

**REASONING ABOUT PERCEPTION UNCERTAINTY  
IN NONLINEAR MODEL PREDICTIVE CONTROL**

**DISSERTATION**

Submitted in Partial Fulfillment of  
the Requirements for  
the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY  
TANDON SCHOOL OF ENGINEERING

by

Armand Jordana

January 2025

# REASONING ABOUT PERCEPTION UNCERTAINTY IN NONLINEAR MODEL PREDICTIVE CONTROL

DISSERTATION

Submitted in Partial Fulfillment of  
the Requirements for  
the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY  
TANDON SCHOOL OF ENGINEERING

by

Armand Jordana

January 2025

Approved:



---

Department Chair Signature

December 16, 2024

---

Date

Approved by the Guidance Committee:

Major: Electrical Engineering



---

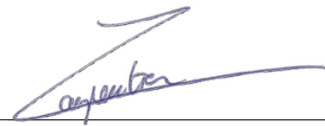
**Prof. Ludovic Righetti**

Associate Professor  
NYU Tandon School of Engineering

December 16th 2024

---

Date



---


**Dr. Justin Carpentier**

INRIA Researcher

December 16th 2024

---

Date



---

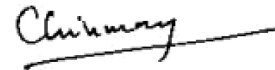
**Prof. Giuseppe Loianno**

Associate Professor  
NYU Tandon School of Engineering

12/16/2024

---

Date



---

**Prof. Chinmay Hegde**

Associate Professor  
NYU Tandon School of Engineering

12/16/2024

---

Date

*S Rangan*

iii

---

**Prof. Sundeep Rangan**

Professor

NYU Tandon School of Engineering

**16 Dec 2024**

---

Date



Microfilm or other copies of this dissertation are obtainable from

UMI Dissertation Publishing

ProQuest CSA

789 E. Eisenhower Parkway

P.O. Box 1346

Ann Arbor, MI 48106-1346

## Vita

Armand Jordana received his M.Sc. degree in Mathematics, Computer Vision, and Machine Learning from École Normale Supérieure Paris-Saclay in 2019. During his master's studies, he completed a research internship at INRIA Paris in the WILLOW team under the supervision of Justin Carpentier. From 2019 to 2020, he was a research intern in the Machines in Motion Laboratory at New York University (NYU), supervised by Ludovic Righetti.

Since Fall 2020, he has been pursuing the Doctor of Philosophy degree in Electrical Engineering at NYU, Tandon School of Engineering, under the supervision of Ludovic Righetti and Justin Carpentier. His research interests include model predictive control, risk-sensitive control, estimation, and robotics. During his doctoral studies, he received the Best Poster Award at the 2024 Workshop on Advancements in Trajectory Optimization and Model Predictive Control for Legged Systems.

## Acknowledgements

I would like to express my deepest gratitude to my advisors Justin and Ludovic for their scientific guidance and support. They both taught me a lot.

Justin introduced me to robotics in 2019 at INRIA Paris. This was a wonderful opportunity for which I am most grateful. Since then, Justin has continued to guide me throughout my PhD. In spite of the physical distance, he has closely supervised my work and encouraged me. Justin's commitment to accommodate me in the best conditions in Paris was essential to my research. Thank you, Justin!

Ludovic offered me the unique opportunity to perform a PhD in his laboratory, and I am extremely thankful for this. He has been a great source of inspiration; his kindness, integrity, and expertise fostered my academic and personal growth. Throughout my PhD journey, he consistently cultivated a space where I could be myself and freely conduct research under his guidance. Thank you, Ludo!

Moreover, I would like to thank all current and past members of the Machines in Motion laboratory. In this regard, I want to pay a special recognition of appreciation to Sébastien and Avadesh, with whom I had the chance to collaborate and become friends. Ahmad, it was wonderful to have you in New York City for a few months. Majid, thank you for all the insightful discussions and for hosting me in Munich.

I would like to thank Nicolas and all the Gepetto team for their warm welcome in Toulouse. During both of my visits, they provided me with optimal working conditions and made me feel at home.

Finally, I would like to thank my friends and family who have continuously supported me throughout this journey. This would not have been possible without you.

Armand Jordana, Brooklyn, January 2025

To my family

**ABSTRACT****REASONING ABOUT PERCEPTION UNCERTAINTY IN  
NONLINEAR MODEL PREDICTIVE CONTROL**

by

**Armand Jordana****Advisors: Prof. Ludovic Righetti, Prof. Justin Carpentier****Submitted in Partial Fulfillment of the Requirements for  
the Degree of Doctor of Philosophy (Electrical Engineering)****January 2025**

In robotics, nonlinear Model Predictive Control (MPC) has emerged as a promising tool to generate complex motions while enabling online adaptation of the robot behavior as the environment changes. However, the lack of efficient computational methods hindered its widespread deployment on real hardware. In practice, MPC formulations and computational methods are often simplified to obtain real-time capable controllers. In this thesis, we develop efficient numerical methods to fully leverage the promises of MPC on robots by ensuring safe and

globally optimal plans while being aware of the uncertainty resulting from the partial sensing of the environment.

In the first part of this thesis, we present several contributions to state-feedback MPC. Specifically, we introduce a state-of-the-art solver for constrained nonlinear MPC. We propose the first demonstration of closed-loop nonlinear MPC with hard constraints on an industrial manipulator. Then, we present a method to obtain globally optimal plans using function approximation. Next, we investigate how online estimation enables us to leverage force sensor information in state-feedback MPC. This results in state-of-the-art performance on challenging tasks involving contact interaction.

While online estimation can enable complex sensor feedback, this is insufficient to reason about the uncertainty resulting from the partial sensing of the world. In the second part of this thesis, we show how solving the estimation and control jointly enables optimality under uncertainty. We propose an efficient numerical method to solve dynamic game control with imperfect information and demonstrate the ability of the approach to significantly improve performance on hardware in the face of uncertainties.

# Table of Contents

|   |           |
|---|-----------|
| Vita . . . . .  | v         |
| Acknowledgements . . . . .  | vi        |
| Abstract . . . . .  | viii      |
| List of Publications . . . . .                                      | xiii      |
| List of Figures . . . . .   | xviii     |
| List of Tables . . . . .  | xx        |
| <b>1 Introduction</b>   | <b>1</b>  |
| 1.1 Model Predictive Control . . . . .                              | 3         |
| 1.2 Reasoning about uncertainty . . . . .                           | 8         |
| 1.3 Statement of purpose . . . . .                                  | 10        |
| 1.4 Contributions . . . . .   | 11        |
| 1.5 Outline . . . . .   | 12        |
| 1.6 Notations . . . . .   | 13        |
| <b>I Model Predictive Control</b>                                   | <b>14</b> |
| <b>2 Stagewise Resolution of Optimal Control Problems</b>           | <b>16</b> |
| 2.1 Sequential Quadratic Programming for Optimal Control Problems . | 21        |

|  |            |
|--|------------|
|  | xi         |
| 2.2 Solving the unconstrained QP . . . . .   | 25         |
| 2.3 Solving the constrained QP . . . . .   | 28         |
| 2.4 Practical implementation of the SQP . . . . .  | 33         |
| 2.5 Experiments . . . . .  | 37         |
| 2.6 Discussion . . . . .   | 52         |
| 2.7 Conclusion . . . . .   | 53         |
| <b>3 Infinite-Horizon Value Function Approximation for Model Predictive Control</b>        | <b>55</b>  |
| 3.1 Infinite horizon MPC . . . . .   | 58         |
| 3.2 Approximate infinite horizon MPC via function approximation . . .                      | 60         |
| 3.3 Experiments . . . . .  | 64         |
| 3.4 Discussion . . . . .   | 76         |
| 3.5 Conclusion . . . . .   | 78         |
| <b>4 Force Feedback Model-Predictive Control via Online Estimation</b>                     | <b>79</b>  |
| 4.1 Background: MPC with rigid contact . . . . .   | 82         |
| 4.2 Force-feedback MPC via online estimation . . . . .                                     | 84         |
| 4.3 Experimental study . . . . .   | 90         |
| 4.4 Conclusion . . . . .   | 98         |
| <b>II Reasoning about the Perception Uncertainty</b>                                       | <b>100</b> |
| <b>5 Stagewise Newton Method for Dynamic Game Control With Imperfect State Observation</b> | <b>102</b> |
| 5.1 Dynamic game control with imperfect state observation . . . . .                        | 103        |



|   |            |
|---|------------|
|   | xii        |
| 5.2 Stagewise Newton method . . . . .   | 106        |
| 5.3 Experiments . . . . .   | 115        |
| 5.4 Conclusion . . . . .  | 120        |
| <b>6 Risk-Sensitive Extended Kalman Filter</b>  | <b>121</b> |
| 6.1 Background: Extended Kalman Filter . . . . .  | 122        |
| 6.2 Risk-sensitive filter . . . . .   | 123        |
| 6.3 Experiments . . . . .   | 128        |
| 6.4 Conclusion . . . . .  | 141        |
| <b>7 Conclusion</b>   | <b>142</b> |
| <b>A Stagewise Implementations of Sequential Quadratic Programming<br/>for Model-Predictive Control</b> | <b>144</b> |
| <b>B Stagewise Newton Method for Dynamic Game Control with Im-<br/>perfect State Observation</b>        | <b>156</b> |
| B.1 Problem statement . . . . .   | 156        |
| B.2 Characterization of the Newton step . . . . .   | 158        |
| B.3 Future stress . . . . .   | 164        |
| B.4 Past stress . . . . .   | 169        |
| B.5 Coupling . . . . .  | 174        |

# List of Publications

This dissertation is based on the following publications:

**A. Jordana\***, S. Kleff\*, A. Meduri\* et al. "Stagewise implementations of Sequential Quadratic Programming for Model Predictive Control", Submitted to T-RO, 2024

**A. Jordana** et al. "Infinite-Horizon Value Function Approximation for Model Predictive Control" Submitted to RAL, 2024

**A. Jordana\***, S. Kleff\*, et al. Force feedback Model-Predictive Control via Online Estimation. IEEE International Conference on Robotics and Automation (ICRA), 2024

**A. Jordana** et al. "Stagewise newton method for dynamic game control with imperfect state observation." IEEE Control Systems Letters, 2022.

**A. Jordana** et al. "Risk-Sensitive Extended Kalman Filter." IEEE International Conference on Robotics and Automation (ICRA). 2024.

# List of Figures

|     |   |    |
|-----|---|----|
| 1.1 | Construction workers on high metallic beams . . . . .   | 2  |
| 1.2 | MPC illustration . . . . .  | 4  |
| 2.1 | Percentage of problem solved as a function of the maximum number<br>of iterations allowed $n_{\text{iter}}$ on 4 randomized unconstrained OCPs for<br>the 4 solvers: DDP, FDDP with default line-search, FDDP with filter<br>line-search and our SQP. Our SQP exhibits a faster and more robust<br>convergence on difficult problems, such as the humanoid taichi task. | 38 |
| 2.2 | Average time per iteration for each solver on the 4 benchmark<br>problems (Kuka, Quadrotor, Pendulum, Taichi) . . . . .   | 42 |
| 2.3 | KKT residual norm and number of iterations for the circle tracking<br>task. Our SQP solver converges within 3 iterations while FDDP hits<br>the maximum number of iterations ( $n_{\text{iter}} = 5$ ) without reaching the<br>desired tolerance. . . . .   | 43 |

|     |  |    |
|-----|--|----|
| 2.4 | Solo center-of-mass tracking task with friction cone constraints. The continuous lines represent the ratio $\frac{F_T}{F_N}$ at the front left foot, and the gray dashed line represent the friction cone constraint. Observe in the $F_z$ plots the unconstrained OCP solution (green) crossing the friction cone constraint while the constrained OCP solution (blue) remains within the constraint. . . . . | 44 |
| 2.5 | Snapshots of standing motion. The red arrows represent the contact forces and the white cones are the friction constraint ( $\mu = 0.8$ ). In the 3 <sup>rd</sup> and 4 <sup>th</sup> frames, one can see the tangential forces lying on the boundary on the friction cone. . . . .  | 45 |
| 2.6 | Joint position $q_1$ for the circle task in the constrained (blue) and unconstrained case (green). The gray-shaded area represents the infeasible region. . . . .  | 45 |
| 2.7 | End-effector position trajectories in the $(y, z)$ -plane during the circle tracking task, subject to nonlinear constraints in Cartesian space. The gray-shaded areas represent the infeasible regions. . . . .  | 46 |
| 2.8 | The SQP solver keeps the end-effector on a straight line constraint (black dotted line) even during unexpected perturbations. The SQP solver is able to rapidly find robot configurations needed to satisfy the constraints while tracking the desired goal. . . . .   | 46 |
| 2.9 | Comparison between the compute time of 25 iterations OSQP_OCP and OSQP. . . . .  | 49 |
| 3.1 | Approximated infinite horizon value function . . . . .   | 66 |
| 3.2 | Rollout of MPC controllers with different horizon lengths using the learned value function as a terminal cost. . . . .   | 67 |

|     |   |    |
|-----|---|----|
| 3.3 | Trajectories avoiding the local optima for different initial conditions.<br>The red dot represents the target $x^*$ .   | 68 |
| 3.4 | Error between the ground truth and the learned value function during training for various horizon length. The larger the horizon in Eq (3.7) is, the faster the algorithm converges to the ground truth.  | 70 |
| 3.5 | We run 1000 MPC simulations starting from random initial states with increasing horizon for each controller. Horizon 0 corresponds to the policy.   | 71 |
| 3.6 | Snapshots of pick-and-place task with static obstacle avoidance for the default MPC without value function (bottom) and the proposed MPC with value function (top). The green dots represent the end-effector targets that must be reached alternatively while avoiding collision with the black rod placed in the center.                | 74 |
| 3.7 | End effector trajectory. The robot must reach a target on the over side of the rod marked by the green dots, while avoiding the rod (black dot). The proposed approach (blue) can achieve the task by choosing a path going above the rod, while the default MPC (red) remains stuck in a local minima (trying to go underneath the rod). | 75 |
| 3.8 | Cumulative cost. The proposed approach (blue) achieves a lower cost by avoiding the rod and reaching the target.  | 76 |
| 3.9 | Target tracking with static obstacle. Collision distance between the robot's links (approximated as capsules) and the obstacle (moving rod).  | 76 |

|     |  |     |
|-----|--|-----|
| 4.1 | Normal force trajectories of the medium-velocity polishing task. The blue curve is the classical MPC without friction compensation, the green curve is the classical MPC with the Coulomb model compensation, and the red curve is the classical MPC with FL compensation.   | 93  |
| 4.2 | Normal force (top) and end-effector position error (bottom) for the polishing task: in blue the classical MPC (4.1), in green the classical MPC with the FL compensation term (4.2.3), in red the proposed approach with FL compensation and the force offset in the predictive model (4.2.2.2).                               | 95  |
| 4.3 | Lateral force trajectories in the $e_x$ direction for the force step tracking task: the blue curve is the classical MPC (Default), the green curve is the classical MPC with with integral control (Integral) and the red curve is the force offset estimation $\Delta F$ included in the predictive model ( $\Delta F$ (PM)). | 97  |
| 5.1 | Initial plan for different values of $\mu$ . The larger $\mu$ the more the controller plans to be pushed against the obstacle.   | 117 |
| 5.2 | Average trajectory. Compared the neutral controller, the dynamic game controller ( $\mu = 6$ ) exhibits a risk sensitive behavior as it remains further from the high cost area representing the obstacle.   | 117 |
| 5.3 | Average control trajectories. Compared the neutral controller, the dynamic game controller ( $\mu = 6$ ) has a larger standard deviation.  | 118 |
| 5.4 | End-effector position vs time. The dashed lines represent the target.  | 120 |
| 6.1 | Mass estimation for both EKF and RS-EKF.   | 131 |
| 6.2 | Quadrotor trajectory for both the EKF-MPC and the RS-EKF-MPC.  | 132 |

|     |  |     |
|-----|--|-----|
| 6.3 | End effector trajectory on a tracking task for both the EKF-MPC and the RS-EKF-MPC. An unexpected force is applied between 1s and 2s. . . . .  | 134 |
| 6.4 | Median MSE on the tracking over 10,000 experiments with random external disturbances. The envelope represents the 25th and 75th percentiles. . . . .   | 135 |
| 6.5 | Comparison of both methods after an external force of 20 N is applied by pulling the robot vertically. The vertical line indicates the moment when the robot is dropped. The top sub-figure overlays the trajectory of both the EKF in blue and RS-EKF in solid. The RS-EKF-based controller is more reactive to the perturbation and returns to the reference sooner. . . . . | 136 |
| 6.6 | Comparison of the RS-EKF and EKF when initialized with a wrong prior of 20 N on the estimated vertical external force. . . . .   | 140 |
| 6.7 | Comparison of the RS-EKF and EKF when initialized with a wrong prior of $-10$ N on the estimated vertical external force. . . . .  | 141 |

# List of Tables

|     |  |    |
|-----|--|----|
| 2.1 | Solvers characteristics. . . . .   | 39 |
| 2.2 | QP solvers characteristics. . . . .  | 50 |
| 2.3 | Mean solving time in milliseconds and number of QP iterations for<br>different problems . . . . .  | 50 |
| 4.1 | Mean-absolute error (MAE) of the normal force (in N) for the polish-<br>ing task over 10 circles: classical MPC (Default), FL compensation<br>(4.10) and Coulomb model (4.12). . . . .   | 94 |
| 4.2 | MAE of the normal force (in N) for the polishing task: force offset<br>$\Delta F$ vs. torque offset $\Delta\tau$ , used in the control loop either in the<br>”predictive model” way of 4.2.2.2 or in the ”corrective control” way<br>of 4.2.2.1. . . . .   | 94 |
| 4.3 | MAE of the normal force error for a step tracking task for different<br>controllers: classical MPC (Default), force offset estimation ( $\Delta F$ ),<br>torque offset estimation ( $\Delta\tau$ ) and integral control. $\Delta F$ and $\Delta\tau$ are<br>used as corrective control (4.2.2.1) or in the predictive model (4.2.2.2). . . . . | 97 |



4.4 Average squared torque and total cost for each controller for the energy task: classical MPC (Default), force offset estimation ( $\Delta F$ ), torque offset estimation ( $\Delta\tau$ ) and integral control.  $\Delta F$  and  $\Delta\tau$  are used as corrective control (4.2.2.1) or in the predictive model (4.2.2.2). 98

# Chapter 1

## Introduction

In recent years, we have seen a tremendous amount of progress in Robotics. Robots are starting to be able to manipulate complex objects [8, 37, 79], and legged robots can now traverse challenging terrains [1, 73, 84]. Nevertheless, robots are mostly confined to factories or research laboratories. While in the past decades, a large number of repetitive and difficult tasks were automated, there still remain a wide variety of physically challenging and dangerous jobs. Today, sectors such as construction and agriculture still suffer high rates of injuries or fatal accidents <sup>1</sup>. Enabling the deployment of robots outside factories could yield major positive impacts in those fields. From a technical perspective, one of the key challenges is the necessity of deriving controllers that can consistently adapt to novel situations. Factories can be designed to facilitate the deployment of robots and in many cases, robot motions can be preplanned once and for all. In contrast, environments such as field crops or construction sites are never the same, and one can hardly plan coherent robot motions ahead of time.

---

<sup>1</sup><https://www.bls.gov/charts/census-of-fatal-occupational-injuries/civilian-occupations-with-high-fatal-work-injury-rates.htm>



Figure 1.1: Construction workers on high metallic beams

Figure 1.1 shows construction workers on high metallic beams executing complex tasks. Most likely, those workers have never been to those places. Today, deploying robots in such settings remains extremely challenging for several reasons. First, our modeling of the world does not encompass all physical phenomena. For instance, in robotics, it is extremely challenging to model actuator dynamics, friction, joint elasticity, or communication delays accurately. Second, our perception of the state of novel environments is limited. For instance, one can hardly know beforehand how slippery the metallic beams are. Lastly, deploying robots outside factories always comes with some danger. In our example, there is a high *risk* of falling from the metallic beams.

While the concept of *risk* can be defined in many ways, we will consider *risk* to be the effect of uncertainty on objectives. In settings such as those depicted in Figure 1.1, a robot would have to carefully reason about uncertainties to account for the risk. In these scenarios, the objective could be defined as completing a manipulation task without falling while one of the sources of uncertainty could be the slipperiness of the beam. This unknown slipperiness could affect the objective by making the robot fall. Without any uncertainty, everything could be perfectly planned ahead of time, and the robot could technically move quickly while stepping

next to the edges of the beams. However, in the presence of uncertainty, we expect a controller to reason about the potential inaccuracies of its plan. In practice, this could mean stepping only at the center of the beams and avoiding dynamic motions in order to always get the time to stabilize the robot when something does not go as planned.

Humans or animals can naturally adapt in the face of risk and behave cautiously. However, automatically adapting the degree of cautiousness according to the level of danger and uncertainty is an open problem in Robotics. In this thesis, we aim to make progress in that direction. More precisely, we aim to address the following question: **How can we design controllers that reason online about novel situations despite perception uncertainty?**

## 1.1 Model Predictive Control

Model Predictive Control (MPC) has become popular for online robot decision-making. It has shown compelling results with all kinds of robots ranging from industrial manipulators [102], quadrupeds [129, 139, 164] to humanoids [44, 104]. The general idea of MPC is to formulate the robot motion generation problem as a numerical optimization problem, i.e., a finite horizon Optimal Control Problem (OCP), and solve it online at every control cycle using the current state estimate as the initial state. By re-planning at each control cycle, the controller can adapt the robot behavior as the state of the system and environment change. Figure 1.2 provides an illustration. Unlike standard robotics controllers such as PID or Inverse Dynamics (ID), MPC does not track a pre-planned trajectory. In case a disturbance is encountered, the controller searches for a new optimal path and does

not necessarily aim to reject the perturbation. Intuitively, it will either ignore or reject the disturbance depending on whether it aligns with the high-level objective.

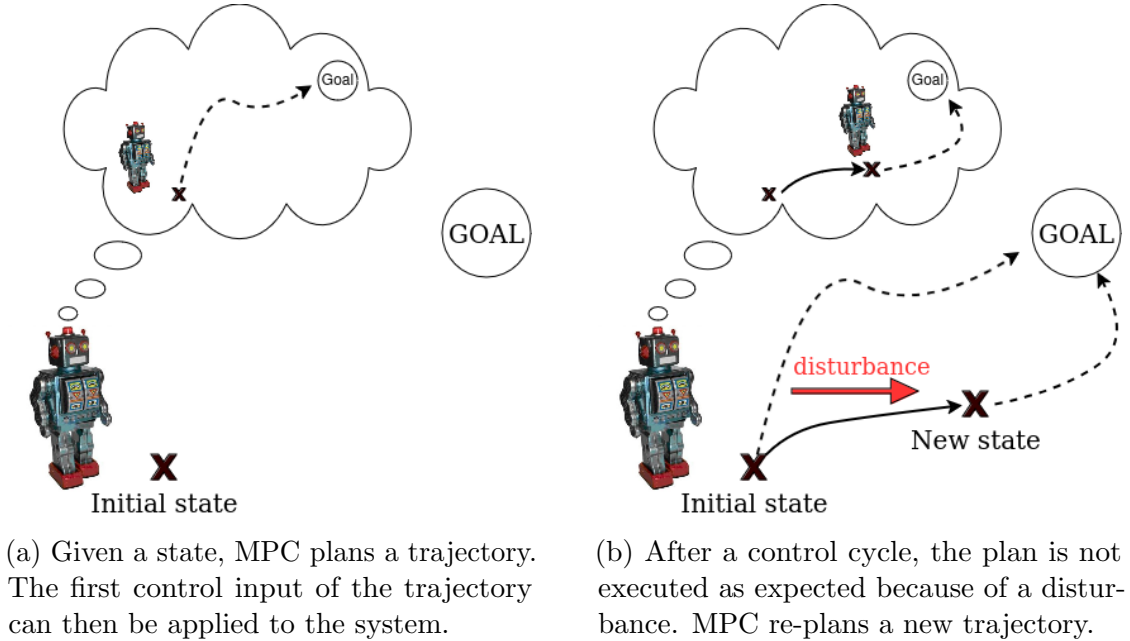


Figure 1.2: MPC illustration

### 1.1.1 The real-time constraint

MPC was originally developed to control chemical plants which have very slow dynamics [128]. In contrast, robots require extremely high replanning frequencies. For instance, when the leg of a quadruped enters into contact with the ground at an unexpected time, it is crucial to re-plan and adapt within milliseconds [192]. This raises many technical challenges as solving an OCP can be very computationally demanding. Furthermore, in order to execute complex motions and interact with the environment, it is crucial to directly control the torque of the robot [89, 102]. This allows the MPC to reason simultaneously about the joint space and the forces applied to the environment, a crucial requirement to perform locomotion and

manipulation. However, this implies optimizing over the full nonlinear dynamics of the robot, which can make the optimization problem nonconvex.

Furthermore, ensuring safety with MPC is challenging. One way to guarantee safety is to incorporate hard constraints in the OCP. Unfortunately, this can greatly slow down the solving time. For this reason, to date, in robotics, MPC has mostly been used as a planner by combining it with simpler controllers. Consequently, the final behavior obtained by the robot can differ from the optimal whole-body trajectory since the lower controller is free to act as a filter and modify the trajectory [71, 164]. Ideally, a robot should adapt with versatility to perturbations during a dynamic task and not reject perturbations naively. For instance, if a human operator helps a robotic arm by pushing it towards its goal, the robot should be compliant and not reject the perturbation. For these reasons, closing the loop with constrained MPC on torque-controlled robots holds the promise of obtaining safe, agile, and compliant robots.

In robotics, Differential Dynamic Programming (DDP) [125] is a popular choice to solve OCPs because it exploits the problem’s structure well. This advantage has led to a bustling algorithmic development over the past two decades [5, 68, 87, 92, 93, 114, 119, 120, 122, 141, 148, 161, 170]. In light of the increasing number of variations of DDP, one might naively ask: why not use well-established optimization algorithms [143]? Is there anything special about MPC that cannot be tackled by, for example, an efficient implementation of Sequential Quadratic Programming (SQP) [187]? In this thesis, we show that special implementations of numerical methods developed by the optimization-based control community [53, 65, 106, 144, 174, 183] are, in fact, sufficient to perform closed-loop constrained MPC on a torque-controlled robot.

### 1.1.2 Overcoming local behavior

One limitation of existing MPC formulations in robotics is that without an appropriate design of the terminal cost and constraint set, MPC with a finite horizon is not guaranteed to be globally stable, optimal, or recursively feasible [126]. In robotics applications, MPC exhibits local behaviors for two main reasons. The first is the use of a finite horizon, which creates local minima. The second is that practitioners often rely on local Trajectory Optimization (TO) techniques. In practice, this has led roboticists to spend a lot of time designing cost functions to avoid local minima. Infinite-horizon constrained MPC is a compelling framework as it ensures global stability [76]. In other words, it can guarantee that any initial state converges to a zero-cost stable state while satisfying hard constraints. Unfortunately, in the general case, the problem is intractable and has to be approximated [22].

Reinforcement Learning (RL) [22, 168] appears as a good candidate to move the compute time offline by approximating policies or value functions. It has recently shown impressive results in locomotion [1, 84] and manipulation [37, 79]. However, these techniques are subject to the curse of dimensionality and can be unsafe outside of their training distribution. Also, incorporating safety by imposing hard constraints is challenging with current RL tools. In contrast, by re-planning online, constrained MPC can adapt to novel situations and ensure safety.

Combining the advantages of online and offline decision making is appealing, and it has proven to be highly effective in the context of games such as Go or Chess [162]. In the context of robotics, it could allow maintaining safety and ensuring hard constraints while leveraging the full potential of function approximation. In this thesis, we propose to combine MPC and value function approximation. The infinite horizon value function is approximated using neural networks and trained using

value iteration and local gradient-based optimization. Then, during deployment, the approximated value is used as a terminal cost for the OCP. We will show that the approximated value function provides a global behavior while the online optimization compensates for approximation errors and ensures hard constraint satisfaction even outside of the training distribution.

### 1.1.3 The challenges of force control

Another limitation of the current MPC formulations in robotics is the difficulty to reason about output signals such as force. Many tasks require accurate control of contact forces exerted on the environment: polishing, grinding, grasping, etc. This skill, trivial to humans, remains beyond most robots’ abilities despite continuous progress in robotics research over the past decades. While MPC affords the online synthesis of complex motions, it remains fundamentally limited in its ability to control physical interaction. As a matter of fact, although force sensors have been used since the early days of robotics [184], they remain notably absent from modern control techniques relying on model-based optimization.

This is partly because predicting the evolution of contact forces is challenging in general and involves sophisticated models [41] that are too specific or impractical for real-time applications. Hence, the contact models used in practice for optimization-based control are kept simple for algorithmic convenience [60]. However, these simplifications hinder the ability to derive meaningful control policies in contact with explicit force feedback. To this day, the predictive feedback control of contact forces remains an open problem.

In this thesis, we show that a simple reformulation of the optimal control problem combined with standard estimation tools enables us to achieve state-of-the-



art performance in force control while preserving the benefits of model-predictive control, thereby outperforming traditional force control techniques. This paves the way toward a more systematic integration of force sensors in model predictive control.

## 1.2 Reasoning about uncertainty

While high frequency re-planning can be combined with online estimation to adapt the model of the controller to unforeseen situations, this is not sufficient to safely reason about the risk due to the perception uncertainty. Indeed, by separating estimation and control, the model predictive controller only relies on a state feedback and fully trusts the perception. Consequently, the uncertainty that results from the partial sensing of the world is ignored. In practice, a controller should adapt to the degree of certainty or confidence in the robot’s belief about the world. For instance, coming back to the example depicted in Figure (1.1), unless we have a perfect model of the beam’s slipperiness, we expect a robot to behave conservatively because of the risk of falling.

Nevertheless, the common practice in robotics is to decouple estimation and control (i.e., assume that the certainty equivalence principle holds) [44, 101, 108, 139, 164]. This approach is often chosen due to the availability of separate and tractable control and estimation algorithms that can be deployed on the robot. The estimation module is often a variation of a Gaussian filter, such as an Extended Kalman Filter (EKF) [97], which computes both the mean and uncertainty of the state estimates from sensor information. In control, MPC can then adapt its behavior online based on the current robot and environment states. During

deployment, the estimation module is used to compute the mean of the state estimate, which is then passed on to the controller to compute the optimal behavior [44, 101, 108, 139, 164]. Unfortunately, relying on the most likely outcome can lead to catastrophic behavior. For instance, on a load-carrying task with a quadruped where the mass of the load is unknown, the notion of mean might not be appropriate as this could lead the quadruped to apply insufficient force on the ground and then fall.

Some approaches try to address this issue by adding robustness or safety bounds in either the estimation or control block while keeping them independent. For instance, Robust Extended Kalman filtering [57] adds robustness to inaccuracies of the EKF or the model. However, the control objective is disregarded, and therefore the controller cannot be robust to estimation uncertainties. Robust MPC has been studied and applied to robots. [176] used tube-based MPC to control a biped. [67] used linear stochastic MPC to account for uncertainties in bipedal walking. However, this line of work assumes the state to be known. In contrast to such approaches, we aim to link estimation and control by adding to the estimation module a notion of control performance to improve robustness to the estimation uncertainty.

As Mayne [126] advocates, one meaningful way to properly take into account the measurement uncertainty is to use a minimax formulation linking control and estimation. This problem has been extensively studied in [39, 40], which shows that a specific dynamic game formulation leads to MPC approaches with bounded state trajectories and provides an explicit characterization of these bounds. However, the minimax problem was solved with an interior point method without taking into account the specific structure of the problem and the sparsity induced by time.

In this thesis, we derive an explicit iterative solution that fully exploits sparsity, resulting in an algorithm that linearly scales with the time horizon length and which can be easily warm-started for use in MPC schemes [121].

Despite having been widely studied theoretically, to the best of our knowledge, dynamic game control with imperfect state observations has not been approached from a numerical optimization point of view. In this thesis, we consider the general problem of dynamic game control with imperfect state observation and present a numerically efficient provably convergent algorithm to solve it. The solution presented generalizes commonly used techniques such as the extended Kalman smoother and Differential Dynamic Programming. The proposed solution fully exploits the sparsity of the problem and scales linearly with time, making it a promising method for online reasoning about perception uncertainty.

### 1.3 Statement of purpose

In this thesis, we aim to derive controllers that **reason online about novel situations despite perception uncertainty**. Performing Model Predictive Control on torque-controlled robots is a promising way to synthesize complex and compliant motions in novel environments. However, as of now, MPC practitioners in robotics have suffered from three main limitations. The first limitation is the challenge of incorporating hard constraints. While these are crucial for safety, hard constraints can significantly slow down the compute time and introduce local minima. The second limitation is that existing MPC frameworks can hardly reason about output signals like force. Furthermore, state-feedback MPC naively trusts the most probable state estimate and therefore ignores the risk associated with the

partial sensing of the environment. In this thesis, we propose several contributions to overcome these limitations.

## 1.4 Contributions

In this thesis, we propose a set of algorithmic, experimental, and software contributions.

First, a state-of-the-art solver for nonlinear constrained MPC is introduced and deployed on an industrial manipulator. To the best of our knowledge, this is the first demonstration of closed-loop nonlinear MPC with hard constraints on real hardware. Then, we propose a method to overcome the local behaviors of MPC using function approximation. Furthermore, we show how to endow MPC with force feedback using online estimation and demonstrate how the proposed method results in state-of-the-art performance on tasks involving force tracking.

Second, an efficient numerical method is proposed to solve dynamic game control with imperfect state observation. We demonstrate the ability of the method to reason about risk by directly considering sensor information. This formulation is then used to derive a risk-sensitive filter. Robot experiments demonstrate the ability of the approach to significantly improve performance in the face of uncertainties. To the best of our knowledge, this is the first time that a nonlinear risk-sensitive output-feedback MPC controller has been deployed on a robot.

All the proposed algorithms are accompanied by open-source software to ease reproducibility.

## 1.5 Outline

Part I introduces several contributions in state-feedback Model Predictive Control.

- Chapter 2 provides a comprehensive review of the literature on numerical methods for optimal control and introduces an efficient implementation of an SQP solver relying on a ADMM QP tailored for optimal control.
- Chapter 3 shows how to use function approximation to overcome the local behavior of constrained trajectory optimization.
- Chapter 4 shows how to use online model adaptation to perform force-feedback in MPC.

Part II investigates how online optimization can be used to reason about the risk due to the perception uncertainty by solving the estimation and control problem jointly.

- In Chapter 5, we derive a stagewise Newton method for dynamic game control with imperfect state observations. The proposed solver couples estimation and control by merging an iterative optimal control algorithm similar to minimax DDP and an iterative risk-sensitive Kalman smoother.
- In chapter 6, we present a risk-sensitive Extended Kalman Filter that can adapt its estimation to the control objective, hence allowing safe output-feedback MPC.

## 1.6 Notations

The partial derivative of a function  $f$  with respect to a vector  $v$  is denoted by  $\partial_v f$  or  $f^v$ . The second-order derivatives with respect to vectors  $u, v$  are denoted as  $\partial_u \partial_v f$  or  $f^{uv}$ . The gradient of a scalar function  $f$  with respect to a vector  $v$  is denoted by  $\nabla_v f$ . The Hessian with respect to vectors  $u, v$  is denoted as  $\nabla_{uv}^2 f$ . Bold characters are used to denote a sequence of vectors indexed over a time horizon, e.g.,  $\mathbf{v} = \{v_1, \dots, v_k, \dots\}$ . If  $(v_i)_{i \in \mathbb{N}}$  is a sequence of vectors, then  $v_{k:t}$  is a vector concatenating  $v_k \dots v_t$ .  $\mathbf{1}_{x \in A}$  is the indicator function, which equals 1 if  $x \in A$  and 0 otherwise.  $I_n$  denotes the identity matrix of size  $n$  by  $n$ .

## Part I

# Model Predictive Control





## Chapter 2

# Stagewise Resolution of Optimal Control Problems

The promise of model-predictive control in robotics has led to extensive development of efficient numerical optimal control solvers in line with differential dynamic programming because it exploits the sparsity induced by time. In this Chapter<sup>1</sup>, we argue that this effervescence has hidden the fact that sparsity can be equally exploited by standard nonlinear optimization. In particular, we show how a tailored implementation of sequential quadratic programming achieves state-of-the-art model-predictive control. Then, we clarify the connections between popular algorithms from the robotics community and well-established optimization techniques. Further, the sequential quadratic program formulation naturally encompasses the

---

<sup>1</sup>This chapter is adapted from the original publication : **A. Jordana\***, S. Kleff\*, A. Meduri\*, et al., "Stagewise implementations of Sequential Quadratic Programming for Model-Predictive Control. Submitted to Transactions on Robotics (T-RO). This work is the result of a collaboration with equal contribution between Armand Jordana, Sebastien Kleff, and Avadesh Meduri. In particular, the dissertation's author (Armand Jordana) contributed to the development and implementation of the algorithms, the literature review, the hardware experiments, the benchmarks, and open-sourcing of the mim-solvers library.

constrained case, a notoriously difficult problem in the robotics community. Specifically, we show that it only requires a sparsity-exploiting implementation of a state-of-the-art quadratic programming solver. We illustrate the validity of this approach in a comparative study and experiments on a torque-controlled manipulator. To the best of our knowledge, this is the first demonstration of closed-loop nonlinear model-predictive control with constraints on a real robot.

Mayne first introduced DDP [125] as an efficient algorithm to solve nonlinear OCPs by iteratively applying a backward pass over the time horizon and a nonlinear forward rollout of the dynamics. This algorithm notably exhibits linear complexity in the time horizon and local quadratic convergence [137]. More recently, Todorov revived the interest in DDP by proposing the iterative Linear Quadratic Regulator (iLQR) [114], a variant discarding the second-order terms of the dynamics. It has since gained a lot of traction within the robotics community [93, 104, 120, 139, 164], and its similarity to Gauss-Newton optimization has been established [14, 161]. This family of algorithms is often referred to as single-shooting algorithms [68]. All of them only optimize over the control inputs while the state variables are indirectly optimized through the dynamic constraints using either a linear or non-linear rollout. Single shooting methods are efficient because they optimize over fewer variables. However, these approaches face two main limitations: 1) as a single shooting method, it requires a dynamically feasible initial guess and therefore, it can be difficult to warm-start it with a specific state trajectory, an essential requirement to reduce computation times [68] and 2) enforcing equality and inequality constraints is not straightforward. The common practice is to enforce constraints softly using penalty terms in the cost function. But this approach is heuristic (i.e., it requires cost weight tuning) and tends to cause numerical issues [71].

Bock and Plitt [26] introduced multiple shooting for optimal control to address the first limitation: it accepts an infeasible initial guess. Multiple shooting methods optimize over both the state and control variables independently, which means they do not need dynamic feasibility at the start. During optimization, multiple shooting methods reduce the dynamic infeasibility or gaps between the state and control variables. Finally, at convergence, a dynamically feasible solution that minimizes the cost is returned [68]. Later, several multiple-shooting variants of DDP/iLQR were proposed [68, 120] with significantly improved convergence abilities, which have enabled nonlinear MPC at high frequency on real robots [44, 102, 120, 139]. Hybrid algorithms - algorithms that use a combination of single and multiple shooting techniques - such as iLQR-GNMS [68] have also been proposed.

The second issue of enforcing constraints inside a DDP-like algorithm has been addressed in several works. Tassa et al. [170] use a DDP-based projected Newton method to bound control inputs. This approach has further been improved and deployed on a real quadruped robot in [122]. More recently, augmented Lagrangian methods have been used to enforce constraints in iLQR/DDP algorithms [5, 87, 92, 148]. However, their convergence behavior is less understood than DDP, whose seminal paper [125] was followed by sophisticated proofs [137]. To the best of our knowledge, it has not yet been shown that those recent DDP-based algorithms exhibit global convergence (i.e., convergence from any initial point to a stationary point) and quadratic local convergence.

Besides, an open topic of debate is whether to use a nonlinear or a linear rollout to enforce dynamics constraints in the forward pass. Previous work on single shooting methods has argued that a nonlinear rollout can reduce the number of iterations [115], while Zimmermann et al. [196] & Baumgärtner et al. [14] showed

mixed results both experimentally and theoretically. Recently, a nonlinear rollout was implemented by a popular multiple-shooting variant of iLQR [120]. However, we are unaware of any comparisons between linear and nonlinear rollouts in the context of multiple shooting methods.

In the face of these challenges, we propose to take a fresh look at the earlier literature. Indeed, Dunn et al. showed that Newton’s method could also be implemented in a DDP-like fashion and equally benefit from linear complexity in the time horizon and quadratic convergence [55]. This finding indicates that optimal control does not fundamentally require new nonlinear optimization tools but only tailored implementations that exploit the time-induced sparsity structure. This naturally has led to numerous extensions to the constrained case. For example, Wright proposed to use an SQP formulation with an active set to address arbitrary nonlinear constraints [187]. However, because of the limitations of active set methods, Wright focused on the single shooting case resolution. Then, Pantoja et al. proposed an efficient implementation of SQP for control inequality constraints [146]. Di Pillo et al. proposed a tailored implementation of a Quasi-Newton method with an augmented Lagrangian-based approach [46]. Dohrmann et al. studied equality-SQP for optimal control [51]. Additionally, several others studied the tailored implementation of Interior Point Methods for optimal control [52, 56, 150, 166, 189].

In the late 2000’s, with the promise of high-frequency linear MPC, a large body of work studied how to tailor Quadratic Programming (QP) for the Constrained Linear Quadratic Regulator (CLQR) problem. This was done with active set methods [61, 152], ADMM-based solvers [144] and interior-point methods [65, 183]. We refer the reader to Kouzoupis et al. [106] for an extensive survey. More recently, efficient ADMM implementations for GPUs and onboard microcontrollers have

been proposed [2, 25].

To solve the nonlinear case, Verschueren et al. [174] recently proposed efficient software with an SQP implementation for OCP. Domahidi et al. [53] proposed to use interior points with a tailored implementation in the case of affine dynamics. More recently, Vanroye et al. [173] studied how to exploit the sparsity induced by time in IPOPT [180]. Unfortunately, this line of work has not benefited from as much experimental study as DDP-like algorithms. Recently, Gragnia et al. [72] showed impressive experimental results on quadrupeds using a tailored SQP implementation based on HPIPM [65], which lies in the continuity of previous works using [65, 174] on real hardware [28, 145, 153]. However, to the best of our knowledge, we are not aware of closed-loop constrained nonlinear MPC on torque-controlled robots.

Recently, Katayama et al. [98] reviewed the models and optimization algorithms used in robotics in the context of MPC. As emphasized in this survey, we argue that there is a gap between the optimization-based control and the robotics community. On the one hand, an important part of the robotics community followed the successes of Tassa et al. [169] and continued to propose DDP-like algorithms. On the other hand, the optimization-based control community followed the work of Wright and Pantoja et al. [146, 187] and proposed efficient implementations of established optimization algorithms [106, 173, 174]. In this thesis, we aim to bridge this gap.

In this Chapter, we follow the line of thought of the optimization-based control community in order to push the limits of closed-loop nonlinear MPC in robotics. First, we shed light on the direct connection between modern multiple-shooting DDP-like algorithms and textbook SQP algorithms. Second, we show through an experimental study that a standard stagewise SQP formulation is, in fact, superior

to the state-of-the-art FDDP [120]. Third, we propose a stagewise alternating direction method of multipliers (ADMM) QP solver inspired by OSQP [167]. The proposed QP solver, OSQP\_OCP, leverages Riccati recursions in order to maintain linear complexity in the time horizon while also enforcing constraints. Using this custom QP implementation inside the SQP formulation, we can solve arbitrary nonlinear constrained OCPs efficiently while inheriting the well-known convergence properties of standard SQPs. Lastly, we demonstrate the ability of this SQP formulation to perform closed loop nonlinear constrained MPC on a torque-controlled manipulator. To the best of our knowledge, this is the first demonstration of closed-loop nonlinear MPC with hard constraints on a real robot. The optimization software is integrated with Crocoddyl [120] and is available open-source ([https://github.com/machines-in-motion/mim\\_solvers](https://github.com/machines-in-motion/mim_solvers)).

