents a literature review of biologically inspired control, explaining human control behavior and its strong links with intermittent control. Learning and adaptation are introduced and conventional adaptive control techniques are mentioned. Model-based control and data-driven control are compared, where Data Informativity is discussed. Then the use of Reinforcement Learning and Gaussian Processes for control are investigated, followed by other advanced techniques.

Chapter Three introduces the theoretical background of the Intermittent Control implementation. It explains the implementation similarity with continuous control with the addition of triggering and the presence of a generalized Hold. Reinforcement Learning theory is presented followed by the Data Informativity framework. Then, Gaussian Processes theory is explained, focusing on the data processing, optimization of the GP, and prediction using Single Task and Multi-Task GP. An overall discussion ends this chapter.

Chapter Four looks into system identification techniques such as Reinforcement Learning and Data Informativity within the Intermittent Control Framework. This chapter focuses first on the implementation and the assessment of the quality of the system identification. Then, experimental results are presented, starting with Data Informativity and followed by Reinforcement Learning, and a discussion.

Chapter Five focuses on the integration of Gaussian Processes into the Intermittent Control Framework. Implementation and assessment of GP in IC are described followed by experimental results, looking at single-task, multi-task, and online retraining of the GP.

In Chapter Six, system identification techniques from Chapter Four and Gaussian processes based Hold from Chapter Five are combined to implement a fully data-driven Intermittent Controller. This chapter looks at the impact of Intermittent control and its parameters in the context of adaptation. Experimental results are shown followed by a discussion.

Chapter seven is devoted to a general discussion, summarizing the findings of this work. It is split between the stochastic and the data-driven implementation of the intermittent control.

The final chapter presents the overall conclusion of this thesis, followed by the limitations and additional ideas as potential future work.

Chapter 2

Literature Review

2.1 Introduction

Standing, walking, or even running are part of our daily routine. However, despite the apparent simplicity of these tasks, human control mechanisms are still not fully understood by researchers. For many years, scientists and engineers have been aiming to understand the mechanism and develop physiological models of the human control process. In this chapter, biological-inspired control literature is reviewed, introducing sensory feedback as well as the Psychological Refractory period. Continuous control and intermittent control approaches are compared. In the second section, learning and adaptation are investigated, first by looking at conventional adaptive control before moving toward Gaussian Processes and Reinforcement Learning applied to control. Finally, other approaches not used for this work will be mentioned.

2.2 Biological inspired control

In this section, human quiet standing is described, followed by an overview of the two main approaches used to mimic human behavior: continuous and intermittent control.

2.2.1 Representation of human quiet standing

Composed of muscles, and nerves, as well as non-linear dynamics such as joint visco-elasticity, the human biomechanics can be considered as a very complex mechanical system. Despite the human upright body being described as an unstable system, neural-sensorial mechanisms present in the human body can compensate for this instability (Cotoros and Baritz 2010). In 1931, Telford 1931 discovered experimentally the presence of the Psychological Refractory period (PRP), where the human control action is not affected by any sensory feedback for a small amount of time following the previous action. This mechanism can be seen as a delay that can be correlated to a temporary bottleneck in processing, due to the cognitive system still processing the previous stimulus. Hence, the human control action is behaving on an open-loop regime for this short period. Moreover, stochastic fluctuations in neuronal activity, synaptic transmission, and sensory processing are also part of the Central Nervous System (CNS) which can be represented as noise (Faisal et al. 2008).

Variability is an important characteristic in the context of human control. For example, in the case of a repeated task, the control input generated by the human controller is not deterministic and each trajectory is slightly different from the previous one. As demonstrated by Jones et al. 2002; Faisal et al. 2008; Gollee et al. 2017; Martín et al. 2021, this variability is a combination of factors such as neural spike initiation-propagation, as well as a variation in the decision making processes happening in the Central Nervous System (CNS).

To start studying human control, it is necessary to introduce a simpler representation of the human body. The Single Inverted Pendulum (SIP) is often used as a model of the human body sway during quiet standing as it displays similar behavior when actuated at the ankle joint (Winter et al. 1998; Loram and Lakie 2002; Bottaro et al. 2005; Morasso et al. 2020) (see Figure 2.1). However, other models have also been introduced to represent human standing such as a model with an additional degree of freedom and controlled using reciprocal and synergistic muscle action at multiple joints (Jacobs 1997) as well as a linear pinned-polymer model of posture control including quadratic non-linearity (Alonso-Sanchez and Hochberg 2000). In this work, the model presented by Lakie et al. 2003; Loram et al. 2005; Loram et al. 2009; Gawthrop et al. 2011 is used and equations are presented in Appendix A. To increase the complexity of the SIP model, a cartpole model has been used as an analogy for balancing a stick (Yoshikawa et al. 2016). Model details are presented in Appendix B.

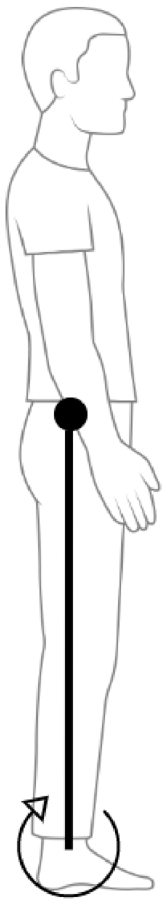

Figure 2.1: Pendulum as a representation of the human standing

2.2.2 Continuous Control based modelling

Wolpert et al. 1995 ran experiments to prove the existence of an internal model in the CNS. A set of three experiments has been conducted, using no applied force, followed by an assistive and then resistive force exerted on the arm of participants. The goal was to estimate with the free arm the position of the hidden hand. Bias and variance were evaluated and then modeled using a Kalman filter. The latter was able to reproduce the bias and variance of the estimated a forward model of the hand, under all three conditions.

Winter et al. 1998 has used stiffness control to model human control during quiet standing. Experiments were performed with eyes opened and eyes closed and have shown that vision does not appear to play a role during quiet standing. In addition, stiffness control presents potential advantages in the presence of disturbances in the standing. In another study, Peterka et al. 2000 used a PID continuous feedback, closed-loop control to generate a realistic Stabilogram Diffusion Function (SDF) that summarizes the displacement of the Center-of-Pressure (COP). This model uses variations in time delays and a neural controller to interpret change in SDFs as opposed to open-loop and closed-loop behavior.

Additionally, Todorov and Jordan 2002 used optimal feedback control to model human control behavior. The minimal intervention principle is used to explain the variability present in human control actions. They also demonstrated that mechanical redundancy plays a role in the solution to perform the motor system's task well. Motor variability is seen as an opportunity to perform system identification and is proposed as the strongest support for the optimal feedback control framework.

The models by Winter et al. 1998 and Peterka et al. 2000 represent stable posture control. However, some of the sway characteristics are impacted by the intrinsic muscle stiffness (Winter et al. 1998) and the background noise introduced (Peterka et al. 2000). Bottaro et al. 2005 used sliding mode control theory to match more consistently experimental postural data using Intermittent stabilization's models. Sway movement around the reference was assessed and shows that attention level and quality of sensory information can influence the amplitudes of those movements. The combination of variable structure systems and sliding motion control is proposed as an appropriate framework to model the intermittent stabilization process. In addition, Loram et al. 2014 proposed an intermittent controller that helps explain the potential neuro-physiological basis in human movement motion, such as the refractory period.

2.2.3 Intermittent Control based modelling

Even though the Continuous Control framework presented above has helped in understanding human control (Wolpert et al. 1995; Peterka et al. 2000; Todorov and Jordan 2002; Bottaro et al. 2005), fundamental aspects of human control such as intermittent feedback, temporal refractory periods or triggered responses are not covered. Craik 1947 introduced that human control systems operate using intermittent feedback loops as opposed to continuous control. He also mentions key concepts such as resilience, efficiency, and optimization. The presence of intermittency enhances the resilience of the control input by acting as a sort of filtering and smoothing. Intermittent feedback also helps in reducing cognitive load, hence improving efficiency by only providing feedback sensory information when necessary. ibid. is also proposing that intermittent control helps in minimizing the use of resources, and then optimizing control processes on the relevant feedback information.

Following the work by Telford 1931 and Craik 1947, Vince 1948 and Navas and Stark 1968 investigated the relationship between PRP and intermittency in human control. Craik 1947 and Vince 1948 showed that the PRP can be explained as a delay in the feedback loop. Based on experiment results as well as physiological considerations, Navas and Stark 1968 suggests that sampling of sensory information is irregular and asynchronous, hence indicating input-synchronized feedback rather than clock-driven intermittency.

The first algorithmic intermittent control model was introduced by Neilson et al. 1988 and applied to visual pursuit tracking tasks. Experimental results were modeled using intermittent control and incorporating internal model inaccuracy and some speed-accuracy trade-offs. By doing so, multiple tracking behaviors were accounted for such as inaccuracies in the stimulus responses.

Based on Neilson et al. 1988 and adapted for control-engineering purposes, Ronco et al. 1999 introduced an intermittently moving-horizon approach during which the control signal evolves in an open-loop fashion. Similarly explained in Craik 1947, intermittency in the control loop is helping reduce the computational burden. In addition, Intermittent Continuous-time Generalised Predictive Control (ICGPC) shows robustness to measurement noise. Also, OLIFO (Open-Loop Intermittent Feedback Optimal) control behavior is compared to biological motor control and shows consistent similarities. This work was expanded by Gawthrop and Wang 2007, where clock-driven intermittent control is presented for its flexibility in the control. If the intermittent interval Δ is very small ($\Delta = 1ms$), the intermittent controller is behaving closely to a continuous controller. However, by increasing this intermittent interval, the control input coming from the intermittent controller changes but also deteriorates if increased too much. Modelling of human stick balancing has also been investigated by Gawthrop et al. 2013.

Another implementation of this model is to use event-driven intermittent control as presented by Gawthrop and Wang 2009. In this implementation, triggering is now based on the difference between the states of the plant and the states estimated by the model in the Intermittent Controller Hold. With this implementation, the notion of threshold is introduced. The triggering occurs if the difference explained above is above this threshold. However, this threshold can be set to 0, hence falling back to a clock-driven intermittent control if needed.

Also mentioned in ibid. is the capability of the intermittent control to have different Hold dynamics during the open-loop intervals. Two approaches are presented there: Zero Order Hold (ZOH) and System Matched Hold (SMH). While ZOH simply holds the control input until the next event, the SMH approach uses a closed-loop representation of the plant, hence evolving during the open-loop interval. Results explained that ZOH is inappropriate for event-driven intermittent control due to the time-varying open-loop interval and SMH is showing useful properties. As SMH is showing similar behavior as continuous control during the open-loop regime, differentiating between the two can be difficult; this has been named masquerading properties (Gawthrop and Wang 2009). As previously mentioned, delays due to PRP need to be compensated similarly to the continuous controller case by using a predictor as they might introduce instability and loss of performance as shown by Hongxia Wu et al. 2002.

Later, tapping has also been introduced as an alternative to ZOH and SMH implementations by Gawthrop et al. 2011. Contrary to the continuous-like behavior from SMH, tapping is considered a discontinuous approach by using the Laguerre function and it shows good stability and precision (Huang and Mason 2000). The main advantage of tapping control is its capability of overcoming the effect of nonlinear friction in systems as presented by Gawthrop and Gollee 2012 where continuous contact of the hand of participants with a joystick is compared to a tapping strategy. Intermittent Tapping Control (ITC) uses fixed-interval pulses of variable amplitude during an open loop. This type of fixed-interval pulse can be compared to pulses used in functional electrical stimulation of muscle. In addition, the notion of 'taps' has been observed in human motion control (Loram et al. 2011).

Even though this thesis relies on the Intermittent Predictive Control implementation by Gawthrop et al. 2011; Gawthrop and Wang 2009; Gawthrop et al. 2011; Gawthrop and Gollee 2012; Gawthrop et al. 2013; Gollee et al. 2017, it is worth mentioning additional frameworks where intermittent controller is used differently. While this implementation is always outputting a control input, using sensory feedback at triggering or using generated states from the hold during the open-loop interval, the intermittent controller called Act-and-Wait (AAW) by Stépán and Insperger 2006 is switching periodically the control input on and off. One key feature of the AAW controller is being a suitable alternative for control systems with feedback delays. Commonly called the state-space intermittent feedback strategy by Morasso et al. 2020, this intermittent controller lets the system evolve itself and then applies a new control action if the system reaches an unstable area of the state-space (Gawthrop et al. 2014b) for an analysis of the different IC approaches.

2.3 Learning and Adaptation

In complex systems, it is important to understand the system dynamics to yield a robust controller while maintaining performance. Two common approaches are *learning* and *adaptation*. In this section, the concept of learning and adaptation is described as well as the role of machine learning, more specifically RL, to overcome these two problems (Tsypkin 1971).

Learning Control, is dealing with an unknown system in addition to these unknown parameters. Learning control is based on the ability of the control system to develop a mathematical representation of the system's behavior. It implies that the control system contains sufficient computational ability to achieve it. This type of control can be seen as an extend of adaptive control.

Adaptive Control, is the capability for the controller to update its mode of operation to achieve the best control law for a particular situation. There are three main inherent functions for any adaptive system to be able to evolve in the best manner: (i) getting continuous access to the current states of the systems or being able to identify the underlying process, (ii) being able to compare the current performance of the control to know if adaptation is needed to achieve a defined optimal performance, and (iii) being able to properly modify its characteristics to drive the control system to the target. In the case of adaptive control, the system's dynamics are known but the dynamic's coefficients are unknown. These coefficients also called parameters, are estimated over time and used to update the control law accordingly (Åström and Wittenmark 2013; Hardy et al. 2019).

To know enough information about a system, it is necessary for the controller to *explore* the state-action space. However, in some cases, the exploration can be dangerous as the system can have some restricted working area on the state-action space; it is necessary to *exploit* the already known information to improve the present decisions. There is a trade-off between exploration and exploitation and this is called the exploitation and exploration paradox and it is summarized in Figure 2.2. RL by being focused on exploitation and exploration problems have a good potential to be integrated within the context of IC.

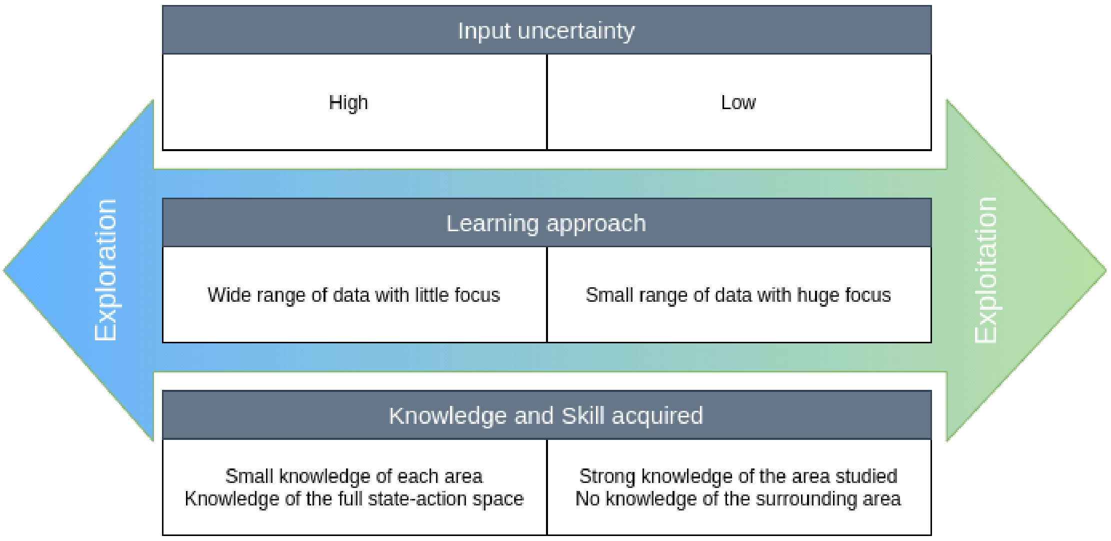
Input uncertainty
High
Low
Learning approach
Wide range of data with little focus
Small range of data with huge focus
Knowledge and Skill acquired
Small knowledge of each area
Knowledge of the full state-action space
Strong knowledge of the area studied
No knowledge of the surrounding area
Exploration
Exploitation

Figure 2.2: Summary of Exploration-Exploitation paradox (modified from Fig.1 in Hardy et al. 2019)

2.3.1 Conventional Adaptive Control

As presented by Åström and Wittenmark 2013, adaptive control can be split into three different schemes: Gain Scheduling, Model-Reference Adaptive System (MRAS), and Self-Tuning Regulators (STR).

Gain Scheduling

Gain Scheduling involves measuring changes in the system's operating conditions and to update the controller gain accordingly. This results in being able to overcome changes in the process. This was originally used as a flight control system, when the gain of the controller depends on some variable, for example, the current speed of the aircraft or the altitude. Based on some operating conditions, the controller gain is updated (Nichols et al. 1993; Leith and Leithead 2000). This scheme is presented in Figure 2.3 and is composed of two loops: the inner loop is composed of the controller and the system, while the outer loop is composed of the *Gain Schedule* block which updates the gain of the controller based on operating conditions.

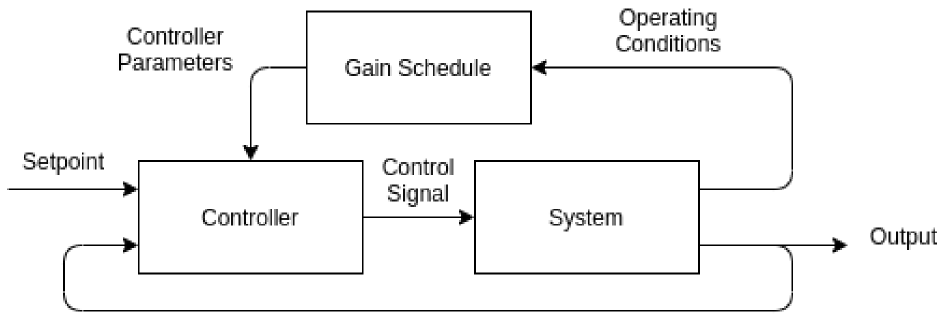

Figure 2.3: Block diagram of the Gain Scheduling Scheme (modified from Fig. 1.17 in Åström and Wittenmark 2013)

Model-Reference Adaptive System (MRAS)

MRAS has been implemented to solve problems where performance specifications are given in terms of a reference model (Kojabadi 2005). It is based on a reference model that represents how the process output should respond to a specific input (see Fig. 2.4). The adjustment mechanism is based on the error between the reference model output and the plant output. The objective is to bring the error to zero by adjusting the parameters of the controller (Åström and Wittenmark 2013).

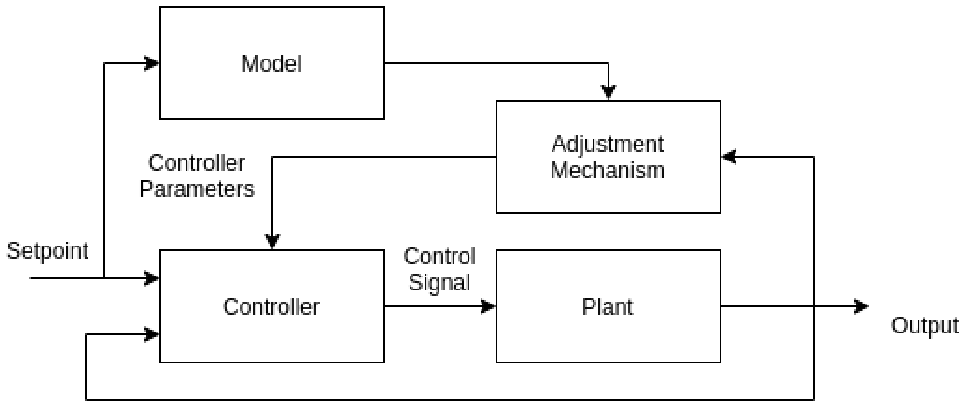

Figure 2.4: Block diagram of the Model-Reference Adaptive System (MRAS) Scheme (modified from Fig. 1.18 in Åström and Wittenmark 2013)

Self-Tuning Regulators (STR)

STR is based on a system identification approach. Contrary to the two previous schemes, the STR scheme estimates the process parameters for the first time. Then, the controller is designed based on the new parameters estimation (P.E. Wellstead 1979). It is composed of two loops (see Fig. 2.5): the inner one is used as a feedback loop, while the outer one is composed of an *estimation* block, as well as a *controller design* block. This loop is used to update the controller parameters.

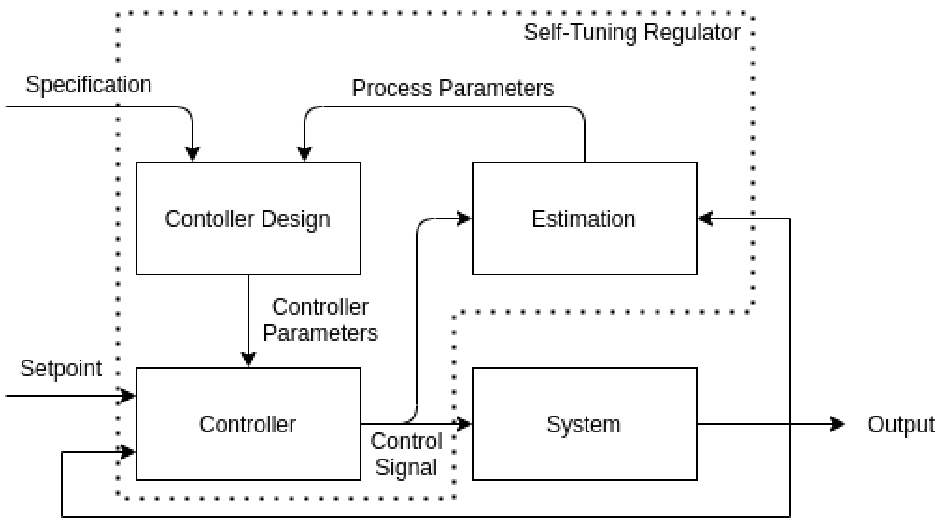

Figure 2.5: Block diagram of the Self-Tuning Regulators (STR) Scheme (modified from Fig. 1.19 in Åström and Wittenmark 2013)

These three adaptive schemes are split into two different methods: *direct* or *indirect*. As the indirect method is estimating the plant dynamics to design the controller, the direct approach is directly updating controller parameters. The indirect approach is split into two parts: system identification and controller design.

As presented in section 2.2.3, Intermittent Control requires a Hold system, that represents the system close loop dynamics in the case of the SMH. System identification techniques have been used in the past as presented by Martín 2018, where Kalman filters have been integrated with the IC framework proposed by Gawthrop et al. 2014a.

Gaussian Processes can be seen as a sort of indirect method (system identification) whereas Reinforcement Learning, on the other hand, can be visualized as the direct method.

2.3.2 From model based control to data-driven control

Before 1960, most control designs relied on well-established methods such as the Bode and Nyquist plot as well as Ziegler-Nichols charts. Most of these implementations rely on graphical design methods (Gevers 2002). Since 1960, modern control theory has been developed in the following areas: system identification, optimal control, and adaptive control. This increase in research papers on these topics is related to the newly introduced parametric state space representation created by Kalman 1960. Whilst system identification allows to create a mathematical representation of the actual dynamical system (Ljung 1999), optimal control and adaptive rely on the model produced by the previously stated method to design a correctly designed controller.

2.3.2.1 Model Based control

The introduction of parametric state space representations by Kalman 1960 in combination with optimal control techniques led to a specific type of control named Model-Based Control (MBC). Controller design usually relies on well-trusted linear control algorithms such as LQR design, pole placement, or robust control. As presented in Brosilow and Joseph 2002, a multitude of model-based controllers have been implemented over the years such as feed-forward, cascade, output constraint as well and model predictive control. Hjalmarsson et al. 1996 mentioned that using closed-loop data for system identification can obtain a more accurate controller if the identification time is long enough. This approach links to iterative identification.

As explained above, all previously stated control-designed techniques rely on having a mathematical representation of the plant. Most identification theories have been focused on getting the perfect representation of the ’true system’ by implementing complicated modeling techniques (Gevers 2002). However, moving from an ideal representation to an approximation of the true model started appearing in the literature in the 1980s. Ljung 1999 shifted the attention from an ideal system identification towards a use-case identification dependency. He also mentions that identifying a multi-output system can be difficult, and it is possible to replace it with multiple single-output models when used for simulation purposes.

Following this idea of only getting an approximation of the model, Gevers 2002 mentioned that low order model-based controller can lead to higher performance. This can be explained by the reduced order of the controller, hence reducing computation load. Ljung 1999 has mentioned linear and ready-made models that can be used when physical insight is not known enough to model dynamical systems. The idea is to only specify the order of the model for the ready-made model. The four most common ones are based on transfer function models: Output Error (OE) (used by Carrillo et al. 2009 to identify system operating in closed-loop), ARX, Box-Jenkins (BO), and ARMAX model. Subspace Estimation Techniques for State Space Models are also mentioned, as this is helping identifying the state space matrices, for a localized set of sample data, using the subspace projection technique. Stenman 1999 demonstrated that the ’model-on-demand’ approach can generate prediction errors with similar more advanced techniques such as as fuzzy identification and artificial neural networks. Others techniques, such as frequency-domain identification are also mentioned.

2.3.2.2 Data-driven control

In contrast to model-based control, data-driven control is using online or offline data to optimize a policy (Hou and Jin 2013). This means that the controller does not need an explicit representation of the dynamical system to be designed. As presented by Hjalmarsson 2002, Iterative feedback tuning can be used on linear and non-linear systems. This technique approximates the correct gradient of the underlying system properties which leads to a robust controller as it covers a large class of systems.

More recently, Guo et al. 2019 have extended data-driven control to be applied to Multi-Input/Multi-Output (MIMO) non-linear systems. This model-free adaptive predictive control (MFAPC) method is based on a time-varying pseudo-jacobian matrix as well as dynamic linearization techniques. It is stated that due to the addition of predictive control, the algorithm has strong robustness as well as excellent tracking performances. Luppi et al. 2022 have used data-driven control to stabilize non-linear systems. Whilst conditions are met to ensure that the system is stabilizable, the algorithm is only using input and output data generated during the experiment to directly design a state feedback controller.

Data-driven control has the advantage of being able to solve complex stabilization problems without the need for a model by simply using data recorded from an experiment. However, this assumes that data is carrying enough information for the algorithm. Even without explicitly mentioning it, algorithms rely on data sufficiently rich to converge towards accurate results (Mareels and Gevers 1988). The concept of persistence of excitation was introduced in 1966 by Åström and Torsten 1965 and has been used thoroughly since by Ljung 1971; Moore 1983; Bai and Sastry 1985; Goodwin and Eam Khwang Teoh 1985; Green and Moore 1986; Mareels et al. 1987; Mareels and Gevers 1988.

Willems et al. 2005 uses the persistence of excitation as well as the controllability condition to assert that using a single trajectory from a linear system, it is possible to generate all trajectories. These results introduced important implications for system identification methods as well as data-driven control. Van Waarde et al. 2020 has extended Willems et al. 2005 fundamental lemma by implementing an algorithm that allows using multiple trajectories. This improvement has shown its capability of identifying linear systems where data samples are missing. It is also possible to use this implementation to model unstable systems by using multiple small trajectories. Then, Yu et al. 2021 extended this algorithm by replacing the controllability condition with controllable subspace, unobservable subspace, as well as certain subspace associated with the measured trajectories. This paper also proves that the excitation signal can be reduced to a certain degree of minimal polynomial. Additional results show the equivalent between model predictive control and data-driven predictive control, even applied to uncontrollable systems, with the advantage of reducing drastically the amount of data to design such a controller.

van Waarde et al. 2020 introduced the notion of Data Informativity. It has been introduced to assess whether the data is carrying enough information for system analysis and control design. This algorithm is also able to reduce the rank condition on the persistence of excitation signal in some cases, such as stabilization using state feedback control. As stated in ibid., the proposed data-driven approach can solve controllability analysis and stabilization problems where traditional system identification could not, due to the lack of information in the data. Data Informativity is a important concept that allows the data-driven technique to outperform traditional system identification techniques in some cases. Whilst ibid. has been focused on noise-free data, this work was extended to noisy input-state data in Van Waarde et al. 2023.

2.3.2.3 Limitations and challenges

As presented above, model-based control and data-driven control are two available options, whether system dynamics are known or not. The main difference is the need to know or not the system dynamics to design the controller.

As the order of the system increases, model-based control becomes more and more unfeasible by the increase in the order of the controller and becomes not suitable for real-life applications. It is necessary to use model reduction techniques first. However, as the complexity of the approximation of the system decreases, the risk of instability arises due to un-modeled dynamics. The trade-off is between the complexity and accuracy of the model versus the effort and cost attributed to the design of the controller.

As presented in Hou 2013, the Data-Driven control theory presents some robustness issues due to the un-modeled dynamics. Moreover, data gathered for the estimation might be contaminated by noise or missing data due to sensor issues. Another issue with data-driven control is the data processing method available for online processing. Most algorithms, such as machine learning ones are used offline due to the intensive computation.

Model-based control and data-driven can also be updated online to answer the problem of adaptation. However, iterative identification and control can lead to unplanned instability. Moreover, in some adaptive control algorithm implementations, controller parameters are updated at each sample step. This constant change in the control's parameters can be seen as non-linearity in the closed-loop dynamics, hence making the stability analysis very complex (Gevers 2002). Other generic problems are also affecting the implementation of model-based and data-driven adaptive controllers: problems of transient instability, suddenly unstable close loops, or even impractical control objectives. As presented in Anderson and Dehghani 2008, data-driven adaptive control is limited by its application beyond experimental examples.

2.3.3 Reinforcement Learning

Over the past few decades, interest in Artificial Intelligence and Machine Learning has risen considerably. Machine Learning can be split into three main categories: Supervised, Unsupervised, and Reinforcement Learning. As Supervised Learning is oriented to solve classification and regression problems, and Unsupervised Learning focuses on association and clustering problems, RL is focused on exploitation and exploration problems. The main objective is to teach an agent (the controller) to interact with its environment (the system). This chapter describes the origin and background of RL and explains the difference between model-free and model-based RL algorithms.

2.3.3.1 Background of Reinforcement Learning

Introduced in the 1950s by Richard Bellman, dynamic programming is a mathematical optimization that consists split into recursive sub-problems a complicated one to simplify it. It is now widely considered the only feasible way of solving general stochastic optimal control problems (Sutton and Barto 2018). As described by Bellman himself as ”the curse of dimensionality”, this approach is limited by the number of state variables as it grows exponentially with it. Even with this computational limitation, it is still more efficient and widely applicable than other general methods.

Dynamic programming has been improved by integrating extensions to partially observable Markov Decision Processes (MDPs). This mathematical framework is used for modeling decision-making processes where the outcomes it are split between randomness and under the influence of a control law. Machine Learning algorithms rely on these Processes. MDPs can only be applied if the Markov property is respected for a stochastic process: the distribution probability of the future state is only dependent on the current state without involving a prior event.

The MDPs are trying to create a mapping between states, actions, and rewards. This mapping is commonly called a policy and can be seen as a control law. RL is the formalization of dynamic programming applied as an optimal control of incompletely-known MDPs. The objective of RL is to learn a task by trying to maximize a numerical reward signal. Same as the MDPs, the RL algorithm tries to map the best action to take according to the current states based on the best reward.

RL is based on a *Environment-Agent* framework. The *Environment* is a representation of the system we are trying to control. On the other hand, the *Agent* is the active decision-maker on the environment to control it. The basic architecture is presented in Figure 3.4 on page 54.

The most common way to understand RL is as a *Trial-Error* scheme. The Agent is improving its knowledge of the Environment by exploring the state-space area by trying an action and receiving a reward related to it. It is important to notice that in some cases, actions may affect the next state of the system, which subsequently affects the following rewards. RL is simply trying to capture the most important aspect of a dynamical system, to be able to control it to achieve a specific goal (Sutton and Barto 2018).

RL is moving towards an important integration with mathematical, statistical, or even optimization problematic. The mathematical framework of RL can be split into two different categories: Model-Free or Model-Based Algorithm. The following section goes into more detail about each scheme.

2.3.3.2 Model-Free and Model-Based approach

From learning how to play tic-tac-toe to balancing a cart-pole pendulum, RL can be used to solve very different problems. One common distinction is the difference of *action-state* space between them. While a system such as a tic-tac-toe game has a finite number of actions and states, the action-state space of other systems is continuous.

In the case of discrete *action-state* space, it is theoretically possible for an agent to explore every state-action and learn about the reward associated with it. On the continuous action-state space, it is more generalized to find a value function which is a function of the current state s, current action $\mathbf{A}$, and an end-state s'. Some studies have shown the possibility of discretizing the state-space to estimate the value function using Q-learning discrete algorithm (Gaskett et al. 1999; Smart and Kaelbling 2000).

In RL, the policy π or $\pi(s|a)$ corresponds to the mapping of some state s to the probabilities of selecting each possible action given that state. There are two types of policies: the *On-Policy* and the *Off-Policy.* While the *Off-Policy* is independent of the agent's actions, the *On-Policy* is basing decisions on the previous state-action pair.

RL is based on two different methods: *Model-Free* and *Model-Based* model. The difference between them is the prior knowledge or not of the environment, i.e the system to be controlled. Each of these two models has some pros and cons which are summarized in Table 2.1.

Model	Pros	Cons
Model-Free	- Computationally less complex - Accurate representation of the environment not needed	- Need experiences for gathering training data - Exploration more dangerous - No explicit idea of how environmental dynamics affects the system
Model-Based	- Safe to plan exploration and can train from simulated experiences	- Agent Quality based on the model implementation - The model can become difficult to learn - Computationally more complex

Table 2.1: Model-Free and Model-Based: Pros and Cons.

The model-free method is used when there is no prior knowledge of the Environment, and the gain of information is based on the *trial-error* scheme. The model-based method, on the other hand, is focused on a model. Using a baseline policy, such as random actions or any educated policy, the output trajectory data is used to fit a model. Iteratively, the model is improving based on its new knowledge of the Environment.

As presented in Table 2.1, Model-free and Model-Based approaches have both advantages and inconveniences. The exploration aspect of the model is an important factor. It is important to introduce the terminology exploitation and exploration. While exploitation is trying to get as good as possible based on what is already known, exploration is trying to reach an unknown part of the state-action space to improve its learning. *unsafe* areas are highly recommended to be avoided. A model-free algorithm can be dangerous to use in that case where not knowing the dynamics of the systems can lead to an exploration of these *unsafe* regions. Studies on robotic tasks show that in some cases model-free systems reach similar or better performance than Model-based ones (Renaudo et al. 2015). This is partially due to the high number of states and the difficulty in modeling the environment.

2.3.3.3 Limitations

While RL shows promising results, some limitations have to be taken into consideration. As presented by Lin 1991, RL is dependent on the representation of the system. One possible approach, for example, is to reduce the number of inputs. ibid. is also describing the slow speed of convergence.

The *unwise use of sensing* is another limitation. To comprehend the environment dynamics, the agent needs to sample the state-action domain. The presence of *unsafe* area can complicate this exploration phase. In real-world systems, an agent cannot take into consideration all world states as they may be irrelevant to the system. It is necessary to decide how to use sensors efficiently while still being able to control the plant (Whitehead and Ballard 1991; Tan 1991). As stated by Kober et al. 2013, RL is difficult to apply in the robotics fields due to the high-dimensional continuous state spaces. One possible approach to solve these problems is to split the model into a multitude of simpler models as described by Narendra et al. 2016. This approach is stated as an order of magnitude faster than the use of a single model.

Cutler and How 2015 is the training of a real-world system based on an already pre-trained policy via simulation using *Probabilistic Inference for Learning Control* (PILCO). To reduce the number of samples needed to train a model, the merge of RL and Gaussian processes has increased over the last years (Engel et al. 2003, 2005; Kuss and Rasmussen

2003). Furthermore, Osband et al. 2018 highlighted the importance of deep Reinforcement Learning uncertainties in its benefits towards efficient exploration. This approach is also able to scale better than other approaches, especially focusing on linear representations. More recently, Miller et al. 2024 used nearest neighbors look-up to implement a direct sparsification method of the dataset. This can be used to improve computation time.

2.3.4 Gaussian Processes for Control

The GP learning method, contrary to Reinforcement Learning is closer to system identification as the main two goals are to identify an interpretation of the data (system modeling) or be able to predict unseen data (predictive aspect) (Rasmussen 1997). One of the main aims of machine learning is to be able to generalize and identify the relationship between input and output observed data. One common approach is to use parametric modeling when the underlying order of the function can be approximated. However, selecting a form of the function for the algorithm to fit some coefficients based on training data can be quite difficult and needs a lot of trial and error. As presented by Seeger 2004; Lawrence et al. 2002, when the underlying function expresses high non-linearity, hence difficult to describe with a predesigned function, a non-parametric algorithm can be quite powerful in the identification.

Due to their simplicity, flexibility, and non-parametric design, Gaussian Processes (GP) are commonly used to solve complex machine learning problems (Seeger 2004). GP models are considered black-box modeling, ideal for non-linear functions. Furthermore, contrary to other commonly used modeling techniques for non-linear architectures, GP can be fitted to the data without the need to use non-convex optimization routines, which simplifies the optimization (Lawrence et al. 2002). In addition, GP do not use a set of basis functions but relies on finding the relationship between input-output data (Kocijan 2016; Petelin and Kocijan 2011), which can be easily adapted to any kind of function. Also, GP rely on Bayesian theory, hence adding a probabilistic element that is suitable not only for classification problems but also regression (Quiñonero-Candela et al. 2007).

Gaussian processes are composed of two main elements: one or multiple covariance functions and some associated hyperparameters. Covariance functions, also called kernel functions, help to determine the shape of the prior and posterior distribution of the GP and are parameterized by a vector of hyperparameters. As presented by Rasmussen and Williams 2006, kernels can take multiple forms such as Squared Exponential, Matern, or Rational Quadratic. In addition, covariance functions can be combined, with addition, subtraction, or dot product. However, each of these kernels has some hyperparameters that need to be optimized for the GP model to represent accurately the input-output data. These hyperparameters are optimized to fit the input-output data relationship. In the case of the Squared Exponential (SE) Kernel, also called Radial Basis Function (RBF) Kernel, the unique associated hyperparameter is a *length-scale*. The learning of these hyperparameters is sometimes ignored by the literature as mentioned by Quiñonero-Candela et al. 2007. It is important to note that accurate hyperparameter values are crucial and the optimization routine used for finding them can be quite computationally expensive. For example, the algorithm described by Storn and Price 1997 can be used to optimize GP hyperparameters, but the good convergence property is getting in the way of optimization speed. This can become difficult when applied to real-time applications. Another approach by Durichen et al. 2015 proposed using repeated training phases with different initial hyperparameters and keeping the best optimization based on the Negative Log Marginal Likelihood (NLML) (Rasmussen and Williams 2006). The one approach that has been trusted is the deterministic conjugate gradients minimizer as presented by Rasmussen 1997, compared to alternative optimization techniques in Rasmussen and Williams 2006 and used in Deisenroth and Rasmussen 2011; Deisenroth 2010; Bonilla et al. 2007.

In the engineering field, GP can be quite an attractive way for modeling unknown dynamical systems. As presented by Petelin and Kocijan 2011; Kocijan 2016, due to its design, the number of training data for modeling can be quite small, especially compared to alternative self-learning approaches such as neural networks (Nguyen and Widrow 1990) or fuzzy models (Jang et al. 1992). From a practical point of view, GPs provide simplicity through their non-parametric approach and low amount of points for training compared to alternative approaches. But in addition to this, the Automatic Relevance Detection (ARD) naturally present in GPs (Kocijan et al. 2005), allows understanding the importance of each input using the values of the hyperparameters. In addition, prior knowledge can also be included in the design and GP is also able to handle noisy data, often present in sensor reading, due to its probabilistic underlying approach (Kocijan 2016). Once the GP model is created, predictions can be generated for unseen data points. Each prediction comes from a normal distribution, with a mean value, prediction output, as well as a variance, representing uncertainties, that can be associated with the confidence of the predicted output (Petelin and Kocijan 2011; Kocijan 2016).

Looking at dynamical systems, most applications have been using a linear state space representation of the systems for state estimation as well as system identification routine. As presented in Eleftheriadis et al. 2017, Gaussian Processes have been used to model non-linear state space model and is named Gaussian Process State Space Model (GPSSM). By using GP, this representation of the plant can be based on a low number of sampling points as well as incorporating probabilistic properties. As mentioned by Kocijan 2016, even though the output of the model is continuous, it is possible to use discretized data as training data.

2.3.4.1 GP for Regression and control

In the engineering field, it is quite common to consider a model structure to be linear due to the well-established identification algorithm available. Although this approximation can be enough for control using techniques such as feedback control design or models predictive control, the underlying non-linearity can introduce non-desirable behavior in the control, especially when the complexity of the model is increasing. This is where non-linear modeling techniques are required.

As presented by Kocijan et al. 2005, GP models can model from identification data non-linear dynamical model. Rasmussen 1997 has bench-marked the use of GP for regression purposes against several methods, using both generated as well as simulation data: GP models can perform similarly to neural networks both in modeling and prediction. Whilst this approach is an alternative to other identification methods, it has been used over the years within model-based control (Girard 2004; Kocijan et al. 2003; Rasmussen and Williams 2006; Williams et al. 2008; Petelin and Kocijan 2011; Kocijan 2016) due to its probabilistic capability.

Girard 2004 has explored the benefits of using Gaussian Processes over the previously stated method. The focus has been to use the model's uncertainties to improve the modeling as well as the forecasting and control, especially for non-linear dynamical systems. This work has introduced the propagation of the uncertainty method, to help unrealistic growing uncertainties coming from a single step prediction. This method has been used to implement a cautious controller. This controller uses an automated regularising behavior to adapt to local uncertainties. Results show that including the variance improves robustness and tracking performances (Murray-Smith et al. 2003).

Kocijan et al. 2003 has been a main actor in the use of model-based predictive control based on Gaussian processes. This implementation relies on a fixed model that has been identified offline and, hence is not adaptive. Even if some issues are presented such as some concern about the efficiency of the non-linear optimization approach or the lack of stability verification, it has been shown that using non-linear model predictive control based on Gaussian Processes has a higher level of robustness due to the information contained in the model.

Rasmussen and Williams 2006 has presented an example of how Gaussian Processes Regression (GPR) can outperform linear regression methods, but also non-linear methods such as Local Polynomial Wavelet Regression (LPWR) in the case of high dimensional system (seven degrees-of-freedom SARCOS anthropomorphic robot arm). Williams et al. 2008 presents similar results, comparing Linear Regression to single-task GP applied to a six-jointed manipulator arm (PUMA 560). The amount of samples used for training is also investigated. Linear Regression (LR) errors are stabilized around 200 samples while GP predictions keep improving as the amount of sample increases.

In Kocijan 2016 summarize adaptive control using GP and focus on gain scheduling as well as iterative learning. The concept of evolving GP is also mentioned. Petelin and Kocijan 2011 introduced the concept of evolving GP (eGP) models. This concept has been introduced for scenarios where systems might be changing over time, or the operating region keeps changing. Whilst this type of control problem can be solved using iterative learning control (Hjalmarsson 2002), it is not currently implemented for GPs. Contrary to previously stated GP-based model predictive control, where adaptation is not covered, this concept of eGP is recursively adapting online the overall GP structure (regressors, basis vectors, type of covariance function, and hyperparameters) using incoming data. The author mentions that eGP can be considered as a higher level of adaptation of non-parametric and probabilistic models. Results are presented against a stable non-linear system and show that it can be successfully controlled.

As previously mentioned, multiple independent single-output GP can be used to represent each output of a single system. Even though this method is widely used due to its simplicity, it can be considered as sub-optimal (Rasmussen and Williams 2006) or detrimental (Caruana 1997), hence multi-output learning can be beneficial in the learning. Bonilla et al. 2007 has implemented a multi-output GP also called multi-task GP. This method relies on the modeling of the dependencies between each output/task of the model. This implementation uses a shared covariance function across each output to learn the inter-dependencies between each output. Additionally, a “free-form” task-similarity matrix is used based on the assumption that tasks are correlated inside the same cluster. Using this multi-task GP approach, it is necessary to update the likelihood function to optimize to take in consideration the additional hyperparameter in the task-similarity matrix. Experiments have been based on two commonly used datasets and results show that using Multi-Task GP can provide a reduction of the mean absolute error by 6 times. The assumption of an inter-task similarity matrix presented here has been proven inherent in the work presented by Williams et al. 2008. Experiments have demonstrated the improvement of the learning when sample data is shared between tasks. It has been demonstrated that multi-output GP performs well for extrapolation tasks in the context of learning inverse dynamics models.

Following Bonilla et al. 2007 and Williams et al. 2008, Leen et al. 2012 has extended the multi-task GP framework by looking at asymmetric multi-task learning. Even though Chai 2009 applied a symmetrical multi-task model to an asymmetrical structure, Leen et al. 2012 has implemented an asymmetric multi-task GP model, also called ”focus multi-task GP”. The main takeaway from this approach is to simplify the independence assumption related to the secondary tasks. As presented by Durichen et al. 2015, GP modeling is healthcare is often focused on a single univariate output time series, in other words, single-task GP. To improve the robustness of the tools as well as using multiple sensors, multi-task GP can be used to model multiple correlated multivariate physiological time series simultaneously. As explained in the research paper, the most important advantage is to be able to incorporate uneven sample time series into a single model, without the need to use techniques such as down-sampling or interpolation. Experiments on real-world data sets are presented such as vital-sign signals as well as motion compensation in radiotherapy. It has been shown that multi-task GP (MTGP) can be used as a flexible approach in various biomedical applications.

2.3.4.2 GP Challenges

Although Gaussian Processes have been used as modeling techniques, there are some challenges present in using them in a wide range of applications. As mentioned by Kocijan 2016, due to the probabilistic aspect present in this framework, GP modeling can provide estimated prediction over the entire space. However, one common problem present in the control system is to obtain meaningful samples as the data lies around the equilibrium region (Girard 2004). There are four main GP challenges: assessing model accuracy, handling noisy data, correct selection of the kernel, and optimization of the modeling based on the number of samples.

Due to its non-parametric nature, assessing the GP model accuracy with the system can be quite complicated. Instead, assessing performance might be an option to consider, such as mean squared error or negative log predictive probability. Another possible approach is to use a measure related to the speed of the approximation. This could be looking at the learning time of the hyperparameter or the number of iterations for the optimization routine. As proposed by Quiñonero-Candela et al. 2007, one possible approach is to fix a “computational budget” of available time, and can be considered as a computationally efficient approximation measure.

Whilst Gaussian Processes can handle noisy data, it is worth mentioning that the noise distribution is considered to be Gaussian distributed. In the case of non-Gaussian noise, the assumption used for learning the hyper-parameter using the marginal likelihood is not correct anymore (Kocijan 2016). However, this issue can also be present in other parametric and non-parametric approaches used for system identification.

One of the main design parameters present in the GP framework is the use of kernels functions, also commonly called covariance functions. A lot of kernels are available to pick from, however, this can be quite difficult to select the correct one. As mentioned by Quiñonero-Candela et al. 2007, the main driver for this selection is based on the complexity of the underlying system, as well as the number of inputs present and the presence of noise. In other words, kernels are fully dependent on the system (Kocijan 2016). Some research investigated the potential of an automated kernel selection framework (Duvenaud et al. 2013). This approach uses a combination of base kernels and has been shown to outperform traditional kernel selection in multiple tasks.

Finally, one of the main drawbacks of the Gaussian Processes modeling is the computational burden related to the amount of training data (Kocijan et al. 2005; Quiñonero-Candela et al. 2007; Lawrence et al. 2002; Durichen et al. 2015; Seeger 2004). On top of the specific issues with the dynamical system, such as data outside the equilibrium space, the optimization routine relies on the inverse of a high dimensional covariance matrix. This can be considered as a concern, this can be seen as a serious problem in the case of real-time applications, with a computation increasing by $\mathscr{O}(n^3)$, where n represents the amount of sample used for training. However, as mentioned by Seeger 2004 and presented by Lawrence et al. 2002; Quiñonero-Candela et al. 2007; Snelson and Ghahramani 2005, some work has been focused on introducing sparse approximation to reduce the complexity of the optimization for modeling.

Sparse Gaussian processes have been introduced to help the computational effort required by the optimization algorithm. Quiñonero-Candela et al. 2007 has analyzed the performance of different sparse algorithms using a uniformed interpretation presented as ”exact inference with an approximate prior”, to enable a direct comparison. Presented as Fully Independent Training Conditional (FITC) in ibid., Snelson and Ghahramani 2005 introduced the Sparse Pseudo-input Gaussian Processes (SPGP) likelihood approximation. This approach shows significant improvement especially when looking for an extremely sparse solution. The method relies on moving the data point, called the inducing point, to an optimal location without constraining it. This leads to a better solution and it can model some non-stationary effects.

In more recent research by Grancharova et al. 2023 and Krivec et al. 2021, new algorithms are proposed to improve GP optimization such as using sub-optimal distributed predictive control or variational GP-NARX models with the use of Graphical Processing Unit (GPU). Alternatively, other approaches are also available, such as conventional system identification, fuzzy modeling (Jang et al. 1992), neural networks (Nguyen and Widrow 1990) as well as Support Vector Machine (SVM) to model dynamical systems.

2.4 Summary

In this chapter, biological-inspired control as well as the concept of learning and adaptation was investigated. Two main approaches have been proposed: Continuous Control and Intermittent Control. Whilst Continuous Control helped in understanding human control, multiple fundamental aspects of human control are not covered. Intermittent Control has been introduced to be able to model uncovered aspects such as the psychological refractory period. The Intermittent Control framework used for this work is based on the implementation made by Gawthrop and Wang 2009.

This implementation relies on an underlying Continuous Controller with an additional triggering mechanism as well as a Hold. The Hold is used to generate trajectory during the open-loop intervals. This Hold is originally based on two main approaches: ZOH and SMH. Whilst ZOH simply holds a constant value during the entire open-loop interval, SMH relies on the linear representation of the closed-loop dynamics of the system. This requires to know the system matrices in the state-space representation of the system.

Learning and adaptation have been previously integrated within the Intermittent Control framework (Martín 2018). The work presented in this thesis combines two approaches to tackle the redesign of the controller when needed. First, Reinforcement Learning (direct approach) is proposed to estimate the optimal state feedback gains present in the controller. Secondly, Data Informativity (indirect approach) is proposed to redesign the Hold block as well as the state feedback gains. This approach can also redesign the hold as the system matrices are getting estimated by the algorithm. Finally, Gaussian Processes are introduced as an alternative to model the system dynamics inside the controller. This allows incorporating a probabilistic non-linear element into it.

Chapter 3

Intermittent Control and Learning

3.1 Introduction

Open-loop trajectories combined with intermittent feedback is the key design principal of the Intermittent Controller. As described by Gawthrop and Wang 2011, the intermittent control relies on a continuous controller model, following *observer-predictor-feedback* design (Kleinman 1969). Figure 3.1 shows the block diagram of this implementation in the context of human motor control and is composed of four main parts:

– A plant, corresponding to the Neuro-muscular system (NMS) block combined in series with the system block. The use of a NMS block here is to represent a model that captures the dynamics of the muscles/actuators in the human body. The output of this block $\mathbf{u}_e$, combined with a disturbance $\mathbf{d}$ is used as an input for the system. The output of the system is represented by $\mathbf{y}$.

– An observer block, which provides an estimate of the missing states that cannot be accessed directly from the system.

– A predictor and a delay block, which are used to introduce a delay in the feedback. This represents delays present in the human controller, for example as due to signal propagation in the central nervous system. The predictor block is here to predict states to overcome these delays.
– A state feedback block, which has the purpose to compute the control input needed for the plant. This is based on the predicted states and not the actual states of the system.

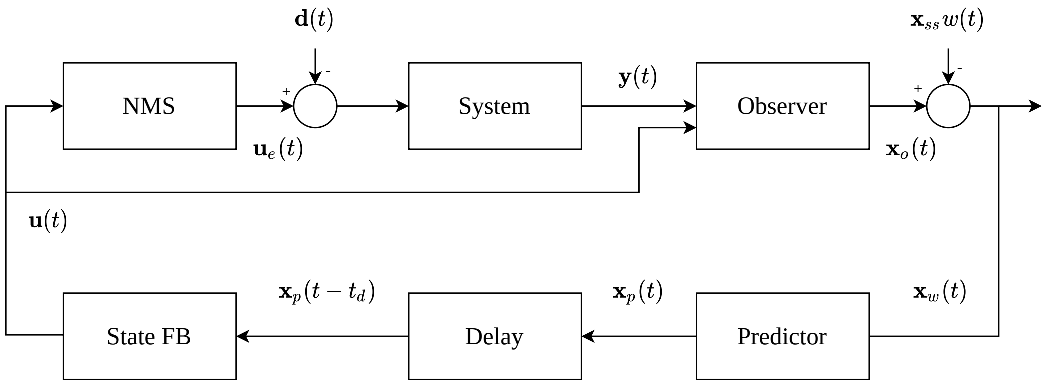

Figure 3.1: **u** and $\mathbf{u}_e$ represent the input and the output of the Neuro-Muscular System block and d represents the disturbance, respectively. y is the output of the system. x_o is the observed states and the product $x_{ss}w$ represents the state version of the set-point w. x_p are the predicted states and t_d is the time delay compensated by the predictor. The thin and thick arrows are respectively the scalar signals and vector versions for the case of a single-input, single-output system. This figure is based on the representation given by Gawthrop and Wang 2011

In this chapter, first the theory of Continuous control is presented. Then in section 3.3, Intermittent Control theory is explained, augmenting the continuous control implementation with triggering and a generalized hold. System identification theory is then presented, starting with Reinforcement Learning and Data Informativity in section 3.4 and 3.5 respectively, followed by Gaussian Processes in section 3.6.

3.2 Continuous control

The fundamental aspect of implementing an intermittent controller relies on the stability of the design of the continuous controller shown in Figure 3.1. To correctly design this underlying continuous controller, linear control theory is used, and it is necessary to perform the following steps: observer design, state-predictor design, state feedback design, and finally steady-state design. The details of the implementation of each of these blocks are described in section 3.2.2, 3.2.3 and 3.2.4.

3.2.1 System

The state-space system follows these equations:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{B}_d\mathbf{d}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \tag{3.1}$$

where:

- $\mathbf{x}(t)$ is the $n \times 1$ states vector at time t,
- $\mathbf{u}(t)$ is the $n_u \times 1$ control input vector at time t,
- $\mathbf{d}(t)$ is the $n_u \times 1$ disturbance vector at time t,
- $\mathbf{y}(t)$ is the $n_o \times n$ task vector at time t,
- $\mathbf{x}_0$ is the states vector at initial time $t = 0$
- $\mathbf{A}$ is the $n \times n$ matrix corresponding to the dynamics of the plant,
- $\mathbf{B}$ is the $n \times n_u$ input matrix,
- $\mathbf{B}_d$ is the disturbance matrix,
- $\mathbf{C}$ is the $n_y \times n$ output matrix,

– $\mathbf{D}$ is the $n_y \times n$ feed-through matrix.

n, n_u and n_y represent the number of states in the system, the number of input and the number of output respectively. It is implicitly expected that the plant is both *c*ontrollable and *o*bservable.

3.2.2 Observer

Contrary to simulated systems, in a real-world system usually not all states can be read by sensors. In this case, it is necessary to use an observer to estimate the *missing* states. The observer is a dynamical model that reconstructs the full states of the system x_o from measurements of the system outputs. Using the state-space approach, it is possible to design an observer based on Kwakernaak and Sivan 1974:

$$\dot{\mathbf{x}}_o(t) = \mathbf{A}_o\mathbf{x}_o(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{L}[\mathbf{y}_o(t) - \mathbf{v}_y(t)] \tag{3.2}$$

where:

– $\mathbf{A}_o = \mathbf{A} - \mathbf{L}\mathbf{C}_o$ is the matrix describing the observer dynamics.
– $\mathbf{x}_o(t)$ is the observed states at time t,
– $\mathbf{y}_o(t)$ is the measured output signals from the system at time t,
– $\mathbf{v}_y(t)$ is the measurement noise at time t,
– $\mathbf{C}_o$ is the $n_o \times n$ observer matrix,
– $\mathbf{L}$ is the $n \times n_o$ Observer Gain Matrix, which can be design using different approach, such as pole placement or linear-quadratic optimization.

Moreover, the observer can be augmented as a disturbance observer, if the plant contains disturbance dynamics (Goodwin et al. 2001).

3.2.3 Predictor

Time delays can introduce performance issues and instability in the system if they are not handled properly on the feedback loop and are present in human control systems. To overcome the time delay that can be contained in some systems and controllers, a predictor can be used. Using Smith 1959 as a starting point, multiple versions of a state predictor have been implemented such as Kleinman 1969; Gawthrop 1977. This section focuses on the implementation of Kleinman 1969 and the equation of the state-predictor can be written as:

$$\mathbf{x}_p(t+\Delta) = e^{\mathbf{A}\Delta}\mathbf{x}_0(t) + \int_0^{\Delta} e^{\mathbf{A}t'}\mathbf{B}\mathbf{u}(t-t')dt' \tag{3.3}$$

where:

- Δ is time interval where the predicted stated are estimated,
- $\mathbf{x}_p(t)$ are the predicted states at time $t+\Delta$,
- $\mathbf{x}_{p,0}(t)$ are the states at the initial condition of this estimation.

The first term of 3.3 is the state-transition and the second term is known as a convolution integral. This integral can be a bottleneck for real-time systems: it is important to keep in consideration the solution accuracy in opposition to the execution speed.

3.2.4 State Feedback

The state feedback control law can be defined as follows:

$$\mathbf{u}(t) = -\mathbf{k}\mathbf{x}_p(t) \tag{3.4}$$

where:

– $\mathbf{x}_p(t)$ is the predicted state vector at time t,

– $\mathbf{k}$ is feedback gain vector,

– $\mathbf{u}(t)$ is the control input at time t.

This feedback gain $\mathbf{k}$ can be computed via different methods such as Pole Placement (PP) (Laub and Wette 1984) as well as Linear Quadratic Regulator (LQR) (Anderson and Moore 1990) using dynamic programming. $\mathbf{k}$ has to be designed carefully to ensure the system's stability. To do that, the real part of all eigenvalues of the matrix A_c need to be negative. This ensures an exponential convergence to zeros as $t \to \infty$. The following section explains the LQR approach and the way to solve it.

