Controlling and Learning Constrained Motions for Manipulation in Contact

João Miguel Pousa de Moura

Submitted for the degree of

Doctor of Philosophy

Heriot-Watt University

School of Engineering and Physical Sciences

Awarded Jointly With

The University of Edinburgh

July, 2021

Abstract

Many practical tasks in robotic systems involving contact interaction with the environment, such as cleaning windows, writing or grasping, are inherently constrained, in that both the task and the environment impose constraints on the robot's motion. While constraints from manipulation motions in contact represent a challenge when modelling and controlling such robotic systems, they might also be an opportunity, if exploited for decomposing complex controllers into simpler ones that are easier to design, implement, test and even learn from data.

Modelling such systems requires incorporating these constraints in the robot's dynamic model. In this thesis, I define the class of Task-based Constraints (TbCs) and prove that the forward dynamic models of a constrained system obtained through the Projected Dynamics (PD) and the Operational Space Formulation (OSF) are equivalent. Establishing such equivalence required: reformulating the PD constraint inertia matrix, generalizing all its previous distinct algebraic variations; and generalizing the OSF to rank deficient constraint Jacobian matrices. This generalization allows us to numerically handle redundant constraints and singular configurations, without having to use different controllers in the vicinity of such configurations.

Furthermore, I show that we can recover both operational space control with constraints and the hybrid position/force control in the operational space from a multiple Task-based Constraint abstraction. I then propose a control and trajectory tracking approach for wiping the train cab front panels, using a velocity controlled robotic manipulator and a force/torque sensor attached to its end-effector, without using any surface model or vision-based surface detection. The control strategy consists of a hybrid position/force controller, adapted from the Operational Space Formulation, that aligns the cleaning tool with the surface normal, maintaining a set-point normal force, while simultaneously moving along the surface. The trajectory tracking strategy consists of specifying and tracking a two dimensional path that, when projected onto the train surface, corresponds to the desired pattern of motion. An experiment with the Baxter robot to wipe a highly curved surface with both a spiral and a raster scan motion patterns validates the approach. I also implemented the same approach in a scaled robot prototype, specifically designed to wipe a 1/8 scaled version of a train cab front, using a raster scan pattern.

Learning these type of control policies subject to constraints is a challenging problem. This thesis proposes a Constraint-aware Policy Learning (CaPL) method that solves the policy learning problem on redundant robots which execute a policy acting in the null-space of a constraint. This learning approach allows the generalization of learnt control policies across constraints that are unknown during the training phase. The CaPL method splits the combined problem of learning constraints and policies into: first estimating the constraint, and then estimating an unconstrained policy using the remaining degrees of freedom. For a linear parametrization, there

is a closed-form solution for the problem of estimating constraints based on Singular Value Decomposition (SVD). In this thesis, I propose another closed-form solution for constraint estimation for the TbC case, which includes estimating the task component without affecting the norm of the constraint matrix, based on Generalized Singular Value Decomposition (GSVD). I also discuss a metric for comparing the similarity of estimated constraints, which is useful to pre-process the trajectories recorded in the demonstrations. An experiment consisting in: learning a wiping task from human demonstration on flat surfaces; and reproducing it on an unknown curved surface using a force/torque based controller, to achieve tool alignment, validates the CaPL method. Despite the differences between the training and validation scenarios, the learnt policy still provides the desired wiping motion.

Dedico esta etapa à minha mãe e ao meu pai,
Maria da Assunção e Edgar Firmino,
por todo o amor, dedicão e amparo.

Acknowledgements

When coming to the conclusion of this last four years journey, I ought to recognize the ones that in so many ways contributed to my accomplishments and propelled me forward along the way.

First, I would like to thank my supervisors Mustafa Suphi Erden, for all his wise advice and the most thorough reviews, and Sethu Vijayakumar, for welcoming me in his research group and providing me all the support in pursuing my research.

I would like to acknowledge the School of Engineering & Physical Sciences (EPS) at Heriot-Watt University for awarding me the Postgraduate Research James Watt Scholarship to fund my studies. I would also like to acknowledge the Edinburgh Centre for Robotics (ECR) for accepting me in the Centre for Doctoral Training in Robotics and Autonomous Systems (CDT-RAS) programme, for supporting my research and enriching my overall experience as PhD student, with countless events and training opportunities. Many are the people responsible for the ECR success story, but I have particular appreciation for Anne Murphy, who made my life so much easier by helping me navigate the complex administrative mazes, and the ECR founders and directors, David Lane and Sethu Vijayakumar, who taught me a great deal in all the executive meetings I have attended in the past four years, as the 2015 cohort students rep.

I would like to thank all my colleagues from the CDT 2015 cohort, for creating such a community like environment, my colleagues from the Institute of Perception, Action and Behaviour (IPAB) at The University of Edinburgh, for integrating me in the institute and providing me with the opportunity for attending numerous events and engaging talks from its members and, finally, to all my colleagues from the Statistical Machine Learning and Motor Control Group (SLMC), for welcoming me in all the meetings, labs and research discussions, teaching me so much about robotics and profoundly impacting my growth as a researcher. Some of these colleagues became very close friends. A very special thanks to Tatiana López-Guevara and Theodoros Stouraitis, for the companionship and friendship, when getting through the downs and ups of the PhD, and for our countless chats about research and life.

Outside of academia, I would like to praise my very dear friend and travelling companion Petchroi Petchreing, for all the kindness in constantly presenting me with words of optimism and encouragement. I would like to thank my Portuguese best friends Tiago Silva and Sandra Monteiro, for their longstanding friendship and always being there for me. I would like to express my gratitude to all my extended family — aunts, uncles, cousins and grandparents — for the kind of attachment that only family can give, making my life complete and meaningful. Finally, I would greatly like to thank my brother Luís Pedro, my father Edgar Firmino and my mother Maria da Assunção, for all their love and unconditional support and dedication. To them I dedicate all my work and successes.

Contents

List of Acronyms

CaPL Constraint-aware Policy Learning.
CCL Constraint Consistent Learning.
CCPE Constraint Consistent Policy Error.
CPE Constrained Policy Error.
CSE Constraint Space Error.

DMP Dynamic Movement Primitive.
DoF Degrees of Freedom.
DPE Direct Policy Error.
DPL Direct Policy Learning.

EVD Eigenvalue Decomposition.

GEVD Generalized Eigenvalue Decomposition.
GMM Gaussian Mixture Model.
GSVD Generalized Singular Value Decomposition.

LfD Learning from Demonstration.
LS Least Squares.
LWPR Locally Weighted Projection Regression.

MSE Mean Square Error.

nCPE normalized Constrained Policy Error.
nDPE normalized Direct Policy Error.
NSPE Null Space Policy Error.
nUPE normalized Unconstrained Policy Error.

OSF Operational Space Formulation.

PbD Programming by Demonstration.
PD Projected Dynamics.
PID Proportional-Integral-Derivative.
POE Projected Observation Error.
ProMP Probabilistic Movement Primitive.

QP Quadratic Programming.

SVD Singular Value Decomposition.

TbC Task-based Constraint.

UPE Unconstrained Policy Error.

WLS Weighted Least Squares.

List of Symbols

The following list describes the notation and symbols used throughout this thesis.

$[]_\times$ transformation of a vector in a skew-symmetric matrix, page 55

β control policy parameters, see equation (4.1), page 81

$\ddot{q}$ configuration acceleration or vector of generalized accelerations, page 16

$\ddot{q}_\star$ unconstrained configuration acceleration, page 16

$\ddot{x}$ task-space acceleration, page 18

Δt sampling interval, page 83

$\dot{A}(\cdot)$ time derivative of the constraint Jacobian, page 15

$\dot{q}$ configuration velocity or vector of generalized velocities, page 14

$\dot{q}_\varepsilon$ arbitrary configuration acceleration, page 16

$\dot{q}_\varepsilon$ arbitrary configuration velocity, page 14

$\dot{x}(\cdot)$ task-space velocity, see equation (2.20), page 20

$\hat{\beta}$ estimated control policy paramters, see equation (4.3), page 82

$\hat{\boldsymbol{\pi}}$ estimated control policy, see equation (4.4), page 82

$\kappa(\cdot)$ condition number, page 23

λ task-space external force, page 18

$\langle\,,\rangle$ Euclidean inner product, page 15

$\langle\,,\rangle_M$ Inertia-weighted inner product, page 15

$\mathbb{R}$ set of real numbers, page 14

$\mathbb{S}^n_{++}$ set of symmetric positive definite matrices, page 16

$\mathbb{Z}$ set of integer numbers, page 82

$\mathcal{G}(\cdot)$ Gauss's function, see equation (2.10), page 17

$\mathcal{X}$ dataset containing pairs of observed states and actions, see equation (4.5), page 82

μ mean value, page 68

Ω selection matrix, see equation (3.1), page 43

$\otimes$ Kronecker product operator, see equation (4.13), page 84

$\overline{\omega}_m$ importance weighting for model m, page 83

$\overline{A}$ inertia-weighted generalized inverse of A, page 15

$\phi(\cdot)$ contraint forward model, see equation (2.1), page 14

σ standard deviation value, page 68

τ vector of generalized forces, page 16

τ_ε aribitrary generalized force, page 18

$\boldsymbol{\pi}(\cdot)$ control policy function, see equation (4.1), page 81

$\varepsilon(\cdot)$ error metric, see equation (4.4), page 82

$\varsigma(\cdot)$ singular values, page 23

$A(\cdot)$ constraint Jacobian, see equation (2.2), page 14

$A^\dagger$ Moore-Penrose inverse of A, page 15

$A_1^\#$ partial dynamically consistent inverse of A_1, page 29

$b(\cdot)$ constraint-space velocity, see equation (2.2), page 14

$c(\cdot)$ constraint-space acceleration, see equation (2.5), page 15

f task-space force, page 18

G generalized inverse of A, page 14

$h(\cdot)$ Coriolis, centrifugal and gravitational generalized force contribution, page 16

I_n $n \times n$ identity matrix, page 14

$M(\cdot)$ robot inertia matrix, page 16

M_c constraint inertia matrix, see equation (2.26), page 21

M_x task-space inertia matrix, see equation (2.16), page 18

n_b constraint-space or task-space dimensionality, page 14

n_q number of generalized coordinates or configuration dimensionality, page 14

P Orthogonal projection matrix, equivalent to P_I, page 15

P_M	inertia-weighted projection matrix, page 15
q	robot configuration or vector of generalized coordinates, page 14
R	Rotation matrix, see equation (3.14), page 54
s	robot state, see equation (4.1), page 81
t	time variable, page 14
u	robot commands or actions, see equation (4.1), page 81
$x(\cdot)$	task-space position, page 19

List of Figures

List of Tables

Chapter 1

Introduction

"Art lives from constraints and dies from freedom"

Leonardo da Vinci

This chapter introduces the thematic of the constrained nature of motions in contact and its relevance to the field of robotics. It states the goal and original contributions of the thesis and, finally, presents the thesis outline.

1.1 Motivation

Motions in Contact

Our unique ability to physically interact with the surrounding environment is one of those key skills that we strongly rely upon in most of our daily life activities, such as: walking, where we use the contact with the floor to propel our body forward; moving and manipulating all sorts of objects; assisting our mobility when, for instance, partially supporting our weight in an handrail; or even in activities that involve motion in permanent contact with the environment, such as cleaning a surface or ironing our clothes. Therefore, in our quest for taking robots out of the factories into more unstructured environments, we must empower them with similar capabilities, so they can also explore and exploit the world through physical interactions.