## 2.1 Sequential Quadratic Programming for Optimal Control Problems

In this Chapter, we study constrained optimal control problems of the form:

$$\min_{\mathbf{x}, \mathbf{u}} \sum_{k=0}^{T-1} \ell_k(x_k, u_k) + \ell_T(x_T) \quad (2.1a)$$

$$\text{subject to } x_{k+1} = f_k(x_k, u_k), \quad 0 \leq k < T, \quad (2.1b)$$

$$c_k(x_k, u_k) \geq 0, \quad 0 \leq k < T, \quad (2.1c)$$

$$c_T(x_T) \geq 0, \quad (2.1d)$$

where  $\mathbf{x} = (x_1, \dots, x_T)$  is the state sequence,  $\mathbf{u} = (u_0, u_1, \dots, u_{T-1})$  the control inputs and  $x_0$  is the initial state provided by the user (e.g., estimated state). The transition functions,  $f_k$ , the cost functions,  $\ell_k$  and the constraint functions,  $c_k$ , are supposed to be twice differentiable. To keep notations simple, we formulated the OCP with inequality, which also encompasses equality constraints. Yet, equality constraints can be treated separately from general inequalities in practice.

Considering the state sequence as an optimization variable allows the optimization procedure to iterate through infeasible trajectories, yielding a **multiple-shooting** approach [26].

**Remark 1.** *One may choose to optimize only on the sequence of control inputs and define the dynamics constraints implicitly, yielding a **single shooting** approach. In that case, if no inequality constraints are considered, the problem is unconstrained and can be solved via an efficient implementation of Newton's method [55] or with a modified Newton's method with a nonlinear rollout, namely DDP [125].*

In this thesis, we focus on multiple-shooting approaches. The associated Lagrangian is:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = & \ell_T(x_T) - \mu_T^\top c_T(x_T) + \\ & \sum_{k=0}^{T-1} \ell_k(x_k, u_k) - \lambda_{k+1}^\top (x_{k+1} - f_k(x_k, u_k)) - \mu_k^\top c_k(x_k, u_k), \end{aligned} \quad (2.2)$$

where  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_T)^\top$  and  $\boldsymbol{\mu} = (\mu_0, \dots, \mu_T)^\top$  are Lagrange multipliers.

Intuitively, SQPs iteratively solve QPs to find a tuple  $(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu})$  that satisfies the KKT conditions [143]. Note that we consider the constraint  $x_0 = \bar{x}_0$  to be implicit. Therefore, no Lagrange multiplier is associated with that constraint. For

simplicity, we slightly abuse the notation by referring to  $l_0(x_0, u_0)$  instead of  $l_0(u_0)$  even if  $x_0$  is fixed.

**Remark 2.** *Note that one might also want to consider a generic constraint on the initial condition  $c(x_0)$  (e.g., in the context of robust control [124]). To do so, the initial condition must be considered an optimization variable. Although we do not consider this setting to clarify derivations, our work naturally extends to this case.*

More precisely, at the  $n^{\text{th}}$  iteration, given a guess on the tuple  $(\mathbf{x}^{[n]}, \mathbf{u}^{[n]}, \boldsymbol{\lambda}^{[n]}, \boldsymbol{\mu}^{[n]})$ , we aim to find a correction on the guess by solving the following QP problem (Algorithm 18.1, [143]):

$$\begin{aligned} \min_{\Delta \mathbf{x}, \Delta \mathbf{u}} \sum_{k=0}^{T-1} \frac{1}{2} & \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k \\ S_k^\top & R_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^\top \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \\ & + \frac{1}{2} \Delta x_T^\top Q_T \Delta x_T + \Delta x_T^\top q_T \end{aligned} \quad (2.3a)$$

$$\text{s.t. } \Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k + \gamma_{k+1}, 0 \leq k < T, \quad (2.3b)$$

$$D_k \Delta x_k + E_k \Delta u_k + c_k \geq 0, \quad 0 \leq k < T. \quad (2.3c)$$

$$D_T \Delta x_T + c_T \geq 0. \quad (2.3d)$$

where  $q_T = \nabla_x \ell_T(x_T^{[n]})$  and

$$Q_T = (\nabla_{xx}^2 \ell_T - \mu_T^\top \nabla_{xx}^2 c_T)(x_T^{[n]}) \quad (2.4)$$

$$Q_k = (\nabla_{xx}^2 \ell_k + \lambda_{k+1}^\top \nabla_{xx}^2 f_k - \mu_k^\top \nabla_{xx}^2 c_k)(x_k^{[n]}, u_k^{[n]})$$

$$S_k = (\nabla_{xu}^2 \ell_k + \lambda_{k+1}^\top \nabla_{xu}^2 f_k - \mu_k^\top \nabla_{xu}^2 c_k)(x_k^{[n]}, u_k^{[n]})$$

$$R_k = (\nabla_{uu}^2 \ell_k + \lambda_{k+1}^\top \nabla_{uu}^2 f_k - \mu_k^\top \nabla_{uu}^2 c_k)(x_k^{[n]}, u_k^{[n]})$$



$$\begin{aligned}
A_k &= \nabla_x f_k(x_k^{[n]}, u_k^{[n]}), \quad B_k = \nabla_u f_k(x_k^{[n]}, u_k^{[n]}) \quad 0 \leq k < T. \\
q_k &= \nabla_x \ell_k(x_k^{[n]}, u_k^{[n]}), \quad r_k = \nabla_u \ell_k(x_k^{[n]}, u_k^{[n]}) \\
D_k &= \nabla_x c_k(x_k^{[n]}, u_k^{[n]}), \quad E_k = \nabla_u c_k(x_k^{[n]}, u_k^{[n]}) \\
c_k &= c_k(x_k^{[n]}, u_k^{[n]}), \quad D_T = \nabla_x c_T(x_k^{[n]})
\end{aligned} \tag{2.5}$$

$Q_k, S_k, R_k$  are the Hessians of the Lagrangian with respect to the state and control inputs,  $A_k$  and  $B_k$  are the dynamics Jacobian,  $D_k, E_k, D_T$  are the constraint Jacobian and  $\gamma_{k+1} = f_k(x_k^{[n]}, u_k^{[n]}) - x_{k+1}^{[n]}$  is the constraint violation which is often referred to as dynamic gaps [120] or defects [68]. Note that this formulation of SQP and its associated QP is precisely the same as the one proposed in the seminal multiple-shooting article [26].

**Remark 3.** *In his seminal work introducing multiple shooting [26], Bock exploits the sparsity of the quadratic problem of Eq. (2.3) induced by the multiple shooting structure yielding a cubic complexity in the time horizon,  $T$ . Instead, this work exploits the sparsity induced by time in order to achieve a linear complexity.*

The solution of the QP  $(\Delta \mathbf{x}, \Delta \mathbf{u})$  is then used to update the guess:

$$\mathbf{x}^{[n+1]} = \mathbf{x}^{[n]} + \alpha \Delta \mathbf{x} \tag{2.6a}$$

$$\mathbf{u}^{[n+1]} = \mathbf{u}^{[n]} + \alpha \Delta \mathbf{u} \tag{2.6b}$$

Here,  $\alpha$  is the step size and is chosen using a line search method.  $\boldsymbol{\lambda}^{[n+1]}$  and  $\boldsymbol{\mu}^{[n+1]}$  are then replaced by the associated Lagrange multiplier of the QP (2.3) [143]. Provided assumptions on the Problem (2.1), SQPs can guarantee local quadratic convergence or super-linear convergence in the case of quasi-Newton approximation [143].

## 2.2 Solving the unconstrained QP

In this section, we review the case when the OCP has no constraints to recall the equivalence between the backward Riccati recursions used in DDP and a special type of Gaussian elimination for tridiagonal matrices, specialized to the sparsity pattern of the KKT system arising in OCPs. This result, insufficiently known in the robotics community, will then be exploited to propose a novel extension to the constrained case.

**Proposition 1.** *Without inequality constraints, the KKT conditions of Problem (2.3) can be written as a block tri-diagonal symmetric matrix equation.*

$$\begin{bmatrix} \Gamma_1 & M_1^\top & 0 & 0 & \dots & 0 \\ M_1 & \Gamma_2 & M_2^\top & 0 & \dots & 0 \\ 0 & M_2 & \Gamma_3 & M_3^\top & \dots & 0 \\ 0 & 0 & M_3 & \Gamma_4 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & \Gamma_T \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ s_T \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \vdots \\ g_T \end{bmatrix} \quad (2.7)$$

where:

$$\Gamma_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^\top \\ 0 & Q_k & I \\ -B_{k-1} & I & 0 \end{bmatrix}, \quad M_k = \begin{bmatrix} 0 & S_k^\top & 0 \\ 0 & 0 & 0 \\ 0 & -A_k & 0 \end{bmatrix},$$

$$\text{and } s_k = \begin{bmatrix} \Delta u_{k-1} \\ \Delta x_k \\ -\lambda_k \end{bmatrix}, \quad g_k = \begin{bmatrix} -r_{k-1} \\ -q_k \\ \gamma_k \end{bmatrix}. \quad (2.8)$$

The proof is only computational and is detailed in the supplementary material [A](#). Note that we slightly abuse the notation by using the  $\lambda$  as the Lagrange multiplier for both the QP and the nonlinear problem. The most straightforward way to solve such a system is to apply Gaussian elimination by using the well-known Thomas algorithm [111] (Algorithm 1) to achieve a complexity in  $O(Tm^3)$  where  $m$  is the size of the control vector.

---

**Algorithm 1:** Thomas algorithm

---

```

1  $\bar{\Gamma}_T \leftarrow \Gamma_T$ 
2  $\bar{g}_T \leftarrow \Gamma_T^{-1} g_T$ 
   /* backward pass */
3 for  $k \leftarrow 1$  to  $T - 1$  do
4    $\bar{\Gamma}_k \leftarrow \Gamma_k - M_k^\top \bar{\Gamma}_{k+1}^{-1} M_k$ 
5    $\bar{g}_k \leftarrow \bar{\Gamma}_k^{-1} (g_k - M_k^\top \bar{g}_{k+1})$ 
   /* forward pass */
6  $s_1 \leftarrow \bar{g}_1$ 
7 for  $k \leftarrow 1$  to  $T - 1$  do
8    $s_{k+1} \leftarrow \bar{g}_{k+1} - \bar{\Gamma}_{k+1}^{-1} M_k s_k$ 

```

---

In particular, we use the knowledge of the structure (sparsity pattern) in both  $\Gamma_k$  and  $M_k$  to recover simpler recursions. This leads to another way of obtaining the backward Riccati equations of LQR, which is accepted knowledge in the optimization-based control community [14, 47, 55, 106, 187] but is largely ignored in robotics.

**Proposition 2.** *By applying the Thomas algorithm, we recover the well-known Riccati recursions. Specifically, the **backward pass** can be done by initializing*

$V_T = Q_T$  and  $v_T = q_T$ , and then by applying the following equations:

$$h_k = r_k + B_k^\top (v_{k+1} + V_{k+1} \gamma_{k+1}) \quad (2.9)$$

$$G_k = S_k^\top + B_n^\top V_{k+1} A_k \quad K_k = -H_k^{-1} G_k$$

$$H_k = R_k + B_k^\top V_{k+1} B_k \quad k_k = -H_k^{-1} h_k$$

$$V_k = Q_k + A_k^\top V_{k+1} A_k - K_k^\top H_k K_k$$

$$v_k = q_k + K_k^\top r_k + (A_k + K_k B_k)^\top (v_{k+1} + V_{k+1} \gamma_{k+1})$$

Then, the **forward pass** initializes  $\Delta x_0 = 0$  and unrolls the linearized dynamics:

$$\Delta x_{k+1} = (A_k + B_k K_k) \Delta x_k + B_k k_k + \gamma_{k+1} \quad (2.10a)$$

$$\Delta u_k = K_k \Delta x_k + k_k \quad (2.10b)$$

$$\lambda_k = V_k \Delta x_k + v_k \quad (2.10c)$$

*Proof.* The proof is mainly computational and relies on the analytical inversion of  $\bar{\Gamma}_k$  in order to prove by recursion that:

$$\bar{\Gamma}_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^\top \\ 0 & V_k & I \\ -B_{k-1} & I & 0 \end{bmatrix} \quad (2.11a)$$

$$\bar{g}_k = \bar{\Gamma}_k^{-1} \begin{bmatrix} -r_{k-1} \\ -v_k \\ \gamma_k \end{bmatrix} \quad (2.11b)$$

In other words, the recursion on  $\bar{\Gamma}_k$  and  $\bar{g}_k$  can be substituted by the recursion on the value function Hessian and gradient. The forward recursion can then be

recovered using  $s_{k+1} = \bar{g}_{k+1} - \bar{\Gamma}_{k+1}^{-1} M_k s_k$  and the analytical inversion of  $\bar{\Gamma}_k$ . Detailed derivations are provided in the supplementary material [A](#).  $\square$

This result shows that the backward Riccati equations used for Linear Quadratic Regulators (LQR) are an efficient technique for solving Quadratic Programs with a tri-diagonal symmetric KKT matrix. Consequently, these equations are ideal for solving OCPs with linear dynamics and quadratic costs.

**Remark 4.** *A vast amount of literature exists on tri-diagonal matrices. Understanding the structure of the KKT matrices inherent to OCP enables the use of any of these techniques directly. There are methods to factor those matrices or to solve them more efficiently. For instance, parallel cyclic reduction can be used in order to achieve a  $O(\log(T))$  complexity, which might be interesting for problems with long time horizons [\[63, 142, 155, 188\]](#).*

## 2.3 Solving the constrained QP

We are now in the position to extend the discussion of the previous section to the case with inequality constraints. When inequalities are added, the underlying KKT system associated with the QP defined in [\(2.3\)](#) will have the same tri-diagonal structure. Therefore, similar recursions can be used to solve it efficiently. For instance, HPIPM [\[65\]](#) derives efficiently an interior point method. We chose to tailor OSQP [\[167\]](#) for OCPs because it can handle infeasibility better than HPIPM and ADMM-based methods are known to find a reasonably good solution in a few iterations [\[27, 64\]](#), two appealing features for MPC applications. Finally, as originally discussed by [\[144\]](#), ADMM-based methods can easily be implemented sequentially. We illustrate this by deriving a tailored implementation of the modern

ADMM-based solver OSQP [167] for OCP. However, we would like to highlight that the reader could choose any other desired sparsity-exploiting QP solver depending on their preference.

### 2.3.1 Background on ADMM

Here, we follow [27, 167] to briefly summarize ADMM. Using generic notations, Problem (2.3) aims to solve a problem of the form:

$$\min_{v \in \text{Dom}(g)} g(v) \text{ s.t. } Pv \in \mathcal{C} \quad (2.12)$$

Note that we use generic notations of the QP literature in this section. Hence,  $v$  denotes the optimization variable. Furthermore,  $P$  is a matrix,  $g$  is a convex function, and  $\mathcal{C}$  is a convex set. The ADMM updates are then:

$$\begin{aligned} \tilde{v}^{j+1} = \underset{v \in \text{Dom}(g)}{\text{argmin}} & g(v) + \frac{\rho}{2} \|Pv - z^j + \rho^{-1}y^j\|_2^2 \\ & + \frac{\sigma}{2} \|v - v^j\|_2^2 \end{aligned} \quad (2.13a)$$

$$\tilde{z}^{j+1} = \alpha P \tilde{v}^{j+1} + (1 - \alpha) z^j \quad (2.13b)$$

$$v^{j+1} = \alpha \tilde{v}^{j+1} + (1 - \alpha) v^j \quad (2.13c)$$

$$z^{j+1} = \Pi_{\mathcal{C}} (\tilde{z}^{j+1} + \rho^{-1}y^j) \quad (2.13d)$$

$$y^{j+1} = y^j + \rho (\tilde{z}^{j+1} - z^{j+1}) \quad (2.13e)$$

where  $\Pi_{\mathcal{C}}$  is the projection operator on the convex set  $\mathcal{C}$ ,  $\rho$  is the penalty parameter that encourages consensus between  $v$  and  $z$ , and  $\alpha \in (0, 2)$  is an over-relaxation parameter that improves convergence speed (e.g., when set between 1.5 and 1.8 [27]).

### 2.3.2 Tailored ADMM for optimal control

Here, we present the ADMM-based QP solver [27], which is tailored for optimal control problems. The presented algorithm is mainly inspired by [144]. We further introduce more recent techniques developed in ADMM [27, 167] to the previous work [144] to improve solver performance. Finally, we provide an efficient implementation of the proposed solver, enabling reliable real-robot deployment.

In order to exploit the time-induced sparsity and leverage the Riccati recursions specific to the OCP formulation, we design a QP solver that always maintains the feasibility of the dynamics. More precisely, we consider the optimization variable to be

$$v \triangleq (\Delta \mathbf{x}, \Delta \mathbf{u})^\top \quad (2.14)$$

and  $g$  to be the cumulative cost function defined in Eq. (2.3a). We propose to define  $\text{Dom}(g)$  to encode the set of feasible linearized dynamics (2.3b) while  $Pv \in \mathcal{C}$  encodes the path and terminal constraint (2.3c) (2.3d). Importantly, considering the linearized dynamics throughout the domain of  $g$  (instead of incorporating them in  $Pv \in \mathcal{C}$ ) is key to leverage the Riccati recursions. This choice makes our solver differ from OSQP and we will show that this can lead to a faster convergence in terms of the number of iterations. In the end, Eq. (2.13a) can therefore be written as:

$$\begin{aligned}
& \min_{\Delta \mathbf{x}, \Delta \mathbf{u}} \sum_{k=0}^{T-1} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k \\ S_k^\top & R_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^\top \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \\
& + \Delta x_T^\top Q_T \Delta x_T + \Delta x_T^\top q_T + \frac{\rho}{2} \|D_T \Delta x_T - z_T^j + \rho^{-1} y_T^j\|_2^2 \\
& + \sum_{k=0}^{T-1} \frac{\rho}{2} \|D_k \Delta x_k + E_k \Delta u_k - z_k^j + \rho^{-1} y_k^j\|_2^2 \\
& + \sum_{k=0}^T \frac{\sigma}{2} \|\Delta x_k - \Delta x_k^j\|_2^2 + \sum_{k=0}^{T-1} \frac{\sigma}{2} \|\Delta u_k - \Delta u_k^j\|_2^2 \tag{2.15a}
\end{aligned}$$

$$\text{s.t.} \quad \Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k + \gamma_{k+1}. \tag{2.15b}$$

We denote  $\mathbf{y}^j = (y_0^j, y_1^j, \dots, y_T^j)^\top$  and  $\mathbf{z}^j = (z_0^j, z_1^j, \dots, z_T^j)^\top$ . Therefore, Eq. (2.13a) has the exact same structure as the unconstrained case mentioned in Section 2.2. Note that the stagewise nature of the inequality constraints allows us to write the  $\rho$  dependent regularization terms in a block-sparse way as well. More precisely, as each inequality only depends on one time-step, the  $\rho$  dependent regularization term can be written as a stagewise sum of quadratic cost. Consequently, Eq. (2.13a) can be solved with a backward and forward pass similar to those of the unconstrained case, ensuring a linear complexity in the time horizon. Furthermore, due to the stagewise nature of the inequality constraint, Eq. (2.13b) - (2.13e) can be implemented recursively or in parallel. Algorithm 2 summarizes the procedure.

### 2.3.3 Details on the QP

In our OSQP\_OCP, we take  $\sigma = 10^{-6}$ ,  $\alpha = 1.6$  and consider the same schedule as OSQP for the  $\rho$ -update [167]. Subsequently, the new backward recursion is only



---

**Algorithm 2:** ADMM tailored for OCP

---

**Input:**  $\Delta \mathbf{x}^j, \Delta \mathbf{u}^j, \mathbf{z}^j, \mathbf{y}^j$   
 /\* Minimization problem \*/  
 1 Solve (2.15) using LQR to get  $\widetilde{\Delta \mathbf{x}}^{j+1}, \widetilde{\Delta \mathbf{u}}^{j+1}$   
 /\* Lagrange multiplier update \*/  
 2 **for**  $k \leftarrow 1$  **to**  $T - 1$  **do**  
 3      $\tilde{z}_k^j = \alpha(D_k \Delta x_k^{j+1} + E_k \Delta u_k^{j+1}) + (1 - \alpha)z_k^j$   
 4      $\Delta x_k^{j+1} = \alpha \widetilde{\Delta x}_k^{j+1} + (1 - \alpha) \Delta x_k^j$   
 5      $\Delta u_k^{j+1} = \alpha \widetilde{\Delta u}_k^{j+1} + (1 - \alpha) \Delta u_k^j$   
 6      $z_k^j = \max(c_k, \tilde{z}_k^j + \rho^{-1} y_k^j)$   
 7      $y_k^{j+1} = y_k^j + \rho(\tilde{z}_k^j - z_k^{j+1})$   
 8      $\tilde{z}_T^j = \alpha D_T \Delta x_T^{j+1} + (1 - \alpha) z_T^j$   
 9      $\Delta x_T^{j+1} = \alpha \widetilde{\Delta x}_T^{j+1} + (1 - \alpha) \Delta x_T^j$   
 10     $z_T^j = \max(c_T, \tilde{z}_T^j + \rho^{-1} y_T^j)$   
 11     $y_T^{j+1} = y_T^j + \rho(\tilde{z}_T^j - z_T^{j+1})$   
**Output:**  $\Delta \mathbf{x}^{j+1}, \Delta \mathbf{u}^{j+1}, \mathbf{z}^{j+1}, \mathbf{y}^{j+1}$

---

needed when  $\rho$  is updated (once in 25 ADMM iterations). This makes the algorithm highly efficient when compared to a standard QP solver and the solver proposed in [144] because the forward passes are very cheap (only matrix-vector multiplications), and the inversion in the backward pass happens very few times. Furthermore, the QP is warm-started by the solution of the problem with only equality constraint (dynamic-feasibility). This comes at the cost of Riccati recursions but dramatically reduces the total number of ADMM iterations required. Lastly, we compute a relative primal and dual tolerance after each ADMM iteration as discussed in [27]. We use these quantities as termination criteria.

## 2.4 Practical implementation of the SQP

### 2.4.1 Gauss-Newton approximation

A common practice is to ignore the second-order term of the constraints as it is expensive to compute [68, 114, 120]. Subsequently, the second-order cost terms become:

$$\begin{aligned} Q_T &= \nabla_{xx}^2 \ell_T(x_T^{[n]}), & Q_k &= \nabla_{xx}^2 \ell_k(x_k^{[n]}, u_k^{[n]}) \\ S_k &= \nabla_{xu}^2 \ell_k(x_k^{[n]}, u_k^{[n]}), & R_k &= \nabla_{uu}^2 \ell_k(x_k^{[n]}, u_k^{[n]}), \end{aligned} \quad (2.16)$$

for  $0 \leq k < T$ . A direct consequence is that (2.3) is now independent of the multipliers. In this unconstrained case, the resulting SQP can be efficiently solved by sequentially constructing a QP at each iterate and solving it with LQR. This method was initially proposed as the Gauss-Newton Multiple Shooting (GNMS) method in [68]. However, the connection to SQPs was not discussed in the paper. Thus, GNMS is an efficient SQP algorithm to solve equality-constrained nonlinear optimization problems with block separable constraints and costs when the second-order terms are ignored.

**Remark 5.** *Note that more sophisticated schemes estimating the exact Hessians iteratively could be used as done in [26]. Furthermore, in the unconstrained case, the recent work of [141] suggests that the convergence benefits might compensate for the computational requirement of the second-order computations.*

### 2.4.2 Linear rollout

By nature of the SQP algorithm, the update step (2.6) is equivalent to making a linear rollout of the dynamics (cf. (2.10a)). However, a nonlinear rollout is often favored in DDP-like algorithms, e.g., in the popular Feasible-DDP algorithm [120] or in ALTRO [87]. While [125, 137] studied well the local and global convergence property of DDP, to the best of our knowledge there is no guarantee that these nonlinear rollout formulations maintain global convergence in the multiple shooting context. Hence, a theoretical analysis of algorithms such as Feasible-DDP remains to be done. In contrast, SQPs benefit from a very exhaustive theoretical analysis, convergence guarantees, and experimental validation in a variety of fields [143]. Furthermore, allowing the gaps to close in one step, as in [120], seems to go against the spirit of the original formulation of multiple shooting with SQP [26].

### 2.4.3 Line search

Now that we essentially use SQPs with a QP that exploits the block sparsity, we can leverage the vast literature of line search algorithms to select the right step length (Eq. (2.6)). We compared both a filter line-search [62] and a merit function-based line-search [143]. Let's define the total cost of a trajectory, the total gap violation, and the total constraint violation as:

$$\begin{aligned}\ell(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) &= \sum_{k=0}^{T-1} \ell_k(x_k^{[j]}, u_k^{[j]}) + \ell_T(x_T^{[j]}), \\ \gamma(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) &= \sum_{k=0}^{T-1} \left\| x_{k+1}^{[j]} - f_k(x_k^{[j]}, u_k^{[j]}) \right\|_{\infty}, \\ \mathbf{c}(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) &= \sum_{k=0}^{T-1} \left\| \left[ c_k(x_k^{[j]}, u_k^{[j]}) \right]_- \right\|_{\infty} + \left\| \left[ c_T(x_T^{[j]}) \right]_- \right\|_{\infty}.\end{aligned}\tag{2.17}$$

With the filter line search, a candidate  $\mathbf{x}^{[n+1]}, \mathbf{u}^{[n+1]}$  is accepted if:

$$\forall j \leq n, \quad \ell(\mathbf{x}^{[n+1]}, \mathbf{u}^{[n+1]}) < \ell(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) \quad (2.18a)$$

$$\text{or } \forall j \leq n, \quad \gamma(\mathbf{x}^{[n+1]}, \mathbf{u}^{[n+1]}) < \gamma(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) \quad (2.18b)$$

$$\text{or } \forall j \leq n, \quad \mathbf{c}(\mathbf{x}^{[n+1]}, \mathbf{u}^{[n+1]}) < \mathbf{c}(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) \quad (2.18c)$$

In contrast, the merit function is of the form:

$$\ell(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) + \mu_\gamma \gamma(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}) + \mu_c \mathbf{c}(\mathbf{x}^{[j]}, \mathbf{u}^{[j]}), \quad (2.19)$$

where  $\mu_\gamma$  and  $\mu_c$  are weights defined by the user. A step is accepted if the candidate allows a decrease in the merit function. The downside of the merit function is that the weight between each term has to be tuned, which can be cumbersome. On the other hand, we found that the filter line search yielded good performances and was more practical as no parameter tuning was required. In this filter line search, the cost values, gap norm, and constraint violation of all the previous iterates are stored and browsed. In our implementation, we provide the possibility of keeping only a restricted number of iterates in memory and refer to this parameter as the filter size. Although a filter keeping all the past estimates can help with problems that require many iterations (e.g., to meet a strict tolerance), it can reasonably be shortened or even discarded in practice for MPC applications. In our experience, one can select a filter size of 1 without significant loss in convergence.

#### 2.4.4 Termination criteria

The solver is terminated once the infinity norm of the KKT condition is below a certain threshold. More precisely, the following termination criterion is used:

$$\begin{aligned}
\|\nabla_x \mathcal{L}(\mathbf{x}^{[n]}, \mathbf{u}^{[n]}, \boldsymbol{\lambda}^{[n]}, \boldsymbol{\mu}^{[n]})\|_\infty &\leq \epsilon_{SQP} \\
\|\nabla_u \mathcal{L}(\mathbf{x}^{[n]}, \mathbf{u}^{[n]}, \boldsymbol{\lambda}^{[n]}, \boldsymbol{\mu}^{[n]})\|_\infty &\leq \epsilon_{SQP} \\
\|x_{k+1}^{[n]} - f_k(x_k^{[n]}, u_k^{[n]})\|_\infty &\leq \epsilon_{SQP}, \quad 0 \leq k < T, \\
\left\| \begin{bmatrix} c_k(x_k^{[n]}, u_k^{[n]}) \end{bmatrix}_- \right\|_\infty &\leq \epsilon_{SQP}, \quad 0 \leq k < T. \\
\left\| \begin{bmatrix} c_T(x_T^{[n]}) \end{bmatrix}_- \right\|_\infty &\leq \epsilon_{SQP}.
\end{aligned} \tag{2.20}$$

We use here the convergence criteria widely used in the optimization community [143]. Note that we do not employ a real-time iteration scheme [48] as we found in practice that letting the solver converge led to better experimental results than re-planning faster with a single SQP iteration.

As mentioned in [143],  $\boldsymbol{\lambda}^{[n+1]}, \boldsymbol{\mu}^{[n+1]}$  are given by the Lagrange multiplier associated with (2.3). Note that we can compute them very efficiently due to the recursions. In the unconstrained case, we can use (2.10c). In the constrained case, we have,

$$\lambda_k = V_k \widetilde{\Delta x_k} + v_k \tag{2.21}$$

$$\mu_k = y_k \tag{2.22}$$

where  $V_k, v_k$  are derived from the Riccati recursions defined by Problem (2.15) and where  $\widetilde{\Delta x_k}, y_k$  are the primal and dual solutions of the QP (at time  $k$ ).

## 2.5 Experiments

In this section, we demonstrate the practical benefits of using a sparsity-exploiting SQP implementation for nonlinear MPC with and without inequality constraints. The solvers used in the benchmarks and experiments are open-sourced in the `mim_solvers` library<sup>2</sup>, which uses `Crocodyl` [120] as a base software. Rigid-body dynamics computations are done using the `Pinocchio` library [33] and its analytical derivatives [31]. All the benchmarks and figures presented in this Chapter can be reproduced using the dedicated repository `StagewiseSQP`<sup>3</sup>.

Firstly, we benchmark the performance of our tailored SQP in the absence of constraints (Section 2.2) against other popular methods that use non-linear rollouts. We show that our tailored SQP implementation performs as well and sometimes better than other methods to solve challenging OCPs. Secondly, we further reinforce this claim by deploying the tailored SQP in high-frequency nonlinear MPC experiments on a torque-controlled manipulator.

Thirdly, we demonstrate the validity of the SQP approach with our tailored ADMM implementation (Section 2.3) in the presence of nonlinear inequality constraints by deploying it on real hardware in MPC. Our experiments show that the solver can satisfy many nonlinear equality and inequality constraints while maintaining real-time solve rates. Finally, we study the importance of sequential implementations of the SQP in the constrained case, and we benchmark the timings of the proposed `OSQP_OCP` solver with other state-of-the-art QP solvers.

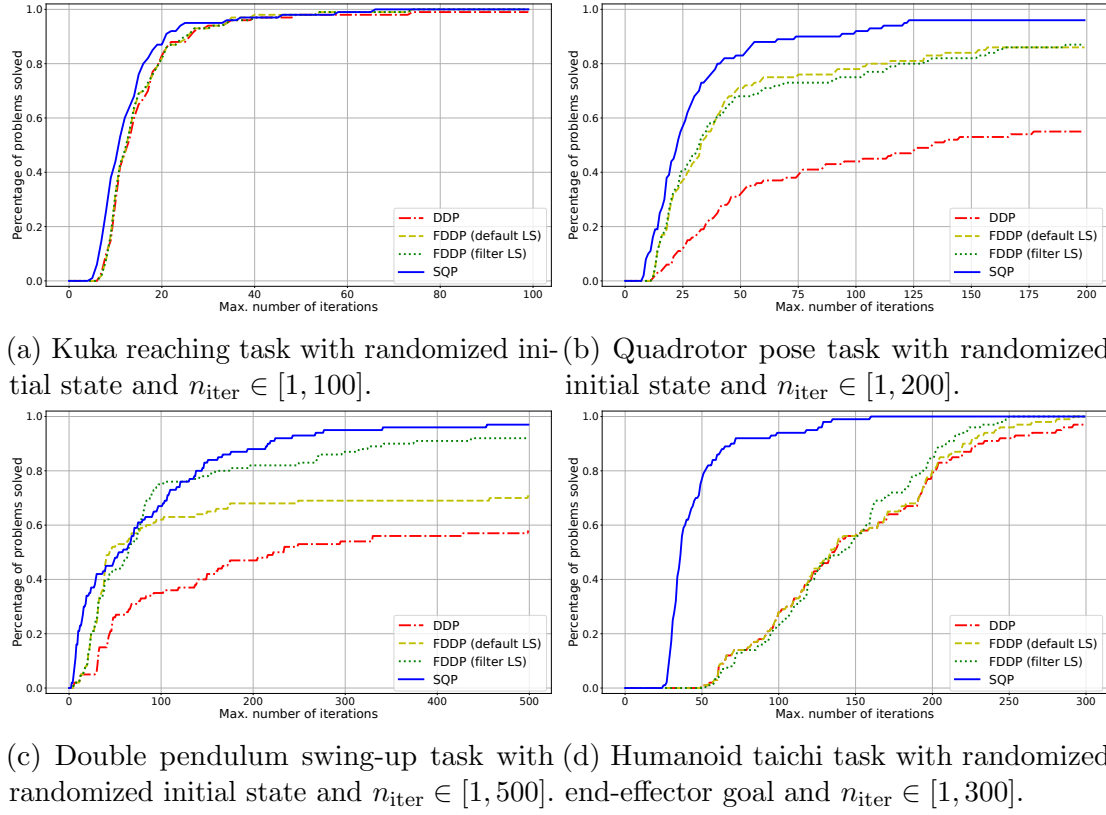


Figure 2.1: Percentage of problem solved as a function of the maximum number of iterations allowed  $n_{\text{iter}}$  on 4 randomized unconstrained OCPs for the 4 solvers: DDP, FDDP with default line-search, FDDP with filter line-search and our SQP. Our SQP exhibits a faster and more robust convergence on difficult problems, such as the humanoid taichi task.

### 2.5.1 Unconstrained case: Benchmarks

In this subsection, we compare the advantages and disadvantages of linear and non-linear rollouts. As emphasized in Section 2.4.1, when only the dynamics constraint is present, the efficient SQP formulation boils down to the GNMS algorithm described in [68]. We propose for the first time to benchmark this algorithm against state-of-the-art optimal control solvers, namely DDP [125] and FDDP [120], on a set of difficult unconstrained OCPs, using a standard line-search procedure and termination criteria drawn from the SQP literature.

#### 2.5.1.1 Benchmark Setup

We present two benchmark setups that compare the convergence and average iteration time of the following 4 solvers, namely DDP, FDDP using the default line-search from [120], FDDP using the proposed filter line-search of Section 2.4.3, and our SQP, on a set of increasingly difficult randomized problems. We summarized the characteristics of these solvers (multiple or single-shooting, type of rollout, and line-search) in Table 2.1. The classical single-shooting DDP and its multiple-

|                   | Multiple shooting | Rollout   | Line-search     |
|-------------------|-------------------|-----------|-----------------|
| DDP               | No                | Nonlinear | Goldstein       |
| FDDP (default LS) | Yes               | Nonlinear | Heuristic [120] |
| FDDP (filter LS)  | Yes               | Nonlinear | Filter [62]     |
| SQP               | Yes               | Linear    | Filter [62]     |

Table 2.1: Solvers characteristics.

shooting variant FDDP (default LS) are the ones implemented in the Crocodyl library [120]. The FDDP (filter LS) was modified to incorporate the same filter

<sup>2</sup>[https://github.com/machines-in-motion/mim\\_solvers](https://github.com/machines-in-motion/mim_solvers)

<sup>3</sup><https://github.com/machines-in-motion/StagewiseSQP>



line-search as our tailored SQP implementation. We use the largest possible filter size for both solvers, i.e., the maximum number of iterations.

To evaluate the performance of these solvers, the following OCPs are used:

- Kuka ( $n_x = 14$ ,  $n_u = 7$ ) : the task is to minimize the Cartesian distance to an end-effector goal (3D reaching task in Cartesian space) under state (joint positions and velocities) and control (joint torques) regularization.
- Quadrotor ( $n_x = 13$ ,  $n_u = 4$ ) : the task is to minimize the distance to a desired pose (in  $\mathbb{SE}(3)$ ) under state and control regularization.
- Double Pendulum ( $n_x = 4$ ,  $n_u = 1$ ) : the task is to minimize the distance to the upward equilibrium position under state and control regularization.
- Humanoid Taichi ( $n_x = 77$ ,  $n_u = 32$ ) : the task is to achieve a desired left foot pose (in  $\mathbb{SE}(3)$ ) while maintaining balance on the right stance foot, starting from a double foot support configuration, under state and control regularization, and log-barrier state limits.

The Quadrotor, Double Pendulum, and Humanoid Taichi examples were copied from the `Crocodyl` library. Each problem is solved for 100 randomized initial configurations (Kuka, Quadrotor, Double Pendulum) or end-effector goal (Humanoid Taichi). The maximum number of iterations allowed  $n_{\text{iter}}$  depends on the problem. For each problem, the solvers are warm-started with the same quasi-static solution. Essentially, a gravity compensation torque is computed for the initial state of the system and provided as an initial guess to the solver. We use the same termination criteria, the KKT residual norm (2.20) with tolerance set to  $\epsilon_{SQP} = 10^{-4}$  on all problems. In order to reflect cases where the maximum number of iterations is hit

without reaching the desired tolerance, we use as a metric the number of solved problems for a given maximum number of iterations. A problem is considered "solved" if it reaches the KKT residual tolerance within the maximum number of iterations allowed.

### 2.5.1.2 Benchmark results

The results obtained for the solver convergence study are shown in Figure 2.1. On the Kuka example, all solvers exhibit a similar behavior. The advantage of multiple-shooting over single-shooting becomes clear in the Quadrotor (Figure 2.1b) and Double Pendulum (Figure 2.1c) examples. These two examples, along with the Humanoid taichi (Figure 2.1d), also highlight clearly the benefit of using a filter line-search in FDDP. Most importantly, the tailored SQP solves more problems in fewer iterations than all the other solvers. In particular, it is clear from these benchmarks that the linear rollout has an advantage over the nonlinear one - we recall that FDDP (filter LS) (green curves) and SQP (blue curves) only differ by their rollouts (nonlinear and linear respectively). In Figure 2.2, we report the average time per iteration for each solver. This shows that each SQP iteration takes about the same or slightly less time compared to the alternative solvers.

For MPC applications, the solver's ability to converge to a desired tolerance within a fixed time is critical because real-time constraints impose a limited computation budget. In that respect, all experiments show that SQP is able to solve more problems in fewer iterations than all other solvers while taking about the same time per iteration. For instance, in difficult problems like the humanoid taichi example, nearly 80% of the problems are solved by the SQP within 50 iterations, while the FDDP (filter LS) requires almost 200 iterations to solve the same amount

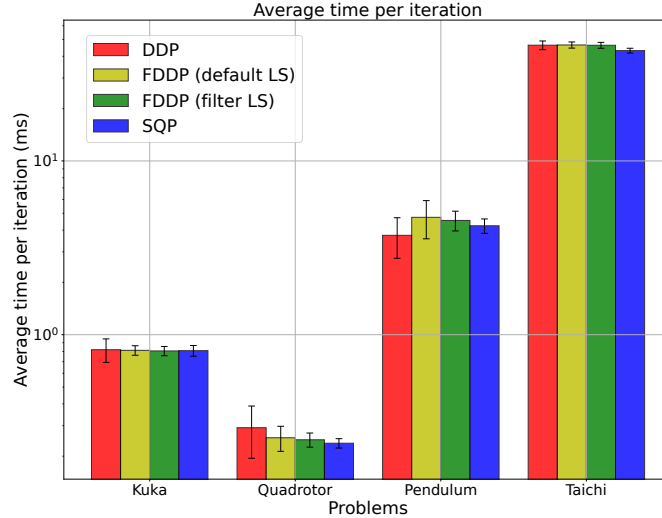


Figure 2.2: Average time per iteration for each solver on the 4 benchmark problems (Kuka, Quadrotor, Pendulum, Taichi)

of problems.

### 2.5.2 Unconstrained Case : MPC experiments

We implemented FDDP (filter LS) and SQP in MPC on the KUKA LBR14 iiwa to execute the task of tracking a circle with the end-effector. The cost function includes state and torque regularization, and an end-effector position tracking term. The robotic setup follows our previous work [102], where more details are available. We used an MPC frequency of 500 Hz, with an OCP discretization of 50 ms, 10 nodes in the horizon, and a maximum number of SQP iterations  $n_{\text{iter}} = 5$ . At each MPC cycle, the solvers are warm-started with the previous trajectory. Although both solvers exhibited equal tracking performance, they differed in their convergence speeds, which corroborates our benchmarks. Indeed, the cumulative costs achieved are similar, but the SQP converges faster to its optimal solution, as shown in Figure 2.3.

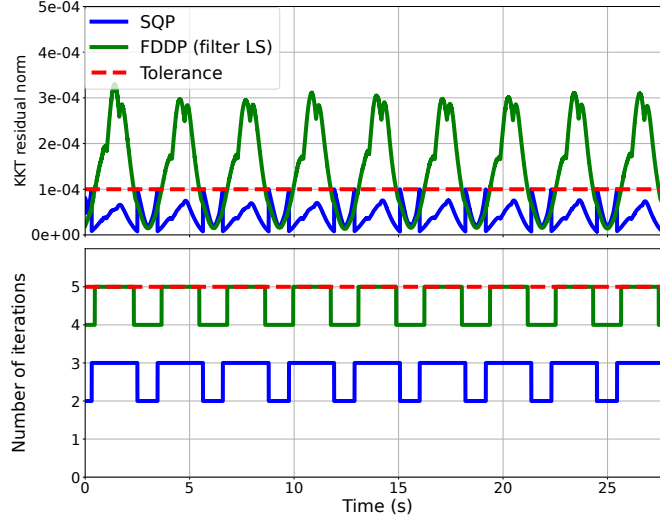


Figure 2.3: KKT residual norm and number of iterations for the circle tracking task. Our SQP solver converges within 3 iterations while FDDP hits the maximum number of iterations ( $n_{\text{iter}} = 5$ ) without reaching the desired tolerance.

### 2.5.3 Constrained case : Robot Experiments

In this subsection, we first show the ability of our tailored SQP solver to solve constrained multi-contact nonlinear OCPs on a simulated quadruped robot Solo12 [75]. Finally, we show that the proposed method can be used in MPC to satisfy arbitrary constraints on a real robot.

#### 2.5.3.1 Quadruped standing task with friction cones

The Solo [75] quadruped is tasked with tracking a desired CoM position while maintaining its contact forces within the friction cone, i.e.,

$$\|F_T\|_2 \leq \mu F_N \quad (2.23)$$

where  $F_T, F_N$  represent the tangential and normal forces, respectively. We use 250 nodes in the OCP, with a discretization of 20 ms. The CoM must track a

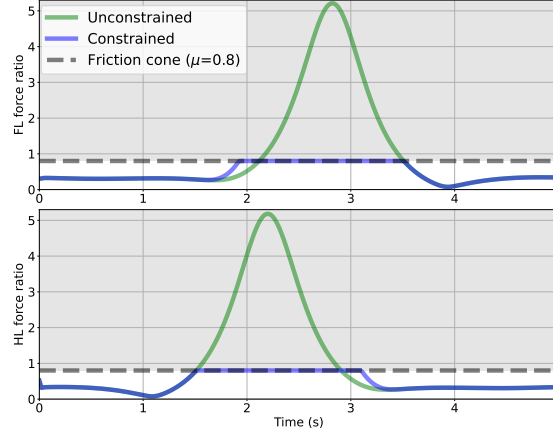


Figure 2.4: Solo center-of-mass tracking task with friction cone constraints. The continuous lines represent the ratio  $\frac{F_T}{F_N}$  at the front left foot, and the gray dashed line represent the friction cone constraint. Observe in the  $F_z$  plots the unconstrained OCP solution (green) crossing the friction cone constraint while the constrained OCP solution (blue) remains within the constraint.

circular trajectory of 13 cm diameter and  $0.2 \text{ rad s}^{-1}$  velocity. The QP absolute and relative tolerances are set to  $10^{-6}$ , and the termination tolerance to  $10^{-4}$ . We use the merit function line-search and the KKT residual termination criteria as previously described. The same OCP was solved with and without constraint (2.23). The solver converged in 34 iterations in the unconstrained case and in 31 iterations in the constrained case. Note that the merit function with default parameter  $\mu_\gamma = \mu_c = 1$  was used in this illustrative example, which seems to benefit the solver’s convergence in the constrained case over the unconstrained case. The accompanying video shows the corresponding motion, and snapshots are provided in Figure 2.5. Figure 2.4 shows the ratio of tangential over normal forces of the unconstrained and constrained solutions. One can see that the task cannot be achieved without slipping when no constraint is enforced. Hence our tailored SQP implementation can enforce nonlinear inequality constraints (Lorentz cones) while also keeping the number of iterations low.

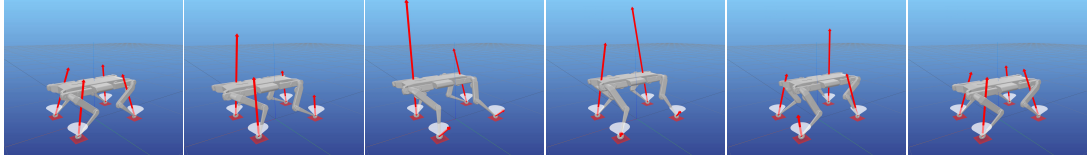


Figure 2.5: Snapshots of standing motion. The red arrows represent the contact forces and the white cones are the friction constraint ( $\mu = 0.8$ ). In the 3<sup>rd</sup> and 4<sup>th</sup> frames, one can see the tangential forces lying on the boundary on the friction cone.