Linear quadratic regulator

The system in (3.1) is subject to the following infinite-horizon optimal control problem:

$$V(\mathbf{x}_0) = \min_{\mathbf{u}(t)} \int_0^\infty \left(\mathbf{x}(\tau)^T \mathbf{Q} \mathbf{x}(\tau) + \mathbf{u}(\tau)^T \mathbf{R} \mathbf{u}(\tau) \right) d\tau \tag{3.5}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are semi-positive and positive definite matrices respectively. The solution to this minimisation problem is obtained by solving an Algebraic Riccati Equation (ARE) of the following form:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} = 0\,. \tag{3.6}$$

Assuming that the system matrices $\mathbf{A}$ and $\mathbf{B}$ are known, then it is possible to solve for $\mathbf{P}$, which is an $n \times n$ real symmetric positive-definite matrix. The state feedback that minimises (3.5) is then given by

$$\mathbf{k} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{3.7}$$

One way to solve this optimization problem is to use Dynamic Programming, where the ARE in Equation (3.6) is solved backward in time, extending the horizon to infinity. This reduces the problem from an infinite horizon optimization (Schochetman and Smith 1989) to a finite one. The only requirement, as stated before, is that the matrices $\mathbf{A}$ and $\mathbf{B}$ should be known.

3.2.5 Steady State design

To respect the control law known as the *regulation* and the *tracking* problems, it is necessary to design the steady-state characteristics carefully to reduce steady-state error to zero. It is possible to re-write (3.1) as:

$$\begin{aligned} \mathbf{0}_{n\times 1} &= \mathbf{A}\boldsymbol{x}_{ss}(t) + \mathbf{B}\boldsymbol{u}_{ss}(t) \\ \boldsymbol{y}_{ss}(t) &= \mathbf{C}\boldsymbol{x}_{ss}(t) \end{aligned} \tag{3.8}$$

where $\boldsymbol{x}_{ss}$, $\boldsymbol{u}_{ss}$ and $\boldsymbol{y}_{ss}$ are the steady-state versions of the states, inputs and outputs. For the case where $\boldsymbol{y}_{ss} = 1$, (3.8) can be rewrite as follows:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{ss} \\ \boldsymbol{u}_{ss} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n\times 1} \\ \mathbf{1} \end{bmatrix} \tag{3.9}$$

By solving for $\boldsymbol{x}_{ss}$ and $\boldsymbol{u}_{ss}$, the control input is equivalent to

$$\begin{aligned} \mathbf{u}(t) &= -\mathbf{k}(\mathbf{x}(t) - \boldsymbol{x}_{ss}w(t)) + \boldsymbol{u}_{ss}w(t) \\ &= -\mathbf{k}\mathbf{x}(t) + (\boldsymbol{u}_{ss} + \mathbf{k}\boldsymbol{x}_{ss})w(t) \end{aligned} \tag{3.10}$$

where $w(t)$ is the setpoint. It is possible to define $r = \boldsymbol{u}_{ss} + \mathbf{k}\boldsymbol{x}_{ss}$ such as:

$$\mathbf{u}(t) = -\mathbf{k}\mathbf{x}(t) + \mathbf{r}w(t) \tag{3.11}$$

One of the main advantages of this approach is the steady-state computation which can be done offline: only the feedback gain $\mathbf{k}$ needs to be determined to ensure the stability of the system. Common approaches can be used such as *Linear Quadratic Regulator* or *Pole-placement* (presented in Section 3.2.4). In addition, the control error $\mathbf{x}_w(t)$ can be computed as:

$$\mathbf{x}_w(t) = \mathbf{x}_o(t) - \mathbf{x}_{ss}w(t) \tag{3.12}$$

3.3 Intermittent control

IC, as discussed by Gawthrop and Wang 2011, is an extension of a continuous controller design. This approach adds a triggering algorithm, a Generalized Hold which introduces a variation on which states are used by the state feedback block as shown in Figure 3.2.

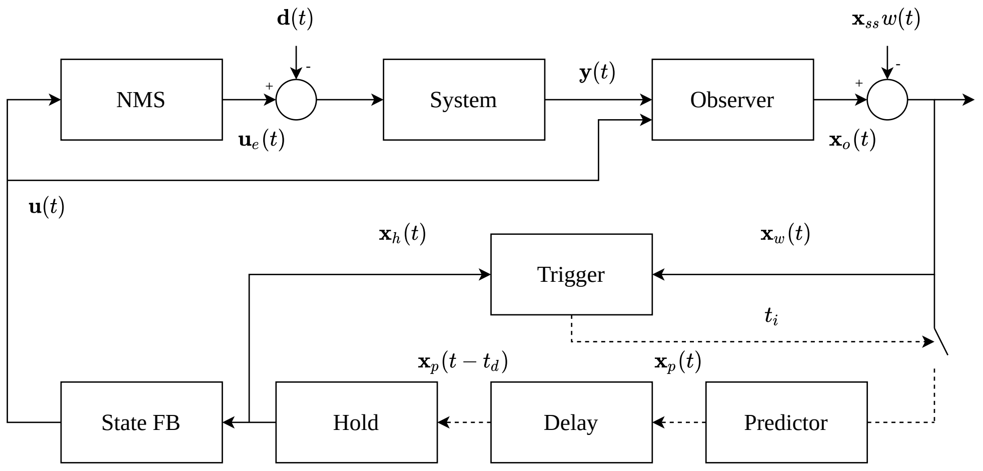

Figure 3.2: Block diagram of the intermittent controller designed by Gawthrop and Wang 2011. The state-estimates x_w is compared to the hold states x_h on the Trigger block. According to a certain criteria, an event at time t_i is generated: this closes the feedback loop. At time t_i, the Hold is reinitialised with the predictor states. The states x_h are used to generate the control input u. The signals shown by dashed lines are only updated at time t_i when the feedback-loop is closed. The thin and thick arrow are scalar signals and vector version for the case of a single-input, single-output system respectively. This diagram is based on the representation given in Gawthrop and Wang 2011

This section introduces and describes the notion of intermittency as well as the different time frames present on the IC. Furthermore, the open-loop period based on a generalized hold is described.

3.3.1 Time frames and Triggering

Different time frames involve in Intermittent Control

In the case of IC, different time frames need to be introduced. Three different time frames are used in this type of control: continuous, discrete, and intermittent times and can be described as follows:

Continuous Time, is the common time used by the system to evolve, denoted as t.

Discrete Time, is the time sampled every intermittent interval $\Delta(i)$, denoted as $t(i)$ and described as:

$$\Delta_i = t(i+1) - t(i) \tag{3.13}$$

Where $t(i)$ represents the time when an event is triggered. In addition, the sampling delay Δ_s is introduced to represent the time delay between the event and the observed states used for the feedback. This delay can be described as follows:

$$t^s(i) = t(i) - \Delta_s \tag{3.14}$$

Intermittent Time, is the time describing the length of an open-loop interval and is denoted as τ. It is reset to zeros at each feedback loop event and is described as:

$$\tau = t - t(i) \tag{3.15}$$

The time between an event and the end of the sampling delay can be describe as:

$$\tau^s = t - t^s(i) \tag{3.16}$$

It is also important to introduce the minimum open-loop interval Δ_{min} which represents the time limit before another event can trigger the feedback loop.

$$0 < \Delta_{min} < \Delta_i \tag{3.17}$$

Trigger

Triggering in IC can be controlled using two different modes: Clock-Driven or Event-Driven. In Clock-Driven mode, the feedback loop every t_i generated by a clock and is constant. This time t_i needs to be larger than Δ_{min}.

In Event-Driven mode, the open-loop hold state x_h and the closed-loop observed state x_w are used to trigger a sampling event. Different approaches can be used, as presented by Gawthrop et al. 2011. One approach is the error quadratic function which can be described as:

$$\begin{aligned} &\mathbf{e}_{hp}(t) = \mathbf{x}_h(t) - \mathbf{x}_w(t) \\ &\mathbf{e}_{hp}^T(t)\mathbf{Q}_t\mathbf{e}_{hp}(t) - q_t^2 >= 0 \end{aligned} \tag{3.18}$$

where:

- $\mathbf{x}_h(t)$ is the open-loop hold state at time t,
- $\mathbf{x}_w(t)$ is the closed-loop observed state at time t,
- $\mathbf{Q}_t$ is a positive semi-definite matrix,
- q_t is an arbitrary threshold,

If the quadratic error exceeds the $t$hreshold q_t and $t_i > \Delta_{min}$, the feedback loop is closed. If the threshold is set to zero, then IC in Event-Driven mode acts similarly to a Clock-Driven IC.

3.3.2 Generalized Hold

During the open-loop period, the control input needed by the system is generated by the hold block. Two approaches can be used as described by (Gawthrop et al. 2011): (1) a System-Matched Hold (SMH) implementation, or (2) the Tapping Hold based on Laguerre functions.

In this section, only the first approach is going to be described. The main idea behind this algorithm is to implement the dynamics of the closed-loop systems matrix A_c as presented by Gawthrop and Wang 2011. The closed-loop state space vector can be defined as $\mathbf{x}_c$ and replacing $\mathbf{u}$ in (3.1), we obtain

$$\begin{cases} \dot{\mathbf{x}}_c(t) = \mathbf{A}_c\mathbf{x}_c(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}_c(t) \\ \mathbf{x}_c(0) = \mathbf{x}_0 \end{cases} \tag{3.19}$$

where

$$\mathbf{A}_c = \mathbf{A} - \mathbf{B}\mathbf{k} \tag{3.20}$$

Figure 3.3 compares the state evolution using ZOH and SMH during multiple open-loop intervals. The states used for the states' feedback depends on whether the trigger is activated or not and follows the following pattern:

$$\mathbf{x}(t) = \begin{cases} \mathbf{x}_p(t) & \text{if trig,} \\ \mathbf{x}_c(t) & \text{otherwise.} \end{cases} \tag{3.21}$$

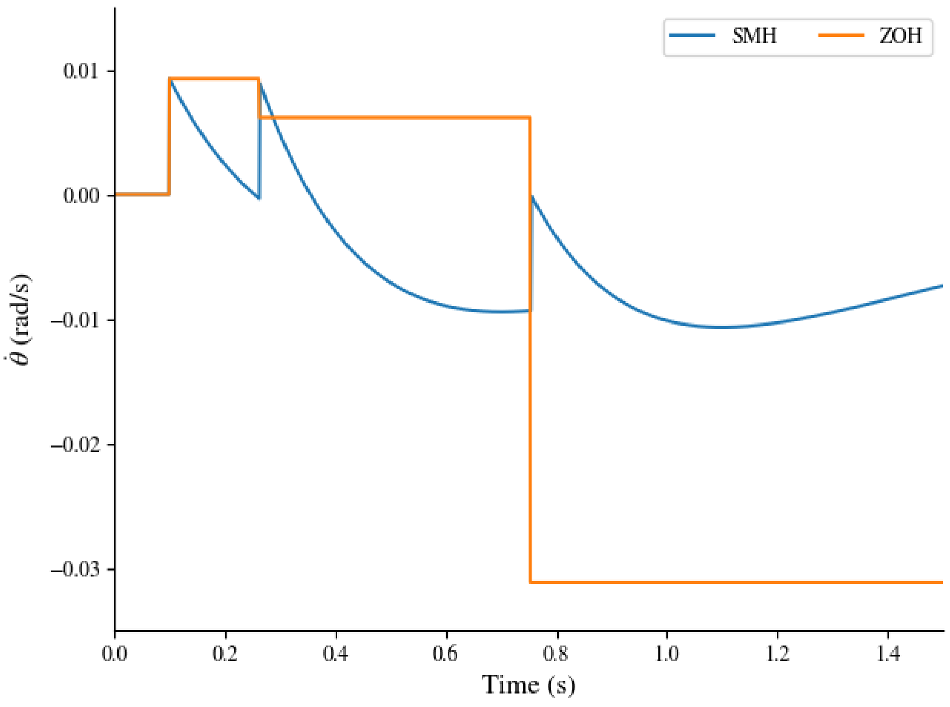

Figure 3.3: Comparison between Zero Order Hold (ZOH) and System Matched Hold (SMH) during multiple open-loop intervals. Initial states are identical for both Holds. ZOH is constant during the entire open-loop whilst SMH generated states are converging towards zero.

3.4 Reinforcement Learning

In this section, an overview of Reinforcement Learning is given. The Policy Iteration algorithm is developed to solve the Algebraic Riccati Equation (ARE) and to extract the state feedback gain vector.

3.4.1 Environment-Agent Structure

To understand the working principle of RL, it is necessary to introduce some terminology. As presented in Figure 3.4, there is an *Environment* and an *Agent*. As introduced in section 2.3.3.1, the environment represents the system (also named the world) that the agent is trying to control. At each iteration, the agent interacts with the Environment using some *action* $\mathbf{A}$ (similar to a control input). According to this action, the Environment returns the (possibly partial) states s of the system as well as a reward r associated with it. Denoted by $V(s)$, the value function efficiently measures the potential future rewards we may obtain based on the current state. The reward is a number representing how good or bad the current state of the Environment is. Through an episode, this loop is performed multiple times.

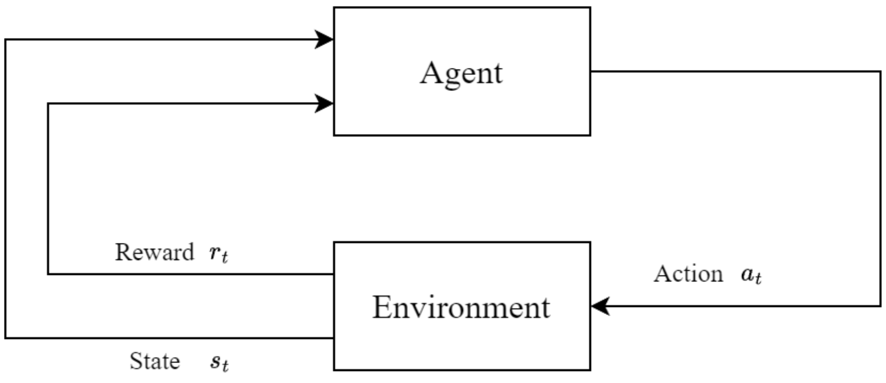

Figure 3.4: Baseline architecture of RL

States, similar as presented in section 3.2.2, might be on partial in some scenario. Two different notations are used: s is the complete description of the states while o is the partial description (similar to the observed states $\mathbf{x}_o$ in the context of IC).

Actions can be represented in two different action spaces. Most artificial settings, e.g. video game environments have a discrete action space, meaning having a finite number of moves available. Real-world systems, such as robots evolve in a continuous action space. This knowledge of the system is very important, as it limits which algorithm can be used in a specific Environment.

The sequence of states and actions is called the *trajectory*. The *Policy* represents a rule under which the agent is trying to find the optimal trajectory for a specific goal. The policy can be either deterministic or stochastic. As deterministic models are fully determined by some initial conditions and parameter values, the stochastic approach is closer to real-world representation due to the presence of inherent randomness.

3.4.2 Solving the ARE via Policy Iteration

The class of algorithms named Policy-Iteration is based on a two step process: the *policy evaluation, equivalent to system identification in Control Theory, and the policy improvement, equivalent to optimal control.* One approach presented by Vrabie et al. 2007 has been to find the optimal control policy without solving directly Bellman's equation, and has shown very good results in finding the optimal state-feedback gain vector in the case of a continuous-time systems with unknown internal dynamics. We used this method to obtain the optimal values of the state-feedback gain in Equation (3.4), with minor modifications to make it work under the IC framework. Moreover, this algorithm estimates $\mathbf{k}$ without the need to know the system matrix $\mathbf{A}$ in (3.1) on page 44.

Assuming that $\mathbf{k}$ is a set of gains that stabilises (3.1), then the closed-loop system can be described by

$$\begin{aligned} \dot{\mathbf{x}}_c(t) &= \mathbf{A}_c\mathbf{x}_c(t) \\ \mathbf{x}_c(0) &= \mathbf{x}_0 \,, \end{aligned} \tag{3.22}$$

where $\mathbf{A}_c = \mathbf{A} - \mathbf{B}\mathbf{k}$. Then, an infinite horizon quadratic cost based on $\mathbf{k}$ can be defined as follows

$$V(\mathbf{x}(t)) = \int_t^\infty \mathbf{x}(\tau)^T \left(\mathbf{Q} + \mathbf{k}^T\mathbf{R}\mathbf{k}\right)\mathbf{x}(\tau)d\tau = \mathbf{x}^T(t)\mathbf{P}\mathbf{x}(t)\,, \tag{3.23}$$

where $\mathbf{P}$ is also the solution of a Lyapunov matrix equation of the following form

$$(\mathbf{A} - \mathbf{B}\mathbf{k})^T\mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{B}\mathbf{k}) = -\left(\mathbf{k}^T\mathbf{R}\mathbf{k} + \mathbf{Q}\right)\,. \tag{3.24}$$

In (3.24), $V(\mathbf{x}(t))$ is a Lyapunov function for the system in (3.1). The cost function in (3.23) can be re-expressed as

$$V(\mathbf{x}(t)) = \int_t^{t+T} \mathbf{x}^T(\tau)\left(\mathbf{Q} + \mathbf{k}^T\mathbf{R}\mathbf{k}\right)\mathbf{x}(\tau)d\tau + V(\mathbf{x}(t+T)) \tag{3.25}$$

By using $V(\mathbf{x}(t)) = \mathbf{x}^T(t)\mathbf{P}\mathbf{x}(t)$ and with an initial stabilising controller $\mathbf{k}_0$, the policy iteration method can be used recursively in an online manner as follows:

$$\mathbf{x}^T(t)\mathbf{P}\mathbf{x}(t) = \int_t^{t+T} \mathbf{x}^T(\tau)\left(\mathbf{Q} + \mathbf{k}^T\mathbf{R}\mathbf{k}\right)\mathbf{x}(\tau)d\tau + \mathbf{x}^T(t+T)\mathbf{P}\mathbf{x}(t+T) \tag{3.26}$$

$$\mathbf{k}_i = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{3.27}$$

This formulation of the policy iteration method comes from the work by Murray et al. 2002.

Notice that (3.26) and (3.27), do not depend on the system matrix **A** but they do assume knowledge of **B** to calculate the optimal gain **k**. Therefore, by observing the states $\mathbf{x}(t)$ and $\mathbf{x}(t+T)$, and knowing **B** it is possible to implement the algorithm online to find the individual entries of the matrix **P**, which is the solution to the ARE defined in (3.24). To do this, we can write $\mathbf{x}^T(t)\mathbf{P}\mathbf{x}(t)$ as:

$$\mathbf{x}^T(t)\mathbf{P}_i\mathbf{x}(t) = \bar{p}_i^T\bar{\mathbf{x}}(t) \tag{3.28}$$

where $\bar{\mathbf{x}}(t)$ is a quadratic polynomial basis vector that has elements of the form

$$\left\{\mathbf{x}_i(t)\mathbf{x}_j(t)\right\}_{i=1,n;j=i,n}. \tag{3.29}$$

Also, $\bar{p}$ can be understood as the result of a vector valued matrix function $v(.)$ that acts on matrix **P** as follows

$$\bar{p} = v(\mathbf{P}). \tag{3.30}$$

The purpose of $v(.)$ is to re-arrange the matrix of the elements of **P** into a column vector by stacking the diagonal terms and also the off-diagonal elements in the upper triangular part of **P**. The off-diagonal terms are taken as $2\mathbf{P}_{ij}$.

Using the formulation in (3.28), we can re-write (3.26) as

$$\begin{aligned}\bar{p}^T\left(\bar{\mathbf{x}}(t)-\bar{\mathbf{x}}(t+T)\right) &= \int_t^{t+T}\mathbf{x}(\tau)^T\left(\mathbf{Q}+\mathbf{k}^T\mathbf{R}\mathbf{k}\right)\mathbf{x}(\tau)d\tau \\ &\equiv d\left(\bar{\mathbf{x}}(t),\mathbf{k}\right)\end{aligned} \tag{3.31}$$

In (3.31), the vector $\bar{p}$ is what we want to estimate, which corresponds to the relevant elements in matrix **P**, the difference $\mathbf{x}(t)-\mathbf{x}(t+T)$ can be measured from the evolution of the system and acts as a regression vector of incoming data. However, the important part of (3.31) lies on the right-hand side, which is equivalent to the calculation of a classical

LQR cost function using a specific control signal defined by a gain **k**:

$$d\left(\bar{\mathbf{x}}(t),\mathbf{k}_i\right) \equiv \int_t^{t+T} \mathbf{x}(\tau)^T \left(\mathbf{Q}+\mathbf{k}_i^T\mathbf{R}\mathbf{k}_i\right)\mathbf{x}(\tau)d\tau \,. \tag{3.32}$$

The cost in (3.32) can be calculated by measuring only the states over the interval $[t, t+T]$, since all the other quantities are known. This is the reason why there is the need for an initial stabilizing gain, which will ensure the correct computation of $d\left(\bar{\mathbf{x}}(t),\mathbf{k}_i\right)$ over the first interval.

3.4.3 Solving for P

The main idea is to obtain the elements of **P**, therefore, at each iteration step i of the algorithm (these steps are different than those imposed by the sampling interval) and after a sufficient number of state-trajectory points are collected (under the influence of the same $\mathbf{k}_i$), a least-squares method can be used to solve for $\bar{p}_i$. This effectively minimises the error between $d\left(\bar{\mathbf{x}}(t),\mathbf{k}_i\right)$ and $\bar{p}_i^T\left(\bar{\mathbf{x}}(t)-\bar{\mathbf{x}}(t+T)\right)$.

The condition of having a sufficient number of state-trajectory points, to solve for **P**, is met if a minimum of $N \geq n(n+1)/2$ measurements $\bar{\mathbf{x}}^i$ are collected using the time-interval T. N in this case corresponds to the number of independent elements in the matrix **P**. For instance, a 3×3 matrix would have 6 independent elements, three corresponding to the diagonal terms, and the other three are in the upper-triangular part. The least-squares solution is then given by:

$$\bar{p}_i = \left(XX^T\right)^{-1}XY \,, \tag{3.33}$$

where

$$X = \left[\bar{\mathbf{x}}^1{}_\Delta \quad \bar{\mathbf{x}}^2{}_\Delta \quad \dots \quad \bar{\mathbf{x}}^N{}_\Delta\right] \,, \tag{3.34}$$

$$\bar{\mathbf{x}}^i{}_\Delta = \bar{\mathbf{x}}^i(t) - \bar{\mathbf{x}}^i(t+T) \,, \tag{3.35}$$

$$Y = \left[d\left(\bar{\mathbf{x}}^1(t), \mathbf{k}_i\right) \quad d\left(\bar{\mathbf{x}}^2(t), \mathbf{k}_i\right) \quad \dots \quad d\left(\bar{\mathbf{x}}^N(t), \mathbf{k}_i\right)\right]^T . \tag{3.36}$$

The resulting architecture brings adaptive capabilities to a standard state-feedback controller with the distinction that the system has to be augmented with an extra state $V(t)$ where $\dot{V}(t) = \mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)$, to compute the associated cost of a given control input $\mathbf{u}(t)$ and subsequently of a feedback gain $\mathbf{k}$. This can be seen as calculating $d\left(\bar{\mathbf{x}}(t), \mathbf{k}_i\right)$ or the right hand side of (3.32). This ensures that the algorithm would eventually converge to the optimal control input after several improvements or updates of the state-feedback gain $\mathbf{k}$. An important condition is that this algorithm requires enough excitation to solve for $\mathbf{P}$, if the excitation is lost because the states reached the steady state then the updates of $\mathbf{k}$ should be stopped to avoid incorrect estimates.

3.5 Data Informativity Framework

In this section, Data Informativity theory is described and explained. Data Informativity has been defined as part of system analysis and control design. The notion of informativity is usually perceived as information contained in the data, hence playing a major role in system identification.

In this implementation of the Data Informativity algorithm, measurements of the data are assumed to be exact, not corrupted by noise. This implementation fully relies on the algorithm described in van Waarde et al. 2020. Van Waarde et al. 2023 has extended this algorithm to handle noisy data, but it is not presented here. Assuming the representation of the true system follows the Equation 3.1, the systems equations can be written in the form:

$$\begin{bmatrix} X_+ \\ Y_- \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} X_- \\ U_- \end{bmatrix} \tag{3.37}$$

where:

- X_- is the state at the current time.
- U_- is the control input at current time.
- Y_- is the output corresponding to the next iteration.
- X_+ is the state at the next iteration.

With multiple data points, X_-, U_-, Y_-, X_+ can be written as an array of states and control input as follow:

$$\begin{aligned} X_- &:= \begin{bmatrix} X_-^1 & X_-^2 & \dots & X_-^q \end{bmatrix} \\ U_- &:= \begin{bmatrix} U_-^1 & U_-^2 & \dots & U_-^q \end{bmatrix} \\ Y_- &:= \begin{bmatrix} Y_-^1 & Y_-^2 & \dots & Y_-^q \end{bmatrix} \\ X_+ &:= \begin{bmatrix} X_+^1 & X_+^2 & \dots & X_+^q \end{bmatrix} \end{aligned} \tag{3.38}$$

where q represent the length of the data.

Then, using the theorem 34 in van Waarde et al. 2020, if

$$\operatorname{rank} \begin{bmatrix} X_- \\ U_- \end{bmatrix} = n + n_u \tag{3.39}$$

where:

- n is the number of states in the system

- n_u is the number of control input

is equivalent to Equation (3.40)

$$\begin{bmatrix} X_- \\ U_- \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.40}$$

It is then possible to solve:

$$\begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} X_- \\ U_- \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.41}$$

To solve Equation (3.41), multiple approaches can be used. However, the implementation within Intermittent Control framework is using the least square algorithm.

Once V_1 and V_2 are computed, it is then possible to compute the following:

$$\begin{bmatrix} \hat{\mathbf{A}} & \hat{\mathbf{B}} \\ \hat{\mathbf{C}} & \hat{\mathbf{D}} \end{bmatrix} = \begin{bmatrix} X_+ \\ Y_- \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix} \tag{3.42}$$

Hence, $\hat{\mathbf{A}} = X_+V_1$ and $\hat{\mathbf{B}} = X_+V_2$. Once $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are estimated, it is possible to use the LQR method with pre-defined Q_c and R_c matrices to compute the state feedback $\mathbf{k}$ (in Section 3.2.4). Due to the data being discretized based on dt, it is necessary to get back to a continuous form of matrices $\mathbf{A}$ and $\mathbf{B}$.

3.6 Gaussian Processes

In this section, the theory behind Single-Task and Multi-Task Gaussian Processes is explained as well as the different covariance functions that can be used with the Intermittent Control Framework. The hyper-parameters optimization via the minimisation of the Negative Log Marginal Likelihood is also explained. Then, the concept of sparse GPs is introduced.

3.6.1 Single-Task GP

A simple way to understand Gaussian Processes (GP) is to start with single-task GP. The objective is to create a mapping of a function $y(t)$ linking a set of inputs $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]^T$ and output $Y = [y_1, y_2, ..., y_N]$. The input X is n multidimensional while the output Y is one dimensional. Consider the system:

$$y(t) = f(x(t)) + \varepsilon(t) \tag{3.43}$$

where $\varepsilon(t) \sim \mathcal{N}(0, \sigma_f)$ is white Gaussian noise with variance σ_f and $x(t) \in \mathbb{R}^n$ is the input state at the instant t. For a multidimensional system, one approach is to create a GP per output state. Another alternative is to use Multi-Task GP.

To find a GP fitting the output data Y, we assume that $y(t) \sim \mathcal{N}(0, K)$ where $K = \Sigma + \sigma_f I$ is the kernel matrix.

3.6.2 Multi-Output GP

As mentioned above, in the case of single-task GP, each GP is only mapping the relation to the input states $\mathbf{x}$ to a single output y. However, each output might be correlated to the other, and treating it as a multitude of single independent tasks could result in missing information and be sub-optimal.

To take into account this potential correlation between tasks, Bonilla et al. 2007 implemented a multi-task model. In addition to the regular single-task kernel present in the single-task GP, the multi-task model adds a "free-form" covariance matrix to model the inter-task dependencies. The covariance matrix is now expressed as:

$$K(x,x') = K_f(x,x') \otimes K_x(x,x') \tag{3.44}$$

where $\otimes$ is the Kronecker product, $K_f(x,x')$ the inter-task matrix and $K_x(x,x')$ is the Kernel Matrix (similar to the single-task case).

There is an addition of $n_t \times n_t$ hyper-parameters where n_t is the number of tasks. This can be reduced by taking the Cholesky decomposition of the matrix $K_f(x,x')$ resulting in $n_t + (n_t \times (n_t - 1))/2$ hyper-parameters. This approach does not scale well when the number of tasks is high.

3.6.3 Hyper-parameters and Covariance functions

Different covariance functions, also called kernel functions, can be used depending on the complexity of the function $y(t)$ that is modeled. The kernel functions presented here are the Square exponential Kernel, the Periodic Kernel, the Matern, and the Noise Kernel. Even though non-stationary kernel functions exist, this work is using previous research such as Deisenroth et al. 2015, where simple non-linear kernel functions can be used to model a dynamical non-linear system. In addition, linear kernel functions are not used in this work as GP is introduced in IC to capture the non-linearity present in the plant. Kernel functions presented here described in Williams and Rasmussen 2006.

To fit a GP to the data, it is necessary to tune the coefficient of the kernel: the hyper-parameters. The number of hyper-parameters is linked with the complexity of the covariance function used.

3.6.3.1 Square exponential Kernel

The Square exponential kernel function, also called Radial Basis Function (RBF) kernel function, is expressed as:

$$K_{SE}(x,x') = \sigma_y^2 exp\left(-\frac{(x-x')^2}{2l^2}\right) \tag{3.45}$$

In the case of multidimensional input, Equation (3.45) can be expressed as follow:

$$K_{SE_{ARD}}(x,x') = \sigma_y^2 exp\left(-\frac{(x-x')^T\mathbf{P}^{-1}(x-x')}{2}\right) \tag{3.46}$$

where $\mathbf{P}$ is the diagonal matrix with each $1/l$ parameters in the case where $\mathbf{x}$ is multi-dimensional (also called Automatic Relevance Determination (ARD)). For this kernel, the list of the hyper-parameters is the following:

$$\zeta_{SE} = [l_{1,1}, l_{1,2}, \ldots, l_{1,n}, \sigma_y] \tag{3.47}$$

where n is the number of inputs. $l_{i,i}$ is the length-scale of the input x_i. The larger $l_{i,i}$ is, the lower is the importance of the input x_i and vice-versa. σ_y is the amplitude coefficient related to the output selected for the training.

The Square exponential kernel function is the most used function when there is no previous knowledge of the system. The main advantage of this kernel function is the smoothness of the output (see Figure 3.5 (first row) on page 68).

3.6.3.2 Periodic Kernel

The Periodic Kernel is expressed as:

$$K_{Periodic}(x,x') = \sigma_y^2 exp\left(-\frac{2sin^2(\pi|x-x'|/p)}{l^2}\right) \tag{3.48}$$

For this kernel, the list of the hyper parameters is the following:

$$\zeta_{Periodic} = [l, p, \sigma_y] \tag{3.49}$$

where p is the periodicity coefficient.

The Periodic Kernel function is very useful to fit the data with a signal with periodic behaviour (see Figure 3.5 (fourth row)). However, this specific function assumes perfect periodicity without any attenuation factor over time. One approach is to combine a periodic and a square exponential kernel function (see Section 3.6.3.5).

3.6.3.3 Matern Kernel

The Matern Kernel is expressed as:

$$K_{Matern}(x,x') = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l} |x-x'| \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}}{l} |x-x'| \right) \tag{3.50}$$

where K_{ν} is a modified Bessel function and $\Gamma(\nu)$ is the gamma function. ν is often set at $\frac{3}{2}$ or $\frac{5}{2}$ called Matern32 or Matern52.

For this kernel, the list of the hyper-parameters is similar as the Square exponential Kernel:

$$\zeta_{Matern} = [l_{1,1}, l_{1,2}, \dots, l_{1,n_x}, \sigma_y] \tag{3.51}$$

where n_x is the number of inputs. The Matern kernel is very similar to the Square Exponential kernel. However, based on the ν value, it is possible to introduce less smooth behaviours (Williams and Rasmussen 2006) as presented in Figure 3.5, rows 2 and 3.

3.6.3.4 Noise Kernel

The Noise Kernel is expressed as:

$$K_{NOISE}(x,x') = \sigma_f^2 I \tag{3.52}$$

For this kernel, there is only one hyper-parameter:

$$\zeta_{Noise} = \left[\sigma_f\right] \tag{3.53}$$

This kernel is very useful for computational purpose to assure a Positive Definite Matrix.

3.6.3.5 Kernel combination

One big advantage of using GP and kernel-based methods is to be able to mix kernels via sum or product and still get a kernel. For example, if the function $y(t)$ is periodic with an attenuation over time, it is possible to multiply a Periodic Kernel, to take into consideration the periodicity, multiplied by a Square Exponential Kernel, to compensate for the attenuation (see Figure 3.5 (last row)).

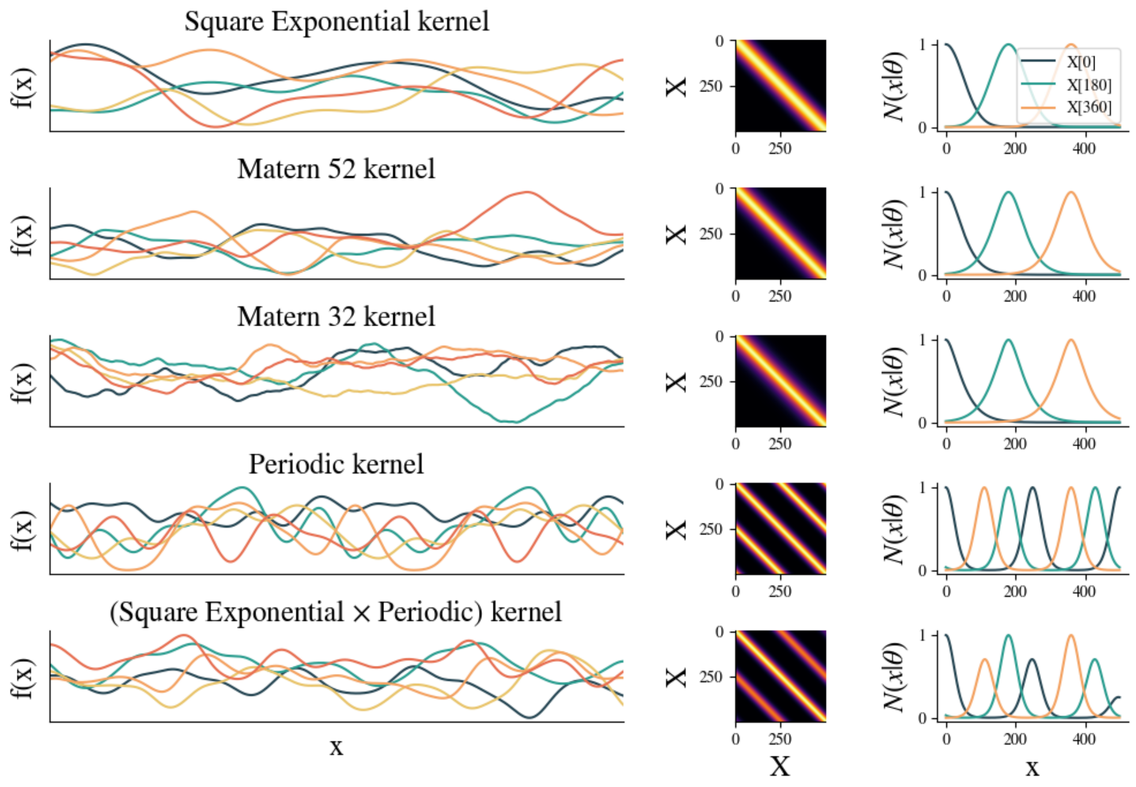

Figure 3.5: Comparison of the different kernel: Square exponential, Matern 52, Matern 32, Periodic, Square exponential × Periodic. Left column: Sample from the prior distribution. Middle column: kernel representation. Right column: slice of the kernel at 0, 180 and 360.

3.6.4 Control input impact

The first implementation of GPs within IC was using closed-loop data for training. Hence, the GP model was modeling the closed-loop dynamics of the plant under the influence of the controller. This approach required training two separate GPs: one for the Hold block, and one for the predictor. Whilst the closed-loop dynamics are required inside the Hold, it is necessary to get an open-loop dynamics model for the predictor. However, the plant operates in an unstable region of the state space, so it is difficult to gather data without the use of a controller.

In a second implementation, the control input is also part of the training, thus the GP can understand the impact of the control input on the output data. This allows us to get a single GP that can represent open and closed-loop dynamics by simply adding the properly designed control input to the GP directly. In this case, the GP is now purely representing the dynamics of the plant, with or without control input. To use the GP as closed-loop behavior inside the Hold, for example, setting the control input to $u = -kx$ is sufficient. For getting the open-loop behavior, the control input has to be set to $u = 0$.

3.6.5 Data pre-processing

Data is the most important part of the estimation to ensure enough information is contained in it. However, depending on the system the GP is trying to model, the range of data can be quite important. To keep any training data in a small state space area, it is necessary to pre-process the data before training the GP.

Starting with the output of the GP, it is converted into a difference between the next output states and the input one.

$$y_{GP} = y(t+1) - y(t) = \Delta X \tag{3.54}$$

where y_{GP} is the output used for training of the GP.

This allows us to keep the output data in a small range, directly linked to the sampling time used by the simulation. By increasing the sampling time, the range of the input is growing as the system has more time for its motion.

The second data processing is converting any angle of the system into a sine and cosine decomposition. Whilst this creates an additional hyperparameter in the GP, this is helping in the case of the pendulum, cart pole, and Furuta pendulum as the pendulum can do multiple spins. This also allows us to limit the data between -1 and 1. In the context of Intermittent Control, GP is using the states of the plant in addition of the control input generated by the controller as training data.

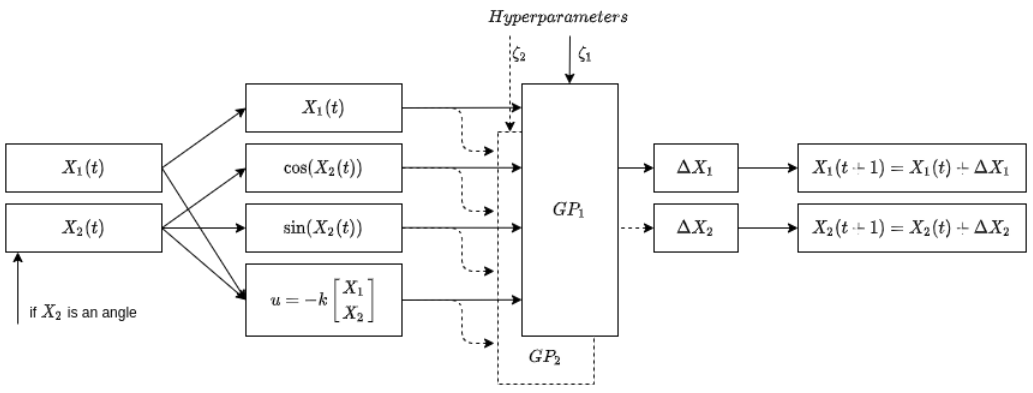

Figure 3.6: Data pre-processing diagram. If a state is an angle, it is decomposed into *sin* and *cos*, else it is directly passed to the GP. In case of the Predictor, $u = 0$ while in the hold, $u = -kX$. This diagram is representing two single task GPs (one per state). In the case of a multi task GP, the GP will output all the states.

3.6.6 Negative Log Marginal Likelihood

To estimate the best hyper-parameters for each covariance function, it is necessary to optimize a likelihood function. A multitude of likelihood functions can be optimized such as Bernoulli, Gaussian, or even Poison Likelihood. For our purpose, the Negative Log Marginal Likelihood is used which is given by:

$$log(p(y_{GP}|\mathbf{x},\zeta)) = \underbrace{-\frac{1}{2}log|\mathbf{K}|}_{\text{Complexity}} \underbrace{-\frac{1}{2}y_{GP}^T\mathbf{K}^{-1}y_{GP}y}_{\text{Data fit}} \underbrace{-\frac{1}{2}log(2\pi)}_{\text{constant}} \tag{3.55}$$

where $\mathbf{x}$ and y_{GP} is the inputs and output used for training respectively, ζ is the hyper-parameters and $\mathbf{K}$ is the Kernel. The objective is to minimize the Negative Log Marginal Likelihood.

For this optimization problem, the *minimize* function implemented by Williams and Rasmussen 2006 (also used by many GP regression algorithms such as PILCO (Deisenroth 2010)) based on the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is used for its reliable performance. In the case of the Multi-Task GP, the Polack-Ribiere algorithm is used. This method being a gradient-based optimizer, it is necessary to compute the derivative of the Negative Log Marginal Likelihood (Williams and Rasmussen 2006):

$$\begin{aligned}\frac{\partial}{\partial \zeta_j} log(p(y_{GP}|\mathbf{x},\zeta)) &= \frac{1}{2} y_{GP}^T K^{-1} \frac{\partial K}{\partial \zeta_j} K^{-1} y_{GP} - \frac{1}{2} tr\left(K^{-1}\frac{\partial K}{\partial \zeta_j}\right) \\ &= \frac{1}{2} tr\left((\alpha\alpha^T - K^{-1})\frac{\partial K}{\partial \zeta_j}\right)\end{aligned} \tag{3.56}$$

where $\alpha = K^{-1}y$ and j is the hyperparameter index.

3.6.7 Prediction

Once the hyper-parameters are estimated by solving the negative log Marginal likelihood, it is possible to predict an unknown point. For the collection of outputs $(y_{GP}, y_{pred})^T$, we can write:

$$\begin{pmatrix} y_{GP} \\ y_{pred} \end{pmatrix} \sim \mathcal{N}(0, K_{pred}) \tag{3.57}$$

where

$$K_{pred} = \begin{bmatrix} [K(\mathbf{x},\mathbf{x})] & [K_*] \\ [K_*^T] & [K_{**}] \end{bmatrix} \tag{3.58}$$

with $K_* = K(\mathbf{x}, x_{pred})$ and $K_{**} = K(x_{pred}, x_{pred})$. x_{pred} is the input state used for the prediction. The prediction is a Gaussian Distribution with mean and variance (Williams and Rasmussen 2006):

$$\begin{aligned} \mu(x_{n+1}) &= K_*(x_{n+1})^T K^{-1} y_{GP} \\ \sigma^2(x_{n+1}) &= K_{**}(x_{n+1}) - K_*(x_{n+1})^T K^{-1} K_*(x_{n+1}) \end{aligned} \tag{3.59}$$

3.6.8 Sampling from the distribution

Gaussian processes can generate randomly generated samples following a kernel using multivariate normal sampling. This can be done by sampling curve from $\mathcal{N}(\mu(x_{n+1}), K_{**}(x_{n+1}))$.

In the case of dynamical systems, it is not possible to obtain x_{n+1} in a single computation; it is necessary to use the GP as a multi-step ahead prediction using the output from the previous iteration. Once the prediction is done for a pre-determined horizon, it is possible to compute $K_{**}(x_{n+1})$. $\mu(x_{n+1})$ is the multi-step ahead prediction output.

However, using this approach can result in unnatural small prediction uncertainties. The next section looks at taking into consideration the uncertainties propagation from the previous iteration.

3.6.9 Prediction with uncertainties propagation

In the case of dynamical systems, the states at the next iteration are dependent on the current state, and even the previous one in some cases. It is necessary to know the current state to compute the following one. Due to this chain reaction problem, it is not possible to query the state at time t_x without computing all the states until reaching this time: it is necessary to use the GP as a multi-step ahead. However, uncertainties can be inaccurate if not taken into consideration correctly.

One approach proposed by Girard 2004 is to use iterative multi-step ahead forecasting. This method uses the history of the state for a certain time L and computes the next iteration using a different kernel which is aware of the uncertainties at the previous iteration. One drawback is the necessity to use the history of states to make this approach work. Moreover, the prediction length directly depends on the length of the state history used.

Based on the previously stated approach, Deisenroth 2010 implemented a similar framework only based on the current uncertainty of the prediction, which makes easier the computation:

$$x_{pred}(t+1), \psi(x_{pred}(t+1)) \sim \mathcal{N}(x_{pred}(t), K, \psi(x_{pred}(t))) \tag{3.60}$$

where $x_{pred}(t)$ is the input for the GP at time t, K is the kernel of the GP and $\psi(x_{pred}(t))$ is the uncertainties associated to $x_{pred}(t)$.

However, only having the uncertainty at the current iteration and not a full kernel, it is more difficult to sample a curve from the distribution respecting the standard deviation envelop computed one step at a time.

3.6.10 Single-Task Sparse GP

Even if GP is considered to be data-efficient, the complexity of the optimization of the hyper-parameters is $\mathcal{O}(N^3)$, where N is the number of sample used for training. The idea behind Sparse GP is to reduce this complexity to $\mathcal{O}(M^2N)$ by taking only M induced points that bring information to the optimization.

Originally called Spatial Gaussian Predictive Process (SGPP) by Snelson and Ghahramani 2005, this approximation has been renamed Fully Independent Training Conditional (FITC) algorithm by Quiñonero-Candela et al. 2007. The optimization is focused on the induce point location. In the FITC algorithm, the most popular inducing point method, the hyper-parameters stay constant during the optimization and the new Likelihood function to minimize is expressed as:

$$-log(p(\mathbf{y}|X,\theta)) = \frac{1}{2}log(|Q_{xx}+G|) + \frac{1}{2}\mathbf{y}^T(Q_{xx}+G)^{-1}\mathbf{y} + \frac{n}{2}log(2\pi) \tag{3.61}$$

where $Q_{xx} = K_{x\varsigma}K_{\varsigma\varsigma}^{-1}K_{\varsigma x}$ and $G = diag[K_{xx} - Q_{xx}] + \sigma_{noise}^2\mathbf{I}$. ς is the array of the inducing points of length M.

More up-to-date algorithms can be used such as Grancharova et al. 2023 and Krivec et al. 2021, however the FITC is the algorithm used in this thesis when using Sparse GPs.

3.7 Conclusion

This Chapter presents the algorithms used in the Thesis. First, the Continuous Control theory is described, followed by the Intermittent Control Theory, where additional blocks are introduced.

Reinforcement Learning Learning theory is explained, focusing on the Policy Iteration algorithm implemented by Vrabie et al. 2007. This algorithm is used to estimate the optimal state-feedback gains required by the controller. Then, Data Informativity is described. Due to its indirect nature, by estimating the systems matrices **A** and **B**, to then compute the feedback gains, the output of the algorithm can be used within the Hold as well as the state feedback block.

Finally, Gaussian Processes theory is introduced, looking at some of the kernel functions commonly used for regression purposes, as well as the hyperparameters. Two different models of GPs have been described: Single-Task and Multi-Task. The purpose of GPs in this application is to model the internal dynamics of the plant, using online data while providing information about uncertainties. Approaches to implement uncertainty propagation for multistep ahead prediciton were also described.

Chapter 4

Intermittent Data Driven Control for adaptation

The Intermittent Controller is proposed as a biological-inspired controller, presented in Section 2.2, helping in understanding the human control behavior, especially during quiet standing. As explained in Section 2.2.1, quiet human standing can be modeled as a Single Inverted Pendulum balancing task. Intermittent control is based on an underlying continuous controller implementation which requires a set of state feedback gains to stabilize the system. As explained in Section 3.2.4, the state feedback can be designed using Pole Placement and the LQR method. However, both of these approaches require knowing the system matrices $\mathbf{A}$ and $\mathbf{B}$ to ensure the stability of the controller (see Equation (3.4)).

In this section, Data-driven approaches are used to estimate the optimal state feedback gains required by the system and it is the first proposed contribution. First, Reinforcement Learning is investigated, as this direct approach estimates the state feedback gains $\mathbf{k}$ without the need to estimate the system's matrices first. Secondly, Data Informativity is used to estimate the system matrices $\mathbf{A}$ and $\mathbf{B}$, then passed to the LQR method to estimate $\mathbf{k}$. This approach can be considered as indirect. Finally, a discussion section compares the main differences between these two approaches.

4.1 Methods

In this section, the implementation of the Reinforcement Learning algorithm as well as the Data Informativity framework with the Intermittent Control framework is explained. Then, evaluation criteria are introduced to assess the output of each of the algorithms based on data generated by an intermittent controller.

4.1.1 Implementation

This section is split between two different implementations. Whilst the data used by both algorithms are coming from the same system, it is getting processed differently. In the first part, implementation details about Reinforcement Learning in the context of Intermittent Control are explained, followed by implementation details of the Data Informativity algorithm within the IC framework. As shown by Gawthrop and Wang 2009, the stability of the Intermittent Controller is ensured.

4.1.1.1 Combined IC-RL framework

In this section we introduce in the form of a diagram, the combination of the policy iteration algorithm and the current IC framework, under the actor-critic architecture. In Figure 4.1, we can see a conceptual representation of how these elements interact. From it, we can see that three blocks are part of this new architecture, the *Critic*, the block labeled as $\dot{V}(t)$, and the *Actor*, which contains a general IC and a *Design* stage. The $\dot{V}(t)$ block has the role of estimating the LQR cost associated with the current input $\mathbf{u}(t)$ (see

Section 3.4.2) by performing Equation (3.5) based on the observed states. If there is no observer then the system states $\mathbf{x}(t)$ are used. The values of the cost $\dot{V}(t)$ are then used by the *Critic* at time intervals defined by T, to estimate the values of the elements in $\mathbf{P}$ once enough information is collected.

The role of the *Actor* is to provide an optimal policy based on several updates of the matrix $\mathbf{k}$ which is obtained from $\mathbf{P}$ using Equation (3.7). The events generated by IC at t_i have the additional functionality of controlling when the updates would happen. With the current formulation, these updates happen only when an event is generated; therefore, if the *Critic* has already computed a new value of $\mathbf{P}$, the re-design will only be triggered when the next event happens. One way to ensure that a new value is considered as soon as possible is to force an event as soon as a new estimate of $\mathbf{P}$ has been found.

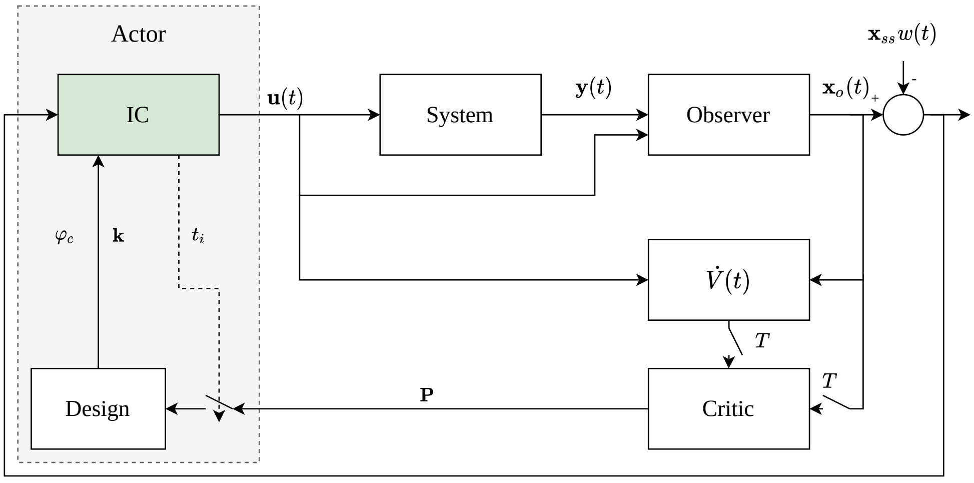

Figure 4.1: The Agent-Critic framework combined with IC and a policy iteration method. The quantities $\mathbf{y}(t)$, $\mathbf{u}(t)$, $w(t)$ represent the output, input, and set-point respectively. The product $x_{ss}w(t)$ is the vector version of the set-point $w(t)$ and the observed states are defined by $\mathbf{x}_o$. The block labeled as $\dot{V}(t)$ has the role of estimating the LQR cost associated with the current input $\mathbf{u}(t)$ by performing $\dot{V}(t) = \mathbf{x}_o^T(t)\mathbf{Q}\mathbf{x}_o(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)$ based on the observed states. If there is no observer then the system states $\mathbf{x}(t)$ are used. The Critic then computes the new values of matrix $\mathbf{P}$ once a sufficient number of samples have been collected at time intervals defined by T. This information is then passed to a re-designed stage that computes the new feedback gain $\mathbf{k}$, which is part of the parameters that the IC needs to operate. IC uses its internal triggering mechanism to start a redesign only when an event is generated at t_i. The grey box represents the elements that are part of the Actor. The black dashed lines represent quantities that are defined only at event times.

The *Design* block has the role of re-computing all the relevant quantities for IC to operate properly. This involves the computation of a new *hold* that is based on the updated **k**. The set of parameters and quantities that are sent to IC as a consequence of a re-design is shown as $\boldsymbol{\varphi}_c$, which includes **k**.

There are some limitations of the stated approach regarding the system and the method. It is necessary to get access to the full states without the need of an observer. In addition, as previously stated, it is necessary to start with a stable controller to gather points for the first estimation.

4.1.1.2 Combined IC-Data Informativity framework

In this section, the adaptation of the Data Informativity to fit within the Intermittent Control is explained. One of the main differences to a continuous controller is the presence of events with the Intermittent Controller. The term occlusion is introduced, which has for purpose to *hide* the data coming from the plant following a triggering event when passed to the algorithm. Figure 4.2 shows graphically how this is implemented.

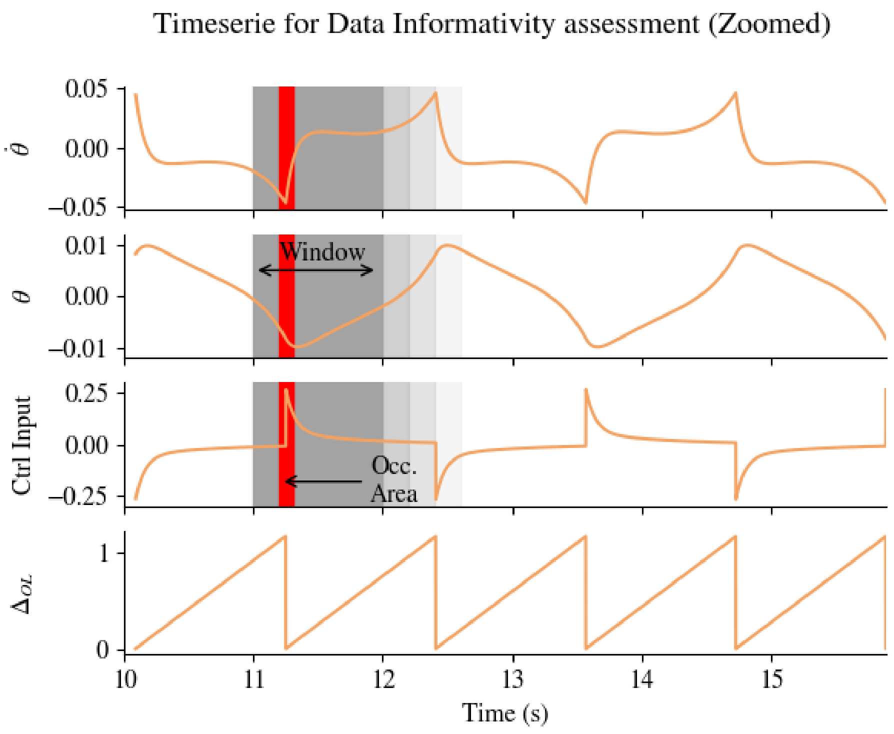

Figure 4.2: Concept of occlusion for Data Informativity. The data is coming from an Intermittent Controller applied to a Single Inverted Pendulum. The grey area represents the window of data passed to the algorithm. The red box shows the occluded data.

Figure 4.2 shows the response of an Intermittent Controller applied to the Single Inverted Pendulum. This figure shows two highlighted areas: the grey one represents the window size of the data passed to the algorithm and the red area shows the data that is occluded, hence not used for the estimation of **A** and **B**. The occluded area is applied to a single time step before the event and can be any length following the event.

Following on the concept of event, and open-loop trajectories within Intermittent Control, it is important to time the use of any algorithm output with the triggering time. The diagram in Figure 4.3 visualizes when the output of the estimation is getting used by the Intermittent Controller. Assuming $t_{DI} = n$, where t_{DI} is the set time to perform the Data

Informativity computation:

$$\begin{cases} \text{True, if } t > t_{DI} \text{ and } trig_0 \\ \text{False, otherwise} \end{cases} \tag{4.1}$$

It is necessary to wait for an event in the Intermittent Controller in addition to the time $t > t_{DI}$. Hence, the Data Informativity is not performed evenly due to the fact that it needs to wait for the next triggering. This can be represented in Figure 4.3.

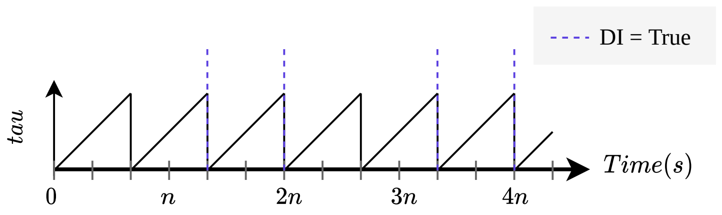

Figure 4.3: Timing of Data Informativity with Intermittent Framework. The data displayed is representing the open-loop interval. Dashed vertical line represents when DI is used. The Data Informativity is running every time $t > n$ and $trig_0$ is true. This introduces an unevenly distributed time between each algorithm run.