Physical interactions require robotic agents to have the capability of establishing and maintaining contact with the surroundings, which poses great robotic research challenges. Some of those challenges, widely explored in the robotics literature, range from contact sensing [16] and perception, to control [77, 167], planning [122] and learning [86, 127]; "proper representation of the contact mechanics for modeling and dynamic simulation of multi-body systems is still a challenge", according to Flores and Lankarani [56]. Vukobratovic et al. [168] categorizes *essential force tasks* as the

contact tasks whose very nature requires the robot end-effector (i.e. last link or tool) to establish physical contact with the environment and exert a given specific force, therefore, requiring simultaneous control of position and interaction force, stating that "The future will certainly hold more tasks for which the interaction with the environment is fundamental".

Constrained Motions

One way of regarding those interactions is as constrained motions, i.e., when interacting with the environment we are effectively restricting the range of motion of our limbs by constraining their position and orientation to the object or surface we are interacting with. For example, when holding a cup of water, we restrict our hand motion to prevent spilling the water by avoiding squeezing or rotating the cup. The same goes for when grabbing the handrail while going up the stairs — we limit our arm motions by fixing the hand position in a specific point of the handrail.

However, constraints can encompass more than just contacts. Kanoun et al. [75] use the same expression to represent constraints (as in rigid-body contacts) and tasks, and mention that this distinction "is just a question of context", which is the same terminology adopted by Dehio [36]. Sentis and Khatib [152] highlight the similarities between the constrained multi-body dynamics problem and the operational space formulation (that deals with tasks), in that both share a common mathematical description. This thesis follows the line of research unifying the treatment of kinematic task motions and rigid body constraints by proposing a new approach defining a class of Task-based Constraint (TbC) that encompasses both concepts. From this point of view, tasks and constraints (as traditionally defined) are one and the same problem. More recently, Nenchev et al. [113] employs the notion of motion task constraints in a similar fashion.

Goal of the Thesis

Effectively dealing with contacts and other motion constraints poses major challenges in the field of robotics that already led to many past research endeavours, and will certainly continue demanding further research efforts. The goal of this thesis is to explore further the dynamic modelling, control and learning of manipulation motions in contact with the environment — or constrained by the environment. More specifically, this thesis explores the Task-based Constraint representation for addressing the dynamic modelling, control and learning of manipulation motions with contacts. The hypothesis is that the TbC abstraction represents a useful mechanism of decoupling the robotic motion control policies into simpler motions, bringing us better understanding, ease of implementation and, finally, generalization capabilities of such simple motions across different environments.

Many robotics' researchers or practitioners have, at some point, realized that, quite often, designing controllers for complex robotics' tasks is as much art as science. Therefore, one might wonder that if, in the same way that "Art lives from constraints and dies from freedom", rather than searching for free/unconstrained motions that avoid contact with obstacles at all cost, how can we use the notion of constraints such that robots can exploit contacts for more effective interactions with the environmnet.

1.2 Contributions

List of Contributions

The work in this this thesis resulted in the following list of contributions to the field for robotics:

(C1) Proposal of a class of holonomic equality constraints — the Task-based Constraint (TbC) — that unifies the mathematical treatment and interpretation of rigid-body constraints and kinematic tasks; **(Section 2.3)**

- **(C1.1)** Reformulation of the constraint inertia matrix used in the Projected Dynamics (PD) approach, generalizing the multiple algebraic expressions originally proposed by Aghili [1]; **(Subsection 2.3.3)**
- **(C1.2)** Derivation of the expression of the constraint inertia matrix with minimum possible condition number; **(Subsection 2.3.3 and Appendix A.1)**
- **(C1.3)** Proof of the mathematical equivalence of the forward Projected Dynamics, from Aghili [1], and the analytical solution for the forward constrained dynamics from Udwadia [163], for Task-based Constraints; **(Subsection 2.3.4 and Appendix A.2)**[1]
- **(C1.4)** Generalization of the dynamically consistent inverse principle, originally proposed by Khatib [77], to rank deficient Jacobian matrices; **(Subsection 2.3.4 and Appendix A.3)**

(C2) Derivation of the expression of the partitioned task-space dynamics for a robotic system subject to two task-based constraints; **(Section 2.4 and Appendix A.4)**

- **(C2.1)** Proof of the equivalence of the operational space equations of motion with rigid-body constraints independently developed by De Sapio and Khatib [34] and Mistry and Righetti [101]; **(Section 3.3)**

[1]Note that at the writing of this document I discovered that Nenchev et al. [113] got to the same conclusion in a book published in the very same year of our publication. However, they got to that conclusion by realizing that both the formulation from Aghili [1] and the result of the Gauss' principle of least constraint, which is used by Udwadia [163] to obtain the analytical solution, are in turn equivalent Maggi's Equations (Null-space Projection Method) [23, 83, 161]. In our publication we produced expressions directly equating both formulations stated in the contribution above.

(C3) Adaptation of the Operational Space Formulation for the simultaneous position and force control of a velocity controlled robot; **(Section 3.4)**
- **(C3.1)** Proposal of a tracking strategy for performing planar motion patterns in non-flat surfaces with smooth but unknown geometry; **(Subsection 3.5.2)**
- **(C3.2)** Experimental validation of the adapted simultaneous position and force control strategy for the task of wiping a curved surface using a standard robotic manipulator. **(Subsection 3.5.2)**
- **(C3.3)** Demonstration of the application of the simultaneous position and force control strategy for the automation of train's cab front cleaning process; **(Subsection 3.5.3)**

(C4) Validation of the Constraint-aware Policy Learning (CaPL) method, based on the Singular Value Decomposition (SVD); **(Section 4.4)**
- **(C4.1)** Simulation comparison of the Constraint-aware Policy Learning (CaPL) method with Direct Policy Learning (DPL), on low dimensional examples; **(Subsections 4.4.1 and 4.4.3)**
- **(C4.2)** Experimental validation of the CaPL generalization capabilities across different tasks and constraints on real robot hardware, for the task of wiping a curved surface using force sensing information; **(Subsection 4.4.5)**

(C5) Proposal of a metric for computing the similarity of the estimated constraints across a dataset; **(Section 4.5)**

(C6) Theoretical analysis of closed-form solutions for constraint estimation that lead to the decomposition of the direct policy error; **(Subsection 4.3.2)**
- **(C6.1)** Proof of the direct policy error decomposition into the sum of the constraint space error and the constrained policy error, under the condition of a semi-orthogonal constraint matrix; **(Appendix A.5)**
- **(C6.2)** Proof of closed-form constraint estimation solution based on the Generalized Eigenvalue Decomposition (GEVD), for the problem of the constraint matrix without task controller and under the condition of a semi-orthogonal constraint matrix, on average; **(Appendix A.6)**
- **(C6.3)** Derivation of closed-form solution, based on the Generalized Singular Value Decomposition (GSVD), for the simultaneous estimation of the constraint matrix and task controller, and under the condition of a semi-orthogonal constraint matrix, on average. **(Appendix A.7)**

Contributions by publication

Most of the contributions listed above originally appeared in the following publications:

[105] Contribution item **(C3.3)** — **(Subsection 3.5.3)** — **João Moura**, Mustafa Suphi Erden. Formulation of a control and path planning approach for a cab front

cleaning robot. In Procedia CIRP, *5th International Conference on Through-life Engineering Services* (TESConf), **2017**.

[10] Contribution item **(C4.2)** — **(Subsection 4.4.5)** — Leopoldo Armesto, **João Moura**, Vladimir Ivan, Antonio Salas, and Sethu Vijayakumar. Learning constrained generalizable policies by demonstration. In *Robotics: Science and Systems XIII* (R:SS), **2017**.

[106] Contribution item **(C3)** — **(Section 3.4)** — **João Moura**, William Mccoll, Gerard Taykaldiranian, Tetsuo Tomiyama, Mustafa Suphi Erden. Automation of Train Cab Front Cleaning with a Robot Manipulator. In *IEEE Robotics and Automation Letters* (RA-L), **2018**. [selected for presentation at the *14th IEEE International Conference on Automation Science and Engineering* (CASE)]

[12] Contribution items **(C4)** and **(C5)** — **(Section 4.4)** and **(Section 4.5)** — Leopoldo Armesto, **João Moura**, Vladimir Ivan, Mustafa Suphi Erden, Antonio Salas, and Sethu Vijayakumar. Constraint-aware Learning of Policies by Demonstration. In *International Journal of Robotics Research* (IJRR), **2018**.

[107] Contribution item **(C1)** — **(Section 2.3)** — **João Moura**, Vladimir Ivan, Mustafa Suphi Erden, and Sethu Vijayakumar. Equivalence of the Projected Forward Dynamics and the Dynamically Consistent Inverse Solution. In *Robotics: Science and Systems XV* (R:SS), **2019**. **[Best Paper Award Finalist]**

Unpublished Contributions

There are, however, some listed unpublished contributions, including:

- Despite the result from Contribution **(C2)** appearing in [107], the published work omits the derivation itself and the numerical condition for which the partitioning — and, hence, the final expression — is valid;
- The Contribution **(C4.1)** includes two examples, and only the second comparing CaPL with DPL, described in the Subsection 4.4.3, appears in [12]. The first example, described in the Subsection 4.4.1 is an additional two dimensional example that more carefully compares the CaPL method, based on SVD, with both the CCL from Howard et al. [67] and the DPL baseline;
- The Contribution **(C6.1)** is a more detailed and formal proof than the analogous proof used in [12], which also identifies the relation of the resulting decoupled error costs with error costs presented by previous literature;
- The Contribution **(C6.2)** extends the analogous proof used in [12] to multiple constraints and also identifies one key validity condition, missed in [12], which renders the solution based on GEVD only valid for constraints without task controller, therefore, invalid for task-based constraints;
- The Contribution **(C6.3)** proposes a new solution based on GSVD which is valid for task-based constraints.

1.3 Thesis Outline

The main body of the thesis consists of three chapters, each containing an introduction, relevant background, main sections, case studies and discussion. Chapter 2 — Task-based Constrained Dynamics — presents the modelling and formulation of the ***dynamics*** of constrained robotic rigid-body systems. Chapter 3 — Simultaneous Position/Force Control for Constrained Motions — develops ***control*** approaches for controlling robots constrained by the contact with the environment. Chapter 4 — Learning Generalizable Constrained Policies by Demonstration — proposes ***learning*** methods for obtaining robotic constrained motions from demonstration. Finally, the Conclusions chapter summarizes the main discussion points and contributions of the thesis and lays out possible future research avenues. Appendix A — Contributed Proofs/Results — contains the main theoretical results of this thesis, whereas Appendix B — Supplementary Proofs/Results — complements the thesis with some known results that are interesting and relevant for the presented material.

Chapter 2

Task-based Constrained Dynamics

"If you can't explain it simply, you don't understand it well enough"

Albert Einstein

This chapter introduces the Task-based Constraint (TbC) abstraction as a type of equality constraint with decoupled time and configuration dependence. It presents the forward dynamics of a TbC multi-body system, using the Principle of Least Constraint, and it proves its equivalence with the Reformulated Projected Dynamics (PD). It also generalizes the concept of dynamically consistent inverse, proposed in the Operational Space Formulation (OSF), to rank deficient Jacobian matrices. Finally, it presents the expression for the task-space dynamics of a partitioned task-based constraint Jacobian.

2.1 Introduction

Motion planning, control, learning, and state estimation, often rely on modelling a robot as a dynamical system. We refer to the robot's motion as unconstrained when its state evolves solely according to its dynamic equations of motion. Any interaction with the environment imposes constraints on the dynamical system, in the form of contacts, rigid connections, tasks and behaviours. For example, consider a floating base robot, such as a humanoid (Figure 2.1), carrying a jar of water. This robot needs to exploit the contact constraints for locomotion while maintaining balance without spilling the water. The latter are tasks that also constrain its dynamic motions. Another example is a robot with structural constraints, such as closed kinematic loops (Figure 2.1). The same robot, might require a compliant behaviour towards safe human robot interaction, which also constrains the dynamical motion of the robot. Our motivation is to model dynamical systems with a generic class of constraints that are useful in developing motion planning and control algorithms.