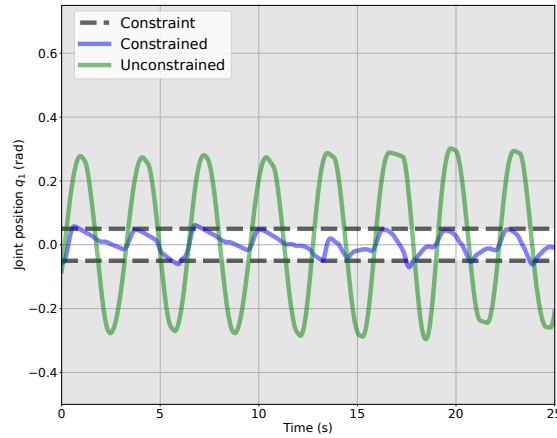


Figure 2.6: Joint position  $q_1$  for the circle task in the constrained (blue) and unconstrained case (green). The gray-shaded area represents the infeasible region.

### 2.5.3.2 MPC experiments setup

We deployed our tailored SQP implementation in MPC on the KUKA robot to achieve various constrained tasks. In all the experiments, the MPC runs at 100 Hz, the OCP discretization is 50 ms and the horizon has 10 nodes. We allow a maximum of 4 SQP iterations and 50 QP iterations (i.e. 50 iterations of Algorithm 2 where  $\rho$  is updated every 25 iterations). Relative and absolute tolerances for the QP are set to  $10^{-5}$  and  $10^{-4}$ . The size of the filter for the line-search is set to 1. The  $\rho$  penalty parameter is not reset throughout the iterations, and the primal solution is warm-started with the previous trajectory (no warm-start on the dual variables  $y$  and  $z$  which are reset to 0).

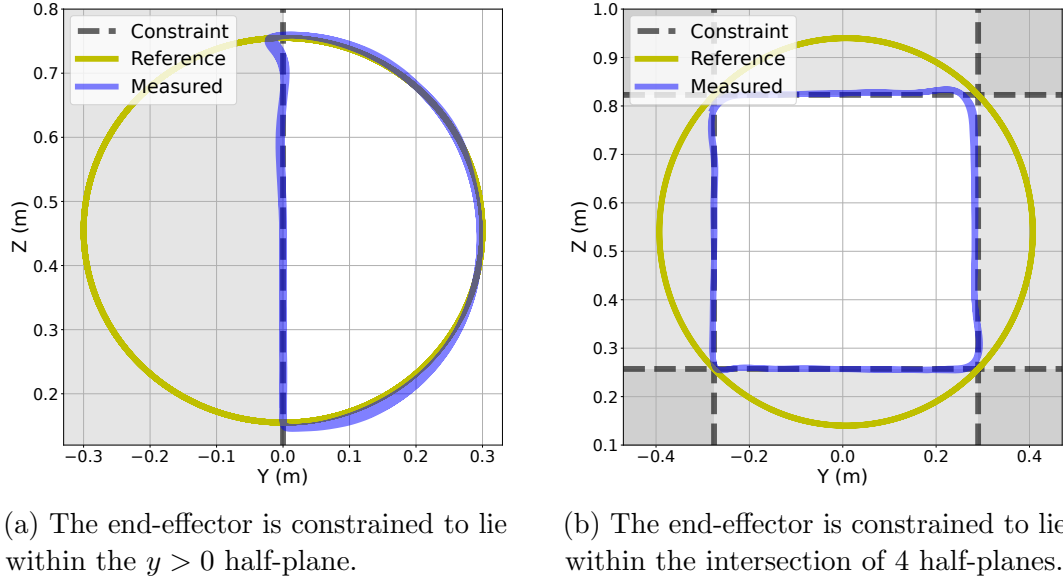


Figure 2.7: End-effector position trajectories in the  $(y, z)$ -plane during the circle tracking task, subject to nonlinear constraints in Cartesian space. The gray-shaded areas represent the infeasible regions.

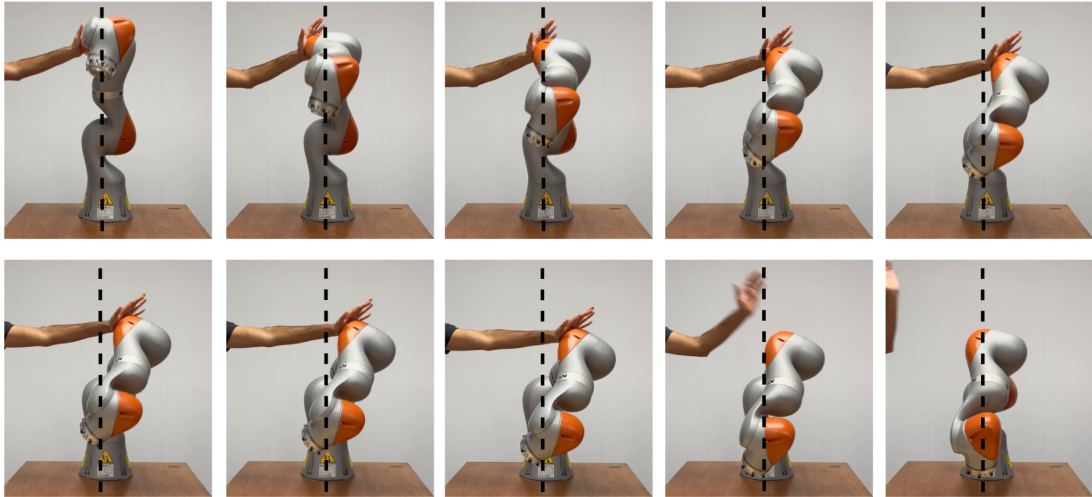


Figure 2.8: The SQP solver keeps the end-effector on a straight line constraint (black dotted line) even during unexpected perturbations. The SQP solver is able to rapidly find robot configurations needed to satisfy the constraints while tracking the desired goal.

Our experiments focus on constrained circle tracking tasks: the robot must track a circle with its end-effector while satisfying various constraints in the joint space or in the end-effector space. The objective includes state and control regularization terms and a term to follow a circle with the end-effector (3D task). We observed that reducing the MPC frequency to 100 Hz (vs 500 Hz in the unconstrained experiments) was beneficial since it allowed us to maintain a reasonable number of QP (50) and SQP (4) iterations and thereby a good convergence.

### 2.5.3.3 MPC experiments results

In the first experiment, the robot must track the circle while keeping the position of the first joint within  $[-0.05 \text{ rad}, +0.05 \text{ rad}]$ . The position of the constrained joint is shown in Figure 2.6. Without the constraint, the circle tracking average absolute error is lower (3.3 cm) than the constrained case (8.5 cm) but the constraint is largely violated. Increasing the end-effector tracking cost weight in the constrained case leads to an improved tracking performance but a more aggressive behavior on the robot due to an increased constraint saturation.

In the second experiment, an end-effector (the center of the last link of the robot) constraint was imposed during the circle task. Figures 2.7a, 2.7b show the Cartesian space trajectories for circle tasks in which the end-effector is constrained to lie within specific half-spaces. We observed that increasing the velocity of the reference circle had the effect of smoothing out the edges of the square. This behavior can be explained by the prediction horizon which enables the controller to anticipate constraints: remaining some distance away from the constraint boundary is the optimal way to minimize the objective while preventing constraint violation in the future. This confirms the intuition that the horizon is crucial in constrained



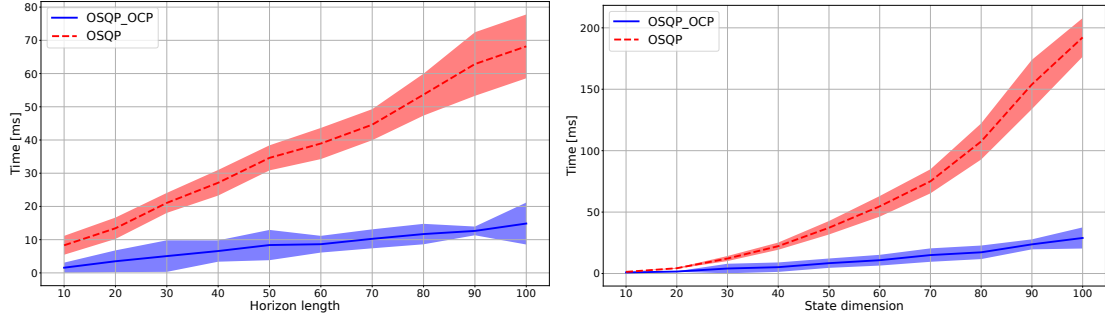
dynamic tasks.

In the third experiment, the end-effector was constrained to remain on a vertical line while tracking a circle (i.e. equality constraint  $Y = 0$  in Figure (2.7)). In addition to this, the robot was perturbed at arbitrary locations and times during the experiments. The resulting behavior on the robot is shown in Figure 2.8. As can be seen, the MPC solver is able to rapidly determine a robot configuration that resists the applied force while keeping the end-effector on the constraint and tracking the desired trajectory in the vertical direction. This result shows the ability of the solver to satisfy tight nonlinear constraints.

All experiments are shown in the accompanying video. We also included a fourth experiment in the video that forces the end-effector to remain on the horizontal plane, even under external disturbances from a human operator. In that case, the robot is again able to adapt its configuration automatically in order to satisfy the constraint. Through these experiments, we confirm that arbitrary constraints can be enforced at real-time rates, without having to re-define the task (i.e. no re-tuning the cost function weights).

#### 2.5.4 Constrained case : Benchmarks

We now discuss the performance of the proposed tailored ADMM. First, we demonstrate the importance of sequential implementations as opposed to general sparse algebra routines by benchmarking the iteration times of the proposed OSQP\_OCP solver against OSQP [167] as the problem size varies. Second, we benchmark the convergence timings of the proposed OSQP\_OCP solver against OSQP and HPIPM\_OCP [65], a state-of-the-art solver tailored for OCPs.



(a) Compute time according to horizon length for a 50 dimensional state. (b) Compute time according to state dimension for a horizon of length 50.

Figure 2.9: Comparison between the compute time of 25 iterations OSQP\_OCP and OSQP.

#### 2.5.4.1 Sequential implementation v.s. Sparse Algebra

We compare our proposed QP solver to its closest alternative - OSQP [167], since OSQP uses ADMM with sparse linear algebra while our solver uses a tailored ADMM that exploits stage-wise sparsity. We set up a benchmark where 100 constrained LQR problems (with random initial state) of increasing horizon length (Figure 2.9a) and state dimension (Figure 2.9b) are solved. For each solver, we measure the time needed to take 25 QP iterations. It is important to note that both these solvers need not converge within the 25 iterations and we only focus on the wall-clock time (convergence time is evaluated in the next subsection 2.5.4.2). As shown in Figure 2.9, the proposed OSQP\_OCP solver displays a linear increase in wall clock time as the problem size increases, unlike OSQP which grows non-linearly.

#### 2.5.4.2 QP Solvers Timings - Setup

In this benchmark, we evaluate and compare the convergence of the following QP solvers: OSQP\_OCP, HPIPM\_OCP, OSQP. We summarized the characteristics of these solvers in Table 2.2.

|           | Sequential | Sparse Algebra | Type           |
|-----------|------------|----------------|----------------|
| ADMM OCP  | Yes        | No             | ADMM           |
| HPIPM OCP | Yes        | No             | Interior Point |
| OSQP      | No         | Yes            | ADMM           |

Table 2.2: QP solvers characteristics.

To evaluate the performance of these solvers, the following OCPs are used:

- Kuka : same setup as in the unconstrained case but with a hard constraint on the terminal end-effector position.
- Solo12 : same setup as in the constrained experiment presented in Section 2.5.3.1.
- Humanoid Taichi : same setup as in the unconstrained case but with force and torque constraints on the right foot.

Each problem is solved for 100 randomized settings. For the Kuka, we sample the initial configurations, for the humanoid Taichi, the end-effector goal, for Solo12, the friction coefficient. The QP tolerances are set to  $\epsilon_{abs} = 10^{-4}$  and  $\epsilon_{rel} = 0$ , and we let the QP solver fully converge to that tolerance. The QP problem corresponds to the first SQP iteration.

#### 2.5.4.3 QP Solvers Timings - Results

|        | OSQP_OCP                       |             | HPIPM_OCP                         |               | OSQP            |                |
|--------|--------------------------------|-------------|-----------------------------------|---------------|-----------------|----------------|
|        | Time                           | Iter        | Time                              | Iter          | Time            | Iter           |
| Kuka   | $0.5 \pm 0.6$                  | $76 \pm 97$ | <b><math>0.16 \pm 0.03</math></b> | $4.1 \pm 0.8$ | $0.98 \pm 0.53$ | $60 \pm 15$    |
| Solo   | $2.6 \pm 0.3$                  | $50 \pm 0$  | <b><math>1.8 \pm 0.57</math></b>  | $5.7 \pm 1.5$ | $3400 \pm 61$   | $2200 \pm 410$ |
| Taichi | <b><math>35 \pm 2.3</math></b> | $180 \pm 0$ | $57 \pm 19$                       | $37 \pm 12$   | $656 \pm 98$    | $990 \pm 170$  |

Table 2.3: Mean solving time in milliseconds and number of QP iterations for different problems

Table 2.3 shows the QP solving times and number of iterations for the 3 problems and all solvers. The results once again show that solvers that exploit stage-wise sparsity (HPIPM\_OCP and our OSQP\_OCP solver) converge to a solution quicker. Further, OSQP\_OCP takes fewer iterations to converge for complicated OCPs like Solo and Taichi, as compared to OSQP. We conjecture that this difference is due to the difference in formulation of the OSQP\_OCP, i.e., we decompose the original QP into two convex subproblems in ADMM. Indeed, this leads to directly handling the dynamic feasibility constraints with a linear rollout in the QP (equation 2.13a) and satisfies linear dynamic feasibility at each iteration of ADMM. On the contrary, OSQP handles the dynamic feasibility as a general equality constraint and ensures its feasibility at convergence (which usually takes several iterations) [167].

The two stage-wise exploiting solvers, HPIPM\_OCP and OSQP\_OCP perform similarly. HPIPM\_OCP has better convergence on problems with smaller matrices (Kuka and Solo examples), probably because it uses the BLASFEO [66] backend for dense matrix operations, which is shown to outperform Eigen routines for smaller matrices. When the matrices become large, like in the Taichi example, the advantage of BLASFEO diminishes and our proposed QP solver converges quicker. Overall, these results suggest that OSQP\_OCP has similar performance as HPIPM\_OCP and the minor difference in performances stem from the dense matrix algebra routines. Further, the author of HPIPM also notes that with the same matrix backend, ADMM based methods can be faster than interior point methods [64], especially for large problems when a reasonable solution is desired in a few iterations. Note that the performance of matrix algebra routines turns out to be critical to implement efficient solvers.

Overall, we would like to highlight that obtaining real-time closed-loop MPC

is possible with both HPIPM\_OCP and OSQP\_OCP, and our results suggest that exploiting stage-wise sparsity is a major factor of efficiency. The OSQP\_OCP solver has two advantages: it easily handles infeasibility just like OSQP and it can return a reasonable solution in fewer iterations thanks to the convergence properties of ADMM [27].

## 2.6 Discussion

The benchmark introduced in the unconstrained case questions what allowed this improvement. We argue that FDDP appears to be a hybrid method laying between single and multiple shooting. This idea comes both from our results on the Humanoid taichi robot, where FDDP behaves like DDP and from the theory, as it is known that once the gaps close in FDDP, they cannot re-open again. Consequently, we believe that the improvement observed in the benchmark comes from the multiple shooting formulation.

Our work follows the line of work started in the 1990s [51, 52, 146, 150, 166, 187, 189] showing that standard optimization tools can be implemented specifically for OCPs by exploiting time-induced sparsity. We simply use the same tenets with SQP and ADMM as they are well-established in the optimization community. Additionally, ADMM has properties (easy to warm start and quick convergence to few iterations) that benefit MPC [27]. However, we would like to insist on the fact that the same principles could be applied to other optimization techniques. Future work would be especially interesting to modify such state-of-the-art QP solvers to exploit stagewise sparsity given that they outperformed OSQP, in general, [10, 157].

Finally, a direct by-product of the tailored implementation is the Riccati-like

gains that can also enforce the additional inequality constraints. This is a natural consequence of using Riccati recursions to efficiently solve the sparse linear matrix equation that appears in the QP solver. So far, we have not used these Riccati gains on a real robot in MPC, however [163] showed encouraging results in simulation. A study on their effect on control performance and constraint satisfaction remains to be done on a robot.

During deployment, we usually early terminate the ADMM-based QP in the interest of higher re-planning frequency (100 Hz). A lower-quality solution seems to be sufficient to ensure higher reactivity and compliance on the robot. One key advantage of ADMM in this context is that the sub-QP solver can reach a reasonable solution in a few iterations and also guarantee the availability of some solution even when the constraints may be infeasible because it is based on the ADMM algorithm [27]. Both of these are desirable in non-linear MPC where obtaining a solution is essential during deployment.

## 2.7 Conclusion

A central message of this Chapter is that the existing nonlinear optimization literature already provides sufficient tools to solve OCPs with and without constraints in real time and thereby achieve state-of-the-art nonlinear MPC on real robots. We substantiated this message through a tailored, sparsity-exploiting SQP formulation that provided a unifying framework clarifying the connections between existing solvers from the robotics literature and a practical approach to achieving state-of-the-art MPC. We demonstrated through various benchmarks and hardware experiments that such a tailored SQP implementation outperformed state-of-the-art

solvers based on DDP on challenging unconstrained problems. We then showed that it can efficiently solve arbitrary constrained nonlinear OCPs for MPC applications. In particular, we could enforce many nonlinear constraints on a real robot in MPC.

In this Chapter, we presented a generic framework to solve efficiently constrained OCP. However, in this form, MPC can exhibit local behavior. One reason is that gradient-based nonlinear optimization is subject to local minima. Another explanation is that the use of a short horizon can generate local behaviors. For instance, in an obstacle avoidance task, the manipulator could get stuck due to a too short horizon – staying in place could yield a lower cost than circumventing an obstacle. More specifically, minimizing the joint torque regularization cost instead of the distance to the target could be more beneficial to the overall cost. In Chapter 3, we will show how to use function approximation to avoid such local behaviors.

## Chapter 3

# Infinite-Horizon Value Function Approximation for Model Predictive Control

While MPC possesses strong theoretical guarantees, the real-time requirement has limited the use of hard constraints and large preview horizons, which are necessary to ensure safety and stability [76, 126]. In practice, practitioners have to carefully design cost functions that can imitate an infinite horizon formulation, which is tedious and often results in local minima. In this Chapter <sup>1</sup>, we study how to approximate the infinite horizon value function of constrained optimal control problems with neural networks. We then demonstrate how to use this value function to endow MPC with global behaviors on an obstacle avoidance task with an industrial manipulator.

The theoretical benefits of an infinite horizon formulation have been extensively

---

<sup>1</sup>This chapter is adapted from the following paper: **A. Jordana** et al. "Infinite-Horizon Value Function Approximation for Model Predictive Control" Submitted to RAL, 2024



studied [36, 76, 88, 127]. Unfortunately, the general case is intractable. Hence, efforts have concentrated on the constrained linear quadratic regulator [19, 74, 165]. In the general case, RL provides a way to approximate the solution [22, 168]. In the discrete action setting, Deep Q-learning is a popular tool [131]. In the continuous case, actor-critics such as DDPG [116] or SAC [77] allow learning simultaneously a policy and a value function. However, despite recent progress [35, 194], incorporating hard constraints in those formulations remains challenging. Also, RL algorithms [168] typically use a discount factor. However, the global stability guarantee of infinite horizon MPC was established in the non-discounted setting [76, 126]. Consequently, we study the non-discounted setting with hard constraints which, to the best of our knowledge, remains understudied in the RL community.

The idea of combining MPC and function approximation is not novel and has fostered a lot of research in the robotics community. The seminal work of Atkeson [7] explored how to use local trajectory optimization together with a global value function. Since then, an extensive amount of work has shown the benefits of using TO with learning to either speed up the training of value functions and policies or improve the controller’s performance at test time [3, 70, 80, 83, 105, 109, 112, 117, 132, 138, 147, 191, 195]. A limitation of these works is their inability to consider hard nonlinear constraints. In practice, constraints have to be enforced softly using penalty terms in the cost function. However, this approach requires tedious weight tuning and can hardly guarantee safety. In this Chapter, we show how the known advantages of combining MPC and RL can be obtained while enforcing hard constraints. In addition, we demonstrate how this combination can provide a controller maintaining safety even outside the training distribution.

More recently, [175, 182] demonstrated the benefits of using online constrained optimization with an approximate value as a terminal model in the context of locomotion. In these works, the authors propose to learn the value with a local TO solver using a long horizon. Consequently, at test time, the controller remains local. In contrast, we use value iteration combined with local TO to approximate the infinite horizon value function, and we demonstrate the ability of the method to avoid local minima.

In this Chapter, we propose to approximate the infinite horizon value function of constrained OCP and use it as a terminal cost function of a Model Predictive Controller. The contributions of this Chapter are threefold:

- First, we demonstrate how a local gradient-based solver allows the use of value iteration to approximate the optimal value function of a constrained OCP with an infinite horizon.
- Second, we provide an experimental study showing how the use of trajectory optimization can compensate for the inaccuracies of the value function approximation.
- Third, we demonstrate the benefits of combining MPC with value function approximation on a reaching task with obstacle avoidance on an industrial manipulator. More precisely, we show how the use of the value function allows avoiding local minima to which MPC is subject and demonstrate how the method remains safe by ensuring hard constraints outside the training distribution of the value function. To the best of our knowledge, this is the first demonstration of MPC using a learned global value function with hard constraints deployed on a robot at real-time rates.

### 3.1 Infinite horizon MPC

Here, we are interested in the infinite-horizon constrained optimal control problem:

$$V(x) = \lim_{T \rightarrow \infty} \min_{u_0, u_1, \dots, u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) \quad (3.1a)$$

$$\text{s.t. } x_0 = x \quad (3.1b)$$

$$x_{k+1} = f(x_k, u_k) \quad (3.1c)$$

$$c(x_k, u_k) \geq 0 \quad (3.1d)$$

Here  $V$  is the infinite horizon value function. The state  $x$  belongs to  $\mathbb{R}^{n_x}$  and the control  $u$  to  $\mathbb{R}^{n_u}$ . The dynamic function  $f$  maps  $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$  to  $\mathbb{R}^{n_x}$ . The constraint function  $c$  maps  $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$  to  $\mathbb{R}^{n_c}$ . The cost function  $\ell$  maps  $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$  to  $\mathbb{R}^+$ . Similarly to [24], to ensure the existence of states yielding a finite value function, we consider that the set of stationary points yielding a zero cost,

$$\mathcal{G} = \{x \mid \exists u \text{ s.t. } \ell(x, u) = 0, x = f(x, u), c(x, u) \geq 0\} \quad (3.2)$$

is not empty. Computing the value function or its associated optimal policy is intractable in general. However, RL provides tools to find an approximation.

Let's denote  $\Omega$ , the space on which the value function is well-defined and finite. For any state  $x \in \Omega$ , the value function satisfies the Bellman equation:

$$\mathcal{B}(V) = V, \quad (3.3)$$

where  $\mathcal{B}$  is the Bellman operator, defined by:

$$\mathcal{B}(V)(x) = \min_u \ell(x, u) + V(f(x, u)) \quad (3.4a)$$

$$\text{s.t. } c(x, u) \geq 0 \quad (3.4b)$$

$$f(x, u) \in \Omega \quad (3.4c)$$

Here, Eq (3.4c) ensures recursive feasibility. In this Chapter, we focus on problems where the set  $\Omega$  can be expressed analytically, which encompasses many problems. For instance, if the constraint is of the form  $c(u)$ , then  $\Omega = \mathbb{R}^{n_x}$ . If the constraint is of the form  $c(x)$ , then  $\Omega = \{x \in \mathbb{R}^{n_x} | c(x) \geq 0\}$ . In the more general case where the constraint is a function of both the state and the control, the set  $\Omega$  is not tractable, and we would have to rely on approximation techniques [11, 50]. However, this is beyond the scope of the Chapter.

In the non-discounted setting, the Bellman equation has multiple solutions. Indeed, if  $V$  is a solution, then  $V + \beta$  (where  $\beta$  is a constant) is also a solution. Therefore, we additionally require that the value function should be zero on stationary points, yielding zero cost. More precisely,

$$\forall x \in \mathcal{G}, \quad V(x) = 0. \quad (3.5)$$

The idea of value iteration is to find a stationary point to the Bellman equation by iteratively applying the Bellman operator:

$$V_{k+1} = \mathcal{B}(V_k). \quad (3.6)$$

Under the assumption that the set  $G$  is not empty, it can be shown that  $V_k$  converges

to  $V$  pointwise [24, 82].

## 3.2 Approximate infinite horizon MPC via function approximation

### 3.2.1 $T$ -step optimal lookahead problem

In this Chapter, we propose to combine online and offline decision-making by using the  $T$ -step optimal lookahead problem [21] online. The idea is to perform MPC by solving an OCP at each time step, using an approximate value function as a terminal cost function. More precisely, given a state  $x$ , we aim to find the optimal action by solving:

$$\min_{u_0, u_1, \dots, u_{T-1}} \sum_{k=0}^{T-1} \ell_k(x_k, u_k) + V_\theta(x_T) \quad (3.7a)$$

$$\text{s.t. } x_0 = x \quad (3.7b)$$

$$x_{k+1} = f(x_k, u_k) \quad (3.7c)$$

$$c(x_k, u_k) \geq 0 \quad (3.7d)$$

$$x_T \in \Omega \quad (3.7e)$$

Here,  $V_\theta$  is the value function approximation. Here, we consider  $V_\theta$  to be a neural network parameterized by weights  $\theta$ . At each time step, the first optimal control input,  $u_0^*$ , is applied to the system, and the other controls are disregarded. In the end, the policy,  $\pi^*(x) = u_0^*$ , depends on the horizon  $T$  and the approximated value function  $V_\theta$ . In continuous action space, RL algorithms rely on a function approximation of the policy [168]. In contrast, we solve Problem (3.7) online.

The benefit of this approach is that the model used during the first  $T$  steps of the optimization can reduce the inaccuracies of the value function and guarantee stability [23, 107]. Furthermore, this ensures hard constraint satisfaction despite the use of the approximated value function.

### 3.2.2 Value function approximation

In this section, we show how to use value iteration to approximate the value function of an infinite-horizon constrained OCP by neural networks. To minimize the Bellman equation (3.4), we propose to use a local gradient-based solver. To reduce the required number of value iterations (i.e. iterations of the Bellman operator), we can apply the minimization over an arbitrary horizon  $T$ . More precisely, we can use trajectory optimization to directly solve:

$$V_{k+1} = \mathcal{B}^{[T]}(V_k) \quad (3.8)$$

where  $\mathcal{B}^{[T]}$  is the Bellman operator over a horizon  $T$ .

$$\mathcal{B}^{[T]}(V) = \underbrace{\mathcal{B} \circ \dots \circ \mathcal{B}}_{T \text{ times}}(V) \quad (3.9)$$

Indeed, it can be shown that iterating  $T$  times the Bellman operator is equivalent to solving an OCP of horizon  $T$ , precisely as in Problem (3.7). Intuitively, this should allow us to perform  $T$  times fewer value function iterations.

We propose to fit the approximated value function at each Bellman iteration in a supervised way. Note that this approach can, therefore, be considered as an instance of Fitted Value Iteration (FVI) [22, 136] adapted to the constrained

and deterministic setting. More precisely, at iteration  $k$ , given a value function  $V_k$ , we sample  $n$  states,  $\{x_j\}_{1 \leq j \leq n}$  and solve the  $n$  associated OCP with the corresponding initial condition and with  $V_k$  as a terminal cost. Then, we train in a supervised way such that  $V_\theta$  maps  $x_j$  to  $\mathcal{B}^{[T]}(V_k)(x_j)$ . Furthermore, to ensure that  $\forall x^s \in \mathcal{G}, V_\theta(x^s) = 0$ , we sample  $m$  stationary points and  $\{x_j^s\}_{1 \leq j \leq m}$  and minimize the value function at those points with the Mean Squared Error (MSE). In the end, we minimize the following loss:

$$\sum_{j=1}^n \|V_\theta(x_j) - \mathcal{B}^{[T]}(V_k)(x_j)\|_2^2 + \alpha \sum_{j=1}^m \|V_\theta(x_j^s)\|_2^2, \quad (3.10)$$

where  $\alpha$  is a penalty parameter. To create the targets, we solve Problem (3.7). To do so, we use the stagewise Sequential Quadratic Programming (SQP) implementation introduced in [95]. This solver can handle hard constraints and exploits the time sparsity of the problem by using Riccati recursions in order to guarantee a linear complexity with the time horizon and quadratic convergence. Algorithm 3 summarizes the method.

Minimizing (3.7) with a gradient-based solver requires the derivatives of the neural network. More specifically, the SQP approach requires the gradient and Hessian of the terminal cost function. To circumvent deriving twice a neural network, we use the Gauss-Newton approximation and define the value function as the squared L2 norm of a residual. More precisely:

$$V_\theta(x) = \frac{1}{2} \|f_\theta(x)\|_2^2 \quad (3.11)$$

where  $f_\theta$  is a neural network whose output lives in  $\mathbb{R}^d$  where  $d$  is a hyperparameter.

---

**Algorithm 3:** Value Iteration

---

**Input:** dynamics  $f$ , cost  $\ell$ , constraint  $c$ , horizon  $T$ , network parameters  $\theta$

```

1 Initialize  $V_1$ 
  /* Main value iteration loop */
2 for  $k \leftarrow 1$  to  $N$  do
  /* Generate data */
3   for  $j \leftarrow 1$  to  $M$  do
4     Sample  $x_j$ 
5     Compute  $\mathcal{B}^{[T]}(V_k)(x_j)$  by optimizing (3.7)
6   Create dataset:  $\mathcal{D} = \{(x_j, \mathcal{B}^{[T]}(V_k)(x_j))_j\}$ 
  /* Fit value function with SGD */
7   for  $j \leftarrow 1$  to  $P$  do
8     Sample batch from  $\mathcal{D}$ 
9     Apply gradient descent with loss (3.10);
10   $V_{k+1} \leftarrow V_\theta$ 
Output:  $V_{N+1}$ 

```

---

Consequently, using a Gauss-Newton approximation, we have:

$$\partial_x V_\theta(x) = \partial_x f_\theta(x)^T f_\theta(x) \quad (3.12a)$$

$$\partial_{xx}^2 V_\theta(x) \approx \partial_x f_\theta(x)^T \partial_x f_\theta(x) \quad (3.12b)$$

where  $\partial f_\theta(x) \in \mathbb{R}^{d \times n_x}$  is the Jacobian matrix of  $f_\theta$  evaluated in  $x$ . Note that this formulation also has the advantage of encoding the positivity of the value function.

For numerical optimization to be performed efficiently, it is crucial to obtain an accurate Jacobian of the value function. While [147] investigated the use of Sobolev learning [43], we found that using *tanh* activation function and an appropriate L2 regularization yielded accurate Jacobians of the network and was sufficient to ensure convergence of the SQP in few iterations. Furthermore, we find that initializing  $V_1$  to the zero function improves the training. To do so, at the first value iteration, we solve the OCP without a terminal cost function.



In order to generate trajectories that are similar to the one encountered at deployment, we rollout trajectories to generate more data. More specifically, after solving Problem (3.7),  $x_1$  is added to the dataset by solving Problem (3.7) using  $x_1$  as an initial condition, and we iterate until either the goal or a maximum number of iterations is reached. In other words, Problem (3.7) is used as an MPC controller to collect additional data. We find this especially relevant on complex examples where the sampling distribution does not cover well the large state space.

### 3.3 Experiments

In this section, we present three problems of increasing complexity. The first two examples are used to illustrate the ability of the method to approximate the infinite horizon problem with a finite horizon and an approximate terminal value function. Lastly, we study a reaching task on an industrial manipulator with an obstacle to demonstrate the scalability of the method. In order to handle a moving scene, the value function is conditioned to the goal of the reaching task and the obstacle pose. First, we provide an analysis of the impact of the horizon both at train and test time. Then, we present results from real experiments.

For all experiments, we use the SQP implementation introduced in [95]. During the training, we solve each OCP in parallel on the CPU. Note that this is crucial to obtain reasonable training times.

#### 3.3.1 Toy Problem 1: Constrained Simple Pendulum

The first test problem we consider is the swing-up of a simple pendulum with torque limits. We illustrate how a finite horizon with an approximate value function

as a terminal cost allows us to approximate the infinite horizon MPC. The state is  $x = \begin{bmatrix} \theta, \dot{\theta} \end{bmatrix}^T$  where  $\theta$  denotes the orientation of the pendulum. The dynamics are defined by applying Euler integration to the following law of motion:

$$\ddot{\theta} = -\frac{g}{L} \sin(\theta) \quad (3.13)$$

The goal is to bring the pendulum to the upward position, which is incentivized with the following cost:

$$\ell(x, u) = \cos(\theta) + 1 + 0.01\dot{\theta}^2 + 0.001u^2 \quad (3.14)$$

Lastly, the control input is constrained to be within  $[-2, 2]$  which makes it impossible to swing up the pendulum without several back and forth.

We sample  $\theta$  uniformly in  $[-\pi, \pi]$  and  $\dot{\theta}$  uniformly in  $[-6, 6]$ . The number of sample points at each value iteration,  $n$ , is set to 500, and the number of goals sampled  $m$  is set to 1 as there is only one state in  $\mathcal{G}$ . We consider a horizon of length  $T = 10$  and perform 1000 value iterations. At each iteration, we perform 80 SGD steps using Adam with default parameters and a weight decay of  $10^{-4}$ . Lastly, we set  $\alpha = 1$ . The network is a three-layer MLP with 64 neurons and an output size of  $d = 64$ . In the end, the training lasts 2 minutes and 40 seconds. Figure 3.1 shows the approximated value function. As expected, the value function outputs its highest value when the pendulum points downwards without velocity, i.e.,  $x = \begin{bmatrix} 0, 0 \end{bmatrix}^T$ .

Figure 3.2 shows the behavior of the MPC controller for different horizon lengths using the approximated value function as a terminal cost. The quality of the control increases as the horizon length increases. This can also be seen by looking at the

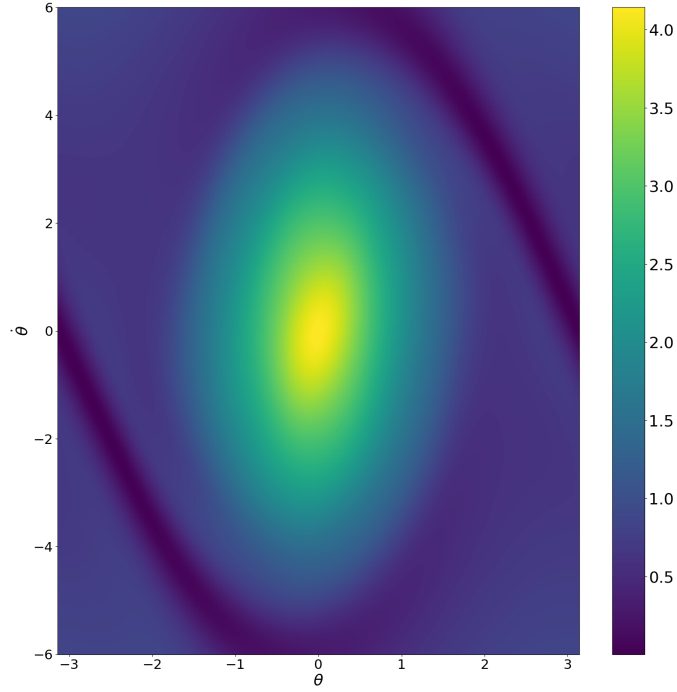


Figure 3.1: Approximated infinite horizon value function

final cumulative cost. The longer the horizon, the more optimal the controller is. For  $T = 1$ , the cost is 4.57, for  $T = 10$ , the cost is 4.50 and for  $T = 20$ , the cost is 4.41. This illustrates how solving online the OCP introduced in Equation (3.7) compensates for the approximation error of the value function.

### 3.3.2 Toy Problem 2: Constrained point

In this second toy example, we illustrate the ability of the method to avoid local minima to which is prone MPC. We consider a 2-dimensional point that has to move around an obstacle to reach a target. The state is denoted by  $x = \begin{bmatrix} x_1, x_2 \end{bmatrix}^T$ , and the control  $u$  denotes the velocities of  $x_1$  and  $x_2$ . The dynamics are defined in

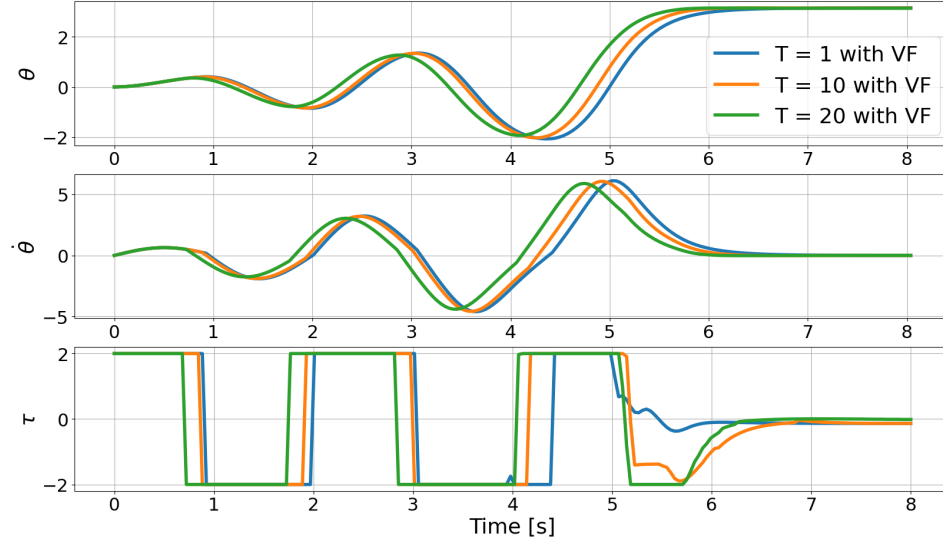


Figure 3.2: Rollout of MPC controllers with different horizon lengths using the learned value function as a terminal cost.

the following way:

$$x_{t+1} = x_t + \Delta t u_t \quad (3.15)$$

where  $\Delta t$  is set to 0.02. The cost function is defined by:

$$\ell(x, u) = \|x - x^*\|_2^2 + 0.1 \|u\|_2^2 \quad (3.16)$$

The constraints are defined by the distance between the point and two capsules that define the obstacle, as illustrated in Figure 3.3. In this experiment, we sample  $x$  uniformly in  $[-1, 1] \times [-1, 1]$  and reject states inside the obstacles. The number of point samples at each value iteration,  $n$ , is set to 2500 and we augment the dataset with the last state of each trajectory. The number of goals sampled  $m$  is set to 1 as there is only one state in  $\mathcal{G}$ . We consider a horizon of 10 and perform 100 value iteration using  $\alpha = 1$ . Lastly, at each iteration, we perform 2000 SGD step using

Adam with default parameters and a weight decay of  $10^{-4}$ . The network is a three-layer MLP with 32 neurons and an output of size  $d = 32$ . In the end, the training lasts 12 minutes. The increase of time compared to the previous experiment is due to the implementation of the model's dynamics in Python. Figure 3.3 shows how using the learned value function as a terminal cost allows bypassing the obstacle in order to reach the goal. Without this learned value function, the controller would get stuck in the inner corner of the obstacle.

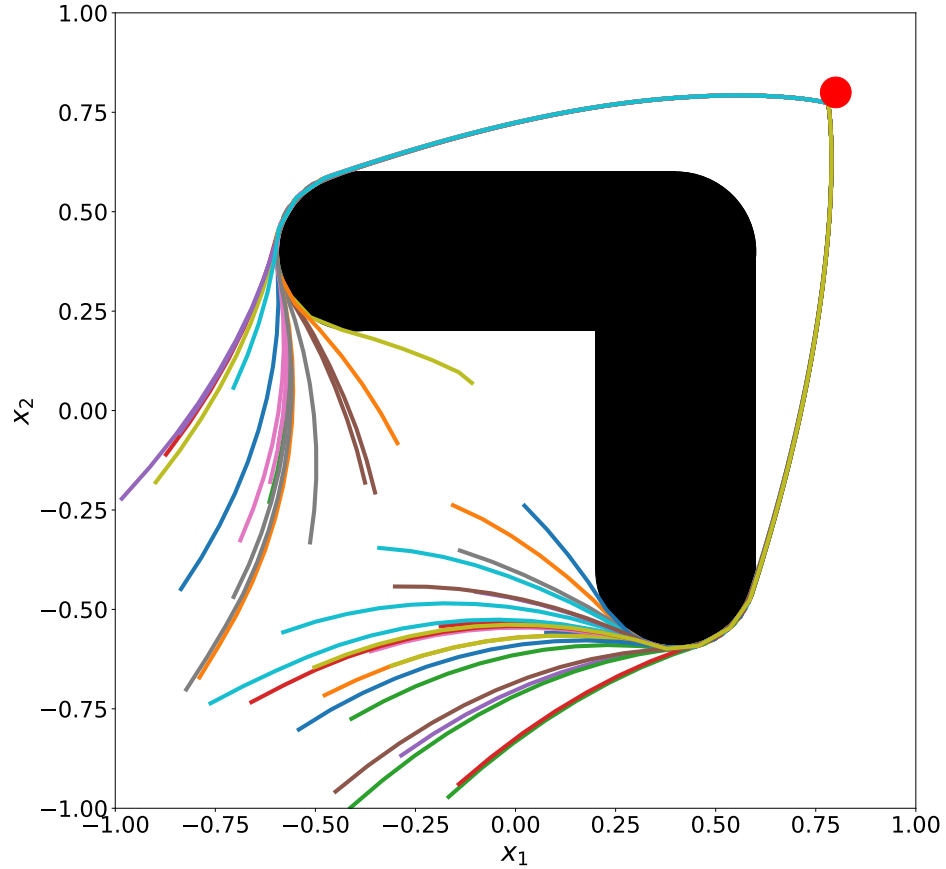


Figure 3.3: Trajectories avoiding the local optima for different initial conditions. The red dot represents the target  $x^*$ .

### 3.3.3 Influence of the horizon

**At train time** In this section, we study the influence of the horizon on the training of the value function. We consider a reaching task with the 7-DoF Kuka iiwa robot without constraints in order to extract the ground truth infinite horizon value function. We fix the last joint and consider a 12-dimensional state containing the joint positions and velocities. The OCP includes an end-effector target reaching cost, joint velocity regularization, and joint torque regularization costs. In this unconstrained setting, the ground truth value can be approximated by the gradient-based solver with a large horizon, e.g.  $T = 200$ . At each value iteration, we collect 10000 trajectories of length 10 by sampling the initial configuration uniformly within the joint and velocity bounds of the robot. Then, we perform 16 epochs. Figure 3.4 shows the MSE between the learned value and the ground truth during training. The larger the horizon is, the faster the algorithm converges to the ground truth. Note that the differences in training time are negligible as most of the time is spent on the SGD to fit the network.