4.1.2 Evaluation criteria

Only computer simulations are used as analytical tools which helps with time-varying system experiements. Multiple measures are computed during the simulation to assess the controller's performance. Five measures are computed in the time domain, using simulation data. First, the Root Mean Square Error (RMSE) is computed for each state ($\dot{\theta}$ and θ for the pendulum) between the measured states and the reference (zero in the regulation case).

$$\dot{\theta}_{RMSE}(\dot{\theta}, w_{\dot{\theta}}) = \sqrt{\frac{\sum_{i=0}^{N-1}(\dot{\theta}_i - w_{\dot{\theta}_i})^2}{N}} \tag{4.2}$$

$$\theta_{RMSE}(\theta, w_{\theta}) = \sqrt{\frac{\sum_{i=0}^{N-1}(\theta_i - w_{\theta_i})^2}{N}} \tag{4.3}$$

The mean of the control input and the mean of the open-loop interval are also computed. This is to assess if the control input is correctly balanced and to measure if the open-loop is close to the SMH response, and not triggering at the minimum open-loop interval. In the IC framework, the open-loop interval between each trigger can be used as a proxy to assess how well the Hold matches with the plant dynamics. If the open-loop interval Δ_{ol} is long, then the match between the output states from the hold closely matches the system's states. On the other hand, if Δ_{ol} is short, the states generated by the hold do not represent the closed-loop system's dynamics correctly (see Figure 4.15) even though the θ_{RMSE} is low.

$$u_{MEAN} = \frac{1}{N}\sum_{t=1}^{N} u(t) \tag{4.4}$$

$$\Delta_{olMEAN} = \frac{1}{N}\sum_{t=1}^{N} \Delta_{ol}(t) \tag{4.5}$$

One additional measure computed is the variability of the control input. This is done using the cross-correlation of the signal between two consecutive open-loop intervals.

$$(a * v)_n = \sum_{m=-\infty}^{\infty} a_m v_{n-m} \tag{4.6}$$

where a is the control signal during one open-loop interval and v is the control signal during the following one. Each signal is normalized using

$$a_{norm} = \frac{a - \overline{a}}{\sigma_a - N} \tag{4.7}$$

where $\overline{a}$ is the mean value of the array a, σ_a is the standard deviation of a and N is the length of the array.

Then the variability is computed as:

$$\Psi = -\max(|(a * v)_n|) + 1 \tag{4.8}$$

By normalizing both arrays a and v, Ψ is bounded between 0 and 1, where 0 represents no variability and 1 shows high variability. Figure 4.4 shows the output of the cross-correlation between two open-loops intervals.

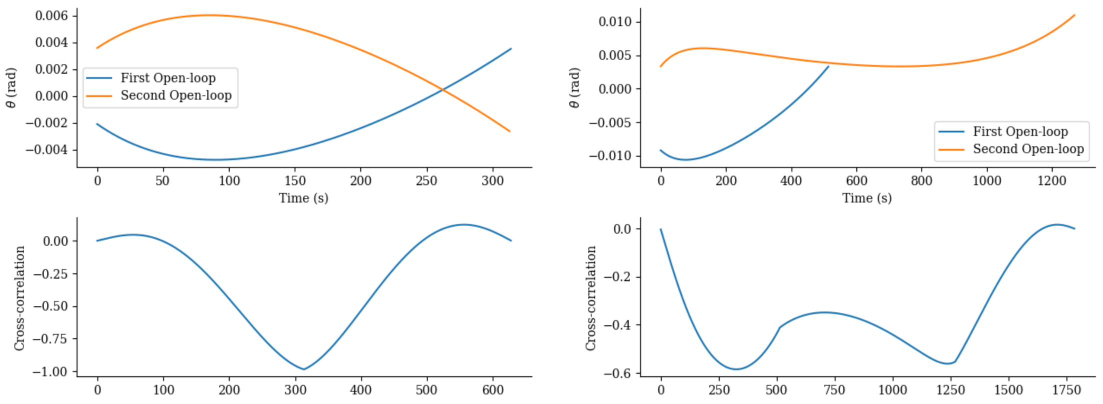

Figure 4.4: Example of cross-correlation applied to consecutive open-loop interval. The top row is showing the angle timeserie coming from the simulation for the current and previous open-loop interval. The bottom row is showing the output of the cross-correlation function. (a) is showing high correlation between the two signal ($\Psi = 0$) and (b) is showing a low correlation ($\Psi = 0.4$)

The output of Reinforcement Learning and Data Informativity are different. Reinforcement Learning is outputting the state feedback gains **k** directly whereas the Data Informativity framework is outputting the system matrices **A** and **B**. It is needed to compared both approach to the optimal value differently. In the case of RL, the error between the optimal gains and the estimated one is investigated. Similarly, when using the DI algorithm, the error is computed between each element of the system matrices **A** and **B**.

In the case of the Single Inverted Pendulum, the system matrices **A** and **B** as well as the state feedback **k** can be represented as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix} \qquad \mathbf{k} = \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \end{bmatrix}$$

Similarly, this can be expanded to the cartpole system as follow:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \mathbf{A}_{1,4} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} & \mathbf{A}_{2,4} \\ \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & \mathbf{A}_{3,4} \\ \mathbf{A}_{4,1} & \mathbf{A}_{4,2} & \mathbf{A}_{4,3} & \mathbf{A}_{4,4} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ \mathbf{B}_4 \end{bmatrix} \qquad \mathbf{k} = \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \mathbf{k}_3 \\ \mathbf{k}_4 \end{bmatrix}$$

4.2 Results

In the following section, Intermittent Control is applied to the Single Inverted Pendulum (see Appendix A). The goal of each algorithm is to estimate correctly the optimal state feedback of the system. First, Reinforcement Learning is assessed followed by Data Informativity. The Continuous Controller is compared with the Intermittent Controller using SMH.

4.2.1 Reinforcement Learning

As previously mentioned, Reinforcement Learning requires an initial stable controller. Figure 4.5 shows the location of the initial poles in relationship to the optimal poles. The optimal poles are located at $[-6.83, -2, 92]$ when using the LQR method with $Q = I$ and $R = 1$. Four sets of initial poles are used for all following RL-based simulations: $[-8.1, -8]$, $[-4.1, -4]$, $[-3.1, -3]$, $[-2.1, -2]$. Two initial sets of poles are located in-between the optimal one, one set on the left of the optimal one, making the initial controller faster, and one set closer to the imaginary axis, making the controller slower.

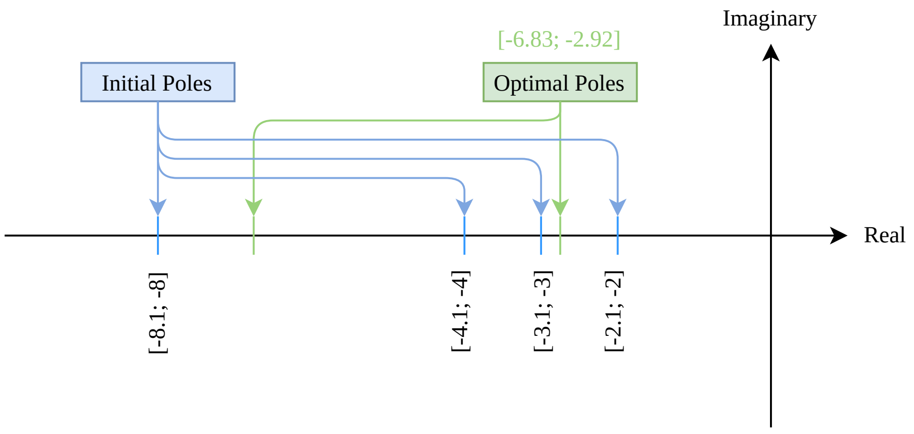

Figure 4.5: Diagram showing the location of the initial poles compared to the optimal poles applied to the SIP system.

4.2.1.1 Continuous Control

Two of the initial poles are located in between the optimal poles and the two other initial poles are located on either side of the optimal ones. These four conditions are tested using a Continuous Controller first, as this algorithm has been designed to work with this specific type of controller. Figure 4.6 shows the results of this algorithm.

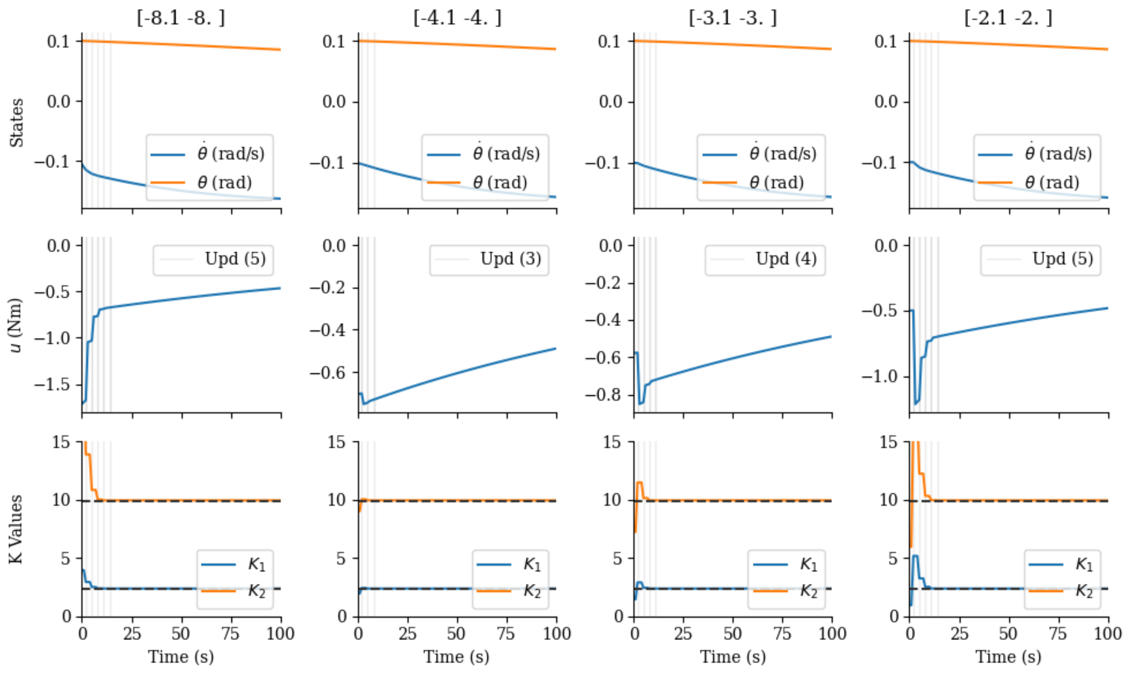

Figure 4.6: Reinforcement Learning applied with a Continuous Controller on the Single Inverted Pendulum. Each column is starting with a different initial pole conditions. From left to right: $[-8.1, -8]$, $[-4.1, -4]$, $[-3.1, -3]$, $[-2.1, -2]$. Top row is the states of the SIP. Second row is the control input. The last row is the estimation of $\mathbf{k}$. The horizontal black dashed line is the optimal value for $\mathbf{k}$. Each vertical grey line representation a new $\mathbf{k}$ estimation.

As presented in Figure 4.6, the Reinforcement Learning algorithm can converge to the optimal poles, meaning the optimal gains for $\mathbf{k}$ for all initial sets of poles. For the four initial poles, it takes between 3 and 5 iterations for the algorithm to converge to the optimal value, which can be achieved below 15ms with $dt = 1$ms.

4.2.1.2 Intermittent Control

Knowing that Continuous Control can be used with this Reinforcement Learning framework, Intermittent Control is now used to assess its capabilities to converge to the optimal **k** values. The following figures are assessing the convergence of the state feedback gain under different sets of minimum open-loop intervals and thresholds within IC. In the first simulation, the Intermittent Controller is operating in clock-driven mode, explained in Section 3.3.1, and the minimum open-loop interval is set to 3ms. These parameters are used to replicate as closely as possible a continuous controller with an intermittent one.

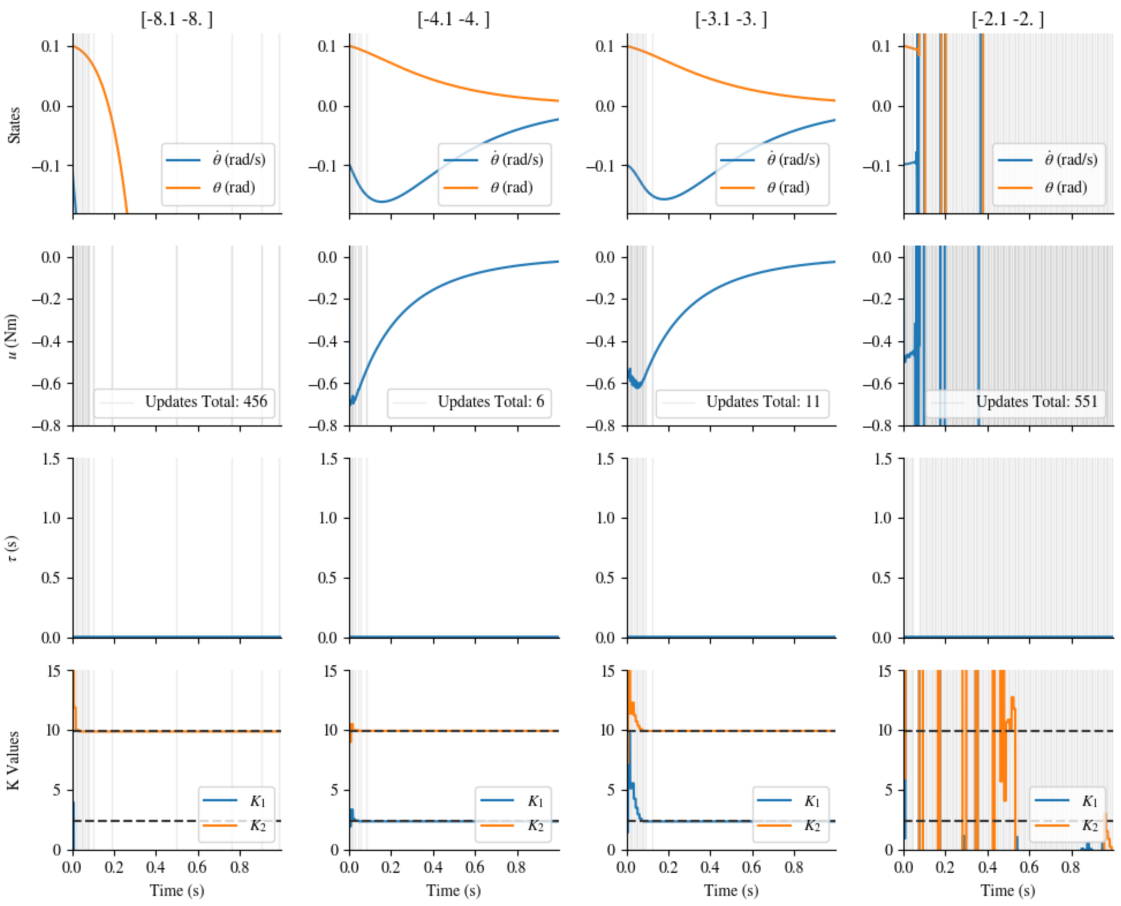

Figure 4.7: Reinforcement Learning applied to Intermittent Control on Single Inverted Pendulum. Clock-driven mode with $\Delta_{ol} = 3$ms. Each column is starting with a different initial pole conditions. From left to right: $[-8.1, -8]$, $[-4.1, -4]$, $[-3.1, -3]$, $[-2.1, -2]$. Top row is the states of the SIP. Second row is the control input. Third row is the open-loop interval, ans last row is the estimation of **k**. The horizontal black dashed line is the optimal value for **k**. each vertical grey line is a **k** estimation.

As presented in Figure 4.7, the state feedback can converge toward the optimal value when the initial poles are located in between the optimal poles. It is taking 6 and 11 iterations respectively, whilst only 4 and 5 iterations are required by the Continuous Controller. In addition, when initial poles are located outside the optimal ones, the algorithm is not able to converge, and this brings instability in the pendulum response. When the initial poles are $[-8.1, -8]$, only $\mathbf{k}_2$ is converging to the correct target. In the following simulations, Intermittent Control is kept in clock-driven mode, however, the minimum open-loop interval is increased to 50ms.

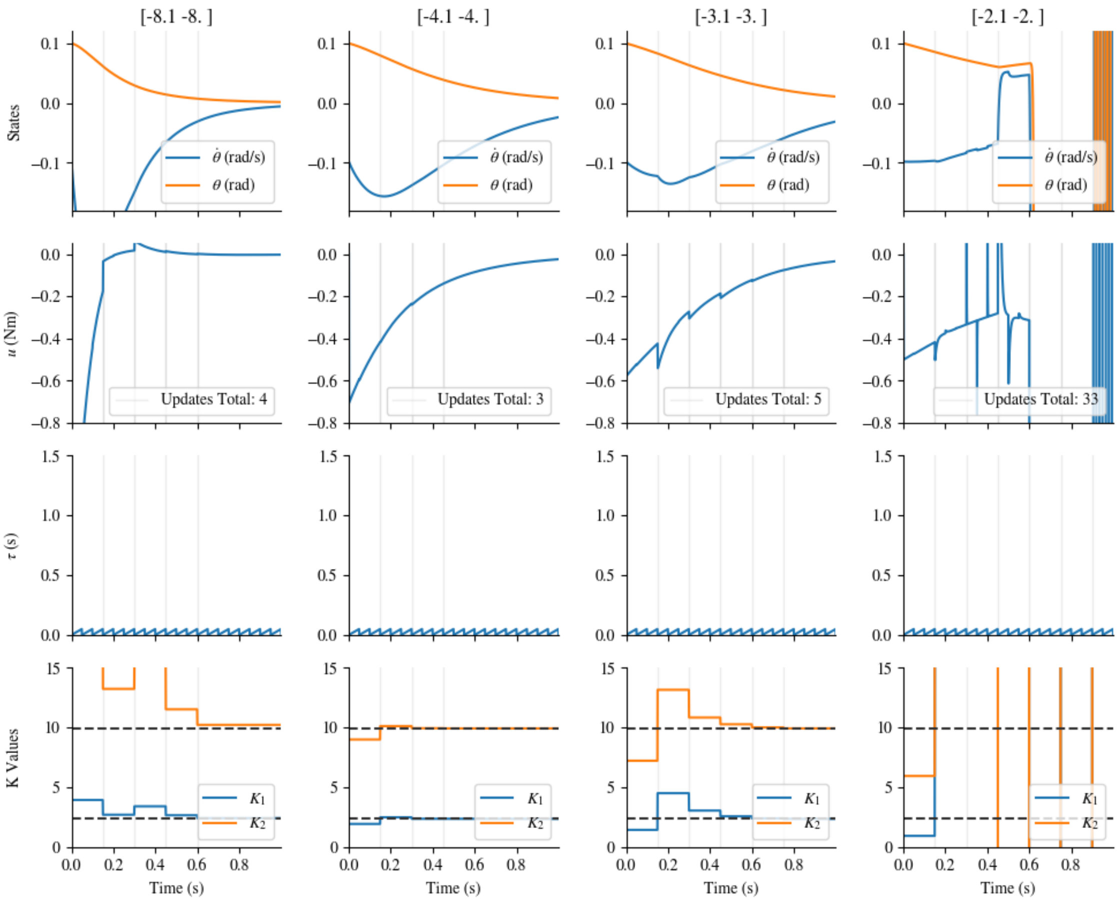

Figure 4.8: Reinforcement Learning applied to Intermittent Control on Single Inverted Pendulum. Clock-driven mode with $\Delta_{ol} = 50$ms. Each column is starting with a different initial pole conditions. From left to right: $[-8.1, -8]$, $[-4.1, -4]$, $[-3.1, -3]$, $[-2.1, -2]$. Top row is the states of the SIP. Second row is the control input. Third row is the open-loop interval, ans last row is the estimation of $\mathbf{k}$. The horizontal black dashed line is the optimal value for $\mathbf{k}$. each vertical grey line is a $\mathbf{k}$ estimation.

In comparison to Figure 4.7, when the minimum open-loop interval is increased, 3 out of 4 initial conditions can converge towards the optimal gains (see Figure 4.8). As the minimum open loop is larger than before, it is now taking longer for the RL algorithm to get data, due to the increased in time between each event. However, it is taking fewer iterations to converge toward the optimal gains: $[-8.1, -8]$: 4 iterations, $[-4.1, -4]$: 3 iterations and $[-3.1, -3]$: 5 iterations. This is now in a similar range as the Continuous Controller. However, it is worth mentioning the time for IC to get the amount of sample needed by the algorithm, which can be too long in the case of a fast unstable system.

In the results presented above, the Intermittent Controller is operating in clock-driven mode. This is forcing a triggering from the controller at a fixed interval. This approach is not monitoring the states of the plant as a triggering mechanism. Next the Intermittent Controller is operating in event-driven mode, which allows different lengths of open-loop interval by comparing the states predicted by the Hold and the actual states from the plant. When operating in event-driven mode, the length of the open-loop interval can be used as a proxy for the match between the hold and the plant. The threshold is now set to $1e^{-4}$ rad and even though the threshold is very small, it is enough for the Intermittent Controller to not trigger at the minimum open-loop interval once the optimal gain is found. Results can be seen in Figure 4.9.

The minimum open-loop is set to 3ms, to allow fast triggering when the gains used by the controller are not optimal. Focusing on the two middle simulations, where initial poles are $[-4.1, -4]$ and $[-3.1, -3]$ respectively, the Intermittent Controller can increase its open-loop interval to around 1.3 seconds once the optimal gain is used for control.

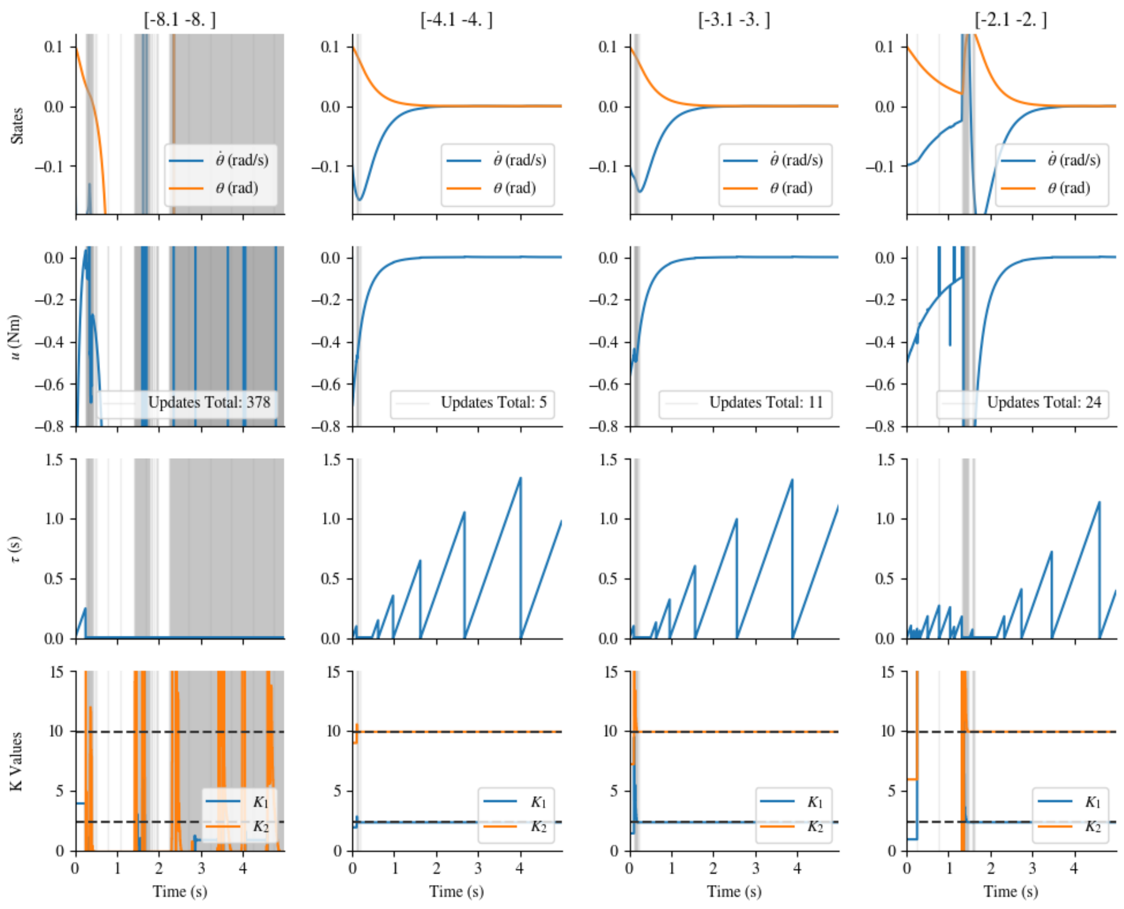

Figure 4.9: Reinforcement Learning applied to Intermittent Control on Single Inverted Pendulum. Event-driven mode with $\Delta_{ol} = 3$ms and Threshold $= 1e^{-4}$. Each column is starting with a different initial pole conditions. From left to right: $[-8.1, -8]$, $[-4.1, -4]$, $[-3.1, -3]$, $[-2.1, -2]$. Top row is the states of the SIP. Second row is the control input. Third row is the open-loop interval, ans last row is the estimation of $\mathbf{k}$. The horizontal black dashed line is the optimal value for $\mathbf{k}$. each vertical grey line is a $\mathbf{k}$ estimation.

In contrast to Figure 4.7, in addition to the initial poles located in-between the optimal one, the simulation where the initial poles are $[-2.1, -2]$ is stable. However, the optimation started converging toward a non-optimal set of gains, which forced the Intermittent Controller to trigger at the minimum open-loop interval for some time. This allowed the algorithm to estimate gains faster and finally converge to the optimal one. Similarly to the clock-driven case, the minimum open-loop intervals are increased to 50ms to assess the reliability of this algorithm with Intermittent Control.

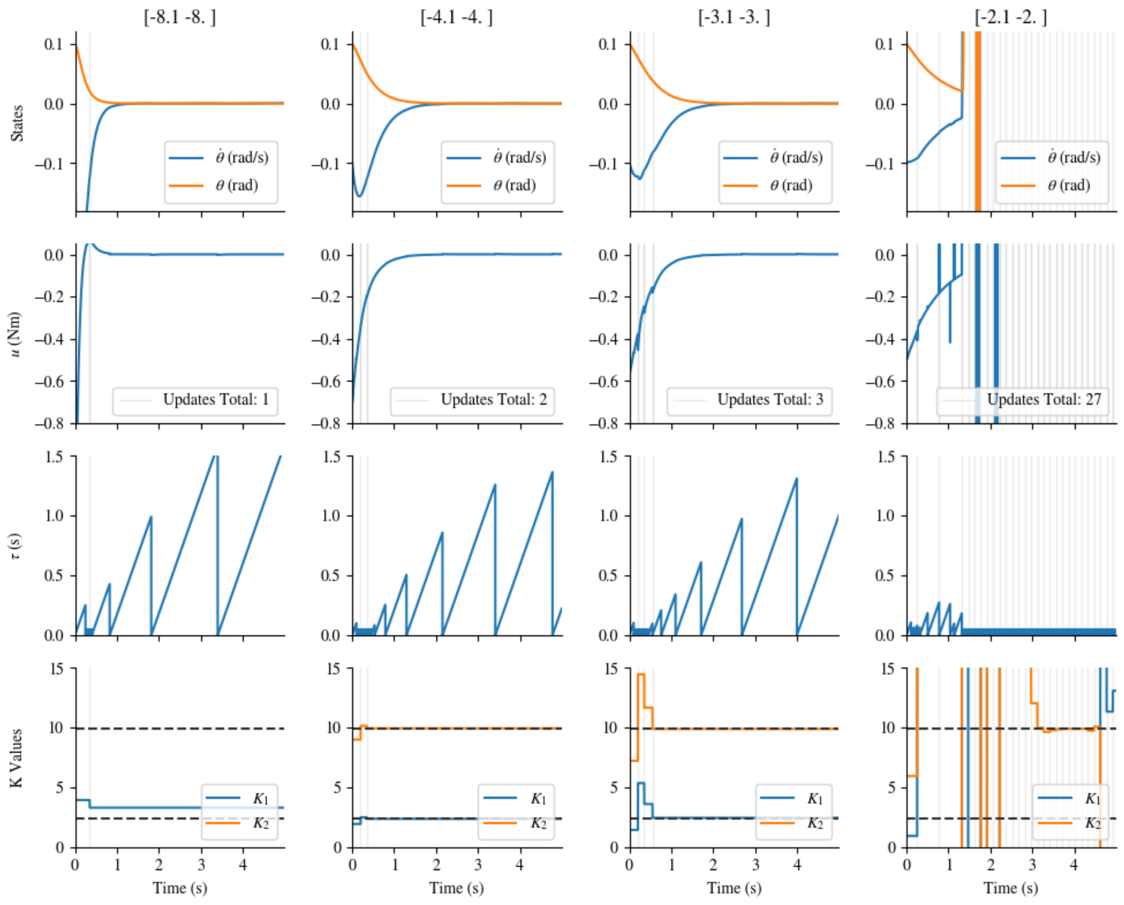

Figure 4.10: Reinforcement Learning applied to Intermittent Control on Single Inverted Pendulum. Event-driven mode with $\Delta_{ol} = 50$ms and Threshold $= 1e^{-4}$. Each column is starting with a different initial pole conditions. From left to right: $[-8.1, -8]$, $[-4.1, -4]$, $[-3.1, -3]$, $[-2.1, -2]$. Top row is the states of the SIP. Second row is the control input. Third row is the open-loop interval, ans last row is the estimation of $\mathbf{k}$. The horizontal black dashed line is the optimal value for $\mathbf{k}$. each vertical grey line is a $\mathbf{k}$ estimation.

Figure 4.10 is based on an Intermittent Controller operating in event-driven mode and using a minimum open-loop interval of 50ms. Similarly to all cases presented before, the initial poles located in between the optimal ones converge to the optimal gains. However, focusing on the initial poles outside the optimal one has changed compared to when the minimum open-loop interval is set to 3ms. When the initial poles are $[-8.1, -8]$, even though the simulation is stable, $\mathbf{k}_2$ is not converging the optimal value. Looking closely at the case where the initial poles are $[-2.1, -2]$ between a minimum open-loop of 3ms (Figure 4.9) and 50ms (Figure 4.10), the plant is getting into an unstable regime. This is due to the minimum open-loop being larger, which limits the Intermittent Controller to trigger fast enough to reach the minimum of 3 samples required by the algorithm.

4.2.2 Data Informativity

As presented in the previous section, Reinforcement Learning has been used to estimate the optimal state feedback for the plant. However, there are two drawbacks to using this algorithm. First, it is required to start with a stable controller. Even though this controller is stable, it might not converge to the optimal gain depending on the IC parameters used. Secondly, this approach is only estimating $\mathbf{k}$ without computing the system matrices $\mathbf{A}$ and $\mathbf{B}$. Hence, this framework cannot be used to redesign the Hold block in the case of adaptation as it is still necessary to get these matrices to design the controller.

Focusing on Intermittent Control, Martín 2018 used the Kalman filter to handle adaptation cases. In this section, Data Informativity is used to perform the system identification. First, this framework is applied to a Continuous Controller, then it is applied to an Intermittent Controller.

4.2.2.1 Continuous Control

In this section, the Data Informativity framework is applied within a Continuous Controller to evaluate its capability without the influence of intermittency present in the Intermittent Controller. The simulation is based on the Single Inverted Pendulum (see Appendix A).

One issue with Continuous Control is the lack of persistence of excitation, hence the Data Informativity framework is not able to solve the least square optimisation problem due to the rank condition not being met (Equation (3.39)). To overcome this, control input noise has been added on top of the state feedback input. This is to ensure that the pendulum is in constant motion throughout the simulation.

As shown in Figure 4.11, Data Informativity is used to estimate the system matrices. The coefficients $A_{1,2}$, B_1, K_1 and K_2 are not being estimated accurately by the algorithm. One important thing to notice is the flip sign of the coefficient $A_{1,2}$ (from positive to negative). This change in sign could bring instability if the output of the Data Informativity was used to redesign the controller. In addition, the control input is always updated due to the noise being propagated back to the controller via the system.

To keep some excitation of the system without the use of noise, it has been decided to move from normally distributed noise to a multisine signal. This allows us to specify a predefined range of frequency, a phase delay as well as an amplitude for each sinusoidal signal. By using this type of signal, it is possible to decompose it by each excited frequency, which can be useful when doing frequency analysis.

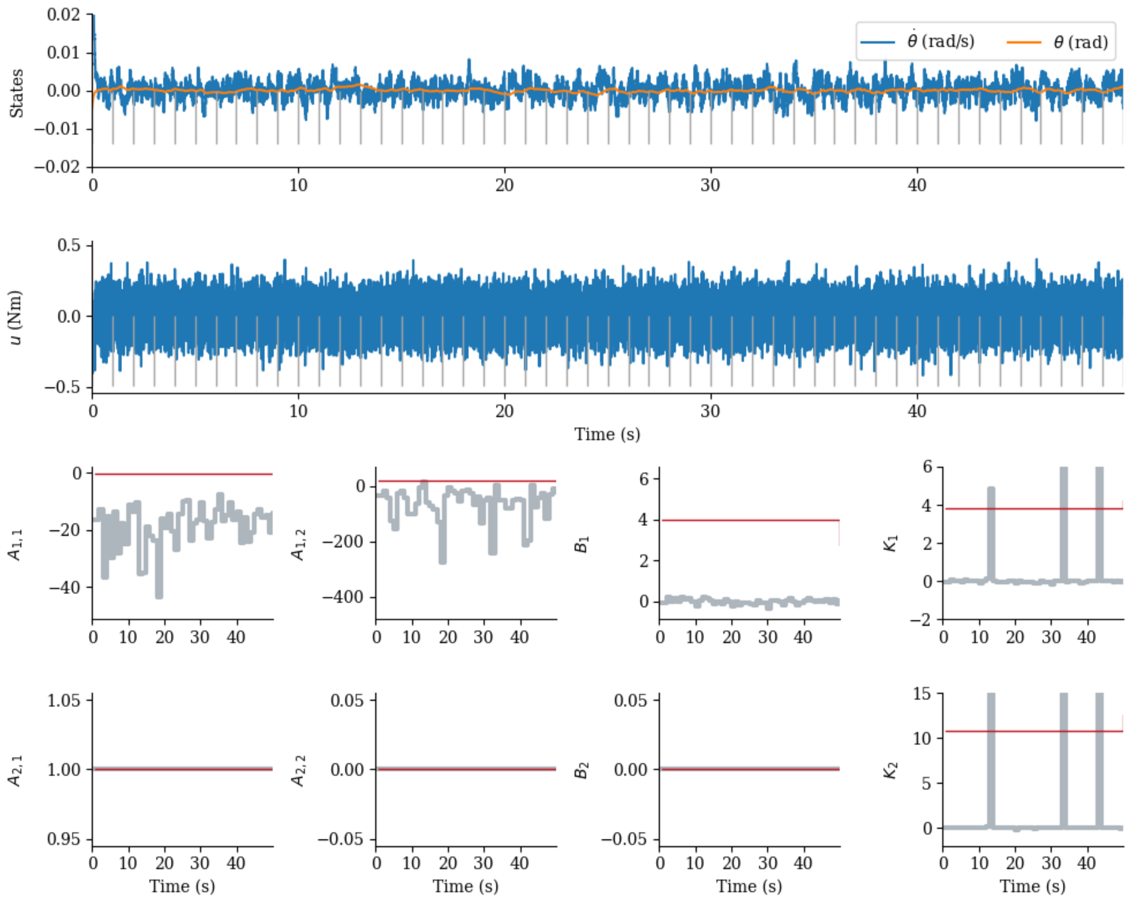

Figure 4.11: Data Informativity applied to noisy Continuous Controller. The Top row are the two states of the Single Inverted Pendulum ($\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$). The second row is the control input with noise generated by a state feedback controller. Row 3 and 4 plot the true value (red) and the estimated one (grey) using the Data Informativity Framework. Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2.

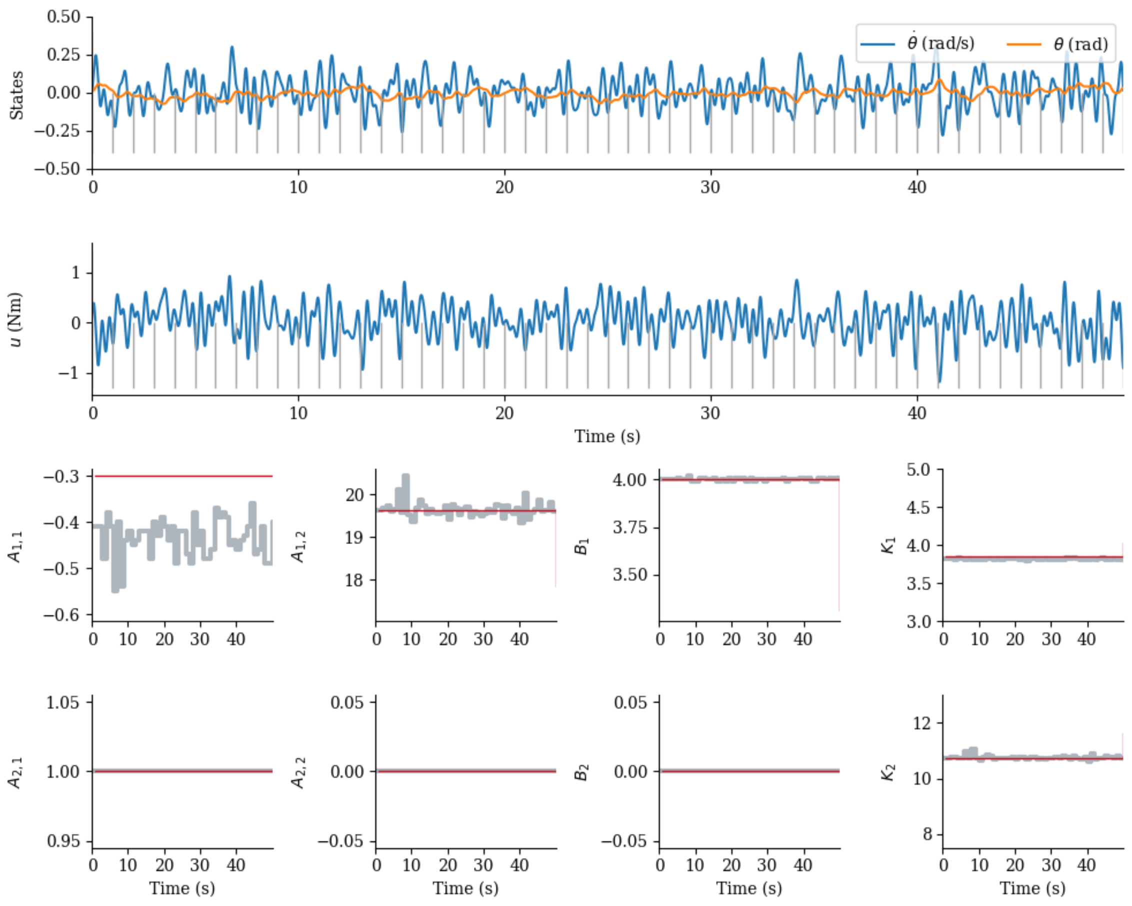

Figure 4.12: Data Informativity applied to Continuous Controller with multisine disturbance. The Top row are the two states of the Single Inverted Pendulum ($\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$). The second row is the control input with noise generated by a state feedback controller. Row 3 and 4 plot the true value (red) and the estimated one (grey) using the Data Informativity Framework. Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2.

In Figure 4.12, moving from a noisy disturbance to a multisine disturbance is helping the Data Informativity algorithm to estimate more accurately the underlying system. However, this approach requires applying a disturbance to the system to reach enough excitation and get an accurate estimation.

This technique has been also applied in the case of time-varying system, where the pendulum length changes through time. This change of length is directly affecting the value of the matrices **A** and **B**, hence modifying the optimal state feedback k coming from the LQR method.

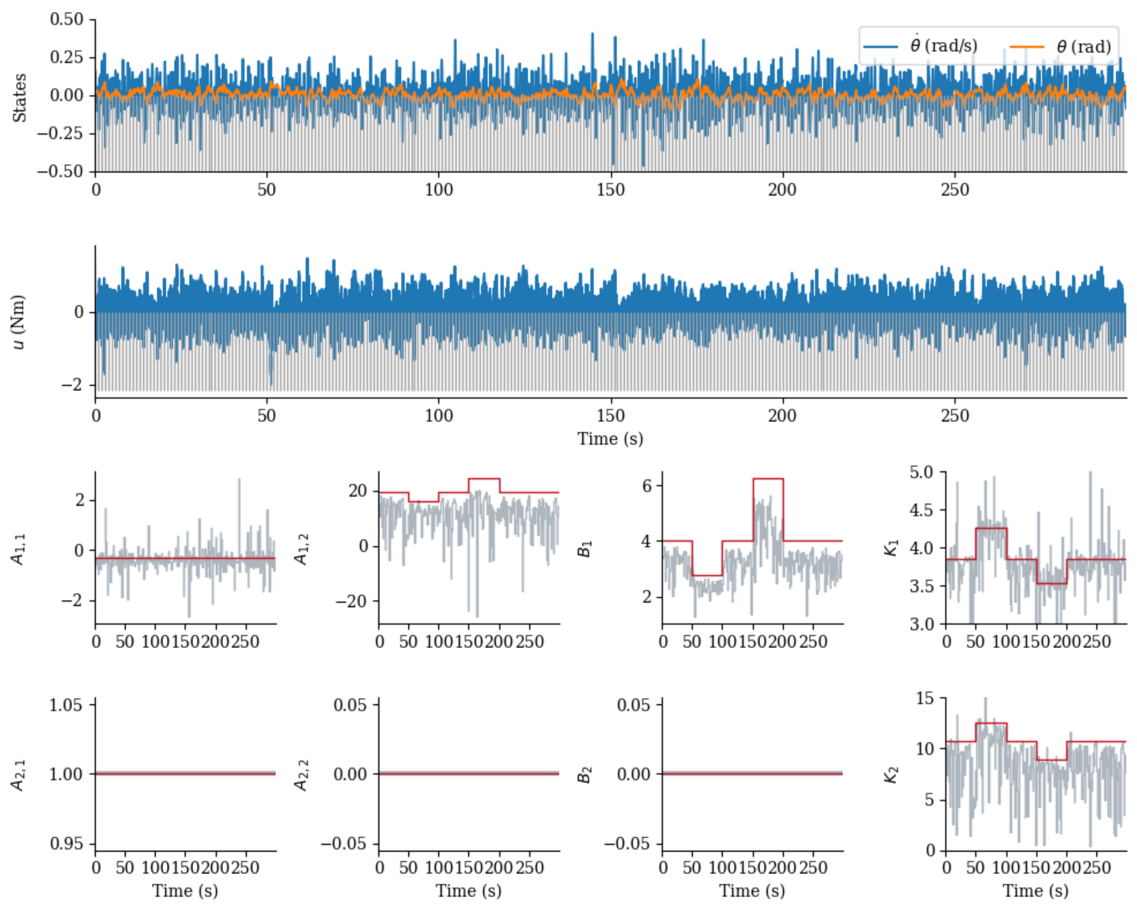

Figure 4.13: Data Informativity applied to Continuous Controller with multisine disturbance. The Top row are the two states of the Single Inverted Pendulum ($\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$). The second row is the control input with noise generated by a state feedback controller. Row 3 and 4 plot the true value (red) and the estimated one (grey) using the Data Informativity Framework. Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2.

As presented in Figure 4.13, the length of the pendulum changes at 50, 100, 150, and 200 seconds, moving from 1 meter to 1.2, back to 1 then dropping to 0.8 to finally be back at 1 meter. Loram et al. 2006 shows the effect of the length of the pendulum on the time constant of the system. They show the intrinsic neuromuscular limitation from the frequency response of the human controller. This simulation helps understand the limit of the algorithm when the system is getting easier or harder to control (increase or decrease, respectively, of the length of the pendulum). The output of the Data Informativity algorithm is quite noisy due to the speed at which the algorithm is used. The estimation is close to the actual value and can be used to capture change in the system. However, it is necessary to add a disturbance signal to ensure persistence of excitation.

4.2.2.2 Intermittent Control

As presented in the section above, the Continuous Controller needs to have an external disturbance signal to keep persistent excitation. In this section, Data Informativity is applied with data coming from an Intermittent Controller to estimate the system matrices **A** and **B**. When system is considered as a time-varying system, adaptive control can be use to ensure the match between the controller and the system. However, adaptive control can be quite complicated when applied to an unstable system. As stated in Section 2.2, Intermittent Control is used to replicate human control behavior, and the combination of open and closed-loop helps. Figure 4.14 highlights the corresponding block in the IC framework which is updated when both matrices are estimated.

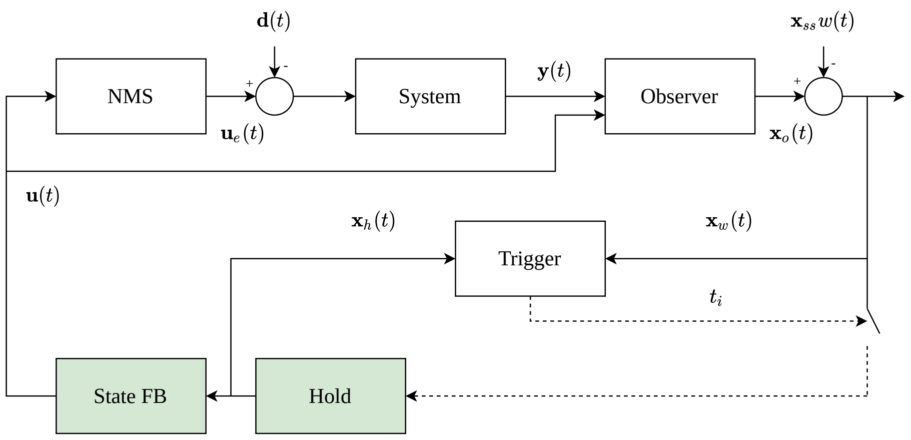

Figure 4.14: Intermittent Control block diagram. Block of the controller updated by the Data Informativity algorithm are highlighted in green.

To understand the impact of a badly designed Hold block in the Intermittent Controller compared to the impact of wrong state feedback, the following set of simulations have been run and presented in Figure 4.15. Each column represents (a) Hold and state feedback correct, (b) Hold correct, state feedback wrong and (c) Hold wrong and state feedback accurate. During the simulation, the pendulum length starting at 1m is increased to 1.2m at 50 seconds and then reduced back to 1m at 100 seconds before going down to 0.8 at 150 seconds. It is then reset to 1m at 200 seconds.

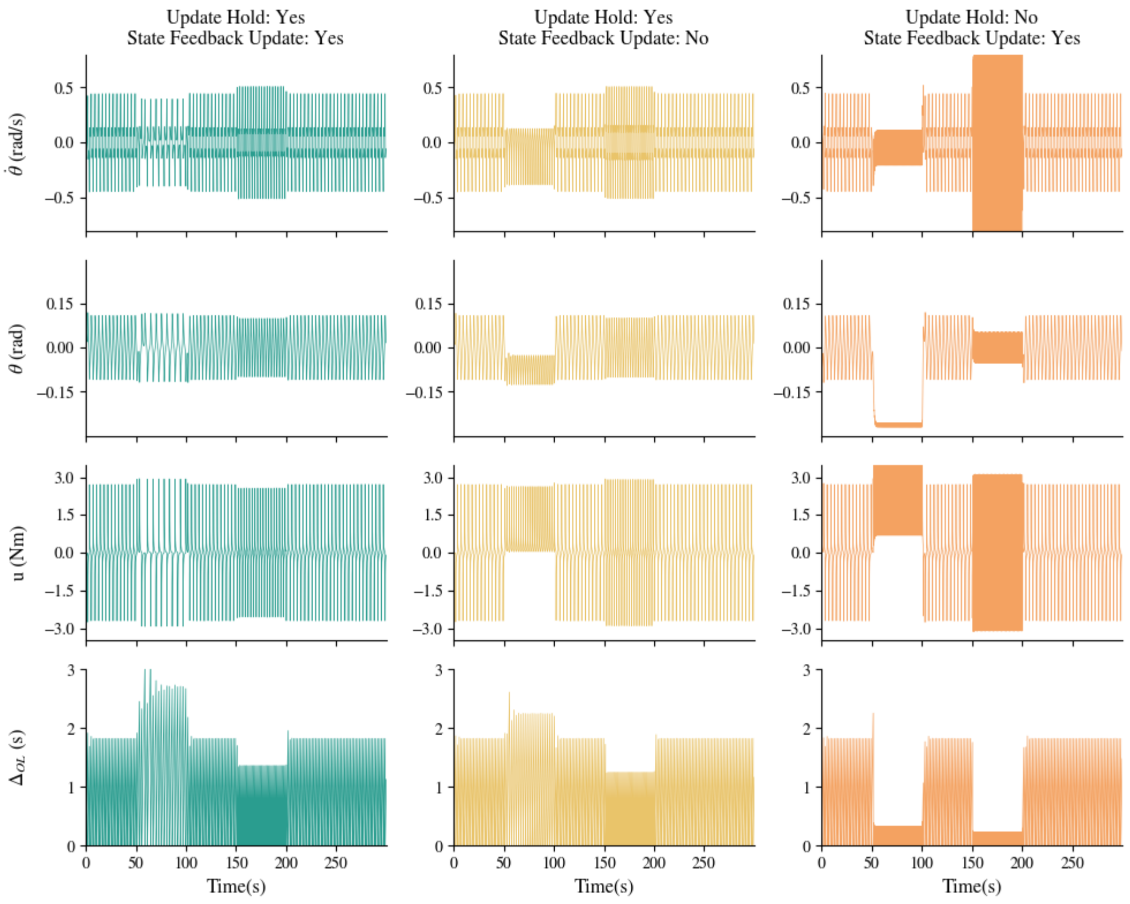

Figure 4.15: Assessment of the impact of having mismatch between Plant and IC. Pendulum is changing length at 50, 100, 150 and 200 seconds to create mismatch between plant and the controller. Each column represents: Update of the Hold and state feedback gains, update of the hold only and update of the state feedback gains only. Top row is the velocity of the pendulum, second row is the angle of the pendulum. Third row is the control input coming from the intermittent controller. Last row is the open loop interval.

As presented in Figure 4.15, having either the hold or the state feedback wrong impacts the overall performance of the intermittent controller. On the left column, the hold and state feedback are applied at the correct time when the system changes. In the middle column, the state feedback keeps its original value, designed for a 1-meter pendulum whilst the pendulum length changes every 50 seconds. The hold is getting updated to match the plant. In the last column, the hold keeps its original design, using the 1-meter pendulum, and only the state feedback is updated to match the pendulum's length.

Looking at the bottom row, the open-loop interval shows the impact of having wrong matrices. In the case of an accurate hold only, the open-loop is quite matching with the right column. However, the control input is getting impacted when the pendulum length is increased to 1.2 meters. This is then impacting the system's states that are balancing the pendulum unevenly. Focusing on the last column, where the hold is not getting updated to match the plant, the overall impact is more noticeable compared to the previous case. On top of a faster triggering when the hold does not match the system, the angle of the pendulum also does not match the reference. The pendulum is now balancing around 0.25 rad. When the pendulum length is decreased to 0.8m, the open-loop interval is also significantly decreased compared to the case when the controller is fully designed on the plant. It is worth mentioning that the controller is still making the system stable in this case.

One significant difference between the control signal from the Intermittent Controller compared to the Continuous Controller is the presence of events, resulting in some discontinuity in the control input. To account for this discontinuity, occlusion is used (see Section 4.1.1.2) and Figure 4.16 summarizes the impact of the window length of the data used for identification and the occlusion window.

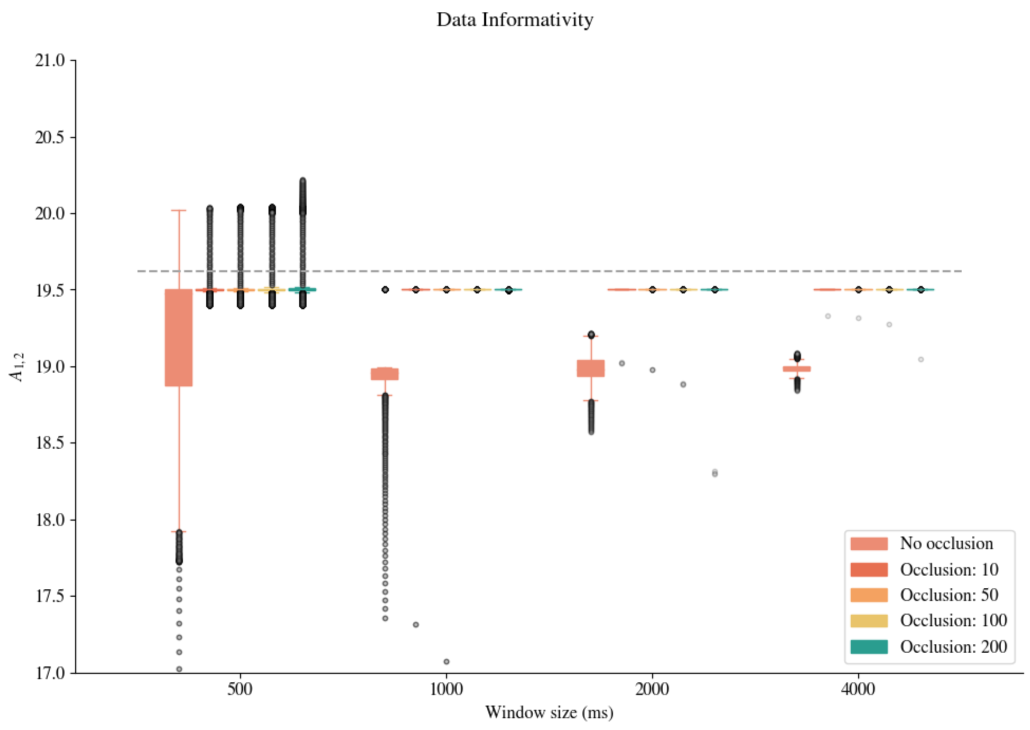

Figure 4.16: Estimation of the coefficient $A_{1,2}$ from the matrix A in the state space representation of the Single Inverted Pendulum (SIP). The x-axis represents the window size in ms, and the y-axis represents the value estimated by the Data Informativity framework. Each window size is having different occlusion window size: None, 10, 50, 100 and 200ms (applied after the event in Intermittent Framework).

Multiple observations can be made from Figure 4.16. This figure is focusing on the coefficient $A_{1,2}$ from the matrix A. This is due to this coefficient being more impacted when the pendulum length is updated. Without using the occlusion technique (light orange), increasing the window size, thus having more data for the estimations, is narrowing the spread of the $A_{1,2}$ coefficient. However, the median value of multiple estimations is diverging from the nominal true value. By adding the occlusion window, the offset of the median value is reduced and is getting closer to the true value. In addition, the spread of multiple estimations is reduced. It is worth noticing that the estimation is not improved by increasing the occlusion window. Based on these results, each following simulation will use an occlusion window of 10ms except where stated differently.

Similarly to the work performed with the Continuous Controller in section 4.2.2.1, Data Informativity is applied using data generated by the controller itself, hence coming from the Intermittent Controller in this case. In this first simulation presented in Figure 4.17, the DI algorithm is run every $t_{DI} = 1sec$. Similarly to the Continuous Controller case, the estimation generated by the DI framework is not used to update the controller online.

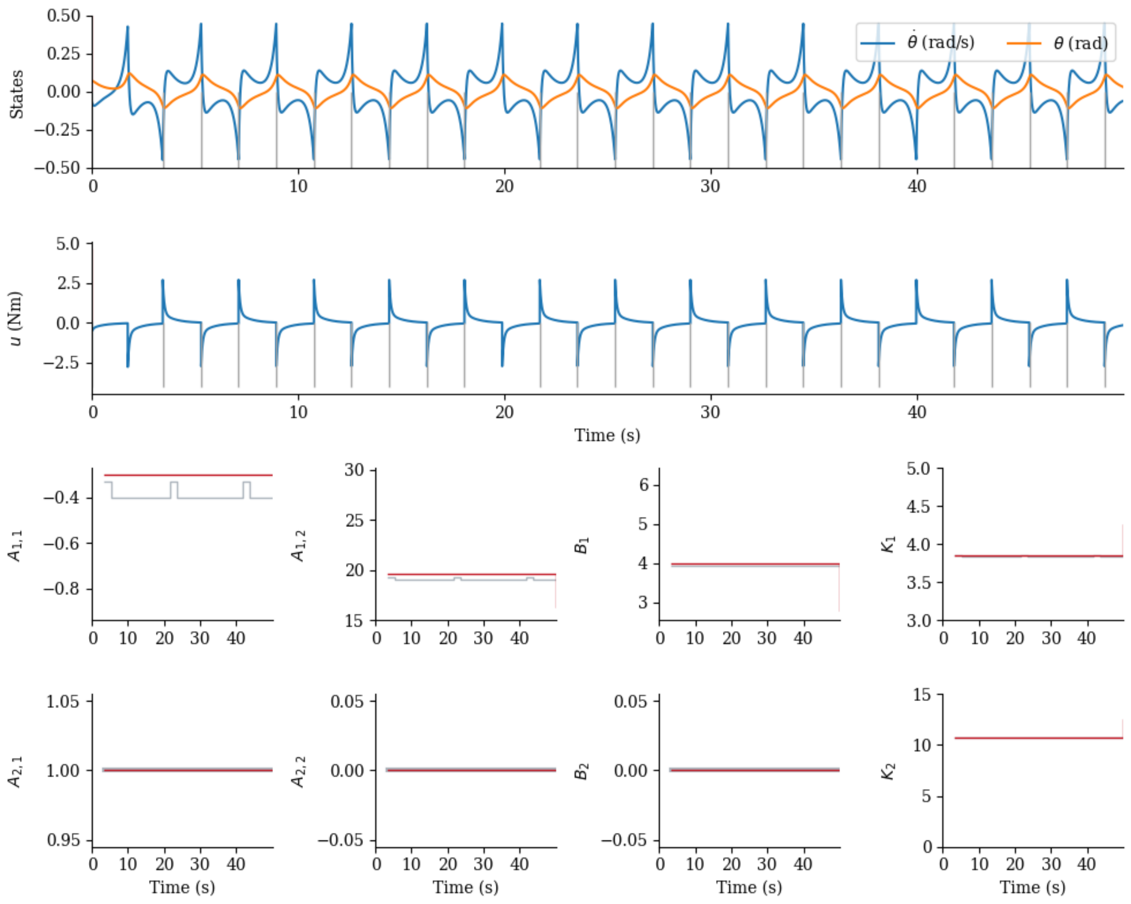

Figure 4.17: Data Informativity applied to SMH Based Intermittent Controller. The top row are the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. Row 3 and 4 plot the true value (red) and the estimated one (grey) using the Data Informativity Framework. Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2. The system is not changing during the simulation.