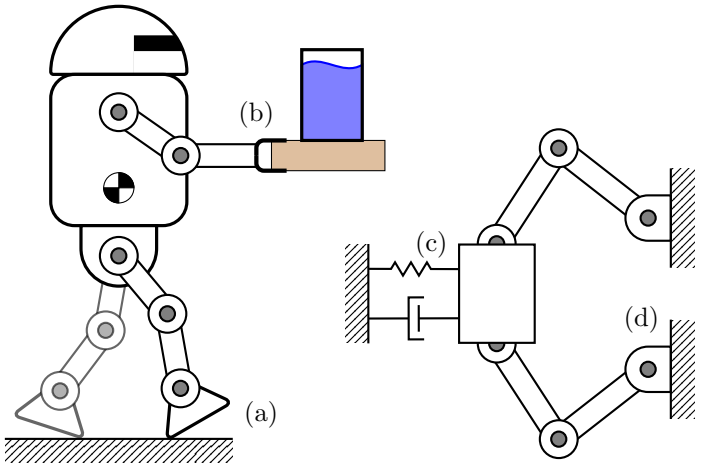


Figure 2.1: Illustration of different constraints imposed by the robot's surroundings or required behaviour. Examples are: (a) using contacts for bipedal locomotion; (b) keeping the balance while executing a critical task such as holding a jar of water; (c) having a compliant behaviour while following a given trajectory; (d) and robots with closed kinematic loops.

However, even for identical multi-body systems, there are distinct motion representations in the literature. These formulations differ in terms of both their algebraic form and some key properties. The main goal of this chapter is to present a derivation for the forward dynamics model of a constrained system, based on known analytical principles of dynamics [164], and relate this result with two of the most widely used approaches of formulation of dynamics of motion found in the robotics literature, namely the Operational Space Formulation (OSF) [77] and the Projected Dynamics (PD) [1]. Based on these two motion representations, various authors propose numerous control structures designed to achieve particular desired behaviours, by optimizing different criteria.

Khatib [77] proposes the Operational Space Formulation (OSF) as a methodology for the description of the end-effector/tool constrained motion task. This work relies on the definition of a dynamically consistent inverse Jacobian, as a way of controlling redundant robots without affecting the specified end-effector task motion. Numerous studies follow and extend this formulation. For instance: Sentis and Khatib [150, 151, 152] propose a task prioritization framework for control of humanoid robots, achieving complex behaviours by the activation or deactivation of different tasks and constraints, and their ordering in the pool of control primitives; Park and Khatib [125] address multiple contacts and the transition between those contacts in the control of humanoid robots; Nakanishi et al. [110] compare it with velocity and acceleration based controllers, concluding that this formulation is quite sensitive to modeling errors when compared to the other kinematic-based approaches; and De Sapio and Khatib [34] incorporate time independent equality constraints (scleronomic constraints) into the operational space formulation, highlighting the symmetry

between constraints and tasks.

Aghili [1, 2] proposes a Projected Dynamics (PD) approach for the derivation of the rigid multi-body dynamic equations of motion, subject to scleronomic equality constraints. This work relies on the definition of a constraint inertia matrix, in order to represent the constrained forward dynamics in the configuration space. Numerous studies follow and extend this approach. For instance: Mistry and Righetti [101] derive operational space controllers for constrained systems with passive joints; Ortenzi et al. [118] integrate the Projected Inverse Dynamics in an optimal control framework for robots in contact; Lin et al. [91] propose a control framework for multi-arm Cartesian impedance control; Dehio et al. [38] model and control multi-arm and multi-legged robots, while compensating for object dynamics, enabling human-robot interaction; Pardo et al. [122] present a planning and control approach in the constraint-consistent subspace for dynamic legged robot locomotion.

When contrasting these two main approaches, we need to analyse both their domain of application and key properties. Regarding the domain of application, the OSF is a framework for describing and controlling task space motions, such as the motion of a humanoid's centre of mass or an industrial robot's end-effector, whereas the PD is an approach for modelling and controlling constrained multi-body systems, for instance two serial arms physically linked together resulting in a closed kinematic loop. De Sapio and Khatib [34] extend the OSF to include constrained systems, and Mistry and Righetti [101] extend the PD to include task motions, in which case the domain of application is the same. Regarding their key properties: the OSF requires a full rank robot Jacobian while the PD handles a rank deficient constraint Jacobian; and the OSF uses an oblique projection (based on the dynamically consistent inverse) whereas the PD relies on an orthogonal projection.

Despite these differences, we show that the expressions for the forward dynamics derived from both these approaches are analytically equivalent. Our work takes inspiration from the more general treatment of equality constraints from Udwadia and Kalaba [164], and the parallels between task space and constraint formulations drawn by De Sapio et al. [35]. We start by defining a class of constraints called Task-based Constraint (TbC), with the aim of unifying both OSF and PD in terms of their domain of application. This class of constraints is in itself a sub-class of a general type of equality time dependent constraint (rheonomic constraints). Figure 2.2 illustrates the relationship between the different types of equality constraints mentioned, and their relation to OSF and PD.

An immediate repercussion of proving the equivalence between OSF and PD is that we can use either formulation for computing the forward dynamics of a constrained system, with both leading to the same dynamically consistent result. From this point of view, regarding the simulation of a multi-body system, the only reason for choosing one method over the other is seeking some numerical advantage, as reduced numerical errors or reduced speed of computation. There are studies that

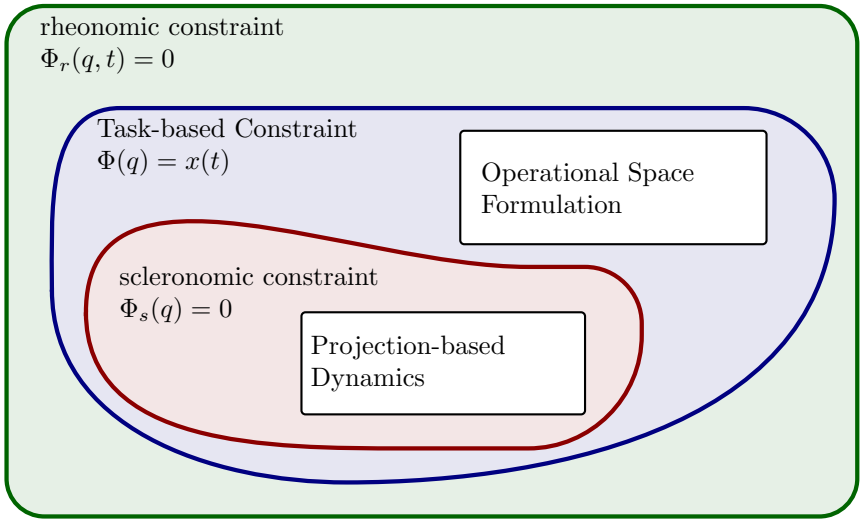


Figure 2.2: Diagram illustrating the categorization of the two forward dynamic approaches discussed in this chapter, regarding their underlying equality holonomic constraint. A rheonomic constraint is a time dependent constraint, a scleronomic constraint is a time independent constraint, and a Task-based Constraint is a time dependent constraint with decoupled dependence on the configuration q and time t. Note that despite the PD being mostly applied to scleronomic constraints, in an earlier work Aghili and Piedbœuf [6] demonstrate its application to rheonomic constraint. The categorization in this diagram corresponds to the context of explicitly considering a task component in the constraint formulation.

demonstrate efficient computation of the dynamically consistent inverse Jacobian, as [169] for a full end-effector Jacobian, and [53] for branched kinematic trees, that might grant the OSF some computational speed gains for these particular cases. On the other hand, given the flexibility of PD approach in choosing different algebraic expressions for the constraint inertia matrix, theoretically we could find a matrix that might grant us some ease in its inversion, both from a computational speed and numerical errors perspective. However, so far we have found no evidence of the numerical superiority of either method for a general case. Additionally, there are some studies focusing on the stability analysis [53] and asymptotic stability for the regulation case of passivity-based controllers [43] done for the OSF, that might readily apply to the PD by using the results presented in this chapter.

Over the years, researchers and engineers have been relying on their ability to find useful abstractions for modelling complex systems, so that they can understand and modify them (prediction, design and control). Likewise, as highlighted above, the synthesis of controllers for complex dynamic robotic systems relies upon appropriate models of those systems. However, when researching the vast literature [7, 101, 111, 129, 162, 163] on dynamic multi-body systems, especially those involving contacts and task goals, the complex expressions resulting from various different formulations and approaches can easly become overwhelming. Yet, the Guass' principle of least constraint, that in principle should support any of the different formulations for constrained dynamics, is remarkably simple and compact [29, 164], almost trivial to understand. Therefore, this chapter aims at finding some clarity on some of these modeling approaches for constrained dynamic systems, because modeling is about explaining and understanding.

2.2 Background

2.2.1 Holonomic Equality Constraints

Let's consider the following generic holonomic equality constraint [52, 164], expressed as

$$\phi(q, t) = 0, \tag{2.1}$$

where $q \in \mathcal{D}_q \subset \mathbb{R}^{n_q}$ is the vector of generalized coordinates or configuration of a rigid multi-body system, with n_q being the number of generalized coordinates or dimensionality of the configuration space, and with $t \in \mathbb{R}$ representing time and $\mathcal{D}_q$ the constraint manifold. In fact, in a more rigorous classification, (2.1) represents a rheonomic constraint, i.e a constraint which depends on both the configuration q and time t. Another subtype of holonomic constraints would be the scleronomic constraint $\phi(q) = 0$, which only depends on the configuration. Figure 2.2 illustrates a possible categorization of these constraints.

Differentiating (2.1) leads to

$$A(q,t)\dot{q} = b(q,t), \tag{2.2}$$

where $A(q,t) = (\partial\phi/\partial q) \in \mathbb{R}^{n_b \times n_q}$ corresponds to the constraint Jacobian, with n_b being the dimension of the constraint space, i.e. number of constraints, $\dot{q} \in \mathcal{Q} \subset \mathbb{R}^{n_q}$ is the generalized velocity and $b = -(\partial\phi/\partial t) \in \mathbb{R}^{n_b}$. For the case of scleronomic constraints then (2.2) becomes

$$A(q,t)\dot{q} = 0, \tag{2.3}$$

which is the so called constraint in the Pfaffian form [84, 108], i.e. constraints are linear with respect to the velocity.

Note that the expression (2.2) can also represent a type of non-holonomic constraints, in case of non-integrable constraint matrix A, i.e. we are unable to obtain ϕ from A. One characteristic of such non-holonomic systems, represented by (2.2), is the existence of more position Degrees of Freedom (DoF) than velocity DoF [52], i.e. $\mathcal{Q}$ has larger dimensionality than $\mathcal{D}_q$. Another typical non-holonomic constraint is $\phi(q,\dot{q},t) = 0$, or generally speaking any inequality constraint $\phi(\ldots) \leq 0$.

Let's consider the case where we want to know what are the admissible constrained velocities, i.e. we want to solve Eq. (2.2) for $\dot{q}$. However, for a redundant robotic system ($n_b < n_q$), there is an infinite number of solutions. The general solution is

$$\dot{q} = Gb + (I_{n_q} - GA)\dot{q}_\varepsilon, \tag{2.4}$$

where G is any generalized inverse (or G-inverse) of A, $\dot{q}_\varepsilon \in \mathbb{R}^{n_q}$ is an arbitrary configuration velocity, and I_{n_q} is $n_q \times n_q$ identity matrix. A G-inverse of A is a matrix that satisfies the condition $AGA = A$ [15, 164].