**At test time** In this study, we show that the use of trajectory optimization online allows us to compensate for the value function approximation error due to learning. We use the same setup as in the previous section and show that at test time, a longer horizon helps reduce the running cost. We perform value iteration with a horizon of 10 and use the same parameters as in the previous section. Furthermore, we illustrate that the improvement due to the horizon is not specific to our training procedure but due to the limitation of the neural network’s expressivity. To do so, we train in a supervised way the value function with 100000 ground truth trajectories of length 10. We use the same number of SGD steps as in the overall

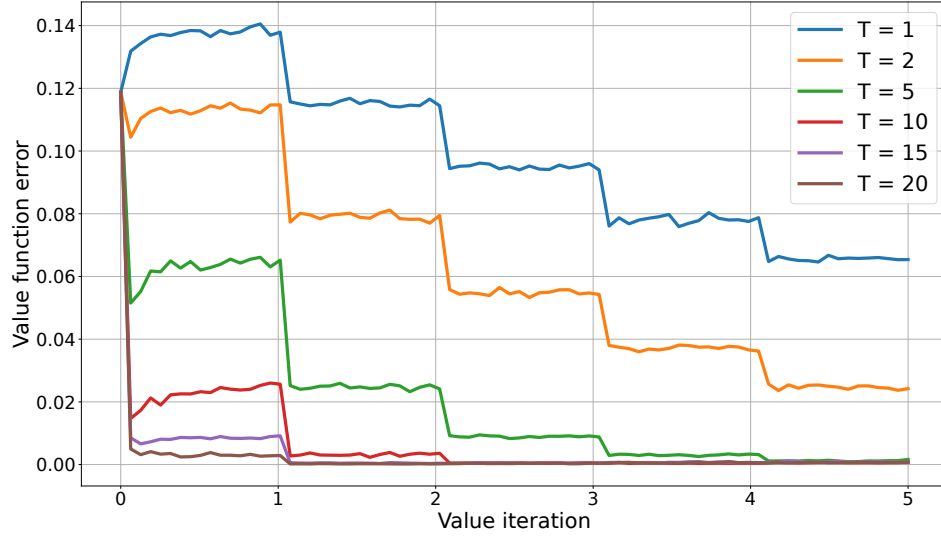


Figure 3.4: Error between the ground truth and the learned value function during training for various horizon length. The larger the horizon in Eq (3.7) is, the faster the algorithm converges to the ground truth.

value iteration learning procedure; the test time performance of this network can be considered as an upper bound on the one of the value iteration network. Lastly, using the ground truth data, we train in a supervised way a policy mapping states to torques (while removing the gravity compensation). Figure 3.5 shows the cost error between various controllers and the ground truth infinite horizon controller. It can be seen that for both value functions, increasing the horizon improves the performance. Also, we can see that the value iteration achieves a performance that is close to the supervised value, which was provided the ground truth values.

### 3.3.4 Experiments on a manipulator

#### 3.3.4.1 Setup

We validated the proposed approach on the KUKA iiwa LBR 1480 in target reaching/tracking and obstacle avoidance tasks. We used a motion capture system

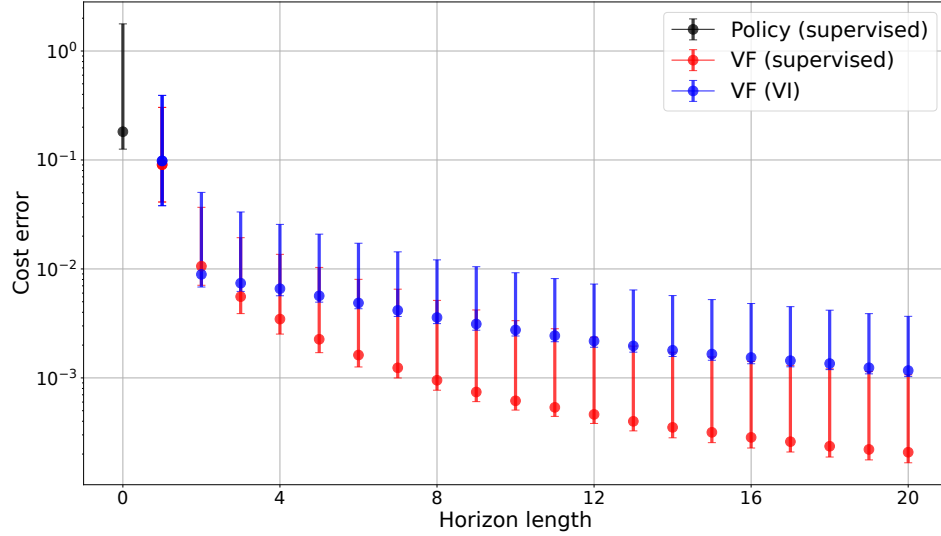


Figure 3.5: We run 1000 MPC simulations starting from random initial states with increasing horizon for each controller. Horizon 0 corresponds to the policy.

(VICON) to track the targets and obstacles. The robot receives joint torque commands and a PD joint state reference at 1 kHz through the FRI. The overall control law applied on the real robot reads

$$\tau = u_0^* + \text{PD}(\hat{x}, x_1^*) \quad (3.17)$$

where  $u_0^*$  and  $x_1^*$  are the optimal control input and the predicted state respectively (computed by the MPC at 100 Hz) and  $\hat{x}$  is the measured state of joint positions and velocities, controlled at 1 kHz by the joint PD

$$\text{PD}(\hat{x}, x_1^*) = -K_P(\hat{q} - q_1^*) - K_D(\hat{\dot{q}} - \dot{q}_1^*) \quad (3.18)$$

We use  $K_P = [150, 150, 100, 100, 50, 10, 10]$  and  $K_D = [25, 25, 20, 20, 14, 6, 6]$ . The measured position is directly read from the robot's encoders, while the velocity is estimated by finite differences. Note that the last joint (wrist "A7") is blocked,



i.e., only controlled by the PD and not part of the model, as it speeds up learning and is not necessary for the end-effector tasks under study (the robot is already redundant for the tasks with 6-DoF).

The obstacle we consider is a thin rod of length 76 cm. To enforce collision avoidance, we cover the robot with capsules and define the distance between the rod and the capsules, as well as between the table and the capsules, as hard constraints. In total, this represents 14 constraints.

In order to deploy the value function on the real system, we condition the network to both the target position and the obstacle pose. The network is a 4-layer MLP with 64 neurons per layer and a residual output of dimension 64. The state is sampled within 70% of the robot’s joint and velocity range, and this range is used as a hard constraint on the state in the OCP. The target’s Cartesian position is sampled within the following bounds  $[0.45, 0.75] \times [-0.2, 0.2] \times [0.15, 0.5]$ . Also, we filtered outlier state samples by rejecting those for which the end-effector lies outside of those bounds, as these are outside the robot’s workspace. Obstacle poses are sampled within a 10 cm cube in front of the robot. We also randomized slightly the orientation through a uniform sampling of the Euler angles within the bound  $[-0.1, 0.1]$ . Lastly, we reject the triplets (state, target, obstacle) for which the robot is in collision with the obstacle or for which inverse kinematics has no solution for the given goal. We found that in this form, this sampling distribution yielded a majority of simple cases, i.e. tasks where the TO can find an optimal path to the goal without getting stuck in local minima. Therefore, to make the distribution more meaningful and representative of the scenario encountered on the real system, we additionally rejected triplets where both the target and the end-effector were above the rod.

We consider a horizon of length  $T = 5$  and we perform 500 value iterations with  $\alpha = 0.01$ . At each iteration, we collect a dataset by sampling 1000 triplets and rolling out trajectories up to a horizon of length 60 or until the robot reaches the target. The target is considered to be reached whenever the running cost is below 0.1. Then, we perform 16 epochs using Adam with a learning rate of 0.0004 and a weight decay of  $10^{-5}$ . The training lasts 3 h 20 min on CPU only.

During deployment, the learned value function is used as a terminal cost in the MPC, while the baseline MPC has no terminal cost. The horizon used is  $T = 10$  nodes, with an OCP discretization of  $\Delta T = 50$  ms and a semi-implicit Euler integration scheme. The maximum number of SQP iterations is set to 6, the termination tolerance to  $10^{-4}$  and the maximum number of QP iterations to 200. The OCP used in the experiments includes a non-collision constraint and state limits (box constraints). The non-collision constraints are defined on all collision pairs existing between the robot links and the rod / table.

#### 3.3.4.2 Pick-and-place with static obstacle

We compared the performance of the proposed approach against the default MPC on a pick-and-place task with a static obstacle. The robot must alternatively reach two end-effector positions while avoiding collision with a fixed rod laying in-between the targets (see Figure 3.6). Figure 3.7 shows the end-effector trajectories of both controllers. It can be seen that the MPC with value function reaches the target (by finding a path moving the end-effector *above* the rod) while the default MPC remains stuck in a local minima (trying to go *underneath* the rod). This behavior can be further understood from Figure 3.8 which shows the cumulative cost over the experiment. The MPC with value function achieves a lower cost since

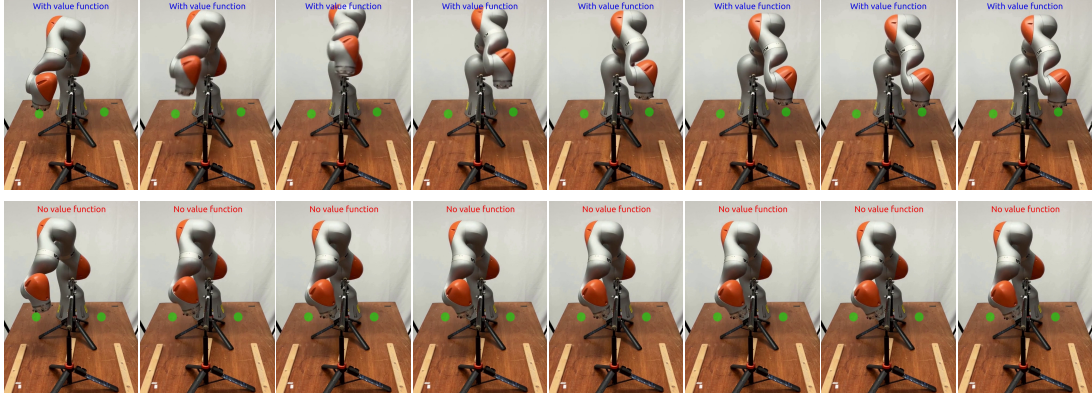


Figure 3.6: Snapshots of pick-and-place task with static obstacle avoidance for the default MPC without value function (bottom) and the proposed MPC with value function (top). The green dots represent the end-effector targets that must be reached alternatively while avoiding collision with the black rod placed in the center.

it eventually reaches the target. But interestingly, this controller initially increases its cost faster than the default MPC. This is explained by higher velocity and torque regularization cost residuals due to the obstacle avoidance motion. Hence this experiment shows the ability of the proposed controller to reason more *globally* thanks to the value function guidance. Indeed, the default MPC remains stuck in a local minima and must trade off the task completion against constraint satisfaction. In contrast, our approach is able to both achieve the task *and* avoid collision with the rod by choosing a different path that initially increases the cost. It is important to remind that both controllers use the same warm-start; the only difference is the terminal cost used.

### 3.3.4.3 Target tracking with static obstacle

In this experiment, we show the ability of our approach to track a moving target while satisfying obstacle avoidance constraints. We use a small cube tracked by

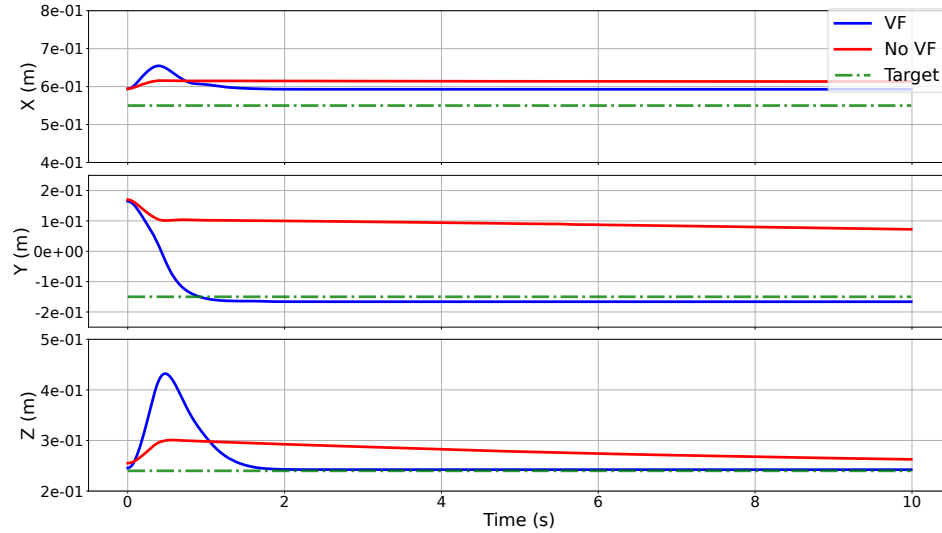


Figure 3.7: End effector trajectory. The robot must reach a target on the over side of the rod marked by the green dots, while avoiding the rod (black dot). The proposed approach (blue) can achieve the task by choosing a path going above the rod, while the default MPC (red) remains stuck in a local minima (trying to go underneath the rod).

the motion capture system to define a moving target. The video shows how the controller can go around the obstacles when the cube is moved from one side to the other of the obstacle. Figure 3.9 depicts the constraints satisfaction between the rod and the capsules.

#### 3.3.4.4 Target tracking with dynamic obstacle

Lastly, we illustrate in the video the ability of the method to deal with out-of-distribution orientation of the rod as well as unexpected disturbances. While it can be seen that the controller is no longer able to systematically avoid local minima due to the obstacle, the hard constraints are maintained. Intuitively, far from the training distribution, the approximated value function is not very meaningful and cannot provide a way to avoid the obstacle. However, the model-based trajectory

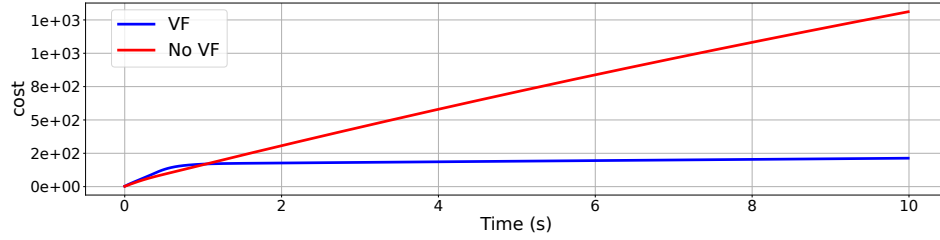


Figure 3.8: Cumulative cost. The proposed approach (blue) achieves a lower cost by avoiding the rod and reaching the target.

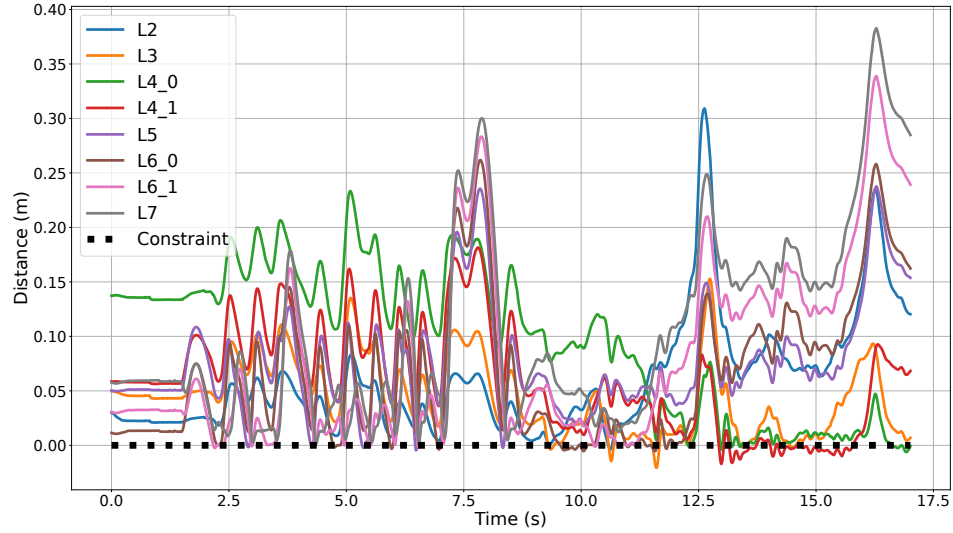


Figure 3.9: Target tracking with static obstacle. Collision distance between the robot's links (approximated as capsules) and the obstacle (moving rod).

optimization maintains the system safe.

### 3.4 Discussion

While value iteration is guaranteed to converge [24], this relies on the assumption that the minimization in Problem (3.7) is global. However, we use a gradient-based solver which can be subject to local minima. Intuitively, the larger  $T$  is, the more the solver is prone to local minima. In practice, we found that tuning appropriately  $T$  was enough to get VI to converge. In the end, the choice of  $T$  regulates the trade-off

between the efficiency of the local solver and the global property of value iteration. Lastly, we have investigated the use of various random warm-starts to search for the global solution; however, the convergence speed-up did not compensate for the additional computational time. Nevertheless, this remains an interesting direction to explore further.

Another limitation of our method is that the gradient-based solver requires a smooth neural network. This constrains the learning architecture and training parameters. More specifically, the SQP solver could not handle a network with ReLu activation and required an appropriate tuning of the weight decay. Without weight decay, the network would overfit and the solver’s number of iterations would diverge. To circumvent the issue, it would be interesting to investigate the use of zero-order methods, which have recently shown promising results [113, 190].

One of the key assumptions of the work is the tractability of the feasible set  $\Omega$ . Although this assumption encompasses a wide set of problems, it would be interesting to study how to generalize to any type of constraint. A naive approach could be to first approximate  $\Omega$  with other methods such as [11, 50] and then apply value iteration. However, it would be interesting to combine those two steps.

This work focuses on the non-discounted setting because this is the original MPC formulation that can guarantee stability [76, 126]. Arguably, using a non-discounted setting requires a cautious design of the problem as it is crucial to ensure that the goal states achieve zero cost. Furthermore, while [24, 82] proved the convergence of value iteration in the non-discounted setting, it is still not clear how to guarantee convergence while approximating the value with neural networks. In contrast, popular RL algorithms usually use a discount factor [168]; it would be interesting to study the impact of the discount parameter during training. However,

it is not clear if the stability guarantees of the infinite horizon [76] will be preserved in that setting.

### 3.5 Conclusion

We have introduced a way to combine constrained TO with RL by using a learned value function as a terminal cost of the MPC. We have demonstrated the benefits of the proposed approach on a reaching task with obstacles on an industrial manipulator. In contrast to traditional MPC, by approximating an infinite horizon OCP, the method can avoid complex local minima. Furthermore, in contrast to RL, the online use of TO allows us to gain accuracy as it can leverage online the model of the robot.

Chapter 2 and 3 demonstrated how to incorporate safety and global optimality in MPC. However, we have so far assumed the state to be known. In practice, we only have access to sensor information. In the next Chapter, we show how online estimation can be used to adapt the MPC model directly from force sensor information.

## Chapter 4

# Force Feedback Model-Predictive Control via Online Estimation

Nonlinear model-predictive control has recently shown its practicability in robotics. However, it remains limited in contact interaction tasks due to its inability to leverage sensed efforts. In this Chapter <sup>1</sup>, we address this issue and show that standard estimation tools [171] together with a reformulation of the optimal control problem, can provide a simple yet effective framework to achieve force-output-feedback MPC.

Force control techniques are classically divided into direct force control and indirect force control [177]. A full introduction is out of the scope, so we only provide here a brief overview and refer the reader to the concise introductory review on active compliant control proposed in [156].

---

<sup>1</sup>This chapter is adapted from the original publication : **A. Jordana**<sup>\*</sup>, S. Kleff<sup>\*</sup>, et al. Force feedback Model-Predictive Control via Online Estimation. IEEE International Conference on Robotics and Automation (ICRA), 2024. This work is the result of a collaboration with equal contribution between Armand Jordana and Sebastien Kleff. In particular, the dissertation's author (Armand Jordana) contributed to the development and implementation of the algorithms as well as the hardware experiments.



Direct methods attempt to regulate the force explicitly using measurement feedback, typically in an integral controller - which is historically considered the best basic strategy for force tracking [178]. It can be combined with motion feedback in complementary task directions [149], or in parallel [38]. While the use of explicit force feedback enables high accuracy tracking, the artificial decoupling of force and motion tasks hides potential conflicts [54, 160] or phenomena such as contact friction [193] and exchange of mechanical work [86].

On the other hand, indirect methods, such as impedance control [85] or admittance control [140, 185], aim at regulating the dynamic relationship between force and motion. While this allows generating stable and compliant contact interactions, such techniques are mainly limited by their force tracking capability: since the force is controlled indirectly through motion regulation, the tracking performance depends on a priori unknown environment parameters [58, 96, 158, 159].

More recently, MPC has shown its ability to accommodate conflicting objectives through constrained nonlinear optimization [59]. Much research has focused on introducing MPC into direct [100, 123] and indirect [16, 69, 99, 130, 181] force control methods, mainly motivated by its ability to satisfy constraints. In contrast to [16, 69, 100, 103, 123], the proposed approach does not require a contact force *dynamics* model, which greatly simplifies the optimization. Unlike [99, 130, 181], we use a force sensor to achieve explicit force tracking rather than impedance/admittance regulation.

Estimation can also be used to improve performance in force tasks. In [154], external forces are estimated with a centroidal model. In [4], a state-dependent force correction model is adapted online. Closer to our work, [110] proposed an active Kalman observer in MPC to reject unmodeled disturbances at the input

level, which can be viewed as a form of model-reference (direct) adaptive control. However, those lines of work do not consider the full dynamics model.

In this Chapter, we propose a novel MPC formulation that allows exploiting direct feedback from force sensors. We show that simple contact models and standard estimation tools allow incorporating force feedback in MPC and achieving state-of-the-art performance. We claim that force feedback in MPC is not as challenging as it seems and that it solves many issues: it circumvents tedious modeling of complex phenomena (contact, friction, etc.), boosts the performance of classical MPC in contact tasks, and does not conflict with optimization, contrary to traditional force control methods.

We propose to use force measurements to estimate online the mismatch between the robot’s dynamics model and measurements. This mismatch is used to correct directly the predictive model or the control objective. This idea resembles that of indirect adaptive control [6], where a model of the plant is identified online to adapt the controller’s parameters. Our approach allows high-quality force tracking accuracy in challenging interaction tasks. Our main contributions are:

- a new framework affording direct force feedback control inside nonlinear MPC based on online estimation and feedback linearization
- a systematic comparative experimental study of our force feedback MPC against traditional techniques.

In particular, we demonstrate that the proposed approach outperforms integral control: it benefits from the same force tracking capability *without* impeding the benefits of MPC. In particular, in contrast to integral control, our approach maintains or improves the MPC running cost performance. It also has the advantage

of being conceptually simple and cheap to implement with existing tools and software.

## 4.1 Background: MPC with rigid contact

In this section, we recall the classical MPC formulation for torque-controlled robots under rigid contacts, and point out its inherent inability to provide force-feedback policies.

### 4.1.1 Classical model-predictive control

MPC solves online the Optimal Control Problem (OCP)

$$\begin{aligned} \min_{x(\cdot), u(\cdot)} \quad & \int_0^T \ell(x(t), u(t), t) dt + \ell_T(x(T)) \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)) \end{aligned} \tag{4.1}$$

where  $x(0) = x^m$  is the initial (measured) state,  $f$  the dynamics model, and  $\ell, \ell_T$  the running and terminal costs. Note that hard constraints on the state and control can be added, as soft penalties or hard constraints - which may be more challenging for real-time applications. This OCP is transcribed into a non-linear program, i.e. the cost and dynamics are discretized using an Euler discretization scheme. This program is solved online at each control cycle. For the remainder, and without limitation, we assume that the robot is fully actuated with  $n$  joints, the state vector  $x = (q, \dot{q}) \in \mathbb{R}^{2n}$  includes the joint positions and velocities and the control vector  $u = \tau \in \mathbb{R}^n$  includes the joint torques.

### 4.1.2 Rigid contact model

In optimization-based control, it is convenient to assume that contacts between the robot and the environment are *rigid*, i.e., pure kinematic constraints that can be resolved at the dynamics level. The dynamics of a robot in contact is given by the following constrained dynamical system corresponding to the KKT conditions of Gauss' principle of least constraint [172]

$$\begin{bmatrix} M(q) & J^T(q) \\ J(q) & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ -F \end{bmatrix} = \begin{bmatrix} \tau - b(q, \dot{q}) \\ -\alpha_0(q, \dot{q}) \end{bmatrix} \quad (4.2)$$

where  $M(q) \in \mathbb{R}^{n \times n}$  is the generalized inertia matrix,  $J(q) \in \mathbb{R}^{n_c \times n}$  the contact Jacobian,  $F \in \mathbb{R}^{n_c}$  the contact force,  $b(q, \dot{q}) \in \mathbb{R}^n$  the nonlinear effects of Coriolis, centrifugal and gravity forces, and  $\alpha_0(q, \dot{q}) \in \mathbb{R}^{n_c}$  the contact acceleration drift. For clarity, the dynamics  $f$  in (4.1), is in fact the solution map of system (4.2), i.e.  $f : (q, \dot{q}, \tau) \mapsto (\ddot{q}, F)$ . The dependencies in  $q, \dot{q}$  will be dropped in the remainder.

### 4.1.3 The challenge of force feedback

While the rigid contact model conveniently fits the MPC framework, it inherently prevents force feedback. The contact force  $F$  corresponds to the Lagrange multiplier of the contact constraint, namely  $J\ddot{q} + \alpha_0 = 0$  (second row of the system (4.2)) [29]. As such, it cannot be controlled in a feedback sense: once  $x = (q, \dot{q})$  and  $F$  are measured,  $u = \tau$  is already completely determined by (4.2). Hence,  $u$  cannot be optimized as a function of  $F$  without creating an algebraic loop. This issue is a typical pathology from control systems with non-zero input-output feedthrough and can be broken by introducing delay [118]. This point was discussed and addressed

in our previous work [103], where actuation was modeled as a low-pass filter, and the joint torques were treated as part of an augmented state. In contrast, we propose in this Chapter to break this coupling thanks to the online estimation without augmenting the state of the MPC.

## 4.2 Force-feedback MPC via online estimation

This section presents a new approach using estimation to leverage force sensor feedback in MPC. It includes an estimator, a reformulation of the MPC problem to include force feedback in the MPC model, and a feedback-linearizing compensation term for unmodeled force directions.

### 4.2.1 Estimation

As explained previously, it is unclear how to achieve force feedback under the rigid contact assumption without introducing delays or more complex contact models. We show here that estimation is a simple way to circumvent this issue by keeping the rigid contact assumption and correcting the model. Indeed, due to numerous model inaccuracies, the force  $F$  predicted by (4.2) rarely matches the force measurement. Hence, a natural idea is to keep track of this mismatch by estimating online the offset between the model and the measurement with standard Kalman filtering [171].

The idea of estimating an offset error to improve the closed-loop performance of the controller is standard in estimation (e.g., [154]). We show that a disturbance  $\Delta$  in the dynamics can incorporate rich force sensor feedback information in the

MPC. We consider a model of the form:

$$M\ddot{q} + b = \tau + J^T F + \mathcal{M}(\Delta), \quad (4.3a)$$

$$J\ddot{q} = -\alpha_0. \quad (4.3b)$$

Here,  $\mathcal{M}$  models how  $\Delta$  offsets the dynamics. While the mismatch can be modeled in many ways, we assume that  $\mathcal{M}$  is linear. Specifically, we consider two different models:

- Torque offset (in joint space) :  $\mathcal{M}(\Delta\tau) = \Delta\tau$
- Force offset (in task space) :  $\mathcal{M}(\Delta F) = J^T \Delta F$

This offset is meant to correct the model mismatch due to inaccurate modeling of, e.g., the dynamics, contact model, external disturbance, etc. The idea is to estimate the offset online, given raw measurement. More precisely, given a prior on the offset  $\hat{\Delta}$ , we use joint positions, velocities, accelerations, torque commands, and force measurements to update the force offset. We assume perfect joint position and velocity measurements, and Gaussian measurement noise:

$$\Delta = \hat{\Delta} + w, \quad w \sim \mathcal{N}(0, P), \quad (4.4a)$$

$$\ddot{q}^m = \ddot{q} + v, \quad v \sim \mathcal{N}(0, Q), \quad (4.4b)$$

$$F^m = F + \eta, \quad \eta \sim \mathcal{N}(0, R), \quad (4.4c)$$

where  $F^m$  is the force measurement and  $\ddot{q}^m$  the acceleration measurement.  $P, Q$  and  $R$  are positive-definite covariance matrices. As it is traditionally done in Kalman filtering, each disturbance distribution is considered to be Gaussian, which allows to solve the Maximum Likelihood Estimation (MLE) problem [171]. Here, the

MLE aims at finding the parameters  $\Delta, \ddot{q}, F$  that maximize the probability density function given the observed measurement and prior force offset:

$$\max_{\Delta, \ddot{q}, F} p(\Delta, \ddot{q}, F \mid \hat{\Delta}, \ddot{q}^m, F^m) \quad (4.5)$$

subject to constraint (4.3a)

Applying the negative logarithm and leveraging the normal distribution assumption, the problem is equivalent to:

$$\begin{aligned} \min_{\Delta, \ddot{q}, F} & \|\Delta - \hat{\Delta}\|_{P^{-1}}^2 + \|\ddot{q} - \ddot{q}^m\|_{Q^{-1}}^2 + \|F - F^m\|_{R^{-1}}^2 \\ & \text{subject to constraint (4.3a)} \end{aligned} \quad (4.6)$$

where  $\|w\|_{P^{-1}}^2 = w^T P^{-1} w$ . If  $\mathcal{M}(\Delta)$  is linear, Problem (4.6) becomes an equality QP and can be solved very efficiently with off-the-shelf solvers. This, in turn, allows high-frequency online estimation, e.g., 5 kHz for a 7 DoF robot. As in a Kalman filter, the obtained estimate  $\Delta$  is used as a prior at the next time step.

Note that other constraints can be considered in the QP, such as inequalities on estimated quantities (e.g. force offset).

**Remark 6.** *If additional inequality constraints are unnecessary, one may solve the problem using a Kalman filter [171]. More specifically, one can use Recursive Least Squares (RLS) [90] with the transition equation,  $\Delta = \hat{\Delta} + w$  along with the observation equation*

$$\begin{bmatrix} \ddot{q}^m \\ F^m \end{bmatrix} = \begin{bmatrix} -M & J^T \\ J & 0 \end{bmatrix}^{-1} \begin{bmatrix} b - \tau - \mathcal{M}(\Delta) \\ -\alpha_0 \end{bmatrix} + \begin{bmatrix} v \\ \eta \end{bmatrix}, \quad (4.7)$$

in order to estimate  $\Delta$  online. Note that if  $\mathcal{M}$  is linear, this observation model is linear, and one can use the RLS equations to derive an update rule on  $\Delta$ .

## 4.2.2 Force feedback in the MPC via estimation

Once estimated, the force offset must be considered by the controller. This will break the coupling between forces and torques discussed in Section 4.1.3 by adding a delay between the measurement and the corrective term  $\Delta F$ .

### 4.2.2.1 Naive inclusion as a corrective control

A naive approach is to add a feedforward term to the optimal torque given by the MPC,  $\tau_{\text{MPC}}$ , to compensate the estimated offset:

$$\tau = \tau_{\text{MPC}} - \mathcal{M}(\Delta). \quad (4.8)$$

Although this work focuses on MPC, this method is agnostic to the nature of the controller.

### 4.2.2.2 Inclusion in the predictive model

Alternatively, the offset can be considered directly in the model used by the MPC. More precisely, we can consider that the offset will be constant over the horizon of the MPC and solve the OCP using as dynamics Eq. (4.3a) (instead of Eq. (4.2)). The MPC model is then updated online at each offset estimate update.

**Remark 7.** *Interestingly, when  $\mathcal{M}(\Delta F) = J^T \Delta F$ , updating the predictive model is in fact equivalent to modifying the force reference in the cost function. More*



specifically, the modified dynamics can be written in the following way:

$$\begin{bmatrix} \ddot{q} \\ F \end{bmatrix} = \begin{bmatrix} -M & J^T \\ J & 0 \end{bmatrix}^{-1} \begin{bmatrix} b - \tau \\ -\alpha_0 \end{bmatrix} - \begin{bmatrix} 0 \\ \Delta F \end{bmatrix}. \quad (4.9)$$

Therefore, the force offset only biases the predicted forces and does not affect the acceleration. This means that this force offset has no impact on the predicted trajectory. The offset will only impact terms of the cost function that include the predicted force. Given a cost of the form  $\ell(x, u, F(x, u, \Delta F))$ , we can simply consider  $\ell(x, u, F(x, u) - \Delta F)$ , and discard  $\Delta F$  from the prediction model. This greatly simplifies the implementation and gives more interpretation to the method. Interestingly, if the cost function does not depend on the force, the force offset will not impact the solution of the OCP.

### 4.2.3 Direct compensation of unmodeled force directions

The above formulation assumes that force can only be exerted in the  $n_c$  constrained dimensions. However, in reality, forces can exist in the other  $6 - n_c$  directions and may interfere with the task if not taken into account (e.g. friction during a polishing task if only the normal force is modeled).

Following [184], instead of using an explicit 6D force model to compute a feed-forward compensation term, we propose to use the force measurements directly. This is in fact a form of Feedback Linearization (FL) as emphasized in [49]. Concretely, we add to the optimal torque given by the MPC the following compensation FL

term

$$\tau = \tau_{\text{MPC}} - J_{6\text{D}}^T S F_{6\text{D}}^m, \quad (4.10)$$

where  $J_{6\text{D}} \in \mathbb{R}^{n \times 6}$  and  $F_{6\text{D}}^m \in \mathbb{R}^6$  are the full 6D Jacobian and measured force, and the selection matrix  $S : \mathbb{R}^6 \rightarrow \mathbb{R}^6$  nullifies the  $n_c$  constrained dimensions. In the experiment section, we will show that this simple FL term will lead to competitive performances with more established yet more complex friction models such as the Coulomb model.

From a control perspective, it could seem unsafe at first glance to use measured forces in the control torque because the robot would always maintain itself in a disturbed state, which would create divergence of the force (e.g., pushing harder). But this would happen only if unmodeled forces are unbounded (i.e. motion is actually constrained by the environment). If the unmodeled forces are bounded, the disturbance would simply generate motion in their directions. For instance, if the normal force on a plane is stably controlled, the lateral forces are bounded by it through the friction cone. In that case, a disturbance increasing the lateral forces would simply make the robot slip. So this FL term is a safe compensation term to use in practical situations.

**Remark 8.** *The FL compensation term in Eq. (4.10) could instead be added directly inside the MPC model, assuming that it remains constant over the whole horizon.*

## 4.3 Experimental study

In this section, we evaluate the performance of the proposed approach through a comparative experimental study on a torque-controlled manipulator. First, we show the major advantage in tracking performance of using explicit force feedback over classical MPC. This benefit is twofold: force feedback enables to effectively cancel friction, and it corrects the model mismatch thanks to online estimation. Second, we demonstrate the benefit of encoding the model mismatch in the task space ( $\Delta F$ ) rather than in the joint space ( $\Delta \tau$ ). Finally, we show how the proposed approach outperforms the most established force control strategy (integral control) by demonstrating that its force tracking performance is identical, but that it additionally aligns with the MPC objectives.

### 4.3.1 Experimental setup

All experiments were performed on the torque-controlled KUKA LBR iiwa R82014. We used an ATI F/T Sensor Mini40 mounted at the tip of the arm on a custom end-effector mount piece. A short MPC horizon (4 nodes of  $6ms$ ) allowed to run the MPC and the estimator synchronously at 1 kHz. The estimation QP problem (4.6) is solved using ProxQP [9], the OCP (4.1) is transcribed using Crocoddyl [120], and rigid-body dynamics are computed using Pinocchio [33]. Our code is publicly available<sup>2</sup>. Moreover, the accompanying video illustrates the robustness of the proposed approach to external disturbances.

---

<sup>2</sup>[https://github.com/machines-in-motion/force\\_observer](https://github.com/machines-in-motion/force_observer)

### 4.3.2 Tasks formulation

#### 4.3.2.1 Polishing task

A constant normal force is exerted on a horizontal plane  $(e_x, e_y)$  while tracking a circular end-effector trajectory. The MPC includes a 1D rigid contact force model ( $n_c = 1$ ) so that the constraint (4.3b) prevents motions in the normal direction  $e_z$ , and ignores tangential forces in the  $(e_x, e_y)$  directions. The cost function is

$$\begin{aligned} \ell(x, u, t) = & w_1 \|x(t) - \bar{x}(t)\|_{Q_1}^2 + w_2 \|u(t) - \bar{u}(t)\|_{Q_2}^2 \\ & + w_3 \|p^{\text{ee}}(t) - \bar{p}^{\text{ee}}(t)\|_{Q_3}^2 + w_4 \|F(t) - \bar{F}(t)\|_{Q_4}^2 \\ & + w_5 \|v^{\text{ee}}(t)\|_{Q_5}^2 + w_6 \|\log_3 (\bar{R}^{\text{ee}}(t)^T R^{\text{ee}}(t))\|_{Q_6}^2 \end{aligned}$$

where  $(w_i, Q_i)_{i=1..6}$  are positive scalar weights and positive diagonal activation matrices,  $\bar{x}(t) = (\bar{q}(t), 0)$  is a reference configuration,  $p^{\text{ee}}(t), F(t), R^{\text{ee}}(t)$  are the position of the end-effector, contact force and end-effector orientation respectively,  $\bar{p}^{\text{ee}}(t), \bar{F}(t), \bar{R}^{\text{ee}}(t)$  are their respective references,  $v^{\text{ee}}(t)$  is the end-effector velocity,  $\bar{u}(t) = g(q(t)) - J^T F(t)$  is the gravity compensation torque under external forces,  $\log_3 : SO(3) \mapsto so(3)$  is the logarithm map on rotations. The circular trajectory  $\bar{p}^{\text{ee}}(t)$  has a diameter of 14 cm and a speed of  $3 \text{ rad s}^{-1}$ , unless otherwise stated. The reference normal force is constant  $\bar{F} = 50 \text{ N}$ .

#### 4.3.2.2 Force step tracking task

A 3D contact force ( $n_c = 3$ ) step signal is tracked. Hence the motion of the end-effector is constrained in normal and tangential directions. The cost function has the same form as the polishing cost function (4.11), with the only differences

that  $F(t), \bar{F}(t)$  are 3D, the reference end-effector position  $p^{\text{ee}}(t)$  is now constant, and the force reference is defined as  $\bar{F}(t) = (\bar{F}_x(t), \bar{F}_y(t), \bar{F}_z(t))$  where  $\bar{F}_x(t)$  is a step signal from  $-10$  N to  $10$  N,  $\bar{F}_y(t) = 0$  N and  $\bar{F}_z(t) = 100$  N are constant.

#### 4.3.2.3 Energy minimization

A sinusoidal joint position trajectory is tracked while maintaining a fixed 3D contact with the horizontal plane and minimizing  $\|\tau\|^2$ . The cost function is similar to the polishing (4.11), except that the reference configuration  $\bar{q}(t)$  is no longer constant, no end-effector cost is used ( $w_3 = w_5 = 0$ ), and the control regularization term is turned into an energy term ( $\bar{u}(t) = 0$ ). The reference joint trajectory is a sine on the A3 joint with an amplitude of  $0.2$  rad and a frequency of  $2$  Hz. Here the force objective acts as a regularization term to avoid slipping and large forces (i.e.  $w_4 \ll w_1, w_2, w_6$ ) and the reference is  $\bar{F}(t) = (0, 0, 50)$ .

### 4.3.3 Friction model vs direct measurement feedback (FL)

We evaluate the effect of force feedback as a direct compensation of the contact friction (Section 4.2.3). We compare its performance on the polishing task against the classical MPC (i.e., without compensation) and the well-known Coulomb's friction model

$$F_T = -\mu \frac{v}{\|v\|} F_N, \quad (4.11)$$

where  $F_T \in \mathbb{R}^2$  is the tangential force,  $F_N \triangleq F \in \mathbb{R}$  is the normal force,  $v \in \mathbb{R}^2$  is the tangential velocity of the contact point and  $\mu$  is the dynamic friction coefficient. This model is clearly discontinuous in  $v$  so in order to avoid chattering phenomena,

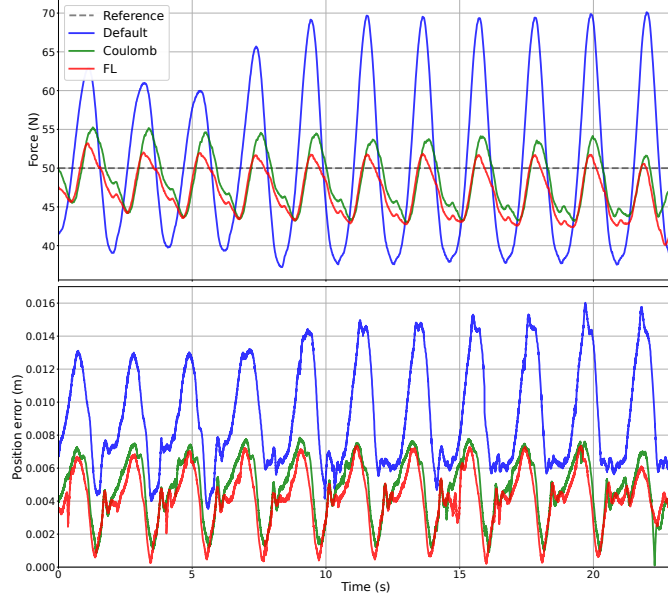


Figure 4.1: Normal force trajectories of the medium-velocity polishing task. The blue curve is the classical MPC without friction compensation, the green curve is the classical MPC with the Coulomb model compensation, and the red curve is the classical MPC with FL compensation.

we consider the following smooth relaxation

$$F_T = -\mu \frac{\tanh(\epsilon \|v\|)}{\sqrt{2}} \frac{v}{\|v\|} F_N, \quad (4.12)$$

where we used  $\mu = 0.35$  and  $\epsilon = 10$ . Our results are reported in Table 4.1 for several polishing speeds. We can see that the Coulomb model is slightly better in fast motions but less performing in slow motions. Figure 4.1 shows the corresponding force trajectories for the medium-speed polishing task. Note that the FL compensation term only uses the 3D Jacobian as the contact torques are negligible in that task. These experiments confirm that considering the friction forces substantially increases performance w.r.t. classical MPC. Moreover, it shows that explicit force feedback from sensors can effectively be used as an FL term to

|        |           | Default          | FL              | Coulomb         |
|--------|-----------|------------------|-----------------|-----------------|
| Slow   | (1 rad/s) | $7.67 \pm 0.55$  | $3.83 \pm 0.17$ | $4.72 \pm 0.21$ |
| Medium | (3 rad/s) | $9.66 \pm 1.38$  | $3.92 \pm 0.56$ | $3.99 \pm 0.33$ |
| Fast   | (6 rad/s) | $16.42 \pm 0.79$ | $5.22 \pm 0.32$ | $4.82 \pm 0.25$ |

Table 4.1: Mean-absolute error (MAE) of the normal force (in N) for the polishing task over 10 circles: classical MPC (Default), FL compensation (4.10) and Coulomb model (4.12).

|                    | $\Delta\tau$    | $\Delta F$      |
|--------------------|-----------------|-----------------|
| Corrective control | $2.01 \pm 0.08$ | $1.55 \pm 0.03$ |
| Predictive model   | $1.95 \pm 0.07$ | $1.55 \pm 0.04$ |

Table 4.2: MAE of the normal force (in N) for the polishing task: force offset  $\Delta F$  vs. torque offset  $\Delta\tau$ , used in the control loop either in the "predictive model" way of 4.2.2.2 or in the "corrective control" way of 4.2.2.1.

directly compensate for friction effects and that it leads to a similar performance to well-established friction models.

As pointed out in Remark 8, it would be interesting to use the Coulomb model inside the MPC so that lateral forces are predicted using velocity and rigid normal force predictions, but this raises challenging issues (non-smoothness, insufficient software, breaks symmetry of KKT (4.2), etc.).

#### 4.3.4 Comparison between force offset and torque offset

In this experiment, we compare the two mismatch models introduced in Section 4.2.1, namely the torque offset  $\Delta\tau$  and the force offset  $\Delta F$ . Although capturing all disturbances in  $\Delta\tau$  seems intuitive, experimental comparisons on the polishing task reveal a higher tracking accuracy for  $\Delta F$ . For each model, we implemented the two ways of incorporating the correction into the MPC, namely

- The "corrective control" way of 4.2.2.1: the correction is added to the optimal torque as a feedforward input

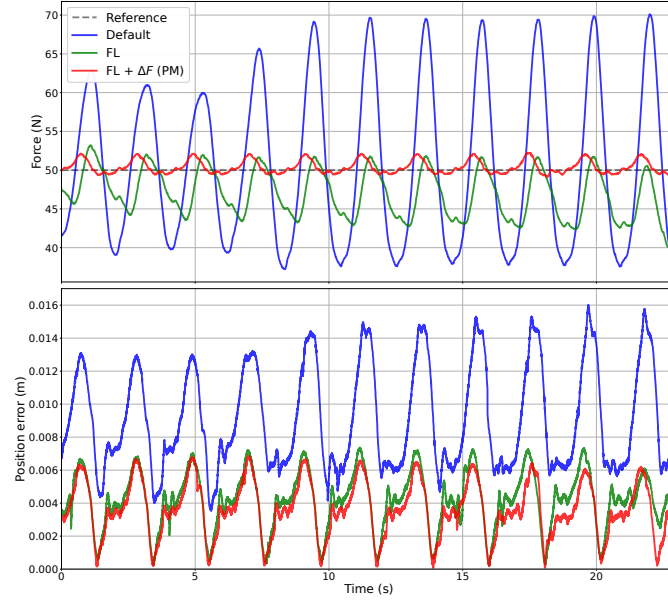


Figure 4.2: Normal force (top) and end-effector position error (bottom) for the polishing task: in blue the classical MPC (4.1), in green the classical MPC with the FL compensation term (4.2.3), in red the proposed approach with FL compensation and the force offset in the predictive model (4.2.2.2).