Based on an Intermittent Controller, the Data Informativity can estimate accurately the matrices **A** and **B** without the need for any external disturbance as the excitation introduced by the Intermittent Controller is enough. The occlusion is enabled in this simulation, however, having $t_{DI} = 1sec$ and the open-loop interval higher than t_{DI}, the Data Informativity is performed at each triggering in IC, thus there is no triggering to occlude in the data.

In contrast to the previous simulation, the following simulation uses the output of the Data Informativity framework to redesign the controller online. The newly designed controller is then used until the next Data Informativity computation. It is important to remember that the DI algorithm is only performed when the controller is closing the loop to avoid updating the controller during an open-loop interval (c.f: Figure 4.3). The following two figures show the impact of the DI windows during the simulation.

Figure 4.18 shows the impact of the variable t_{DI} in conjunction with τ in Intermittent Control for the time series. Figure 4.19 shows the output of the Data Informativity algorithm and few observations can be made. If $t_{DI} < \Delta_{OL}$, the Data Informativity algorithm is only using a single open-loop interval to estimate the plant. When $t_{DI} = 1$ second, the estimation of the matrices coefficients is the furthest away compared to the other t_{DI} value. However, the estimation is close to the true value. Increasing t_{DI} to 2 seconds makes the output of the Data Informativity oscillate between two values. This is due to the algorithm taking a single event or two, based on the timing of the events. The estimation is moving a little around the nominal value due to the timing of each event but stays close to the nominal value. Moreover, increasing t_{DI} to 5 seconds gives enough data to the algorithm to estimate the matrices coefficient very closely to the nominal value as well as removing any type of variability in the DI output.

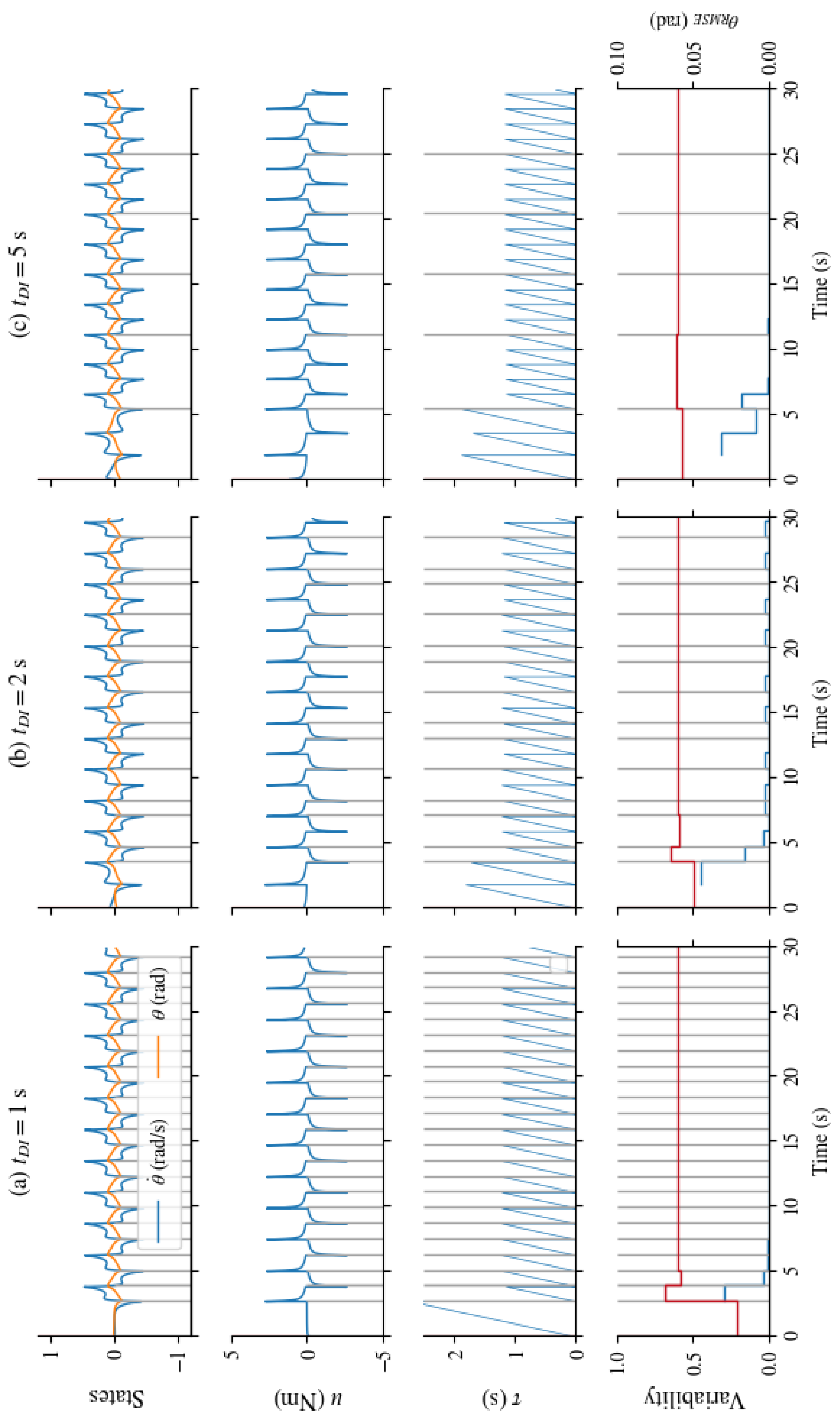

Figure 4.18: Impact of the t_{DI} in Intermittent Control. The three figures are using a different t_{DI}: (a) $t_{DI} = 1sec$, (b) $t_{DI} = 2sec$, (c) $t_{DI} = 5sec$. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in grey) and θ_{RMSE} (in red)

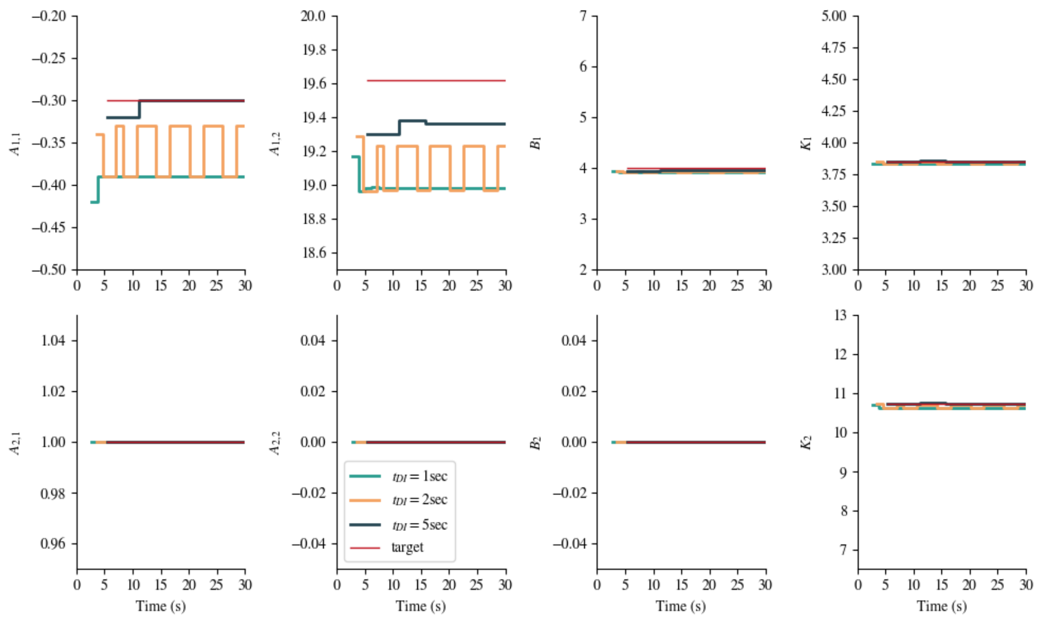

Figure 4.19: Impact of the t_{DI} in Intermittent Control. The three colors correspond respectively to $t_{DI} = 1sec$ (green), $t_{DI} = 2sec$ (orange), $t_{DI} = 5sec$ (blue). Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2.

Whilst the DI algorithm can estimate the system matrices accurately when the system is not changing through time, the focus is now to assess if this technique can handle any change of plant parameters during the simulation. Based on the previous assessment presented in Figure 4.15, it is important to keep the matrices of the plant and the one used to design the controller in sync to ensure better performances, such as regulation and tracking. In the following simulations, the system is getting updated at 30 seconds. The initial pendulum length of 1 meter is increased to 1.2 meters. This modification of the system is shown in the following figures in rows three and four of Figure 4.20 and is changing the matrices **A** and **B** as follows:

$$A = \begin{bmatrix} -0.3 & 19.62(\rightarrow 17.84) \\ 1 & 0 \end{bmatrix} B = \begin{bmatrix} 4(\rightarrow 3.31) \\ 0 \end{bmatrix} \tag{4.9}$$

Figure 4.20 is divided in two. The left set of plots is based on $t_{DI} = 5sec$ whereas the right side is using $t_{DI} = 10sec$. Based on previous results, these simulations are using occlusion of the events. By increasing t_{DI}, the amount of data used by the Data Informativity algorithm is increased. This helps to get a more consistent estimation as presented in Figure 4.18. However, in the case of adaptation, it is important to react quickly to any potential change that could bring instability to the system.

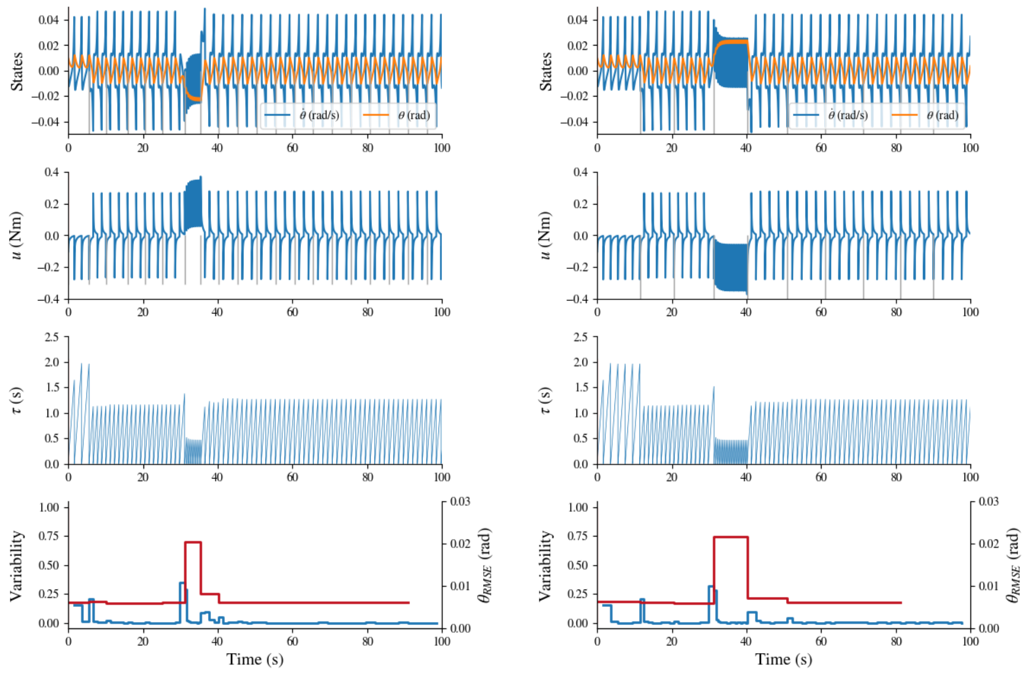

Figure 4.20: Impact of the t_{DI} in Intermittent Control applied to the Single Inverted Pendulum. The two figures are using a different t_{DI}: (a) $t_{DI} = 5sec$ and (b) $t_{DI} = 10sec$. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). The system is getting updated at 30 seconds during the simulation: the length of the pendulum is moving from 1m to 1.2m

Once the system is changing from 1m to 1.2m, there is a noticeable change in the behavior of the system's states. The Data Informativity estimation being consistent in estimating the plant matrices allows it to be run repetitively and, hence can be used to estimate new matrices and be used to redesign the controller when a change occurs during the simulation. In both cases, the algorithm is estimating accurately the new set of matrices introduced by the increase of the pendulum's length as presented in Figure 4.21. The

Intermittent Controller is redesigned after each estimation from the Data Informativity, which helps the states of the system go back to the symmetrical balancing behavior. Whilst increasing t_{DI} is also increasing the amount of data available to the Data Informativity algorithm, the results of the estimation stay accurate with $t_{DI} = 5$ seconds and $t_{DI} = 10$ seconds, and the variability range is unchanged. The variability is only going up when the system is changing hence the two following open-loop intervals are not alike anymore, then it is back to nearly zero once the matrices are correctly estimated.

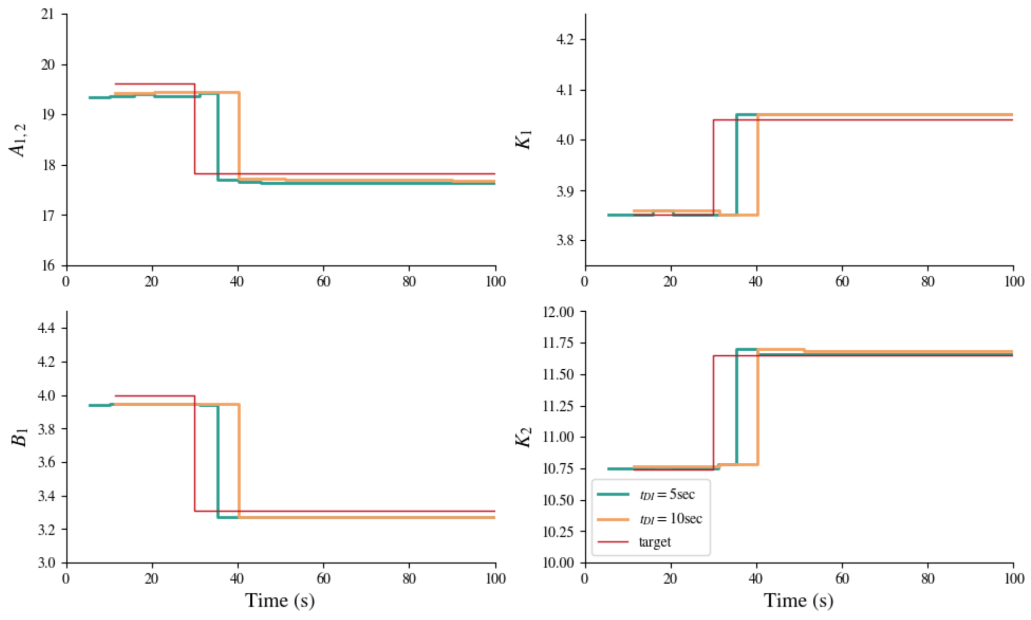

Figure 4.21: Impact of the t_{DI} in Intermittent Control applied to the Single Inverted Pendulum. The two colors correspond respectively to $t_{DI} = 5sec$ (green) and $t_{DI} = 10sec$ (orange). Each subplots represents in order: $A_{1,2}$, K_1, B_1 and K_2. The system is getting updated at 30 seconds during the simulation: the length of the pendulum is moving from 1m to 1.2m.

Based on the previous simulation, it is clear that Data Informativity can be used with IC to detect changes in the system. However, it is important to assess if this framework can handle more difficult changes in the system. Increasing the pendulum length is making this system slower and easier to control. Moreover, even if the controller is not getting updated, it can control the longer pendulum. The following simulation updates the length of the pendulum multiple times, in both directions. The pendulum starts with an initial length of 1m, like previously, and increases to 1.2m at 50 seconds. Then comes back to 1m at 100 seconds before dropping to 0.8m at 150 seconds. Finally, the pendulum is increasing again back to 1m at 200 seconds.

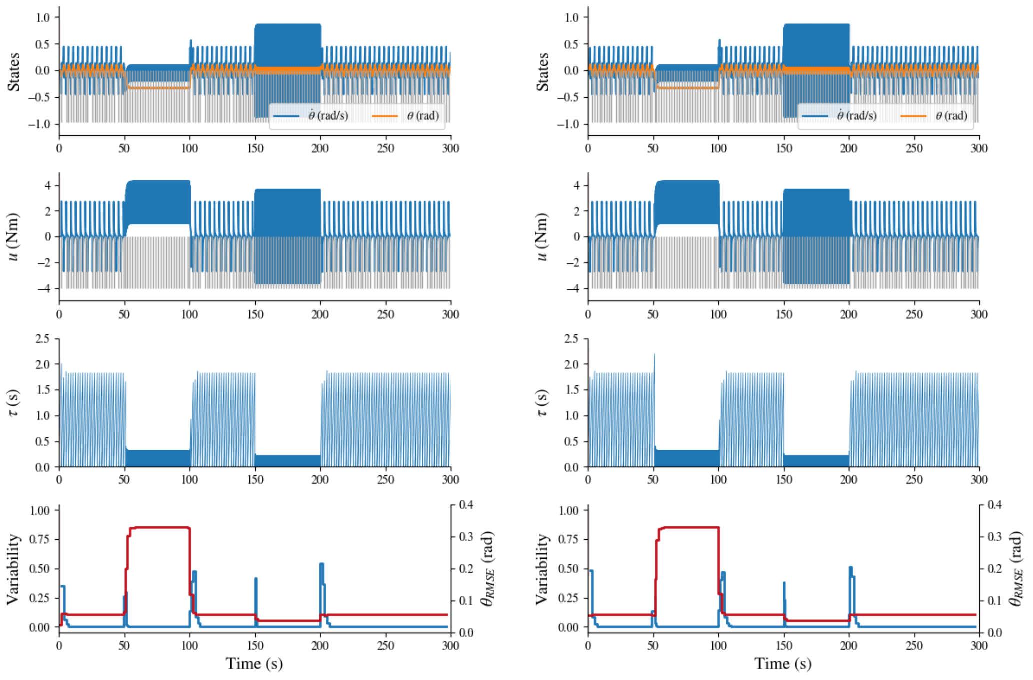

Figure 4.22: Data Informativity output using SMH Based Intermittent Controller without redesign, $t_{DI} = 1sec$. The left side does not have occlusion. Occlusion is enable on the right side. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

Figure 4.22 is using $t_{DI} = 1sec$. The output of the Data Informativity algorithm is not used to update the controller online. This is to emphasize the importance of the occlusion technique where the data passed to the Data Informativity contains discontinuity due to events in the intermittent control input. As explained, the controller can keep the pendulum stable even with different lengths. Alternative approaches such as linear stability analysis can be used as the pendulum is operating around the equilibrium point. The triggering is getting smaller as the length is changing, due to the mismatch between the plant and the controller. When the pendulum's length increases to 1.2m, the algorithm accurately estimates the matrices **A** and **B** with or without occlusion. However, when the pendulum's length decreases to 0.8m, the occlusion effect plays a major role.

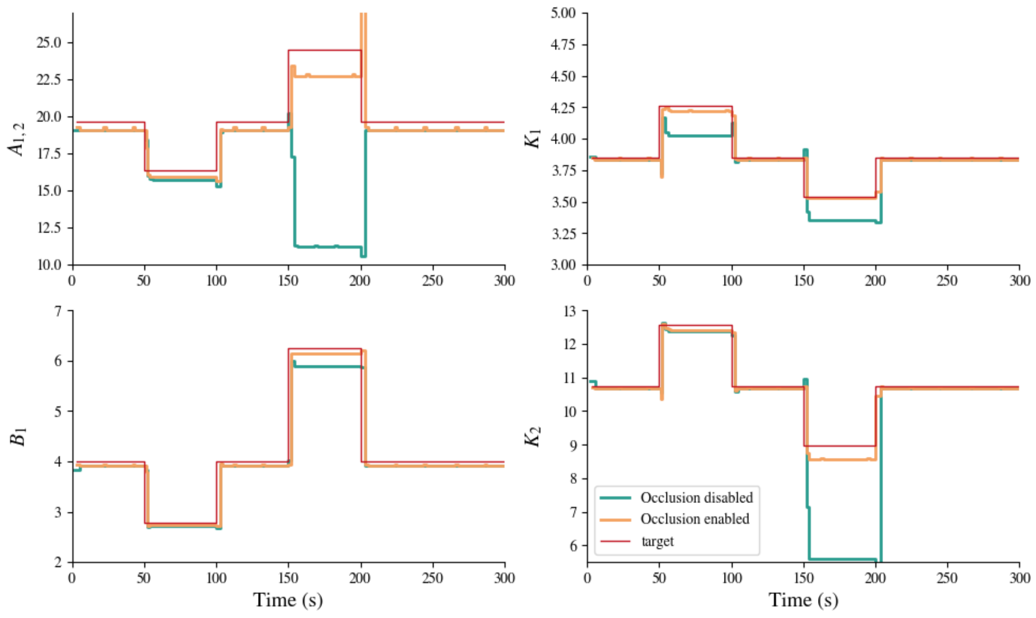

Figure 4.23: Impact of the t_{DI} in Intermittent Control applied to the Single Inverted Pendulum without redesign. $t_{DI} = 1sec$. The two colors correspond respectively to $t_{DI} = 5sec$ (green) and $t_{DI} = 10sec$ (orange). Each subplots represents in order: $A_{1,2}$, K_1, B_1 and K_2. The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

Without the occlusion enabled, the coefficient $A_{1,2}$ is not properly estimated: 11.2 instead of 24.5. This is then reflected in the values of the state feedback k between 150 and 200 seconds. With the occlusion enabled, the estimation is closer to the true value: 22.7 instead of 24.5. In addition, the estimation of **B** is also more accurate, hence the state feedback computation is almost identical to the true value during the entire simulation.

Now that the estimation of matrices by the Data Informativity has been verified as accurate, especially using occlusion, the estimation output of Data Informativity is used to redesign the Hold and state feedback gain in the controller, which are highlighted in green in Figure 4.14.

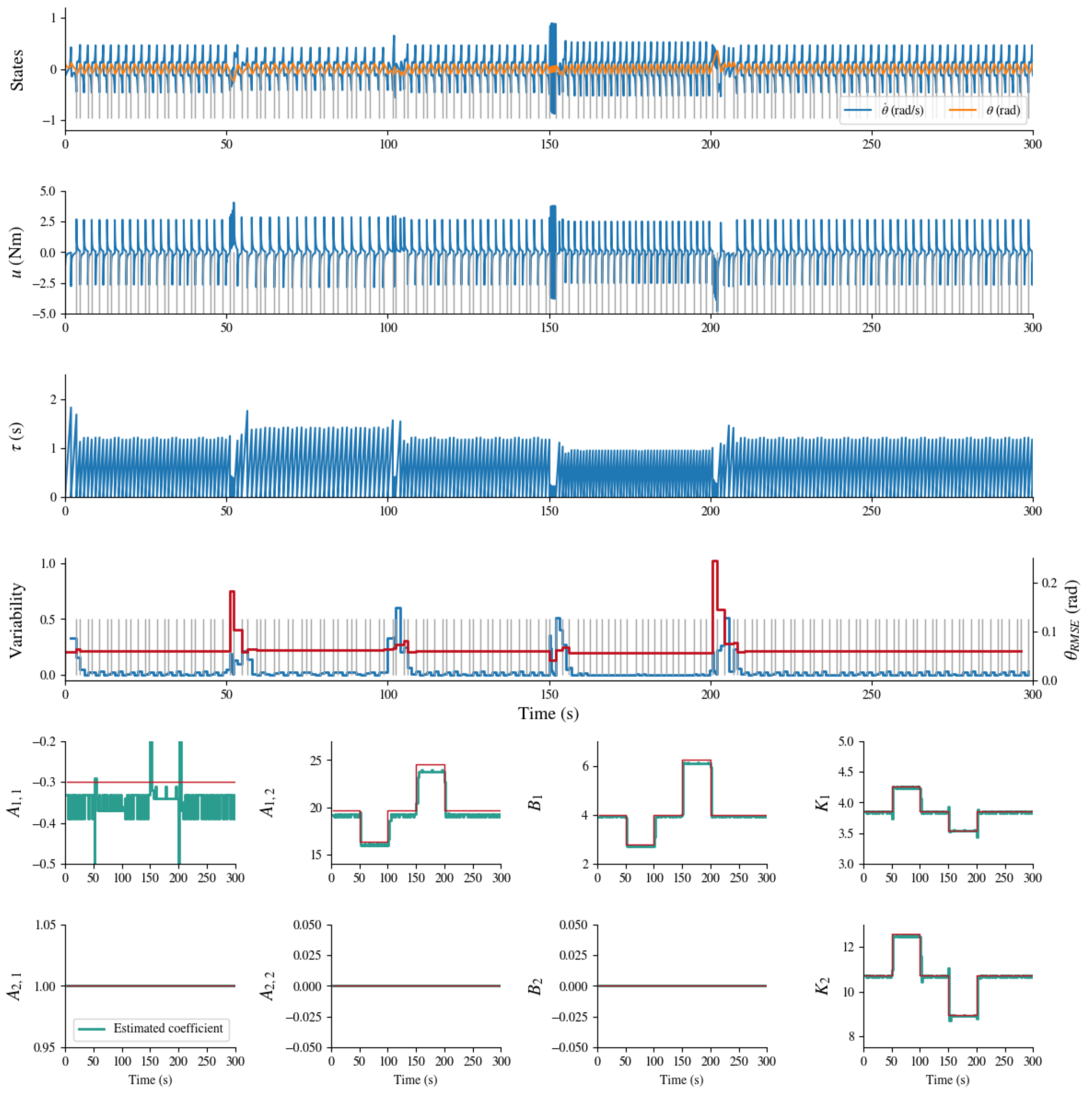

Figure 4.24: Data Informativity output using SMH Based Intermittent Controller with redesign. Occlusion is enabled. The top row are the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. Third row is the open-loop interval τ. Fourth row is the variability of θ (in blue) and θ_{RMSE} (in red). Then, row 5 and 6 represent the matrices coefficients using the Data Informativity Framework: true value (red) and the estimated one (green). Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2. The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

Data presented in Figure 4.24 uses the output of the Data Informativity for the online redesign of the controller, and occlusion is enabled. Significant improvements compared to Figure 4.22 and Figure 4.23 can be noticed: the plant keeps its symmetrical behavior even when the system is changing. Moreover, the DI estimation is almost perfect, especially focusing on the state feedback values. These tests ensured that DI can be used to give an accurate estimation of the plant to design an Intermittent Controller.

One more test focuses on slow and small changes in the system, where states and control input do not change significantly when an update is made, hence more difficult to detect. In Figure 4.25, the system is getting updated every 10 seconds between 30 seconds and 70 seconds, changing the pendulum length from 1m to 1.2m (0.04m / 10 sec). Then the length is getting decreased similarly, from 150 seconds to 240 seconds, going from 1.2m to 0.8m (-0.04m / 10 sec). In the first part of the simulation, as the system is growing, it is getting easier to control. However, in the second half of the simulation, the pendulum is getting harder to control as the poles for this system are getting closer to 0. First, the Intermittent Controller is not getting using the output of the DI algorithm (top row in Figure 4.25). Then, the same experiment in run with the Intermittent Control using the output of the DI algorithm (bottom row in Figure 4.25).

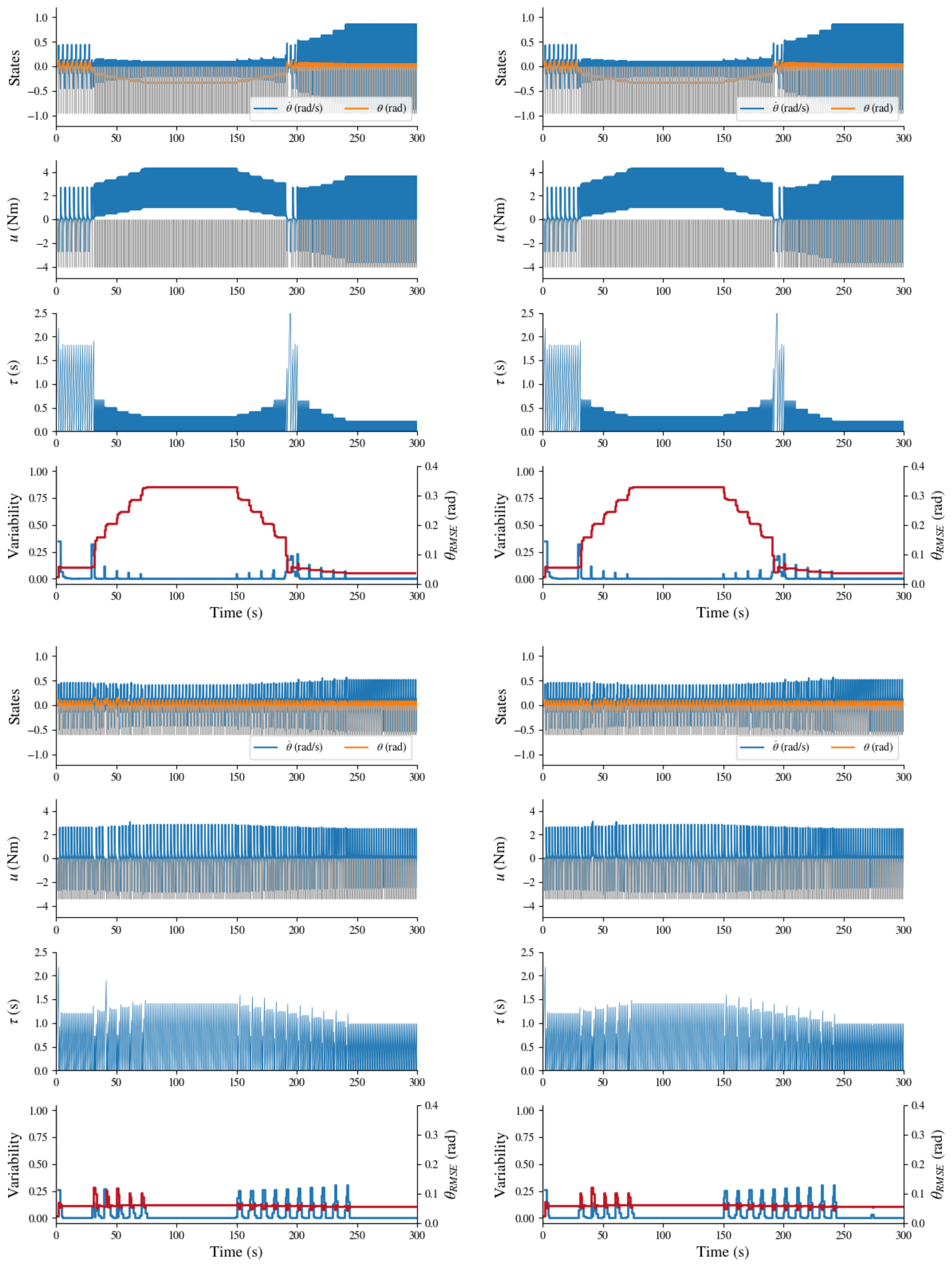

Figure 4.25: Data Informativity using Intermittent Control with and without updating the controller. $t_{DI} = 1$ sec. The top two subplots are not updating the controller with DI output. The bottom two are using the output of the DI for redesign. The two on the left are not using occlusion and the two on the right are using occlusion. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). The system is getting updated every 10 seconds between 30 seconds and 70 seconds, changing the pendulum length from 1m to 1.2m (0.04m / 10 sec). Then the length is getting decreased similarly, from 150 seconds to 240 seconds, going from 1.2m to 0.8m (-0.04m / 10 sec).

Multiple conclusions can be drawn from Figure 4.25. Enabling occlusion is only having an impact when the controller is not getting redesigned. This is due to $t_{DI} = 1sec$ being below the open-loop interval, hence each redesign occurs within an open loop, so no event needs to be removed. The occlusion only helps when the pendulum's length is getting below 1m. Moreover, redesigning the controller is helping in keeping the θ_{RMSE} low as the state is not drifting away from the reference.

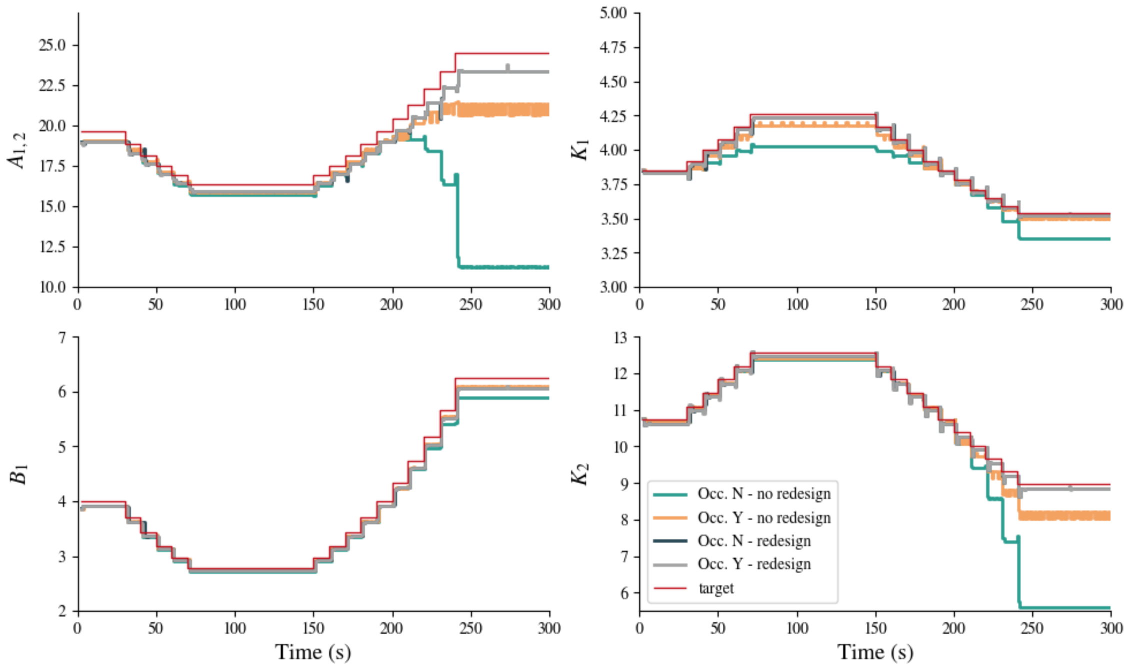

Figure 4.26: Data Informativity using Intermittent Control with and without updating the controller. $t_{DI} = 1sec$. The four colors correspond respectively to: no occlusion and no redesign (green), occlusion enabled and no redesign (orange), no occlusion and with redesign (blue) and occlusion enabled and with redesign (grey). Each subplots represents in order: $A_{1,2}$, K_1, B_1, and K_2. The system is getting updated every 10 seconds between 30 seconds and 70 seconds, changing the pendulum length from 1m to 1.2m (0.04m / 10 sec). Then the length is getting decreased similarly, from 150 seconds to 240 seconds, going from 1.2m to 0.8m (-0.04m / 10 sec).

Furthermore, as the pendulum is getting longer, the Data Informativity can capture with good accuracy the matrices **A** and **B** coefficients (see Figure 4.26). Looking at the time series, the open-loop interval is decreasing significantly due to the pendulum angle θ not being centered on 0 anymore. The control input is not strong enough for the new pendulum length. However, the controller is still ensuring that the system remains stable.

In the second half of the simulation, θ is slowly going back to 0 as the pendulum decreases its length. It is worth to notice the open-loop interval quickly increases when the pendulum is back to the 1-meter length, matching the initially designed controller, showing the proxy of using the open-loop interval as an indicator of the match between the hold and the plant. Once the pendulum length is lower than the one used for designing the controller, the open-loop interval is reduced again. This is due to a control input being too aggressive for the size of the system. This can be captured by looking at $\dot{\theta}$, the velocity of the pendulum, which is increasing at each system change.

In addition, when the occlusion is disabled and the controller is not getting redesigned, the output of the DI algorithm is drifting away from the actual value, especially visible in Figure 4.26 ($\mathbf{A}_{1,2}$). Enabling the occlusion is helping bring back the offset in the estimation, however, this is still not enough to get to the correct value. By redesigning the controller, the data passed to the algorithm is then rich enough to properly estimate $\mathbf{A}$. Interestingly, $\mathbf{B}$ estimation is accurate in all cases. As this estimation is being performed at $t_{DI} = 1$ second, it can be quite noisy as seen in Figure 4.26 ($\mathbf{A}_{1,1}$). However, the state feedback computed using $\mathbf{A}$ and $\mathbf{B}$ is more accurate when occlusion is enabled.

Whilst the previous simulations were using $t_{DI} = 1 sec$, it is worth assessing the impact of the time between retraining. The evolution of the system is similar to the previous simulation where the pendulum's length is increasing and decreasing in a ”smooth” manner. The following simulations use 4 different values for t_{DI}: 1, 3, 5, and 10 seconds. Results are shown in Figure 4.27 and 4.28.

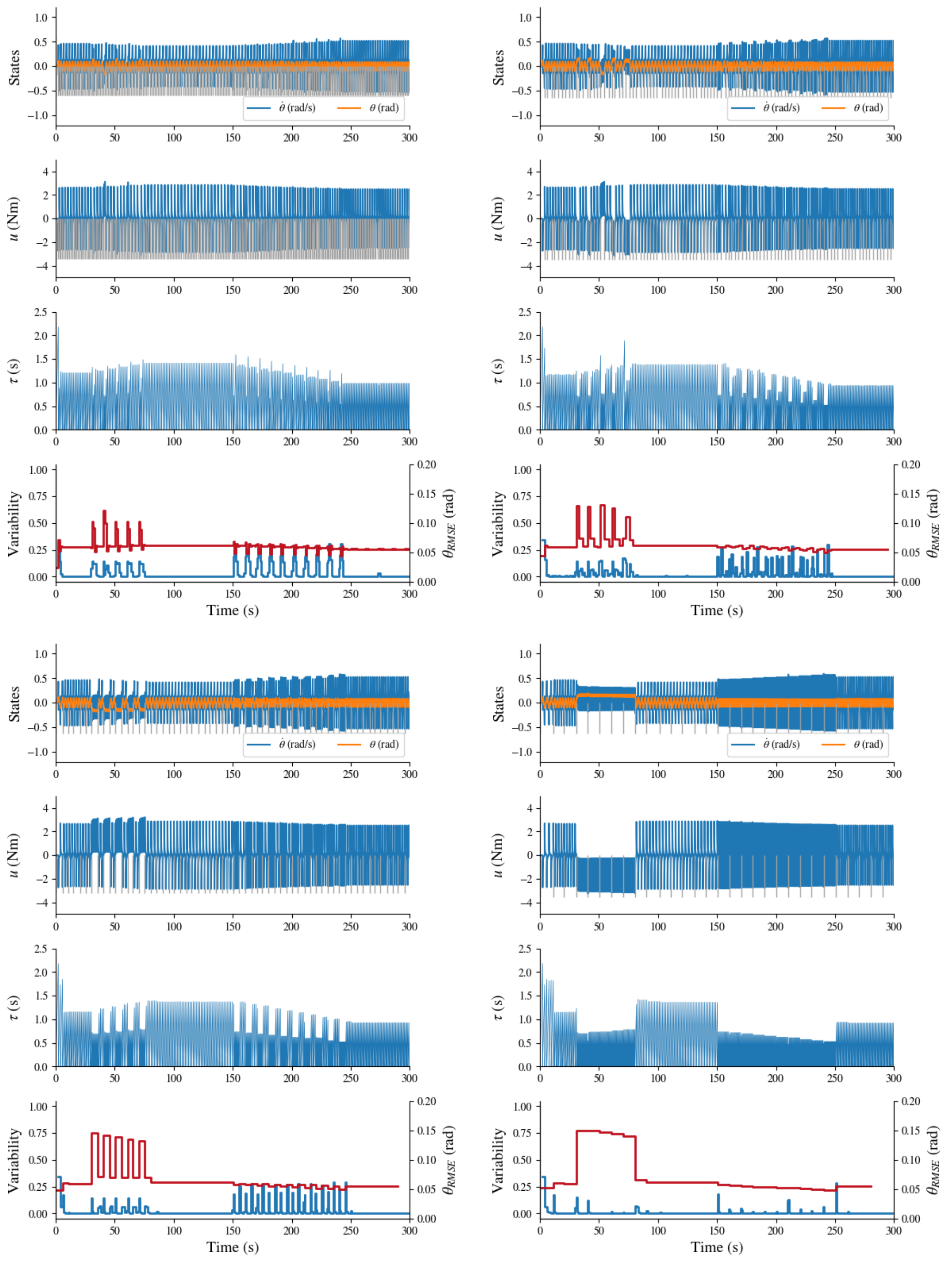

Figure 4.27: Impact of the t_{DI} in Intermittent Control. The four figure are using a different t_{DI}: (a) $t_{DI} = 1$ sec, (b) $t_{DI} = 3$ sec, (c) $t_{DI} = 5$ sec and (d) $t_{DI} = 10$ sec. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). The system is getting updated every 10 seconds between 30 seconds and 70 seconds, changing the pendulum length from 1m to 1.2m (0.04m / 10 sec). Then the length is getting decreased similarly, from 150 seconds to 240 seconds, going from 1.2m to 0.8m (-0.04m / 10 sec).

Focusing on $t_{DI} = 10sec$, this estimation time is the same as the update of the system. First, looking at the time series output in Figure 4.27, the pendulum's angle is not centered on 0 as soon as the pendulum length changes. As the pendulum increases its length, θ starts drifting away from the set point. Due to the update of the controller, the estimation from the DI framework does not diverge as the system changes. However, the Data Informativity algorithm is always lagging in its estimation of the system due to t_{DI} being the same as the timing when the system changes. When the pendulum stops increasing at 70 seconds, the estimation is getting back "in-sync".

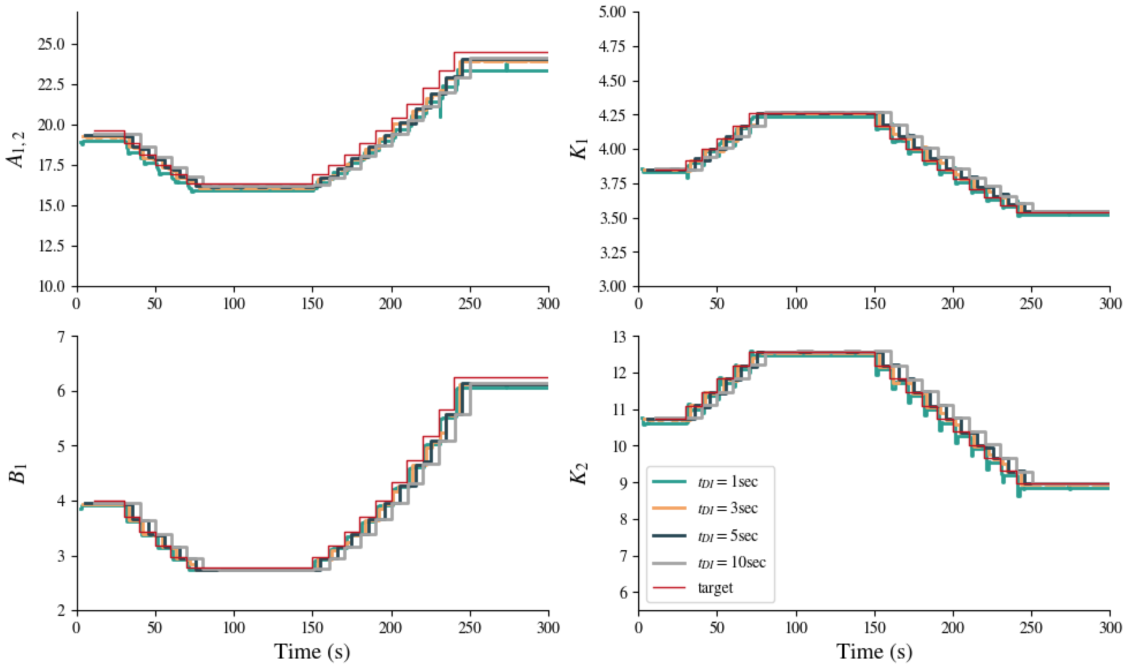

Figure 4.28: Data Informativity using Intermittent Control with different t_{DI}. The four colors correspond respectively to: $t_{DI} = 1$ sec (green), $t_{DI} = 3$ sec (orange), $t_{DI} = 5$ sec (blue) and $t_{DI} = 10$ sec (grey). Each subplots represents in order: $A_{1,2}$, K_1, B_1, and K_2. The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

In the three other simulations, where $t_{DI} = 5sec$, $t_{DI} = 3sec$, and $t_{DI} = 1sec$, the DI algorithm has time to get to the correct matrices for the controller to be used with the appropriate matching system as seen in Figure 4.28. The angle of the pendulum is getting away from the reference and then brought back to the set point once the controller matches the system before the plant changes again. Then the system changes and the matrices estimations follow the same pattern as previously mentioned. It is important to notice that reducing t_{DI} can help the controller to be ”in-sync” with the current plant, however, the amount of data is getting reduced and brings more variability in the estimation.

Even though this implementation is not designed to handle noisy data, it is worth assessing its performance in this situation. This is because the Intermittent Controller is breaking the feedback loop, hence preventing the noise from propagating as much as the Continuous Controller case. In the following simulation, input noise has been added to the controller input before being applied to the system. This is used to represent motor noise. Results are shown in Figure 4.29.

In comparison to Figure 4.24, the output of the Data Informativity presented in Figure 4.29 shows an offset with the true value. This offset is mostly affecting coefficients $\mathbf{A}_{1,2}$ and $\mathbf{B}_1$, which is only affecting the computation of the state feedback $\mathbf{k}_2$. However, it gives a more accurate matrices estimation compared to the Continuous Controller (presented in Figure 4.11). Even with this discrepancy in the matrices estimation, the Intermittent Controller is still able to adapt and control the single inverted pendulum with a high level of noise. This also helps in bringing some variability in the simulation.

Cartpole system To increase the complexity of the system to estimate by the DI algorithm, the plant is moved to a cart pole system for the next simulations (see the model in Appendix B). The DI algorithm is now estimating 16 coefficients (4-by-4 system) for the matrix **A** as well as 4 additional ones for the matrix **B**. Following the similar approach applied to the pendulum, in these simulations, the pole of the cart pole system changes

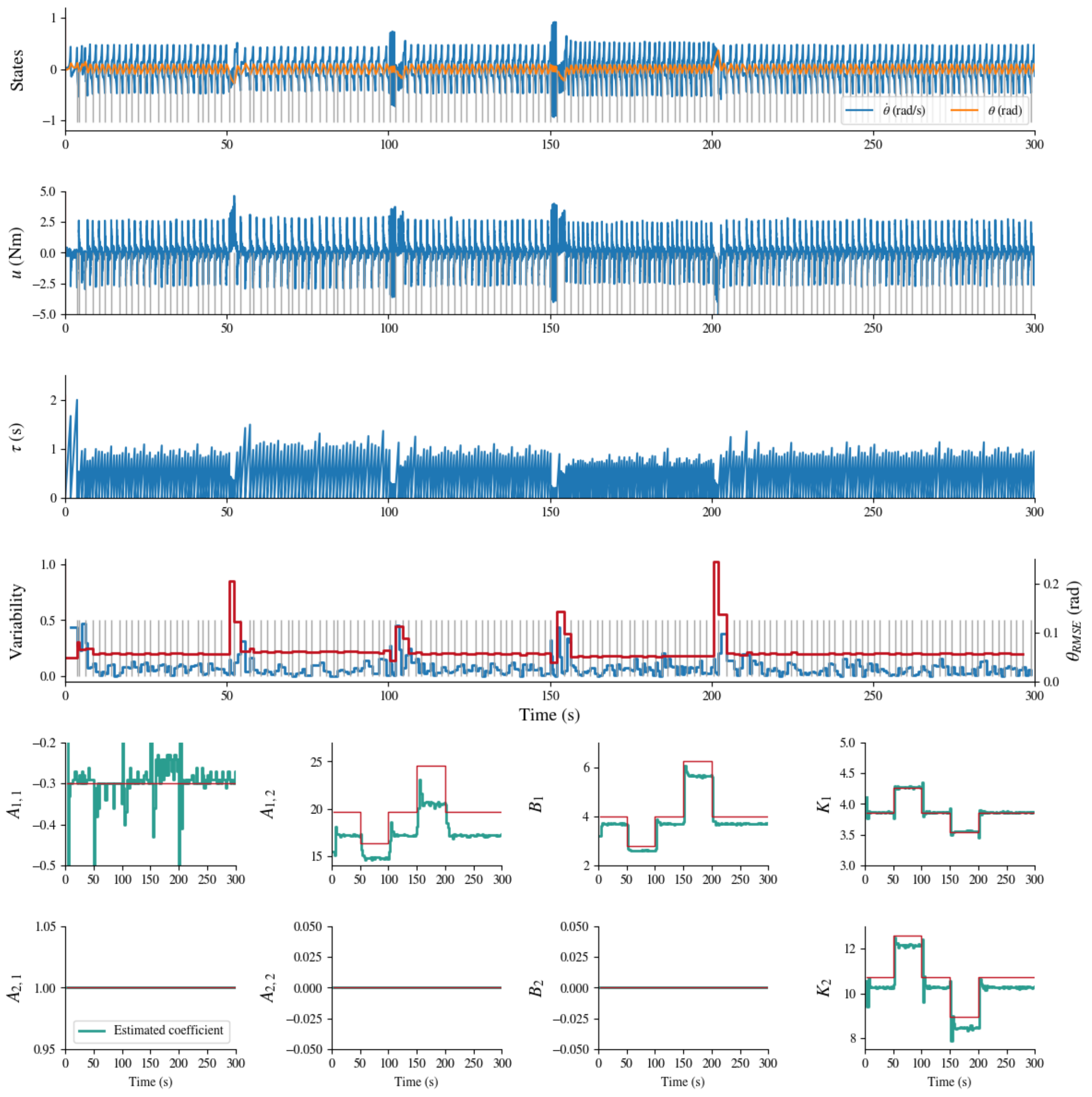

Figure 4.29: Data Informativity output using SMH Based Intermittent Controller with redesign and $v_u = 0.1$. Occlusion is enabled. The top row are the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. Third row is the open-loop interval τ. Fourth row is the variability of θ (in blue) and θ_{RMSE} (in red). Then, row 3 and 4 represents the matrices coeffients using the Data Informativity Framework: true value (red) and the estimated one (green). Each subplots represents in order: $A_{1,1}$, $A_{1,2}$, B_1, K_1, $A_{2,1}$, $A_{2,2}$, B_2 and K_2. The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

through time, from 1m to 1.2m (0.04m / 10 seconds) as well as its mass 1kg to 1.2kg (0.04kg / 10seconds). Similarly to the pendulum, the length and weight then decrease similarly, starting at 150 seconds in the simulation: 0.04m / 10 seconds and -0.04kg / 10 seconds. Results are shown in Figure 4.30.

In these simulations, the weight is also updated to introduce bigger changes in matrices **A**, **B**, and **k**. It has been decided to focus on $\mathbf{A}_{1,3}$, $\mathbf{A}_{3,1}$, $\mathbf{A}_{3,2}$, $\mathbf{A}_{4,2}$, $\mathbf{B}_{1,3}$, $\mathbf{B}_{1,3}$, $\mathbf{k}_{1,2}$ and $\mathbf{k}_{1,4}$ as they are the coefficients that are changing the most in matrices **A**, **B** and **k** (see Appendix B Equation B.3). Figure 4.30 and 4.31 show the impact of the t_{DI} value in getting the correct estimation, in conjunction with the presence of the occlusion or not.

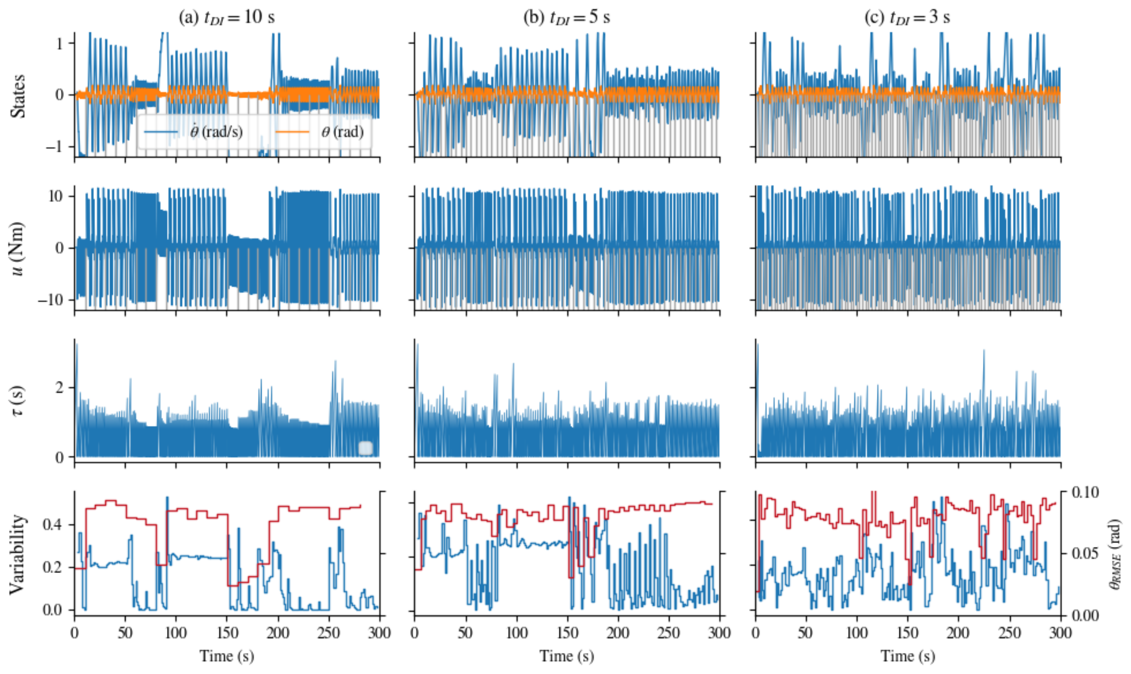

Figure 4.30: Impact of the t_{DI} in Intermittent Control applied to the Cart Pole with occlusion enabled. The three figures are using a different t_{DI}: (a) $t_{DI} = 10sec$, (b) $t_{DI} = 5sec$ and (c) $t_{DI} = 3sec$. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). The system is getting updated every 10 seconds between 30 seconds and 70 seconds, changing the pendulum length (and mass) from 1m (kg) to 1.2m (kg) (0.04m (kg)/ 10 sec). Then the length is getting decreased similarly, from 150 seconds to 240 seconds, going from 1.2m (kg) to 0.8m (kg)(-0.04m (kg)/ 10 sec)

Even having to estimate more coefficients for this system, the DI algorithm is still able to accurately estimate the matrices **A** and **B** for the cart pole system. Moving to a system with a higher number of states is amplifying the importance of the occlusion windows to hide the sharp discontinuity in the data passed to the algorithm.

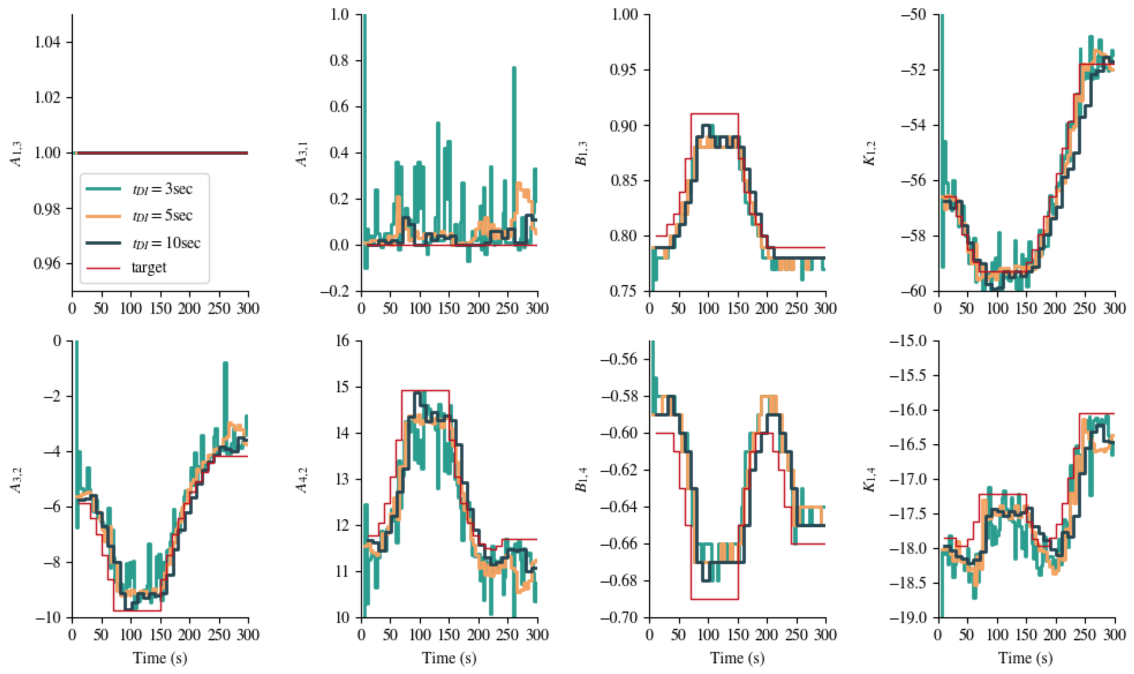

Figure 4.31: Impact of the t_{DI} in Intermittent Control applied to the Cart Pole with occlusion enabled. The three figures are using a different t_{DI}: (a) $t_{DI} = 3$ sec, (b) $t_{DI} = 5$ sec and (c) $t_{DI} = 10$ sec. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The last row is the variability of θ (in grey) and θ_{RMSE} (in red). The system is getting updated every 10 seconds between 30 seconds and 70 seconds, changing the pendulum length (and mass) from 1m (kg) to 1.2m (kg) (0.04m (kg)/ 10 sec). Then the length is getting decreased similarly, from 150 seconds to 240 seconds, going from 1.2m (kg) to 0.8m (kg)(-0.04m (kg)/ 10 sec)

Similarly to the pendulum simulations, t_{DI} has been decreased gradually to ensure that the controller designed through the DI algorithm output is applied ”in sync” with the system. These simulations have occlusion enabled. For the cart pole system, going from $t_{DI} = 10$ sec to $t_{DI} = 5$ sec is already introducing significant variability in the matrices estimations. Enabling the occlusion is helping in that regard. Similarly to results presented in Figure 4.17, moving to $t_{DI} = 3$ sec without occlusion compared to occlusion enabled is not improving the estimation. This is mostly related to the estimation being performed every open-loop interval.