The most widely applied type of G-inverse in the robotics literature is the Moore-Penrose inverse (MP-inverse), often called pseudo-inverse [49, 154]. The MP-inverse of A is the unique matrix $G = A^\dagger$ that satisfies the 4 conditions: (i) $AGA = A$ (ii) $GAG = G$ (iii) $AG = (AG)^\top$ (iv) $GA = (GA)^\top$, and it emerges from the solution $\dot{q} = A^\dagger b$ to (2.2) of minimum-norm $\|\dot{q}\|^2 = \langle q, q\rangle$ that minimizes the least-square error $\|A\dot{q} - b\|^2$ [45, 164], where $\langle\,,\rangle$ represents an Euclidean inner product. For full row rank A, the MP-inverse assumes the widely used closed form solution $A^\dagger = A^\top \left(AA^\top\right)^{-1}$, known as right inverse.

Another widely used unique G-inverse is the inertia-weighted generalized inverse $G = \overline{A}$ [35], that arises from the solution $\dot{q} = \overline{A}b$ to (2.2) that minimizes the instantaneous kinetic energy $\frac{1}{2}\|\dot{q}\|_M^2 = \frac{1}{2}\langle\dot{q},\dot{q}\rangle_M = \frac{1}{2}\langle\dot{q}, M\dot{q}\rangle$ of a multi-body system, with inertia matrix M, while minimizing the least-square error $\|A\dot{q} - b\|^2$. This G-inverse satisfies the first 3 MP-inverse conditions, with the additional condition $MGA = (MGA)^\top$ [15]. For full row rank A we get the closed form solution [77]

$$\overline{A} = M^{-1}A^\top(AM^{-1}A^\top)^{-1}. \tag{2.5}$$

The second term of the sum in (2.4) is a projection matrix $(I_{n_q} - GA)$, that projects any arbitrary configuration velocity $\dot{q}_\varepsilon$ to the null space of A, i.e. $(I_{n_q} - GA)\dot{q}_\varepsilon \in \mathcal{N}(A)$. We can define this projection operator for both $\overline{A}$, where $P_M \triangleq (I_{n_q} - \overline{A}A)$ is an oblique inertia-weighted projection matrix [35], and for $A^\dagger$, where $P \triangleq (I_{n_q} - A^\dagger A)$ is an orthogonal projection, i.e. $P = P^\top$. Where we use P to mean P_I to simplify notation.

Taking the time derivative of (2.2), corresponding to the second time derivative of (2.1), results in

$$A(q,t)\ddot{q} = \underbrace{\dot{b}(q,t) - \dot{A}(q,t)\dot{q}}_{\triangleq c(q,\dot{q},t)}, \tag{2.6}$$

where $\ddot{q} \in \mathcal{A} \subset \mathbb{R}^n$ is the generalized acceleration, i.e (2.6) corresponds to the constraint (2.1) written in the acceleration space. We can than write a more general non-holonomic acceleration equality constraint as

$$A(q,\dot{q},t)\ddot{q} = c(q,\dot{q},t), \tag{2.7}$$

where in this case A is non-integrable.

Similarly to the general solution for the constrained velocity in (2.4), the general solution for the constrained configuration acceleration from (2.7) is

$$\ddot{q} = Gc + (I_{n_q} - GA)\ddot{q}_\varepsilon, \tag{2.8}$$

with $\ddot{q}_\varepsilon \in \mathbb{R}^n$ being an arbitrary configuration acceleration. Therefore, any forward dynamics solution, i.e. expression for computing a $\ddot{q}$, will always be a particular case of the general solution (2.8), corresponding to particular choices of G and $\ddot{q}_\varepsilon$, and resulting from optimizing different cost functions. For instance: $\ddot{q} = A^\dagger c$ minimizes $\|\ddot{q}\|^2$, being a particular case of (2.8) for which $G = A^\dagger$ and $\ddot{q}_\varepsilon = 0$; $\ddot{q} = \overline{A}c$ minimizes $\frac{1}{2}\|\dot{q}\|_M^2$ and corresponds to the case where $G = \overline{A}$ and $\ddot{q}_\varepsilon = 0$.

2.2.2 Gauss's Principle of Least Constraint

In the search for describing the motion of a constrained system, let's start by describing the equations of motion of an unconstrained system in the configuration space, as

$$M(q_\star)\ddot{q}_\star + h(q_\star,\dot{q}_\star) = \tau \tag{2.9}$$

where $h \in \mathbb{R}^n$ contains the Coriolis, centrifugal, and gravitational terms, $M \in \mathbb{S}^{n_q}_{++}$ is the unconstrained inertia matrix, $\tau \in \mathbb{R}^{n_q}$ is the generalized force vector in the configuration space, and $q_\star, \dot{q}_\star, \ddot{q}_\star \in \mathbb{R}^{n_q}$ are, respectively, the unconstrained generalized position, velocity, and acceleration. For an unconstrained system where M is a symmetric positive definite matrix, we can compute the forward dynamics by simply inverting M as

$$\ddot{q}_\star = M^{-1}(\tau - h). \tag{2.10}$$

There are of course more efficient methods for computing $\ddot{q}_\star$ [52], we shall use this expression just for derivation purposes.

The previous section ended with the presentation of a general solution for the constrained acceleration in (2.8). One can obtain a particular solution of (2.8), by using a fundamental principle of mechanics - the Gauss's Principle of Least Constraint - which mainly states that if a given configuration acceleration $\ddot{q}$ simultaneously satisfies the constraint and minimizes the Gauss function

$$\mathcal{G}(\ddot{q}) = \langle \ddot{q}_\star - \ddot{q}, \ddot{q}_\star - \ddot{q} \rangle_M, \tag{2.11}$$

where $\ddot{q}_\star$ is the unconstrained acceleration, then $\ddot{q}$ is the correct acceleration the constrained system will acquire, i.e. that is the acceleration that actually materializes [29, 35, 164]. The result of that minimization is

$$\begin{aligned} \ddot{q} &= \overline{A}b + P_M \ddot{q}_\star \\ &= \overline{A}b + P_M M^{-1}(\tau - h) \end{aligned} \tag{2.12}$$

where we can see that it is a particular solution of Eq. (2.8) for which $G = \overline{A}$ and $\ddot{q}_\varepsilon = \ddot{q}_\star$.

Udwadia and Kalaba [164] derive the Fundamental Equation

$$\ddot{q} = \ddot{q}_\star + \overline{A}\left(b - A\ddot{q}_\star\right), \tag{2.13}$$

which is an equivalent way of writing (2.12). Furthermore, Udwadia and Kalaba [164] consider the case of a rank deficient A, by using the following inertia-weighted generalized inverse,[1]

$$\overline{A} = M^{-1}A^\top \left(AM^{-1}A^\top\right)^\dagger. \tag{2.14}$$

In the following subsections and sections we will relate this remarkably simple result with the constraint forward dynamic solutions from the OSF and the PD.

2.2.3 Operational/Task Space Dynamics

One way of taking into account the contribution of a task motion in the dynamics equation of motion of a multi-body system is to simply add an extra force component to the equation of motion (2.9) as,

$$M\ddot{q} + h - A^\top \lambda = \tau, \tag{2.15}$$

[1] Udwadia and Kalaba [164] discuss other expressions for the inertia-weighted generalize inverse that still comply with the Gauss's Principle of Least Constraint.

where $\lambda \in \mathbb{R}^m$ is the force coming from the task-space. We obtain the task/-operational space dynamics equation of motion by applying the operational space formulation from [77], resulting in

$$M_x \ddot{x} + h_x - \lambda = f, \tag{2.16}$$

where

$$M_x \triangleq \left(AM^{-1}A^\top\right)^{-1} = \overline{A}^\top M \overline{A} \tag{2.17}$$

is the task space inertia matrix [54], with $h_x \triangleq \overline{A}^\top h - M_x \dot{A}\dot{q}$, for A full row rank, $f \triangleq \overline{A}^\top \tau$ being task-space commanded force and $\ddot{x}$ the task-space acceleration.

Similar to the case in (2.4) where we decompose $\dot{q}$ into a task component and null-space component using a G-inverse, Khatib [77] decomposes τ in its task and null-space components using $A^\top$ as

$$\tau = A^\top f + (I_{n_q} - A^\top G^\top)\tau_\varepsilon, \tag{2.18}$$

where $\tau_\varepsilon \in \mathbb{R}^n$ is an arbitrary generalized force vector.

Khatib [78] defines the Dynamically Consistent Inverse of a robot Jacobian A as the matrix G that satisfies the condition

$$AM^{-1}\left(I_{n_q} - A^\top G^\top\right)\tau_\varepsilon = 0, \tag{2.19}$$

which corresponds to the solution that decouples the generalized force τ into a component $A^\top f$ only acting in the robot's end-effector, and a component $(I_{n_q} - A^\top G^\top)\tau_\varepsilon$ that only affects the internal motion of the robot, i.e. the task space motion is unaffected by τ_ε. Furthermore, Khatib [78] proves that for full rank A, the only G-inverse that satisfies such condition is the inertia-weighted generalized inverse [54].

Peters et al. [129] and Bruyninckx and Khatib [29] already discuss the equivalence of the operational space control expressions with the results obtained using the Gauss's Principle of Least Constraint. We can easily show that by rearranging (2.16) with respect to the constraint force λ and substituting it in (2.15) and then inverting the inertia matrix M, we obtain the same solution of (2.12), for $b = \ddot{x} - \dot{A}\dot{q}$. However, from a domain point of view, these are typically seen as two different formulations. In the Fundamental Equation of the Analytical Dynamics [164], A refers to the Jacobian of a constraint, whereas in the Operational Space formulation [77], A refers to the robot Jacobian, mapping the joint space to some task space of interest — typically the end-effector of a robot or it's centre of mass. However, already highlighted by De Sapio and Khatib [34], those two domains share the same mathematical machinery. The following section formalizes a type of equality constraint that aims at unifying the treatment of these two domains.

2.3 Task-based Constraints

As identified in the previous section, the treatment of task motions and constraints can be identical, differing only from a domain point of view. While some works explicitly differentiate motions and constraints as two seperate domains [34, 35], other works treat these as one, for instance, using the term "motion task constraints" [67, 113]. Although, constraints such as a rigid connection have the unique property of maintaining their own dynamic consistency [101], by sustaining — in theory — any applied force, we can have task motion controllers that can also sustain applied forces, acting effectively as constraints to the overall multi-body dynamic system motion.

In order to unify these two domains, we define a Task-based Constraint as any constraint written in the form

$$\phi(q) = x(t), \tag{2.20}$$

where $x \in \mathcal{D}_x \subset \mathbb{R}^{n_x}$ is the vector of the task space coordinates, with $n_x \leq n_q$, and t represents time. The function $\phi : \mathcal{D}_q \mapsto \mathcal{D}_x$ - that maps the configuration space to task space - is, in general, a non linear function that captures the geometric model of the given task-based constraint. From a constraint categorization, (2.20) represents a special type of rheonomic constraint — $\phi_r(q, t)$ — where we can decouple dependence on the two variables, the configuration q and time t. Figure 2.2 captures this constraint categorization. This definition is essentially an artifact to dress the traditional forward kinematics expression, which is identical to (2.20), in the form of a constraint definition. Note we can use $x(t)$ to model any time dependent process which is independent of the configuration.