- The "predictive model" way of 4.2.2.2: the correction is added directly to the model

Figure 4.2 illustrates how force feedback improves both the force tracking and the end-effector position tracking. Our results are summarized in Table 4.2. There is a notable performance difference between  $\Delta F$  and  $\Delta \tau$  with a clear advantage for the force offset. Intuitively, the torque offset estimates perturbations unrelated to the contact (e.g. joint stiction) while the force offset only corrects what is necessary to improve the force tracking. There is, however, no clear difference in performance between using the estimate as a corrective control or in the predictive model. There seems to be a slight advantage for the predictive model, but the performance gap is too shallow to draw any conclusions.



### 4.3.5 Integral force control

Our approach is now compared to the most established direct force control approach - integral control. We were not able to find a difference in performance between using the integral term in the predictive model or as a corrective control. This question being out of the scope of this Chapter, we propose to consider only the latter:

$$\tau = \tau_{\text{MPC}} - J(q)^T \left( -K_I \int_0^t (F(t') - \bar{F}(t')) dt' \right) \quad (4.13)$$

Note that we deliberately chose not to include a proportional and a derivative control term as Volpe et al. [178] demonstrated both theoretically and experimentally that pure integral gain control was the best choice for accurate force tracking.

#### 4.3.5.1 Polishing

We observed the same force tracking performance on the polishing task for the integral controller ( $1.69 \pm 0.05$  N) than for the proposed approach (cf. Table 4.2,  $\Delta F$  as corrective control).

#### 4.3.5.2 Step experiment

We show in this experiment that the proposed approach and integral control have equivalent force tracking performances on a force step tracking task. The force trajectories are in Figure 4.3. We also report the average force tracking error of all the controllers in Table 4.3.

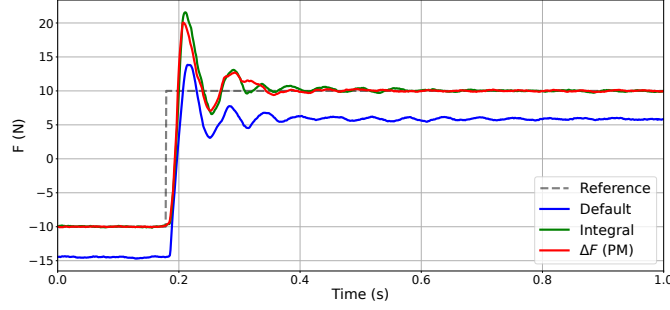


Figure 4.3: Lateral force trajectories in the  $e_x$  direction for the force step tracking task: the blue curve is the classical MPC (Default), the green curve is the classical MPC with with integral control (Integral) and the red curve is the force offset estimation  $\Delta F$  included in the predictive model ( $\Delta F$  (PM)).

|                                    | Avg. error |
|------------------------------------|------------|
| Default                            | 1.99       |
| $\Delta F$ (predictive model)      | 0.71       |
| $\Delta F$ (corrective control)    | 0.60       |
| $\Delta \tau$ (predictive model)   | 0.80       |
| $\Delta \tau$ (corrective control) | 0.87       |
| Integral control                   | 0.68       |

Table 4.3: MAE of the normal force error for a step tracking task for different controllers: classical MPC (Default), force offset estimation ( $\Delta F$ ), torque offset estimation ( $\Delta \tau$ ) and integral control.  $\Delta F$  and  $\Delta \tau$  are used as corrective control (4.2.2.1) or in the predictive model (4.2.2.2).

#### 4.3.5.3 Energy minimization

In this experiment, we illustrate the ability of force feedback MPC to achieve contact tasks with conflicting objectives. Table 4.4 shows how the proposed force estimation approach aligns with the MPC objectives by trading off force tracking against energy minimization: its overall cost is lower than the integral controller, which conflicts with the MPC and generates a high cost. These results also show interestingly that somehow, the torque offset estimation ( $\Delta \tau$ ) uses less energy than the force offset estimation ( $\Delta F$ ), although it yields a slightly higher cost overall. This suggests that encoding the mismatch as a joint torque offset may have its own

|                                    | Avg. $\ \tau\ ^2$ | Total cost       |
|------------------------------------|-------------------|------------------|
| Default                            | $136 \pm 21$      | $0.44 \pm 0.02$  |
| $\Delta F$ (predictive model)      | $139 \pm 13$      | $0.43 \pm 0.01$  |
| $\Delta F$ (corrective control)    | $145 \pm 18$      | $0.43 \pm 0.02$  |
| $\Delta \tau$ (predictive model)   | $131 \pm 21$      | $0.48 \pm 0.01$  |
| $\Delta \tau$ (corrective control) | $132 \pm 22$      | $0.51 \pm 0.02$  |
| Integral control                   | $1052 \pm 29$     | $0.82 \pm 0.027$ |

Table 4.4: Average squared torque and total cost for each controller for the energy task: classical MPC (Default), force offset estimation ( $\Delta F$ ), torque offset estimation ( $\Delta \tau$ ) and integral control.  $\Delta F$  and  $\Delta \tau$  are used as corrective control (4.2.2.1) or in the predictive model (4.2.2.2).

benefits, other than accurate force tracking. The accompanying video illustrates the relative importance of  $w_2 \|\tau\|_{Q_2}^2$  w.r.t. the total cost.

## 4.4 Conclusion

In this Chapter, we proposed a simple approach to achieve force feedback in MPC that relies on the online estimation of the mismatch between the predicted forces and the force measurements. Our experiments showed that force feedback effectively cancels friction and brings the force tracking performance to the level of the most established direct force control strategies. We also studied two variants of our approach: the estimation of a torque offset in the joint space, and the estimation of a force offset in the task space. Our experiments show that the force offset yields a more accurate force tracking while the torque offset is more generic and can enhance other criteria (e.g., energy minimization).

Through this study of force-feedback MPC, we have seen how online estimation can be used to adapt the model of the controller to unforeseen situations. While this can be very effective, this ignores the estimation uncertainty and therefore cannot

reason about the notion of risk. In safety-critical applications, naively considering the estimated state to be the real state can be suboptimal or lead to catastrophic results. In the next Part, we investigate how to solve efficiently formulations that consider the estimation and control problem jointly in order to reason about the risk due to the perception uncertainty.

## Part II

# Reasoning about the Perception Uncertainty



## Chapter 5

# Stagewise Newton Method for Dynamic Game Control With Imperfect State Observation

In this Chapter<sup>1</sup>, we study dynamic game optimal control with imperfect state observations. This formulation solves jointly the estimation and control problem in order to reason online about the perception uncertainty. While this formulation has been widely studied theoretically, the lack of an efficient solver has hindered its deployment on hardware. Hence, we introduce a stagewise implementation of the Newton method to efficiently solve *dynamic game control with imperfect state observation* in the nonlinear case.

---

<sup>1</sup>This Chapter is adapted from the following publication: **A. Jordana** et al. "Stagewise newton method for dynamic game control with imperfect state observation." IEEE Control Systems Letters, 2022.

## 5.1 Dynamic game control with imperfect state observation

Similar to [39, 40], this work studies a special class of nonlinear dynamic games with imperfect state observation [94]. Given a history of measurements  $y_{1:t}$ , a history of control inputs,  $u_{0:t-1}$  and a prior on the initial state  $\hat{x}_0$ , we aim to find a control sequence  $u_{t:T-1}$  that minimizes a given cost  $\ell$  while an opposing player aims to find the disturbances  $(w_{0:T}, \gamma_{1:t})$  that maximize this cost  $\ell$  minus a weighted norm of the disturbances. Such a problem is formally written as:

$$\begin{aligned} \min_{u_{t:T-1}} \max_{w_{0:T}} \max_{\gamma_{1:t}} \sum_{j=0}^{T-1} \ell_j(x_j, u_j) + \ell_T(x_T) \\ - \frac{1}{2\mu} \left( \omega_0^T P^{-1} \omega_0 + \sum_{j=1}^t \gamma_j^T R_j^{-1} \gamma_j + \sum_{j=1}^T w_j^T Q_j^{-1} w_j \right) \end{aligned} \quad (5.1)$$

$$\text{subject to } x_0 = \hat{x}_0 + w_0, \quad (5.2a)$$

$$x_{j+1} = f_j(x_j, u_j) + w_{j+1}, \quad 0 \leq j < T, \quad (5.2b)$$

$$y_j = h_j(x_j) + \gamma_j, \quad 1 \leq j \leq t. \quad (5.2c)$$

where  $\mu > 0$ .  $x_j$  is the state,  $\omega_j$  the process disturbance,  $\gamma_j$  the measurement disturbance,  $T$  the time horizon,  $t$  the current time. The transition model  $f_j$ , the measurement model  $h_j$  and the cost  $\ell_j$  are assumed to be  $\mathcal{C}^2$ . The measurement uncertainty  $R_j$ , the process uncertainty  $Q_j$  and the initial state uncertainty  $P$  are positive definite matrices.

Interestingly, this problem encompasses various formulations of control and



estimation. If  $t = 0$  and if  $w_0$  is fixed to zero, we recover dynamic game control with perfect state information. Additionally, if  $t = 0$ , in the limit where  $\mu$  tends to zero, we find the generic optimal control formulation [30]. And lastly, if  $t = T$  and if we consider all the cost  $\ell_j$  to be null, then, (5.1) is equivalent to maximizing the MAP.

In the linear dynamics and quadratic cost case, Jacobson [91] showed that dynamic game control is equivalent to risk-sensitive control and derived a closed-form solution. Later, Whittle [186] extended the results to the linear quadratic case with imperfect state observations. In [78], a first iterative version of Whittle's solution was introduced to tackle the nonlinear risk-sensitive problem with imperfect observations. However, the stochastic nature of the problem hindered the development of theoretical guarantees. Although dynamic game and risk-sensitive control are equivalent in the linear quadratic case [91], this is no longer the case in the nonlinear setting [30]. Nonetheless, dynamic game control is tightly connected to robust and risk-sensitive control. In [30, 94], James and Campi showed that dynamic game control can be interpreted as the limit case of risk-sensitive control when noise tends to zero. Recently, Başar [12] presented a detailed overview of the connections between both problems in continuous time. Additionally, Başar and Bernhard [15, 20] established the connections between dynamic game control and  $H^\infty$ -Optimal Control both in the perfect and imperfect state information cases.

For nonlinear systems, estimating a state trajectory corresponding to some given measurements is usually intractable analytically. A common approach is to model the noise as Gaussian and to maximize the Maximum A Posteriori (MAP) [42]. If the dynamics are affine, then the problem can be solved analytically with the so-called Rauch–Tung–Striebel (RTS) smoother [151]. The RTS smoother is made

of a forward recursion which resembles the Kalman Filter (KF) and a backward recursion. In the nonlinear case, iterative schemes are usually used. A popular choice is the iterative Kalman smoother which is equivalent to a Gauss-Newton method on the MAP [17]. The estimation part of our proposed solution resembles a risk-sensitive version of this smoother.

In optimal control or dynamic game control with perfect state information, various numerical optimization algorithms have been developed to iteratively find solutions in the nonlinear case. In optimal control, the most analogous to our work are Differential Dynamic Programming (DDP) [137] and the stagewise implementation of Newton's method [55]. The stagewise Newton method is an exact implementation of Newton's method that exploits the specific structure of the Hessian matrix in order to scale linearly with the time horizon. DDP is an iterative algorithm that takes an update step on the control input by applying dynamic programming on a quadratic approximation of the value function. In [137], Murray showed that DDP is very similar to a Newton step and inherits its convergence properties. For dynamic game control with perfect state information, the seminal work from [133, 135] introduced minimax DDP showing that DDP could be extended to zero-sum two-player games. Recently, [45] further extended the concepts of stagewise Newton method and DDP to nonzero-sum games with an arbitrary number of players in the full information case.

## 5.2 Stagewise Newton method

### 5.2.1 First order conditions for a Nash equilibrium

The main challenge in the problem formulation (5.1) is the equality constraint maintaining the dynamic feasibility. A popular approach [133] is to derive a DDP-like algorithm by sequentially taking quadratic approximations of the value function recursion. However, a stagewise Newton step can readily be derived. One of the key features of the dynamic game (5.1) is that by changing the decision variable of the opposing player, one can transform the problem into an unconstrained one [94]. Indeed, we can use the equality constraints of Eq. (5.2) to replace disturbance maximization into a maximization over the state sequence. Then the problem loses its equality constraints and can be formulated as the search of the saddle point of

$$\begin{aligned}
 J(x_{0:T}, u_{t:T-1}) &= \sum_{j=0}^{T-1} \ell_j(x_j, u_j) + \ell_T(x_T) \\
 &\quad - \frac{1}{2\mu} (x_0 - \hat{x}_0)^T P^{-1} (x_0 - \hat{x}_0) \\
 &\quad - \frac{1}{2\mu} \sum_{j=1}^t (y_j - h_j(x_j))^T R_j^{-1} (y_j - h_j(x_j)) \\
 &\quad - \frac{1}{2\mu} \sum_{j=0}^{T-1} (x_{j+1} - f_j(x_j, u_j))^T Q_{j+1}^{-1} (x_{j+1} - f_j(x_j, u_j)).
 \end{aligned} \tag{5.3}$$

However, without convexity and concavity assumptions, we cannot aim at finding global solutions of the minimax problem. Hence, we restrict our attention to local Nash equilibrium, namely a point  $(x_{0:T}^*, u_{t:T-1}^*)$  such that there exists  $\delta > 0$  such that for any  $(x_{0:T}, u_{t:T-1})$  satisfying  $\|x_{0:T} - x_{0:T}^*\| < \delta$  and  $\|u_{t:T-1} - u_{t:T-1}^*\| < \delta$ ,

we have

$$J(x_{0:T}, u_{t:T-1}^*) \leq J(x_{0:T}^*, u_{t:T-1}^*) \leq J(x_{0:T}^*, u_{t:T-1}). \quad (5.4)$$

A standard approach to this problem is to search for a stationary point [13]:

$$\begin{pmatrix} \frac{\partial J}{\partial x_{0:T}}(x_{0:T}^*, u_{t:T-1}^*) \\ \frac{\partial J}{\partial u_{t:T-1}}(x_{0:T}^*, u_{t:T-1}^*) \end{pmatrix} = 0 \quad (5.5)$$

Interestingly, the change of decision variable in Eq. 5.3 turned the problem into an unconstrained one but made no assumption on the structure of the cost. The only required assumption is that each disturbance is in a one-to-one map with the state at each time step.

### 5.2.2 About the Linear Quadratic case

In the linear quadratic case, Whittle has shown that under some conditions, the saddle point of (5.4) is global and can be computed analytically. More precisely, the order of the minimization and maximization in (5.1) can be interchanged, namely the lower value and upper value of the game are equal. Despite this result, one of the difficulties of the problem (5.1) is that it links estimation and control. One of the major contributions of Whittle is the introduction of the notion of past stress and future stress, showing that the KF and LQR principles can still be applied. In other words, the problem can be solved by performing a backward recursion on the controls and future states and a forward recursion on the past states. The future stress recursion can be interpreted as a value function recursion similar to LQR, while the past stress can be interpreted as a rollout of KF. Here, we use these two

principles to efficiently solve iteratively the nonlinear case.

### 5.2.3 A stagewise Newton's method

In this section, a stagewise formulation of the Newton method to find a stationary point of  $J$  is introduced. While a naive implementation of the Newton method would yield a complexity of  $O(T^3)$ , it is shown that the special structure of the Hessian induced by time can be exploited in order to obtain a linear complexity in time  $O(T)$ . In the perfect state observation case, [55] and [45] derived a stagewise Newton method with a backward recursion on the controls. However, with imperfect state observation, it is no longer clear how to do this with only one recursion. Instead, we show that the principles introduced by Whittle can be applied.

To ensure that the proposed method is well defined and to guarantee convergence, we assume that the cost satisfies smoothness and non-degeneracy conditions required for the convergence of Newton's method [143]. As the cost (5.3) is unconstrained, the gradients and Hessian of the cost can readily be computed. At iteration  $i$ , given a guess  $x_0^i, x_1^i, \dots, x_T^i, u_t^i, \dots, u_{T-1}^i$ , also referred to as the nominal trajectory, the Newton step, denoted by  $p$ , satisfies

$$\begin{pmatrix} \frac{\partial^2 J}{\partial x_{0:T} \partial x_{0:T}} & \frac{\partial^2 J}{\partial x_{0:T} \partial u_{t:T-1}} \\ \frac{\partial^2 J}{\partial u_{t:T-1} \partial x_{0:T}} & \frac{\partial^2 J}{\partial u_{t:T-1} \partial u_{t:T-1}} \end{pmatrix} p = \begin{pmatrix} \frac{\partial J}{\partial x_{0:T}} \\ \frac{\partial J}{\partial u_{t:T-1}} \end{pmatrix} \quad (5.6)$$

Here,  $p = \begin{pmatrix} p_{x_{0:T}}^T & p_{u_{t:T-1}}^T \end{pmatrix}^T$  where  $p_{x_{0:T}} \in \mathbb{R}^{(T+1)n_x}$  is a stack of vectors  $p_{x_k} \in \mathbb{R}^{n_x}$  with  $n_x$  being the dimension of the state space. Similarly,  $p_{u_{t:T-1}} \in \mathbb{R}^{(T-t)n_u}$  is a stack of vectors  $p_{u_k} \in \mathbb{R}^{n_u}$  with  $n_u$  being the dimension of the control space. To simplify the notations, we define an augmented Hessian of the cost that contains

the second-order derivatives of the dynamics for all  $k < T$ .

$$\begin{aligned}\bar{\ell}_k^{xx} &= \ell_k^{xx} + \mu^{-1} w_{k+1}^i{}^T Q_{k+1}^{-1} f_k^{xx} + \mu^{-1} \mathbf{1}_{1 \leq k \leq t} \gamma_k^i{}^T R_k^{-1} h_k^{xx}, \\ \bar{\ell}_k^{xu} &= \bar{\ell}_k^{ux^T} = \ell_k^{xu} + \mu^{-1} w_{k+1}^i{}^T Q_{k+1}^{-1} f_k^{xu}, \\ \bar{\ell}_k^{uu} &= \ell_k^{uu} + \mu^{-1} w_{k+1}^i{}^T Q_{k+1}^{-1} f_k^{uu},\end{aligned}\tag{5.7}$$

where the derivatives are evaluated at the current guess and where:

$$w_{k+1}^i := x_{k+1}^i - f_k(x_k^i, u_k^i) \tag{5.8}$$

$$\gamma_k^i := y_k - h_k(x_k^i) \tag{5.9}$$

Here, the second order derivatives of the dynamics are tensors. The exact definition of the tensor indexing and the tensor product is provided in Appendix B.

The next three propositions are analogous to the principles introduced by Whittle: the past stress recursion, the future stress recursion, and the coupling of the past and future stress recursions. The first proposition, analogous to the future stress recursion, expresses every future state and control update step as a function of  $p_{x_t}$ .

**Proposition 3** (Future stress). *In Equation (5.6), the last  $(T - t)(n_x + n_u)$  rows are equivalent to:*

$$\begin{aligned}\forall k \geq t, \quad p_{u_k} &= G_k p_{x_k} + g_k \\ p_{x_{k+1}} &= (I - \mu Q_{k+1} V_{k+1})^{-1} (f_k^x p_{x_k} + f_k^u p_{u_k} \\ &\quad + \mu Q_{k+1} v_{k+1} - w_{k+1}^i)\end{aligned}\tag{5.10}$$

where  $V_k$  and  $v_k$  are solutions of the backward recursion:

$$\Gamma_{k+1} = I - \mu V_{k+1} Q_{k+1} \quad (5.11)$$

$$Q_{uu} = \bar{\ell}_k^{uu} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^u$$

$$Q_{ux} = \bar{\ell}_k^{ux} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x$$

$$Q_u = \ell_k^u + f_k^{uT} \Gamma_{k+1}^{-1} (v_{k+1} - V_{k+1} w_{k+1}^i)$$

$$G_k = -Q_{uu}^{-1} Q_{ux}$$

$$g_k = -Q_{uu}^{-1} Q_u$$

$$V_k = \bar{\ell}_k^{xx} + f_k^{xT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x + Q_{ux}^T G_k$$

$$v_k = \ell_k^x + f_k^{xT} \Gamma_{k+1}^{-1} (v_{k+1} - V_{k+1} w_{k+1}^i) + Q_{ux}^T g_k$$

with the terminal condition

$$V_T = \ell_T^{xx}, \quad v_T = \ell_T^x. \quad (5.12)$$

In those equations,  $\mu$  intervenes only in  $\Gamma_k$  and the augmented terms of the cost. Interestingly,  $\Gamma_k^{-1}$  shifts, at each time step, the value function terms  $V_k$  and  $v_k$ . Then, the second proposition, analogous to the past stress recursion, expresses every past state update step as a function of  $p_{x_t}$ .

**Proposition 4** (Past stress). *In Equation (5.6), if  $t \geq 1$ , the first  $(t-1)n_x$  rows are equivalent to:  $\forall k = 0, \dots, t-1$ ,*

$$p_{x_k} = E_{k+1}^{-1} (f_k^{xT} Q_{k+1}^{-1} (w_{k+1}^i + p_{x_{k+1}}) + P_k^{-1} \hat{\mu}_k + \mu l_k^x) \quad (5.13)$$

where  $P_k$  and  $\hat{\mu}_k$  are solutions of the forward recursion:

$$\begin{aligned}
E_{k+1} &= P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{l}_k^{xx} \\
\bar{P}_{k+1} &= Q_{k+1} + f_k^x (P_k^{-1} - \mu \bar{\ell}_k^{xx})^{-1} f_k^{xT} \\
K_{k+1} &= \bar{P}_{k+1} h_{k+1}^{xT} (R_{k+1} + h_{k+1}^x \bar{P}_{k+1} h_{k+1}^{xT})^{-1} \\
P_{k+1} &= (I - K_{k+1} h_{k+1}^x) \bar{P}_{k+1} \\
\hat{\mu}_{k+1} &= (I - K_{k+1} h_{k+1}^x) (f_k^x \hat{\mu}_k - w_{k+1}^i) + K_{k+1} \gamma_{k+1}^i \\
&\quad + \mu P_{k+1} Q_{k+1}^{-1} f_k^x E_{k+1}^{-1} (\bar{\ell}_k^{xx} \hat{\mu}_k + \ell_k^x)
\end{aligned} \tag{5.14}$$

with the initialization

$$P_0 = P, \quad \hat{\mu}_0 = \hat{x}_0 - x_0^i. \tag{5.15}$$

Interestingly, if all the cost terms  $\ell_j$  are zero and if  $t = T$ , the method is equivalent to a Newton method on the MAP. Furthermore, if the second-order derivatives of the measurement function are omitted, then the algorithm is equivalent to the iterative Kalman Smoother. Finally, the third proposition shows how both past stress and future stress recursions can be coupled to find the update step  $p_{x_t}$ .

**Proposition 5** (Coupling). *In Equation (5.6), the remaining rows (from  $tn_x + 1$  to  $(t + 1)n_x$ ) are equivalent to*

$$p_{x_t} = (P_t^{-1} - \mu V_t)^{-1} (P_t^{-1} \hat{\mu}_t + \mu v_t). \tag{5.16}$$

In the limit case when  $\mu$  tends to zero, the estimation and control are decoupled and we recover the usual certainty equivalence principle:  $p_{x_t} = \hat{\mu}_t$ . The algorithm



is then equivalent to an iterative estimator and an iterative controller running independently. More precisely, at each iteration, the controller uses the current estimate of the smoother.

*Proof.* The proof follows from the analytical derivations of the gradient and the Hessian of (5.3), a forward induction from 0 to  $t$  and a backward induction from  $T$  to  $t$ . The complete proof is provided in the Appendix B.  $\square$

---

**Algorithm 4:** Stagewise Newton step

---

**Input:**  $x_0^i, x_1^i, \dots, x_T^i, u_t^i, \dots, u_{T-1}^i$   
**// Estimation forward pass**  
1  $P_0 \leftarrow P, \hat{\mu}_0 \leftarrow \hat{x}_0 - x_0^i$   
2 **for**  $k = 0, \dots, t-1$  **do**  
3    $P_{k+1}, \hat{\mu}_{k+1} \leftarrow \text{Eq. (5.14)}$   
**// Control backward pass**  
4  $V_T \leftarrow \ell_T^{xx}, v_T \leftarrow \ell_T^x$   
5 **for**  $k = T-1, \dots, t$  **do**  
6    $V_k, v_k \leftarrow \text{Eq. (5.11)}$   
**// Estimation and control coupling**  
7  $p_{x_t} \leftarrow (P_t^{-1} - \mu V_t)^{-1} (P_t^{-1} \hat{\mu}_t + \mu v_t)$   
**// Estimation backward pass**  
8 **for**  $k = t-1, \dots, 0$  **do**  
9    $p_{x_k} \leftarrow \text{Eq. (5.13)}$   
**// Control forward pass**  
10 **for**  $k = t, \dots, T-1$  **do**  
11    $p_{u_k}, p_{x_{k+1}} \leftarrow \text{Eq. (5.10)}$   
**Output:**  $p_{x_0}, p_{x_1}, \dots, p_{x_T}, p_{u_t}, \dots, p_{u_{T-1}}$

---

In the end, the update step,  $p$ , can be computed with a forward recursion on the past indexes, a backward recursion on the future indexes, a coupling equation, a backward recursion on the past indexes, and a forward recursion on the future indexes. Algorithm 4 summarizes those steps. Clearly, the complexity is linear in time. Instead of inverting a matrix of size  $((T+1)n_x + (T-t)n_u)$ , Algorithm 4 only operates with matrices of size  $n_x$  or  $n_u$  and the number of operations is proportional to  $T$ .

### 5.2.4 Line-search and convergence

In the linear quadratic case, Algorithm 4 is equivalent to Whittle's derivations and only one iteration is required to find a solution. However, in the general nonlinear case, several iterations of Newton's step are required. A common approach to guarantee the convergence of the overall iterative procedure is to introduce a line search and a merit function [143]. Given a guess at iteration  $i$  and a direction  $p$ , the next guess is defined by

$$\begin{pmatrix} x_{0:T}^{i+1} \\ u_{t:T-1}^{i+1} \end{pmatrix} = \begin{pmatrix} x_{0:T}^i \\ u_{t:T-1}^i \end{pmatrix} + \alpha_i \begin{pmatrix} p_{x_{0:T}} \\ p_{u_{t:T-1}} \end{pmatrix}, \quad (5.17)$$

where the step length  $\alpha_i$  is chosen in order to decrease the merit function. As advocated by Nocedal et al. [143], a Newton step provides a descent direction for the merit function

$$f_{\mathcal{M}}(x_{0:T}, u_{t:T-1}) = \frac{1}{2} \sum_{j=0}^T \left\| \frac{\partial J}{\partial x_j} \right\|^2 + \frac{1}{2} \sum_{j=t}^{T-1} \left\| \frac{\partial J}{\partial u_j} \right\|^2. \quad (5.18)$$

This merit function can be derived analytically. The exact derivations of each gradient are provided in the Appendix B. By construction, the expected decrease of the direction  $p$  derived by Algorithm 4 is  $p^T \nabla f_{\mathcal{M}} = -\|\nabla J\|_2^2$  and the following convergence guarantees hold.

**Proposition 6.** *Assuming that the norm of the inverse of the Hessian of  $J$  is bounded and that the step length  $\alpha_i$  satisfies the Wolfe conditions for the merit function (5.18), the sequence  $(x_{0:T}^i, u_{t:T-1}^i)_i$  defined by the update rule (5.17) with steps from Algorithm 4 is globally convergent to a stationary point of (5.3). Fur-*

thermore, when the iterate is sufficiently close to the solution, the sequence has quadratic convergence.

*Proof.* This proposition is a direct consequence of the fact that Algorithm 4 yields a Newton step. A detailed proof of the convergence guarantees of the Newton method is provided in [143].  $\square$

One may ask under which conditions the Hessian of  $J$  is non-degenerate. Intuitively, large values of  $\mu$  can make the problem ill-defined. Indeed, if the opposing player can choose large disturbances, then the controller might not be able to compensate. In the linear quadratic case, Whittle [186] studied this maximum value for  $\mu$  that makes the problem ill-defined. Although we do not study this limit value in the nonlinear case, we note that analogously to the linear quadratic case, the algorithm is well defined if  $\Gamma_k$  and  $P_k^{-1} - \mu \bar{\ell}_k^{xx}$  are positive-definite, which is the case when  $\mu$  is small enough.

### 5.2.5 About the cooperative case

So far, only the case,  $\mu > 0$  has been considered, but the case  $\mu < 0$  is also well defined. Indeed, the search for a stationary point of  $J$  can be done for an arbitrary sign of  $\mu$ . However, such a stationary point would now be a way to find a local minimum of  $J$  with respect to the variables  $(x_{0:T}, u_{t:T-1})$ . Interestingly, this scenario can be interpreted as a cooperative scenario between the controller and the opposing player. In fact, the disturbances can be seen as a second controller minimizing the cost,  $\ell$ , and maximizing the likelihood of the disturbances. Clearly, this change of sign does not affect the derivation of the stagewise Newton method. However, it can be noted that in that case, one can directly use the cost  $J$  as a

merit function.

## 5.3 Experiments

A Python implementation of the proposed method is available online <sup>2</sup>. It is based on the Crocoddyl software [121], a state-of-the-art (risk-neutral) DDP solver that provides analytical derivatives of robot dynamics. In this section, two numerical examples illustrate the proposed method.

### 5.3.1 Planar quadrotor

We study a quadrotor moving in a plane aiming to reach the position  $(p_x \ p_y) = (2 \ 0)$  starting at the origin without initial velocity. The state is

$$x = (p_x \ p_y \ \theta \ \dot{p}_x \ \dot{p}_y \ \dot{\theta})^T \quad (5.19)$$

where  $\theta$  is the orientation of the quadrotor. The system dynamics is

$$\begin{aligned} m\ddot{p}_x &= -(u_1 + u_2) \sin(\theta), \\ m\ddot{p}_y &= (u_1 + u_2) \cos(\theta) - mg, \\ J\ddot{\theta} &= r(u_1 - u_2), \end{aligned} \quad (5.20)$$

where the control input  $u = (u_1 \ u_2)^T \in \mathbb{R}^2$  represents the force at each rotor and  $g$  is gravitational acceleration. An exponential cost models the presence of an

---

<sup>2</sup>[https://github.com/machines-in-motion/dynamic\\_game\\_optimizer](https://github.com/machines-in-motion/dynamic_game_optimizer)

obstacle

$$\begin{aligned}
\ell_k(x_k, u_k) &= 0.3 \exp \left( -10(p_{x_k} - 1)^2 - 0.5(p_{y_k} + 0.1)^2 \right) \\
&\quad + 0.005 \|u_k - \bar{u}\|_2^2 + 0.05(x_k - x_\star)^T L(x_k - x_\star) \\
\ell_T(x_T) &= (x_T - x_\star)^T L(x_T - x_\star),
\end{aligned} \tag{5.21}$$

where  $\bar{u} = \frac{mg}{2}(1 \ 1)^T$ ,  $x_\star = (2 \ 0 \ 0 \ 0 \ 0 \ 0)^T$ , and  $L = \text{diag}(100 \ 100 \ 100 \ 1 \ 1 \ 1)$ . Only the position,  $p_x$ ,  $p_y$ , and orientation  $\theta$  are part of the measurements. The integration and discretization of the model is done with Runge Kutta 4 with a time step of 0.05 and the total horizon,  $T$ , is 60. Furthermore,  $P_0 = Q = 10^{-5}I_6$ ,  $R = 10^{-4} \text{diag}(1 \ 1 \ 0.01)$  and  $\hat{x}_0$  is the origin. A backtracking line-search is used – a step is accepted if  $f_{\mathcal{M}}^{i+1} \leq f_{\mathcal{M}}^i + \alpha_i c p_i^T f_{\mathcal{M}}^i$  where  $f_{\mathcal{M}}^i = f_{\mathcal{M}}(x_{0:T}^i, u_{t:T-1}^i)$  with  $c = \frac{1}{4}$ . The iterative process is stopped when the decrease in the merit function is lower than  $10^{-12}$ .

Figure 5.1 illustrates the solution obtained by the solver for different values of  $\mu$ . The neutral controller, the limit when  $\mu$  tends to zero, aiming at minimizing the cost without accounting for disturbances, is solved using DDP. Interestingly, when  $\mu$  is positive (the non-cooperative case), the opposing player chooses disturbances that will push the quadrotor towards the obstacle, and when  $\mu$  is negative (the cooperative case), the opposing player chooses disturbances that will push the quadrotor away from the obstacle. For this experiment, we found that the value of  $\mu$  for which  $\Gamma_k$  and  $P_k^{-1} - \mu \bar{\ell}_k^{xx}$  are no longer positive-definite matrices was around 20.

Next, we illustrate the risk-sensitive behavior of the resulting controller in closed loop in a simulation with disturbances following Gaussian distributions:

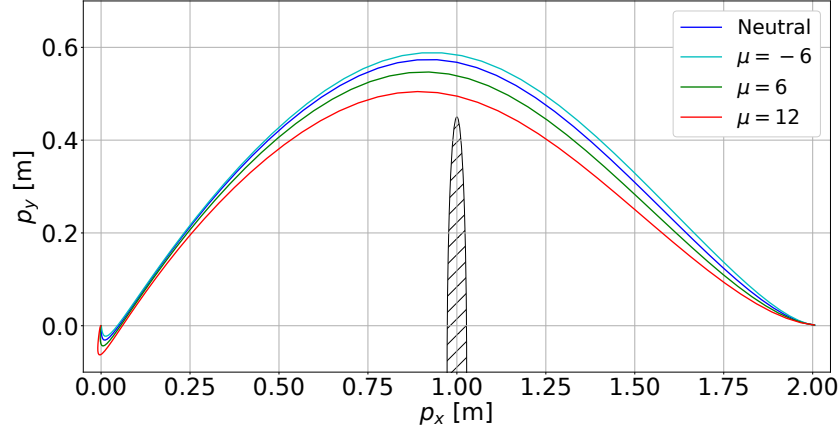


Figure 5.1: Initial plan for different values of  $\mu$ . The larger  $\mu$  the more the controller plans to be pushed against the obstacle.

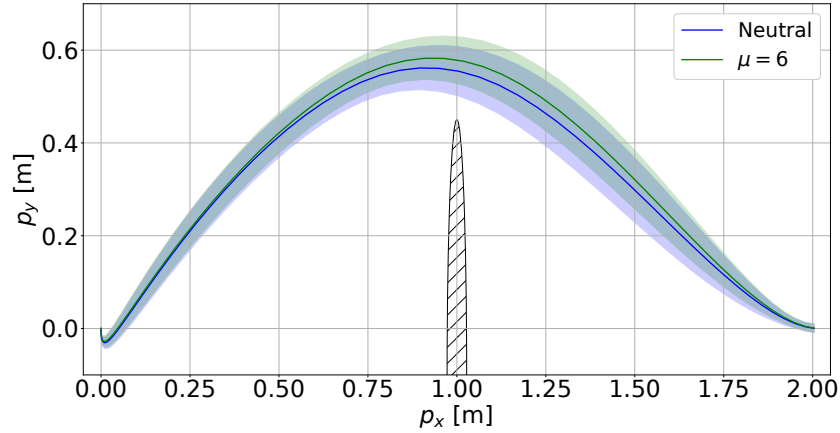


Figure 5.2: Average trajectory. Compared the neutral controller, the dynamic game controller ( $\mu = 6$ ) exhibits a risk sensitive behavior as it remains further from the high cost area representing the obstacle.

$x_0 \sim \mathcal{N}(\hat{x}_0, P_0)$ ,  $w_k \sim \mathcal{N}(0, Q)$  and  $\gamma_k \sim \mathcal{N}(0, R)$ . We set  $\mu = 6$  and at each time step of the simulation, Algorithm 4 is run and  $u_t$  is applied to the system. Additionally, we compare to the neutral controller – an iterative Kalman smoother is used for the filtering and the controller uses DDP with the last state estimate from the smoother as an initial condition. Figure 5.2 depicts the average and standard deviation over one thousand simulations. We can see that, in this MPC

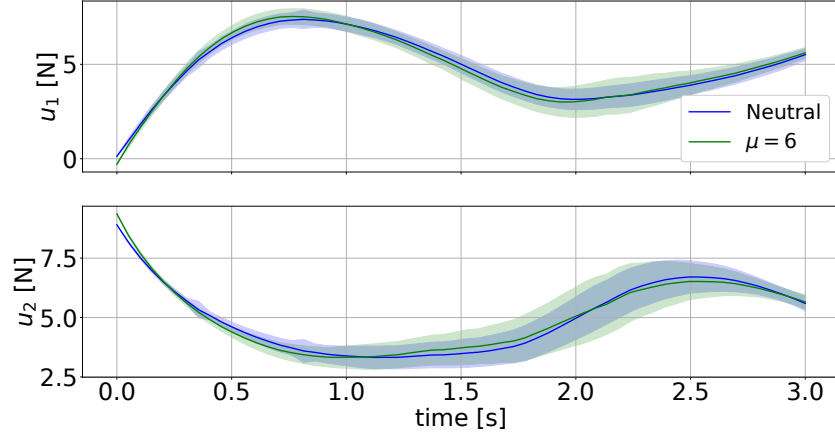


Figure 5.3: Average control trajectories. Compared the neutral controller, the dynamic game controller ( $\mu = 6$ ) has a larger standard deviation.

scheme, the dynamic game controller maintains a larger distance from the obstacle, resulting in safer behavior. Figure 5.3 shows the distribution of control trajectories. Interestingly, the dynamic game controller has a larger standard deviation.

### 5.3.2 An industrial robot

In this example, we consider the 7-DoF torque-controlled KUKA LWR iiwa R820 14. The dynamics of the robot are provided by Pinocchio [32]. The 14-dimensional state is composed of the joint positions and velocities. We consider the following prior on the initial condition  $\hat{x}_0 = \begin{pmatrix} 0.1 & 0.7 & 0. & 0.7 & -0.5 & 1.5 & 0. \end{pmatrix}^T$ . The control input is a 7-dimensional vector of the torque applied on each joint. The goal is to move the end effector to a desired position,  $p_{\text{target}} = \begin{pmatrix} -0.4 & 0.3 & 0.7 \end{pmatrix}$

with the following cost:

$$\begin{aligned}
\ell_k(x_k, u_k) &= 10^{-3} \|x_k - \hat{x}_0\|_2^2 + 10^{-6} \|u_k - \bar{u}(x_k)\|_2^2 \\
&\quad + 10^{-1} \|p_{\text{target}} - \bar{p}(x_k)\|_2^2 \\
\ell_T(x_T) &= \|p_{\text{target}} - \bar{p}(x_k)\|_2^2 + 10^{-3} \|x_k - \hat{x}_0\|_2^2,
\end{aligned} \tag{5.22}$$

where  $\bar{u}(x_k)$  is the gravity compensation and  $\bar{p}(x_k)$  the position of the end-effector. For the measurement model, we assume that only the joints' positions are measured. The initial control inputs  $u_0, \dots, u_{t-1}$  are generated with DDP. Given those  $t$  initial control inputs,  $t$  measurements are generated according to an undisturbed trajectory. More precisely, for  $1 \leq k \leq t$ , the observations are defined by  $y_k = h_k(f^{(k)}(\hat{x}_0, u_{1:k-1}))$  where  $f^{(k)}(\hat{x}_0, u_{1:k-1})$  denotes the state value at the  $k^{\text{th}}$  step integrated from the initial guess  $\hat{x}_0$ . The horizon,  $T$ , is 100 and  $t$  is 5 while the sensitivity parameter is set to  $\mu = 1.2$ . Here  $P_0 = Q = 0.01 \times I_{14}$  and  $R = 0.5 \times I_7$ . For this experiment, the second-order derivatives of the dynamics were approximated to be null as the former are not provided by most standard rigid body dynamics libraries such as Pinocchio [32]. Note that this is a common practice for state-of-the-art optimal control algorithms in robotics [121]. Our solver converges nevertheless, suggesting that this approximation might be used to scale to a large number of degrees of freedom for real-time computations (e.g. MPC). In Figure 5.4, we plot the solution of the dynamics game solver compared to DDP in the end-effector space; the shaded grey area represents the estimation part of the solution. We can see that the dynamic game controller plans that the disturbances will slow down the reaching task.



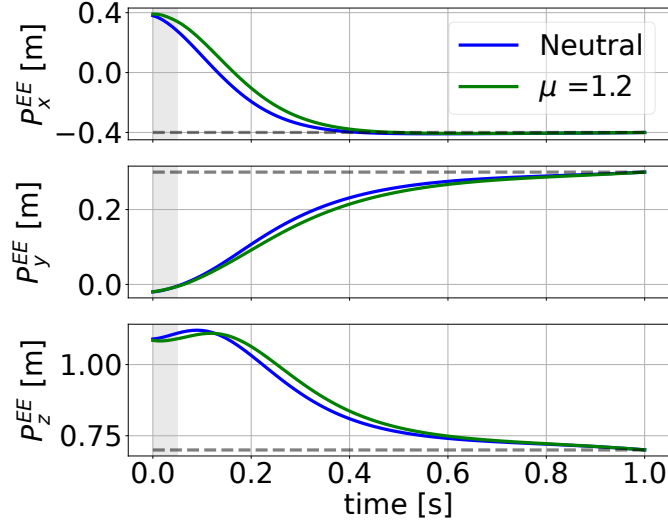


Figure 5.4: End-effector position vs time. The dashed lines represent the target.

## 5.4 Conclusion

In this Chapter, we introduced an iterative solver to find local Nash equilibrium of dynamic game with imperfect state measurements. The proposed algorithm is proven to be equivalent to Newton's method and benefits from its convergence properties while scaling linearly with the time horizon.

Solving simultaneously over a horizon of past measurements and future control allows to reason about the perception uncertainty. However, in practice, estimation does not always require a large history of measurement. In fact, the Kalman filter, the most widely used filter, only requires one past measurement to provide an estimate. In the next Chapter, we will show that under the same assumptions as the established Kalman filter, we can derive a filter reasoning about risk.

## Chapter 6

# Risk-Sensitive Extended Kalman Filter

In this Chapter<sup>1</sup>, we leverage our previous theoretical results on dynamic game control with imperfect state observation (Chapter 5) to introduce the Risk-Sensitive Extended Kalman Filter (RS-EKF), a novel filter that enables online risk-sensitive output feedback MPC. The RS-EKF computes state estimates robust to measurement uncertainty while taking into account the value function provided by the controller, i.e., the estimator tailors risk reduction to the control objectives. This, in turn, enables automatic modification of robot decisions to be cautious in times of high environmental perturbation. Furthermore, RS-EKF has a similar computational cost to an EKF, allowing real-time deployment. To demonstrate the ability of the filter, we use it together with a DDP-based online non-linear controller to perform risk-sensitive output-feedback MPC on various simulated

---

<sup>1</sup>This Chapter is adapted from the following publication: **A. Jordana** et al. "Risk-Sensitive Extended Kalman Filter." IEEE International Conference on Robotics and Automation (ICRA) 2024.

robots, such as a quadrotor subjected to arbitrary changes in its mass, and a KuKa robot facing unforeseen environmental disturbances. Finally, we test the filter on a real quadruped robot Solo12 [75] to perform an external force estimation and balancing task. These experiments demonstrate that the robots are more robust to perturbations with the RS-EKF algorithm than a classical EKF. To the best of our knowledge, this is the first time that a non-linear risk-sensitive output-feedback MPC controller has been deployed on a robot.