Whilst Data Informativity gives an accurate estimation of the system matrices **A** and **B**, this algorithm can become difficult in the case of a high dimension system changing fast as presented in the cart pole system. Moreover, this implementation of the algorithm is not made to handle noisy data, hence it introduces an estimation issue and implies a wrongly designed controller.

4.2.2.3 Summary

As presented in this section, Data Informativity can be used with the Intermittent controller framework. Moreover, it can detect changes in the system accurately and this information can be used to redesign the controller. In comparison to the Continuous Controller, it is not necessary to add any external signal to get an accurate estimation of the plant via this algorithm. This is due to the constant motions introduced by the intermittency. However, this intermittency is also bringing discontinuity in the control signals. As presented above, Data Informativity can be improved by simply using occlusion around the triggering event happening in Intermittent Control. The focus has been on the single inverted pendulum, an unstable plant, however, this has been also applied to the cart pole system and showed similar results even though the system is more complex.

Whilst the Data Informativity framework is quite efficient for the noiseless case, the estimation starts to deviate from the correct value when some noise is added to the simulation. It is known that this algorithm has been designed for noiseless simulation. However, by opening the loop, Intermittent Control can reduce the noise propagation, hence still being able to use to a certain extent the noiseless implementation of this framework.

4.3 Discussion

In this chapter, the Reinforcement Learning and Data Informativity frameworks have been applied in the context of Intermittent Control. Both of these frameworks have been originally designed for continuous control, where the control input is continuously generated from the current state of the plant.

Focusing on the Reinforcement Learning implementation, it can be used within Intermittent Control to find the optimal set of gains for the pendulum system. However, this algorithm needs an initial stable set of controller gains to start with. Even though this condition is met, convergence is not ensured using Intermittent Control due to the latency in getting samples induced by the triggering mechanism.

The main drawback of this approach is the data that can be passed to the algorithm. This data can only be captured at the time of an event. This is due to the similarity with the continuous controller when the loop is closed. Hence operating the Intermittent Controller in a clock-driven mode can be the best option to ensure stability. This is disabling the advantages of Intermittent Control in opening the loop. However, even though the Intermittent Controller is looking very similar to Continuous Control when the minimum open-loop is set to a small value, this is enough for the estimation to perform accurately as presented in Section 4.2.1. In addition, this algorithm is only able to estimate the state feedback without estimating the system matrices **A** and **B** which makes this approach not suitable for adaptation.

In contrast to the Reinforcement Learning implementation, the Data Informativity can use all data generated by the Intermittent Controller, hence it is not necessary to operate in a clock-driven mode. In addition, this can be applied at each triggering event to react quickly to any changes in the system. In comparison to the continuous controller, it is not necessary to apply any additional disturbance signal to get an accurate estimation of the matrices. This is an important advantage as it is possible to keep the system operational and use this data for estimation without having to disturb regular operating mode. Moreover, by adjusting the timing t_{DI}, it is possible to get a stable estimation or get some variability due to the constant refresh of the controller during simulation. Finally, even with the presence of noise, it is possible to get an estimation good enough to keep stability in the system even when it is changing over time.

Whilst both approaches have advantages and drawbacks, the following work is focused on Data Informativity only, as this can capture accurate changes in the system as well as estimating both $\mathbf{A}$ and $\mathbf{B}$ matrices instead of simply getting the estimation of the state feedback. This allows a full redesign of the intermittent controller when needed as simply updating k is not enough when the system is changing over time, creating a mismatch between the plant and the Hold in the controller (see Figure 4.15).

4.4 Summary

An overview of the work accomplished in this Chapter is summarized in Table 4.1. This table focuses on the pros and cons of using the RL / DI framework with a SMH based IC applied to Single Inverted Pendulum system. In addition, limitations and generalizations are mentioned. Everything is based on simulation data only, allowing easy assessment of the Intermittent Controller against different set of parameters, as well as system properties changing through time.

Algorithms	Pros	Cons	Limitations
RL + SMH	Capability to converge to optimal state feedback gains	Only able to use data at triggering time, when the loop is closed	Initial stable poles need to be located in between the optimal poles
DI + SMH	The algorithm is able to use open-loop generated data. Fast and accurate estimation of system matrices.	Need occlusion of events to improve identification	No variability in the output signal from the controller.

Table 4.1: Summary table covering Chapter 4 contributions.

Some generalisation can be made for each algorithm. First, in a real application, it is not possible to get access to the optimal poles of the system. Using RL with SMH can help to determine the approximate location of the optimal poles if the initials poles are converging or not. Focusing on the DI with SMH, Intermittent Control estimate the open-loop dynamics of the system compared to the Continuous Controller which estimate the closed-loop dynamics.

Chapter 5

Stochastic Intermittent Control

Intermittent control relies on knowing the system's matrices **A** and **B** in order to design the hold matrix A_c as well as the state feedback. However, these matrices can be difficult to obtain using Continuous Controller as presented in Section 4.2.2.1. The need of knowing the system matrices to design the hold is limiting the opportunities of Intermittent Control to be applied more widely. In this section, traditional System-Matched Hold (SMH) is replaced by Gaussian Processes. These Gaussian Processes are helping in moving forward a data-driven implementation of the Intermittent Controller, hence avoiding the need of knowing the underlying dynamics of the system.

First, Single Task Gaussian Processes is being investigated as well as how GP computing could be optimized, such as using Sparse GP, using system knowledge to reduce the number of GP as well as discarding states in some situations. After that, Multi Task GPs will be introduced to reduce the number of GP optimisation to a single one for any system dimensions. Then, adaptation using GP is investigated using online data for retraining.

5.1 Methods

When Gaussian Processes are used to model the system's dynamics, they are used to replace the Hold block in the Intermittent Control framework as presented in Figure 5.1. However, due to the parameterized nature of GPs using hyperparameters, they don't provide a direct representation of the system matrices **A** and **B**. Hence, the state feedback cannot be estimated from this model. The GP is using the control error as input when the Intermittent Controller is triggering. During the open-loop, GPs are using their own output as input for the next iteration. As presented in Figure 3.6.5, the GP is using output ΔX which is getting added to the current state. The trigger block (see Figure 5.1) is also updated, now using the GP uncertainties as an alternative triggering mechanism.

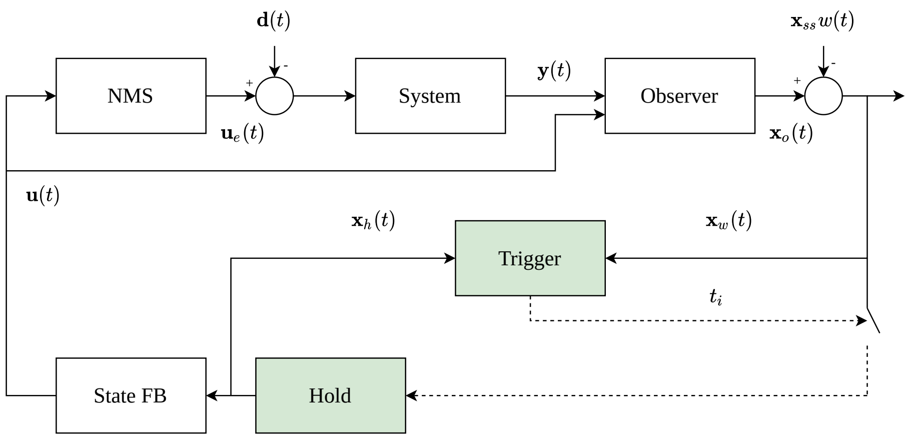

Figure 5.1: Intermittent Control block diagram. Block of the controller updated by the Gaussian Processes algorithm are highlighted in green.

5.1.1 Implementation

Using the same process as described in Section 4.2.2, the idea is to excite the system (in a stable regime) with a multisine input for the GPs to represent the system dynamic. If the system is unstable, a controller is needed in order to gather data in the correct region of the state space. However, this controller does not need to be perfect.

In the context of Gaussian Processes Regression (GPR), the goal is to find a representation of a dynamical function. In our case, we are trying to learn the dynamics of the plant. In the following discretized system:

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{A}_D\mathbf{x}(t)+\mathbf{B}_D\mathbf{u}(t)\\ \mathbf{x}(t+1) &= \begin{bmatrix}\mathbf{A}_D & \mathbf{B}_D\end{bmatrix}\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix}\end{aligned} \tag{5.1}$$

the state at the next iteration depends on the current state as well as the control input. It is similar as:

$$\mathbf{x}(t+1) = f(\mathbf{x}(t),\mathbf{u}(t)) \tag{5.2}$$

Assuming that $f(\mathbf{x}(t),\mathbf{u}(t)) \sim \mathscr{N}(0,K)$, the system becomes:

$$\begin{pmatrix} Y \\ x_n(t+1)\end{pmatrix} \sim \mathscr{N}(0,K_{pred}) \tag{5.3}$$

where n correspond to the n^{th} state in the case of the single-task GP and Y is the output recorded during training.

In this section, Gaussian Processes are integrated into the Intermittent Control Framework. For now, the focus is on the generalized hold and the predictor.

5.1.1.1 GP based Hold

Considering an Intermittent Controller using SMH as generalized hold, the Hold state $x_h(t+1)$ is computed using (see section 3.3.2):

$$\mathbf{x}_h(t+1) = \mathbf{A}_c\mathbf{x}_h(t) \tag{5.4}$$

which is equivalent to (using Equation (3.20)):

$$\begin{aligned}\mathbf{x}_h(t+1) &= (\mathbf{A}-\mathbf{B}\mathbf{k})\mathbf{x}_h(t) \\ &= \mathbf{A}\mathbf{x}_h(t) - \mathbf{B}\mathbf{k}\mathbf{x}_h(t)\end{aligned} \tag{5.5}$$

Taking the Equation (5.2) and $u(t) = -kx$, and replacing in Equation (5.1):

$$\begin{aligned}\mathbf{x}(t+1) &= f(\mathbf{x}(t), u(t)) \\ &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t)\end{aligned} \tag{5.6}$$

Using the Equation (5.3)

$$\mathbf{A}\mathbf{x}_h(t) \sim \mathcal{N}(0, K_{pred}) \tag{5.7}$$

then we obtain:

$$x_h(t+1) = \mathcal{N}(x_h(t)|0, K_{pred}) \tag{5.8}$$

Step-by-step

At each iteration, the previous estimated state is used as an input. In addition, there are two ways of getting a sample from $\mathcal{N}(x_h(t)|0, K_{pred})$:

- use $x_{h_{tmp}} = \mu$, where μ is the mean of the distribution. This is called the ”naive” approach and bring very small variance in the case of multiple step ahead prediction (Kocijan 2008).
- use a sample of $x_{h_{tmp}} \in \mathcal{N}(\mu, \sigma)$ as returned value. This allows to bring uncertainties into the IC framework.

Pre-computed trajectory

Another approach is to use a pre-computed trajectory for the next open-loop interval. This is using the GP as a multi-step ahead prediction instead of a single step. This approach helps us on the uncertainties being unnaturally low in the case of single-step prediction (ibid.). In addition, the pre-computed trajectory can be based on the mean of the distribution or the possibility to draw a sample from the distribution in order to increase the variability of the response of Intermittent Control. Figure 5.2 shows the mean sampling compared to sampling from the distribution.

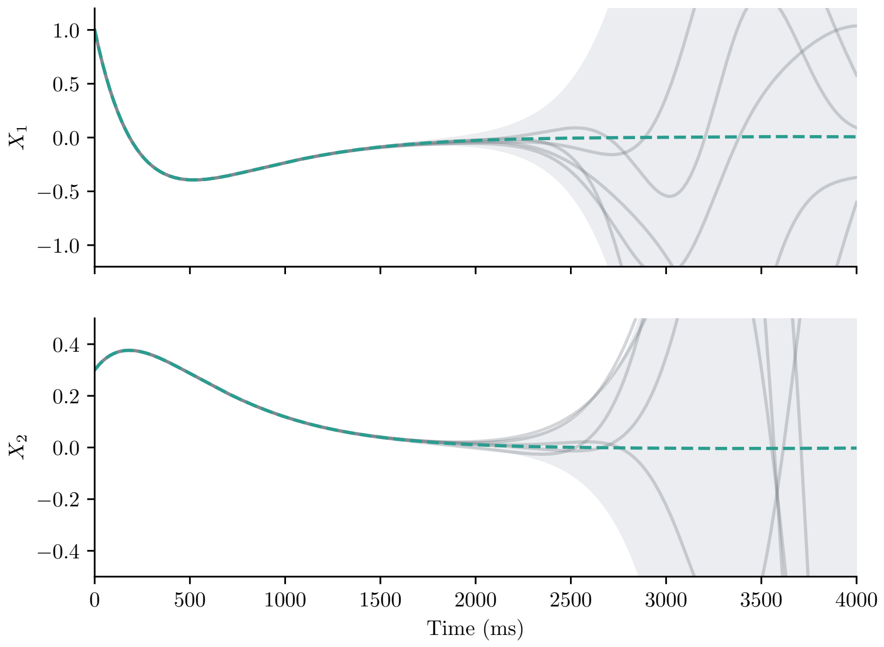

Figure 5.2: Example of sampling from a distribution. The dash line is the mean of the distribution. The shaded area is the standard deviation of the prediction. Each gray line represents a sample from the distribution.

It is necessary to fix a maximum open-loop interval in the Intermittent Control Framework to implement this in order to put an upper limit on the calculation of the distribution. The GP being the representation of the dynamical system, it is necessary to repeat the prediction one step at a time as the next point to query is the output of the previous prediction.

5.1.1.2 GP based Predictor

Similar to the GP based Hold, it is possible to extend the GP to the predictor. The idea is identical with the small modification of a multi-step ahead prediction instead of a single-step.

At the time of an event, the input of the predicted states are reset to the system states $x(t)$.

$$x_p = x(t) \tag{5.9}$$

Then, depending on the sampling time of the data during training of the GP, it is possible to compensate the delay as:

$$x_p = \mathscr{N}(x_p(t)|0, K_{pred}) \tag{5.10}$$

The Equation (5.10) has to be repeated to compensate for any delay present in the controller. In the case of the predictor, the GP has to represent the open-loop dynamics of the system. In this case, the control input given to the GP is fixed at zero. Similarly to the Hold, the output of the distribution can be based only on the mean or as a sample of $\mathscr{N}(\mu, \sigma)$.

5.1.1.3 Prediction uncertainties

Moving from a deterministic approach such as System Matched Hold, Gaussian processes brings new information from the Hold block. There is now an uncertainty linked to the prediction. This can be used to make the Intermittent Control smarter in two different way:

- If the uncertainties of a prediction are rapidly growing over time, this might indicate that the underlying GP could benefit from retraining. This would help deciding if the GP needs to be retrained.
- If the uncertainties are high but the mean prediction is accurate, it could be used as a safety triggering. This will reset the uncertainties to zeros and the GP will be aware of the current states of the plant.

5.1.2 Evaluation criteria

As presented above, Gaussian Processes are not a direct representation of the system matrices. Instead, hyperparameters are estimated to represent the link between input and output of the system. Hence, it is not possible to use regular control techniques to assess how close the GP representation is from the actual system by comparing matrices directly.

In order to evaluate the goodness of the GP (see Equation (5.11)), two additional measures are computed, both in time and frequency domain, in addition to those presented in Section 4.1.2. The two measures are based on simulations ran outside the Intermittent Control context, directly computed with the GP. The GP is paired with a continuous controller in order to simulate the model of the plant under the influence of a state feedback controller with a multisine disturbance signal. The output generated by the GP is assessed against the true output of the plant under the same conditions. The fit between the two dataset is compute using the following equation:

$$Fit = 100 \times \frac{1 - \|S_p - \hat{S}\|}{\|S_p - \bar{\hat{S}}\|} \tag{5.11}$$

where S_p is the signal generated by the plant and $\hat{S}$ is the signal generated by the GP. The response is investigated in both time and frequency domain. Assessing the GP in these conditions is similar to investigate the model accuracy inside the Hold as it is modeling the closed-loop dynamics of the plant.

5.2 Results

As summarized in Figure 5.3, GP can be implemented in different ways such as Single Task GP (STGP) or Multi Task GP (MTGP). In this section, the Hold block in Intermittent Control is replaced by Gaussian Processes. In the first results section, the focus is on Single Task GP, covering the use of single step and multi step ahead prediction, as well as Uncertainty Prediction. The computation needed for GP is also being considered and some options are layed out and tested to improve computation time. Following this, Intermittent Control Hold is using Multi Task GP. The main driver for this implementation is to have a single GP that can represent all the states of the system, hence removing complexity in the modelling of multiple separated GPs. Finally, online retraining is looked at in the case of Single Task GP in order to improve the accuracy of the GP over time, using data coming from the simulation itself. The simulations presented in this Chapter are applied to the Single Inverted Pendulum (see Appendix A.

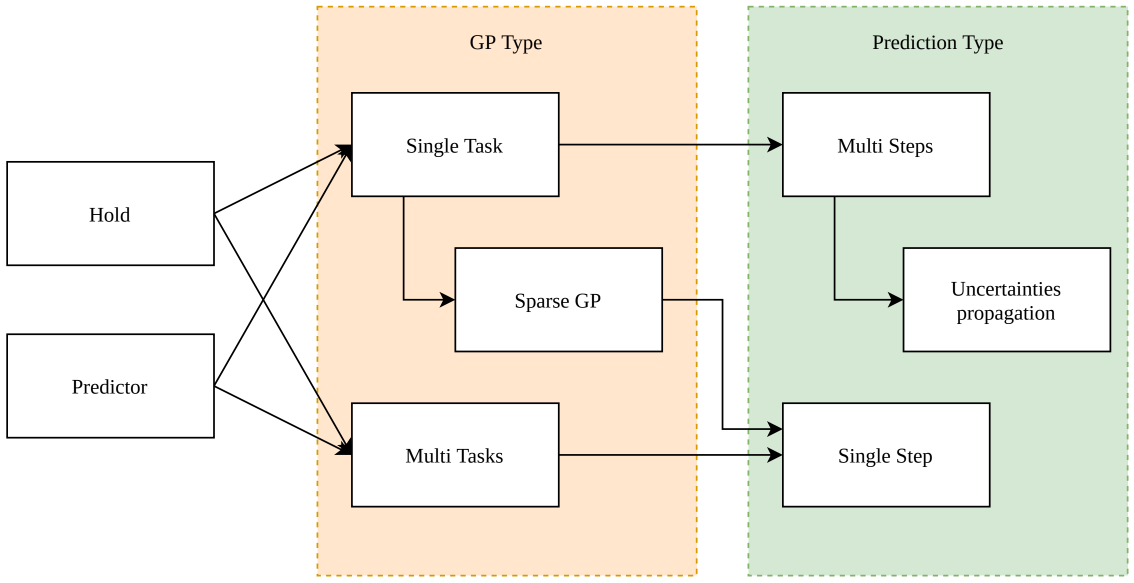

Figure 5.3: Overview of GP Hold (GPH) implementation within the IC framework.

Even though it is possible to replace the predictor with Gaussian processes as presented in Section 5.1.1.2, all following simulations are not using any delay, hence the use of a predictor is not required. In addition, in these simulations, the observer (see section 3.2.2) is not used as it is assuming that all states of the system can be measured.

5.2.1 Single Task GP

This section is focusing on the response of the Intermittent Controller using a Single Task Gaussian Process as the Hold. First, the output of the Hold will use the mean of the prediction of the GP assuming that the previous iteration is not coming from a probabilistic process. Then, Uncertainty Propagation (UP) is used to consider the probability associated with the previous prediction during an open-loop interval. Lastly, the optimisation of the GP computing is investigated, in order to improve the possibility of using GP in a real time settings.

5.2.1.1 Full GP

As explained in Section 5.1.1.1, each state of the system is represented by its own GP. In the case of the pendulum, two GPs are being trained using each state as the output and are using the same set of inputs ($\dot{\boldsymbol{\theta}}$, $\boldsymbol{\theta}$, u). Figure 5.4 is summarizing the implementation of Single Task GP within the Intermittent Controller.

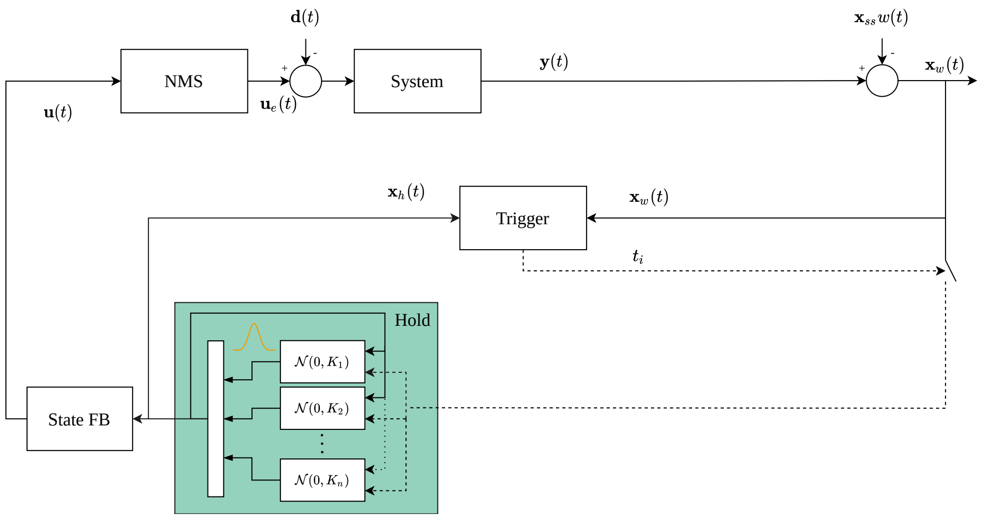

Figure 5.4: Intermittent Control block diagram with detailed of Gaussian Process Hold when using Single Task GP.

The mean of the GP is used as the states predicted by the Hold. This approach can be seen as similar to a Kalman Filter. At each iteration, each GP is using the data from the previous iteration as input similarly to SMH during the open-loop, and it is using the measure states as input when IC is triggering. This implementation is the default whenever GP Hold (GPH) is used, except in Section 5.2.1.2, where Multi-Task GP is used. In this section, Single Task is used in different operation modes. Single step is looked at first, then the focus is moved toward multi step ahead prediction followed by use of Uncertainties Propagation.

Single Step prediction

As an initial simulation, presented in figure 5.5, IC is using a System Matched Hold from 0 to 20 seconds plus next event. This first part of the simulation is then passed to the GP optimization routine which randomly selected 20 points from the first 20 seconds of the simulation in order to train the two GPs (one per state). Then, for the rest of the simulation, the Intermittent Controller is using the Gaussian Process based Hold (GPH). It is necessary to train the GPs before applying it in IC using system data, hence Intermittent Control has been used to generate it. Even though the output generated by the Intermittent Controller appears to be periodic, this is only due to the triggering mechanism. The kernel function used is a combination of the Square Exponential function with the added Noise covariance function. Based on the range of motion of the pendulum, linear covariance function could have been used as well.

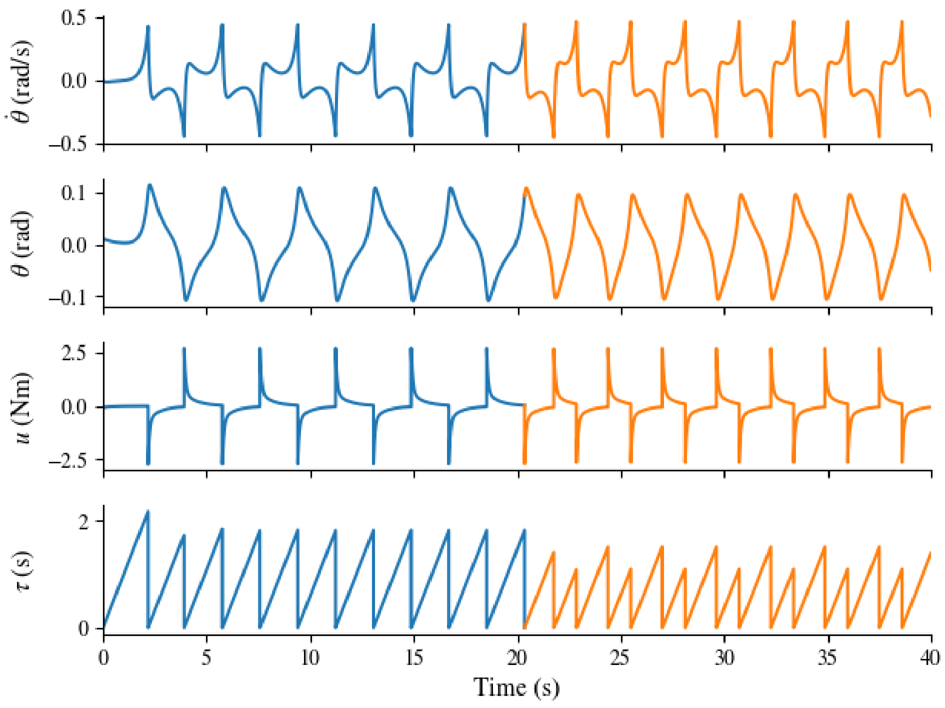

Figure 5.5: Intermittent Control applied to Single inverted Pendulum. SMH is used from 0 to 20 seconds (+ next event) (blue line) and GP based Hold is used after (orange line). GP is trained using SMH data and is using single step ahead prediction.

As explained in Section 5.1.2, multiple measures are computed to assess the Gaussian processes estimation, outside and within the Intermittent Controller. These measures are presented in the following table, showing the differences between SMH and GPH. This computation is discarding the transient from the simulation as it is not present for both holds.

	$\dot{\theta}_{RMSE}$ (rad/s)	θ_{RMSE} (rad)	u_{MEAN} (Nm)	τ_{MEAN} (s)	τ_{STD}	Ψ
SMH	0.141	0.054	-0.002	1.85	0.107	0.04
GPH	0.177	0.059	-0.023	1.29	0.206	0.31

Table 5.1: Assessment of SMH vs GPH. $\dot{\theta}_{RMSE}$ and θ_{RMSE} are the Root Mean Square Error for $\dot{\theta}$ and θ respectively. u_{MEAN} is the mean of the control input. τ_{MEAN} and τ_{STD} are the mean and standard deviation of the open-loop interval. Ψ is the variability of the open-loop interval (see Section 4.1.2).

As presented in Table 5.1, System-Matched Hold and GP based Hold are similar when comparing the RMSE of both states. However, the mean of the control input is increased as well as the variability. The mean of the open-loop interval is decreased when using the GPH, which can be seen in Figure 5.5. In addition to these measures, the goodness of the GP (see Equation (5.11)) is also assessed outside Intermittent control and it is presented in Figure 5.6. This estimation is giving 99.59% fit for the multisine signal and 99.13% fit regarding the transfer function estimation between the state space representation of the system and the GP in the frequency domain.

As presented in Figure 5.5, the GPs are trained using the first part of the simulation using an Intermittent Controller with SMH. GP is very reliable in estimating the correct system dynamics. However, this is using pre-designed intermittent controller with a System matched Hold trajectory. Pre-recorded trajectory from the plant can be used for training the GP in order to start the simulation with a GP based Hold. In order to get these

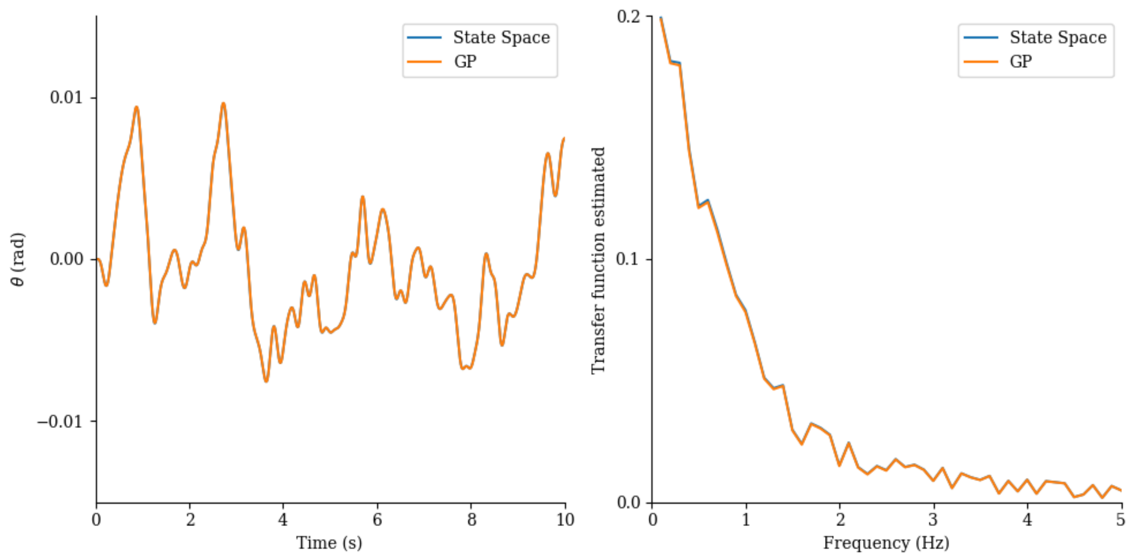

Figure 5.6: Assessment of goodness of the GP. Left plot: response of the GP compared to the state space representation of the plant. Right plot: Frequency response of both State space and GP from the multisine input. Fit are respectively 99.59% and 99.13%

pre-recorded trajectories, a combination of a state feedback controller is used, to ensure stabilisation of the plant in a unstable location in conjunction with multisine disturbance with frequency ranging from 0.1Hz to 5Hz, with a step size of 0.1Hz. The Figure 5.7 is showing the evolution of the states under the previously stated control input.

As presented in Figure 5.7, the data is centered around 0 and has some excitation due to the multisine disturbance. However, in order to keep the system stable, it is necessary to add a state feedback controller to stabilize the plant around the unstable location. Due to the presence of the controller, the system dynamics estimated by the GP are representing the closed-loop behavior of the plant. This model can be used directly in the Hold, replacing the A_c matrix. In addition, the number of points used for training the GPs can be arbitrarily chosen: this is having a big impact on the overall performance of the GPs, and impacts the performances of the controller. However, it is possible to change this value in order to get more or less variability.

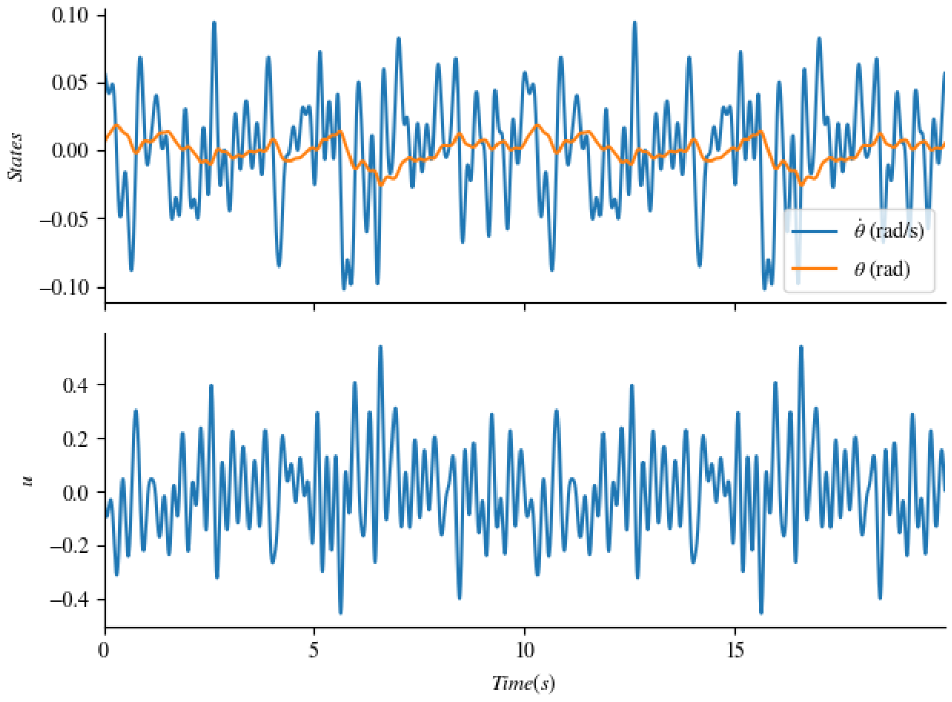

Figure 5.7: Example of training data passed to the GP for training. The top row shows the Single Inverted Pendulum's states during the simulation under the control input applied (bottom row). The control input is composed of a state feedback controller in addition of a multisine disturbance signal.

Pre-recorded data from the plant can be used for training GPs in order to start the simulation with a GP based Hold. Training data for the GPs has been generated using a multisine disturbance applied to the system. In order to keep the system stable, it is necessary to add a state feedback controller. Hence, the system estimated by the GP will represent the close loop dynamics of the plant. This model can be used as is directly in the hold, replacing the A_c matrix. For each simulation, the training data is randomly generated using a range of frequency between 0.1Hz and 5Hz. In this specific scenario, the GPs are representing the closed-loop dynamics of the system, hence it is not necessary to pass a control input to the GP when being used in the hold.

In order to assess the reliability of the GPs using a multisine as training excitation signal, multiple conditions have been tested: influence of the number of points used for training as well as using different Δ_{OL} and Threshold coefficient inside IC. Figure 5.8 is showing the impact of the number of point used for training as well as the impact of Δ_{OL} and Threshold coefficients.

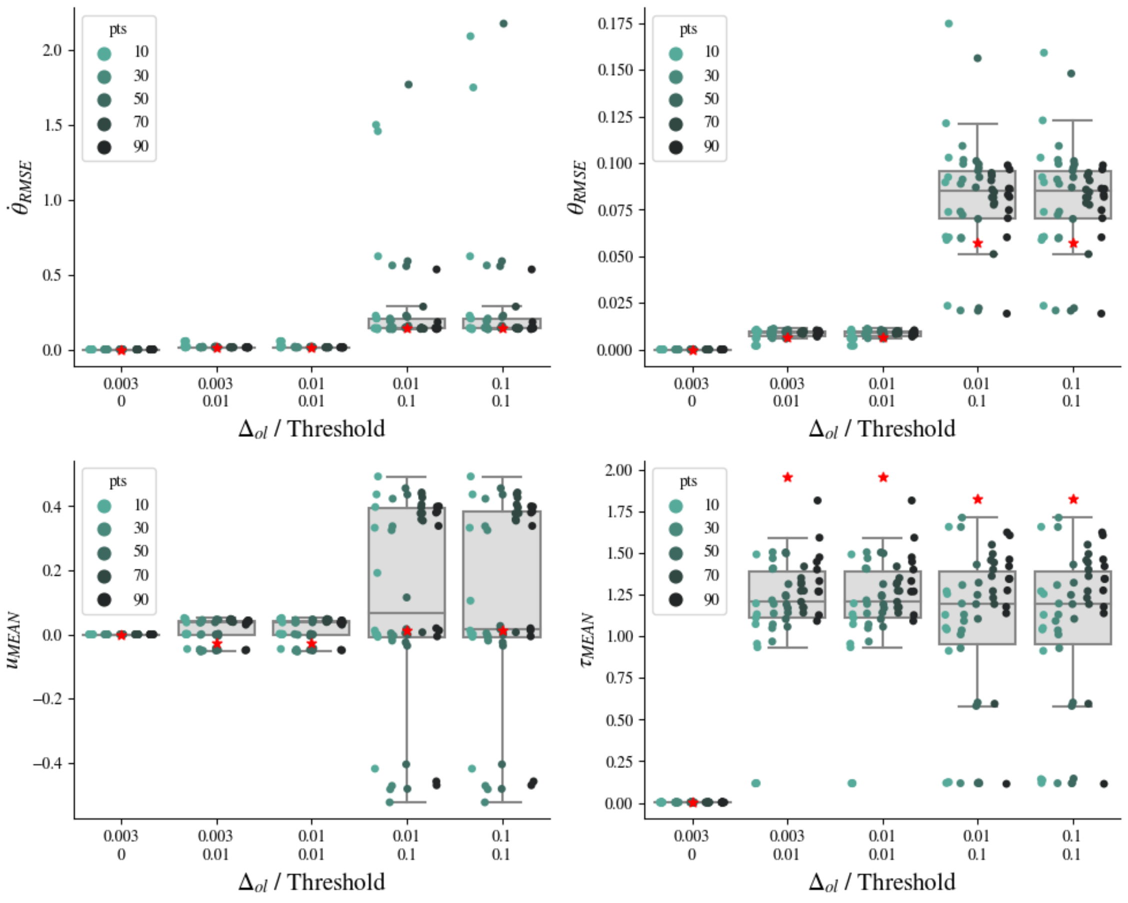

Figure 5.8: Summary of GP trained using multisine control input. The first row is showing the RMSE for (a) $\dot{\boldsymbol{\theta}}$ and (b) $\boldsymbol{\theta}$. The second row is (c) the mean of the control input and (d) the mean of $\boldsymbol{\tau}$, the open-loop interval. The red star is representing the output from a SMH based Intermittent Controller. The box plot is showing the output from multiple GP based IC. Each shade of green is representing the amount of points used for the GP training.

In the case of clock-driven Intermittent control (when the threshold is set to zero), GP behaviour is identical to the SMH case. In addition, setting $\Delta_{OL} = 0.003s$ is forcing the Intermittent to act very similarly to a continuous controller.

Increasing the threshold from 0 to 0.01, hence having event-driven Intermittent Controller is allowing bigger open-loop intervals as shown in Figure 5.8, quadrant (d). However, even if the RMSE of $\dot{\theta}$ and θ are closely matching with the SMH, the open-loop interval is only sligtly above half of the one coming from the SMH simulation (1.25sec compared to 2sec).

Another interesting finding is the influence of the number of points used for training. For all the different values of Δ_{OL} and Threshold, 70 points for the GPs is giving the lowest spread in all the different simulations. Comparing to 10 or 90 points, some of the simulations are showing some outliers, forcing the Intermittent Controller to trigger at the minimum open-loop interval (in the case of $\Delta_{OL} = 0.1$).

In addition to the RMSE and mean value shown above, the variability of the pendulum angle θ has been assessed using the cross-correlation between two following open-loop intervals. The main idea of using GP is to introduce variability in the simulation without the need of additional disturbance such as noise. Figure 5.9 is showing the impact of the number of point used for the training of the GP with the overall variability of the simulation.

Focusing on the clock-driven case, where the Threshold is 0, there is very low variability. This is due to the Intermittent Controller acting very closely to a Continuous Control. Increasing the Threshold coefficient is giving more freedom for the GP to generate a non-ideal state trajectory. This is emphasize when the number of points use for the initial training is particularly low, such as the 10 and 30 points case. Increasing the number of points is cancelling the opportunities for the GP to have some variability. However, as presented in the figure (figure above with RMSE), adding more point than necessary could then reduce the overall performance of the Intermittent Controller, such as getting in a minimum open-loop interval regime.

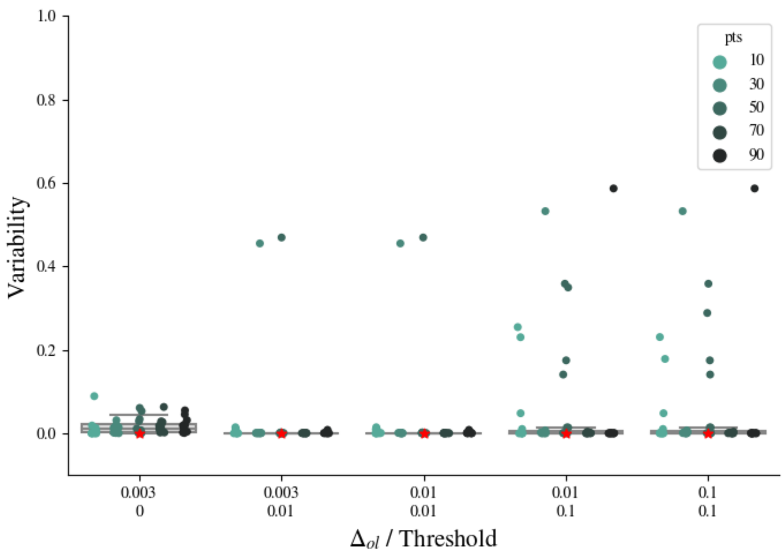

Figure 5.9: Assessment of the variability when using a GP based Hold in Intermittent Control. The x-axis is representing different set of parameters inside IC: Δ_{OL} and Threshold. The y-axis is showing the variability computed from the cross correlation between two consecutive open-loop interval.

Multi Step ahead prediction

In previous results, GPs are used similarly to the System Matched Hold, computing one step at a time. In this section, GPs are pre-computing the next open-loop interval when the Intermittent Controller is closing the loop. This is an intermediate step between single step ahead prediction and the use of uncertainties propagation described in the following section. By having future trajectories computed at triggering, it is necessary to fix a maximum open-loop interval in advance to set a prediction horizon. Figure 5.10 is showing the impact of using multi step ahead prediction compared to single step prediction, by changing prediction mode during the simulation.

Using multi step prediction is not changing the behaviour seen in the single step prediction as long as the maximum open-loop interval is larger than the Δ_{OL} of the simulation. This is simply due to the prediction being made ahead of time without having any impact on the algorithm used. However, this approach can be quite limiting in a real-time

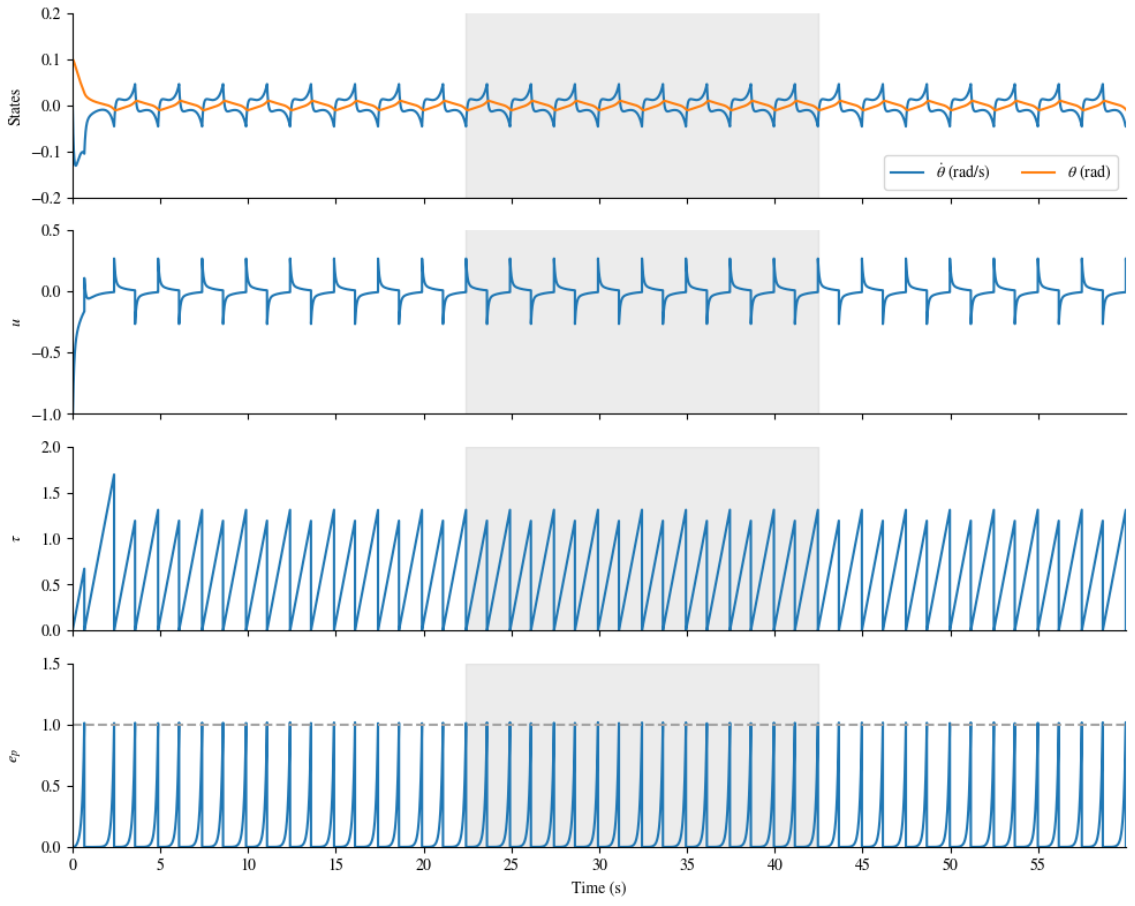

Figure 5.10: Intermittent Control switching between single-step prediction and multi-step prediction. The grey box is showing when the multi-step ahead prediction is enabled. The top row represents the state of the plant: $\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$. The second row is the control input generated by the controller. The third row is the open-loop interval. Last row is the error between the plant and the hold states.

implementation as the next horizon is computed when an event occurred. This is similar to Model Predictive Control, however, the prediction is coming from a Gaussian Process and it requires higher computational effort. In addition, forcing a maximum open-loop interval can force an event if this is reached, even if it is not required.

Using Uncertainties Propagation

Previously, uncertainties have not been used as part of the computation of the next state prediction by the GP. It was assumed that the state passed to the GP has not uncertainties associated with it. However, in the case of dynamical system, where the next state is dependant on the previous states, skipping these uncertainties results in abnormally low uncertainties for the prediction of the GP. Even though only the mean of the output is used as the output of the prediction, using the uncertainties propagation is impacting the computation of the mean.

In this section, the prediction of the next state is using the current states as well as the associated uncertainties if it is coming from a previous prediction. To understand the impact of the uncertainties propagation, the following simulation is starting with GPs using multi-step ahead prediction, then, uncertainties propagation is enabled around 20 seconds. Uncertainties propagation is then disabled again to verify that the controller is going back to its original behaviour. The GPs used for this simulation are the same through the entire simulation, only the prediction mode is changing.

Presented in Figure 5.11, the controller is switching between two modes: with and without uncertainties propagation enable, represented by the grey box. Multiple observations can be made from this simulation. Firstly, the behaviour of the Intermittent Controller is changing when the uncertainty propagation is enabled. The states of the system as well as the control input is switching from the symmetric repeated behaviour to one with more variability. Furthermore, the open-loop interval is slightly increasing for some events during the uncertainties propagation.

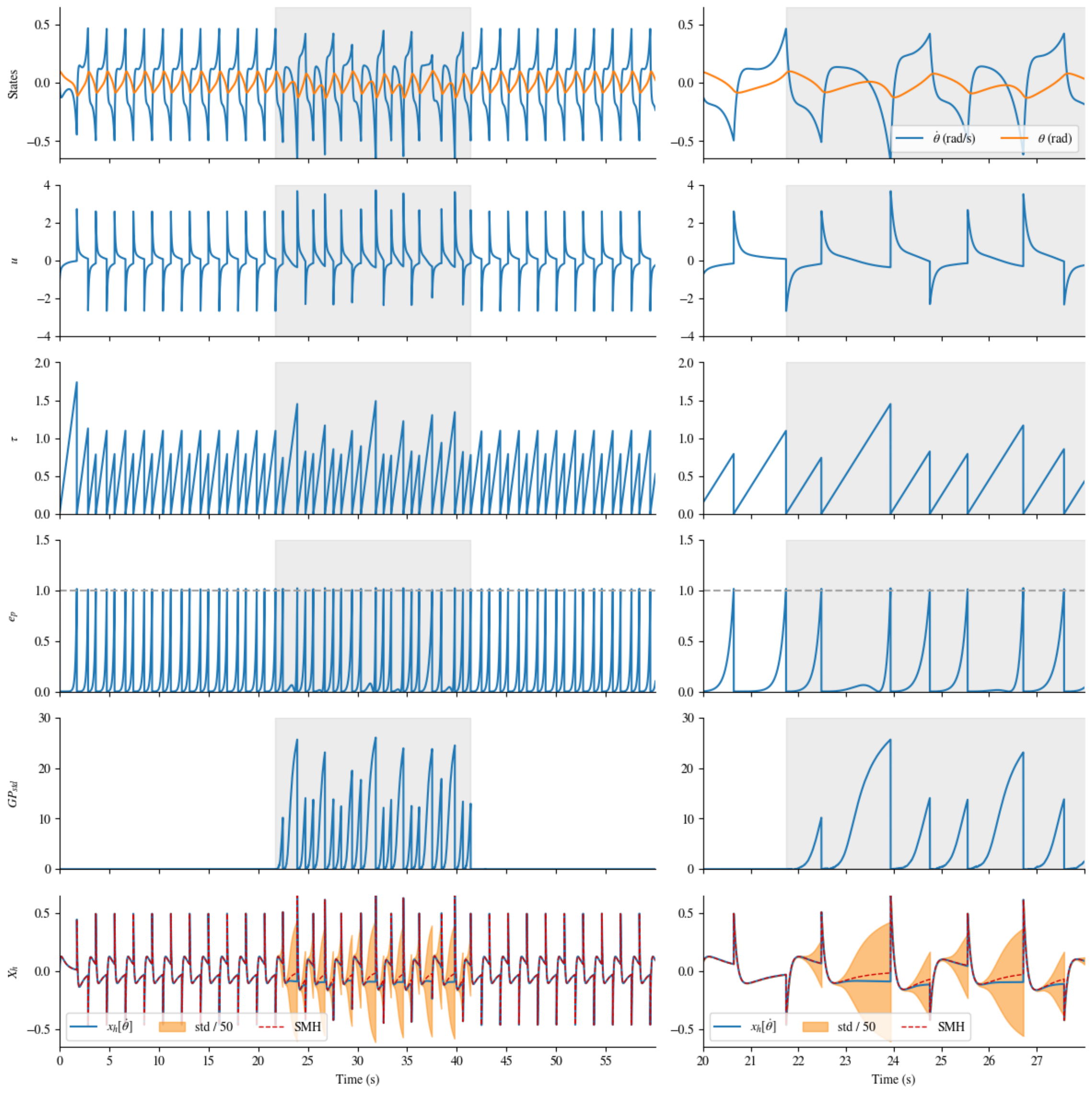

Figure 5.11: Intermittent Control switching between regular prediction and uncertainties propagation prediction. The grey box is showing when the uncertainties propagation is enabled. The top row represents the state of the plant: $\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$. The second row is the control input generated by the controller. The third row is the open-loop interval. Fourth row is the prediction error. Fifth row is the standard deviation generated by the velocity GP. Last row is the pendulum velocity generated by the Hold. The right column is a zoomed version between 20 seconds and 28 seconds.

Another observation is related to the standard deviation coming from the GPs. When GP is not using uncertainties propagation, the standard deviation is abnormally low due to the fact that the GP is not aware that the input is coming from a previous prediction, hence the following prediction is considering the prediction as the first one of an open-loop, when readings are made from the plant. However, when enabling the uncertainties propagation mode, the standard deviation is now increasing in a more natural way. As the open-loop is increasing, the standard deviation is growing accordingly. As presented in the last row of Figure 5.11, the hold states are compared to the hold states coming from a System-Matched Hold. As the uncertainties are increasing, the hold states estimated by the GP are not matching the SMH states anymore. The standard deviation is divided by 50 in the figure in order to be able to read the hold states values. The amplitude of the standard deviation is impacted by the small number of points used for the training of the GP as well as the length of the prediction. In this simulation, the simulation sampling time is $DT = 1ms$ and the prediction horizon is around 1.3 seconds, equivalent to 1300 steps prediction.

Now that the standard deviation of the GPs is outputting reliable values, it is possible to use it as another triggering mechanism within the Intermittent Trigger Block. The following simulation is using a combination of triggering. In addition to the standard triggering relying on the difference between the Hold states and the system states, another threshold is based on the amplitude of the standard deviation of the output of the GP. This can be seen as a triggering when the GP model is becoming too uncertain of the prediction. In this simulation, the standard deviation threshold is only applied to the GP predicting the velocity.

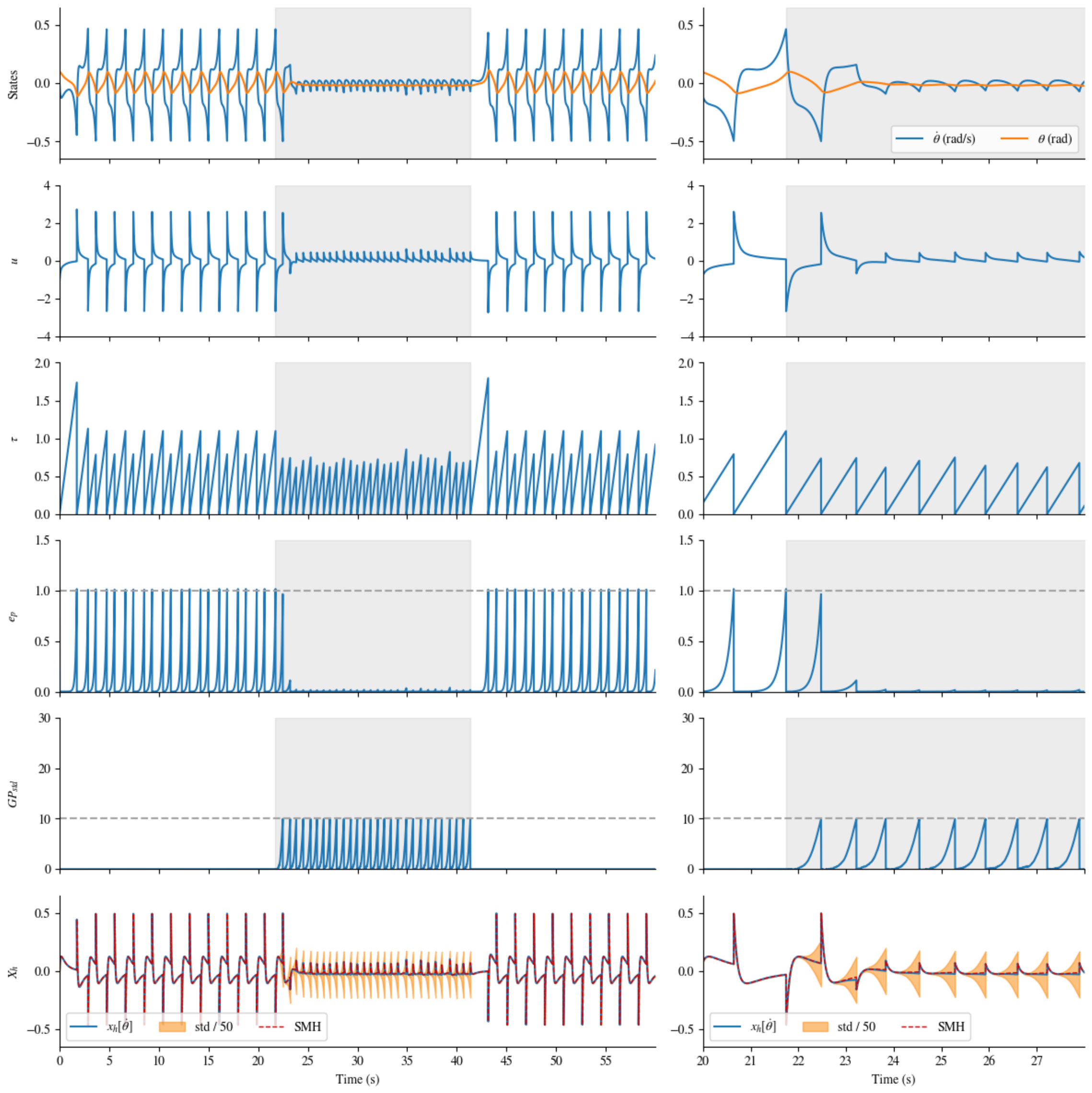

Figure 5.12: Intermittent Control switching between regular prediction and uncertainties propagation prediction. The grey box is showing when the uncertainties propagation is enabled. Triggering is modified to trigger on both the prediction error e_p and GP_{std}. The top row represents the state of the plant: $\dot{\theta}$ and θ. The second row is the control input generated by the controller. The third row is the open-loop interval. Fourth row is the prediction error. Fifth row is the standard deviation generated by the velocity GP. Last row is the pendulum velocity generated by the Hold. The right column is a zoomed version between 20 seconds and 28 seconds.

Figure 5.12 is showing the impact of introducing a new triggering mechanism based on the uncertainties of the GP prediction. Similarly to Figure 5.11, the beginning of the simulation is without the uncertainties propagation enabled. Then, uncertainties propagation is switched on, which changes the response of the controller. Focusing on original triggering mechanism, the prediction error e_p is not reaching the threshold when uncertainties propagation is enabled. This is due to the standard deviation of the GP exceeding its own threshold, hence triggering earlier.

Even though these simulations are bringing novelty into the behaviour of the control input due to the newly introduced triggering mechanism, it is important to notice two main drawbacks. First, the computation time of uncertainties propagation is longer due to the algorithm used. Using single step ahead prediction is taking roughly a similar amount of time as the simulation. When using uncertainties propagation, the simulation is taking 10 times the simulation length: each minute of simulation is approximately taking 10 minutes to run. The algorithm used is the one developed by Deisenroth and Rasmussen 2011, and has been translated from Matlab to Python. In addition, with low number of points GP, the computation of the standard deviation is not stable due to the non convergence of the least square algorithm when solving Equation (3.60).

5.2.1.2 Optimization of GP computing

As presented above, GP can be beneficial in the intermittent control framework to bring some variability into the control signal and to get towards a data-driven Intermittent controller. However, it is important to notice the computational burden of this technique, especially when the number of points is getting bigger. Whilst this is not too much of a concern in the case of simulations, it is something to consider in the case of real-time implementation.

To improve this aspect, multiple approaches have been implemented to improve the computation issue. Two possible techniques are presented in this section: using Sparse GP to limit the number of points used by the GP and using known information about the system to remove the need for extra input states of the GP by using the derivative of the GP output for velocity trajectories.

Sparse GP

In this section, sparse GP is used instead of the traditional full GP implementation in the Hold and applied to the Cartpole (see Appendix B). Similarly to the Full GP implementation, each state is having its own Sparse GP representation. In this section, the behaviour of the Intermittent Controller is assessed as the number of points removed from the Full GP to the Sparse GP is increasing. Figure 5.13 is comparing the impact of the number of inducing points used by the sparse GP from the Full GP.

The first half of the simulation is using a SMH then the hold is swapped to a GP hold. The first column is using a full GP while the three following columns are based on the sparse GP of the full GP. Whilst the dynamics between the Full GP and the Sparse GP using 30 and 10 points are similar, the dynamics change quite substantially when the sparse GP is using only 5 inducing points.