From a robotics perspective, the task x might be: nonexistent, i.e. $x = 0$, corresponding to some rigid link connection; externally enforced but still time varying, which might be through connection to a moving rail or caused by another agent (human or robotic); or enforced by the robot itself by an appropriate task controller. Regardless of task enforcing mechanism, the effect is always a reduction of the configuration space domain $q \in \mathcal{D}_q = \{q \in \mathbb{R}^n, x \in \mathcal{D}_x \mid \phi(q) = x\}$.

2.3.1 Task-based Constrained Dynamics

By analogy with the more general rheonomic constraint in (2.1), we can easily observe that the task-based constraint represented in the velocity space corresponds to (2.2) where $c(t) = \dot{x}(t)$, i.e.

$$A(q)\dot{q} = \dot{x}(t), \tag{2.21}$$

with the respective general solution being

$$\dot{q} = G\dot{x}(t) + (I_{n_q} - GA)\dot{q}_\varepsilon. \tag{2.22}$$

Likewise, the task-based constraint represented in the acceleration space corresponds to (2.6) where $c(q, \dot{q}, t) = \ddot{x}(t) - \dot{A}(q)\dot{q}$, i.e.

$$A(q)\ddot{q} = \ddot{x}(t) - \dot{A}(q)\dot{q}, \tag{2.23}$$

and so we can write the forward dynamics of a task-based constrained multi-body system, corresponding to (2.12) and (2.13), as

$$\begin{aligned} \ddot{q} &= \overline{A}(\ddot{x} - \dot{A}\dot{q}) + P_M M^{-1}(\tau - h), \\ &= \ddot{q}_\star + \overline{A}\left(\ddot{x} - \dot{A}\dot{q} - A\ddot{q}_\star\right). \end{aligned} \tag{2.24}$$

2.3.2 Projected Dynamics

The previous subsection derived the forward dynamics for a task-based constrained system. We can find in the literature various other formulations for the forward and inverse dynamics of constrained systems. Recent works in robotics [38, 91] started using one of such particular formulations which relies on the concept of orthogonal projections. Aghili [1, 2] defines the Projected Inverse Dynamics of a multi-body system by pre-multiplying Equation (2.15) with the orthogonal projector P, obtaining

$$PM\ddot{q} = P(\tau - h). \tag{2.25}$$

For computing the forward dynamics equation of motion, as PM is singular, we pre-multiply (2.23) with $A^\dagger$, obtaining

$$(I_{n_q} - P)\ddot{q} = A^\dagger(\ddot{x} - \dot{A}\dot{q}), \tag{2.26}$$

and combine it with (2.25), obtaining

$$M_c\ddot{q} = P(\tau - h) + C_c(\ddot{x} - \dot{A}\dot{q}) \tag{2.27}$$

where M_c is the constraint inertia matrix. As M_c is invertible, we get

$$\ddot{q} = M_c^{-1}P(\tau - h) + M_c^{-1}C_c(\ddot{x} - \dot{A}\dot{q}), \tag{2.28}$$

which Aghili [2] calls equation of motion of a constrained system in a compact form.[2]

Depending on different ways of combining Equation (2.25) and Equation (2.26), both M_c and C_c take different forms. Aghili [1, 2] derives the following different combinations of M_c and C_c:

[2]Note that here we followed the same derivation process as in [1] but adding as task acceleration $\ddot{x}$. By making $\ddot{x} = 0$, we recover the original results. Therefore, to accommodate $\ddot{x}$ we removed $\dot{A}$ from C_c.

(1) $M_c^{(1)} = PM + (I_{n_q} - P)$, $C_c^{(1)} = -A^\dagger$,
(2) $M_c^{(2)} = M + PM + (PM)^\top$, $C_c^{(2)} = -MA^\dagger$,
(3) $M_c^{(3)} = PMP + (I_{n_q} - P)M(I_{n_q} - P)$,
$C_c^{(3)} = -(I_{n_q} - 2P)MA^\dagger$,
(4) $M_c^{(4)} = PM + \gamma(I_{n_q} - P)$, $C_c^{(4)} = -\gamma A^\dagger$,

where γ is a non-negative scalar. Aghili [2] also proves some key numerical properties for each of the previous results. For instance, for M symmetric and positive definite, $M_c^{(2)}$ is positive definite but not symmetric, $M_c^{(3)}$ is both symmetric and positive definite, and $M_c^{(4)}$ is always invertible but it is neither symmetric nor positive definite. Trade-offs of using one result over the others are for instance $M_c^{(4)}$ requiring less computation effort than the other alternatives, while $M_c^{(3)}$ entailing three additional matrix multiplication operations, compared to $M_c^{(2)}$, but physically exhibiting the numerical characteristics of an inertia matrix M.

2.3.3 Projected Dynamics Reformulation

We reformulate the constraint inertia matrix M_c as

$$M_c \triangleq PM + R(I_{n_q} - P), \tag{2.29}$$

and C_c as

$$C_c \triangleq -RA^\dagger. \tag{2.30}$$

It is straightforward to show, through simple algebraic manipulations, that all choices of M_c and C_c presented by Aghili [2] are particular instances of (2.29) and (2.30), where the matrix R respectively takes the following expressions:

(1) $R^{(1)} = I_{n_q}$,
(2) $R^{(2)} = M$,
(3) $R^{(3)} = (I_{n_q} - 2P)M$,
(4) $R^{(4)} = \gamma I_{n_q}$.

We can then re-write (2.28) as

$$\ddot{q} = M_c^{-1}P(\tau - h) + M_c^{-1}RA^\dagger(\ddot{x} - \dot{A}\dot{q}). \tag{2.31}$$

In fact, we will show later that R can be any square matrix provided that M_c is full rank. We can even use a matrix $R^{(r)}$ with randomly generated elements, as long as we check the rank of M_c, and the solution of Equation (2.31) will remain the same. This happens, because the purpose of combining (2.25) and (2.26) is to invert the projected inverse dynamics (2.25) and, therefore, we have to add some component to (2.25) in order to make PM full rank. We can even look at this solution as a special type of regularization, where the regularization term R only affects the complement space of the motion of interest.

Two important benefits of this reformulation are: firstly, any proof done for these generalized M_c and C_c is directly valid for all the results discussed in the Subsection 2.3.2; and secondly, we have now a mechanism of obtaining new M_c and C_c based on finding R such that M_c and C_c satisfy some desired property. We could try to find R that would confer M_c a particular structure of interest, such as for instance making it a block diagonal matrix, that would improve the computational efficiency of its inversion. We have unsuccessfully attempted to find such structures. Alternatively, we can find R that leads to the M_c with the smallest condition number among all possible constrained inertia matrices, which is a desirable numerical property [2].

Lemma 2.3.1. The $R^{(*)}$ that minimizes $\kappa(M_c)$, where $\kappa(.)$ represents the condition number, is given by

$$R^{(*)} = \mu I_{n_q} - PM, \tag{2.32}$$

yielding $M_c^{(*)} = PMP + (\mu I_{n_q} - P)$, for some $\mu \in \mathbb{R}$ such that $\{\varsigma_{min}(PMP) \neq 0\} \leq \mu \leq \varsigma_{max}(PMP)$, where $\varsigma(.)$ represents singular values. Furthermore, $\kappa(M_c^{(*)})$ is equivalent to $\kappa(Z_2^\top M Z_2)$, where Z_2 is a basis for $\mathcal{N}(A)$ such that $P = Z_2 Z_2^\top$.

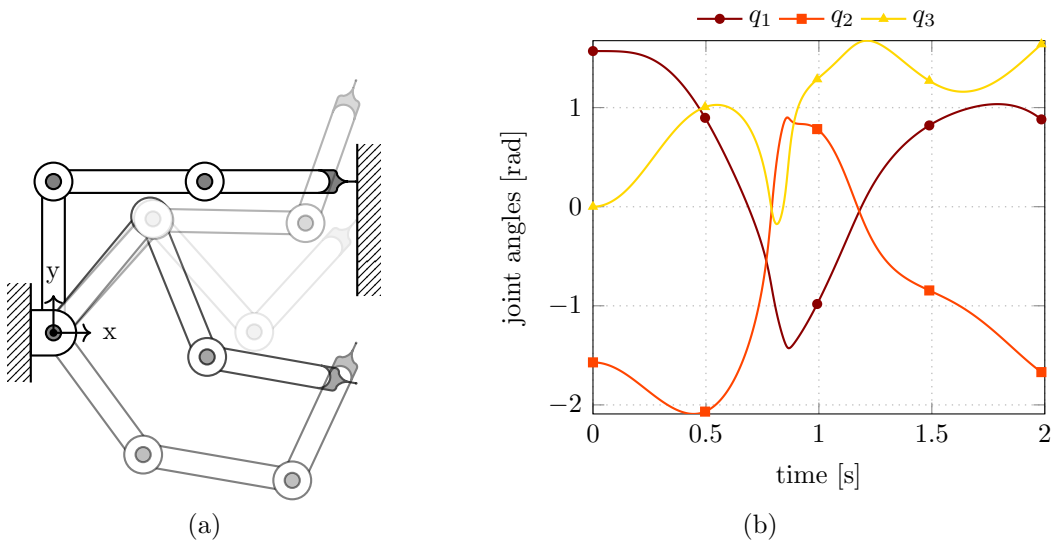
Appendix A.1 contains the proof of Lemma 2.3.1.

Remark. Note that $M_c^{(*)}$ is inconsistent from a units point of view, i.e. we are summing quantities of different physical units. The same problem arises for the case of $M_c^{(1)}$, as noted by Aghili [1]. Doty et al. [45] warns about the development of robotic methods with nonuniform physical units, that potentially will result in physically inconsistent results. The reason that in this case the acceleration results are unaffected by physical units inconsistency is because the R matrix is essentially just an artifact for regularization purposes, with no physical meaning and, as we shall see later, it vanished in the computation process of the generalized acceleration.

Case Study for the Inertia Matrix's Condition Number

Aghili [2] provides a detailed analysis on the symmetry and positive definiteness properties of each M_c proposed, key properties for its inversion. Another key property to consider for the inversion of M_c is its condition number. For illustration purposes, we simulated a free fall motion of a frictionless three link planar serial robot arm with its end-effector constrained to a vertical slider, as shown in Figure 2.3, meaning that the Task-based Constraint is $\phi(q) = \mathrm{x}(q) - \mathrm{x}_c = 0$ and the constraint Jacobian matrix is $A = \frac{\partial \mathrm{x}}{\partial q}$, where x is the robot's end-effector coordinate in the axis perpendicular to the vertical slider and x_c is the position of the vertical slider in the same axis. We then computed the condition number for M and for the different M_c's discussed in this chapter, as shown in Figure 2.4.

The robot arm used consists of a planar serial robot arm (Figure 2.3a), composed of three identical links with equidistant centre of mass from the joints, and with length, mass, and inertia of 1 m, 1 kg, and 0.1 kg m^2, respectively. We used Corke's [33] MATLAB® toolbox for computing M, h, $\ddot{q}_\star$, and the robot's Jacobian. For the for-



(a) (b)

Figure 2.3: Free fall (i.e. $\tau = 0$) simulation of a frictionless planar serial robot arm with three links and with the end-effector constrained to a vertical slider: (a) five samples of configurations taken during the free fall motion, with less opaque configurations corresponding to further in the time of the simulation; (b) time evolution of the arm joint angles, with $q_{[1]}$ being the angle of the base joint and $q_{[3]}$ the angle of the last joint, and with markers placed at time instances corresponding to the samples in (a).