## 6.1 Background: Extended Kalman Filter

We now discuss the structure of the EKF necessary to derive our filter. The EKF is usually derived by computing the probability a posteriori of the state given measurements, using the linearized dynamics and a Gaussian noise assumption [171]. However, the EKF can also be derived from an optimization point of view [18]. More precisely, the EKF can be seen as a Gauss-Newton step around a well-chosen point on the log-likelihood of the maximum a posteriori probability (MAP), i.e.  $\log(p(x_t, x_{t-1}|y_t))$  [18]. Assuming disturbances follow Gaussian distributions,  $\gamma_t \sim \mathcal{N}(0, R_t)$ ,  $\omega_t \sim \mathcal{N}(0, Q_t)$ , the MAP can be written as:

$$\begin{aligned} \max_{x_t, x_{t-1}} \quad & -(y_t - h_t(x_t))^T R_t^{-1} (y_t - h_t(x_t)) \\ & -(x_t - f_{t-1}(x_{t-1}, u_{t-1}))^T Q_t^{-1} (x_t - f_{t-1}(x_{t-1}, u_{t-1})) \\ & -(x_{t-1} - \hat{x}_{t-1})^T P_{t-1}^{-1} (x_{t-1} - \hat{x}_{t-1}) \end{aligned} \quad (6.1)$$

where  $\hat{x}_{t-1}$  is the prior knowledge on the past state and  $P_{t-1}$  its associated covariance matrix. As shown in [18], a Gauss-Newton step around  $\hat{x}_{t-1}$  and  $\bar{x}_t = f_{t-1}(\hat{x}_{t-1}, u_{t-1})$

on (6.1) leads to the well-known recursion [171]:

$$\bar{P}_t = Q_t + F_{t-1}P_{t-1}F_{t-1}^T \quad (6.2)$$

$$K_t = \bar{P}_t H_t^T (R_t + H_t \bar{P}_t H_t^T)^{-1} \quad (6.3)$$

$$P_t = (I - K_t H_t) \bar{P}_t \quad (6.4)$$

$$\hat{\mu}_t = K_t (y_t - h_t(\bar{x}_t)) \quad (6.5)$$

$$\hat{x}_t = \bar{x}_t + \hat{\mu}_t \quad (6.6)$$

where  $F_{t-1} = \partial_x f_{t-1}(\hat{x}_{t-1}, u_{t-1})$  and  $H_t = \partial_x h_t(\bar{x}_t)$ . Here,  $\hat{x}_t$  is the most likely estimate and  $P_t$  is the covariance uncertainty. In practice, at the next time step,  $\hat{x}_t$  is used as the prior knowledge on the state.

We notice the similar structure of the costs of Problem (5.1) and Eq. (6.1), except that the EKF only uses one measurement and does not include the control cost  $\ell_j$ . Hence, Eq. (5.1) can be seen as a maximization of the estimation log-likelihood up to some cost terms. We will leverage this similarity to derive a risk-sensitive version of the EKF. More precisely, we will add cost-dependent terms in the maximization (6.1) in order for the filter to adapt to the control objective.

## 6.2 Risk-sensitive filter

We now introduce RS-EKF, which builds on the dynamic game defined in Equation (5.1). First, we modify the game to account for typical assumptions made for MPC while keeping the adversarial part that provides the risk-sensitive behavior. Then, we show how the solution can be computed with a Gauss-Newton step similar to the EKF, leading to an algorithm of similar complexity. Our formulation leads

to a modified update in the filter of which the standard EKF can be seen as a limit case.

First, as for the EKF, we consider a history of measurements of length one. Furthermore, we disregard future uncertainties and assume deterministic dynamic equations for the future as is done in classical MPC formulations. Indeed, we expect that the high-frequency re-planning will compensate for model discrepancies. In the end, the problem is supposed to be adversarial only with respect to the uncertainties related to the estimation. This can be written as:

$$\begin{aligned} \min_{u_{t:t+H-1}} \max_{w_t} \max_{w_{t-1}} \max_{\gamma_t} \mathcal{L}_t(u_t, \dots, u_{H-1}) \\ - \frac{1}{2\mu} (\gamma_t^T R_t^{-1} \gamma_t + w_t^T Q_t^{-1} w_t + w_{t-1}^T P_{t-1}^{-1} w_{t-1}) \end{aligned} \quad (6.7)$$

$$\text{s.t. } x_{t-1} = \hat{x}_{t-1} + w_{t-1}, \quad (6.8a)$$

$$x_t = f_{t-1}(x_{t-1}, u_{t-1}) + w_t, \quad (6.8b)$$

$$y_t = h_t(x_t) + \gamma_t. \quad (6.8c)$$

$$x_{j+1} = f_j(x_j, u_j), \quad t < j < T. \quad (6.8d)$$

As presented in Chapter 5, one of the key features of the dynamic game is that some of the constraints can be removed with an appropriate change of variable. Indeed, we can use the equality constraints of Equations (6.8a), (6.8b) and (6.8c)

to replace the disturbance maximization into a maximization over  $x_{t-1}, x_t$ :

$$\begin{aligned}
& \min_{u_{t:t+H-1}} \max_{x_{t-1}, x_t} \mathcal{L}_t(u_t, \dots, u_{H-1}) \\
& - \frac{1}{2\mu} (y_t - h_t(x_t))^T R_t^{-1} (y_t - h_t(x_t)) \\
& - \frac{1}{2\mu} (x_t - f_{t-1}(x_{t-1}, u_{t-1}))^T Q_t^{-1} (x_t - f_{t-1}(x_{t-1}, u_{t-1})) \\
& - \frac{1}{2\mu} (x_{t-1} - \hat{x}_{t-1})^T P_{t-1}^{-1} (x_{t-1} - \hat{x}_{t-1}) \\
& \text{subject to } x_{j+1} = f_j(x_j, u_j), \quad t < j < T,
\end{aligned} \tag{6.9}$$

By definition of the MAP [171], this can be written:

$$\begin{aligned}
& \min_{u_{t:t+H-1}} \max_{x_{t-1}, x_t} \mathcal{L}_t(u_t, \dots, u_{H-1}) - \frac{1}{\mu} \log(p(x_t, x_{t-1}|y_t)) \\
& \text{subject to } x_{j+1} = f_j(x_j, u_j), \quad t < j < T.
\end{aligned} \tag{6.10}$$

Problem (6.10) is intractable in the general case. However, by taking the concave-convex assumption, the minimization and maximization can be interchanged according to the minimax theorem. Consequently, the problem is equivalent to:

$$\max_{x_{t-1}, x_t} \log(p(x_{t-1}, x_t|y_t)) + \mu V_t(x_t), \tag{6.11}$$

where  $V_t$  is the value function of the OCP:

$$V_t(x_t) = \min_{u_{t:t+H-1}} \mathcal{L}_t(u_t, \dots, u_{H-1}) \tag{6.12}$$

Note that in the simplification from Eq. (5.1) to Eq. (6.7), it is not necessary to disregard future uncertainties as the value function could be the one resulting from

minimax DDP [134]. If  $\mu = 0$ , we will obtain the unbiased estimate of Kalman filtering and the estimate will be independent of the control objective. Otherwise, if  $\mu > 0$ , the term  $\mu V(x_t)$  will bias the estimate towards regions with a higher value function, which in turn will force the controller to be more conservative.

We now take a Gauss-Newton step on the objective of Eq. (6.11) around the prior:  $\hat{x}_{t-1}$  and  $\bar{x}_t = f_{t-1}(\hat{x}_{t-1}, u_{t-1})$ .  $V_t(x_t)$  is independent of  $x_{t-1}$ ; therefore, as shown in the appendix, the maximization over  $x_{t-1}$  can be simplified to:

$$\begin{aligned} \max_{x_t} & -\frac{1}{2}(x_t - \hat{x}_t)^T P_t^{-1}(x_t - \hat{x}_t) \\ & + \mu \frac{1}{2}(x_t - \bar{x}_t)^T V_t^{xx}(x_t - \bar{x}_t) + \mu(x_t - \bar{x}_t)^T v_t^x \end{aligned} \quad (6.13)$$

where  $\hat{x}_t$  and  $P_t$  are defined as in Eq. (6.6) and (6.4). where  $V_t^{xx}$  (respectively  $v_t^x$ ) is the Hessian (respectively the gradient) of the value function.

*Proof.* By taking a quadratic approximation of the value function, the Gauss-Newton step can be written as:

$$\begin{aligned} \max_{x_{t-1}} \max_{x_t} & \mu(x_t - \bar{x}_t)^T V_t^{xx}(x_t - \bar{x}_t) + 2\mu(x_t - \bar{x}_t)^T v_t^x \\ & - (\Delta y - H_t \Delta x_t)^T R_t^{-1}(\Delta y - H_t \Delta x_t) - \Delta x_{t-1}^T P_{t-1}^{-1} \Delta x_{t-1} \\ & - (\Delta x_t - F_{t-1} \Delta x_{t-1})^T Q_t^{-1}(\Delta x_t - F_{t-1} \Delta x_{t-1}) \end{aligned} \quad (6.14)$$

where  $\Delta y = y_t - h(\hat{x}_t)$ ,  $\Delta x_{t-1} = x_{t-1} - \hat{x}_{t-1}$ ,  $\Delta x_t = x_t - \hat{x}_t$ . It can then be found

that  $x_{t-1} = \tilde{Q}^{-1}\tilde{q}$ , where:

$$\begin{aligned}\tilde{Q} &= P_{t-1}^{-1} + F_{t-1}^T Q_t^{-1} F_{t-1} \\ \tilde{q} &= -P_{t-1}^{-1}\hat{x}_{t-1} - F_{t-1}^T Q_t^{-1}(x_t - \hat{x}_t) - F_{t-1}^T Q_t^{-1} F_{t-1} \hat{x}_{t-1}\end{aligned}\tag{6.15}$$

by using the Woodbury lemma [171], it can be shown that:

$$\begin{aligned}\max_{x_t} \quad & \mu \frac{1}{2} (x_t - \bar{x}_t)^T V_t^{xx} (x_t - \bar{x}_t) + \mu (x_t - \bar{x}_t)^T v_t^x \\ & - \frac{1}{2} (\Delta y - H_t \Delta x_t)^T R_t^{-1} (\Delta y - H_t \Delta x_t) - \frac{1}{2} \Delta x_t^T \bar{P}_t^{-1} \Delta x_t\end{aligned}$$

where  $\bar{P}_t$  is defined as in (6.2). Finally, using  $P_t = (H_t^T R_t^{-1} H_t + \bar{P}_t^{-1})^{-1} = (I - K_t H_t) \bar{P}_t$ , we can show that:

$$\begin{aligned}\max_{x_t} \quad & \mu \frac{1}{2} (x_t - \bar{x}_t)^T V_t^{xx} (x_t - \bar{x}_t) + \mu (x_t - \bar{x}_t)^T v_t^x \\ & - \frac{1}{2} (x_t - \hat{x}_t - \hat{\mu}_t)^T P_t^{-1} (x_t - \hat{x}_t - \hat{\mu}_t)\end{aligned}\tag{6.16}$$

where  $P_t, \hat{\mu}_t$  are defined as in (6.4), (6.5).

□

Those are typically provided by optimal control algorithms such as DDP. In the end, the solution on the maximization over  $x_t$  is:

$$\hat{x}_t^{RS} = \bar{x}_t + (I - \mu P_t V_t^{xx})^{-1} (\hat{\mu}_t + \mu P_t v_t^x)\tag{6.17}$$

Interestingly, if  $\mu = 0$ , we recover the EKF. This was to be expected as, when  $\mu$  tends to zero, the solution of problem (5.1) is exactly the solution of the neutral



case where estimation and control are solved independently [186]. Otherwise, the estimate is shifted towards regions with higher cost values. Importantly, the magnitude of the shift depends on  $P_t$  the covariance matrix of the estimation. Note that  $\mu$  cannot be arbitrarily large as  $(I - \mu P_{t+1} V_{t+1}^{xx})$  needs to be positive definite. Larger values of  $\mu$  would make the min-max problem defined in Eq. (5.1) ill-posed. More details on this limit value can be found in [186]. In the end, the estimate is shifted towards  $P_t v_t^x$ , i.e. towards a region with a larger cost function, and the magnitude of this shift is increased in the direction corresponding to large eigenvalues of  $P_t V_t^{xx}$ .

We obtained the solution to the maximization problem (6.10). Therefore, the cost function can now be minimized with respect to the control inputs by taking  $\hat{x}_t^{RS}$  as an initial condition of the optimal control problem, which can be solved, for example, with DDP.

Algorithm 5 summarizes the estimation procedure. It can then be used to do output-feedback MPC efficiently. At each time step, given a measurement, past control input, and a quadratic approximation of the value function, a risk-sensitive estimate can be computed. This estimate is then used to minimize the cost function for MPC and the first control input is applied to the real system. Lastly, the quadratic approximation of the value function at  $t + 1$  is saved as it will be used at the next estimation step.

## 6.3 Experiments

This section presents simulation and real robot experiments to illustrate the benefits of the proposed algorithm and demonstrate its applicability to real problems. We study three test problems where we deploy the RS-EKF inside an MPC loop: a

**Algorithm 5:** Risk Sensitive EKF

---

**Input:**  $\hat{x}_{t-1}, u_{t-1}, y_t, P_{t-1}, Q_t, R_t, V_t, v_t$   
 /\* Predict \*/  
 1  $\bar{P}_t \leftarrow Q_t + F_{t-1}P_{t-1}F_{t-1}^T$   
 2  $\bar{x}_t \leftarrow f(\hat{x}_{t-1}, u_{t-1})$   
 /\* Classical Update \*/  
 3  $K_t \leftarrow \bar{P}_t H_t^T (R_t + H_t \bar{P}_t H_t^T)^{-1}$   
 4  $P_t \leftarrow (I - K_t H_t) \bar{P}_t$   
 5  $\hat{\mu}_t \leftarrow K_t (y_t - h_t(\bar{x}_t))$   
 /\* Value function bias \*/  
 6  $p_{x_t} \leftarrow (I - \mu P_t V_t^x)^{-1} (\hat{\mu}_t + \mu P_t v_t^x)$   
 7  $\hat{x}_t^{RS} \leftarrow \bar{x}_t + p_{x_t}$   
**Output:**  $\hat{x}_t^{RS}, P_t$

---

planar quadrotor with a load estimation task where we demonstrate qualitatively that the RS-EKF can bring conservatism appropriately in phases of high uncertainty, a push-recovery experiment on a 7-dof industrial manipulator on which we perform a quantitative study and lastly, an external force estimation task on a real quadruped robot to showcase the viability of the method on real systems. In each experiment, we use the DDP implementation provided by Crocoddyl [120] to solve the optimal control problem given the filter estimate. All the code is available online<sup>2</sup>.

### 6.3.1 Planar quadrotor

In this first scenario, we consider a planar quadrotor executing a load-carrying task. The goal is to move the quadrotor from position  $(p_x, p_y) = (0, 0)$  to position  $(1, 0)$  while carrying a load during the first half of the itinerary. The robot mass is 2 kg and the mass of the load, which is unknown a priori, is 3 kg. The system

<sup>2</sup><https://github.com/machines-in-motion/risk-sensitive-EKF>

dynamics is:

$$\begin{aligned}
m\ddot{p}_x &= -(u_1 + u_2) \sin(\theta), \\
m\ddot{p}_y &= (u_1 + u_2) \cos(\theta) - mg, \\
md\ddot{\theta} &= r(u_1 - u_2),
\end{aligned} \tag{6.18}$$

where  $m$  is the mass of the robot,  $d$  the distance between the rotors,  $\theta$  the orientation of the quadrotor.  $u_1$  and  $u_2$  are the control inputs representing the force applied at each rotor.

In this experiment, we want to estimate online the mass parameter that changes in the middle of the flying phase. As it is standard in parameter identification [179], we augment the system's state with the unknown parameter and let it be estimated recursively by the filter (EKF or RS-EKF). The state of the system is thus:  $x = \begin{pmatrix} p_x & p_y & \theta & \dot{p}_x & \dot{p}_y & \dot{\theta} & m \end{pmatrix}^T$  and it is assumed that  $\dot{m} = 0$  up to some random Gaussian noise. The dynamics are integrated with an Euler scheme and a time step of 0.05. We consider that  $P_0 = 10^{-4}I_7$ ,  $R = 10^{-4}I_3$ , and  $Q$  is a  $7 \times 7$  diagonal matrix where all terms are equal to  $10^{-4}$  except the last one that we set to 2 to represent the uncertainty in the changes of the load. Lastly, we set  $\mu = 4 \times 10^{-3}$ . A stationary cost function of the following form is considered:

$$\begin{aligned}
\ell(x, u) &= \alpha_1 (\|p_x - p_x^{des}\|^2 + \|p_y - p_y^{des}\|^2) + \alpha_2 \|\theta\|^2 \\
&+ \alpha_3 (\|\dot{p}_x\|^2 + \|\dot{p}_y\|^2 + \|\dot{\theta}\|^2) + \alpha_4 \|u - \bar{u}\|^2
\end{aligned} \tag{6.19}$$

where  $\bar{u} = \begin{pmatrix} \frac{mg}{2}, \frac{mg}{2} \end{pmatrix}^T$  and where:  $\alpha_1 = 100$ ,  $\alpha_2 = 10$ ,  $\alpha_3 = 0.01$  and  $\alpha_4 = 0.1$ . We consider a horizon of 20 nodes and re-plan at each new measurement, i.e. every

0.05s. Furthermore, we only measure:  $y = \begin{pmatrix} p_x & p_y & \theta \end{pmatrix}^T$  to illustrate the estimator capabilities. We simulate 4s with both output-feedback MPC controllers: the first one relying on the standard EKF estimate and the second one relying on the RS-EKF estimate.

Figure 6.1 shows the real mass variation and the estimates of both methods. It can be seen that the RS-EKF is more reactive when the load is added or dropped. The RS-EKF estimate spikes in the phases of uncertainty, which adds some robustness. Those spikes can be explained by the fact that the uncertainty increases on the components of the state that are important in the cost function. In other words, some of the eigenvalues of  $P_t V_t^{xx}$  become larger in the phases of uncertainty, which augments the shift on the estimate as shown in Equation (6.17).

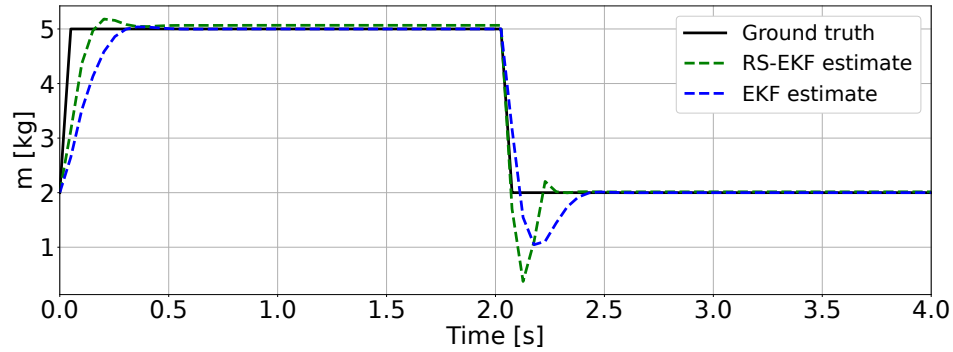


Figure 6.1: Mass estimation for both EKF and RS-EKF.

Figure 6.2 depicts the trajectory in space for both methods. It can be seen that the controller relying on the risk-sensitive estimate is more reactive. To evaluate both controllers, we compute the average Mean Square Error (MSE) relative to the reference trajectory. The MSE of RS-EKF is 0.0011, and the one of EKF is 0.0024. Hence, in that situation, the risk sensitivity brings a 54% improvement

in tracking. Furthermore, the average cost along the trajectory is equal to 0.0569 for the RS-EKF-based controller, while it is equal to 0.0880 for the EKF-based controller, yielding a 35% improvement. This illustrates how a filter informed of the cost objective can improve the controller's performance.

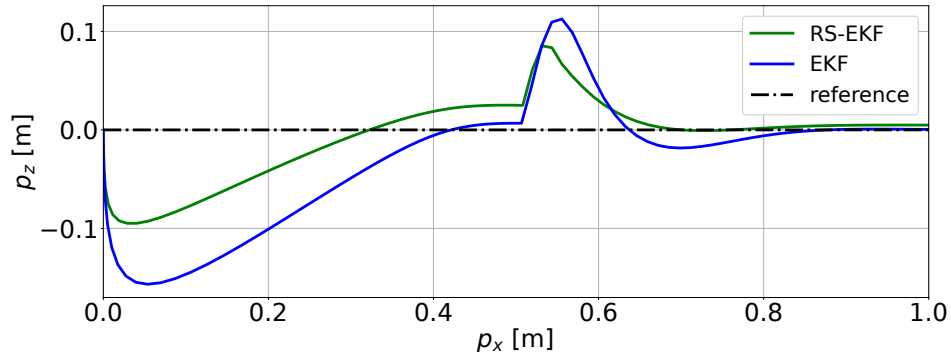


Figure 6.2: Quadrotor trajectory for both the EKF-MPC and the RS-EKF-MPC.

As it can be seen in both Figures 6.1 and 6.2, RS-EKF introduces a steady-state error. This is because the estimation uncertainty never goes to zero due to the nonzero diagonal term of the matrices  $Q$  and  $R$ . Intuitively, it makes sense that the controller that plans for the worst is slightly sub-optimal in the phase with no environment perturbation uncertainty. The risk-sensitive filter demonstrates its benefits when there are perturbations in the environment as this corresponds to phases where the most likely estimate might be far from reality. In the end, this example illustrates the advantages of having a risk-sensitive controller which is conservative only in phases with large environmental perturbations.

### 6.3.2 Kuka robot

In this example, we consider the 7-DoF torque-controlled KUKA LWR iiwa R820 14. The 14-dimensional state is composed of the joint positions and velocities.

The control input is a 7-dimensional vector of the torque applied on each joint. The continuous dynamics and its analytical derivatives are provided by Pinocchio [34]. The goal is to track a reference trajectory with the end effector. To do so, we consider the following cost:

$$\begin{aligned}\ell_k(x_k, u_k) &= 10^{-2}\|x_k - \bar{x}\|_2^2 + 10^{-4}\|u_k - \bar{u}(x_k)\|_2^2 \\ &\quad + 10^2\|p_k^{\text{target}} - \bar{p}(x_k)\|_2^2 \\ \ell_T(x_T) &= 10^2\|p_T^{\text{target}} - \bar{p}(x_T)\|_2^2 + 10^{-2}\|x_T - \bar{x}\|_2^2,\end{aligned}\tag{6.20}$$

$\bar{x}$ , the initial state, is used for the state regularization and is the concatenation of the initial configuration  $\begin{pmatrix} 0.1 & 0.7 & 0 & -1 & -0.5 & 1.5 & 0 \end{pmatrix}^T$  and a 7-dimensional zero vector corresponding to the velocity.  $\bar{u}(x_k)$  is the gravity compensation term given by the rigid body dynamics.  $\bar{p}(x_k)$  is the end-effector position obtained through forward kinematics.  $p_k^{\text{target}}$  is defined such that the end effector follows trajectories forming a circle in the  $xy$  plane. We use a horizon of 20 collocation points with an integration step of 0.05s and re-plan at 500 Hz.

In this experiment, we aim to showcase that the risk-sensitive filter can bring conservatism on phases with large perturbations from the environment, which we simulate with large forces applied on the end effector. For the measurement model, we assume that all the states are observed with high accuracy; therefore, we set  $R = P_0 = 10^{-6}I_{14}$ . However, to model the disturbances in the dynamics, we set  $Q = 10^{-1}I_{14}$ . Finally, we consider  $\mu = 7.5 \times 10^4$ .

Figure 6.3 depicts the end effector trajectory for both controllers and their respective estimates. From time 1s to 2s, an external force of norm 80 is applied on the end-effector on the  $x$  and  $z$  direction. Both controllers are pushed away

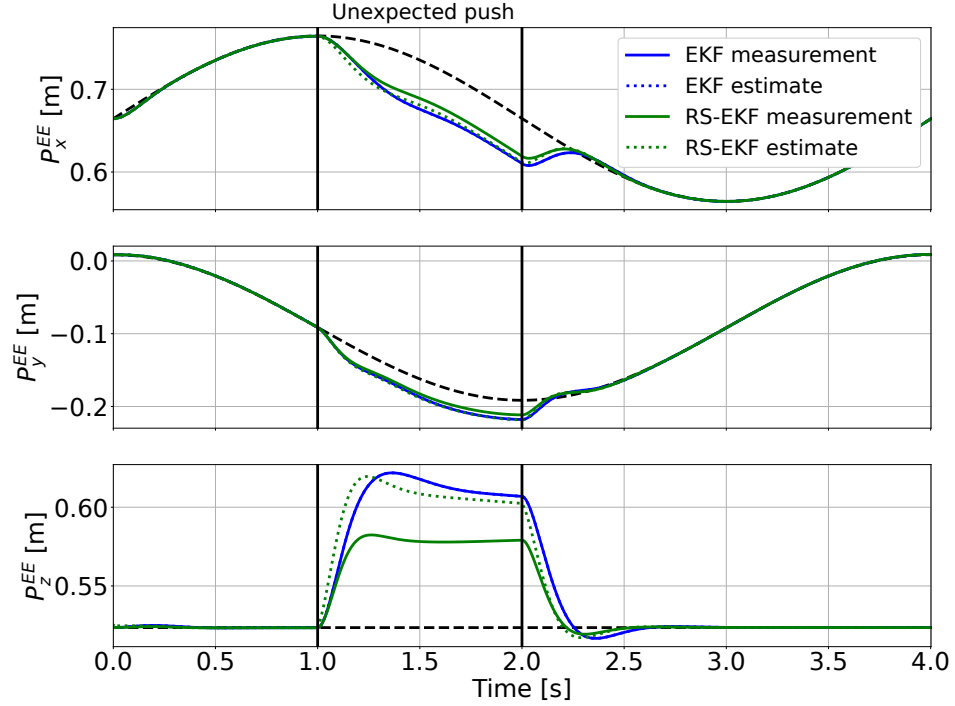


Figure 6.3: End effector trajectory on a tracking task for both the EKF-MPC and the RS-EKF-MPC. An unexpected force is applied between 1s and 2s.

from the reference trajectory. However, the risk-sensitive estimate overestimates the distance between the reference and the end-effector. Consequently, the robot is more aggressive in its response, and the end-effector remains closer to the reference. This illustrates how taking a pessimistic estimate with respect to the cost improves the running cost. Note that both estimates are originally state estimates but are mapped to end effector space through the forward kinematics to draw Figure 6.3. The fact that those estimates are pessimistic with respect to the task originally defined in end effector space illustrates well how the method can handle nonlinear dynamics and cost functions.

To validate the consistency of the filter, 10,000 experiments are performed with random external forces. The timing and direction of the forces are uniformly sampled while the duration is fixed to 1s and the norm is fixed to 80. Figure 6.4

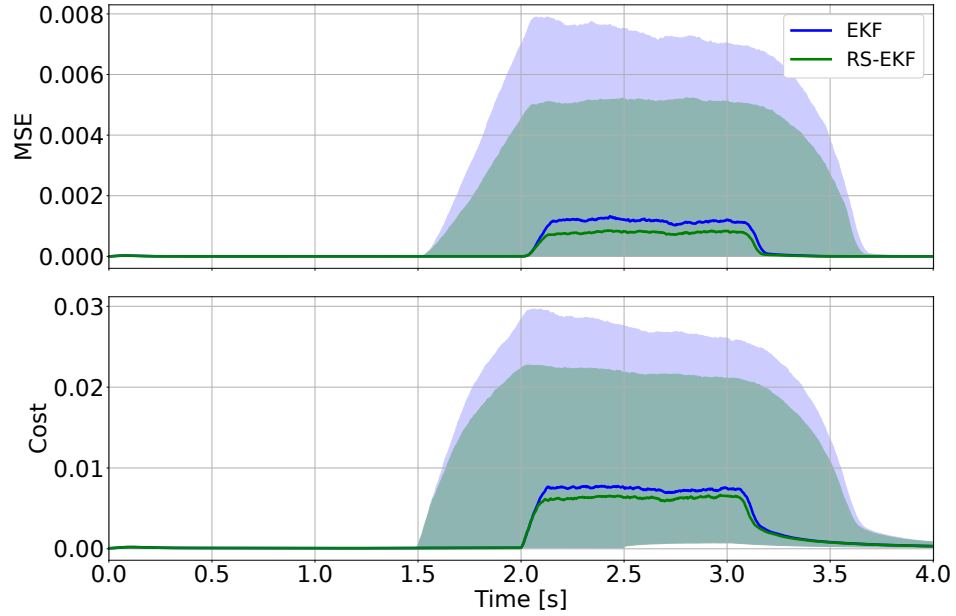


Figure 6.4: Median MSE on the tracking over 10,000 experiments with random external disturbances. The envelope represents the 25th and 75th percentiles.

shows the median end effector error trajectory. On average, we obtain a 32% improvement in the MSE. Furthermore, the mean cost is 22% lower with the risk-sensitive filter.

### 6.3.3 Load estimation on a quadruped robot

In this experiment, we deploy the RS-EKF on a real 12-degree-of-freedom, torque-controlled quadruped robot - Solo12 [75]. We demonstrate the superior performance of the RS-EKF in estimating wrenches applied on Solo12 while it is standing. We generate the standing behavior on Solo12 using a non-linear MPC scheme. At each control cycle, we minimize a cost function using a centroidal model to compute the optimal forces and trajectory that keep the robot's base at a desired height and orientation. Additionally, we use an augmented state to



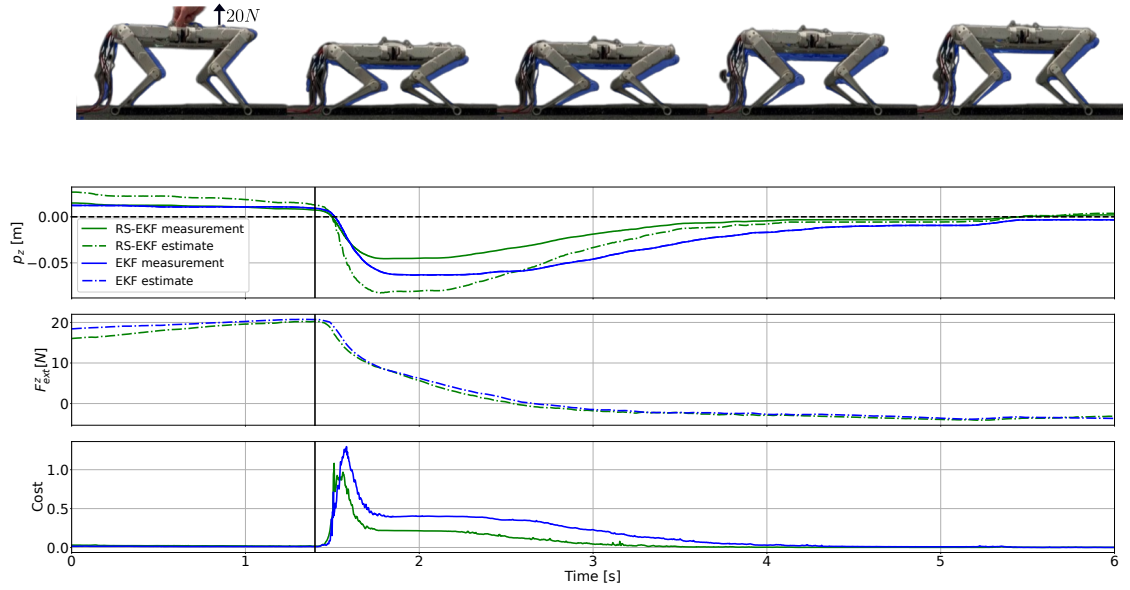


Figure 6.5: Comparison of both methods after an external force of 20 N is applied by pulling the robot vertically. The vertical line indicates the moment when the robot is dropped. The top sub-figure overlays the trajectory of both the EKF in blue and RS-EKF in solid. The RS-EKF-based controller is more reactive to the perturbation and returns to the reference sooner.

estimate external forces-torques applied to the robot [154]:

$$\dot{c} = \frac{1}{m}l \quad (6.21)$$

$$\dot{l} = mg + \sum_{i=1}^{M_c} F_i + F_{\text{ext}} \quad (6.22)$$

$$\dot{k} = \sum_{i=1}^{M_c} (p_i - c) \times F_i + \tau_{\text{ext}} \quad (6.23)$$

$$\dot{F}_{\text{ext}} = 0, \quad \dot{\tau}_{\text{ext}} = 0 \quad (6.24)$$

where  $m$  denotes the mass,  $M_c$  the number of end effectors in contact. Each  $p_i$  represents a contact location. Here, the state is  $x = \begin{pmatrix} c & l & k & F_{\text{ext}} & \tau_{\text{ext}} \end{pmatrix}^T$  which includes the Center of Mass ( $c$ ), linear momentum ( $l$ ), Angular Momentum ( $k$ ) and External Force-torques ( $F_{\text{ext}}, \tau_{\text{ext}}$ ). The measurement is  $y = \begin{pmatrix} c & l & k \end{pmatrix}^T$  up to some noise. A motion capture system measures the base position, velocity, and orientation, and an IMU gives the orientation velocity. Joint encodings are provided, and their velocities are derived with finite differences. Then, given  $q, \dot{q}$ , we can compute  $c, l, k$  with Pinocchio [32, 34] and can then be used as a measurement by the centroidal-based filter.

The control input,  $u = \begin{pmatrix} F_1 & \dots & F_{M_c} \end{pmatrix}^T$ , is a  $M_c \times 3$  dimensional vector, representing the force applied at each end effector. For this experiment, the robot is standing; therefore,  $M_c = 4$ . The cost function for the OCP is:

$$\begin{aligned} \ell_t(x, u) &= (x - x^*)^T H_x (x - x^*) + (u - u^*)^T H_u (u - u^*) \\ &\quad + 10^5 \sum_{i=1}^{M_c} \ell_{\text{barrier}}(u_{3i}) \\ \ell_T(x) &= (x - x^*)^T H_x (x - x^*) \end{aligned} \quad (6.25)$$

where  $H_x = \text{BlockDiag}(10^2 I_3, 10 I_6)$  and  $H_u$  is a diagonal matrix where the diagonal terms are made of  $M_c$  times the following sequence  $(10^{-4}, 10^{-4}, 10^{-6})$ . Lastly,

$$\ell_{\text{barrier}}(u) = \begin{cases} u^2 & \text{if } u < 0 \\ 0 & \text{if } 0 < u < 10 \\ (u - 10)^2 & \text{if } u > 10 \end{cases} \quad (6.26)$$

Here,  $x^*$  is designed to keep a constant height above the ground and to keep the base horizontal,  $u^*$  is the gravity compensation. The reference desired angular momentum for the OCP is adapted to bring the base back to a horizontal position. We do this by computing  $k^* = \frac{1}{T} \log_3(R_t R_{\text{des}}^T)$ , where  $R_t$  is the current base rotation matrix,  $R_{\text{des}}$  the rotation matrix corresponding to the quaternion  $q_{\text{des}} = [0, 0, 0, 1]$  and  $T$  the horizon length. The  $\log_3$  is a mapping from  $SE(3)$  to  $\mathfrak{se}(3)$ . The nominal angular momentum aims to bring the base back to the desired orientation over the time horizon [129]. The  $\ell_{\text{barrier}}$  is a quadratic barrier function that creates a soft constraint on the maximum forces the robot can apply on the ground.

We solve this OCP at 100 Hz using Crocodyl [120] and then track the desired forces using a task space impedance dynamics QP [81] that we solve at 1 kHz using [10].

$$\begin{aligned} \min_{f, \tau, a} \quad & \frac{1}{2} \|f - F\|^2 \\ \text{subject to} \quad & Ma + g = J^T f + S^T \tau + S^T f_{\text{friction}} \\ & Ja = -\dot{J}\dot{q}, \end{aligned} \quad (6.27)$$

where  $J$  is the robot contact Jacobian,  $M$  is the mass matrix,  $S$  is the selection matrix that projects on the actuated joints, and  $g$  is the robot gravity vector at the

current time step. The static friction in each joint was estimated independently and is approximately equal to 0.07. However, to keep a continuous model, we consider  $f_{\text{friction}} = -0.07 \frac{2}{\pi} \arctan(2S\dot{q})$ . The two constraints ensure dynamics satisfaction and model that the end effectors do not move. We update our state estimate (including the external force torque) using the RS-EKF at 200 Hz with  $\mu = 6$ . We also use the EKF in place of the RS-EKF to compare the performance of a risk-sensitive filter while keeping the update frequency the same. For both filters, we consider the following parameters:  $P_0 = Q = \text{BlockDiag}(10^{-3}I_6, 10^{-4}I_3, 10^{-1}I_3, 10^{-2}I_3)$   $R = \text{BlockDiag}(10^{-4}I_3, 10^{-2}I_3, 10^{-4}I_3)$ .

The first experiment ran on the robot is shown in Fig. 6.5. Here, the base of Solo12 is pulled up (in the  $z$  direction) until an external estimate of 20N is computed by both the filters (shown in bottom sub-figure 6.5 at time 1.4s with a vertical line). The base is then released to let Solo12 recover and bring its base back to the nominal desired height. The top sub-figure in Fig 6.5 shows an overlaid pictorial comparison of Solo12 when the two filters (RS-EKF and EKF) are used on the robot. Each frame corresponds to the state of Solo12 at the same time. The opaque Solo12 image shows the state with RS-EKF, and the blue one shows the EKF. This experiment shows that the RS-EKF helps the OCP to react quicker and bring the base to the nominal location sooner. This happens because the RS-EKF underestimates the base height in  $z$  as compared to EKF, which makes the OCP generate higher ground reaction forces to bring the base up sooner. In the end, both filters estimate similar external forces. As it can be seen in Fig. 6.5, the external vertical force does not converge exactly to zero. We find experimentally that this estimated force is due to friction. Lastly, as it can be seen, the cost of the RS-EKF-based controller is lower after the robot is dropped. The average cost of

the RS-EKF is 0.065 while the one of the EKF is 0.130, which corresponds to a 50% improvement. This demonstrates how a filter aware of the cost objective can improve the overall performance.

In order to get a more systematic comparison between the filters and get rid of the human error, we perform two additional experiments where the filters are initialized with exactly the same priors. First, we initialize both the filters with a wrong prior on the external vertical force of 20 N, while in reality, no force is applied on the robot. This experiment creates an identical situation as the previous experiment while also ensuring the very same initial conditions for the robot. The results are shown in Fig 6.6, where the RS-EKF still performs better. Also, the performance is similar to the first experiment. In that experiment, we obtain a 62.9% improvement in the average cost.

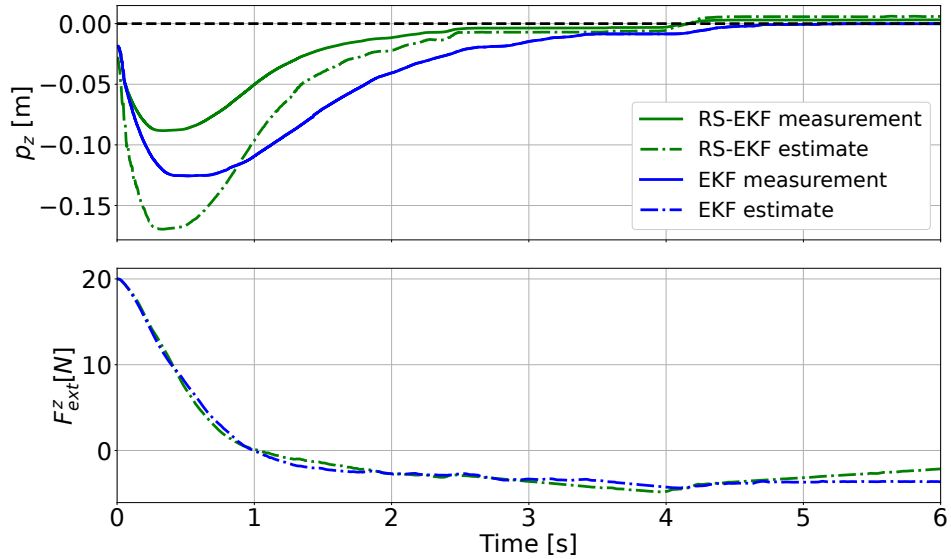


Figure 6.6: Comparison of the RS-EKF and EKF when initialized with a wrong prior of 20 N on the estimated vertical external force.

Finally, we replicate the previous experiment but now, initialize the filters with a wrong prior of  $-10$  N on the external force, while, in fact, there is no force on

the robot. The RS-EKF reacts sooner than EKF once again. It brings the base of Solo12 back to the desired location sooner than EKF, as can be seen in Fig 6.7. In that experiment, we obtain a 58.9% improvement in the average cost.

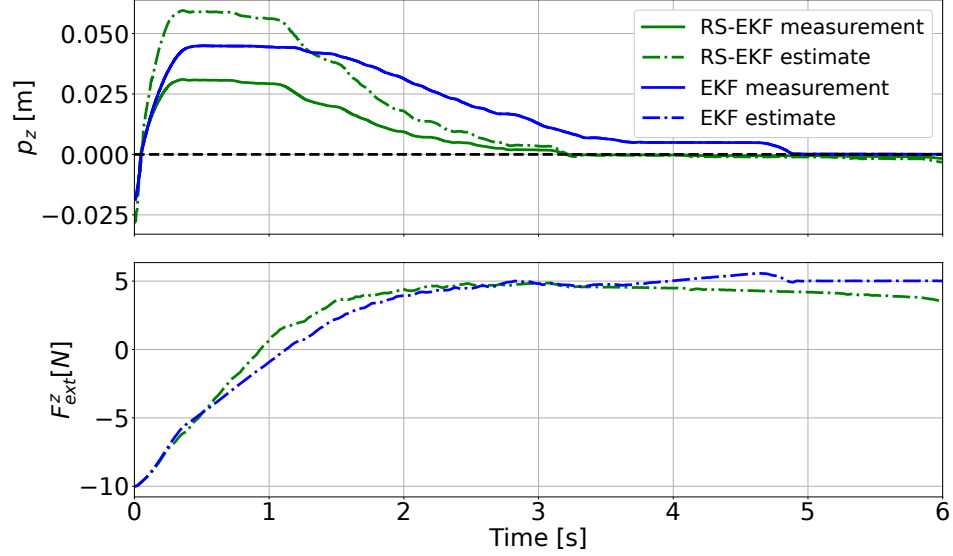


Figure 6.7: Comparison of the RS-EKF and EKF when initialized with a wrong prior of  $-10$  N on the estimated vertical external force.

## 6.4 Conclusion

To conclude, we have introduced a risk-sensitive variation of the EKF based on the zero-sum game introduced by Whittle [186]. The filter biases estimates towards high regions of the control cost which result in more robust controllers. Furthermore, the complexity of this filter is similar to the EKF. Lastly, we have demonstrated on several robotics problems, both in simulation and on real hardware, the benefits of this filter for output-feedback MPC. Importantly, we have shown that the proposed filter makes the controller more conservative in phases of high uncertainty leading to better overall control performance.

# Chapter 7

## Conclusion

In this thesis, we investigated how to fully leverage the promises of MPC on torque-controlled robots.

Part I focused on state-feedback MPC and proposed several contributions to obtain safe and globally optimal plans. In Chapter 2, we showed that standard optimization techniques can already push the limits of closed-loop MPC for torque-controlled robots. By including hard constraints in the MPC, we are now able to ensure safety. In Chapter 3, we introduced a way to overcome the local behavior of MPC by combining constrained TO with RL by using a learned value function as a terminal cost of the MPC. Chapter 4 introduced how to perform force feedback in Model Predictive Control in order to interact accurately with their environment. Our experimental results show that current optimization-based control and estimation techniques are sufficient to incorporate force sensors in model-predictive controllers.

Part II investigated how to endow Model Predictive Control with the perception uncertainty information. In Chapters 5 and 6, we introduced efficient numerical methods to perform risk-sensitive output feedback Model Predictive Control. While

the studied formulations were already well established in the control community, we showed how to implement them efficiently in order to enable them at the frequency required for torque control robots. To the best of our knowledge, we provided the first demonstration on a real robot that a risk-sensitive controller can outperform a neutral controller.

In summary, we presented several contributions to overcome the current limitations of MPC on torque controlled robots. Specifically, we proposed to endow MPC with safety, global reasoning, force-feedback, and perception uncertainty information.

While the presented contributions can be used to design controllers that reason online about novel situations despite perception uncertainty, one could argue that some work remains to be done to deploy robots in complex environments such as the construction sites depicted in Figure 1.1. Clearly, a robot would require more sensory information to navigate such settings. While we have investigated the use of force-feedback in Chapter 4, it should ideally be used in combination with vision. Furthermore, while Chapter 3 studied how to obtain a global solution using offline reasoning, it would be interesting to further investigate how to solve global optimization problems online. Finally, even though it is crucial to have the ability to solve complex nonlinear optimization problems online to adapt to new situations, this can be a waste of computational resources in known situations. For known settings, a policy could be used. The recent successes of RL [22, 168] in locomotion [1, 84] and manipulation [37, 79] demonstrated that a controller based on neural networks can achieve impressive results. Ideally, a robot should know how to transition from offline to online reasoning when it encounters a novel situation. Automatizing this transition appears as an open problem.