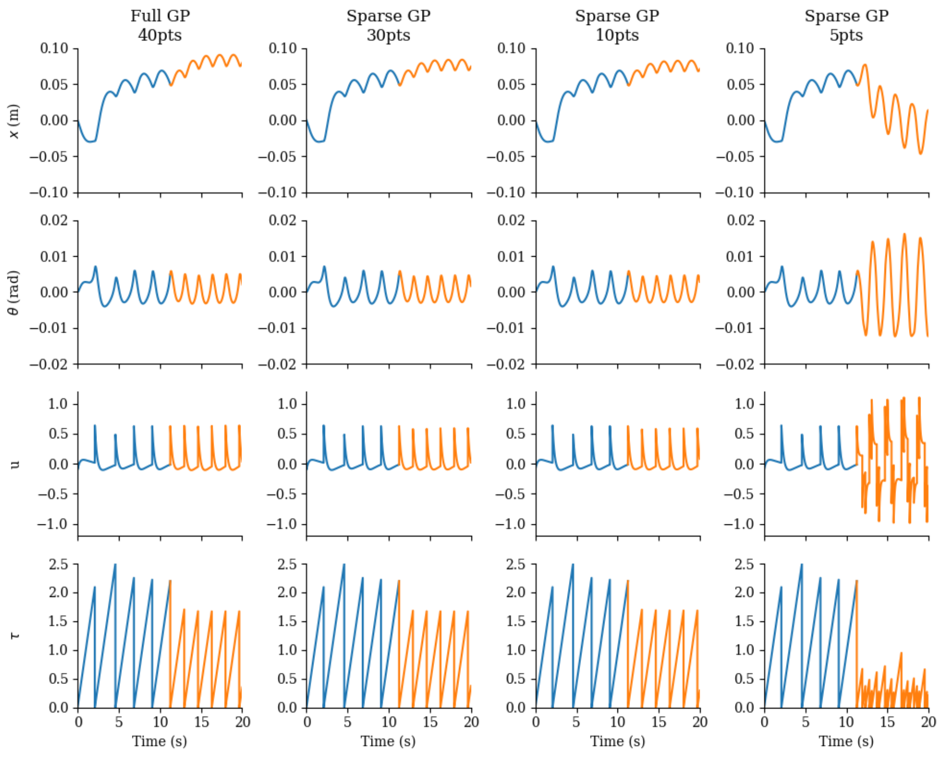

Figure 5.13: Comparison of different sparse inducing point number applied to the cart pole following a SMH simulation. The control swap from SMH to GP at the next event after 10s and the training is based on the multi sine training data with state feedback. (c.f: Figure 5.7)

To understand the differences, multiple measures are computed per different GP cases and are summarized in Table 5.2. First, the RMSE of the cart position is higher when GP is used. However, this is due to the transient response being included in the SMH case. By sparsing the GP from 40 points to 30 and 10, the RMSE of the cart as well as the angle of the pole are not changed. Similar results can be observed in the mean of the control input, in the open-loop interval, for the standard deviation of the latter, and for the variability coefficient. An important change can be observed when the number of inducing points in the GP reaches 5 data points. The behaviour of the controller is now different from all previous cases. This is due to the GP's inability to accurately represent the system dynamics with a small number of points. The total time to run the simulation t_{sim} decreases by using sparse GP compared to the Full GP implementation (from 105 ± 5 seconds to 82 ± 9). However, this is still fifteen times higher than the SMH approach.

Hold Type	x_{RMSE} (rad/s)	θ_{RMSE} (rad)	u_{MEAN} (Nm)	τ_{MEAN} (s)	τ_{STD}	Ψ	t_{sim} (s)
SMH	0.0461	0.002921	0.001112	2.266	0.1481	0.1795	5 ± 0
Full GP	0.0781	0.002778	0.010355	1.677	0.0131	0.0082	105 ± 5
Sparse 30	0.0737	0.002618	0.010056	1.672	0.0045	0.0051	82 ± 9
Sparse 10	0.0727	0.002617	0.009588	1.688	0.0053	0.0059	84 ± 11
Sparse 5	0.0353	0.009466	0.004407	0.449	0.1910	0.3836	76 ± 10

Table 5.2: Assessment of SMH vs GPH. x_{RMSE} and θ_{RMSE} are the Root Mean Square Error for the position of the cart and the angle of the pendulum θ respectively. u_{MEAN} is the mean of the control input. τ_{MEAN} and τ_{STD} are the mean and standard deviation of the open-loop interval. Ψ is the variability of the open-loop interval (see Section 4.1.2). t_{sim} is the simulation time compared across 10 simulations.

Introducing system knowledge

Another approach is to introduce system knowledge to reduce the complexity of the GP-based Hold. Even though the main focus is to rely on a full data-driven approach, dynamical control is often using combined states such as velocity and position, thus these two states are directly related. Focusing on the Single Inverted Pendulum, the two states used for control are θ and $\dot{\theta}$. In previous simulations, each state is represented by its own independent GP. This involves retraining per state, as well as the need to invert the kernel matrices as many times as states. In this section, the idea is to use system knowledge to simplify the representation of the Hold as well as improving computation time. The following approach can be applied to the Single Inverted Pendulum: a GP model is predicting the angle θ, then a simple derivative function ($\frac{x_i - x_{i-1}}{dt}$) is used to compute $\dot{\theta}$.

To assess the computational effort with this approach compared to the traditional ”one GP per state” approach, 10 simulations of 20 seconds each with $dt = 1ms$ were run and timed for both cases. The total time of these 10 simulations is then converted to an iterations per second value (10 $\times$ 20 seconds / dt / total time) to obtain an average of iterations per second. Whilst the traditional case did 1035 ± 40 iterations/second, the derivative case achieved 914 ± 18 iter./s. It is more computationally intensive to run the derivative case for online state estimation in the Hold. This is due to two main factors.

In the ”one GP per state” case, the GP estimates the velocity at time $t+dt$, however, in the derivative version, the velocity computed using the angle value at $t+dt$ is the velocity at t. Hence, it is necessary to iterate through the GP representing the angle twice, using the previous computed angle as an input. The second factor is the state decomposition needs to be done twice in the derivative case, whilst this is done for both GP at once in the other case. One positive improvement is the simplification of the optimisation of the GP, as this only required to optimise a single GP. The derivative case can be seen as a more accurate representation of the system (assuming that the angle GP is accurate) due to the fact that both states are properly evolving in a physical sense. This is not assured in the ”one GP per state” approach as both states are computed independently.

Multi Task GP

Whilst the previous approach is using system knowledge to simplify the representation of the system using GP, this approach can simplify it a step further. Focusing on systems where states cannot be inferred from another one, the multi-task approach presented by Bonilla et al. 2007 can be used. The multi-task approach is using an inter-task matrix to model the system dynamic with a single kernel. With some matrix simplifications, it is possible to keep the number of hyper-parameters below the number needed for the ”one GP per state” approach. Figure 5.14 shows the difference is the implementation in comparison to Figure 5.4.

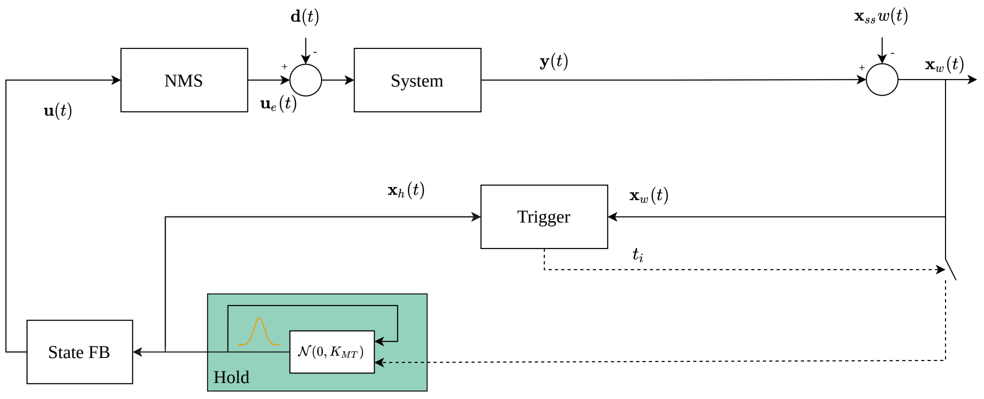

Figure 5.14: Intermittent Control block diagram with detailed of Gaussian Process Hold when using Multi Task GP.

The following simulation is based on a GPH using the multi-task implementation. Similarly to the single task GP, the training data is based on a close loop representation of the plant, as this is using system data which has been generated with the state feedback controller with a multisine disturbance as presented in Figure 5.7.

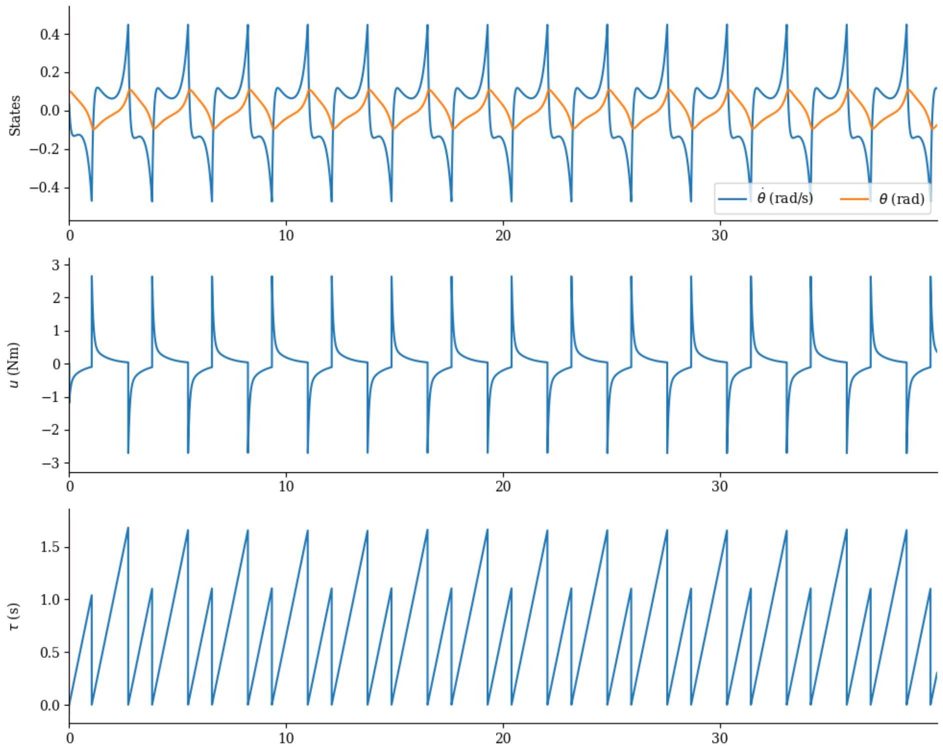

Figure 5.15: Intermittent Control applied to Single inverted Pendulum with multi-task GPH. Top row are the states of the Single Inverted Pendulum. The second row is the control input generated by the Intermittent Controller. The bottom row is the open-loop interval.

In Figure 5.15, the multi-task GPH is using 60 points for training. The fit in the time domain is 96.51% and 99.0% in the frequency domain. Whilst this implementation is able to stabilise the pendulum, getting a good representation of the system's dynamic using multi-task GPs can be more difficult than single-task GP. More training points are necessary to reach good performance compared to the single task GP case, however, this does not slow down the execution significantly. Table 5.3 is summarises the iterations per second value based on the number of points used for the training:

Training points	Iteration per second
10	898 ± 3 it/s
20	892 ± 3 it/s
40	893 ± 2 it/s
60	890 ± 2 it/s
80	889 ± 2 it/s
100	874 ± 23 it/s

Table 5.3: Iteration speed for Intermittent Control using Multi-task GPH.

5.2.2 Online retraining

As presented before, a GP based hold is designed at the start of the simulation and it is not getting updated through the simulation. However, it is possible to add more training points to the GP, needing a new optimisation when points are added. This approach is similar to the Data Informativity (Section 4.2.2) as the goal is to improve the response of the GP based Hold.

Similarly to Figure 5.5, the GPs are retrained using previous data generated by the controller itself. However, in this case, the data is originally coming from a GP based Hold instead of being a SMH.

Depending on the origin of the training data, the GP might be representing the closed-loop dynamics of the plant as a state feedback controller is here to keep the Single Inverted Pendulum stable even with the multi sine applied as a disturbance. However, the training data coming from the Intermittent Controller will be estimating the open-loop dynamics of the Single Inverted Pendulum (see section 4.2.2.2).

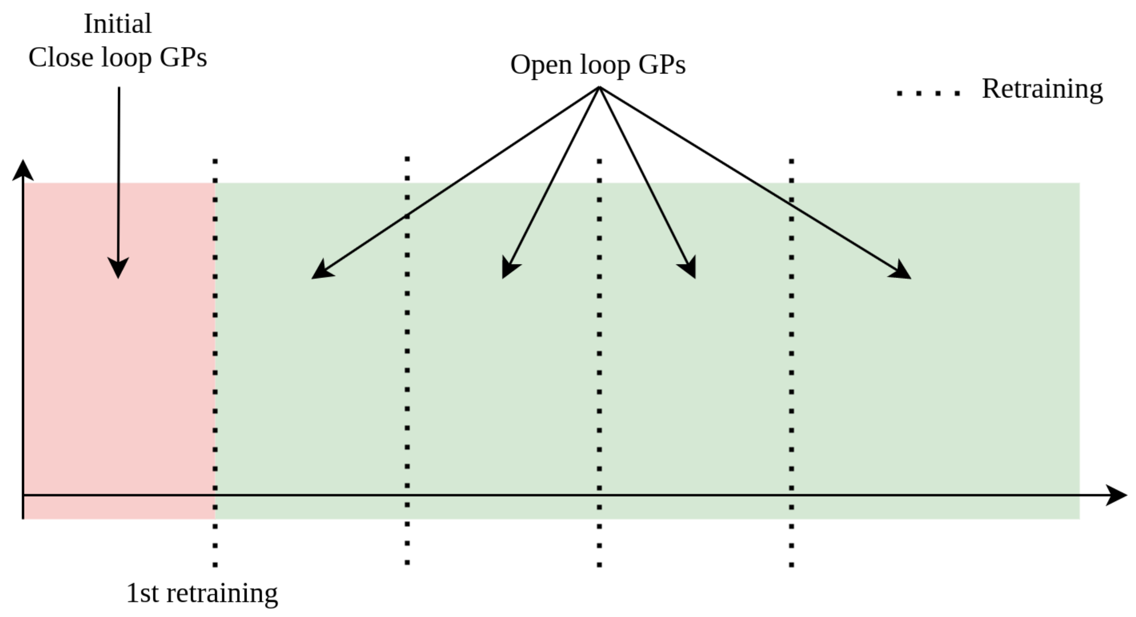

Figure 5.16: Diagram representation of the type of system the GP is modelling.

Following the first retraining, all data from the initial GP is dropped. The training data for the GP cannot be mixed between two different systems. In addition, in the case of the GP being a representation of the open-loop dynamics, it is necessary to pass a state feedback control input $(-kX)$ similarly to the SMH implementation.

Initially, the plant is not changing, and the GP based hold is trying to reach the correct representation of the dynamics of the plant. Then, similarly to the test case used in the Data Informativity framework, the system is changing from an initial length to another one, focusing on assessing the capability of the GP to learn the new system. Then the case of the system changing progressively every 10 seconds is also looked at.

5.2.2.1 Model improvement over time

In this first section, the Single Inverted Pendulum is not changing overtime. The main focus is for the GP to learn the correct dynamics of the plant after multiple retraining. Similarly to the initial training, it is up to the user to decide how many points from the trajectory is used for retraining. However, as presented in the Section 5.2.1.2, adding more points for optimising the GP is making computation slower, for both the optimisation as well as predicting the following state.

Moreover, it is also up to the user to decide when the optimisation is taking place to fit in the Intermittent Control framework. Similarly to the Data Informativity implementation, the GP retraining is performed when the controller is closing the loop. With this implementation, the controller is not changing during an open-loop interval.

Firstly, the influence of the number of points used for retraining is investigated as well as the time used between each retraining. Figure 5.17 shows the evaluation criteria measures in order to assess the optimal combination of points used for training and retraining time of the GPs.

Retraining the GP using online data can improve the open-loop interval. It is also decreasing the spread of the RMSE for both state as well as the mean of the control input. However, keep adding points to the existing GPs is slowing the simulation as well as the optimisation. It is important to notice that adding points to the GP is not always beneficial, as results show when the GP is using 90 training points. The root mean square error spread is getting wider and the open-loop interval is also decreased.

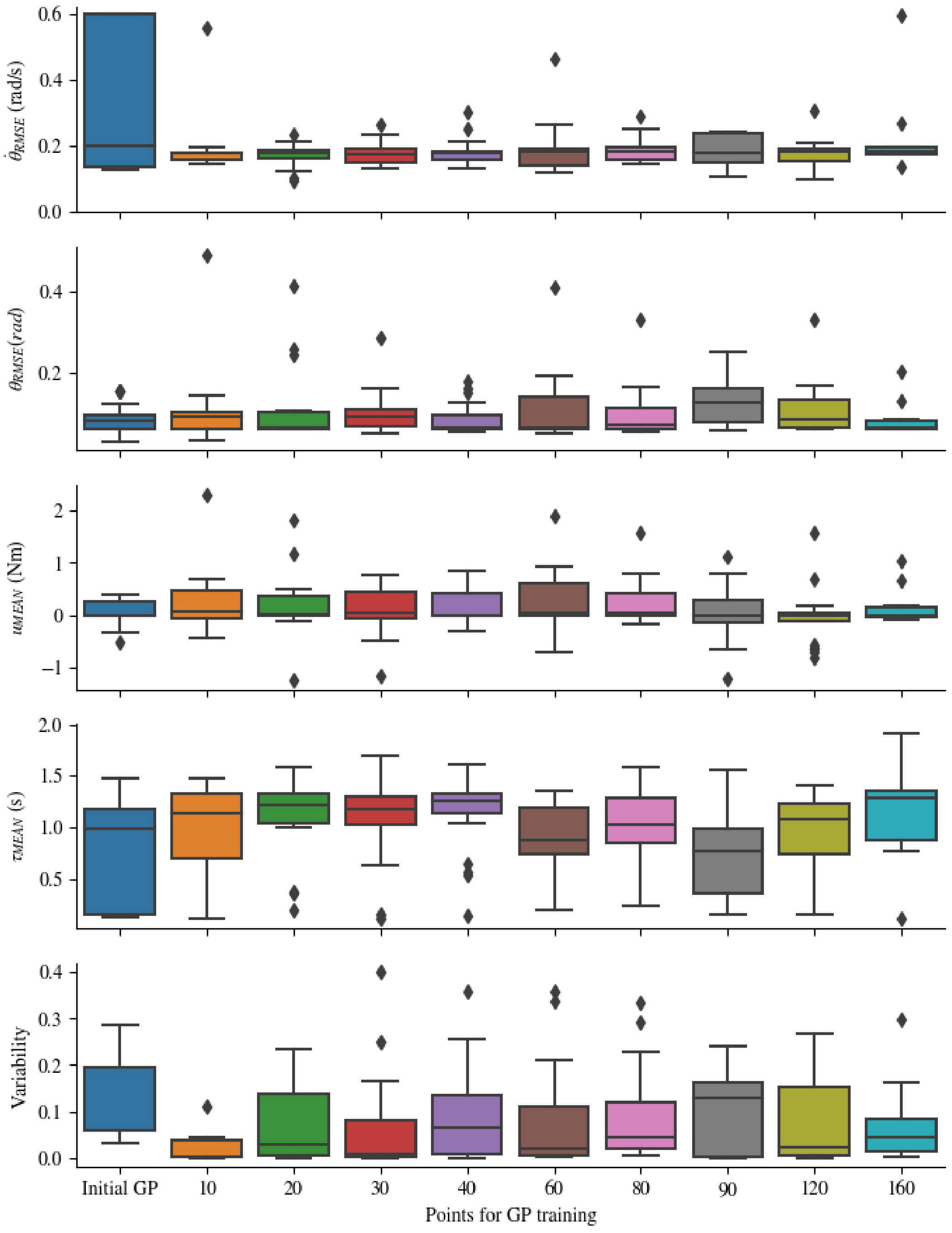

Figure 5.17: Evolution of the five measures for assessing the goodness of the GP when it is retrained online over 10 simulations. The first and second rows are the RMSE for both states of the system. The third row is showing the distribution of the mean of the control input and the fourth row is the mean of the open-loop interval. The last row is the variability over two open-loop.

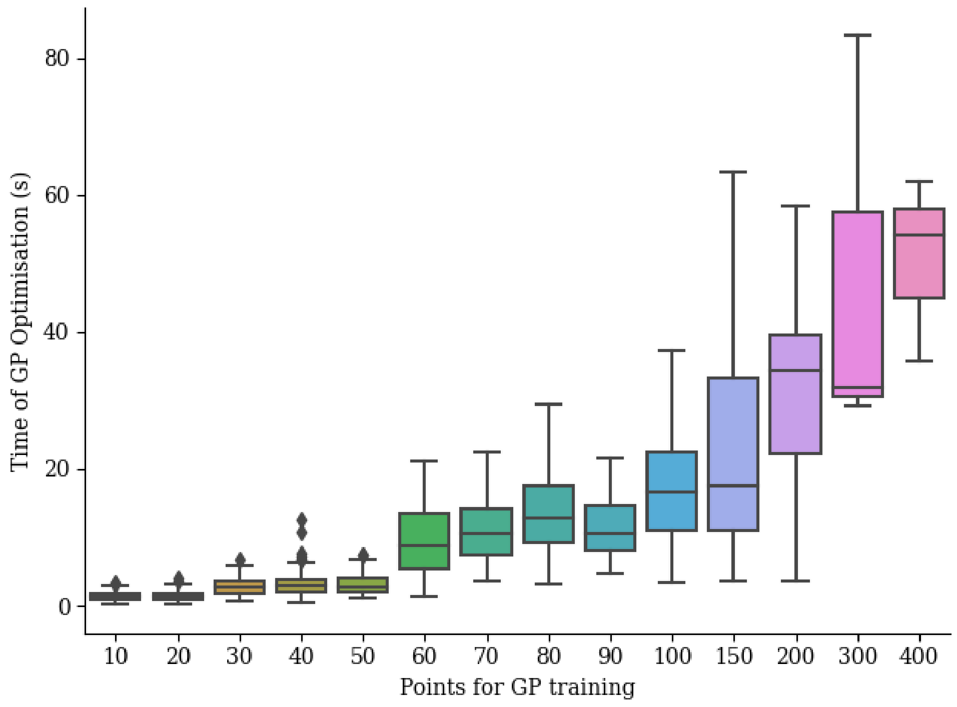

Figure 5.18: GP optimisation time related to the amount of points used for training.

Figure 5.18 shows the impact of the number of points used for retraining the GP. As also demonstrated in section 5.2.1.2, the time is increasing when adding more points for the optimisation of the GP. However, in the previous section, the optimisation time is not having an impact as the GP is computed before the simulation is starting, only the computation of the next states can be considered as a limitation for real-time application. In this scenario, the optimisation is also impacting the possibility of using the GP in a real-time settings, as the GP optimisation is taking multiple seconds in order to find the optimal hyper parameters. This optimisation time is mainly impacted by the implementation of the algorithm (see Section 3.6.6) using Python.

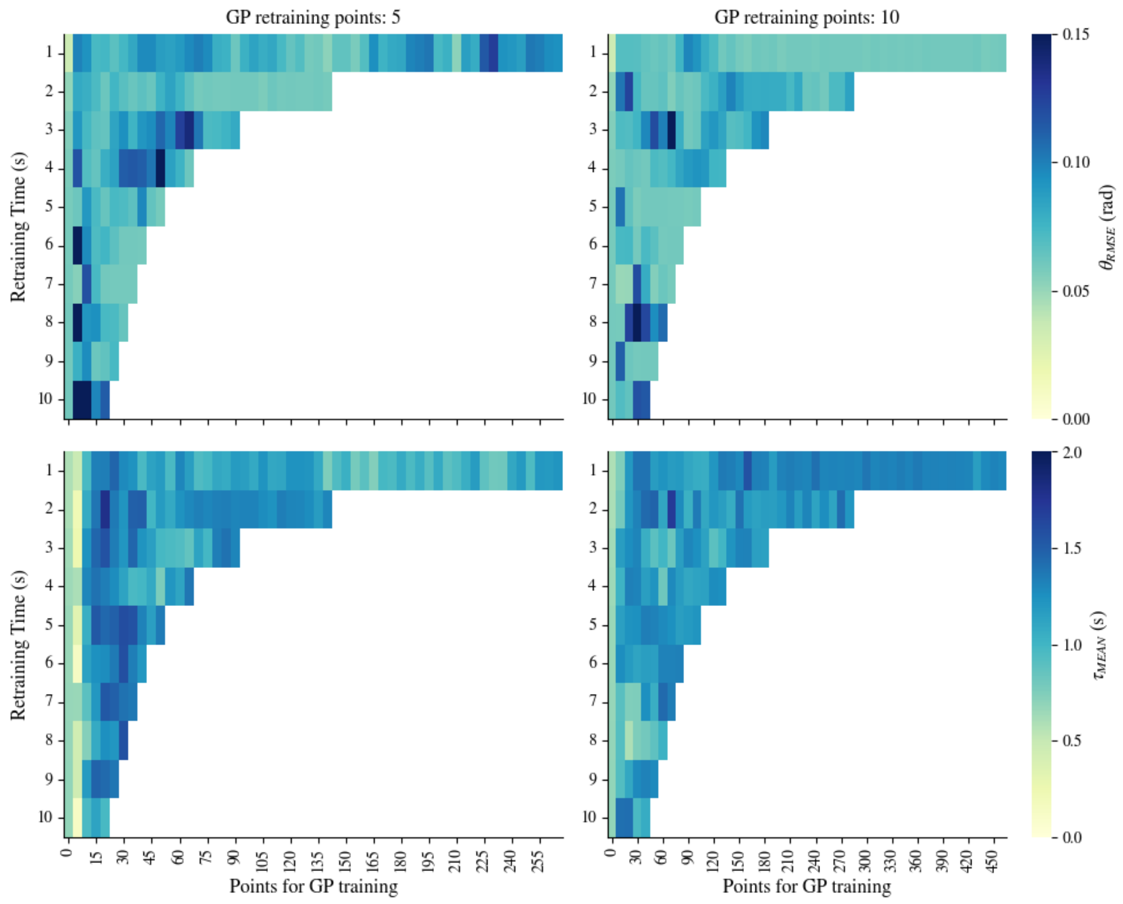

Figure 5.19: Impact of the number of points used for retraining in function of the retraining interval. The left column is adding 5 points to the current GP and the right column is adding 10 points at each retraining. The top row is the RMSE of the angle of the pendulum through the simulation and the bottom row is the mean open-loop interval. Each simulation is 60 seconds.

Figure 5.19 shows the impact of the number of points on the retraining time and the RMSE and open loop interval. As points are taken randomly from the trajectory, it might occur that the response of the controller is degraded following a training compared to the previous iteration. This can be seen in Figure 5.19 (left column), where the Root Mean Square Error of the angle is increasing over multiple retraining to then going back to lower value. The same pattern can be seen when using 10 points for retraining (Figure 5.19 (right column)), however, it seems more consistent. In most cases, this is happening when the Root Mean Square Error of the states and the mean of the control input are close to optimal conditions (see Figure 5.20 (blue line)). Hence, the retraining over multiple instance is only adding similar data points that already exist in the training data set for the GP. In addition, the open-loop interval of the GP is not improving over time as the number of points used for training keeps increasing. Keeping retraining GPs by adding more and more points without discarding old one can not only degrade the GP but also increase the required optimisation time.

As the state's RMSE and the Mean Open-loop interval can be used as proxies to assess the goodness of the GP, these measures can also be used to stop the retraining of the GP. By using this stop-relearning approach, four new thresholds are introduced, to stop the retraining of the GP when criteria are met. The first two thresholds are applied to the RMSE of the two states present in the system, $\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$. The third threshold is applied to the mean of the control input. Then the fourth threshold is applied to the standard deviation of the open-loop interval. The retraining is stopped when the plant is following the reference with a low error, and having consistent open-loop interval with symmetrical control input. The following timeseries is showing the impact of stopping the relearning of the GP when the four conditions stated above are met.

Figure 5.20 shows two time series of the same initial GP with and without the stop relearning flag enabled. The retraining time is set to 10 seconds plus waiting for the next event. When the GP is not stopped from retraining, represented in blue, the angle of pendulum is drifting away from the reference after the second retraining. However, the response of the controller following the first retraining is already giving a low RMSE as well as a mean control input around 0. However, it is going back to a low RMSE after the third retaining.

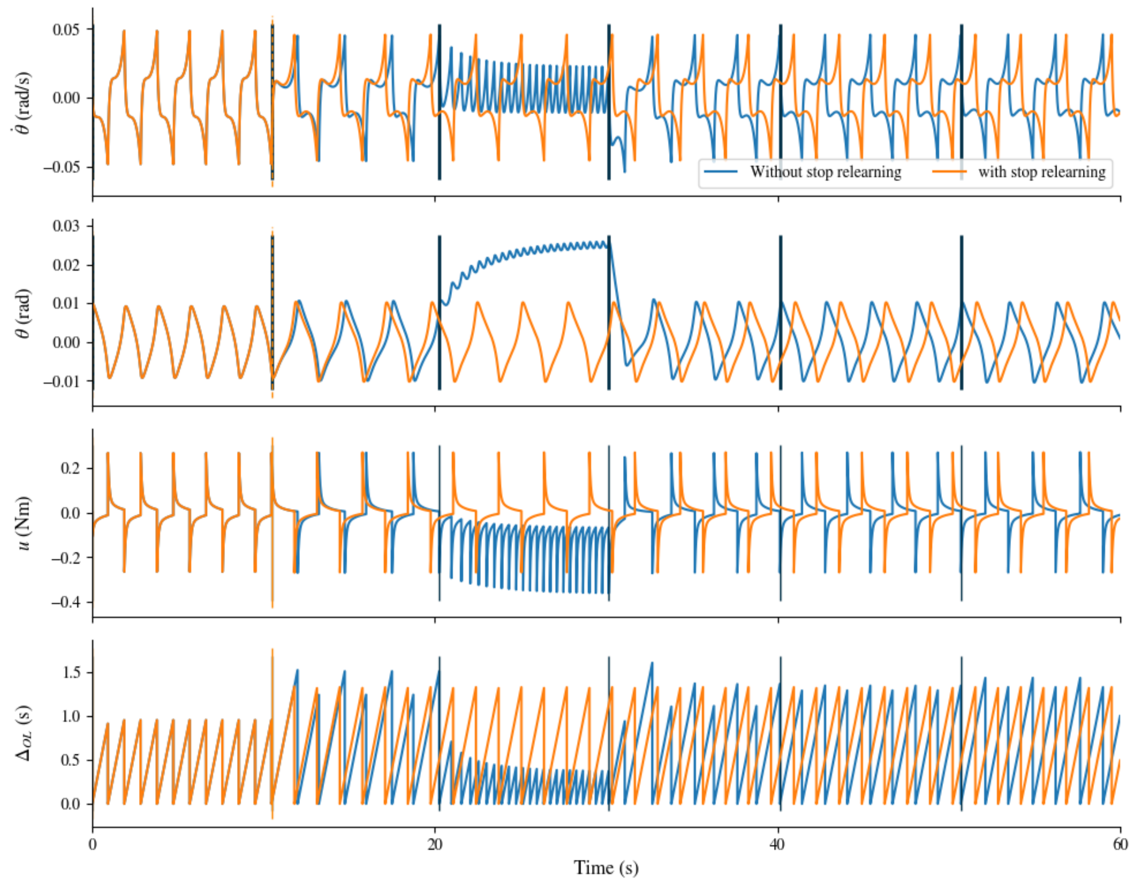

Figure 5.20: GP based Intermittent Controller. Top subplots: without stop retraining. Bottom subplots: With stop retraining. Vertical lines: GP relearning. The top row is the velocity of the pendulum: $\dot{\boldsymbol{\theta}}$. The second row is the angle of the pendulum: $\boldsymbol{\theta}$. The third row is the control input generated with GP based Intermittent Control and the last row is the open-loop interval. The blue is without stop retraining. The orange color is with stop retraining enable.

In comparison, when the stop relearning flag is enabled, in orange in Figure 5.20, a single retraining is done around 10 seconds. When it is time for the second retraining, the four thresholds mentioned above are avoiding an additional retraining, which helps keeping the simulation operating as it should as well as avoid unnecessary GP retraining, thus saving computation time. In addition, the number of points used by the GP not growing is helping in reducing the computational burden at each time step, where the inverse of a matrix is calculated.

Knowing that GP is able to improve using online data coming from the Intermittent Controller, the focus is moved to adaptation using GP in the next section.

5.2.2.2 Adaptation applied to SIP

In this section, the pendulum length is changing through time, similarly to the simulations presented in Section 4.2.2.2. First, the pendulum length is increased a single time during the simulation. Then, in the following set of simulation, the pendulum's length is increasing then decreasing in order to increase the complexity for the adaptation algorithm. Finally, the changes of the plant parameters are evolving progressively, hence the discrepancy between the controller and the plant is less spontaneous.

Initially, the length of the pendulum is changing once during the simulation, from 1meter to 1.2meter. Similarly to Section 4.2.2 where Data Informativity is used, the objective is to assess if this GP based Hold is able to detect and retrain accordingly when the system is changing. It is important to note that GP is only integrated the Hold block inside the Intermittent Controller, hence the state feedback is not updated when the GP gets updated. Previously, the system was not changing over time, so keeping state feedback constant was not an issue. However, in this case, it is important to assess the impact of only updating the Hold without updating the state feedback. First, the state feedback k is updated accordingly with the system's updates. Figure 5.21 shows the impact of the GP retraining when the pendulum's length is changing.

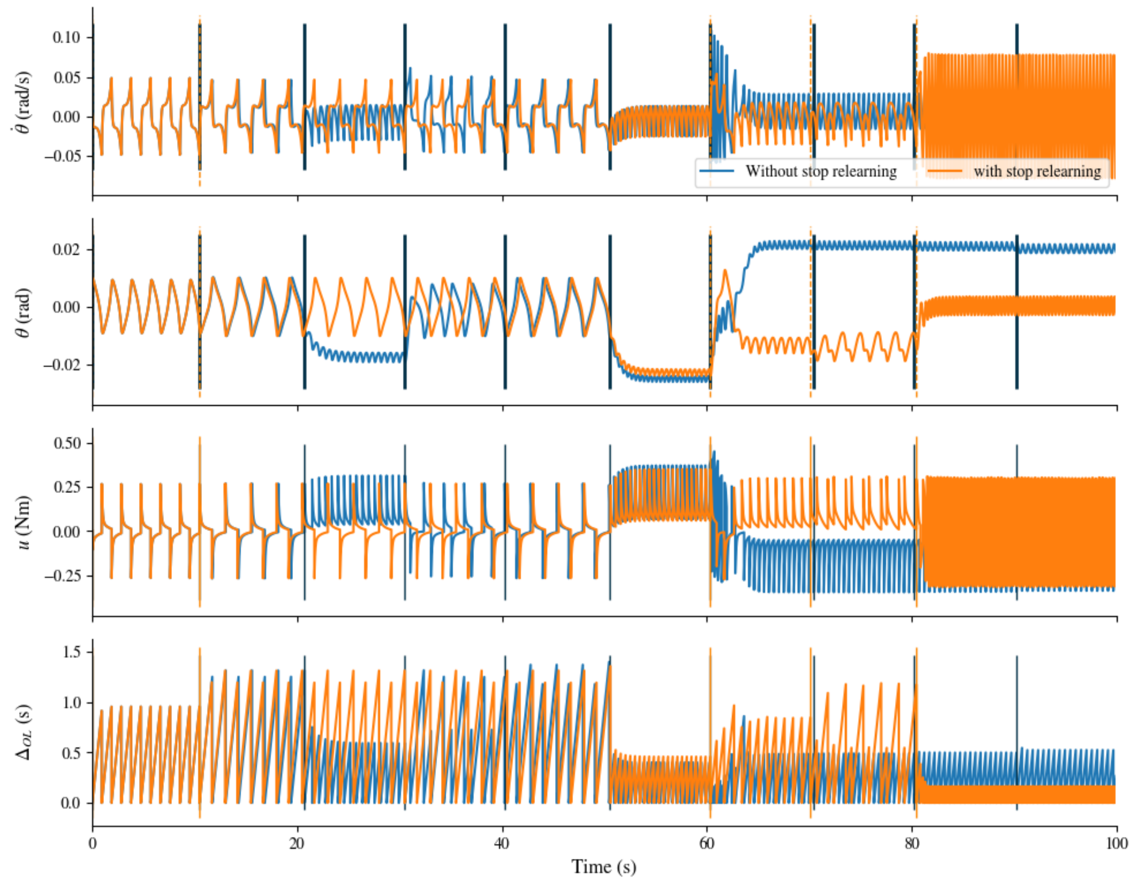

Figure 5.21: GP based Intermittent Controller. Pendulum's length changing at 50 seconds from 1m to 1.2m. Vertical grey line: GP retraining. The top row represents then velocity of the pendulum ($\dot{\boldsymbol{\theta}}$), and the second row is the angle ($\boldsymbol{\theta}$). The third row is the control input generated with GP based Intermittent Control and the last row is the open-loop interval. The blue line represents the timeserie when the GP keeps retraining whereas the orange line is stopping the relearning when pre-defined conditions are met.

In the first half of the simulation, the system stays constant, hence the GPs are only learning about a single system. After the first retraining at around 10 seconds, the open-loop interval is increasing. The states responses as well as the control input are similar to before. However, when the GPs keep retraining a second time, the response of the controller is getting off centered. This is not present when the stop retraining flag is enabled as the response of the Intermittent Controller is considered good enough based on the threshold on the states and control input. Moreover, unnecessary optimisations are not happening in contrast to the blue line: the GPs retrained at 40 seconds are back to similar controller response as the one at 10 seconds.

In the second half of the simulation, the pendulum length is increased to 1.2m. In both simulations, the response of the controller is impacted by this change. This is due to the GP initially gathering data from two different systems and trying to optimize the hyper parameters to fit both output training data. It is easily noticeable that the GP estimation is actually getting degraded when the training data is coming from two different systems. In order to avoid this issue, a forgetting factor has been introduced. This forgetting factor is helping in discarding old data and only use recent data from the optimisation of the GP. This forgetting factor has been set to two retraining times. This means that data used by the GP is not older than 2 previous retraining period. When the system is changing, the GP is getting data points from two different systems for only one optimisation. Then, after the second retraining, all data is coming from the newly updated system. Figure 5.22 shows the impact of using this forgetting factor, where the open-loop interval is getting back to around 1.4 seconds, in contrast to the 0.2 seconds present in Figure 5.21.

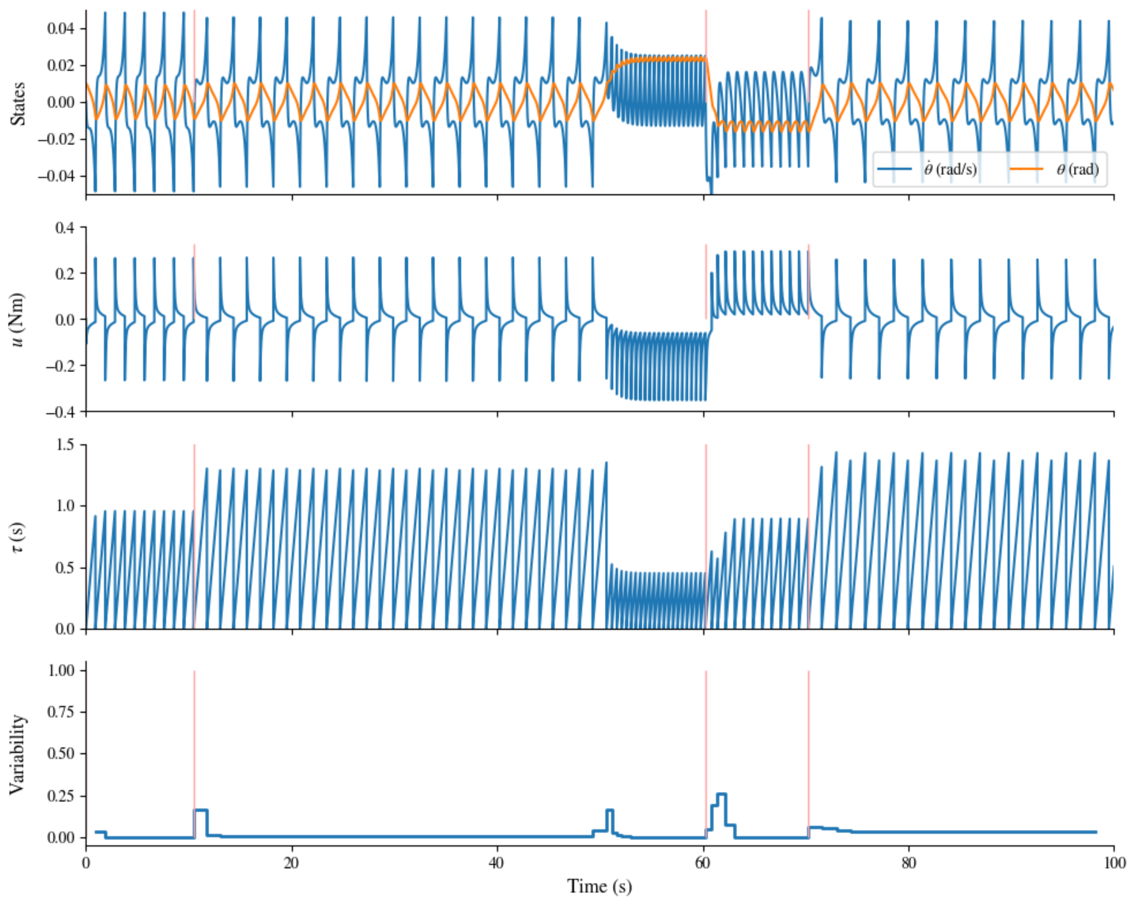

Figure 5.22: GP based Intermittent Controller. Pendulum's length changing at 50 seconds. Stop retraining is enable. Forgetting factor is enabled. Vertical red lines: GP retraining. The top row is the two states of the Single Inverted Pendulum ($\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$). The second row is the control input generated with GP based Intermittent Control and the third row is the open-loop interval. The last row is the variability of $\boldsymbol{\theta}$ (in blue)

At the first retraining after 50 seconds, the GP is retrained using data from the system before and after the change of length, similarly to the previous simulations. However, when the second retraining is happening, all data passed for the retraining belongs to the newly updated length, hence the optimisation of the GP is improved in comparison to the simulation shown in Figure 5.21. In addition minimum open-loop interval threshold has been implemented. This is helping by forcing retraining when the GP response is still below the threshold. This is too avoid the case where the states are matching the set point due to a very small triggering interval.

The same approach is used when the plant is changing following the stairs pattern as used in Section 4.2.2. Figure 5.23 is showing the impact of the retraining interval t_{GP}.

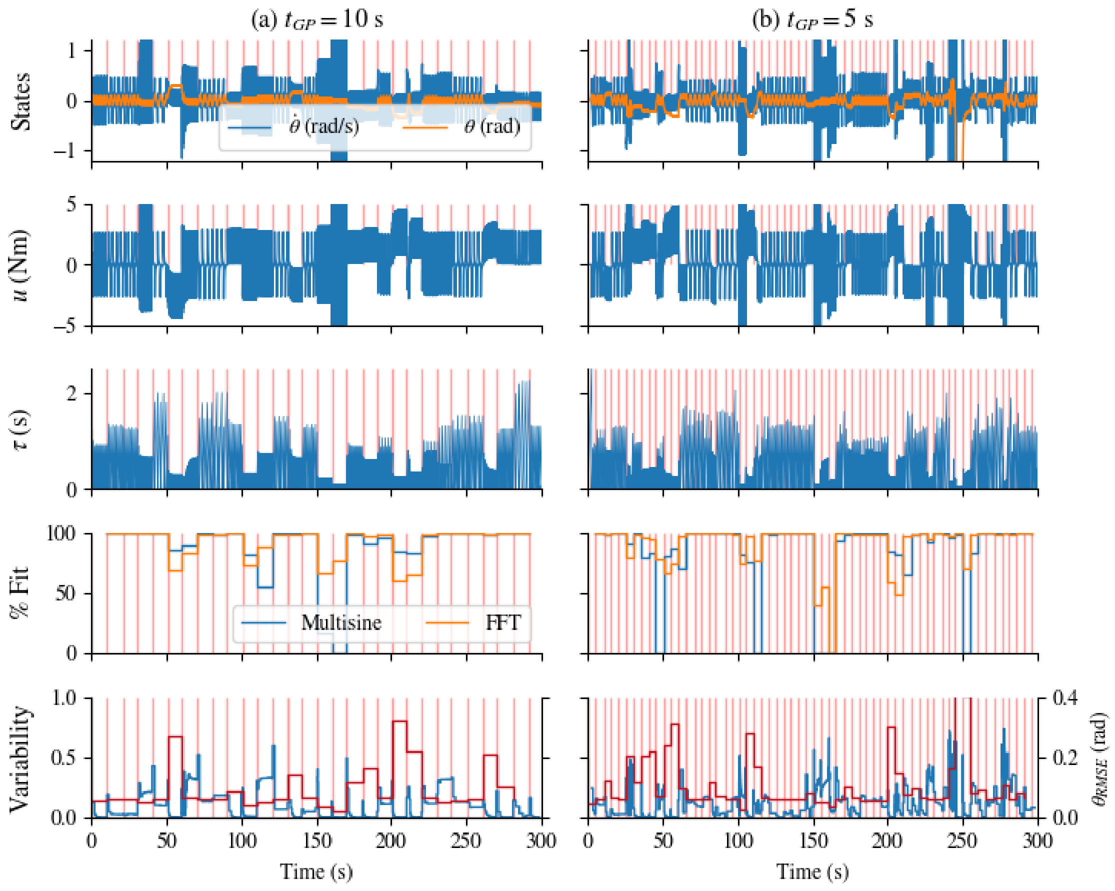

Figure 5.23: GP based Intermittent Controller. Forgetting factor is enabled. (a) $t_{GP} = 10$ sec and (b) $t_{GP} = 5$ sec. Vertical red lines: GP retraining. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input generated with GP based Intermittent Control. The third row shows the open-loop interval. Forth row is the % fit of the GP. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). Vertical lines represents a retraining of the GP (red). The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

As stated in Section 5.1.2, assessing the fit of the GP to the actual plant can be difficult due to the black box representation. The approach used here is based on applying similar control input to both plants (true and GP representation) and to compute the fit percentage between the two. For both t_{GP}, the GP and the true system start from some initial conditions ([0; 0.1]) and a state feedback controller is applied to each of them. This is using the correct state feedback for the current system. In the case where the plant dynamics are fully unknown, getting an accurate representation of $\mathbf{k}$ can be complicated.

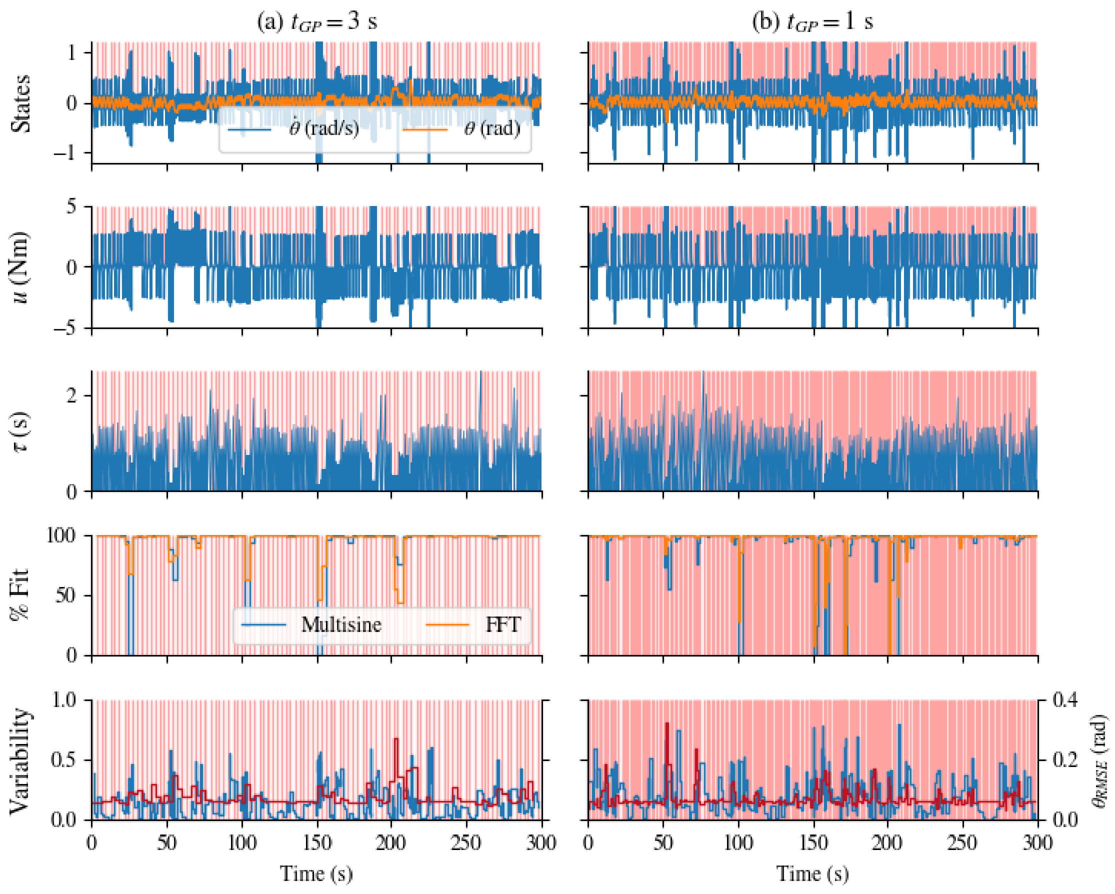

Figure 5.24: GP based Intermittent Controller. Forgetting factor is enabled. (a) $t_{GP} = 3$ sec and (b) $t_{GP} = 1$ sec. Vertical red lines: GP retraining. The top row is the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input generated with GP based Intermittent Control. The third row is the open-loop interval. Forth row is the % fit of the GP. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). Vertical lines represents a retraining of the GP (red). The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

Here, the focus is to understand the capabilities of the GP in estimating accurately the plant. Similarly to the Data Informativity results in Section 4.2.2, the longer the retraining time is, the longer the controller is lagging in term of design to the plant. In these simulations, the system's changes are slow enough for the GP to get multiple retraining iterations per system change.

In the case where $t_{GP} = 10$ sec (Figure 5.23 (a)), the fit of the GP is decreasing when the GP is retrained with the mix of two systems. Then it gets back close to 100%. Similar results can be seen in Figure 5.23 (b) and Figure 5.24 (a) and (b). However, when the retraining time decreases, the fit percentage is getting more variable. This can be clearly seen where the retraining time is 1 second, and the pendulum length is 0.8m (between 150 and 200 seconds in the simulation). In order to keep more stability into the GP representation of the plant, keeping a long retraining time is more beneficial. However, this can bring instability of the control system if the plant is changing toward heavily unstable conditions and the retraining is only happening 10 seconds later. It is worth mentioning that each of these simulations are still stable no matter the retraining time t_{GP}.

Similarly to the Data Informativity results, if the retraining time is larger than the time where the system is changing, the controller is always lagging behind. Decreasing the time between each retraining can help avoiding it. However, due to the lack of data available for training the GP, it is possible that the GP is not reaching the best possible representation of the system. For example, this can be seen in Figure 5.24 (b), when the pendulum's length is decreased to 0.8m: the fit value of the GPs is jumping back an forth compared to a stabilized fit above 80% for the subplots (a), (b) in Figure 5.23 and in subplot (a) in Figure 5.24 (a).

The results above show that Gaussian Processes are able to be used within Intermittent Control to handle adaptation when the pendulum parameters are changing fast. In Figure 5.25 and 5.26, the pendulum's length is changing more slowly. The pendulum's length is getting updated every 10 seconds between 30 seconds and 70 seconds, changing from 1m to 1.2m (0.04m / 10 sec). Then the length is decreasing similarly, from 150 seconds to 240 seconds, going from 1.2m to 0.8m (-0.04m / 10 sec). Similarly to the previous simulations, the different retraining times t_{GP} are compared: 10, 5 (Figure 5.25), 3 and 1 seconds (Figure 5.26).

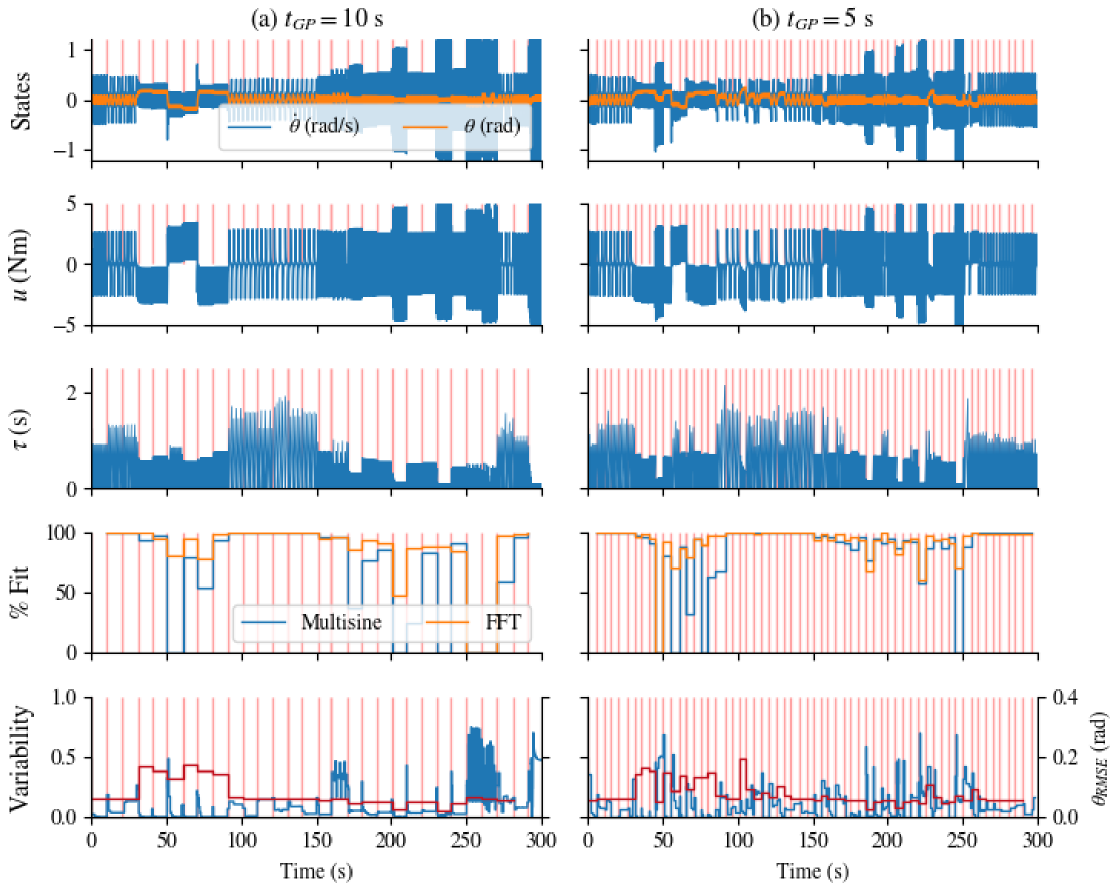

Figure 5.25: GP based Intermittent Controller. Forgetting factor is enabled. (a) $t_{GP} = 10$ sec and (b) $t_{GP} = 5$ sec. Vertical red lines: GP retraining. The top row is the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input generated with GP based Intermittent Control. The third row is the open-loop interval. Forth row is the % fit of the GP. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). Pendulum's length changing from 1m to 1.2m (0.4m/10sec) starting at 30 seconds. Then decrease from 1.2m to 0.8m (-0.4m/10sec).

Retraining the controller every 10 seconds is making the whole design of the controller off-sync similarly to the results presented in Figure 4.28, due to the similar time frame for the plant to update its length. This is noticeable when focusing on the increase of θ_{RMSE} and the decrease of % fit between 30 and 90 seconds. The open loop interval is also decreasing during this period, before going back to an average of 1.5 seconds between 90 and 150 seconds. Similar behaviour can be observed when the pendulum is gradually decreasing its length.

The same observation can be made when $t_{GP} = 5$ seconds, however the % fit in the frequency domain is not becoming as low as before. This is due to the GP being able to retraining with only data coming from the updated plant. Moreover, when $t_{GP} = 3$ seconds, shown in Figure 5.26, the % fit is staying above 75% in the first 100 seconds. The variability however is getting higher due to the more regular update of the GP. Similar results can be observed when $t_{GP} = 1$ second and it is worth mentioning θ_{RMSE} being lower due to a single second where the plant and the controller are not matching. When $t_{GP} = 10$ seconds, this time is increased to 60 seconds, when the pendulum's length stop increasing. Furthermore, the % fit of the GP when $t_{GP} = 1$ second is quickly going from 0% to 100% after a single retraining.