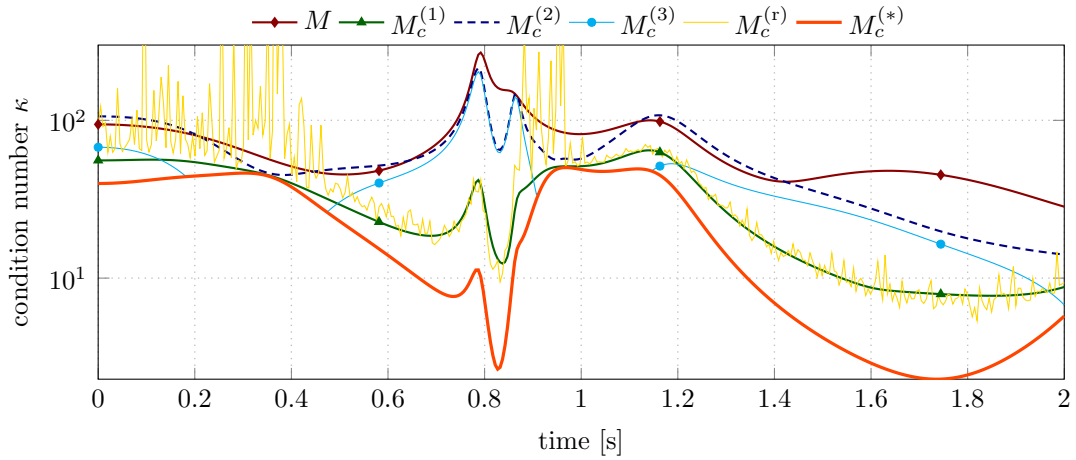


Figure 2.4: Time evolution of the condition number for different constrained inertia matrices and for the unconstrained inertia matrix. We compute $M_c^{(*)}$ using $R^{(*)}$ from (2.32), and $M_c^{(r)}$ using a different $R^{(r)}$ for every time iteration and with elements sampled from an uniform distribution in the interval $[0, 1]$.

ward simulation of the motion we integrated (2.31) using a non-adaptive (fixed step) Dormand-Prince solver of order 5, obtaining 200 samples corresponding to 2s. We set $\tau = 0$ and $\ddot{\mathrm{x}} = 0$, meaning the robot is non-actuated (free fall) and there is no task space (scleronomic constraint). Figure 2.3b shows the resulting joint angles. The initial robot configuration is $\begin{bmatrix}90^\circ & -90^\circ & 0^\circ\end{bmatrix}$ with the $\begin{bmatrix}0^\circ & 0^\circ & 0^\circ\end{bmatrix}$ configuration corresponding to the robot being horizontally stretched along the x axis. Given that the model of this planar robot excludes any friction component, the robot only experiences conservative forces and, therefore, the simulated motion of the robot consists in it bouncing up and down indefinitely.

We repeated the simulation for the various R's discussed in the Subsection 2.3.3. The resulting configurations, shown in Fig. 2.3b, are identical for any of the R options, including the randomly generated one. In Figure 2.4 we can verify, as expected, that the condition number curve corresponding to $M_c^{(*)}$ is a lower bound for all other condition number curves. However, for this illustrative case study M is too well conditioned, i.e. $1/\kappa(M) \gg$ round-off error, to expect any significant impact in the numerical errors resulting from using different M_c's. For example, if we compute the joint accelerations for the trajectory in Figure 2.3b using any two of the M_c's discussed, the mean norm of their difference is in the order of magnitude of the MatLab's round-off error (10^{-15}).

One could hypothesize that, for a badly conditioned M, using the method that requires a matrix inversion with lower condition number would result in smaller simulation errors, specially when using $M_c^{(*)}$. However, in all our experiments (other simulations involved serial arms with non-identical and larger number of links) this hypothesis failed to hold, as we always obtained the same constraint error propagation regardless of the M_c employed or even when using (2.12), where we define constraint error as simply the measure of how much the constraint is violated, i.e. the difference between the horizontal positions of the end-effector and the slider $\varepsilon_\phi = \phi(q) = \mathrm{x}(q) - \mathrm{x}_c$.

Featherstone [51] discusses that the ill-conditioning of M is more than a numerical artifact, but a phenomenon of the underlying mechanism of the multi-body system itself. We might reason that for systems and configurations where $\kappa(M)$ is large, i.e. $1/\kappa(M) \approx$ round-off error, the minimum condition number $\kappa(Z_2^\top M Z_2)$ will also approach a large value due to its dependence on M. Even though some of our experiments confirm our reasoning, until proved or extensively tested, it will remain an open question. We must point out that computing $M_c^{(*)}$ comes with a computational penalty, due to the need of computing the bounds of μ for every iteration, given that they depend on q.

Besides the implementation of the dynamics itself, when for instance using different M_c's in the constrained dynamics Equation (2.31), there are other factors that also affect the propagation of the constraint error, such as the choice of integration method. Note that by describing the constraint in the acceleration space, i.e. $\ddot{\mathrm{x}} = 0$, the integration of the equations of motion will inevitably result in a deviation from

the constraint surface. We found slight variations of the constraint error and simulation time when using different variable step integration solvers, that somehow prefer some M_c structures over the others, but without identifying a distinct pattern. For that reason, we choose a fixed step integration solver, so we could compare the effects of using different M_c's in the constrained equations of motion and removing the unknown influence of the more sophisticated variable step integration solvers. The conclusion is that, for this simple case study and using a fixed step integration method, the choice of M_c seems to have negligible effect in the propagation of the constraint error when simulating the constrained motion of the robot.

2.3.4 The Dynamically Consistent Inverse Solution

In the case study experiments of the previous subsection we verified that using either the Principle of Least Constraint or any of the PD solutions always resulted in the same constrained accelerations. Equipped with our reformulated PD (2.28) parameterized by R, we can now easily compare the Least Constraint solution for task-based constraints, given by (2.24), with the PD. We hypothesize that they are analytically equivalent, which by inspection of expressions (2.24) and (2.28) leads to the following Lemma:

Lemma 2.3.2. For any $R \in \mathbb{R}^{n \times n}$ such that M_c is invertible, we have that

$$\overline{A} = M_c^{-1} R A^{\dagger}, \tag{2.33}$$

$$P_M M^{-1} = M_c^{-1} P \tag{2.34}$$

Appendix A.2 contains the proof of Lemma 2.3.2.

Therefore, we prove that the Forward Projected Dynamics is just another way of writing the accelerations solution for a task-based constrained system, where the original projected dynamics applies only to rigid constraints, i.e. $\dot{x}(t) = 0$. On the other hand, we can also think of the Task-based Constraint formulations as a generalization of the original OSF, where we just replace the end-effector robot Jacobian by any constraint Jacobian A, that splits the space of motion into a task space of interest and its null space. Aghili [1] points out that the PD handles rank deficient A and other authors follow on referring to this property as a key differentiator from OSF [77], which requires A to be full rank. However, it is fairly straightforward to extend the OSF to the case where A is rank deficient, which allow us to numerically handle redundant constraints and singular configurations. For that we simply use the inertia-weighted generalized inverse of (2.14) and redefine the task space inertia matrix from (2.17) as

$$M_x \triangleq \left(A M^{-1} A^{\top}\right)^{\dagger} = \overline{A}^{\top} M \overline{A}. \tag{2.35}$$

Note that for A full rank, then (2.35) is equivalent to (2.17). Furthermore, we can prove that any inertia-weighted generalized inverse is still dynamically consistent, one important principle in the OSF to guarantee that task space controllers remain unaffected by null space controllers.

Lemma 2.3.3. Let $G = \overline{A}$ be the unique inertia-weighted generalized inverse of a rank deficient Jacobian A, then this generalized inverse satisfies the condition (2.19) being, therefore, a dynamically consistent inverse of the Jacobian A.

Appendix A.3 contains the proof of Lemma 2.3.3.

2.4 Multiple Task-based Constraints

So far we treated $A \in \mathbb{R}^{n_b \times n_q}$ as a single constraint, but in reality it can accommodate multiple constraints acting simultaneously, i.e. $n_b > 1$. This section considers the case of splitting the constraint matrix A into two

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, \tag{2.36}$$

and treating it as two separate constraints acting simultaneously.

When splitting the task space vector $\dot{x}$ into two components $\dot{x}_1$ and $\dot{x}_2$, each of these tasks correspond to a different constraint A_1 and A_2. Therefore, it might be useful to obtain the separate task-space dynamics corresponding to each of the constraint components, which requires partitioning the task-space inertia matrix. Appendix A.4 proves that we can write the partitioned task-space inertia matrix, corresponding to the constraint Jacobian partitioning (2.36) under the condition that $\text{rank}(A) = \text{rank}(A_1) + \text{rank}(A_2)$, as

$$\begin{aligned} M_x \triangleq (AM^{-1}A^\top)^\dagger &= \begin{bmatrix} A_1M^{-1}A_1^\top & A_1M^{-1}A_2^\top \\ A_2M^{-1}A_1^\top & A_2M^{-1}A_2^\top \end{bmatrix}^\dagger \\ &= \begin{bmatrix} M_1 & -\overline{A}_1^\top A_2^\top M_2 \\ -\overline{A}_2^\top A_1^\top M_1 & M_2 \end{bmatrix} \\ &= \begin{bmatrix} M_1 & -M_1A_1\overline{A}_2 \\ -M_2A_2\overline{A}_1 & M_2 \end{bmatrix}, \end{aligned} \tag{2.37}$$

where we define the partial task-space inertia matrices as

$$M_1 \triangleq \left(A_1P_{M2}M^{-1}A_1^\top\right)^\dagger,$$

$$M_2 \triangleq \left(A_2P_{M1}M^{-1}A_2^\top\right)^\dagger,$$

with $\overline{A_1}$ and $\overline{A_2}$ being, respectively, the dynamically consistent inverse of A_1 and A_2, and P_{M1} and P_{M2} being the respective projection matrices.

Using (2.37), we can write the inertia-weighted inverse of the stack of the two constraints (2.36) as

$$\begin{aligned}\overline{A} \triangleq M^{-1}A^\top M_x &= \begin{bmatrix} M^{-1}P_{M2}^\top A_1^\top M_1 & M^{-1}P_{M1}^\top A_2^\top M_2 \end{bmatrix} \\ &= \begin{bmatrix} P_{M2}M^{-1}A_1^\top M_1 & P_{M1}M^{-1}A_2^\top M_2 \end{bmatrix} \\ &\triangleq \begin{bmatrix} A_1^\# & A_2^\#, \end{bmatrix}\end{aligned} \tag{2.38}$$

using the result $M^{-1}P_M^\top = P_M M^{-1}$ from (B.35) (Appendix B.5), and where we define $A_1^\# \triangleq P_{M2}M^{-1}A_1^\top M_1$ and $A_2^\# \triangleq P_{M1}M^{-1}A_2^\top M_2$ as the partial dynamically consistent inverses. By partitioning the task force

$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \tag{2.39}$$

we can then obtain the separate sub task dynamic equations, based on (2.16), potentially allowing us to apply different control schemes to each task space. For example, we have

$$\begin{aligned} f_2 &= M_2(\ddot{x}_2 - \dot{A}_2\dot{q}) - \overline{A}_2 A_1^\top M_1(\ddot{x}_1 - \dot{A}_1\dot{q}) + {A_2^\#}^\top h - \lambda_2 \\ &= M_2(\ddot{x}_2 - \dot{A}_2\dot{q}) - M_2 A_2 \overline{A}_1(\ddot{x}_1 - \dot{A}_1\dot{q}) + {A_2^\#}^\top h - \lambda_2 \end{aligned} \tag{2.40}$$

for the task 2, where we can easily identify the specific components of both tasks that affect the task force of the task 2. The task force of the task 1 has an analogous expression.