# Appendix A

## Stagewise Implementations of Sequential Quadratic Programming for Model-Predictive Control

Let's consider the following Quadratic Program:

$$\min_{x_{1:T}, u_{0:T-1}} x_T^T Q_T x_T + x_T^T q_T + \sum_{k=0}^{T-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} x_k \\ u_k \end{bmatrix} \quad (\text{A.1})$$

subject to  $x_0 = 0$ ,

$$x_{k+1} = A_k x_k + B_k u_k + \gamma_{k+1}, \quad 0 \leq k < T.$$

The associated Lagrangian has the following form:

$$\begin{aligned} \mathcal{L}(x_{1:T}, u_{0:T-1}, \lambda_{1:T}) = & x_T^T Q_T x_T + x_T^T q_T \\ & + \sum_{k=0}^{T-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ & - \lambda_{k+1}^T (x_{k+1} - A_k x_k - B_k u_k - \gamma_{k+1}) \end{aligned} \quad (\text{A.2})$$

The KKT condition can be written as a set of linear equations:

$$Q_T x_T + q_T = \lambda_T \quad (\text{A.3})$$

$$Q_k x_k + S_k u_k + q_k + A_k^T \lambda_{k+1} = \lambda_k \quad \forall k \leq 1 \quad (\text{A.4})$$

$$R_k u_k + S_k^T x_k + r_k + B_k^T \lambda_{k+1} = 0 \quad \forall k < T \quad (\text{A.5})$$

$$x_{k+1} = A_k x_k + B_k u_k + \gamma_{k+1}. \quad (\text{A.6})$$

**Proposition 7.** *The KKT conditions can be written as a block tri-diagonal symmetric matrix equation. More precisely:*

$$\begin{bmatrix} \Gamma_1 & M_1^T & 0 & 0 & \cdots & 0 \\ M_1 & \Gamma_2 & M_2^T & 0 & \cdots & 0 \\ 0 & M_2 & \Gamma_3 & M_3^T & \cdots & 0 \\ 0 & 0 & M_3 & \Gamma_4 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & \Gamma_T \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ s_T \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \vdots \\ g_T \end{bmatrix} \quad (\text{A.7})$$

where:

$$\begin{aligned}
\Gamma_k &= \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & Q_k & I \\ -B_{k-1} & I & 0 \end{bmatrix}, \quad M_k = \begin{bmatrix} 0 & S_k^T & 0 \\ 0 & 0 & 0 \\ 0 & -A_k & 0 \end{bmatrix} \\
\text{and } s_k &= \begin{bmatrix} u_{k-1} \\ x_k \\ -\lambda_k \end{bmatrix}, \quad g_k = \begin{bmatrix} -r_{k-1} \\ -q_k \\ \gamma_k \end{bmatrix}
\end{aligned} \tag{A.8}$$

*Proof.* We find that:

- For  $k = 1$ , as  $x_0 = 0$ :

$$\begin{aligned}
\Gamma_1 s_1 + M_1^T s_2 &= \begin{bmatrix} R_0 u_0 + B_0^T \lambda_1 \\ Q_1 x_1 - \lambda_1 \\ -B_0 u_0 + x_1 \end{bmatrix} + \begin{bmatrix} 0 \\ S_1 u_1 + A_1^T \lambda_2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} R_0 u_0 + S_0^T x_0 + B_0^T \lambda_1 \\ Q_1 x_1 - \lambda_1 + S_1 u_1 + A_1^T \lambda_2 \\ -A_0 x_0 - B_0 u_0 + x_1 \end{bmatrix} = \begin{bmatrix} -r_0 \\ -q_1 \\ \gamma_1 \end{bmatrix}
\end{aligned} \tag{A.9}$$

- For  $1 < k < T$ :

$$\begin{aligned}
M_{k-1}s_{k-1} + \Gamma_k s_k + M_k^T s_{k+1} &= \\
\begin{bmatrix} S_{k-1}^T x_{k-1} \\ 0 \\ -A_{k-1} x_{k-1} \end{bmatrix} &+ \begin{bmatrix} R_{k-1} u_{k-1} + B_{k-1}^T \lambda_k \\ Q_k x_k - \lambda_k \\ -B_{k-1} u_{k-1} + x_k \end{bmatrix} + \begin{bmatrix} 0 \\ S_k u_k + A_k^T \lambda_{k+1} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -r_{k-1} \\ -q_k \\ \gamma_k \end{bmatrix} = g_k
\end{aligned} \tag{A.10}$$

- For  $k = T$ :

$$\begin{aligned}
M_{T-1}s_{T-1} + \Gamma_T s_T &= \begin{bmatrix} S_{T-1}^T x_{T-1} \\ 0 \\ -A_{T-1} x_{T-1} \end{bmatrix} + \begin{bmatrix} R_{T-1} u_{T-1} + B_{T-1}^T \lambda_T \\ Q_T x_T - \lambda_T \\ -B_{T-1} u_{T-1} + x_T \end{bmatrix} = \begin{bmatrix} -r_{T-1} \\ -q_T \\ \gamma_T \end{bmatrix} = g_T
\end{aligned} \tag{A.11}$$

□

**Proposition 8.** *By applying Thomas algorithm, we recover the well-known Riccati recursions. Specifically, the **backward pass** can be done by initializing  $V_T = Q_T$  and  $v_T = q_T$ , and then by applying the following equations:*

$$h_k = r_k + B_k^T (v_{k+1} + V_{k+1} \gamma_{k+1}) \tag{A.12}$$

$$G_k = S_k^T + B_n^T V_{k+1} A_k \qquad K_k = -H_k^{-1} G_k$$

$$H_k = R_k + B_k^T V_{k+1} B_k \qquad k_k = -H_k^{-1} h_k$$

$$V_k = Q_k + A_k^T V_{k+1} A_k - K_k^T H_k K_k \quad (\text{A.13})$$

$$v_k = q_k + K_k^T r_k + (A_k + K_k B_k)^T (v_{k+1} + V_{k+1} \gamma_{k+1})$$

Then, the **forward pass** initializes  $\Delta x_0 = 0$  and unrolls the linearized dynamics:

$$\Delta x_{k+1} = (A_k + B_k K_k) \Delta x_k + B_k k_k + \gamma_{k+1} \quad (\text{A.14})$$

$$\Delta u_k = K_k \Delta x_k + k_k \quad (\text{A.15})$$

$$\lambda_k = V_k \Delta x_k + v_k \quad (\text{A.16})$$

*Proof.* For this proof, we will extensively use the following lemma:

**Lemma A.0.1.**

$$\begin{aligned} & \begin{bmatrix} R_k & 0 & -B_k^T \\ 0 & V_{k+1} & I \\ -B_k & I & 0 \end{bmatrix}^{-1} = \\ & \begin{bmatrix} H_{k+1}^{-1} & H_{k+1}^{-1} B_k^T & -H_{k+1}^{-1} B_k^T V_{k+1} \\ B_k H_{k+1}^{-1} & B_k H_{k+1}^{-1} B_k^T & I - B_k H_{k+1}^{-1} B_k^T V_{k+1} \\ -V_{k+1} B_k H_{k+1}^{-1} & I - V_{k+1} B_k H_{k+1}^{-1} B_k^T & -V_{k+1} (I - B_k H_{k+1}^{-1} B_k^T V_{k+1}) \end{bmatrix} \\ & = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & I \\ 0 & I & -V_{k+1} \end{bmatrix} + \begin{bmatrix} H_{k+1}^{-1} & 0 & 0 \\ 0 & B_k H_{k+1}^{-1} & 0 \\ 0 & 0 & -V_{k+1} B_k H_{k+1}^{-1} \end{bmatrix} \begin{bmatrix} I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \end{bmatrix} \end{aligned}$$

where  $H_k = R_k + B_k^T V_{k+1} B_k$

Thomas algorithm uses a forward recursion in order to find a equivalent linear

system of the form:

$$\begin{bmatrix} I & 0 & 0 & 0 & \dots & 0 \\ \bar{\Gamma}_2^{-1}M_1 & I & 0 & 0 & \dots & 0 \\ 0 & \bar{\Gamma}_3^{-1}M_2 & I & 0 & \dots & 0 \\ 0 & 0 & \bar{\Gamma}_4^{-1}M_3 & I & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & I \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ s_T \end{bmatrix} = \begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \\ \bar{g}_3 \\ \bar{g}_4 \\ \vdots \\ \bar{g}_T \end{bmatrix} \quad (\text{A.17})$$

Then, a forward recursion recovers the sequence:  $s_1, \dots, s_T$ . This can be formalized in the following way:

---

**Algorithm 6:** Thomas algorithm

---

```

1  $\bar{\Gamma}_T \leftarrow \Gamma_T$ 
2  $\bar{g}_T \leftarrow \Gamma_T^{-1} g_T$ 
   /* backward pass */
3 for  $k \leftarrow 1$  to  $T - 1$  do
4    $\bar{\Gamma}_k \leftarrow \Gamma_k - M_k^T \bar{\Gamma}_{k+1}^{-1} M_k$ 
5    $\bar{g}_k \leftarrow \bar{\Gamma}_k^{-1} (g_k - M_k^T \bar{g}_{k+1})$ 
   /* forward pass */
6  $s_1 \leftarrow \bar{g}_1$ 
7 for  $k \leftarrow 1$  to  $T - 1$  do
8    $s_{k+1} \leftarrow \bar{g}_{k+1} - \bar{\Gamma}_{k+1}^{-1} M_{k+1} s_k$ 

```

---

Let's now show by recursion that:

$$\bar{\Gamma}_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & V_k & I \\ -B_{k-1} & I & 0 \end{bmatrix} \quad (\text{A.18})$$

and that:

$$\bar{g}_k = \bar{\Gamma}_k^{-1} \begin{bmatrix} -r_{k-1} \\ -v_k \\ \gamma_k \end{bmatrix} \quad (\text{A.19})$$

**Backward pass:**

- $k = T$

As  $V_T = Q_T$  and  $v_T = q_T$  by definition, the property is true for  $k = T$ .

- If the property is true for  $k + 1$ ,

$$\begin{aligned} \bar{\Gamma}_k &= \Gamma_k - M_k^T \bar{\Gamma}_{k+1}^{-1} M_k \quad (\text{A.20}) \\ &= \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & Q_k & I \\ -B_{k-1} & I & 0 \end{bmatrix} \\ &\quad - \begin{bmatrix} 0 & 0 & 0 \\ S_k & 0 & -A_k^T \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} R_k & 0 & -B_k^T \\ 0 & V_{k+1} & I \\ -B_k & I & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & S_k^T & 0 \\ 0 & 0 & 0 \\ 0 & -A_k & 0 \end{bmatrix} \\ &= \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & V_k & I \\ -B_{k-1} & I & 0 \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned}
 V_k &= Q_k + A_k^T V_{k+1} A_k - S_k H_{k+1}^{-1} (S_k^T + B_k^T V_{k+1} A_k) \\
 &\quad - A_k^T V_{k+1} B_k H_{k+1}^{-1} (S_k^T + B_k^T V_{k+1} A_k) \\
 &= Q_k + A_k^T V_{k+1} A_k - (S_k + A_k^T V_{k+1} B_k) (R_k + B_k^T V_{k+1} B_k)^{-1} (S_k^T + B_k^T V_{k+1} A_k)
 \end{aligned} \tag{A.21}$$



then, we have:

$$\begin{aligned}
\bar{\Gamma}_k \bar{g}_k - g_k &= -M_k^T \bar{g}_{k+1} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ S_k & 0 & -A_k^T \\ 0 & 0 & 0 \end{bmatrix} \bar{\Gamma}_{k+1}^{-1} \begin{bmatrix} r_k \\ v_{k+1} \\ -\gamma_{k+1} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ S_k & 0 & -A_k^T \\ 0 & 0 & 0 \end{bmatrix} \left( \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & I \\ 0 & I & -V_{k+1} \end{bmatrix} \right. \\
&\quad \left. + \begin{bmatrix} H_{k+1}^{-1} & 0 & 0 \\ 0 & B_k H_{k+1}^{-1} & 0 \\ 0 & 0 & -V_{k+1} B_k H_{k+1}^{-1} \end{bmatrix} \begin{bmatrix} I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \end{bmatrix} \right) \begin{bmatrix} r_k \\ v_{k+1} \\ -\gamma_{k+1} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & -A_k^T & A_k^T V_{k+1} \\ 0 & 0 & 0 \end{bmatrix} \\
&\quad + \begin{bmatrix} 0 & 0 & 0 \\ S_k H_{k+1}^{-1} & 0 & A_k^T V_{k+1} B_k H_{k+1}^{-1} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \end{bmatrix} \begin{bmatrix} r_k \\ v_{k+1} \\ -\gamma_{k+1} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ -A_k^T v_{k+1} - A_k^T V_{k+1} \gamma_{k+1} \\ 0 \end{bmatrix} \\
&\quad + \begin{bmatrix} 0 & 0 & 0 \\ S_k H_{k+1}^{-1} & 0 & A_k^T V_{k+1} B_k H_{k+1}^{-1} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r_k + B_k^T v_{k+1} + B_k^T V_{k+1} \gamma_{k+1} \\ r_k + B_k^T v_{k+1} + B_k^T V_{k+1} \gamma_{k+1} \\ r_k + B_k^T v_{k+1} + B_k^T V_{k+1} \gamma_{k+1} \end{bmatrix}
\end{aligned}$$

and we get:

$$\begin{aligned}
v_k &= q_k + A_k^T(v_{k+1} + V_{k+1}\gamma_{k+1}) \\
&\quad - (S_k + A_k^T V_{k+1} B_k) H_{k+1}^{-1} (r_k + B_k^T v_{k+1} + B_k^T V_{k+1} \gamma_{k+1}) \\
&= q_k + A_k^T(v_{k+1} + V_{k+1}\gamma_{k+1}) + K_k^T(r_k + B_k^T v_{k+1} + B_k^T V_{k+1} \gamma_{k+1}) \\
&= q_k + K_k^T r_k + (A_k + B_k K_k)^T(v_{k+1} + V_{k+1}\gamma_{k+1})
\end{aligned} \tag{A.22}$$

Hence, the property is also true for  $k$ .

**Forward pass:**

- $k = 1$ :

$$s_1 = \bar{g}_1 \tag{A.23}$$

implies that:

$$\begin{aligned}
\begin{pmatrix} u_0 \\ x_1 \\ \lambda_1 \end{pmatrix} &= \bar{\Gamma}_1^{-1} \begin{bmatrix} -r_0 \\ -v_1 \\ \gamma_1 \end{bmatrix} \\
&= \left( \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & I \\ 0 & I & -V_1 \end{bmatrix} + \begin{bmatrix} H_0^{-1} & 0 & 0 \\ 0 & B_0 H_0^{-1} & 0 \\ 0 & 0 & -V_1 B_0 H_1^{-1} \end{bmatrix} \begin{bmatrix} I & B_0^T & -B_0^T V_1 \\ I & B_0^T & -B_0^T V_1 \\ I & B_0^T & -B_0^T V_1 \end{bmatrix} \right) \begin{bmatrix} -r_0 \\ -v_1 \\ \gamma_1 \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ \gamma_1 \\ -v_1 - V_1 \gamma_1 \end{bmatrix} - \begin{bmatrix} H_0^{-1} & 0 & 0 \\ 0 & B_0 H_0^{-1} & 0 \\ 0 & 0 & -V_1 B_0 H_1^{-1} \end{bmatrix} \begin{bmatrix} r_0 + B_0^T (v_1 + V_1 \gamma_1) \\ r_0 + B_0^T (v_1 + V_1 \gamma_1) \\ r_0 + B_0^T (v_1 + V_1 \gamma_1) \end{bmatrix} \\
&= \begin{bmatrix} k_0 \\ B_0 k_0 + \gamma_1 \\ -V_1 B_0 k_0 - V_1 \gamma_1 - v_1 \end{bmatrix} \tag{A.24}
\end{aligned}$$

Hence,  $\lambda_1 = -V_1 x_1 - v_1$

- $k \geq 1$ :

Then:

$$\bar{s}_k = \bar{g}_k - \bar{\Gamma}_k^{-1} M_{k-1} s_{k-1} \tag{A.25}$$

implies that:

$$\begin{aligned}
\begin{pmatrix} u_k \\ x_{k+1} \\ \lambda_{k+1} \end{pmatrix} &= \bar{\Gamma}_{k+1}^{-1} \begin{pmatrix} -r_k \\ -v_{k+1} \\ \gamma_{k+1} \end{pmatrix} - M_k s_k \\
&= \begin{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & I \\ 0 & I & -V_{k+1} \end{bmatrix} \\ + \begin{bmatrix} H_{k+1}^{-1} & 0 & 0 \\ 0 & B_k H_{k+1}^{-1} & 0 \\ 0 & 0 & -V_{k+1} B_k H_{k+1}^{-1} \end{bmatrix} \begin{bmatrix} I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \\ I & B_k^T & -B_k^T V_{k+1} \end{bmatrix} \end{pmatrix} \begin{bmatrix} -r_k - S_k^T x_k \\ -v_{k+1} \\ \gamma_{k+1} + A_k x_k \end{bmatrix} \\
&= \begin{bmatrix} k_k + K_k x_k \\ (A_k + B_k K_k) x_k + B_k k_k + \gamma_{k+1} \\ V_{k+1} ((A_k + B_k K_k) x_k + B_k k_k + \gamma_{k+1}) + v_{k+1} \end{bmatrix} \tag{A.26}
\end{aligned}$$

and we get:

$$\lambda_{k+1} = V_{k+1} x_{k+1} + v_{k+1} \tag{A.27}$$

□

## Appendix B

# Stagewise Newton Method for Dynamic Game Control with Imperfect State Observation

### B.1 Problem statement

Given a sequence of measurements  $y_{1:t}$ , a sequence of control inputs  $u_{0:t-1}$ , and a prior on the initial state  $\hat{x}_0$ , we study the following minimization-maximization problem:

$$\begin{aligned} \min_{u_{t:T-1}} \max_{w_{0:T}} \max_{\gamma_{1:t}} \sum_{j=0}^{T-1} \ell_j(x_j, u_j) + \ell_T(x_T) \\ - \frac{1}{2\mu} \left( \omega_0^T P^{-1} \omega_0 + \sum_{j=1}^t \gamma_j^T R_j^{-1} \gamma_j + \sum_{j=1}^T w_j^T Q_j^{-1} w_j \right) \end{aligned} \quad (\text{B.1})$$

$$\text{subject to the constraints:} \quad x_0 = \hat{x}_0 + w_0, \quad (\text{B.2a})$$

$$x_{j+1} = f_j(x_j, u_j) + w_{j+1}, \quad 0 \leq j < T, \quad (\text{B.2b})$$

$$y_j = h_j(x_j) + \gamma_j, \quad 1 \leq j \leq t. \quad (\text{B.2c})$$

where  $\mu > 0$ .  $x_j$  is the state,  $\omega_j$  the process disturbance,  $\gamma_j$  the measurement disturbance,  $T$  the time horizon,  $t$  the current time.  $f_j$ , the transition model,  $h_j$ , measurement model and  $\ell_j$ , the controller's cost are assumed to be  $\mathcal{C}^2$ .  $R_j$  the measurement uncertainty,  $Q_j$  the process uncertainty and  $P$  the initial state uncertainty are positive semi-definite matrices. Instead of searching for a global solution, we search for a stationary point of the following unconstrained cost.

$$\begin{aligned} J(x_{0:T}, u_{t:T-1}) = & \sum_{j=0}^{T-1} \ell_j(x_j, u_j) + \ell_T(x_T) \\ & - \frac{1}{2\mu} (x_0 - \hat{x}_0)^T P^{-1} (x_0 - \hat{x}_0) \\ & - \frac{1}{2\mu} \sum_{j=1}^t (y_j - h_j(x_j))^T R_j^{-1} (y_j - h_j(x_j)) \\ & - \frac{1}{2\mu} \sum_{j=0}^{T-1} (x_{j+1} - f_j(x_j, u_j))^T Q_{j+1}^{-1} (x_{j+1} - f_j(x_j, u_j)) \end{aligned}$$

## B.2 Characterization of the Newton step

Let's consider a step

$$p = \begin{bmatrix} p_{x_{0:T}} \\ p_{u_{t:T-1}} \end{bmatrix}, \quad (\text{B.3})$$

we know that the Newton step  $p$  satisfies:

$$Hp = -\nabla J, \quad (\text{B.4})$$

where

$$H = \begin{bmatrix} \frac{\partial^2 J}{\partial x_{0:T} \partial x_{0:T}} & \frac{\partial^2 J}{\partial x_{0:T} \partial u_{t:T-1}} \\ \frac{\partial^2 J}{\partial u_{t:T-1} \partial x_{0:T}} & \frac{\partial^2 J}{\partial u_{t:T-1} \partial u_{t:T-1}} \end{bmatrix} \quad (\text{B.5})$$

and

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial x_{0:T}} \\ \frac{\partial J}{\partial u_{t:T-1}} \end{bmatrix} \quad (\text{B.6})$$

And we use the following notation for the Hessian:

$$\frac{\partial^2 J}{\partial x_{0:T} \partial x_{0:T}} = \begin{bmatrix} \frac{\partial^2 J}{\partial x_0^2} & \frac{\partial^2 J}{\partial x_0 \partial x_1} & \cdots & \frac{\partial^2 J}{\partial x_0 \partial x_T} \\ \frac{\partial^2 J}{\partial x_1 \partial x_0} & \frac{\partial^2 J}{\partial x_1^2} & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_T} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_T \partial x_0} & \frac{\partial^2 J}{\partial x_T \partial x_1} & \cdots & \frac{\partial^2 J}{\partial x_T^2} \end{bmatrix} \quad (\text{B.7})$$

Note that:  $\frac{\partial^2 J}{\partial x_i \partial x_j} \in \mathbb{R}^{n_x \times n_x}$  and  $\frac{\partial^2 J}{\partial x_i \partial u_j} \in \mathbb{R}^{n_x \times n_u}$ . To simplify the derivations, we define the gaps:

$$\gamma_k := y_k - h_k(x_k) \quad (\text{B.8})$$

$$w_{k+1} := x_{k+1} - f_k(x_k, u_k) \quad (\text{B.9})$$

$$w_0 := x_0 - \hat{x}_0 \quad (\text{B.10})$$

In the rest of this section, we show how to derive the Newton step sequentially, namely without inverting the Hessian matrix. To do so, let's retrieve the analytical expression of the gradient and Hessian of  $J$ . We assume that  $t < T$ , otherwise, there is no control input. However, we note that if  $t = T$ , the past stress recursion could be derived similarly. First, we derive the gradient with respect to every state variable and every control inputs:

- $k = 0$

$$\frac{\partial J}{\partial x_0}(x_{0:T}, u_{t:T-1}) = \ell_0^x - \mu^{-1} P^{-1} w_0 + \mu^{-1} f_0^{xT} Q_1^{-1} w_1$$



- $\forall 1 \leq k \leq t-1,$

$$\frac{\partial J}{\partial x_k}(x_{0:T}, u_{t:T-1}) = \ell_k^x - \mu^{-1} Q_k^{-1} w_k + \mu^{-1} f_k^{xT} Q_{k+1}^{-1} w_{k+1} + \mu^{-1} h_k^{xT} R_k^{-1} \gamma_k$$

- $k = t$

$$\frac{\partial J}{\partial x_t}(x_{0:T}, u_{t:T-1}) = \ell_t^x - \mu^{-1} Q_t^{-1} w_t + \mu^{-1} f_t^{xT} Q_{t+1}^{-1} w_{t+1} + \mathbf{1}_{t \geq 1} \mu^{-1} h_t^{xT} R_t^{-1} \gamma_t$$

- $\forall t+1 \leq k \leq T-1,$

$$\frac{\partial J}{\partial x_k}(x_{0:T}, u_{t:T-1}) = \ell_k^x - \mu^{-1} Q_k^{-1} w_k + \mu^{-1} f_k^{xT} Q_{k+1}^{-1} w_{k+1}$$

- $k = T$

$$\frac{\partial J}{\partial x_T}(x_{0:T}, u_{t:T-1}) = \ell_T^x - \mu^{-1} Q_T^{-1} w_T$$

and

$$\forall t \leq k \leq T-1, \quad \frac{\partial J}{\partial u_k}(u_{t:T}, x_{0:T}) = \ell_k^u + \mu^{-1} f_k^{uT} Q_{k+1}^{-1} w_{k+1}$$

Next, we derive each term of the Hessian. All the terms that are not equal to zero are of the form:

$$\frac{\partial^2 J}{\partial x_k \partial x_k} = \begin{cases} \ell_0^{xx} - \mu^{-1} P^{-1} - \mu^{-1} f_0^T Q_1^{-1} f_0^x + \mu^{-1} f_0^{xxT} Q_1^{-1} w_1 & k = 0 \\ \ell_k^{xx} - \mu^{-1} Q_k^{-1} - \mu^{-1} f_k^T Q_{k+1}^{-1} f_k^x + \mu^{-1} f_k^{xxT} Q_{k+1}^{-1} w_{k+1} \\ \quad - \mu^{-1} h_k^{xT} R_k^{-1} h_k^x + \mu^{-1} h_k^{xxT} R_k^{-1} \gamma_k & 1 \leq k \leq t \\ \ell_k^{xx} - \mu^{-1} Q_k^{-1} - \mu^{-1} f_k^T Q_{k+1}^{-1} f_k^x + \mu^{-1} f_k^{xxT} Q_{k+1}^{-1} w_{k+1} & t < k < T \\ \ell_T^{xx} - \mu^{-1} Q_T^{-1} & k = T \end{cases}$$

$$\frac{\partial^2 J}{\partial x_{k+1} \partial x_k} = \mu^{-1} Q_{k+1}^{-1} f_k^x \quad (\text{B.11})$$

$$\frac{\partial^2 J}{\partial x_{k-1} \partial x_k} = \mu^{-1} f_{k-1}^{xT} Q_k^{-1} \quad (\text{B.12})$$

$$\frac{\partial^2 J}{\partial u_{k-1} \partial x_k} = \mu^{-1} f_{k-1}^{uT} Q_k^{-1} \quad (\text{B.13})$$

$$\frac{\partial^2 J}{\partial u_k \partial x_k} = \ell_k^{ux} - \mu^{-1} f_k^{uT} Q_{k+1}^{-1} f_k^x + \mu^{-1} w_{k+1}^T Q_{k+1}^{-1} f_k^{ux} \quad (\text{B.14})$$

$$\frac{\partial^2 J}{\partial x_k \partial u_k} = \ell_k^{xu} - \mu^{-1} f_k^{xT} Q_{k+1}^{-1} f_k^u + \mu^{-1} w_{k+1}^T Q_{k+1}^{-1} f_k^{xu} \quad (\text{B.15})$$

$$\frac{\partial^2 J}{\partial x_{k+1} \partial u_k} = \mu^{-1} Q_{k+1}^{-1} f_k^u \quad (\text{B.16})$$

$$\frac{\partial^2 J}{\partial u_k \partial u_k} = \ell_k^{uu} - \mu^{-1} f_k^{uT} Q_{k+1}^{-1} f_k^u + \mu^{-1} w_{k+1}^T Q_{k+1}^{-1} f_k^{uu} \quad (\text{B.17})$$

where  $f_k^{ux}$  is a tensor and  $w_{k+1}^T Q_{k+1}^{-1} f_k^{ux}$  is a matrix. More precisely,

$$(f_k^{ux})_{i,j,l} = \frac{\partial^2 (f_k)_l}{\partial u_i \partial x_k} \quad (\text{B.18})$$

where  $(f_k)_l$  is the  $l$  component of  $f_k$ . And, the product is defined as follows:

$$(w_{k+1}^T Q_{k+1}^{-1} f_k^{ux})_{i,j} = \sum_{l=1}^n (f_k^{ux})_{i,j,l} (Q_{k+1}^{-1} w_{k+1}^i)_l \quad (\text{B.19})$$

To simplify, let's denote for all  $k < T$ :

$$\begin{aligned} \bar{\ell}_k^{xx} &= \ell_k^{xx} + \mu^{-1} w_{k+1}^T Q_{k+1}^{-1} f_k^{xx} + \mu^{-1} \mathbf{1}_{k \leq t/\gamma_k}^T R_k^{-1} h_k^{xx} \\ \bar{\ell}_k^{xu} &= \bar{\ell}_k^{ux^T} = \ell_k^{xu} + \mu^{-1} w_{k+1}^T Q_{k+1}^{-1} f_k^{xu} \\ \bar{\ell}_k^{uu} &= \ell_k^{uu} + \mu^{-1} w_{k+1}^T Q_{k+1}^{-1} f_k^{uu} \end{aligned} \quad (\text{B.20})$$

Therefore, Equation (B.5) is equivalent to:

- If  $k = 0$  and if  $t \geq 1$ :

$$\frac{\partial^2 J}{\partial x_0 \partial x_0} p_{x_0} + \frac{\partial^2 J}{\partial x_0 \partial x_1} p_{x_1} = -\frac{\partial J}{\partial x_0} \quad (\text{B.21})$$

- $\forall k = 1, \dots, t-1$ :

$$\frac{\partial^2 J}{\partial x_k \partial x_{k-1}} p_{x_{k-1}} + \frac{\partial^2 J}{\partial x_k \partial x_k} p_{x_k} + \frac{\partial^2 J}{\partial x_k \partial x_{k+1}} p_{x_{k+1}} = -\frac{\partial J}{\partial x_k} \quad (\text{B.22})$$

- $k = t$ :

$$\frac{\partial^2 J}{\partial x_t \partial x_{t-1}} p_{x_{t-1}} \mathbf{1}_{t \geq 1} + \frac{\partial^2 J}{\partial x_t \partial x_t} p_{x_t} + \frac{\partial^2 J}{\partial x_t \partial x_{t+1}} p_{x_{t+1}} + \frac{\partial^2 J}{\partial x_t \partial u_t} p_{u_t} = -\frac{\partial J}{\partial x_t} \quad (\text{B.23})$$

- $\forall k = t+1, \dots, T-1$ :

$$\begin{aligned} \frac{\partial^2 J}{\partial x_k \partial x_{k-1}} p_{x_{k-1}} + \frac{\partial^2 J}{\partial x_k \partial x_k} p_{x_k} + \frac{\partial^2 J}{\partial x_k \partial x_{k+1}} p_{x_{k+1}} + \frac{\partial^2 J}{\partial x_k \partial u_{k-1}} p_{u_{k-1}} \\ + \frac{\partial^2 J}{\partial x_k \partial u_k} p_{u_k} = -\frac{\partial J}{\partial x_k} \end{aligned} \quad (\text{B.24})$$

- $k = T$ :

$$\frac{\partial^2 J}{\partial x_T \partial x_{T-1}} p_{x_{T-1}} + \frac{\partial^2 J}{\partial x_T \partial x_T} p_{x_T} + \frac{\partial^2 J}{\partial x_T \partial u_{T-1}} p_{u_{T-1}} = -\frac{\partial J}{\partial x_T} \quad (\text{B.25})$$

- $\forall k = t, \dots, T-1$ :

$$\frac{\partial^2 J}{\partial u_k \partial x_k} p_{x_k} + \frac{\partial^2 J}{\partial u_k \partial x_{k+1}} p_{x_{k+1}} + \frac{\partial^2 J}{\partial u_k \partial u_k} p_{u_k} = -\frac{\partial J}{\partial u_k} \quad (\text{B.26})$$

### B.3 Future stress

In this section, we derive the recursion for the future stress:

- From (B.24),  $\forall k = t + 1, \dots, T - 1$ :

$$\begin{aligned}
& \mu^{-1} Q_k^{-1} f_{k-1}^x p_{x_{k-1}} \\
& + \left( \bar{\ell}_k^{xx} - \mu^{-1} Q_k^{-1} - \mu^{-1} f_k^{xT} Q_{k+1}^{-1} f_k^x \right) p_{x_k} \\
& + \mu^{-1} f_k^{xT} Q_{k+1}^{-1} p_{x_{k+1}} \\
& + \mu^{-1} Q_k^{-1} f_{k-1}^u p_{u_{k-1}} \\
& + \left( \bar{\ell}_k^{xu} - \mu^{-1} f_k^{xT} Q_{k+1}^{-1} f_k^u \right) p_{u_k} \\
& + \ell_k^x - \mu^{-1} Q_k^{-1} w_k + \mu^{-1} f_k^{xT} Q_{k+1}^{-1} w_{k+1} = 0
\end{aligned} \tag{B.27}$$

- From (B.25), for  $k = T$ :

$$\begin{aligned}
& \mu^{-1} Q_T^{-1} f_{T-1}^x p_{x_{T-1}} + \left( \ell_T^{xx} - \mu^{-1} Q_T^{-1} \right) p_{x_T} + \mu^{-1} Q_T^{-1} f_{T-1}^u p_{u_{T-1}} \\
& + \ell_T^x - \mu^{-1} Q_T^{-1} w_T = 0
\end{aligned} \tag{B.28}$$

- From (B.26),  $\forall k = t, \dots, T - 1$ :

$$\begin{aligned}
& \left( \bar{\ell}_k^{ux} - \mu^{-1} f_k^{uT} Q_{k+1}^{-1} f_k^x \right) p_{x_k} + \mu^{-1} f_k^{uT} Q_{k+1}^{-1} p_{x_{k+1}} \\
& + \left( \bar{\ell}_k^{uu} - \mu^{-1} f_k^{uT} Q_{k+1}^{-1} f_k^u \right) p_{u_k} + \ell_k^u + \mu^{-1} f_k^{uT} Q_{k+1}^{-1} w_{k+1} = 0
\end{aligned} \tag{B.29}$$

We define

$$\forall k = t, \dots, T-1 \quad \lambda_{k+1} := \mu^{-1} Q_{k+1}^{-1} (p_{x_{k+1}} - f_k^x p_{x_k} - f_k^u p_{u_k} + w_{k+1}) \quad (\text{B.30})$$

and find that (B.27), (B.29) and (B.28) can be written:

$$\forall k = t+1 \dots T-1 \quad \bar{\ell}_k^{xx} p_{x_k} + \bar{\ell}_k^{xu} p_{u_k} + \ell_k^x + f_k^{xT} \lambda_{k+1} = \lambda_k \quad (\text{B.31})$$

$$\forall k = t \dots T-1 \quad \bar{\ell}_k^{ux} p_{x_k} + \bar{\ell}_k^{uu} p_{u_k} + \ell_k^u + f_k^{uT} \lambda_{k+1} = 0 \quad (\text{B.32})$$

$$\ell_T^{xx} p_{x_T} + \ell_T^x = \lambda_T \quad (\text{B.33})$$

**Proposition 9.**

$$\begin{aligned} k = t, \dots, T-1, \quad V_k p_{x_k} + v_k &= \bar{\ell}_k^{xx} p_{x_k} + \bar{\ell}_k^{xu} p_{u_k} + \ell_k^x + f_k^{xT} \lambda_{k+1} \\ V_T p_{x_T} + v_T &= \lambda_T \end{aligned} \quad (\text{B.34})$$

where  $V_k$  and  $v_k$  are solutions of the backward recursion:

$$\Gamma_{k+1} = I - \mu V_{k+1} Q_{k+1} \quad (\text{B.35})$$

$$Q_{uu} = \bar{\ell}_k^{uu} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^u$$

$$Q_{ux} = \bar{\ell}_k^{ux} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x$$

$$Q_u = \ell_k^u + f_k^{uT} \Gamma_{k+1}^{-1} (v_{k+1} - V_{k+1} w_{k+1})$$

$$G_k = -Q_{uu}^{-1} Q_{ux}$$

$$g_k = -Q_{uu}^{-1} Q_u$$

$$V_k = \bar{\ell}_k^{xx} + f_k^{xT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x + Q_{ux}^T G_k$$

$$v_k = \ell_k^x + f_k^{xT} \Gamma_{k+1}^{-1} (v_{k+1} - V_{k+1} w_{k+1}) + Q_{ux}^T g_k$$

with the terminal condition:

$$V_T = \ell_T^{xx}$$

$$v_T = \ell_T^x \quad (\text{B.36})$$

Furthermore,

$$\begin{aligned} p_{x_{k+1}} &= (I - \mu Q_{k+1} V_{k+1})^{-1} (f_k^x p_{x_k} + f_k^u p_{u_k} + \mu Q_{k+1} v_{k+1} - w_{k+1}) \\ p_{u_k} &= G_k p_{x_k} + g_k \end{aligned} \quad (\text{B.37})$$

*Proof.* Clearly:

$$V_T = \ell_T^{xx}$$

$$v_T = \ell_T^x \quad (\text{B.38})$$

Let  $t \leq k \leq T-1$ . Assuming the property is true at  $k+1$ , then, from (B.31) and (B.33), we must have  $\lambda_{k+1} = V_{k+1}p_{x_{k+1}} + v_{k+1}$ . Now, let's show that the property also holds for the index  $k$ . From (B.30), we have:

$$\begin{aligned} V_{k+1}p_{x_{k+1}} + v_{k+1} &= \mu^{-1}Q_{k+1}^{-1} (p_{x_{k+1}} - f_k^x p_{x_k} - f_k^u p_{u_k} + w_{k+1}) \\ (I - \mu Q_{k+1}V_{k+1})p_{x_{k+1}} &= f_k^x p_{x_k} + f_k^u p_{u_k} + \mu Q_{k+1}v_{k+1} - w_{k+1} \end{aligned} \quad (\text{B.39})$$

We define  $\Gamma_{k+1} := (I - \mu V_{k+1}Q_{k+1})^{-1}$  and find that:

$$\begin{aligned} \lambda_{k+1} &= V_{k+1} (I - \mu Q_{k+1}V_{k+1})^{-1} (f_k^x p_{x_k} + f_k^u p_{u_k} + \mu Q_{k+1}v_{k+1} - w_{k+1}) + v_{k+1} \\ &= \Gamma_{k+1}^{-1} V_{k+1} (f_k^x p_{x_k} + f_k^u p_{u_k} + \mu Q_{k+1}v_{k+1} - w_{k+1}) + v_{k+1} \\ &= \Gamma_{k+1}^{-1} V_{k+1} (f_k^x p_{x_k} + f_k^u p_{u_k} - w_{k+1}) + \Gamma_{k+1}^{-1} v_{k+1} \end{aligned} \quad (\text{B.40})$$

as

$$V_{k+1} (I - \mu Q_{k+1}V_{k+1})^{-1} = (V_{k+1}^{-1} - \mu Q_{k+1})^{-1} = (I - \mu V_{k+1}Q_{k+1})^{-1} V_{k+1} \quad (\text{B.41})$$

Now from (B.32)

$$\begin{aligned} \bar{\ell}_k^{ux} p_{x_k} + \bar{\ell}_k^{uu} p_{u_k} + \ell_k^u \\ + f_k^{uT} (\Gamma_{k+1}^{-1} V_{k+1} (f_k^x p_{x_k} + f_k^u p_{u_k} - w_{k+1}) + \Gamma_{k+1}^{-1} v_{k+1}) &= 0 \end{aligned} \quad (\text{B.42})$$

$$\begin{aligned} (\bar{\ell}_k^{uu} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^u) p_{u_k} + (\bar{\ell}_k^{ux} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x) p_{x_k} \\ + \ell_k^u + f_k^{uT} (-\Gamma_{k+1}^{-1} V_{k+1} w_{k+1} + \Gamma_{k+1}^{-1} v_{k+1}) &= 0 \end{aligned} \quad (\text{B.43})$$



which can be written

$$p_{u_k} = G_k p_{x_k} + g_k \quad (\text{B.44})$$

where

$$Q_{uu} = \bar{\ell}_k^{uu} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^u \quad (\text{B.45})$$

$$Q_{ux} = \bar{\ell}_k^{ux} + f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x$$

$$Q_u = \ell_k^u + f_k^{uT} \Gamma_{k+1}^{-1} v_{k+1} - f_k^{uT} \Gamma_{k+1}^{-1} V_{k+1} w_{k+1}$$

$$G_k = -Q_{uu}^{-1} Q_{ux}$$

$$g_k = -Q_{uu}^{-1} Q_u$$

Finally, we get:

$$\begin{aligned} & \bar{\ell}_k^{xx} p_{x_k} + \bar{\ell}_k^{xu} p_{u_k} + \ell_k^x + f_k^{xT} \lambda_{k+1} \\ &= \bar{\ell}_k^{xx} p_{x_k} + \bar{\ell}_k^{xu} p_{u_k} + \ell_k^x + f_k^{xT} \Gamma_{k+1}^{-1} (V_{k+1} (f_k^x p_{x_k} + f_k^u p_{u_k} - w_{k+1}) + v_{k+1}) \\ &= \bar{\ell}_k^{xx} p_{x_k} + Q_{xu} p_{u_k} + \ell_k^x + f_k^{xT} \Gamma_{k+1}^{-1} (V_{k+1} (f_k^x p_{x_k} - w_{k+1}) + v_{k+1}) \\ &= V_k p_{x_k} + v_k \end{aligned} \quad (\text{B.46})$$

with:

$$V_k = \bar{\ell}_k^{xx} + f_k^{xT} \Gamma_{k+1}^{-1} V_{k+1} f_k^x - Q_{xu} Q_{uu}^{-1} Q_{ux} \quad (\text{B.47})$$

$$v_k = \ell_k^x + f_k^{xT} \Gamma_{k+1}^{-1} v_{k+1} - f_k^{xT} \Gamma_{k+1}^{-1} V_{k+1} w_{k+1} - Q_{xu} Q_{uu}^{-1} Q_u$$

## B.4 Past stress

In this section, we derive the recursion for the past stress. Note that those derivations are valid only if  $t \geq 1$ . First, we define,  $\forall k = 0, \dots, t-1$ :

$$\bar{\lambda}_k := \mu^{-1} f_k^{xT} Q_{k+1}^{-1} f_k^x p_{x_k} - \bar{\ell}_k^{xx} p_{x_k} - \mu^{-1} f_k^{xT} Q_{k+1}^{-1} (w_{k+1} + p_{x_{k+1}}) - \ell_k^x \quad (\text{B.48})$$

- From (B.21), for  $k = 0$ :

$$\begin{aligned} & (\bar{\ell}_0^{xx} - \mu^{-1} P^{-1} - \mu^{-1} f_0^{xT} Q_1^{-1} f_0^x) p_{x_0} \\ & + \mu^{-1} f_0^{xT} Q_1^{-1} p_{x_1} + \ell_0^x - \mu^{-1} P^{-1} w_0 + \mu^{-1} f_0^{xT} Q_1^{-1} w_1 = 0 \\ & - \mu^{-1} P^{-1} (p_{x_0} + w_0) = \bar{\lambda}_0 \end{aligned} \quad (\text{B.49})$$

- From (B.22),  $\forall k = 1, \dots, t-1$ :

$$\begin{aligned} & \mu^{-1} Q_k^{-1} f_{k-1}^x p_{x_{k-1}} + (\bar{\ell}_k^{xx} - \mu^{-1} Q_k^{-1} - \mu^{-1} f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu^{-1} h_k^{xT} R_k^{-1} h_k^x) p_{x_k} \\ & + \mu^{-1} f_k^{xT} Q_{k+1}^{-1} p_{x_{k+1}} + \ell_k^x - \mu^{-1} Q_k^{-1} w_k + \mu^{-1} f_k^{xT} Q_{k+1}^{-1} w_{k+1} + \mu^{-1} h_k^{xT} R_k^{-1} \gamma_k = 0 \end{aligned}$$

$$Q_k^{-1} f_{k-1}^x p_{x_{k-1}} - (Q_k^{-1} + h_k^{xT} R_k^{-1} h_k^x) p_{x_k} - Q_k^{-1} w_k + h_k^{xT} R_k^{-1} \gamma_k = \mu \bar{\lambda}_k \quad (\text{B.50})$$

**Proposition 10.**  $\forall k = 1, \dots, t$

$$\begin{aligned} Q_k^{-1} f_{k-1}^x p_{x_{k-1}} - (Q_k^{-1} + h_k^{xT} R_k^{-1} h_k^x) p_{x_k} - Q_k^{-1} w_k + h_k^{xT} R_k^{-1} \gamma_k &= -P_k^{-1} (p_{x_k} - \hat{\mu}_k) \\ -P^{-1} (p_{x_0} + w_0) &= -P_0^{-1} (p_{x_0} - \hat{\mu}_0) \end{aligned} \quad (\text{B.51})$$

where  $P_k$  and  $\hat{\mu}_k$  are solutions of the forward recursion:

$$\begin{aligned} E_{k+1} &= P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx} \\ \bar{P}_{k+1} &= Q_{k+1} + f_k^x (P_k^{-1} - \mu \bar{\ell}_k^{xx})^{-1} f_k^{xT} \\ K_{k+1} &= \bar{P}_{k+1} h_{k+1}^{xT} (R_{k+1} + h_{k+1}^x \bar{P}_{k+1} h_{k+1}^{xT})^{-1} \\ P_{k+1} &= (I - K_{k+1} h_{k+1}^x) \bar{P}_{k+1} \\ \hat{\mu}_{k+1} &= (I - K_{k+1} h_{k+1}^x) (f_k^x \hat{\mu}_k - w_{k+1}) + K_{k+1} \gamma_{k+1} \\ &\quad + \mu P_{k+1} Q_{k+1}^{-1} f_k^x E_{k+1}^{-1} (\bar{\ell}_k^{xx} \hat{\mu}_k + \ell_k^x) \end{aligned} \quad (\text{B.52})$$

with the initialization:

$$\begin{aligned} P_0 &= P \\ \hat{\mu}_0 &= \hat{x}_0 - x_0^i \end{aligned} \quad (\text{B.53})$$