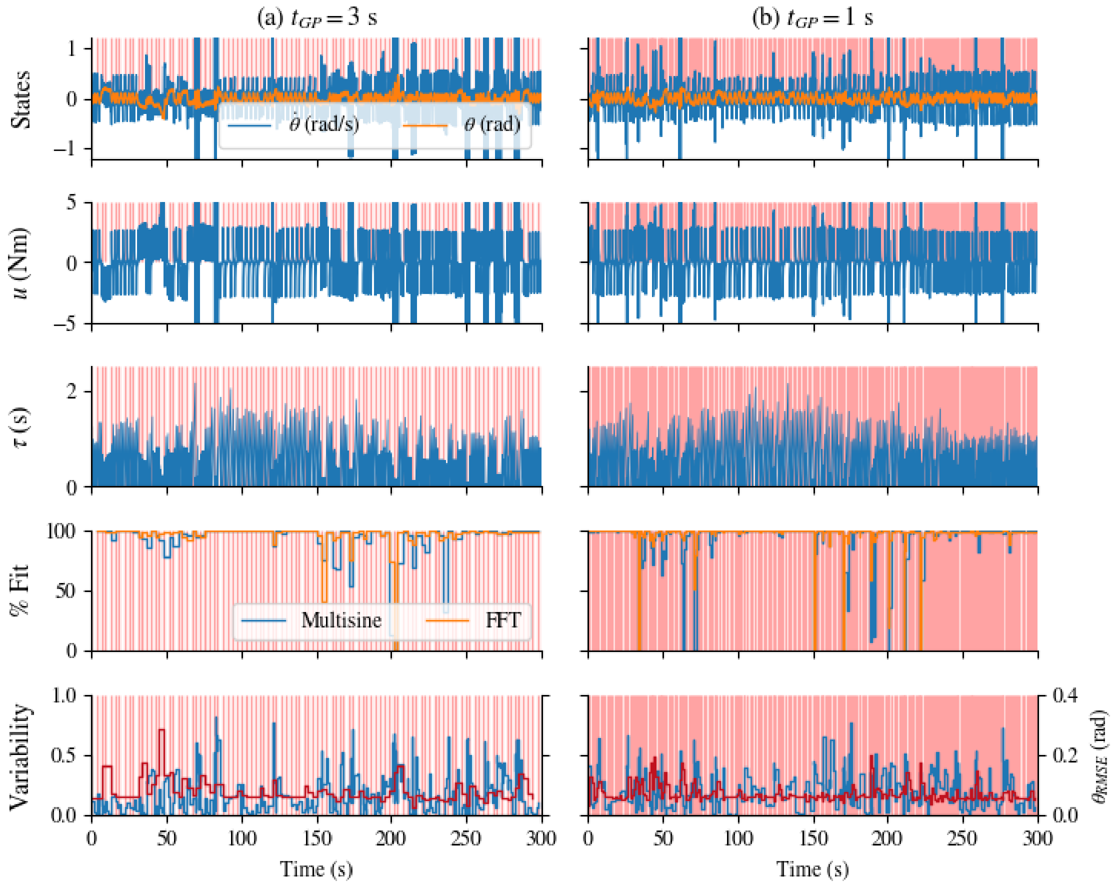

Figure 5.26: GP based Intermittent Controller. Forgetting factor is enabled. (a) $t_{GP} = 3$ sec and (b) $t_{GP} = 1$ sec. Vertical red lines: GP retraining. The top row is the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input generated with GP based Intermittent Control. The third row is the open-loop interval. Forth row is the % fit of the GP. The last row is the variability of θ (in blue) and θ_{RMSE} (in red). Pendulum's length changing from 1m to 1.2m (0.4m/10sec) starting at 30 seconds. Then decrease from 1.2m to 0.8m (-0.4m/10sec).

The retraining time t_{GP} is having an important role in the behaviour of the Intermittent Controller. If the objective is to keep a low variability, having a longer t_{GP} can be beneficial. However, if the system is changing through time, it is important to have t_{GP} below this change rate. This is allowing the GP to get multiple retraining with data coming from a unique plant. The number of retrainings per unique system is the value set in the forgetting factor used for the results presented in Figure 5.22. In order to get a low θ_{RMSE}, decreasing t_{GP} is beneficial but this is increasing the variability of the simulation as well as the open-loop interval.

The incorporation of GP to replace the traditional SMH has interesting benefits, especially in the adaptation case. However, this approach is not able to estimate the state feedback gain required by the Intermittent Controller. In this section, the state feedback gains have been redesigned in sync with the change of the system. Hence, this is assuming that the system changes are known. However, based on results presented in Figure 4.15, whether the state feedback are updated or not is not introducing much differences.

5.3 Discussion

Switching from a deterministic hold approach to a stochastic one, this chapter has shown the feasibility of Gaussian Processes to model system dynamics instead of using the traditional System Matched Hold in the Intermittent Control Framework. One of the advantages of using Gaussian Process is moving toward a data-driven approach. By using data generated by the Intermittent Controller, the GPs are able to model the open-loop system dynamics in contrast to the closed-loop one when using data coming from a Continuous Controller as presented in Figure 5.16 when training the initial GP using a mutisine disturbance (Figure 5.7).

In addition, GP is able to introduced variability in two different ways with the need of adding additional disturbance signals such as noise. By enabling uncertainties propagation, it is possible to generate a new set of trajectories which shift away from a perfect representation of the plant's dynamics as presented in Figure 5.11. Another way of adding variability into the controller output is to often retrain the GP. This optimisation is updating the hyper parameters of the GP which are slightly different from the previous ones due to the newly added data. Moreover, GP is able to accurately model the system's dynamics with a low number of points, which then can keep the optimisation as well as the next state prediction fast.

Gaussian Processes are able to take advantage of the open-loop trajectories generated by the Intermittent Controller in order to retrain using online data. With the effect of the triggering, hence constant motion of the plant, the data gathered is around the operating area and are ideal to train a GP. Furthermore, it is possible to adjust GP specific parameters to influence the controller behaviour. For example, adjusting the number of point used for training could lead to a more accurate representation of the systems dynamics, however this is not without some limitations.

Whilst using GP is helping moving towards a fully data driven Intermittent Controller, this approach can show some limitations. As shown in Figure 5.18, the optimisation time of the GP is highly impacted by the number of points used for training. In the case of simulation, it is possible to artificially pause the simulation until the optimisation is finished. However, this is not realistic when applied to real-time systems. Another downside of using GP is focusing on the state feedback estimation. This approach is only able to model an approximation of the system's dynamics, however, due to GP being based on hyper parameter, it is not possible to get matrices $\mathbf{A}$, $\mathbf{B}$ or even $\mathbf{k}$ due to the non-linear covariance function used to model the data.

As presented in Figure 5.20, it is possible to stop the retraining of the GP when the behaviour of the controller is aligned with our requirements. However, there is not an empirical way of fixing these ”stop relearning” thresholds, and it needs to be done manually depending on the Intermittent Controller parameters used. Finally, it is worth mentioning that while GP is able to accurately estimate the system's dynamics using online data, it is necessary to gather data for a certain period of time before retraining. This time period can be an issue if the system changes are too important and loose stability before the GP retraining.

As presented in chapter 4.2, it is possible to use Data Informativity or Reinforcement Learning to estimate the accurate state feedback for the controller. In this chapter, the Hold is using a Gaussian Processes model. In the following chapter, both state feedback estimation frameworks are used in a GP based Intermittent Controller, in order to move towards a fully Adaptive Stochastic Intermittent Data Driven Controller.

5.4 Summary

An overview of the work accomplished in this Chapter is summarized in Table 5.4. This table focuses on the pros and cons of using different types of GP within IC. In addition, limitations and generalizations are mentioned. Everything is based on simulation data only, allowing easy assessment of the Intermittent Controller against different sets of parameters, as well as system properties changing through time and not needing an observer to access all states.

Algorithms	Pros	Cons	Limitations
Full GP	Can use uncertainties propagation to introduce variability	Each state is considered independent	Optimisation Time
Sparse GP	Reduce the complexity of the GP	Double optimisation (Full GP then Sparse)	Level of sparsity can heavily affect the controller behavior
Multi Task GP	Matrices that model inter-task relationship	More training data is required for training	This does not improve computation speed for real-time application
Uncertainty Propagation	Introduce variability without external disturbances	Slow computation time	Multi step ahead prediction required to sample from the distribution

Table 5.4: Summary table covering Chapter 5 contributions.

The fundamental limitation of these approaches is about the state feedback gains. Even though, these algorithms can model the dynamics of the plant, $\mathbf{k}$ is not getting estimated.

The main outcome of this chapter is that GP is able to model the open-loop dynamics of the plant even under the influence of the controller. In addition, the Square Exponential Kernel function is able to model the dynamics of the pendulum and the cartpole systems.

Chapter 6

Adaptive Stochastic Intermittent Data Driven Control

As presented in Chapter 4, both Reinforcement Learning and Data Informativity can get an accurate state feedback estimation that can be used within the Intermittent Controller. Data Informativity estimates the system matrices **A** and **B** used to design the Hold block and the state feedback. However, as presented in Section 4.2.2.2, the variability of this approach is low due to the high accuracy of the matrices estimation as well as the deterministic nature of the Hold in both the noiseless and noisy cases.

Whilst the Reinforcement Learning implementation is not able to estimate the system matrices **A** and **B** that are necessary to obtain an accurate controller, there is a need to find an alternative for the implementation of the Hold block. Gaussian Processes have been used as a replacement for the traditional SMH, using a data-driven approach and results are presented in Chapter 5. In this chapter, the focus is to apply Reinforcement Learning and Data Informativity in conjunction with a GP-based Hold and assess its characteristics in multiple scenarios.

6.1 Methods

Similarly to previous sections, the following block diagram highlights which parts of the Intermittent Controller are getting updated online with both the Reinforcement Learning and Data Informativity algorithms, as well as the Gaussian Processes relearning.

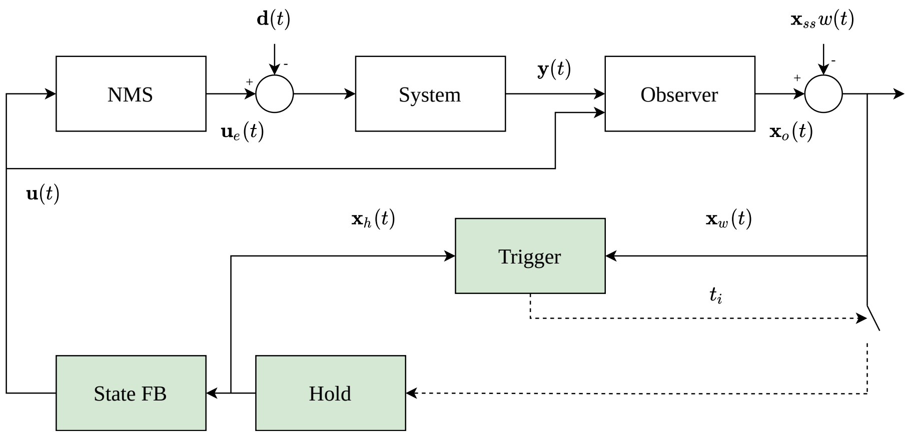

Figure 6.1: Intermittent Control block diagram of the controller updated by the Reinforcement Learning, Data Informativity and GP algorithms are highlighted in green.

The Hold is updated using the GP model, whereas the State Feedback block is updated with the estimation coming from the Reinforcement Learning or Data Informativity framework. Even though results are showing the estimation of system matrices **A** and **B** where Data Informativity is used, these are only passed to the LQR design method to generate a state feedback gain. These matrices are not used in the hold in any of the results in this Chapter. The trigger block is updated using an approach similar to the one presented in Section 5.2.1.1, where Uncertainty can be used as a triggering mechanism.

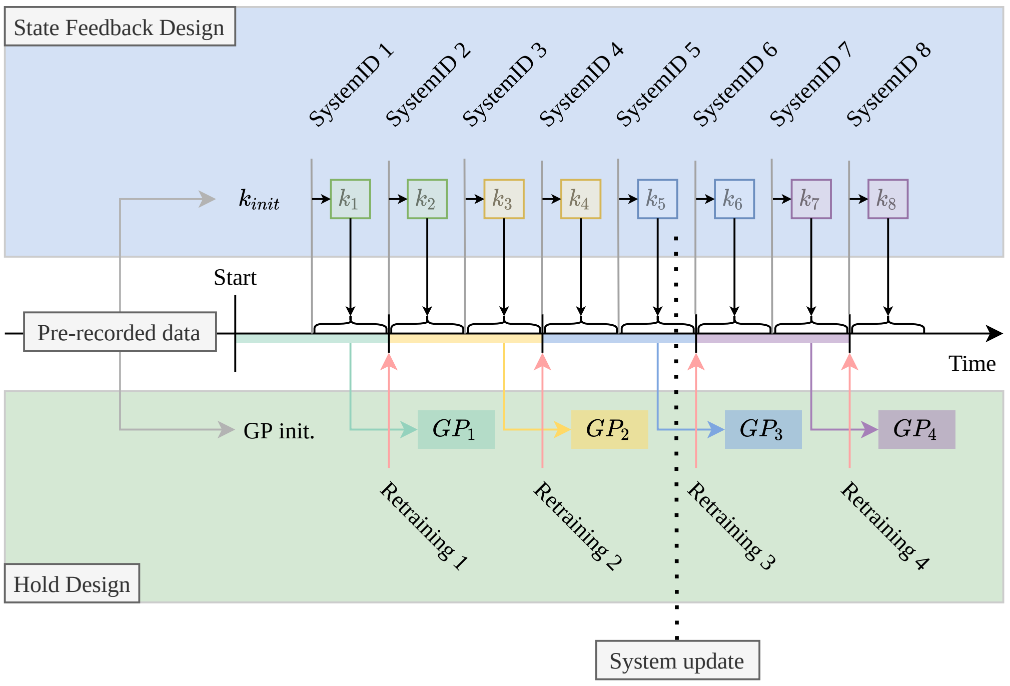

Figure 6.2: Diagram representation of the timing between Hold and State feedback redesign. Hold Design is based on Gaussian Processes. State feedback design can use Reinforcement Learning or Data Informativity.

The diagram in Figure 6.2 shows the implementation of the framework combining the Gaussian process re-learning process and the state feedback relearning process. The state feedback relearning process can use the Reinforcement Learning algorithm or the Data Informativity. Both of the redesigns are using their own timing which can be set to any value and do not need to be in sync with each other. However, similarly to previous individual retraining presented in Chapter 4, it is necessary to be synced with the triggering mechanism of Intermittent Control, hence avoiding any change of controller during an open-loop interval.

Focusing on the state feedback estimation part, only data recorded between each retraining is used to estimate the next set of matrices. This approach is feasible due to the consistency in the estimation, especially when using the Data Informativity algorithm (results presented in Section 4.2.2.2). For the Gaussian Process optimization, the data from the previous retraining to the current time can be combined with the data already passed to the GP currently in use. This combination of overlapping data between multiple retraining can be controlled by increasing or decreasing the forgetting factor (results presented in Section 5.2.2, see Figure 5.22) that allows to select the amount of previous retraining data used for the current retraining. Even if the GP training is quite reliable in estimating the correct system dynamics, having a wrong Hold has a bigger impact than having inaccurate state feedback coming from the system identification process as presented in Figure 4.15.

In this Chapter, simulations are applied the Single Inverted Pendulum (see Appendix A). Similarly to Chapter 4 and Chapter 5, the pendulum length is changing to assess the responsivity of the Intermittent Controller.

6.2 Results

This section presents the results of simulations where system identification algorithms are used in conjunction with a GP-based Hold. Firstly, Reinforcement Learning and Gaussian Processes are analyzed, and then, Data Informativity and GP-based Hold are looked into.

6.2.1 Reinforcement Learning with GP based Hold

In this section, Intermittent Control using the traditional SMH is compared to using GPH when Reinforcement Learning is used to estimate the state feedback gain and it is applied to a time invariant Single Inverted Pendulum. To assess the capabilities of GPH over the SMH, multiple conditions are tested including the variation of the threshold, allowing to switch between clock-driven and event-driven mode, as well as the initial poles used by the controller.

As explained in Section 3.4.2, the algorithm must start with a stable controller $\mathbf{k}_0$. When $Q_c = I$ and $R_c = 0.1$, the LQR method is resulting in an optimal gain of $\mathbf{k}_{opt} = [2.36, 9.91]$. This corresponds to the following poles: $[-6.83, -2.92]$. It has been decided to start with three different sets of initial stable poles: $[-3.1, -3]$, $[-8.1, -8]$ and $[-2.1, -2]$. In the first set of poles, both of them are located in between the two optimal poles, and in the second set, the control is designed with poles to the left of the optimal poles, thus the controller is faster. In the last condition, the controller is slower, hence closer to being unstable.

In addition, it is also necessary to sample point only when the loop is closed as presented in Section 4.2.1. Results, with clock-driven mode enabled, with controller poles located at $[-3.1, -3]$ and with a minimum open loop interval of 3ms, are shown in Figure 6.3.

As presented in Figure 6.3, both controllers are outputting very similar control signals and the estimation of the state feedback gain converges to the optimal value after 11 estimations for both types of Hold. The controller operating in clock-driven mode with a very low open-loop interval is similar to a continuous controller.

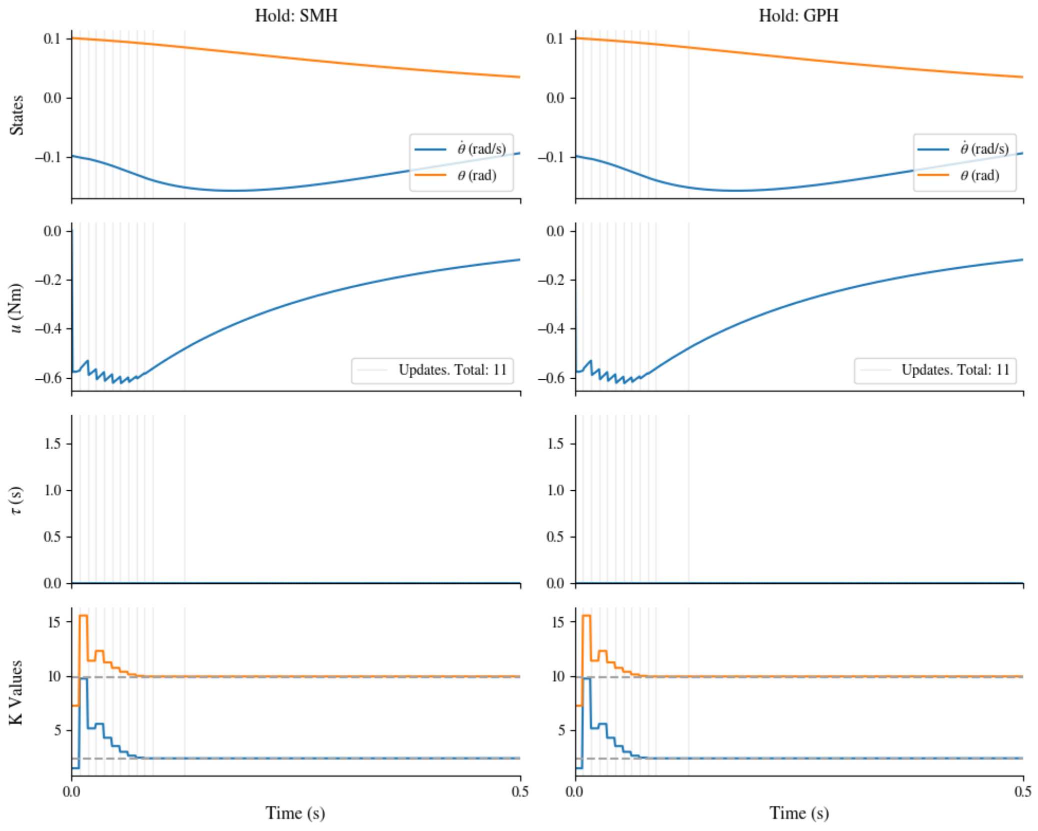

Figure 6.3: Reinforcement Learning with GP based Hold. The system is the Single inverted Pendulum. Top row is the states of the SIP. Second row is the control input outputted by the Intermittent controller. The third row is the open-loop interval. The last row is the estimation of state feedback gain. The dash line represents $\mathbf{k}_{opt}$. The left side is using SMH and the right side is using GPH. Intermittent parameters: $min\Delta_{ol} = 3ms$, clock-driven mode, controller poles: $[-3.1, -3]$.

Next, the minimum open loop interval is increased to 50ms. This is forcing more time between each retraining as the algorithm is expecting 3 samples to calculate the controller gains. Moreover, the poles of the controller are set now to $[-8.1, -8]$. Results are shown in Figure 6.4.

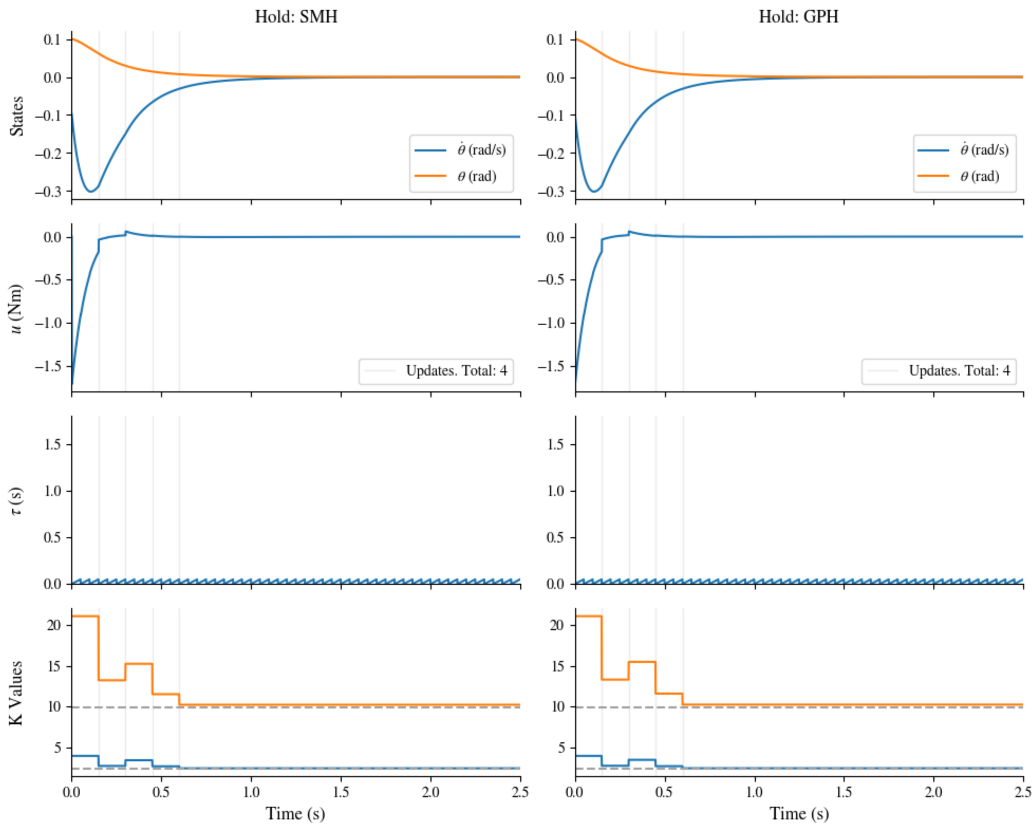

Figure 6.4: Reinforcement Learning with GP based Hold. The system is the Single inverted Pendulum. Top row is the states of the SIP. Second row is the control input outputted by the Intermittent controller. The third row is the open-loop interval. The last row is the estimation of state feedback gain. The dash line represents $\mathbf{k}_{opt}$. The left side is using SMH and the right side is using GPH. Intermittent parameters: $min\Delta_{ol} = 50$ms, clock-driven mode, controller poles: $[-8.1, -8]$

Similarly to Figure 6.3, Figure 6.4 shows the capability for both SMH and GPH to estimate the correct state feedback gain even though the initial gain is not located around the optimal location. For both hold types, the estimation takes 4 iterations to converge to the correct gains. By increasing the mimimum open loop interval to 50ms, the states of the system can vary more between each sample, thus reducing the amount of estimation needed. However, the last optimization is around 550ms, which is much longer than the case where the mimimum open loop interval equal 3ms, where the last optimization is at around 125ms.

Keeping the same minimal open loop interval, the initial controller poles are changed to the more difficult case, where $c_{poles} = [-2.1, -2]$.

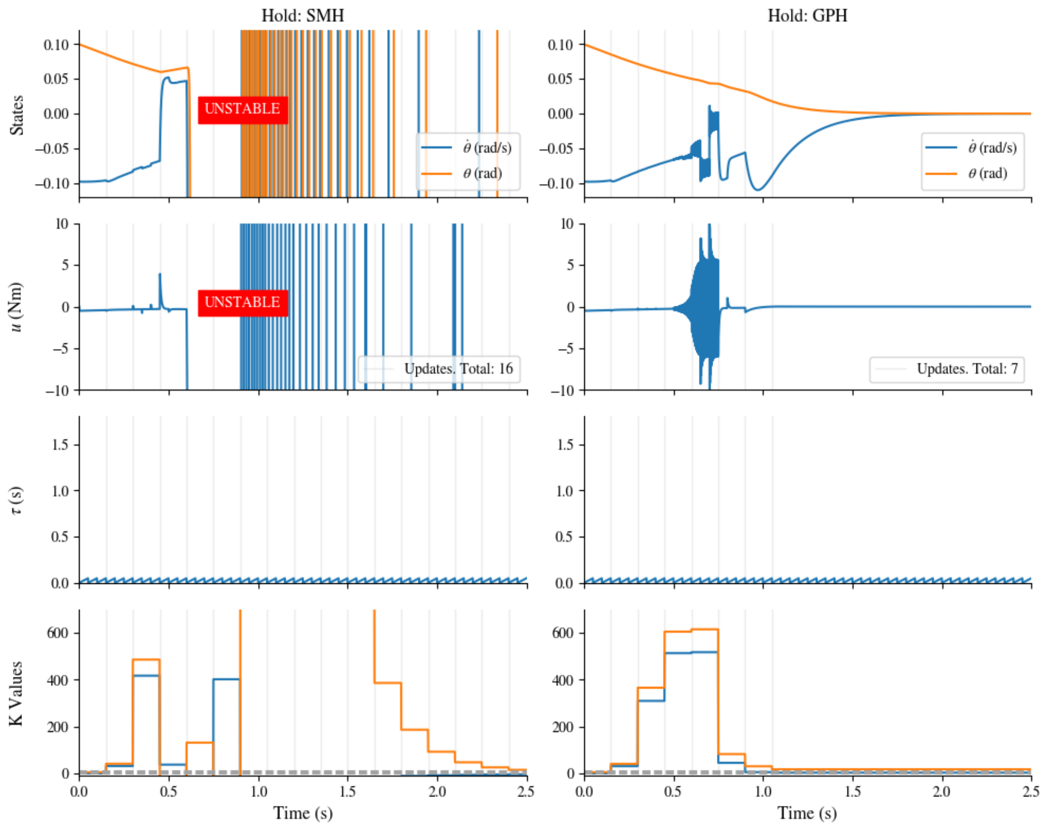

Figure 6.5: Reinforcement Learning with GP based Hold. The system is the Single inverted Pendulum. Top row is the states of the SIP. Second row is the control input outputted by the Intermittent controller. The third row is the open-loop interval. The last row is the estimation of state feedback gain. The dash line represents $\mathbf{k}_{opt}$. The left side is using SMH and the right side is using GPH. Intermittent parameters: $min\Delta_{ol} = 50ms$, clock-driven mode, controller poles: $[-2.1, -2]$

The SMH case is unstable whilst the GPH case is keeping the Single Inverted Pendulum stable. However, both of these simulations are not converging towards the optimal solution. The final state feedback gain computed for SMH is $[-3.1, 17]$ and the one for GPH is $[4.1, 18.1]$. In the case of SMH, the first gain is estimated as negative which is the reason for the instability. When GPH is used, the state feedback gain estimation diverges, creating more excitation due to the probabilistic nature of the hold (around 700ms), with the result that the system is able to get back closer to the optimal gains.

Moving away from the clock-driven case, the Intermittent Control threshold has been set to 0.0001. This is setting up the Intermittent Controller to event mode, allowing for open-loop intervals to be longer than the mimimum open loop interval. Both of these simulations are keeping the SIP stable, however, GPH is only updating the state feedback gain once whereas the SMH case is updating twice, allowing it to get closer to the optimal gain. This is due to the threshold of relearning present in the Reinforcement Learning algorithm that needs to be exceeded by the cost value. This threshold can have quite a negative impact on the simulation if set too low, as the retraining will never stop even though the optimal value is found. This can result in instability, especially when the controller is clock-driven with a very low minimum open loop interval, resulting in a lack of persistence of excitation.

Even though Reinforcement Learning can converge towards an accurate state feedback estimation in some cases, it does not use the full potential of Intermittent Control features specifically the open-loop data. It is worth highlighting the ability of GPH to keep the system stable even though the state feedback is wrong for a certain period. In addition, Reinforcement Learning required a significant amount of fine-tuning when applied to the GPH as seen in Figure 6.6, where the gains are only estimated once.

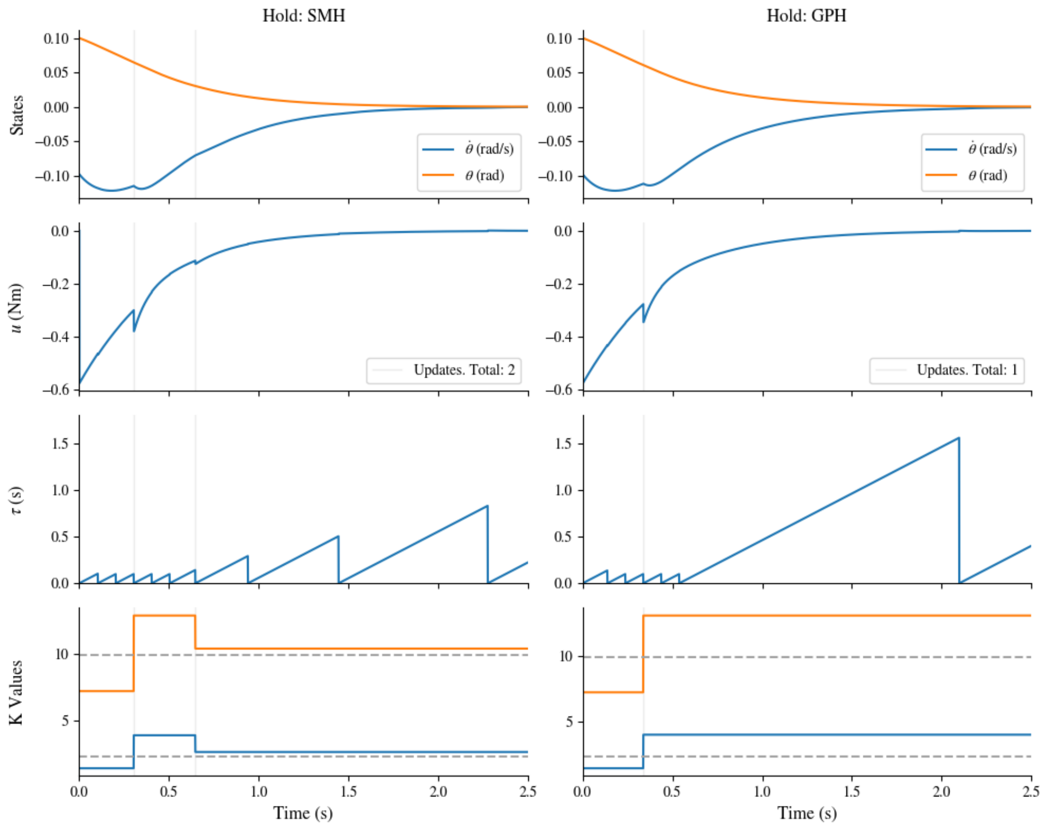

Figure 6.6: Reinforcement Learning with GP based Hold. The system is the Single inverted Pendulum. Top row is the states of the SIP. Second row is the control input outputted by the Intermittent controller. The third row is the open-loop interval. The last row is the estimation of state feedback gain. The dash line represents $\mathbf{k}_{opt}$. The left side is using SMH and the right side is using GPH. Intermittent parameters: $min\Delta_{ol} = 100$ms, Threshold $=$ 0.0001, controller poles: $[-3.1, -3]$

6.2.2 Data Informativity with GP based Hold

As presented in the previous section, the Intermittent Controller with GPH can be used with the Reinforcement Learning algorithm. However, it is not able to use open-loop intervals as previously mentioned. In this section, Intermittent Control uses a GP-based Hold, and Data Informativity is used to estimate the state feedback gains required by the controller for a time variant Single Inverted Pendulum (see Appendix A).

This section is separated into two parts: (1) understanding the influence of timing between the GP retraining and the Data Informativity timing t_{DI}, (2) understanding the impact of the IC parameters, such as threshold and minimum open-loop interval.

The assessment of the performance is a combination of the assessment presented in Sections 4.1.2 and 5.1.2. In addition, to compare different values for t_{DI} and t_{GP}, the data used to compute the measures is based on the last 20 seconds before the system changes its length. This is to ensure that the transient effect of changing the system is not included. Figure 6.7 is a diagram representation of the data selection.

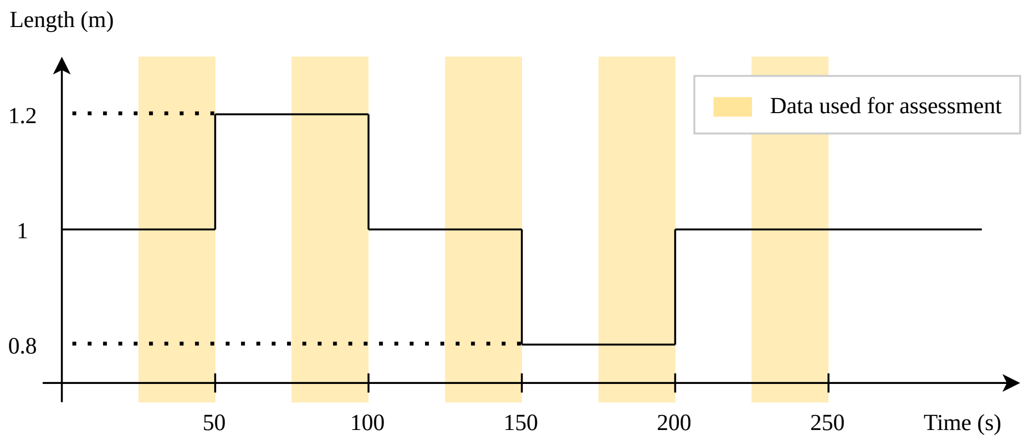

Figure 6.7: Diagram representation of the data used for the assessment. Pendulum length is updated every 50 seconds and data selected for assessment is inside yellow boxes.

6.2.2.1 Influence of relearning timings

In this section, the influence of the timing used for the Data Informativity compared to the one used for the GP retraining is assessed. Multiple timing combinations are presented in Figure 6.8 where t_{DI} is 1 or 5 seconds and t_{GP} is 1, 5, or 10 seconds. Each box plot contains 3 simulations. The threshold in IC is set to 0.1 rad and the minimum open loop interval is set to 10ms.

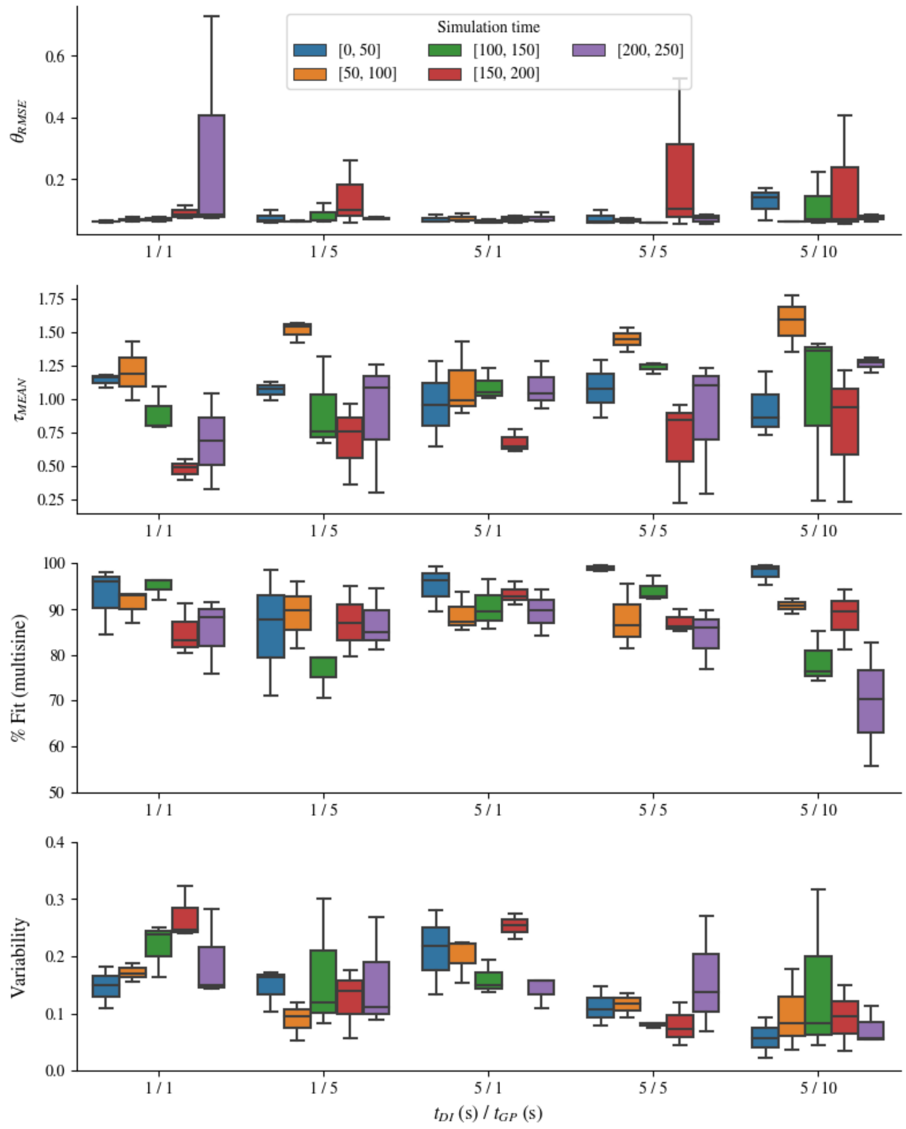

Figure 6.8: Assessment of the impact of the timings t_{DI} and t_{GP}. Top row is θ_{RMSE}, second row is τ_{MEAN}, third row is % fit using multisine signal and last row is the variability. The system (Single Inverted Pendulum) changes at 50, 100, 150 and 200 seconds as shown in Figure 6.7.

As shown in Figure 6.8, depending on the timing of t_{DI} and t_{GP}, it is possible to get different behaviors from the controller. When t_{DI} and t_{GP} are both set to 1 second, the variability of the simulation is the highest of all cases. In addition, θ_{RMSE} stays low and consistent across the set of the 3 simulations. However, when the pendulum length is going back to 1 meter at the end of the simulation, 1 of the 3 simulations is getting a higher error. Looking at τ_{MEAN}, the triggering is faster at the end of the simulation, which can then be seen as less variability, presented in the last row. When t_{DI} and t_{GP} are both equal to 5 seconds, even though both retraining are in sync like the previous case, the main difference is the overall variability in the 20-second window. This is due to the redesign of the GP which introduces a bump in the variability more often. This can be seen in Figure 5.23 and 5.24, for example, when t_{GP} is 1 or 10 seconds.

Two different tests have been run where t_{DI} is faster than t_{GP}: 1/5 and 5/10. Focusing on $t_{DI} = 1$ second and $t_{GP} = 5$ seconds, the overall θ_{RMSE} across different pendulum length is more consistent than when $t_{DI} = t_{GP}$. The τ_{MEAN} however, is in a similar range. The % fit when the pendulum is going back to 1 meter halfway through the simulation is reduced to around 77%, which can be seen also in the variability increasing due to the low accuracy in the modeling of the system's dynamics. When $t_{DI} = 5$ seconds and $t_{GP} = 10$ seconds, the % fit is the lowest of all different timing cases, which can be translated on the other measures. τ_{MEAN} spread is the widest of all, and the standard deviation of θ_{RMSE} is also large for most of the pendulum lengths.

Finally when t_{DI} is slower than t_{GP}, θ_{RMSE} is the most consistent between each simulation and across the different pendulum's lengths. The overall % fit is also consistent and τ_{MEAN} is around 1 second. It is getting lower when the pendulum length is decreased to 0.8 meters, due to the system being harder to control.

In conclusion, based on the desired behavior of IC, it is possible to change t_{DI} and t_{GP} accordingly. To ensure repeated behavior without the variability becoming too large, using a long t_{GP} is recommended. If variability is required, decreasing t_{GP} is having the most significant impact compared to t_{DI}. Changing t_{DI} does not have a strong impact, as long as it is quick enough to capture changes in the system matrices **A** and **B** to compute **k** in time before the system becomes unstable. Hence, keeping t_{DI} low and only varying t_{GP} is recommended.

6.2.2.2 Impact of IC parameters

In this section, the impact of Δ_{ol} and the threshold is assessed to understand the limit of the combined implementation of Data Informativity with Gaussian Processes. t_{DI} and t_{GP} are set to 5 and 10 seconds respectively. Δ_{ol} values are: 3ms, 10ms and 100ms and the threshold value is changing from 0.01 rad to 0.1 rad. The threshold is only applied to the angle of the pendulum.

For the results shown in Figures 6.9 and 6.10, the threshold is fixed at 0.01 rad and Δ_{ol} is varying between 3ms, 10ms, and 100ms. Figure 6.9 shows that by increasing the minimum open-loop interval, the θ_{RMSE} is getting lower, especially when the pendulum length is moving from 1 meter to 0.8 meters. Regarding the % fit, the GP can get to the correct system's dynamics no matter what the minimal open-loop interval is. Figure 6.10 indicates that it is also possible to get to the accurate system matrices **A** and **B** using a GP-based Hold. The values $A_{1,2}$ and B_1 are matching the target value, thus the computation of **k** using the LQR method is also accurate.

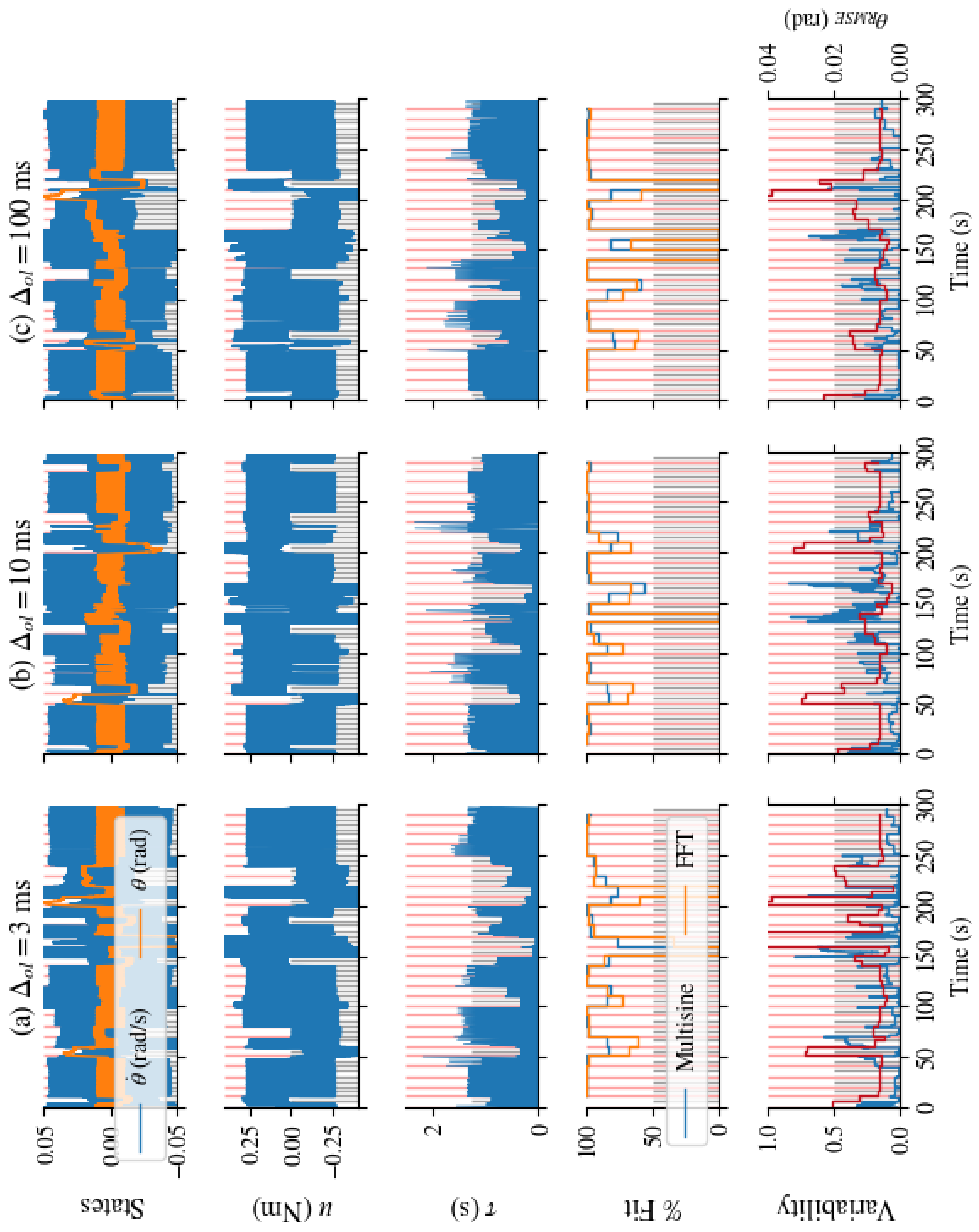

Figure 6.9: Influence of IC parameters applied to the Single inverted Pendulum. Threshold is 0.01 rad (angle). Different minimum open-loop: (a) $\Delta_{ol} = 3$ ms, (b) $\Delta_{ol} = 10$ ms and (c) $\Delta_{ol} = 100$ ms. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The third row is the open-loop interval and the fourth row is the GP's fit. The last row is the variability of θ (in grey) and θ_{RMSE} (in red). Vertical lines represents t_{DI} (grey) and t_{GP} (red). The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

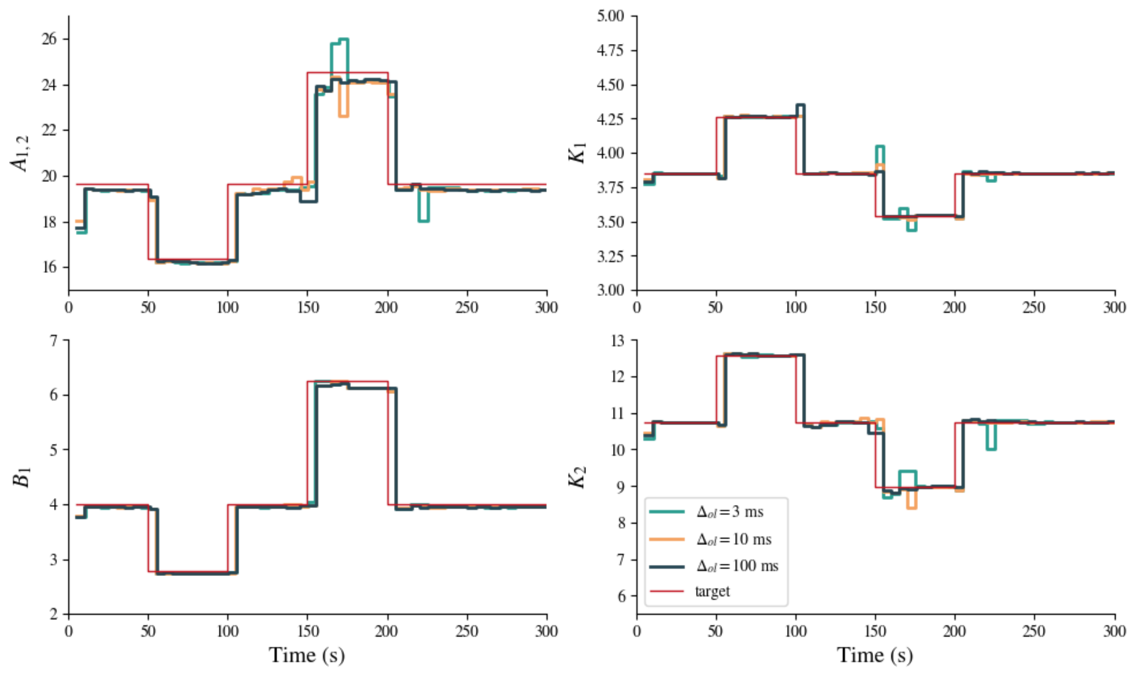

Figure 6.10: Influence of IC parameters applied to the Single inverted Pendulum. Threshold = 0.01 rad (angle). Different minimum open-loop: $\Delta_{ol} = 3$ ms (green), $\Delta_{ol} = 10$ ms (orange) and $\Delta_{ol} = 100$ ms (blue). Each subplots represents in order: $A_{1,2}$, K_1, B_1, and K_2. The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

Intermittent Control can keep the system stable as well as correctly estimating **A** and **B** when using a low threshold of 0.01 rad. Next, the threshold is increased to 0.1 rad, allowing more motion from the system. Similarly to the previous set of simulations, three different Δ_{ol} values are assessed: 3ms, 10ms, and 100ms. Results are shown in Figures 6.11 and 6.12.

Starting with $\Delta_{ol} = 3$ ms and $\Delta_{ol} = 10$ ms, both simulations are keeping the pendulum system stable. In addition, using the lowest Δ_{ol} is helping to improve the fit of the GP model when the system is changing length. Regarding the variability and the $\boldsymbol{\theta}_{RMSE}$, both of those measures stay in the same range. However, increasing Δ_{ol} to 100 ms is bringing instability when the pendulum is reducing its length to 0.8 meters. This is due to the coefficient $A_{1,2}$ not being estimated correctly, hence generating an unstable state feedback gain when passed to the LQR method, which can be seen in Figure 6.12. This results in

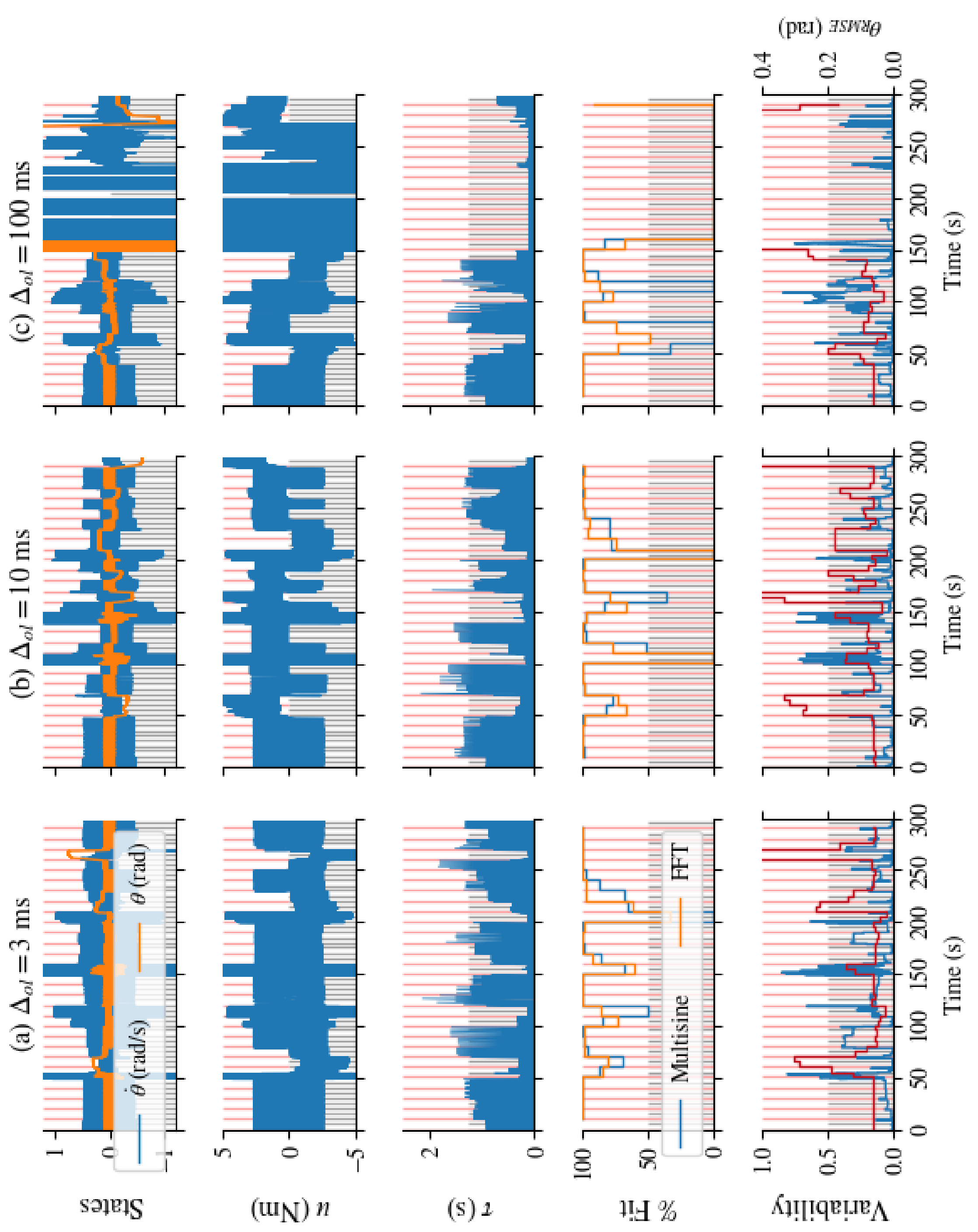

Figure 6.11: Influence of IC parameters applied to the Single inverted Pendulum. Threshold is 0.1 rad (angle). Different minimum open-loop: (a) $\Delta_{ol} = 3$ ms, (b) $\Delta_{ol} = 10$ ms and (c) $\Delta_{ol} = 100$ ms. The top row shows the two states of the Single Inverted Pendulum ($\dot{\theta}$ and θ). The second row is the control input with noise generated by a state feedback controller. The third row is the open-loop interval and the fourth row is the GP's fit. The last row is the variability of θ (in grey) and θ_{RMSE} (in red). Vertical lines represents t_{DI} (grey) and t_{GP} (red). The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

the GP not getting meaningful data to model the plant in the upright position, hence the % fit after 150 seconds is 0. The variability is close to 0 due to triggering at the minimum open-loop interval. Even when the pendulum length is returning to a length of 1 meter, the controller is not able to recover.

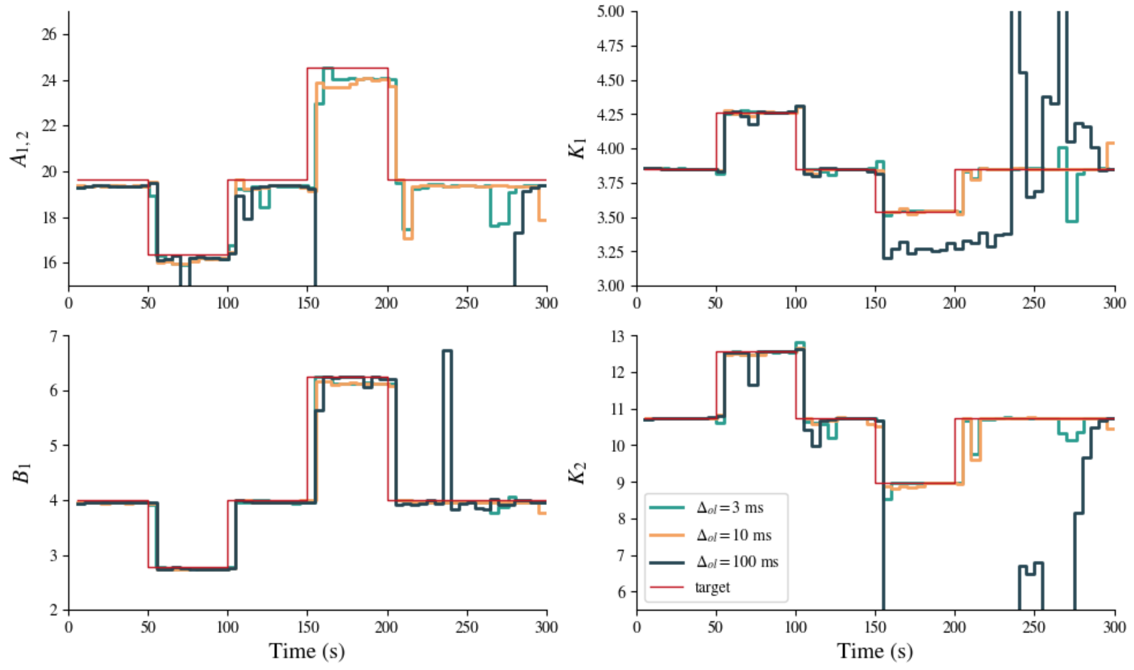

Figure 6.12: Influence of IC parameters applied to the Single inverted Pendulum. Threshold = 0.1 rad (angle). Different minimum open-loop: $\Delta_{ol} = 3$ ms (green), $\Delta_{ol} = 10$ ms (orange) and $\Delta_{ol} = 100$ ms (blue). Each subplots represents in order: $A_{1,2}$, K_1, B_1, and K_2. The system length is changing through the simulation: start at 1m, go to 1.2m at 50s, back to 1m at 100s, then lowered to 0.8 at 150s then back to 1m at 200s.

6.3 Discussion

In this chapter, GP-based Hold and state feedback design algorithms were used simultaneously. This is the first implementation of a fully data-driven probabilistic Intermittent Controller.

Initially, Reinforcement Learning and GPH were used and assessed to evaluate its capability to obtain the optimal state feedback online whilst keeping the plant stable. With results presented in Section 6.2.1, this implementation accurately estimate the corrrect state feedback, similar to System-matched Hold. In addition, in certain cases, GPH can keep the system stable due to its different response compared to SMH as shown in Figure 6.5, where initial poles are close to the imaginary axis. One limitation of this approach is the need to operate in clock-driven mode to avoid instability generated by long open-loop intervals with wrong state feedback gains. This is occurring when the initial control is designed with poles which are not close enough to the optimal ones. This is a limitation regarding the retraining of the GP as the amplitude of motion of the pendulum induced by this controller closely matches a Continuous Controller, hence not giving enough excitation to train the GP.

Next, Data Informativity and GPH were used together to control and estimate the system dynamics. This combination of algorithms was investigated with varying retraining timings as well as different Intermittent Control parameters such as the Δ_{ol} and the Threshold. By varying the timings t_{DI} and t_{GP}, it is possible to introduce more or less variability in the movement of the system while keeping the plant close to the set point. The retraining time also plays a role in the fit percentage of the GP model. One advantage of using Data Informativity is its capability to estimate the system matrices $\mathbf{A}$ and $\mathbf{B}$, unlike the Reinforcement Learning approach where only $\mathbf{k}$ is estimated. Where system dynamics are fully unknown, getting an accurate estimation of $\mathbf{k}$ can be quite difficult. However, the drift in the gains can be used as a proxy to assess an underlying time variant system. This could also be done by observing at the hyperparameters of the GP, however, the translation to a physical component is harder.