2.5 The Fallacy of the Equivalent Projections

We now need to contrast the chapter's main finding — equivalence between the forward OSF and PD approach — with the one from Righetti et al. [137, 138], who also discuss the equivalence between the OSF and the PD, but from the perspective of the inverse dynamics computation (controller). They claim that the output of a torque controller is independent of the projection operator used in that computation, assuming "controllers that achieve perfect tracking of the desired accelerations". Finally, they advocate for the use of an orthogonal projection (instead of an oblique) due to its computational simplicity, free from the inertia parameters. According to our finding, when computing the desired configuration acceleration (the forward dynamics) for a desired task motion, both the OSF and the PD approaches are equivalent, rendering none of these approaches as preferable, given that both of them use kinematic and inertia parameters, only with a different combination.

The claim that all projection operators lead to identical controllers is quite appealing because it leads to simplified controllers — something one can definitely appreciate. However, this section's aim is to challenge that claim. We must start by clarifying that despite [137] presenting the modelling for the quite general case of constrained robots with floating base, which introduces a selection matrix S that indicates the actuated generalized coordinates, i.e.

$$\tau = S^\top \tau_S, \tag{2.41}$$

where τ_S corresponds to the force commands of the actuated joints, the experiments and reasoning only contemplate the over constrained case, i.e. fully actuated systems. This means we can immediately test the claim of the equivalent projection operators in our example from Subsection 2.3.3. To test the claim for the case of under actuated systems, one of the experiments uses the same robot example but with one of the joints being passive.

Equivalent projections Lemma

The original Lemma in [137] goes as follows:

Lemma 2.5.1. "Given two different linear operators P_1 and P_2 that describe the same system with constraints

$$P_1 S^\top \tau_{S1} = P_1(M\ddot{q}_d + h) \tag{2.42}$$

$$P_2 S^\top \tau_{S2} = P_2(M\ddot{q}_d + h) \tag{2.43}$$

with their parameterized set of possible controllers $\tau_{S1}(W, \tau_0)$ and $\tau_{S2}(W, \tau_0)$. The following equality holds"

$$\tau_{S1}(W, \tau_0) = \tau_{S2}(W, \tau_0), \tag{2.44}$$

where

$$\tau_S(W, \tau_0) = (\overline{PS^\top})^W P(M\ddot{q}_d + h) + (I + (\overline{PS^\top})^W PS^\top)W^{-1}\tau_0, \tag{2.45}$$

with $\tau_0 \in \mathbb{R}^{n_q}$, W being any invertible square matrix, and with $\overline{A}^W$ corresponding to the W-weighted inverse of A. The W-weighted inverse of A is the unique matrix $G = \overline{A}^W$ that satisfies the 4 conditions [15]: (i) $AGA = A$ (ii) $GAG = G$ (iii) $AG = (AG)^\top$ (iv) $WGA = (WGA)^\top$. Note that the W-weighted inverse is the general case of both the inertia-weighted inverse, for which we have $\overline{A} = \overline{A}^M$, and the Moore-Penrose inverse, where we have $A^\dagger = \overline{A}^I$. In Lemma 2.5.1, P represents any linear operator (can be a rectangular matrix), whereas in the remaining

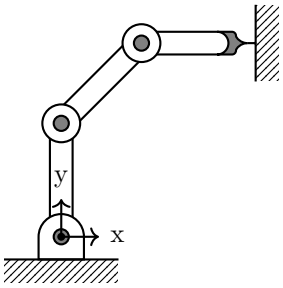


Figure 2.5: Planar serial robot arm with three links in a fixed test configuration $q_t^\top = \begin{bmatrix} 90^\circ & -45^\circ & -45^\circ \end{bmatrix}$.

of this thesis we have been using it to represent only projection matrices (which are squared matrices). Nevertheless, because projection matrices are a special case of a linear operator [173], we can use projection matrices in a counterexample to test Lemma 2.5.1.

Disproving by Counterexample

Let's consider the same planar serial robot arm used in Subsection 2.3.3, composed of three identical links with equidistant centre of mass from the joints, and with length, mass, and inertia of 1 m, 1 kg, and 0.1 kg m^2, respectively. Figure 2.5 illustrates the robot arm for the test configuration

$$q_t = \begin{bmatrix} 90^\circ \\ -45^\circ \\ -45^\circ \end{bmatrix}. \tag{2.46}$$

We used Corke's [33] MatLab® toolbox for computing the inertia matrix

$$M(q_t) = \begin{bmatrix} 6.8784 & 3.4678 & 0.7036 \\ 3.4678 & 2.4071 & 0.7036 \\ 0.7036 & 0.7036 & 0.3500 \end{bmatrix} \text{kg m}^2\,\text{rad}^{-1} \tag{2.47}$$

and the Coriolis, centrifugal, and gravitational terms

$$h(q_t, 0) = \begin{bmatrix} 15.3101 \\ 15.3101 \\ 4.9050 \end{bmatrix} \text{N m}, \tag{2.48}$$

for the test configuration q_t and assuming a static arm, i.e. $\dot{q}_t = 0$. We also computed the constraint matrix

$$A(q_t) \triangleq \begin{bmatrix} 1 & 0 \end{bmatrix} J(q_t) = \begin{bmatrix} -1.7071 & -0.7071 & -0.0000 \end{bmatrix} \text{m rad}^{-1} \tag{2.49}$$

for the test configuration q_t, where $J \in \mathbb{R}^{2\times 3}$ is the Cartesian end-effector Jacobian. Given some desired acceleration

$$\ddot{q}_d = (I_3 - A(q_t)^\dagger A(q_t))\ddot{q}_\varepsilon = \begin{bmatrix} -0.2009 \\ 0.4851 \\ 0.1270 \end{bmatrix} \text{rad}\,\text{s}^{-2}, \tag{2.50}$$

where we randomly generated

$$\ddot{q}_\varepsilon = \begin{bmatrix} 0.8147 \\ 0.9058 \\ 0.1270 \end{bmatrix} \text{m}\,\text{s}^{-2}, \tag{2.51}$$

we can then compute the torques corresponding to the test configuration q_t, the configuration velocity $\dot{q}_t = 0$ and the desired acceleration $\ddot{q}_d$, as

$$\tau_{td} = M(q_t)\ddot{q}_d + h(q_t, 0) = \begin{bmatrix} 24.1445 \\ 20.4050 \\ 6.1599 \end{bmatrix} \text{N}\,\text{m}. \tag{2.52}$$

For testing Lemma 2.5.1 we used the particular case where $W = I$ and $\tau_0 = 0$ and, therefore, (2.52) becomes

$$\tau_S = (PS^\top)^\dagger P(M\ddot{q}_d + h). \tag{2.53}$$

We then computed τ_S in (2.53) for two distinct scenarios: $S = I_3$, i.e. the manipulator is fully actuated; and

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.54}$$

i.e. the second joint is passive, denoting the torques of the first scenario as τ_{I_3} and the torques of the second simply by τ_S. Finally, as Lemma 2.5.1 implies that the values τ_S remain unaffected by the choice of the operator P, we propose to test three different projection operators. We obtain all the three projection operators based on a N-weighted projection operator

$$P_N = I - \overline{A}^N A, \tag{2.55}$$

for which we use the following weighting matrices: (1) $N_1 = I_3$; (2) $N_2 = M$; and (3) $N_3 = \text{diag}(3, 2, 1)$. Table 2.1 summarizes the results showing that, indeed, using different projection operators P_i leads to different commanded torques τ_S for this simple three link manipulator and for both cases of fully actuated and with passive joints. This demonstration disproves Lemma (2.5.1).

Table 2.1: Results of testing the Lemma of equivalent linear operators for a class of inverse dynamics' controllers. The first rows refer to the N-weighted inverse of the constraint Jacobian and the respective weighted projection matrix for three different weightings: (1) $N_1 = I_3$; (2) $N_2 = M$; and (3) $N_3 = \text{diag}(3, 2, 1)$. The last four rows show: the product of the pseudo-inverse of the projection matrix with the projection matrix; and the resulting commanded torques τ_S, for a fully actuated case, i.e. $S = I_3$, and for the case where the second joint is passive.

i	1	2	3
N_i	I_3	M	$\text{diag}(3, 2, 1)$
$(\overline{A}^{N_i})^\top$	$\begin{bmatrix}-0.5 & -0.2071 & 0.0\end{bmatrix}$	$\begin{bmatrix}-0.3636 & -0.5363 & 1.8090\end{bmatrix}$	$\begin{bmatrix}-0.4659 & -0.2895 & 0.0\end{bmatrix}$
P_{N_i}	$\begin{bmatrix}0.1464 & -0.3536 & 0.0\\ -0.3536 & 0.8536 & 0.0\\ 0.0 & 0.0 & 1.0\end{bmatrix}$	$\begin{bmatrix}0.3792 & -0.2571 & 0.0\\ -0.9156 & 0.6208 & 0.0\\ 3.0882 & 1.2792 & 1.0\end{bmatrix}$	$\begin{bmatrix}0.2047 & -0.3294 & 0.0\\ -0.4941 & 0.7953 & 0.0\\ 0.0 & 0.0 & 1.0\end{bmatrix}$
$(P_{N_i} I_3)^\dagger P_{N_i}$	$\begin{bmatrix}0.1464 & -0.3536 & 0.0\\ -0.3536 & 0.8536 & 0.0\\ 0.0 & 0.0 & 1.0\end{bmatrix}$	$\begin{bmatrix}0.9642 & -0.0528 & 0.1782\\ -0.0528 & 0.9221 & 0.2628\\ 0.1782 & 0.2628 & 0.1137\end{bmatrix}$	$\begin{bmatrix}0.2785 & -0.4483 & 0.0\\ -0.4483 & 0.7215 & 0.0\\ 0.0 & 0.0 & 1.0\end{bmatrix}$
$\tau_{I_3 i}^\top$	$\begin{bmatrix}-3.3119 & 7.9955 & 5.1494\end{bmatrix}$	$\begin{bmatrix}15.2165 & 15.1579 & 7.5524\end{bmatrix}$	$\begin{bmatrix}-2.7416 & 4.4125 & 5.1494\end{bmatrix}$
$(P_{N_i} S^\top)^\dagger P_{N_i}$	$\begin{bmatrix}1.0 & -2.4142 & 0.0\\ 0.0 & 0.0 & 1.0\end{bmatrix}$	$\begin{bmatrix}1.0 & -0.6780 & 0.0\\ 0.0 & 3.3731 & 1.0\end{bmatrix}$	$\begin{bmatrix}1.0 & -1.6095 & 0.0\\ 0.0 & 0.0 & 1.0\end{bmatrix}$
$\tau_{Si}^\top$	$\begin{bmatrix}-22.6148 & 5.1494\end{bmatrix}$	$\begin{bmatrix}4.9391 & 58.6812\end{bmatrix}$	$\begin{bmatrix}-9.8434 & 5.1494\end{bmatrix}$

Where is the fallacy?

We showed one particular example disproving Lemma 2.5.1, therefore its original proof must contain some fallacy. Let's start by reproducing the very first sentence of the proof used in [138]:

"This can be shown first by noting that the two controllers $\tau_{S1}(W, 0)$ and $\tau_{S2}(W, 0)$ must be equal since they both are controllers that minimize the cost $\tau_S^\top W \tau_S$ while there exists only one controller that minimizes this cost."[3]

The fallacy is in assuming that two optimization problems with the same objective function but different constraints result in the same solution. Indeed, $\tau_{S1}(W, 0)$ is the result of the constrained optimization problem

$$\begin{aligned} \tau_{S1} = \arg\min_{\tau} \quad & \tau^\top W \tau \\ s.t. \quad & P_1 S^\top \tau = P_1(M\ddot{q}_d + h), \end{aligned} \tag{2.56}$$

and $\tau_{S2}(W, 0)$ is the result of the constrained optimization problem

$$\begin{aligned} \tau_{S2} = \arg\min_{\tau} \quad & \tau^\top W \tau \\ s.t. \quad & P_2 S^\top \tau = P_2(M\ddot{q}_d + h). \end{aligned} \tag{2.57}$$

The optimization problems (2.56) and (2.57) are different, and in general produce different results, because their equality constraints are different.