Furthermore:

$$\forall k = 0, \dots, t-1, \quad p_{x_k} = E_{k+1}^{-1} (f_k^{xT} Q_{k+1}^{-1} (w_{k+1} + p_{x_{k+1}}) + P_k^{-1} \hat{\mu}_k + \mu \ell_k^x) \quad (\text{B.54})$$

*Proof.* The initialization is clear:

$$P_0 = P \quad (\text{B.55})$$

$$\hat{\mu}_0 = -w_0 \quad (\text{B.56})$$

If the prop is true at time  $k$ , from (B.49) and (B.50), we must have  $\bar{\lambda}_k = -\mu^{-1}P_k^{-1}(p_{x_k} - \hat{\mu}_k)$ . Now let's show that the property holds for  $k+1$ . From (B.48), we have:

$$\begin{aligned} -\mu^{-1}P_k^{-1}(p_{x_k} - \hat{\mu}_k) &= \mu^{-1}f_k^x Q_{k+1}^{-1}f_k^x p_{k_k} - \bar{\ell}_k^{xx} p_{k_k} \\ &\quad - \mu^{-1}f_k^x Q_{k+1}^{-1}(w_{k+1} + p_{x_{k+1}}) - \ell_k^x \\ (P_k^{-1} + f_k^x Q_{k+1}^{-1}f_k^x - \mu\bar{\ell}_k^{xx}) p_{x_k} &= f_k^x Q_{k+1}^{-1}(w_{k+1} + p_{x_{k+1}}) + P_k^{-1}\hat{\mu}_k + \mu\ell_k^x \end{aligned}$$

Hence, we recover equation (B.54). Consequently,

$$\begin{aligned} &-Q_{k+1}^{-1}f_k^x p_{x_k} + (Q_{k+1}^{-1} + h_{k+1}^x{}^T R_{k+1}^{-1} h_{k+1}^x) p_{x_{k+1}} + Q_{k+1}^{-1}w_{k+1} - h_{k+1}^x{}^T R_{k+1}^{-1} \gamma_{k+1} = \\ &-Q_{k+1}^{-1}f_k^x (P_k^{-1} + f_k^x Q_{k+1}^{-1}f_k^x - \mu\bar{\ell}_k^{xx})^{-1} (f_k^x Q_{k+1}^{-1}(w_{k+1} + p_{x_{k+1}}) + P_k^{-1}\hat{\mu}_k + \mu\ell_k^x) \\ &+ (Q_{k+1}^{-1} + h_{k+1}^x{}^T R_{k+1}^{-1} h_{k+1}^x) p_{x_{k+1}} + Q_{k+1}^{-1}w_{k+1} - h_{k+1}^x{}^T R_{k+1}^{-1} \gamma_{k+1} \\ &= P_{k+1}^{-1}(p_{x_{k+1}} - \hat{\mu}_{k+1}) \end{aligned} \quad (\text{B.57})$$

where:

$$P_{k+1}^{-1} = Q_{k+1}^{-1} + h_{k+1}^x {}^T R_{k+1}^{-1} h_{k+1}^x - Q_{k+1}^{-1} f_k^x (P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx})^{-1} f_k^{xT} Q_{k+1}^{-1} \quad (\text{B.58})$$

$$= h_{k+1}^x {}^T R_{k+1}^{-1} h_{k+1}^x + \underbrace{(Q_{k+1} + f_k^x (P_k^{-1} - \mu \bar{\ell}_k^{xx})^{-1} f_k^{xT})^{-1}}_{:= \bar{P}_{k+1}^{-1}}$$

and:

$$\begin{aligned} & P_{k+1}^{-1} \hat{\mu}_{t+1} \\ &= -Q_{k+1}^{-1} w_{k+1} + h_{k+1}^x {}^T R_{k+1}^{-1} \gamma_{k+1} \\ &+ Q_{k+1}^{-1} f_k^x (P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx})^{-1} (f_k^{xT} Q_{k+1}^{-1} w_{k+1} + P_k^{-1} \hat{\mu}_k + \mu \ell_k^x) \\ &= -\left( Q_{k+1}^{-1} - Q_{k+1}^{-1} f_k^x (P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx})^{-1} f_k^{xT} Q_{k+1}^{-1} \right) w_{k+1} \\ &+ h_{k+1}^x {}^T R_{k+1}^{-1} \gamma_{k+1} + Q_{k+1}^{-1} f_k^x (P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx})^{-1} (P_k^{-1} \hat{\mu}_k + \mu \ell_k^x) \\ &= -\bar{P}_{k+1}^{-1} w_{k+1} + h_{k+1}^x {}^T R_{k+1}^{-1} \gamma_{k+1} \\ &+ Q_{k+1}^{-1} f_k^x (P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx})^{-1} (P_k^{-1} \hat{\mu}_k + \mu \ell_k^x) \end{aligned} \quad (\text{B.59})$$

Let's define:

$$\begin{aligned} K_{k+1} &:= (\bar{P}_{k+1}^{-1} + h_{k+1}^x {}^T R_{k+1}^{-1} h_{k+1}^x)^{-1} h_{k+1}^x {}^T R_{k+1}^{-1} \\ &= \bar{P}_{k+1} h_{k+1}^x {}^T (R_{k+1} + h_{k+1}^x \bar{P}_{k+1} h_{k+1}^x {}^T)^{-1} \end{aligned} \quad (\text{B.60})$$

$$E_{k+1} := P_k^{-1} + f_k^{xT} Q_{k+1}^{-1} f_k^x - \mu \bar{\ell}_k^{xx} \quad (\text{B.61})$$

We have:

$$\begin{aligned} (I - K_{k+1}h_{k+1}^x)\bar{P}_{k+1} &= (I - (\bar{P}_{k+1}^{-1} + h_{k+1}^{xT}R_{k+1}^{-1}h_{k+1}^x)^{-1}h_{k+1}^{xT}R_{k+1}h_{k+1}^x)\bar{P}_{k+1} \\ &= (\bar{P}_{k+1}^{-1} + h_{k+1}^{xT}R_{k+1}^{-1}h_{k+1}^x)^{-1} = P_{k+1} \end{aligned} \quad (\text{B.62})$$

and get:

$$\hat{\mu}_{t+1} = -(I - K_{k+1}h_{k+1}^x)w_{k+1} + K_{k+1}\gamma_{k+1} + P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}(P_k^{-1}\hat{\mu}_k + \mu\ell_k^x) \quad (\text{B.63})$$

but:

$$\begin{aligned} &P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}P_k^{-1} \\ &= P_{k+1}Q_{k+1}^{-1}f_k^x(P_k^{-1} + f_k^{xT}Q_{k+1}^{-1}f_k^x - \mu\bar{\ell}_k^{xx})^{-1}P_k^{-1} \quad (\text{B.64}) \\ &= P_{k+1}Q_{k+1}^{-1}f_k^x - P_{k+1}Q_{k+1}^{-1}f_k^x(P_k^{-1} + f_k^{xT}Q_{k+1}^{-1}f_k^x - \mu\bar{\ell}_k^{xx})^{-1}(f_k^{xT}Q_{k+1}^{-1}f_k^x - \mu\bar{\ell}_k^{xx}) \\ &= P_{k+1}\left(Q_{k+1}^{-1} - Q_{k+1}^{-1}f_k^x(P_k^{-1} + f_k^{xT}Q_{k+1}^{-1}f_k^x - \mu\bar{\ell}_k^{xx})^{-1}f_k^{xT}Q_{k+1}^{-1}\right)f_k^x \\ &+ P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}\mu\bar{\ell}_k^{xx} \\ &= P_{k+1}\left(Q_{k+1} + f_k^x(P_k^{-1} - \mu\bar{\ell}_k^{xx})^{-1}f_k^{xT}\right)^{-1}f_k^x + P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}\mu\bar{\ell}_k^{xx} \\ &= P_{k+1}\bar{P}_{k+1}^{-1}f_k^x + P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}\mu\bar{\ell}_k^{xx} \\ &= (I - K_{k+1}h_{k+1}^x)f_k^x + P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}\mu\bar{\ell}_k^{xx} \end{aligned}$$

Therefore,

$$\begin{aligned} \hat{\mu}_{k+1} &= (I - K_{k+1}h_{k+1}^x)(f_k^x\hat{\mu}_k - w_{k+1}) \\ &+ K_{k+1}\gamma_{k+1} + \mu P_{k+1}Q_{k+1}^{-1}f_k^xE_{k+1}^{-1}(\bar{\ell}_k^{xx}\hat{\mu}_k + \ell_k^x) \end{aligned} \quad (\text{B.65})$$

## B.5 Coupling

In this section, we study (B.23) and show that it allows us to couple the past and future stress:

- Case 1. From (B.23), if  $t = 0$  :

$$\begin{aligned} & (\bar{\ell}_0^{xx} - \mu^{-1}P_0^{-1} - \mu^{-1}f_0^{xT}Q_1^{-1}f_0^x) p_{x_0} + \mu^{-1}f_0^{xT}Q_1^{-1}p_{x_1} \\ & + (\bar{\ell}_0^{xu} - \mu^{-1}f_0^{xT}Q_1^{-1}f_0^u) p_{u_0} + \ell_0^x - \mu^{-1}P_0^{-1}w_0 + \mu^{-1}f_0^{xT}Q_1^{-1}w_1 = 0 \end{aligned} \quad (\text{B.66})$$

From (B.51), we have:

$$(\bar{\ell}_0^{xx} - \mu^{-1}f_0^{xT}Q_1^{-1}f_0^x) p_{x_0} + \mu^{-1}f_0^{xT}Q_1^{-1}p_{x_1} + (\bar{\ell}_0^{xu} - \mu^{-1}f_0^{xT}Q_1^{-1}f_0^u) p_{u_0} \quad (\text{B.67})$$

$$+ \ell_0^x + \mu^{-1}f_0^{xT}Q_1^{-1}w_1 = \mu^{-1}P_0^{-1}(p_{x_0} - \hat{\mu}_0) \quad (\text{B.68})$$

- Case 2. From (B.23), if  $t \geq 1$ :

$$\begin{aligned} & \mu^{-1}Q_t^{-1}f_{t-1}^{xT}p_{x_{t-1}} + (\bar{\ell}_t^{xx} - \mu^{-1}Q_t^{-1} - \mu^{-1}f_t^{xT}Q_{t+1}^{-1}f_t^x - \mu^{-1}h_t^{xT}R_t^{-1}h_t^x) p_{x_t} \\ & + \mu^{-1}f_t^{xT}Q_{t+1}^{-1}p_{x_{t+1}} \\ & + (\bar{\ell}_t^{xu} - \mu^{-1}f_t^{xT}Q_{t+1}^{-1}f_t^u) p_{u_t} \\ & + \ell_t^x - \mu^{-1}Q_t^{-1}w_t + \mu^{-1}f_t^{xT}Q_{t+1}^{-1}w_{t+1} + \mu^{-1}h_t^{xT}R_t^{-1}\gamma_t = 0 \end{aligned} \quad (\text{B.69})$$

and from (B.51), we have:

$$\begin{aligned} -Q_t^{-1} f_t^x p_{x_{t-1}} + (Q_t^{-1} + h_t^{xT} R_t^{-1} h_t^x) p_{x_t} \\ + Q_t^{-1} w_t - h_t^{xT} R_t^{-1} \gamma_t = P_t^{-1} (p_{x_t} - \hat{\mu}_t) \end{aligned}$$

Therefore, in both cases, we find that:

$$\begin{aligned} (\bar{\ell}_t^{xx} - \mu^{-1} f_t^{xT} Q_{t+1}^{-1} f_t^x) p_{x_t} + \mu^{-1} f_t^{xT} Q_{t+1}^{-1} p_{x_{t+1}} + (\bar{\ell}_t^{xu} - \mu^{-1} f_t^{xT} Q_{t+1}^{-1} f_t^u) p_{u_t} \\ + \ell_t^x + \mu^{-1} f_t^{xT} Q_{t+1}^{-1} w_{t+1} = \mu^{-1} P_t^{-1} (p_{x_t} - \hat{\mu}_t) \end{aligned} \quad (\text{B.70})$$

And from (B.34)

$$\begin{aligned} V_t p_{x_t} + v_t &= \bar{\ell}_t^{xx} p_{x_t} + \bar{\ell}_t^{xu} p_{u_k} + \ell_t^x + f_t^{xT} \lambda_{t+1} \\ &= \bar{\ell}_t^{xx} p_{x_t} + \bar{\ell}_t^{xu} p_{u_k} + \ell_t^x \\ &\quad + f_t^{xT} (\mu^{-1} Q_{t+1}^{-1} (p_{x_{t+1}} - f_t^x p_{x_t} - f_t^u p_{u_t} + w_{t+1})) \end{aligned}$$

Hence, Equation (B.70) becomes:

$$\begin{aligned} V_t p_{x_t} + v_t &= \mu^{-1} P_t^{-1} (p_{x_t} - \hat{\mu}_t) \\ \mu P_t V_t p_{x_t} + \mu P_t v_t &= p_{x_t} - \hat{\mu}_t \end{aligned}$$



Hence:

$$p_{x_t} = (P_t^{-1} - \mu V_t)^{-1} (P_t^{-1} \hat{\mu}_t + \mu v_t) \quad (\text{B.71})$$

# Bibliography

- [1] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023.
- [2] Anoushka Alavilli, Khai Nguyen, Sam Schoedel, Brian Plancher, and Zachary Manchester. Tinympc: Model-predictive control on resource-constrained microcontrollers. *arXiv preprint arXiv:2310.16985*, 2023.
- [3] Elisa Alboni, Gianluigi Grandesso, Gastone Pietro Rosati Papini, Justin Carpentier, and Andrea Del Prete. Cacto-sl: Using sobolev learning to improve continuous actor-critic with trajectory optimization. In *6th Annual Learning for Dynamics & Control Conference*, pages 1452–1463. PMLR, 2024.
- [4] Walid Amanhoud, Mahdi Khoramshahi, Maxime Bonnesoeur, and Aude Billard. Force Adaptation in Contact Tasks with Dynamical Systems. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6841–6847, 2020. ISSN 10504729. doi: 10.1109/ICRA40945.2020.9197509.
- [5] Yuichiro Aoyama, George Boutselis, Akash Patel, and Evangelos A Theodorou. Constrained differential dynamic programming revisited. In *2021 IEEE*

*International Conference on Robotics and Automation (ICRA)*, pages 9738–9744. IEEE, 2021.

- [6] K J Åström. *Adaptive Control*, pages 437–450. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991. ISBN 978-3-662-08546-2. doi: 10.1007/978-3-662-08546-2\_24.
- [7] Christopher Atkeson. Using local trajectory optimizers to speed up global optimization in dynamic programming. *Advances in neural information processing systems*, 6, 1993.
- [8] Alp Aydinoglu, Adam Wei, Wei-Cheng Huang, and Michael Posa. Consensus complementarity control for multi-contact mpc. *IEEE Transactions on Robotics*, 2024.
- [9] Antoine Bambade, Sarah El-Kazdadi, Adrien Taylor, and Justin Carpentier. PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond. In *RSS 2022 - Robotics: Science and Systems*, 2022.
- [10] Antoine Bambade, Sarah El-Kazdadi, Adrien Taylor, and Justin Carpentier. Prox-qp: Yet another quadratic programming solver for robotics and beyond. In *RSS 2022-Robotics: Science and Systems*, 2022.
- [11] Somil Bansal and Claire J Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2021.
- [12] Tamer Başar. Robust designs through risk sensitivity: An overview. *Journal of Systems Science and Complexity*, 34(5):1634–1665, 2021.

- [13] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.
- [14] Katrin Baumgärtner, Florian Messerer, and Moritz Diehl. A unified local convergence analysis of differential dynamic programming, direct single shooting, and direct multiple shooting. In *2023 European Control Conference (ECC)*, pages 1–7. IEEE, 2023.
- [15] Tamer Başar and Pierre Bernhard.  $H^\infty$ -Optimal control and related minimax design problems: A dynamic game approach. *IEEE Trans. Autom. Control.*, 41, 1996.
- [16] Maciej Bednarczyk, Hassan Omran, and Bernard Bayle. Model Predictive Impedance Control. *Proceedings - IEEE International Conference on Robotics and Automation*, (1):4702–4708, 2020. ISSN 10504729. doi: 10.1109/ICRA40945.2020.9196969.
- [17] Bradley M Bell. The iterated kalman smoother as a gauss–newton method. *SIAM Journal on Optimization*, 4(3):626–636, 1994.
- [18] Bradley M Bell and Frederick W Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Transactions on Automatic Control*, 38(2): 294–297, 1993.
- [19] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.
- [20] P. Bernhard. Minimax versus stochastic partial information control. In

- Proceedings of 1994 33rd IEEE Conference on Decision and Control*, volume 3, pages 2572–2577 vol.3, 1994. doi: 10.1109/CDC.1994.411532.
- [21] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.
  - [22] Dimitri Bertsekas. *Reinforcement learning and optimal control*, volume 1. Athena Scientific, 2019.
  - [23] Dimitri P Bertsekas. Dynamic programming and suboptimal control: A survey from adp to mpc. *European journal of control*, 11(4-5):310–334, 2005.
  - [24] Dimitri P Bertsekas. Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE transactions on neural networks and learning systems*, 28(3):500–509, 2015.
  - [25] Arun L Bishop, John Z Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. Relu-qp: A gpu-accelerated quadratic programming solver for model-predictive control. *arXiv preprint arXiv:2311.18056*, 2023.
  - [26] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.
  - [27] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1): 1–122, 2011.

- [28] Angelo Bratta, Michele Focchi, Niraj Rathod, and Claudio Semini. Optimization-Based Reference Generator for Nonlinear Model Predictive Control of Legged Robots. *Robotics*, 12(1):1–18, 2023. ISSN 22186581. doi: 10.3390/robotics12010006.
- [29] Rohan Budhiraja, Justin Carpentier, Carlos Mastalli, and Nicolas Mansard. Differential dynamic programming for multi-phase rigid contact dynamics. In *IEEE Humanoids*, 2018. doi: 10.1109/HUMANOIDS.2018.8624925.
- [30] Marco C Campi and Matthew R James. Nonlinear discrete-time risk-sensitive optimal control. *International Journal of Robust and Nonlinear Control*, 6(1):1–19, 1996.
- [31] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and systems (RSS 2018)*, 2018.
- [32] Justin Carpentier, Florian Valenza, Nicolas Mansard, et al. Pinocchio: fast forward and inverse dynamics for poly-articulated systems. <https://stack-of-tasks.github.io/pinocchio>, 2015–2021.
- [33] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiriaux, Olivier Stasse, and Nicolas Mansard. The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. *Proceedings of the 2019 IEEE/SICE International Symposium on System Integration, SII 2019*, pages 614–619, 2019. doi: 10.1109/SII.2019.8700380.
- [34] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiriaux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++

- library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.
- [35] Elliot Chane-Sane, Pierre-Alexandre Leziart, Thomas Flayols, Olivier Stasse, Philippe Souères, and Nicolas Mansard. Cat: Constraints as terminations for legged locomotion reinforcement learning. *arXiv preprint arXiv:2403.18765*, 2024.
- [36] Hong Chen and Frank Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.
- [37] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.
- [38] Stefano Chiaverini and Lorenzo Sciavicco. The Parallel Approach to Force/Position Control of Robotic Manipulators. *IEEE Transactions on Robotics and Automation*, 9(4):361–373, 1993. ISSN 1042296X. doi: 10.1109/70.246048.
- [39] David A Copp and Joao P Hespanha. Nonlinear output-feedback model predictive control with moving horizon estimation. In *53rd IEEE Conference on Decision and Control*, pages 3511–3517. IEEE, 2014.
- [40] David A Copp and Joao P Hespanha. Simultaneous nonlinear model predictive control and state estimation. *Automatica*, 77:143–154, 2017.
- [41] Eduardo Corral, R. Moreno, María J. Gómez García, and Cristina Castejón.

- Nonlinear phenomena of contact in multibody systems dynamics: a review. *Nonlinear Dynamics*, 104:1269 – 1295, 2021.
- [42] Henry Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on automatic control*, 9(1):5–12, 1964.
- [43] Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.
- [44] Ewen Dantec, Maximilien Naveau, Pierre Fernbach, Nahuel Villa, Guilhem Saurel, Olivier Stasse, Michel Taïx, and Nicolas Mansard. Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 638–644. IEEE, 2022.
- [45] Bolei Di and Andrew Lamperski. Newton’s method, bellman recursion and differential dynamic programming for unconstrained nonlinear dynamic games. *Dynamic Games and Applications*, pages 1–49, 2021.
- [46] Gianni Di Pillo, Luigi Grippo, and Francesco Lampariello. A class of structured quasi-newton algorithms for optimal control problems. In *Applications of Nonlinear Programming to Optimization and Control*, pages 101–107. Elsevier, 1984.
- [47] Moritz Diehl. Lecture notes on optimal control and estimation. *Lecture Notes on Optimal Control and Estimation*, 2014.
- [48] Moritz Diehl, Hans Georg Bock, Holger Diedam, Pierre Brice Wieber, Pierre-



- brice Wieber Fast, and Direct Multiple. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. page 28, 2009.
- [49] Alexander Dietrich, Xuwei Wu, Kristin Bussmann, Marie Harder, Maged Iskandar, Johannes Engelsberger, Christian Ott, and Alin Albu-Schäffer. Practical consequences of inertia shaping for interaction and tracking in robot control. *Control Engineering Practice*, 114, 2021. ISSN 0967-0661. doi: <https://doi.org/10.1016/j.conengprac.2021.104875>.
- [50] Badis Djeridane and John Lygeros. Neural approximation of pde solutions: An application to reachability computations. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3034–3039. IEEE, 2006.
- [51] CR Dohrmann and RD Robinett. Efficient sequential quadratic programming implementations for equality-constrained discrete-time optimal control. *Journal of Optimization Theory and Applications*, 95:323–346, 1997.
- [52] CR Dohrmann and RD Robinett. Dynamic programming method for constrained discrete-time optimal control. *Journal of Optimization Theory and Applications*, 101:259–283, 1999.
- [53] Alexander Domahidi, Aldo U Zgraggen, Melanie N Zeilinger, Manfred Morari, and Colin N Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pages 668–674. IEEE, 2012.
- [54] Joseph Duffy. The fallacy of modern hybrid control theory that is based on “orthogonal complements” of twist and wrench spaces. *Journal of Robotic Systems*, 7(2):139–144, 1990.

- [55] Joseph C Dunn and Dimitri P Bertsekas. Efficient dynamic programming implementations of newton’s method for unconstrained optimal control problems. *Journal of Optimization Theory and Applications*, 63(1):23–38, 1989.
- [56] Jonathan Eckstein and Michael C Ferris. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, 10(2):218–235, 1998.
- [57] Garry A Einicke and Langford B White. Robust extended kalman filtering. *IEEE Transactions on Signal Processing*, 47(9):2596–2599, 1999.
- [58] D Erickson, M Weber, and I Sharf. Contact Stiffness and Damping Estimation for Robotic Systems. *International Journal of Robotics Research*, 22(1):41–57, 2003.
- [59] Farbod Farshidian, Edo Jelavic, Asutosh Satapathy, Markus Giftthaler, and Jonas Buchli. Real-Time motion planning of legged robots: A model predictive control approach. *IEEE-RAS International Conference on Humanoid Robots*, pages 577–584, 2017. ISSN 21640580. doi: 10.1109/HUMANOIDS.2017.8246930.
- [60] Roy Featherstone. *Rigid Body Dynamics Algorithms*. 2008. ISBN 9780387743141. doi: 10.1007/978-1-4899-7560-7.
- [61] Hans Joachim Ferreau, Hans Georg Bock, and Moritz Diehl. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18(8):816–830, 2008.

- [62] Roger Fletcher and Sven Leyffer. Nonlinear programming without a penalty function. *Mathematical programming*, 91:239–269, 2002.
- [63] Janick V Frasch, Sebastian Sager, and Moritz Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical programming computation*, 7:289–329, 2015.
- [64] Gianluca Frison. General rights Algorithms and Methods for High-Performance Model Predictive Control. 2015. URL [www.compute.dtu.dk](http://www.compute.dtu.dk).
- [65] Gianluca Frison and Moritz Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, 53(2):6563–6569, 2020.
- [66] Gianluca Frison, Dimitris Kouzoupis, Tommaso Sartor, Andrea Zanelli, and Moritz Diehl. Blasfeo: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–30, 2018.
- [67] Ahmad Gazar, Majid Khadiv, Andrea Del Prete, and Ludovic Righetti. Stochastic and robust mpc for bipedal locomotion: A comparative study on robustness and performance. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 61–68. IEEE, 2021.
- [68] Markus Gifftthaler, Michael Neunert, Markus Stäuble, Jonas Buchli, and Moritz Diehl. A family of iterative gauss-newton shooting methods for nonlinear optimal control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

- [69] Tobias Gold, Andreas Völz, and Knut Graichen. Model predictive interaction control for robotic manipulation tasks. *IEEE Transactions on Robotics*, 39(1):76–89, 2023. doi: 10.1109/TRO.2022.3196607.
- [70] Gianluigi Grandesso, Elisa Alboni, Gastone P Rosati Papini, Patrick M Wensing, and Andrea Del Prete. Cacto: Continuous actor-critic with trajectory optimization—towards global optimality. *IEEE Robotics and Automation Letters*, 8(6):3318–3325, 2023.
- [71] Ruben Grandia, Farbod Farshidian, René Ranftl, and Marco Hutter. Feedback mpc for torque-controlled legged robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4730–4737. IEEE, 2019.
- [72] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive Locomotion Through Nonlinear Model-Predictive Control. *IEEE Transactions on Robotics*, 39(5):3402–3421, 2023. ISSN 19410468. doi: 10.1109/TRO.2023.3275384.
- [73] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 39(5):3402–3421, 2023.
- [74] Pascal Grieder, Francesco Borrelli, Fabio Torrisi, and Manfred Morari. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40(4):701–708, 2004.
- [75] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier,

- Thomas Flayols, et al. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2): 3650–3657, 2020.
- [76] Lars Grune and Anders Rantzer. On the infinite horizon performance of receding horizon controllers. *IEEE Transactions on Automatic Control*, 53(9):2100–2111, 2008.
- [77] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [78] Bilal Hammoud, Armand Jordana, and Ludovic Righetti. irisc: Iterative risk sensitive control for nonlinear systems with imperfect observations. *arXiv preprint arXiv:2110.06700*, 2021.
- [79] Ankur Handa, Arthur Allshire, Viktor Makoviyshuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984. IEEE, 2023.
- [80] Nathan Hatch and Byron Boots. The value of planning for infinite-horizon model predictive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7372–7378. IEEE, 2021.
- [81] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimmering, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse

- dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40:473–491, 2016.
- [82] Ali Heydari. Revisiting approximate dynamic programming and its convergence. *IEEE transactions on cybernetics*, 44(12):2733–2743, 2014.
- [83] David Hoeller, Farbod Farshidian, and Marco Hutter. Deep value model predictive control. In *Conference on Robot Learning*, pages 990–1004. PMLR, 2020.
- [84] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.
- [85] Neville Hogan. Impedance Control Part1-3. *Transaction of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 107(March 1985): 1–24, 1985.
- [86] Neville Hogan. Contact and Physical Interaction. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):1–25, 2022. ISSN 2573-5144. doi: 10.1146/annurev-control-042920-010933.
- [87] Taylor A Howell, Brian E Jackson, and Zachary Manchester. ALTRO: A fast solver for constrained trajectory optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679. IEEE, 2019.
- [88] Bei Hu and Arno Linnemann. Toward infinite-horizon optimality in nonlinear model predictive control. *IEEE Transactions on Automatic Control*, 47(4): 679–682, 2002.

- [89] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 38–44. IEEE, 2016.
- [90] Syed Aseem Ul Islam and Dennis S Bernstein. Recursive least squares for real-time implementation [lecture notes]. *IEEE Control Systems Magazine*, 39(3):82–85, 2019.
- [91] David Jacobson. Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games. *IEEE Transactions on Automatic control*, 18(2):124–131, 1973.
- [92] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. Constrained differential dynamic programming: A primal-dual augmented lagrangian approach. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13371–13378. IEEE, 2022.
- [93] Wilson Jallet, Nicolas Mansard, and Justin Carpentier. Implicit differential dynamic programming. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1455–1461. IEEE, 2022.
- [94] Matthew R James, John S Baras, and Robert J Elliott. Risk-sensitive control and dynamic games for partially observed discrete-time nonlinear systems. *IEEE transactions on automatic control*, 39(4):780–792, 1994.
- [95] Armand Jordana, Sébastien Kleff, Avadesh Meduri, Justin Carpentier, Nicolas

- Mansard, and Ludovic Righetti. Stagewise implementations of sequential quadratic programming for model-predictive control. *Preprint*, 2023.
- [96] Seul Jung, T. C. Hsia, and R. G. Bonitz. Force tracking impedance control for robot manipulators with an unknown environment: Theory, simulation, and experiment. *International Journal of Robotics Research*, 20(9):765–774, 2001. ISSN 02783649. doi: 10.1177/02783640122067651.
- [97] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- [98] Sotaro Katayama, Masaki Murooka, and Yuichi Tazaki. Model predictive control of legged and humanoid robots: models and algorithms. *Advanced Robotics*, 37(5):298–315, 2023.
- [99] Khalid J. Kazim, Johanna Bethge, Janine Matschek, and Rolf Findeisen. Combined Predictive Path Following and Admittance Control. *Proceedings of the American Control Conference*, 2018-June:3153–3158, 2018. ISSN 07431619. doi: 10.23919/ACC.2018.8431272.
- [100] Marc D. Killpack, Ariel Kapusta, and Charles C. Kemp. Model predictive control for fast reaching in clutter. *Autonomous Robots*, 40(3):537–560, 2016. ISSN 15737527. doi: 10.1007/s10514-015-9492-6.
- [101] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.



- [102] Sébastien Kleff, Avadesh Meduri, Rohan Budhiraja, Nicolas Mansard, and Ludovic Righetti. High-frequency nonlinear model predictive control of a manipulator. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7330–7336. IEEE, 2021.
- [103] Sébastien Kleff, Ewen Dantec, Guilhem Saurel, Nicolas Mansard, and Ludovic Righetti. Introducing force feedback in model predictive control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13379–13385, 2022. doi: 10.1109/IROS47612.2022.9982003.
- [104] Jonas Koenemann, Andrea Del Prete, Yuval Tassa, Emanuel Todorov, Olivier Stasse, Maren Bennewitz, and Nicolas Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3346–3351. IEEE, 2015.
- [105] Milan Korda, Didier Henrion, and Colin N Jones. Controller design and value function approximation for nonlinear dynamical systems. *Automatica*, 67: 54–66, 2016.
- [106] Dimitris Kouzoupis, Gianluca Frison, Andrea Zanelli, and Moritz Diehl. Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, 46(4):863–882, 2018.
- [107] Arthur J Krener. Adaptive horizon model predictive control and al’brekht’s method. In *Encyclopedia of Systems and Control*, pages 27–40. Springer, 2021.
- [108] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai

- Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016.
- [109] Benoit Landry, Hongkai Dai, and Marco Pavone. Seagul: Sample efficient adversarially guided learning of value functions. In *Learning for Dynamics and Control*, pages 1105–1117. PMLR, 2021.
- [110] Andreas Lawitzky, Anselm Nicklas, Dirk Wollherr, and Martin Buss. Determining states of inevitable collision using reachability analysis. *IEEE International Conference on Intelligent Robots and Systems*, pages 4142–4147, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6943146.
- [111] WT Lee. Tridiagonal matrices: Thomas algorithm. *MS6021, Scientific Computation, University of Limerick*, 2011.
- [112] Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013.
- [113] Albert H Li, Preston Culbertson, Vince Kurtz, and Aaron D Ames. Drop: Dexterous reorientation via online planning. *arXiv preprint arXiv:2409.14562*, 2024.
- [114] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229. Citeseer, 2004.
- [115] Li-zhi Liao and Christine A Shoemaker. Advantages of differential dynamic programming over newton’s method for discrete-time optimal control problems. Technical report, Cornell University, 1992.

- [116] TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [117] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.
- [118] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2007.
- [119] Josep Marti-Saumell, Joan Solà, Carlos Mastalli, and Angel Santamaria-Navarro. Squash-box feasibility driven differential dynamic programming. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7637–7644. IEEE, 2020.
- [120] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020.
- [121] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [122] Carlos Mastalli, Wolfgang Merkt, Josep Marti-Saumell, Henrique Ferrolho,

- Joan Solà, Nicolas Mansard, and Sethu Vijayakumar. A feasibility-driven approach to control-limited ddp. *Autonomous Robots*, 46(8):985–1005, 2022.
- [123] Janine Matschek, Johanna Bethge, Pablo Zometa, and Rolf Findeisen. Force Feedback and Path Following using Predictive Control: Concept and Application to a Lightweight Robot. *IFAC-PapersOnLine*, 50(1):9827–9832, 2017. ISSN 2405-8963. doi: 10.1016/J.IFACOL.2017.08.898.
- [124] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, jun 2000. ISSN 0005-1098. doi: 10.1016/S0005-1098(99)00214-9.
- [125] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.
- [126] David Q Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.
- [127] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.
- [128] D.Q. Mayne. Optimization in model based control. In *Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes (DyCORD’95)*, IFAC Postprint Volume, pages 229–242. Pergamon, Oxford, 1995. ISBN 978-0-08-042368-5. doi: <https://doi.org/10.1016/B978-0-08-042368-5.50041-1>. URL <https://www.sciencedirect.com/science/article/pii/B9780080423685500411>.

- [129] Avadesh Meduri, Paarth Shah, Julian Viereck, Majid Khadiv, Ioannis Havoutis, and Ludovic Righetti. Biconmp: A nonlinear model predictive control framework for whole body motion planning. *IEEE Transactions on Robotics*, 2023.
- [130] Maria Vittoria Minniti, Ruben Grandia, Kevin F  h, Farbod Farshidian, and Marco Hutter. Model predictive robot-environment interaction control for mobile manipulation tasks. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1651–1657, 2021.
- [131] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [132] Igor Mordatch and Emo Todorov. Combining the benefits of function approximation and trajectory optimization. In *Robotics: Science and Systems*, volume 4, page 23, 2014.
- [133] J. Morimoto, G. Zeglin, and C.G. Atkeson. Minimax differential dynamic programming: application to a biped walking robot. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 2, pages 1927–1932, 2003. doi: 10.1109/IROS.2003.1248926.
- [134] Jun Morimoto and Christopher Atkeson. Minimax differential dynamic programming: An application to robust biped walking. *Advances in neural information processing systems*, 15, 2002.
- [135] H Mukai, A Tanikawa, I Tunay, IA Ozcan, IN Katz, H Schattler, P Rinaldi,

- GJ Wang, L Yang, and Y Sawada. Game-theoretic linear quadratic method for air mission control. In *39th IEEE Conference on Decision and Control*, volume 3, pages 2574–2580. IEEE, 2000.
- [136] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.
- [137] DM Murray and SJ Yakowitz. Differential dynamic programming and newton’s method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, 1984.
- [138] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.
- [139] Michael Neunert, Markus Stäuble, Markus Gifftthaler, Carmine D Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.
- [140] W. S. Newman. Stability and performance limits of interaction controllers. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 114(4):563–570, 1992. ISSN 15289028. doi: 10.1115/1.2897725.
- [141] John Ndegwa Nganga, He Li, and Patrick Wensing. Second-order differential dynamic programming for whole-body mpc of legged robots. In *Embracing Contacts-Workshop at ICRA 2023*, 2023.
- [142] Bethany L Nicholson, Wei Wan, Shivakumar Kameswaran, and Lorenz T Biegler. Parallel cyclic reduction strategies for linear systems that arise in dy-

- namic optimization problems. *Computational Optimization and Applications*, 70:321–350, 2018.
- [143] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.
- [144] Brendan O’Donoghue, Giorgos Stathopoulos, and Stephen Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6):2432–2442, 2013.
- [145] Artemiy Oleinikov, Sergey Soltan, Zarema Balgabekova, Alberto Bemporad, and Matteo Rubagotti. Scenario-based model predictive control with probabilistic human predictions for human–robot coexistence. *Control Engineering Practice*, 142(November 2023):105769, 2024. ISSN 09670661. doi: 10.1016/j.conengprac.2023.105769. URL <https://doi.org/10.1016/j.conengprac.2023.105769>.
- [146] JFA De O Pantoja and DQ Mayne. Sequential quadratic programming algorithm for discrete optimal control problems with control inequality constraints. *International Journal of Control*, 53(4):823–836, 1991.
- [147] Amit Parag, Sebastien Kleff, Leo Saci, Nicolas Mansard, and Olivier Stasse. Value learning from trajectory optimization and sobolev descent: A step toward reinforcement learning with superlinear convergence properties. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1410–1416, 2022. ISSN 10504729. doi: 10.1109/ICRA46639.2022.9811993.
- [148] Brian Plancher, Zachary Manchester, and Scott Kuindersma. Constrained un-

- scented dynamic programming. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5674–5680. IEEE, 2017.
- [149] Marc Raibert and J J Craig. Hybrid Position / Force Control of Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(June 1981): 126–133, 1981.
- [150] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99(3):723–757, 1998. ISSN 00223239. doi: 10.1023/A:1021711402723.
- [151] Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.
- [152] Luis A Rodriguez and Athanasios Sideris. An active set method for constrained linear quadratic optimal control. In *Proceedings of the 2010 American Control Conference*, pages 5197–5202. IEEE, 2010.
- [153] Angel Romero, Robert Penicka, and Davide Scaramuzza. Time-Optimal Online Replanning for Agile Quadrotor Flight. *IEEE Robotics and Automation Letters*, 7(3):7730–7737, 2022. ISSN 23773766. doi: 10.1109/LRA.2022.3185772.
- [154] Nicholas Rotella, Alexander Herzog, Stefan Schaal, and Ludovic Righetti. Humanoid momentum estimation using sensed contact wrenches. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 556–563. IEEE, 2015.
- [155] Simo Särkkä and Ángel F García-Fernández. Temporal parallelization of



- dynamic programming and linear quadratic control. *IEEE Transactions on Automatic Control*, 68(2):851–866, 2022.
- [156] Marie Schumacher, Janis Wojtusich, Philipp Beckerle, and Oskar von Stryk. An introductory review of active compliant control. *Robotics and Autonomous Systems*, 119:185–200, 2019. ISSN 09218890. doi: 10.1016/j.robot.2019.06.009.
- [157] Roland Schwan, Yuning Jiang, Daniel Kuhn, and Colin N Jones. Piqp: A proximal interior-point quadratic programming solver. *arXiv preprint arXiv:2304.00290*, 2023.
- [158] Homayoun Seraji. ADAPTIVE ADMITTANCE CONTROL: An Approach to Explicit Force Control. *Jet Propulsion*, pages 2705–2712, 1994.
- [159] Homayoun Seraji and Richard Colbaugh. Force tracking in impedance control. *International Journal of Robotics Research*, 16(1):97–117, 1997. ISSN 02783649. doi: 10.1177/027836499701600107.
- [160] Bruno Siciliano. Parallel force/position control of robot manipulators. In *Robotics Research*, pages 78–89. Springer London, 1996. ISBN 978-1-4471-1021-7.
- [161] Athanasios Sideris and James E Bobrow. An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2275–2280. IEEE, 2005.
- [162] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda

- Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [163] Sumeet Singh, Jean-Jacques Slotine, and Vikas Sindhwani. Optimizing trajectories with closed-loop dynamic sqp. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5249–5254. IEEE, 2022.
- [164] Jean-Pierre Sleiman, Farbod Farshidian, Maria Vittoria Minniti, and Marco Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6(3):4688–4695, 2021.
- [165] Giorgos Stathopoulos, Milan Korda, and Colin N Jones. Solving the infinite-horizon constrained lqr problem using accelerated dual proximal methods. *IEEE Transactions on Automatic Control*, 62(4):1752–1767, 2016.
- [166] Marc C Steinbach. *A structured interior point SQP method for nonlinear optimal control problems*. Springer, 1994.
- [167] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.
- [168] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [169] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

- [170] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.
- [171] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3): 52–57, 2002.
- [172] Firdaus E. Udwadia and Robert E. Kalaba. A new perspective on constrained motion. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1906):407–410, 1992. ISSN 0962-8444. doi: 10.1098/RSPA.1992.0158.
- [173] Lander Vanroye, Ajay Sathya, Joris De Schutter, and Wilm Decré. Fat-rop: A fast constrained optimal control problem solver for robot trajectory optimization and control. *arXiv preprint arXiv:2303.16746*, 2023.
- [174] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.
- [175] Julian Viereck, Avadesh Meduri, and Ludovic Righetti. Valuenetqp: Learned one-step optimal control for legged locomotion. In *Learning for Dynamics and Control Conference*, pages 931–942. PMLR, 2022.
- [176] Nahuel A Villa and Pierre-Brice Wieber. Model predictive control of biped walking with bounded uncertainties. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 836–841. IEEE, 2017.

- [177] Luigi Villani and Joris De Schutter. *Force Control*, pages 161–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5\_8.
- [178] Richard Volpe and Pradeep Khosla. Theoretical and experimental investigation of explicit force control strategies for manipulators. *IEEE Transactions on Automatic Control*, 38(11):1634–1650, 1993. ISSN 00189286. doi: 10.1109/9.262033.
- [179] Henning U Voss, Jens Timmer, and Jürgen Kurths. Nonlinear dynamical system identification from uncertain and indirect measurements. *International Journal of Bifurcation and Chaos*, 14(06):1905–1933, 2004.
- [180] Andreas Wächter and Lorenz T. Biegler. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, volume 106. 2006. ISBN 1010700405. doi: 10.1007/s10107-004-0559-y.
- [181] Arne Wahrburg and Kim Listmann. MPC-based admittance control for robotic manipulators. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 7548–7554, 2016. doi: 10.1109/CDC.2016.7799435.
- [182] Jiayi Wang, Sanghyun Kim, Teguh Santoso Lembono, Wenqian Du, Jaehyun Shim, Saeid Samadi, Ke Wang, Vladimir Ivan, Sylvain Calinon, Sethu Vijayakumar, et al. Online multi-contact receding horizon planning via value function approximation. *IEEE Transactions on Robotics*, 2024.
- [183] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on control systems technology*, 18(2):267–278, 2009.

- [184] D. Whitney. Historical perspective and state of the art in robot force control. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 262–268, 1985. doi: 10.1109/ROBOT.1985.1087266.
- [185] Daniel E Whitney. Force feedback control of manipulator fine motions. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 99(2):91–97, 1977. ISSN 15289028. doi: 10.1115/1.3427095.
- [186] Peter Whittle. Risk-sensitive linear/quadratic/gaussian control. *Advances in Applied Probability*, 13(4):764–777, 1981.
- [187] Stephen J Wright. Solution of discrete-time optimal control problems on parallel computers. *Parallel Computing*, 16(2-3):221–237, 1990.
- [188] Stephen J Wright. Partitioned dynamic programming for optimal control. *SIAM Journal on optimization*, 1(4):620–642, 1991.
- [189] Stephen J Wright. Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, 77(1):161–187, 1993.
- [190] Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. *arXiv preprint arXiv:2409.15610*, 2024.
- [191] Alan Yang and Stephen Boyd. Value-gradient iteration with quadratic approximate value functions. *Annual Reviews in Control*, 56:100917, 2023.
- [192] William Yang and Michael Posa. Impact-invariant control: Maximizing control authority during impacts. *arXiv preprint arXiv:2303.00817*, 2023.

- [193] Tsuneo Yoshikawa. Force control of robot manipulators. *Proceedings - IEEE International Conference on Robotics and Automation*, 1(April):220–226, 2000. ISSN 10504729. doi: 10.1109/robot.2000.844062.
- [194] Linrui Zhang, Li Shen, Long Yang, Shixiang Chen, Bo Yuan, Xueqian Wang, and Dacheng Tao. Penalized proximal policy optimization for safe reinforcement learning. *arXiv preprint arXiv:2205.11814*, 2022.
- [195] Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 100–107. IEEE, 2013.
- [196] Simon Zimmermann, Roi Poranne, and Stelian Coros. Dynamic manipulation of deformable objects with implicit integration. *IEEE Robotics and Automation Letters*, 6(2):4209–4216, 2021.