In conclusion, the two state feedback estimation approaches presented, Reinforcement learning and Data Informativity, can be used with a stochastic Intermittent Controller based on GPs. However, the Data Informativity approach is preferred due to its ease of use as well as full advantage of the open-loop trajectories present in the intermittent Controller.

6.4 Summary

An overview of the work accomplished in this Chapter is summarized in Table 6.1. This table focuses on the pros and cons of using GP with the RL and DI framework. In addition, limitations are also explained.

Algorithms	Pros	Cons	Limitations
RL + GP	Improved results compared to SMH: states estimation able to converge where initials poles are close to imaginary axis	Not able to use data sample generated during open-loop trajectories	IC in Clock driven mode recommended to ensure fix triggering to gather data in time for next optimisation
DI + GP	Able to introduce variability from the control input compared to SMH + DI. Able to use open-loop generated data	Combination of GP with DI might result in instability in especially with large minimum open-loop interval and high threshold	Limited by the speed of the GP optimisation

Table 6.1: Summary table covering Chapter 6 contributions.

Some generalisation can be made for each algorithm. Similarly to RL with SMH, RL with GP can be used to determine the approximated location of the optimal poles if the estimation does not converge. Regarding using DI with GP, keeping a small minimum open-loop interval is beneficial where a high threshold is required.

Chapter 7

Discussion

The discussion is separated into two sections. First, the introduction of stochastic elements into the Intermittent Control framework by using Gaussian Processes is discussed by using Gaussian Processes, followed by an evaluation of the full data-driven implementation of IC. Figure 7.1 is a summary diagram representation of all algorithms used within the intermittent controller during this project.

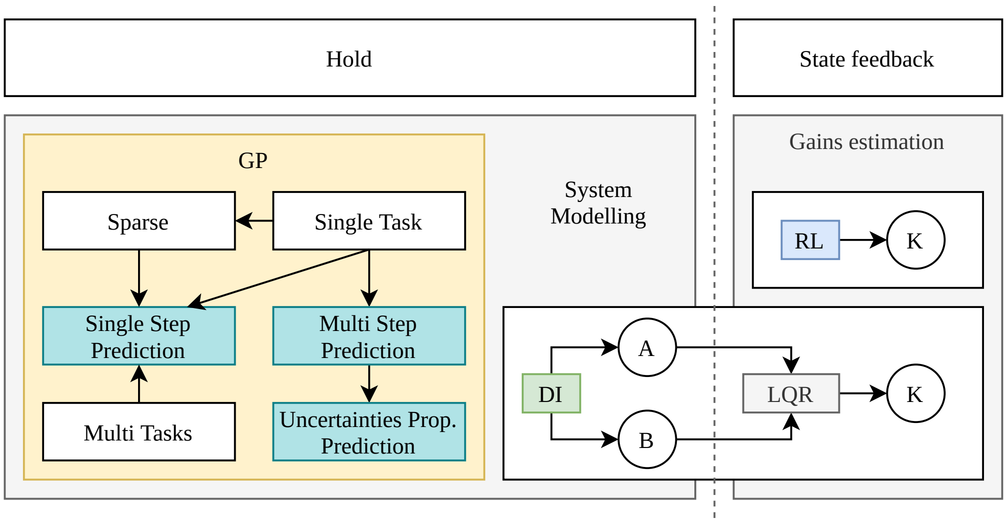

Figure 7.1: Summary diagram covering this work within the IC framework. GP: Gaussian Processes. DI: Data Informativity. RL: Reinforcement Learning. A and B are the system matrices. K is the state feedback gain vector. In GP, blue represents different type of prediction, white is the type of GP.

7.1 GP as non-linear probabilistic based Hold

In the implementation originally presented by Gawthrop and Wang 2007, the hold present in the Intermittent Controller is based on a deterministic representation of the closed-loop system dynamics. This hold is directly derived from the system itself, hence its name of System-Matched Hold. However, this approach relies on knowing the system's equations to be able to implement this hold. As presented in Chapter 5, Gaussian Processes can be used in place of the hold, to integrate a stochastic element into the intermittent controller. Due to their probabilistic nature, GPs can generate open-loop trajectories that are not deterministic, hence acting as a new source of variability into the control. Note that in Continuous Control, it would be required to add a disturbance signal such as noise to generate such trajectories.

Gaussian process modeling uses system's data to optimize its hyperparameters. However, when using data coming from a continuous controller, the GP is modeling the closed-loop dynamics with the influence of the controller. Although this representation can be used in the Intermittent Controller's Hold as such, this model is only able to generate closed-loop trajectories whereas a GP trained based on Intermittent Controller generated data is modeling an open-loop representation. In the current implementation, GP models can be based on different types of GP such as single-task or multi-task (Bonilla et al. 2007).

Even though it is necessary to get a system model before starting any controller, GP has the capability of using online data for retraining, hence improving model representation gradually throughout the simulation. Chapter 5 shows simulation results of this approach. GP can take advantage of the open-loop trajectories generated by the intermittent controller to gather data around the operating target, allowing exploration and exploitation. The number of points used for retraining as well as the history of points have an impact on the overall performance of the controller. In the case of adaptation to a system which changes over time, it is important to not mix training data that are part of two different systems for the GP to model. A forgetting factor has been introduced to help get an accurate GP, even when the system is changing during the simulation. This can be seen as a short-term memory parameter when compared to human motor control.

In addition to being able to control a system with a GP-based IC, it is also possible to generate different types of control by simply modifying some GP parameters, similar to modifying IC parameters. For example, as the GP relies on training data to model the system dynamics, changing the number of training points has an impact on the overall model as well as the computational effort required for the optimization as presented in Figure 5.18. Moreover, enabling uncertainty propagation is allowing any perfect GP to generate trajectories that drift away from the traditional SMH representation (Figure 5.11), allowing more variability in the control. Additionally, it is possible to use the prediction uncertainty from the GP to modify the triggering mechanism: this allows a shorter time between events during the learning process when the uncertainty of the GP is growing faster.

Gaussian processes present many advantages compared to the traditional SMH implementation, by bringing non-linearity as well as probabilistic trajectories generation. The non-linearity can be considered as an advantage as it helps to model system dynamics outside the linear range. However, there are still some limitations when using this implementation in a real-time application, such as the computational burden of this approach. As the number of training data is increasing, the computational cost is growing as well. In addition, the computation of the GP optimization in the adaptation case is also an issue. Depending on the amount of data, the GP optimization can take multiple seconds, which makes it unsuitable for real-time applications. Finally, this approach is only able to update the hold present in IC but it is also required to update the state feedback gains accordingly.

7.2 Data Driven Intermittent Controller

As discussed above, Gaussian Processes were implemented as a hold alternative to the traditional SMH. This approach is helping move towards a data-driven intermittent controller. However, the state feedback is also required to be known, and updated in the case of adaptation to ensure stability of the plant. To estimate the state feedback gains required by the system, two approaches have been compared: (a) a direct approach, using Reinforcement Learning, and (b) an indirect approach, using Data Informativity. Other Indirect approaches have previously been evaluated within Intermittent Control, such as Kalman Filters (Martín 2018).

As presented in Chapter 4, Reinforcement Learning and Data Informativity are both able to estimate the correct state feedback gains for the controller. However, both approaches are different in the way of computing it: Reinforcement Learning requires a stable initial controller whereas Data Informativity does not. As these approaches have been designed to be used with Continuous Controllers, adaptation to the IC framework was required. Reinforcement Learning is only able to use data when an event occurs. Operating IC in clock-driven mode is helping to ensure getting data for the algorithm in a fixed amount of time. However, as results show in Section 4.2.1, keeping open-loop intervals small is required for stability. Two main comments can be made: (i) when using the RL algorithm, all data generated when the IC is operating in an open-loop fashion is not used, and (ii) the IC is not operating with long open-loop intervals, hence the system response is closed to a continuous controller, without much motion introduced.

On the other hand, Data Informativity has shown advantages when used with Intermittent over Continuous Control. As no excitation is present in CC without adding an external disturbance signal, Data Informativity is not able to perform accurately. In addition, when tested with some added noise, the estimation of the system matrices **A** and **B** were inaccurate. However, adding a multi-sine signal to provide external excitation was sufficient to improve accuracy. When used with IC simulation data, the DI algorithm can accurate estimation of the system dynamics, represented by $\mathbf{A}$ and $\mathbf{B}$ without the need for additional disturbance signals. The intrinsic motion generated by the controller itself is enough to converge to the correct matrices while not forcing it to disturb the regular operating mode of the system to be able to estimate accurately its dynamics.

In addition, by occluding events and only using open-loop trajectory data, the algorithm can get closer to the true value of the plant. Compared to the RL algorithm, DI can take full advantage of open-loop trajectories generated by IC to improve the matrix estimation. It is also possible to adjust the time t_{DI} to improve the stability of the matrix estimation. Reducing this value below the open-loop interval of the IC forces an update of the state feedback at each event. Whilst this brings some variability in the response, it can also cause some instability, especially when the minimum open-loop interval and threshold in IC are set to large values.

As presented in Chapter 6, both RL and DI can be used with the GP-based hold. Focusing on RL and GP, the improvement over SMH is quite small due to the need to use clock-driven mode. This implementation can get to the optimal state feedback gains, however, convergence depends on the initial controller pole location. In addition, adaptation with RL and GP is not covered due to the lack of excitation of the system for GP retraining.

When combining DI and GP, results show great potential, especially when adapting to a changing system; both GPs and estimated system matrices are reaching the optimal values. By varying both t_{DI} and t_{GP}, it is possible to introduce more or less variability into the simulation. This is also having an impact on the GP modeling. However, these two timings need to be matched to the IC parameters to ensure the stability of the estimation, especially when the minimum open-loop interval is set to a large value.

In conclusion, both RL and DI can get to the correct state feedback gains, in combination with a GP-based Hold. However, due to the current RL algorithm limitation of operating in clock-driven mode, the recommended approach is to use Data Informativity to take full advantage of open-loop trajectories generated by the Intermittent Controller.

Chapter 8

Conclusion and Future work

This chapter summarizes the work achieved during this PhD. It is divided into three sections: conclusion, limitations of the current work, and future work.

8.1 Conclusion

The main aim of this research was to investigate the potential of data-driven techniques within the intermittent control framework for rehabilitation and engineering systems purposes. The results presented in this thesis can be summarized as follows:

- Using a Gaussian Process-based Hold in Intermittent Control, it is possible to bring different open-loop dynamics compared to the traditional System-Matched Hold without the need for external disturbances. Changing the GP parameters allows to switch between symmetric repeated behavior, similar to SMH in the noise-free case, to more variability in the signal when uncertainty propagation is enabled. In addition, GP can model open-loop dynamics even though it is using data generated with a controller.

- Whilst Reinforcement Learning is quite limited when using it with Intermittent Control, Data Informativity has shown its capabilities in estimating accurate state feedback gains by using intermittent control generated trajectories. This algorithm is also capable of handling cases where the system's parameters are varying with time.
- When combining the GP and DI algorithms, the intermittent controller is fully data-driven. Results show the capabilities of such a controller to cope with adaptation. In addition, tuning IC or GP parameters can help switch the control input from a deterministic to a stochastic one, by generating probabilistic open-loop trajectories.

As an overall conclusion, fully data-driven stochastic intermittent control could help in understanding learning mechanisms in a human motor control context due to its probabilistic nature. This implementation can also apply to engineering systems, where adaptation is prioritized in relation to accuracy.

8.2 Limitations

The current implementation of the Non-linear Probabilistic model-based Data-driven Intermittent Controller has some limitations as presented below:

- Real-time implementation: With the introduction of Gaussian Processes as an alternative to the System Matched Hold, real-time issues were introduced. This is mainly due to two issues: (a) optimization of the Gaussian Processes online that cannot be achieved in less than one discrete time iteration due to the computational complexity, and (b) the prediction of the next iteration, where it is necessary to compute the inverse of a matrix for each of the GP. The second reason cannot be solved easily as discussed in Chapter 5.2.1.2 by only introducing previous knowledge of the system or using Multi-Task GP. This issue can be addressed by using faster hardware and/or using a different programming language which is more suitable for real-time application. However, improving the optimisation of the GP has not been looked as part of this thesis.

- Partial non-linear controller: the original Intermittent Controller is based on linear system equations, hence all tracking and regulation equations are using linear algebra, similarly to the computation of the state feedback using the LQR method. With the approach presented in Chapter 5, the use of Gaussian processes is only partially introducing non-linearity into the Intermittent Controller Framework. Even if the use of such a hold is helping to capture dynamics of the plant more accurately, the state feedback gain is only accurate for a small range around the linearisation.
- System Identification with noisy data: As shown in Chapters 4 and 6, Data Informativity and Reinforcement Learning Frameworks both have difficulty in estimating the correct state feedback gain if noise is introduced into the simulation. Similarly to the noiseless case, this framework can estimate the system matrices $\mathbf{A}$ and $\mathbf{B}$ in contrast to the Reinforcement Learning approach, where only the state feedback k is estimated, hence not being feasible as the only relearning mechanism.
- Reinforcement Learning algorithm: The currently implemented algorithm is only able to use data at the time of an event. This approach does not use the full potential of the trajectories generated by the open-loop behavior of the intermittent controller.
- The current work presented in this thesis has been applied to simple dynamical systems such as the Single Inverted Pendulum and the Cartpole system. Even though this approach has only been apply to two and four state system, the methods shown here be easy to apply to higher order systems.

8.3 Future work

In this section, potential follow-up ideas are explored to improve and overcome the limitations listed above:

- Real-time application of data-driven IC
- Full non-linear Intermittent Controller
- Noisy data and Data Informativity with Intermittent Control
- Adaptation with RL and GP

8.3.1 Real-time application of data-driven IC

Real-time application is not feasible with the current implementation of the algorithm. One possible approach would be to use some sort of pipelining, where the GP is being optimized at the same time that the simulation is running to keep the system stable. Once the GP is optimized, it can be substituted into the hold for the rest of the simulation. However, if the system is under unstable conditions, the GP rapidly is very important, to avoid bringing the system into an unstable area. Using the proposed approach to a real-life case study is feasible with the current state of technology. In addition, improving the computation time for each state prediction from the GP should also be investigated. Advances in computer performance, such as using GPU could contribute to reaching a real-time application.

8.3.2 Full non-linear IC

The Intermittent Controller is based on an underlying continuous controller. Even if the GP can represent the entire dynamics of the system, for example, a full rotation of the pendulum, all controller components are based on a linear control: state feedback and a triggering mechanism. One possible avenue is to create an intermittent implementation of the PILCO algorithm by Deisenroth and Rasmussen 2011.

8.3.3 Noisy data and DI with IC

The current implementation of the Data Informativity framework is only based on the noiseless case. However, a new implementation by Van Waarde et al. 2023 can handle noisy data. It could improve the estimation of the current noise-free implementation in Intermittent Control when input noise is present.

8.3.4 Adaptation with RL and GP

The current implementation of the RL algorithm can only gather data when an event in IC occurs. However, using the open-loop data generated by the Intermittent Controllercan be beneficial, similar to the Data Informativity case when used compared to a Continuous Controller. This modification in the algorithm could also be beneficial for the GP as the system's motions are required to get an accurate representation of the system's dynamics. Currently, operating in clock-driven mode is recommended, hence system states closely match those of a Continuous Controller.

Appendices

Appendix A

Single Inverted Pendulum system

The single inverted pendulum model is a traditional example in control systems and it is widely used to illustrate the results and the implementation details of many control strategies. In addition, some authors have used this model to describe the human balance control problem, contrasting simulation and experimental data (Loram et al. 2009; Nomura et al. 2013). For the purpose of this model, the dynamic bias model of human standing described in (Lakie et al. 2003; Loram et al. 2005; Loram et al. 2009; Gawthrop et al. 2011) is used. This model considers that the control signal that is applied to maintain a human inverted pendulum balanced is generated by a tendon that is connected in series with a contractile element (in this case the calf muscle) which is in charge of generating a torque. The equation of motion of the pendulum is as follows

$$J\ddot{\theta} = mgh\sin(\theta) + T\,, \tag{A.1}$$

where J is the moment of inertia, θ is the angular position with respect to the vertical, m is the mass of the pendulum, g is the gravitational acceleration, and h is the distance from the joint to the centre of mass. The small angle approximation $\theta = \sin(\theta)$ is used to simplify the equation to a linear model. The ankle torque T is defined as

$$T = -cmgh(\theta - \theta_0) - V\dot{\theta}\,, \tag{A.2}$$

with c being the ratio between the tendon stiffness k_p, the load stiffness defined by the product $k_e = mgh$, and V is the ankle viscosity. The input is provided by θ_0 (known as the bias) which represents the active muscle shortening in angular terms. Considering the pendulum angle θ (in radians) and the bias θ_0 as output and input respectively, the model can be written as a transfer function, resulting in

$$\theta = \frac{\frac{cmgh}{J}}{s^2 + \frac{V}{J}s + (c-1)\frac{mgh}{J}}\theta_0 \,. \tag{A.3}$$

as reported in Loram et al. 2009. The fact that c is smaller than 1 implies that the tendon stiffness is not enough to stabilise the pendulum on its own, requiring additional control effort provided by the muscle. In ibid., this model was implemented in simulation, where the input θ_0 was provided by a subject holding a joystick, thus θ_0 was proportional to the motion of the joystick. In Fig. A.1 a simple diagram of the inverted pendulum is presented.

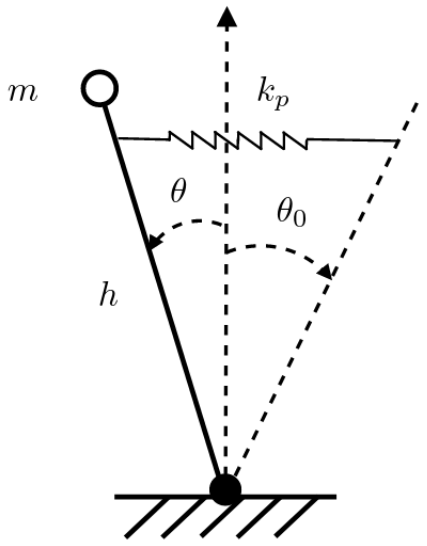

Figure A.1: Inverted pendulum model of human standing. The input θ_0 represents the contraction of the muscle as an angle that influences the pendulum via a spring of stiffness k_p, which represents the ankle joint tendon. The output θ is the angle of the pendulum with respect of the vertical line, m is the mass, and h is the distance from the joint to the centre of mass.

A Non-linear equations

One way to simulate dynamical systems is to use a numerical integration algorithm such as the Runge-Kutta method (Dormand and Prince 1980) to solve the differential equations that describe the dynamics. In order to do this, we can substitute (A.2) in (A.1) as follows to obtain a suitable form for these type of algorithms:

$$\ddot{\theta} = \frac{mgh\sin(\theta) - cmgh(\theta - \theta_0) - V\dot{\theta}}{J} \,. \tag{A.4}$$

Knowing that θ_0 is the control input to the system, we can replace it with $\mathbf{u}$ to match common control engineering terminology. Also, if we use the following assignments: $\dot{\theta} = \mathbf{x}_0$ and $\theta = \mathbf{x}_1$, we can re-write expression (A.4) to get

$$\begin{aligned} \dot{\mathbf{x}}_0 &= \frac{mgh\sin(\mathbf{x}_1) - cmgh(\mathbf{x}_1 - \mathbf{u}) - V\mathbf{x}_0}{J} \\ \dot{\mathbf{x}}_1 &= \mathbf{x}_0 \,. \end{aligned} \tag{A.5}$$

Expression (A.5) can now be implemented in a simulation environment to be solved using numerical integration.

B State-space representation

Consider the following linear dynamical system description of order n

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \,, \end{aligned} \tag{A.6}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^{n_y}$ and $\mathbf{u} \in \mathbb{R}^{n_u}$ correspond to the system state, output and input respectively, and t represents continuous time. $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{B}$ and $\mathbf{B}_d$ are $n \times n_u$, and $\mathbf{C}$ is $n_y \times n$. If we define the state vector $\mathbf{x}(t) = \begin{bmatrix} \dot{\theta} & \theta \end{bmatrix}^T$, where $\dot{\theta}$ is the angular velocity, and use the parameters of the inverted pendulum described in the previous section, the transfer function in (A.3) can be written as state-space model as follows

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -\frac{V}{J} & \frac{(1-c)mgh}{J} \\ 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} \frac{cmgh}{J} \\ 0 \end{bmatrix} \mathbf{u}(t) \tag{A.7}$$

$$\mathbf{y} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}(t)\,. \tag{A.8}$$

Using the values for the following parameters $V = 2.9$, $c = 0.85$, $m = 70$ kg, $h = 92$cm, $g = 9.81$ and $J = 77$, the final state-space model is:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -0.037 & 1.231 \\ 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 6.98 \\ 0 \end{bmatrix} \mathbf{u}(t) \tag{A.9}$$

$$\mathbf{y} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}(t)\,. \tag{A.10}$$

Appendix B

Cartpole system

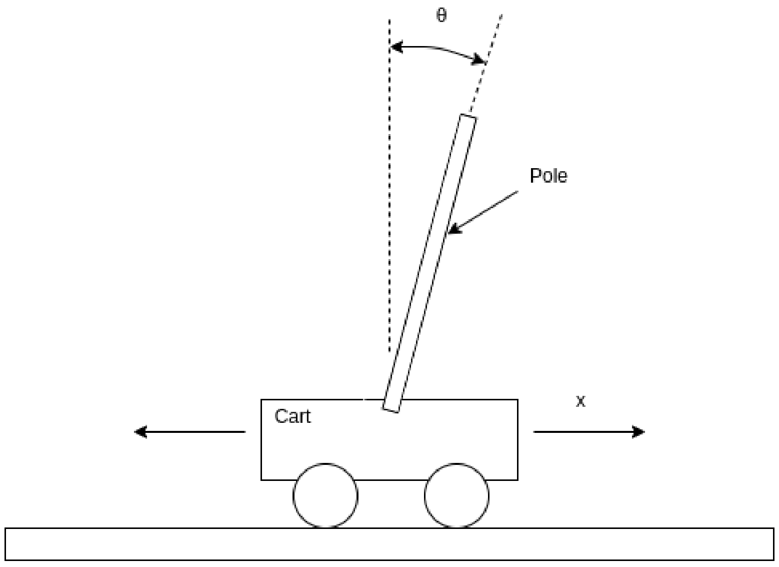

Figure B.1: Cartpole system model use for simulation. The cart can only move to the right or left (x axis) to keep the pole balanced.

A Non-linear equations

Similar to the Single Inverted Pendulum system, the simulation of the cartpole system is based on the Runge-Kutta method (Dormand and Prince 1980) to solve the differential equations that describe the dynamics. This system is based on the following non-linear equations:

$$\dot{\mathbf{x}}_0 = \mathbf{x}_2$$

$$\dot{\mathbf{x}}_1 = \mathbf{x}_3$$

$$\dot{\mathbf{x}}_2 = \frac{\begin{array}{r} -(I_p + M_p l_p^2) B_{eq} \mathbf{x}_2 - (M_p^2 l_p^3 + I_p M_p l_p) * \sin(\mathbf{x}_1) \mathbf{x}_3^2 - M_p l_p \cos(\mathbf{x}_1) B_p \mathbf{x}_3 \\ + (I_p + M_p l_p^2) u + M_p^2 l_p^2 g \cos(\mathbf{x}_1) \sin(\mathbf{x}_1) \end{array}}{(M_c + M_p) I_p + M_c M_p l_p^2 + M_p^2 l_p^2 \sin^2(\mathbf{x}_1)} \tag{B.1}$$

$$\dot{\mathbf{x}}_3 = \frac{\begin{array}{r} (M_c + M_p) M_p g l_p \sin(\mathbf{x}_1) - (M_c + M_p) B_p \mathbf{x}_3 - M_p^2 l_p^2 \sin(\mathbf{x}_1) \cos(\mathbf{x}_1) \mathbf{x}_3^2 \\ - M_p l_p \cos(\mathbf{x}_1) B_{eq} \mathbf{x}_2 + M_p l_p \cos(\mathbf{x}_1) u \end{array}}{(M_c + M_p) I_p + M_c M_p l_p^2 + M_p^2 l_p^2 \sin^2(\mathbf{x}_1)}$$

Expression (B.1) can now be implemented in a simulation environment to be solved using numerical integration.

B State-space representation

Consider the following linear dynamical system description of order n

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t)\,, \end{aligned} \tag{B.2}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^{n_y}$ and $\mathbf{u} \in \mathbb{R}^{n_u}$ correspond to the system state, output and input respectively, and t represents continuous time. $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{B}$ and $\mathbf{B}_d$ are $n \times n_u$, and $\mathbf{C}$ is $n_y \times n$. If we define the state vector $\mathbf{x}(t) = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^T$, where $\dot{x}$ is the velocity of the cart and $\dot{\theta}$ is the angular velocity of the pendulum, the state-space model as follows:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{M_p^2 l_p^2 g}{\Delta} & \frac{-B_{eq}(I_p + M_p l_p^2)}{\Delta} & \frac{-M_p l_p B_p}{\Delta} \\ 0 & \frac{M_p l_p g (M_p + M_c)}{\Delta} & \frac{-B_{eq} M_p l_p}{\Delta} & \frac{-B_p (M_p + M_c)}{\Delta} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ \frac{I_p M_p l_p^2}{\Delta} \\ \frac{M_p l_p}{\Delta} \end{bmatrix} \mathbf{u}(t) \tag{B.3}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}(t)\,. \tag{B.4}$$

Using the values of the following parameters $B_{eq} = 4.3$, $B_p = 0.0024$, $I_p = 0.0079$, $M_p = 0.23$ Kg, $M_c = 0.7031$ Kg, $l_p = 0.3302$ m and $g = 9.81$ m.s^{-2} and $\Delta = (I_p M_c + M_c M_p l_p^2 + M_p I_p) = 0.025$, the final state-space model is:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 2.2629 & -5.6713 & 0.0072 \\ 0 & 27.8037 & -13.0609 & -0.0895 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1.3189 \\ 3.0374 \end{bmatrix} \mathbf{u}(t) \tag{B.5}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}(t)\,. \tag{B.6}$$

Bibliography

Alonso-Sanchez, Francisco and David Hochberg (Nov. 2000). ‘Renormalization Group Analysis of a Quivering String Model of Posture Control’. In: *Physical Review E* 62.5, pp. 7008–7023. ISSN: 1063-651X, 1095-3787. DOI: `10.1103/PhysRevE.62.7008`. arXiv: `cond-mat/0007400`.

Anderson, Brian D O and Arvin Dehghani (2008). ‘Challenges of Adaptive Control–Past, Permanent and Future’. In: *Annual Reviews in Control*.

Anderson, Brian D. O. and John B. Moore (1990). *Optimal Control: Linear Quadratic Methods*. USA: Prentice-Hall, Inc. ISBN: 0136385605.

Åström, Karl J and Björn Wittenmark (2013). *Adaptive control*. Courier Corporation.

Åström, Karl-Johan and Bohlin Torsten (Sept. 1965). ‘Numerical Identification of Linear Dynamic Systems from Normal Operating Records’. In: *IFAC Proceedings Volumes* 2.2, pp. 96–111. ISSN: 14746670. DOI: `10.1016/S1474-6670(17)69024-4`.

Bai, E.W. and S.S. Sastry (Aug. 1985). ‘Persistency of Excitation, Sufficient Richness and Parameter Convergence in Discrete Time Adaptive Control’. In: *Systems & Control Letters* 6.3, pp. 153–163. ISSN: 01676911. DOI: `10.1016/0167-6911(85)90035-0`.

Bonilla, Edwin V, Kian M Chai and Christopher Williams (2007). ‘Multi-Task Gaussian Process Prediction’. In: *Advances in neural information processing systems* 20.

Bottaro, Alessandra et al. (Aug. 2005). ‘Body Sway during Quiet Standing: Is It the Residual Chattering of an Intermittent Stabilization Process?’ In: *Human Movement Science* 24.4, pp. 588–615. ISSN: 01679457. DOI: `10.1016/j.humov.2005.07.006`.

Brosilow, Coleman and Babu Joseph (2002). *Techniques of Model-Based Control.* Prentice-Hall International Series in the Physical and Chemical Engineering Sciences. Upper Saddle River, N.J: Prentice Hall. ISBN: 978-0-13-028078-7.

Burns, Stephen P. et al. (Nov. 1997). 'Recovery of Ambulation in Motor-Incomplete Tetraplegia'. In: *Archives of Physical Medicine and Rehabilitation* 78.11, pp. 1169–1172. ISSN: 00039993. DOI: 10.1016/S0003-9993(97)90326-9.

Carrillo, F.J., A. Baysse and A. Habbadi (2009). 'Output Error Identification Algorithms for Continuous-Time Systems Operating in Closed-Loop'. In: *IFAC Proceedings Volumes* 42.10, pp. 408–413. ISSN: 14746670. DOI: 10.3182/20090706-3-FR-2004.00067.

Caruana, Rich (1997). 'Multitask learning'. In: *Machine learning* 28, pp. 41–75.

Chai, Kian M (2009). 'Generalization errors and learning curves for regression with multi-task Gaussian processes'. In: *Advances in neural information processing systems* 22.

Cotoros, Diana and Mihaela Baritz (2010). 'Biomechanical analyzes of human body stability and equilibrium'. In: *Proceedings of the World Congress on Engineering.* Vol. 2.

Craik, Kenneth J. W. (1947). 'Theory of the human operator in control systems'. In: *British Journal of Psychology. General Section* 38.2, pp. 56–61. ISSN: 0373-2460. DOI: 10.1111/j.2044-8295.1947.tb01141.x.

Cutler, Mark and Jonathan P. How (May 2015). 'Efficient Reinforcement Learning for Robots Using Informative Simulated Priors'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA).* Seattle, WA, USA: IEEE, pp. 2605–2612. ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139550.

Deisenroth, Marc and Carl E Rasmussen (2011). 'PILCO: A model-based and data-efficient approach to policy search'. In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472.

Deisenroth, Marc Peter (2010). *Efficient reinforcement learning using Gaussian processes.* Vol. 9. KIT Scientific Publishing.

Deisenroth, Marc Peter, Dieter Fox and Carl Edward Rasmussen (Feb. 2015). 'Gaussian Processes for Data-Efficient Learning in Robotics and Control'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2, pp. 408–423. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2013.218. arXiv: 1502.02860 [cs, stat].

Dormand, John R and Peter J Prince (1980). ‘A family of embedded Runge-Kutta formulae’. In: *Journal of computational and applied mathematics* 6.1, pp. 19–26.

Durichen, Robert et al. (Jan. 2015). ‘Multitask Gaussian Processes for Multivariate Physiological Time-Series Analysis’. In: *IEEE Transactions on Biomedical Engineering* 62.1, pp. 314–322. ISSN: 0018-9294, 1558-2531. DOI: `10.1109/TBME.2014.2351376`.

Duvenaud, David et al. (May 2013). *Structure Discovery in Nonparametric Regression through Compositional Kernel Search*. arXiv: `1302.4922 [cs, stat]`.

Eleftheriadis, Stefanos et al. (2017). ‘Identification of Gaussian Process State Space Models’. In: *Advances in neural information processing systems* 30.

Engel, Yaakov, Shie Mannor and Ron Meir (2003). ‘Bayes meets Bellman: The Gaussian process approach to temporal difference learning’. In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 154–161.

– (2005). ‘Reinforcement learning with Gaussian processes’. In: *Proceedings of the 22nd international conference on Machine learning*, pp. 201–208.

Faisal, A. Aldo, Luc P. J. Selen and Daniel M. Wolpert (Apr. 2008). ‘Noise in the Nervous System’. In: *Nature Reviews Neuroscience* 9.4, pp. 292–303. ISSN: 1471-003X, 1471-0048. DOI: `10.1038/nrn2258`.

Gaskett, Chris, David Wettergreen and Alexander Zelinsky (1999). ‘Q-Learning in Continuous State and Action Spaces’. In: *Advanced Topics in Artificial Intelligence*. Ed. by G. Goos et al. Vol. 1747. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 417–428. ISBN: 978-3-540-66822-0 978-3-540-46695-6. DOI: `10.1007/3-540-46695-9_35`.

Gatev, Plamen et al. (Feb. 1999). ‘Feedforward Ankle Strategy of Balance during Quiet Stance in Adults’. In: *The Journal of Physiology* 514.3, pp. 915–928. ISSN: 0022-3751, 1469-7793. DOI: `10.1111/j.1469-7793.1999.915ad.x`.

Gawthrop, P. J. (1977). ‘Studies in identification and control’. PhD thesis. Oxford University.

Gawthrop, P J (Aug. 2009). ‘Frequency-Domain Analysis of Intermittent Control’. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 223.5, pp. 591–603. ISSN: 0959-6518, 2041-3041. DOI: `10.1243/09596518JSCE759`.

Gawthrop, P. J. and L Wang (Nov. 2007). 'Intermittent Model Predictive Control'. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 221.7, pp. 1007–1018. ISSN: 0959-6518, 2041-3041. DOI: `10.1243/09596518JSCE417`.

Gawthrop, Peter, Henrik Gollee and Ian Loram (July 2014a). *Intermittent Control in Man and Machine*. arXiv: `1407.3543 [cs, q-bio]`.

Gawthrop, Peter and Liuping Wang (Dec. 2011). 'The System-Matched Hold and the Intermittent Control Separation Principle'. In: *International Journal of Control* 84.12, pp. 1965–1974. ISSN: 0020-7179, 1366-5820. DOI: `10.1080/00207179.2011.630759`.

Gawthrop, Peter et al. (Feb. 2011). 'Intermittent Control: A Computational Theory of Human Control'. In: *Biological Cybernetics* 104.1-2, pp. 31–51. ISSN: 0340-1200, 1432-0770. DOI: `10.1007/s00422-010-0416-4`.

Gawthrop, Peter et al. (Dec. 2013). 'Human Stick Balancing: An Intermittent Control Explanation'. In: *Biological Cybernetics* 107.6, pp. 637–652. ISSN: 0340-1200, 1432-0770. DOI: `10.1007/s00422-013-0564-4`.

Gawthrop, Peter et al. (Apr. 2014b). 'Intermittent Control Models of Human Standing: Similarities and Differences'. In: *Biological Cybernetics* 108.2, pp. 159–168. ISSN: 0340-1200, 1432-0770. DOI: `10.1007/s00422-014-0587-5`.

Gawthrop, Peter J and Henrik Gollee (Oct. 2012). 'Intermittent Tapping Control'. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 226.9, pp. 1262–1273. ISSN: 0959-6518, 2041-3041. DOI: `10.1177/0959651812450114`.

Gawthrop, Peter J. and Liuping Wang (Dec. 2009). 'Event-Driven Intermittent Control'. In: *International Journal of Control* 82.12, pp. 2235–2248. ISSN: 0020-7179, 1366-5820. DOI: `10.1080/00207170902978115`.

Gevers, Michel (2002). 'Modelling, Identification and Control'. In: *Identification and Control*. DOI: `https://doi.org/10.1007/978-1-4471-0205-2_1`.

Ghavamzadeh, Mohammad et al. (2015). 'Bayesian Reinforcement Learning: A Survey'. In: *Foundations and Trends® in Machine Learning* 8.5-6, pp. 359–483. ISSN: 1935-8237, 1935-8245. DOI: `10.1561/2200000049`. arXiv: `1609.04436 [cs, stat]`.

Girard, Agathe (2004). 'Approximate methods for propagation of uncertainty with Gaussian process models'. PhD thesis.

Gollee, Henrik et al. (Nov. 2017). 'Visuo-manual Tracking: Does Intermittent Control with Aperiodic Sampling Explain Linear Power and Non-linear Remnant without Sensorimotor Noise?' In: *The Journal of Physiology* 595.21, pp. 6751–6770. ISSN: 0022-3751, 1469-7793. DOI: 10.1113/JP274288.

Goodwin, G. and Eam Khwang Teoh (June 1985). 'Persistency of Excitation in the Presence of Possibly Unbounded Signals'. In: *IEEE Transactions on Automatic Control* 30.6, pp. 595–597. ISSN: 0018-9286. DOI: 10.1109/TAC.1985.1103997.

Goodwin, Graham Clifford, Stefan F Graebe, Mario E Salgado et al. (2001). *Control system design*. Vol. 240. Prentice Hall Upper Saddle River.

Grancharova, Alexandra et al. (2023). 'Distributed predictive control based on Gaussian process models'. In: *Automatica* 149, p. 110807.

Green, Michael and John B. Moore (Sept. 1986). 'Persistence of Excitation in Linear Systems'. In: *Systems & Control Letters* 7.5, pp. 351–360. ISSN: 01676911. DOI: 10.1016/0167-6911(86)90052-6.

Guo, Yuan et al. (2019). 'Data-Driven Model-Free Adaptive Predictive Control for a Class of MIMO Nonlinear Discrete-Time Systems With Stability Analysis'. In: 7.

Hardy, Jay H., Eric Anthony Day and Winfred Arthur (June 2019). 'Exploration - Exploitation Tradeoffs and Information-Knowledge Gaps in Self-Regulated Learning: Implications for Learner-Controlled Training and Development'. In: *Human Resource Management Review* 29.2, pp. 196–217. ISSN: 10534822. DOI: 10.1016/j.hrmr.2018.07.004.

Hjalmarsson, Håkan (2002). 'Iterative feedback tuning—an overview'. In: *International journal of adaptive control and signal processing* 16.5, pp. 373–395.

Hjalmarsson, Håkan, Michel Gevers and Franky De Bruyne (Dec. 1996). 'For Model-Based Control Design, Closed-Loop Identification Gives Better Performance'. In: *Automatica* 32.12, pp. 1659–1673. ISSN: 00051098. DOI: 10.1016/S0005-1098(96)80003-3.

Hongxia Wu, Hui Ni and G.T. Heydt (2002). ‘The Impact of Time Delay on Robust Control Design in Power Systems’. In: *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.02CH37309)*. Vol. 2. New York, NY, USA: IEEE, pp. 1511–1516. ISBN: 978-0-7803-7322-8. DOI: `10.1109/PESW.2002.985276`.

Hou, Zhongsheng (2013). ‘From Model-Based Control to Data-Driven Control: Survey, Classification and Perspective’. In: *Information Sciences*.

Hou, Zhongsheng and Shangtai Jin (2013). *Model free adaptive control*. CRC press Boca Raton, FL, USA:

Huang, Wesley H and Matthew T Mason (2000). ‘Mechanics, planning, and control for tapping’. In: *The International Journal of Robotics Research* 19.10, pp. 883–894.

Jacobs, Ron (Aug. 1997). ‘Control Model of Human Stance Using Fuzzy Logic’. In: *Biological Cybernetics* 77.1, pp. 63–70. ISSN: 0340-1200, 1432-0770. DOI: `10.1007/s004220050367`.

Jang, Jyh-Shing R et al. (1992). ‘Self-learning fuzzy controllers based on temporal back-propagation’. In: *IEEE Transactions on neural networks* 3.5, pp. 714–723.

Jones, Kelvin E., Antonia F. De C. Hamilton and Daniel M. Wolpert (Sept. 2002). ‘Sources of Signal-Dependent Noise During Isometric Force Production’. In: *Journal of Neurophysiology* 88.3, pp. 1533–1544. ISSN: 0022-3077, 1522-1598. DOI: `10.1152/jn.2002.88.3.1533`.

Kalman, R. E. (Mar. 1960). ‘A New Approach to Linear Filtering and Prediction Problems’. In: *Journal of Basic Engineering* 82.1, pp. 35–45. ISSN: 0021-9223. DOI: `10.1115/1.3662552`.

Kleinman, D (1969). ‘Optimal control of linear systems with time-delay and observation noise’. In: *IEEE Transactions on Automatic Control* 14.5, pp. 524–527.

Kober, Jens, J. Andrew Bagnell and Jan Peters (Sept. 2013). ‘Reinforcement Learning in Robotics: A Survey’. In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274. ISSN: 0278-3649, 1741-3176. DOI: `10.1177/0278364913495721`.

Kocijan, J. et al. (2003). ‘Predictive Control with Gaussian Process Models’. In: *The IEEE Region 8 EUROCON 2003. Computer as a Tool.* Vol. 1. Ljubljana, Slovenia: IEEE, pp. 352–356. ISBN: 978-0-7803-7763-9. DOI: `10.1109/EURCON.2003.1248042`.

Kocijan, Juš (2008). 'Gaussian process models for systems identification'. In: *Proc. 9th Int. PhD Workshop on Sys. and Cont*, pp. 8–15.

Kocijan, Juš (2016). *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Advances in Industrial Control. Cham: Springer International Publishing. ISBN: 978-3-319-21020-9 978-3-319-21021-6. DOI: 10.1007/978-3-319-21021-6.

Kocijan, Juš et al. (Dec. 2005). 'Dynamic Systems Identification with Gaussian Processes'. In: *Mathematical and Computer Modelling of Dynamical Systems* 11.4, pp. 411–424. ISSN: 1387-3954, 1744-5051. DOI: 10.1080/13873950500068567.

Kojabadi, H Madadi (2005). 'Simulation and Experimental Studies of Model Reference Adaptive System for Sensorless Induction Motor Drive'. In: *Simulation Modelling Practice and Theory*.

Krivec, Tadej, Gregor Papa and Juš Kocijan (2021). 'Simulation of variational Gaussian process NARX models with GPGPU'. In: *ISA transactions* 109, pp. 141–151.

Kuss, Malte and Carl Rasmussen (2003). 'Gaussian processes in reinforcement learning'. In: *Advances in neural information processing systems* 16.

Kwakernaak, H. and R. Sivan (1974). *Linear Optimal Control Systems*. Vol. 19. USA: John Wiley & Sons, Inc., pp. 631–632. ISBN: 0471511102. DOI: 10.1109/TAC.1974.1100628.

Lakie, M, N Caplan and ID Loram (2003). 'Human balancing of an inverted pendulum with a compliant linkage: neural control by anticipatory intermittent bias'. In: *Journal of Physiology* 551, pp. 357–370.

Laub, AJ and M Wette (1984). *Algorithms and software for pole assignment and observers*. Tech. rep. California Univ., Santa Barbara (USA). Dept. of Electrical and Computer ...

Lawrence, Neil, Matthias Seeger and Ralf Herbrich (2002). 'Fast sparse Gaussian process methods: The informative vector machine'. In: *Advances in neural information processing systems* 15.

Leen, Gayle, Jaakko Peltonen and Samuel Kaski (Oct. 2012). 'Focused Multi-Task Learning in a Gaussian Process Framework'. In: *Machine Learning* 89.1-2, pp. 157–182. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-012-5302-y.

Leith, D. J. and W. E. Leithead (2000). 'Survey of gain-scheduling analysis and design'. In: *International Journal of Control* 73.11, pp. 1001–1025. DOI: 10.1080/002071700411304.

Lin, Long-Ji (1991). 'Self-Improvement Based On Reinforcement Learning, Planning and Teaching'. In: *Machine Learning Proceedings 1991*. Elsevier, pp. 323–327. ISBN: 978-1-55860-200-7. DOI: `10.1016/B978-1-55860-200-7.50067-2`.

Ljung, Lennart (1971). *Characterization of the Concept of 'Persistently Exciting' in the Frequency Domain*. English. Research Reports TFRT-3038. Department of Automatic Control, Lund Institute of Technology (LTH).

– (1999). *System Identification: Theory for the User*. 2nd ed. Prentice Hall Information and System Sciences Series. Upper Saddle River, NJ: Prentice Hall PTR. ISBN: 978-0-13-656695-3.

Loram, Ian D, Peter J Gawthrop and Martin Lakie (2006). 'The frequency of human, manual adjustments in balancing an inverted pendulum is constrained by intrinsic physiological factors'. In: *The Journal of physiology* 577.1, pp. 417–432.

Loram, Ian D. and Martin Lakie (May 2002). 'Human Balancing of an Inverted Pendulum: Position Control by Small, Ballistic-like, Throw and Catch Movements'. In: *The Journal of Physiology* 540.3, pp. 1111–1124. ISSN: 0022-3751, 1469-7793. DOI: `10.1113/jphysiol.2001.013077`.

Loram, Ian D., Constantinos N. Maganaris and Martin Lakie (2005). 'Human postural sway results from frequent, ballistic bias impulses by soleus and gastrocnemius'. In: *The Journal of Physiology* 564.1, pp. 295–311. ISSN: 1469-7793. DOI: `10.1113/jphysiol.2004.076307`. URL: `http://dx.doi.org/10.1113/jphysiol.2004.076307`.

Loram, Ian D. et al. (Jan. 2011). 'Human Control of an Inverted Pendulum: Is Continuous Control Necessary? Is Intermittent Control Effective? Is Intermittent Control Physiological?' In: *The Journal of Physiology* 589.2, pp. 307–324. ISSN: 0022-3751, 1469-7793. DOI: `10.1113/jphysiol.2010.194712`.

Loram, Ian David et al. (2014). 'Does the motor system need intermittent control?' In: *Exercise and Sport Sciences Reviews* 42.3, pp. 117–125.

Loram, ID, M Lakie and P Gawthrop (2009). 'Visual control of stable and unstable loads: what is the feedback delay and extent of linear time-invariant control?' In: *The Journal of Physiology* 587.Pt 6, pp. 1343–65.

Luppi, Alessandro, Claudio De Persis and Pietro Tesi (2022). 'On Data-Driven Stabilization of Systems with Nonlinearities Satisfying Quadratic Constraints'. In.

Mareels, I. M. Y. and M. Gevers (Oct. 1988). 'Persistency of Excitation Criteria for Linear, Multivariable, Time-Varying Systems'. In: *Mathematics of Control, Signals, and Systems* 1.3, pp. 203–226. ISSN: 0932-4194, 1435-568X. DOI: 10.1007/BF02551284.

Mareels, I.M.Y. et al. (Jan. 1987). 'How Exciting Can a Signal Really Be?' In: *Systems & Control Letters* 8.3, pp. 197–204. ISSN: 01676911. DOI: 10.1016/0167-6911(87)90027-2.

Martín, J. Alberto Álvarez (2018). 'Adaptive multivariable intermittent control: theory, development, and applications to real-time systems'. PhD thesis. University of Glasgow.

Martín, J. Alberto Álvarez et al. (Oct. 2021). 'Intermittent Control as a Model of Mouse Movements'. In: *ACM Transactions on Computer-Human Interaction* 28.5, pp. 1–46. ISSN: 1073-0516, 1557-7325. DOI: 10.1145/3461836.

Michimoto, Kenjiro et al. (Aug. 2016). 'Reinforcement Learning for Stabilizing an Inverted Pendulum Naturally Leads to Intermittent Feedback Control as in Human Quiet Standing'. In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Orlando, FL, USA: IEEE, pp. 37–40. ISBN: 978-1-4577-0220-4. DOI: 10.1109/EMBC.2016.7590634.

Miller, Caleb J et al. (2024). *Exploration with Scalable Gaussian Process Reinforcement Learning*. Tech. rep. Lawrence Livermore National Laboratory (LLNL), Livermore, CA (United States).

Moore, J. (Jan. 1983). 'Persistence of Excitation in Extended Least Squares'. In: *IEEE Transactions on Automatic Control* 28.1, pp. 60–68. ISSN: 0018-9286. DOI: 10.1109/TAC.1983.1103142.

Morasso, Pietro, Amel Cherif and Jacopo Zenzeri (May 2020). 'State-Space Intermittent Feedback Stabilization of a Dual Balancing Task'. In: *Scientific Reports* 10.1, p. 8470. ISSN: 2045-2322. DOI: 10.1038/s41598-020-64911-7.

Murray, J.J. et al. (May 2002). 'Adaptive Dynamic Programming'. In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 32.2, pp. 140–153. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2002.801727.

Murray-Smith, Roderick et al. (Sept. 2003). ‘Adaptive, Cautious, Predictive Control with Gaussian Process Priors’. In: *IFAC Proceedings Volumes* 36.16, pp. 1155–1160. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)34915-7.

Narendra, Kumpati S., Yu Wang and Snehasis Mukhopadhay (Dec. 2016). ‘Fast Reinforcement Learning Using Multiple Models’. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. Las Vegas, NV, USA: IEEE, pp. 7183–7188. ISBN: 978-1-5090-1837-6. DOI: 10.1109/CDC.2016.7799377.

Navas, Fernando and Lawrence Stark (Feb. 1968). ‘Sampling or intermittency in hand control system dynamics’. In: *Biophysical Journal* 8.2, pp. 252–302. ISSN: 00063495. DOI: 10.1016/S0006-3495(68)86488-4.

Neilson, P. D., M. D. Neilson and N. J. O’Dwyer (Jan. 1988). ‘Internal Models and Intermittency: A Theoretical Account of Human Tracking Behavior’. In: *Biological Cybernetics* 58.2, pp. 101–112. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00364156.

Nguyen, Derrick H and Bernard Widrow (1990). ‘Neural networks for self-learning control systems’. In: *IEEE Control systems magazine* 10.3, pp. 18–23.

Nichols, R.A., R.T. Reichert and W.J. Rugh (June 1993). ‘Gain Scheduling for H-infinity Controllers: A Flight Control Example’. In: *IEEE Transactions on Control Systems Technology* 1.2, pp. 69–79. ISSN: 10636536. DOI: 10.1109/87.238400.

Nomura, Taishin et al. (2013). ‘Modeling human postural sway using an intermittent control and hemodynamic perturbations’. In: *Mathematical Biosciences* 245.1. SI : BIOCOMP 2012, pp. 86 –95. ISSN: 0025-5564. DOI: http://dx.doi.org/10.1016/j.mbs.2013.02.002. URL: http://www.sciencedirect.com/science/article/pii/S0025556413000485.

Osband, Ian, John Aslanides and Albin Cassirer (2018). ‘Randomized prior functions for deep reinforcement learning’. In: *Advances in Neural Information Processing Systems* 31.

P.E. Wellstead D. Prager, P. Zanker (Aug. 1979). ‘Pole assignment self-tuning regulator’. English. In: *Proceedings of the Institution of Electrical Engineers* 126 (8), 781–787(6). ISSN: 0020-3270. URL: https://digital-library.theiet.org/content/journals/10.1049/piee.1979.0171.

Peng, Shige (Feb. 1992). ‘A Generalized Dynamic Programming Principle and Hamilton-Jacobi-Bellman Equation’. In: *Stochastics and Stochastic Reports* 38.2, pp. 119–134. ISSN: 1045-1129. DOI: 10.1080/17442509208833749.

Petelin, Dejan and Jus Kocijan (Apr. 2011). ‘Control System with Evolving Gaussian Process Models’. In: *2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*. Paris, France: IEEE, pp. 178–184. ISBN: 978-1-4244-9978-6. DOI: 10.1109/EAIS.2011.5945910.

Peterka, Robert J et al. (2000). ‘Postural Control Model Interpretation of Stabilogram Diffusion Analysis’. In: *Biological cybernetics* 82.4, pp. 335–343.

Quiñonero-Candela, Joaquin, Carl Edward Rasmussen and Christopher K. I. Williams (Aug. 2007). ‘Approximation Methods for Gaussian Process Regression’. In: *Large-Scale Kernel Machines*. Ed. by Léon Bottou et al. The MIT Press, pp. 203–224. ISBN: 978-0-262-25579-0. DOI: 10.7551/mitpress/7496.003.0011.

Rasmussen, Carl Edward (1997). ‘Evaluation of Gaussian processes and other methods for non-linear regression’. PhD thesis. University of Toronto Toronto, Canada.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press. ISBN: 978-0-262-18253-9.

Renaudo, Erwan et al. (2015). ‘Respective Advantages and Disadvantages of Model-based and Model-free Reinforcement Learning in a Robotics Neuro-inspired Cognitive Architecture’. In: *Procedia Computer Science* 71, pp. 178–184. ISSN: 18770509. DOI: 10.1016/j.procs.2015.12.194.

Ronco, Eric, Peter J Gawthrop and David J Hill (1999). *Open-Loop Intermittent Feedback Optimal Control: reference manual to an on-line simulation package*. Tech. rep. Citeseer.

Schochetman, Irwin E and Robert L Smith (1989). ‘Infinite horizon optimization’. In: *Mathematics of Operations Research* 14.3, pp. 559–574.

Seeger, Matthias (2004). ‘Gaussian processes for machine learning’. In: *International journal of neural systems* 14.02, pp. 69–106.

Smart, William D and Leslie Pack Kaelbling (2000). ‘Practical reinforcement learning in continuous spaces’. In: *ICML*, pp. 903–910.

Smith, O. J. (1959). 'A controller to overcome dead time'. In: *ISA Journal* 6, pp. 28–33.

Snelson, Edward and Zoubin Ghahramani (2005). 'Sparse Gaussian processes using pseudo-inputs'. In: *Advances in neural information processing systems* 18.

Stenman, Anders (1999). *Model on Demand: Algorithms, Analysis and Applications.* Linköping Studies in Science and Technology Dissertation 571. Linköping: Univ. ISBN: 978-91-7219-450-2.

Stépán, Gábor and Tamás Insperger (Jan. 2006). 'Stability of Time-Periodic and Delayed Systems — a Route to Act-and-Wait Control'. In: *Annual Reviews in Control* 30.2, pp. 159–168. ISSN: 13675788. DOI: `10.1016/j.arcontrol.2006.08.002`.

Storn, Rainer and Kenneth Price (1997). 'Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces'. In: *Journal of global optimization* 11, pp. 341–359.

Strehl, Alexander L. and Michael L. Littman (Dec. 2008). 'An Analysis of Model-Based Interval Estimation for Markov Decision Processes'. In: *Journal of Computer and System Sciences* 74.8, pp. 1309–1331. ISSN: 00220000. DOI: `10.1016/j.jcss.2007.08.009`.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction.* Second edition. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-03924-6.

Tan, Ming (1991). 'Case-Sensitve Reinforcement Learning for Adaptive Classification and Control'. In: *Proceedings of the Eighth International Conference, Evanston, Illinois*, pp. 358–362.

Telford, CW (1931). 'The refractory phase of voluntary and associative responses'. In: *J. of Exp. Psychol.* 14, pp. 1–36.

Todorov, Emanuel and Michael I. Jordan (Nov. 2002). 'Optimal Feedback Control as a Theory of Motor Coordination'. In: *Nature Neuroscience* 5.11, pp. 1226–1235. ISSN: 1097-6256, 1546-1726. DOI: `10.1038/nn963`.

Tsypkin, Y.Z. (1971). '3. Adaptation and Learning'. In: *Adaptation and Learning in Automatic Systems.* Ed. by Ya.Z. Tsypkin. Vol. 73. Mathematics in Science and Engineering. Elsevier, pp. 44 –75.

Van Otterlo, Martijn and Marco Wiering (2012). ‘Reinforcement Learning and Markov Decision Processes’. In: *Reinforcement Learning.* Ed. by Marco Wiering and Martijn Van Otterlo. Vol. 12. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 3–42. ISBN: 978-3-642-27644-6 978-3-642-27645-3. DOI: `10.1007/978-3-642-27645-3_1`.

van Waarde, Henk J. et al. (Jan. 2020). *Data Informativity: A New Perspective on Data-Driven Analysis and Control.* arXiv: `1908.00468 [math]`.

Van Waarde, Henk J et al. (2020). ‘Willems’ fundamental lemma for state-space systems and its extension to multiple datasets’. In: *IEEE Control Systems Letters* 4.3, pp. 602–607.

Van Waarde, Henk J. et al. (Dec. 2023). ‘The Informativity Approach: To Data-Driven Analysis and Control’. In: *IEEE Control Systems* 43.6, pp. 32–66. ISSN: 1066-033X, 1941-000X. DOI: `10.1109/MCS.2023.3310305`.

Vince, Margaret A (Mar. 1948). ‘The intermittency of control movements and the psychological refractory period’. In: *British Journal of Psychology.* 38.3, pp. 149–157. ISSN: 0373-2460. DOI: `10.1111/j.2044-8295.1948.tb01150.x`.

Vrabie, D., O. Pastravanu and F. L. Lewis (2007). ‘Policy iteration for continuous-time systems with unknown internal dynamics’. In: *2007 Mediterranean Conference on Control Automation*, pp. 1–6.

Whitehead, Steven D. and Dana H. Ballard (July 1991). ‘Learning to Perceive and Act by Trial and Error’. In: *Machine Learning* 7.1, pp. 45–83. ISSN: 0885-6125, 1573-0565. DOI: `10.1007/BF00058926`.

Willems, Jan C et al. (2005). ‘A note on persistency of excitation’. In: *Systems & Control Letters* 54.4, pp. 325–329.

Williams, Christopher et al. (2008). ‘Multi-task gaussian process learning of robot inverse dynamics’. In: *Advances in neural information processing systems* 21.

Williams, Christopher K and Carl Edward Rasmussen (2006). *Gaussian processes for machine learning.* Vol. 2. MIT press Cambridge, MA.

Winter, David A. et al. (Sept. 1998). ‘Stiffness Control of Balance in Quiet Standing’. In: *Journal of Neurophysiology* 80.3, pp. 1211–1221. ISSN: 0022-3077, 1522-1598. DOI: `10.1152/jn.1998.80.3.1211`.

Wolpert, Daniel M., Zoubin Ghahramani and Michael I. Jordan (Sept. 1995). 'An Internal Model for Sensorimotor Integration'. In: *Science* 269.5232, pp. 1880–1882. ISSN: 0036-8075, 1095-9203. DOI: `10.1126/science.7569931`.

Yoshikawa, Naoya et al. (Apr. 2016). 'Intermittent Feedback-Control Strategy for Stabilizing Inverted Pendulum on Manually Controlled Cart as Analogy to Human Stick Balancing'. In: *Frontiers in Computational Neuroscience* 10. ISSN: 1662-5188. DOI: `10.3389/fncom.2016.00034`.

Yu, Yue et al. (Apr. 2021). *On Controllability and Persistency of Excitation in Data-Driven Control: Extensions of Willems' Fundamental Lemma.* arXiv: `2102.02953 [cs, eess]`.

Zgonnikov, Arkady and Ihor Lubashevsky (Nov. 2015). 'Double-Well Dynamics of Noise-Driven Control Activation in Human Intermittent Control: The Case of Stick Balancing'. In: *Cognitive Processing* 16.4, pp. 351–358. ISSN: 1612-4782, 1612-4790. DOI: `10.1007/s10339-015-0653-5`.