It was important to demonstrate the fallacy of Lemma 2.5.1 because it contrasts with one of the main results of this thesis. When we proved that the expressions used in the forward dynamics of the OSF and of the PD are equivalent, that does not mean to say that the use of orthogonal or oblique projections is equivalent, but rather that different particular derivations, one using an orthogonal projection and the other using an oblique projection, lead to different expressions that compute the same result. In [137, 138], the indication is that there is an inverse dynamics expression that leads to the same results irrespectively of the linear operators employed, i.e. we can swap out those linear operators by other ones. This section challenges that conclusion.

2.6 Discussion

This chapter defines a class of Task-based Constraints (TbCs) and derives the equations of motion of a multi-body system subject to that type of constraint. By con-

[3] Here we added the subscript S to the variable τ in the original quote, in order to keep consistency with our notation.

trasting our solution with the ones from the Operational Space Formulation (OSF) and Projected Dynamics (PD), we proved in [107] that those approaches are equivalent regarding the computation of the forward dynamics of a constrained multi-body system. In order to enable such equivalence, we generalized the OSF to a rank deficient Jacobian, and reformulated the PD to generalize all previous alternative algebraic expressions of the constraint inertia matrix. A benefit of this Task-based Constraint abstraction is the convenience in expressing constraints using a task space formalization.

Section 2.4 splits a constraint matrix into two sub-constraints acting simultaneously on a multi-body dynamical system. This split of the Jacobian originally appears in [34] in order to devise a separate controller for contact force (using a contact Jacobian) and a controller for the task motion (regular robot Jacobian). However, Section 2.4 derives a more specific formula for each task force component (whereas De Sapio and Khatib [34] only indicate the computation process), which will be essential for showing the equivalence between the operational/task space controllers with constraints proposed independently by Mistry and Righetti [101] and De Sapio and Khatib [34], in Section 3.3.

As future work, it will be interesting to analyse the relation between the formulas obtained for simultaneous task-based constraints, derived in Section 2.4, and several results from the extensive literature on hierarchical dynamic task controllers, such as [26, 39, 40, 61, 131, 132, 142, 143]. The idea being to find what exactly are the physical and/or numerical conditions for which these different results equate and/or differ. For example, Siciliano and Slotine [153] and Chiacchio et al. [30] propose kinematic hierarchical schemes with task priority strategies. Mistry et al. [102] prove that stacking all the task Jacobian matrices in a single one, as we did in (2.36), is equivalent to those hierarchical strategies under the condition that the extended Jacobian be full rank, with the advantage that inverting a single extended Jacobian is easier to implement and faster to compute. However, Chiacchio et al. [30] had already noted such equivalence but for a more relaxed condition of that the sum of the ranks of the individual Jacobian matrices be equal to the rank of the extended Jacobian — the same condition we used for the result (2.37). There is also an extensive literature on kinematic hierarchical task approaches [8, 14, 48, 75, 82, 94, 98, 124], which highlights the complexity of the topic and also the dissonant views about what are preferable approaches. For instance, there is a disagreement between the more optimization based prioritization approaches [96] or the more numerically stable (singularity free) approach based on null-space projections [31]. Regarding hierarchical controllers/models on multiple tasks there are also works on stability [42] and works proposing other consistency metrics (besides the dynamical consistency discussed in this chapter), such as stiffness [41].

Finally, we obtained a fairly complex expression for the task force (2.40) from a simple constraint split (2.36). This splitting approach, in the case of the task force expression, provides us with its condition of validity and a simpler way of implement-

ing it computationally. From that standpoint, this chapter aimed at simplifying our understanding of constrained multi-body systems, by: merging the existing concepts of motion tasks and rigid constraints under a single Task-based Constraint abstraction; framing the Projected Dynamics as another result of the simple Principle of Least Constraint; and finding that deficient Jacobian inverses are still dynamically consistent.

Chapter 3

Simultaneous Position/Force Control for Constrained Motions

"Never confuse movement with action"

Ernest Hemingway

This chapter presents two widely used approaches for simultaneous control of position and force tasks using torque controlled robotic manipulators — a selection matrix approach and the operational space controllers with constraints. It relates these two approaches with the task-space controller resulting from the multi Task-based Constraint modelling from the previous chapter. We then implement the selection matrix approach for a velocity controlled manipulator and validate it, together with a surface tracking strategy, using both the Baxter robot and a robot prototype specifically designed for the task of cleaning the front panels of the train cabs.

3.1 Introduction

Interaction with constrained environments, such as rigid surfaces, mainly relying on haptic communication — using touch/force sensing — is a skill that humans often employ in ordinary everyday tasks, such as cleaning a surface. This is considered to be a constrained motion because the physical boundaries of the surface restrict our hand motion. Therefore, the movement of the hand is constrained to the object surface while keeping contact with it during the task [100]. We easily handle this kind of interaction with a variety of rigid objects without breaking them or hurting our arms, by properly adjusting our hands' positions and stiffness [174]. Even without using our vision — with our eyes closed — we can still trace an unknown surface using only tactile information.

A robotic system endowed with similar capabilities would be able to automate a

(a) Manual cleaning of a fuel powered train. Wembley depot, London.

(b) Manual cleaning of a electric train. Willesden depot, London.

Figure 3.1: Manual cleaning operation of two train cab fronts. London, 9th of June of 2016.

significant number of manual labour operations. One such operation is the cleaning process of the train cab fronts. The current process includes mechanised train washers that are unable to clean the train cab front ends due to complicated shapes. As a result, human operators manually clean the train cab fronts, as shown in Figure 3.1, raising health and safety concerns, including working in non ergonomic postures while subject to bad weather conditions.

Despite being a task easily achieved by humans, few works address the challenge of using a robot to clean a 3 dimensional (3D) surface. Hess et al. [62] state that at that time they were unable to find any manipulator robot that could clean arbitrary 3D surfaces, and propose an algorithm to cover a 3D surface using a redundant manipulator, by making use of an explicit surface model generated from a point cloud obtained using a Kinect sensor. However, the known poor performance of Kinect sensors in outdoor environments, under ambient infra-red radiation, renders this sensing modality unsuitable for a train cleaning application. Moreover, guaranteeing a safe pressure between the end-effector and the train surface would require fairly precise measurements, which are hard to obtain in practice for such large surfaces in outdoor environments. Notwithstanding, a skilful person would easily complete this task (even with the eyes closed) by using haptic information to adjust the force applied in the right direction, rather than guaranteeing a precise position of the hand.

The cleaning manipulator has to be able to wipe the train surface without damaging it, while applying enough contact pressure to guarantee the contact between the cleaning tool and the train surface. This particular application highlights the importance of the study and implementation of force control techniques, a quite mature and developed field — Villani and De Schutter [167] present us with a comprehensive review on the force control literature. Jamisola et al. [72, 73] and Oetomo et al. [115] successfully employ a mobile manipulator in the operation of aircraft canopy polishing, using simultaneous force and position control and the Operational

Space Formulation (OSF), proposed in [77, 79]. The aircraft polishing manipulator end-effector tool applies a roughly constant force (10 ± 4N) in a bidirectional polishing motion along a surface without requiring its explicit model, while subject to perturbations introduced by the operator moving the manipulator's mobile base. The controller adjusts the end-effector orientation in order to minimize the torques between the polishing tool and the surface, resulting in a tool always perpendicular to the surface. By aligning the tool with the surface normal, the direction of the polishing force is known, allowing the control of the polishing force in the right direction.

For the process of cleaning a train cab front, we propose a similar control approach used in the polishing manipulator work, with the adaptation that the output commands are joint velocities rather than joint torques, similar to the hybrid position/-force control from Raibert and Craig [135], allowing the use of typical industrial robots that lack torque control capabilities. Moreover, we incorporate the global information of the workspace area and location to satisfy coverage of the surface. Furthermore, we show the equivalence of the simultaneous position/force control approach, often used in these practical applications, with the multiple task-based constrained abstraction discussed in the previous chapter. The previous chapter focused on modeling the equations of motion of a task-based constrained system, which is key for understanding the movement of robots in contact, this chapter looks at some of the control — action — aspects of robots moving in contact.

3.2 Background

This section overviews two of the main control strategies for manipulation with contact/force interaction, namely the Hybrid Position/Force Control and the Operational Space Formulation.

3.2.1 Hybrid Position/Force Control

Impedance, stiffness and damping control are useful tools to actively change the dynamic behaviour of a manipulator in order to improve the interaction with the environment [144, 171]. These techniques belong to the class of indirect force control methods, i.e. the robot regulates the force being applied by the end-effector via motion control [167]. In the example of cleaning a surface, for a planar surface with known positioning it might be enough to adjust the stiffness of the arm in the direction perpendicular to the surface and set the goal positions to penetrate the surface [85]. By being compliant in that direction, the arm would comply with the surface and would apply a force to the surface proportional to the displacement between the surface position and the penetration set-point position.

Direct force control, on the other hand, is a control scheme where the feedback closure happens on the measurement of the force instead of the position. Typically, a force-torque sensor at the wrist of the robot measures the applied operational force at the end-effector, rather than using the dynamic model of the manipulator to estimate it through (2.15), as in [174]. A known — almost iconic — approach for direct force control in interaction tasks with the environment, introduced by Raibert and Craig [135] similar to Mason [99] theoretical work, is the hybrid position/force control approach. This approach combines force and position information to simultaneously satisfy position and force constraints, and consists in dividing the operational space into two subspaces. One of the subspaces implements a position controller - the position controller can even be a compliant or impedance controller. The other subspace implements a direct force controller. The hybrid position/force control approach assumes that these two subspaces are orthogonal and, thus, avoids interference between controllers. One needs to analyse and check this assumption, because for the case where the assumption involves the orthogonality of the Twist and Wrench spaces, a difference in units or referential origin can lead to misleading results, as evidenced by Duffy [46].

For the example of tracing a surface, we could divide the constrained operational space in a subspace containing the directions orthogonal to the surface and another subspace containing the directions tangential to the surface. In the first one, a force controller would track a given contact pressure between the manipulator and the surface. In the directions tangential to the surface, a position controller would track some end-effector trajectory for traversing all the surface area.

3.2.2 The Operational Space Formulation

Khatib [77] proposed a hybrid position/force control approach for simultaneous motion and force control of a robot manipulator in the operational space, where the end-effector acceleration $\ddot{x}$ encompasses the two components from both controllers

$$\ddot{x} = \Omega \ddot{x}_m + \overline{\Omega} \ddot{x}_f. \tag{3.1}$$

where $\ddot{x}_m$ and $\ddot{x}_f$ are, respectively, the task-accelerations resulting from the motion and force controllers. The selection matrices Ω and $\overline{\Omega}$ respectively define the direction of motion and force control [126]. Therefore, a task-space constraint corresponds to either a motion-task, with control over the position, or rigid constraint task, with control over the interaction force. Note that all task-space accelerations belong to the same space, i.e. $\ddot{x}, \ddot{x}_m, \ddot{x}_f \in \mathbb{R}^{n_b}$, and consequently $\Omega, \overline{\Omega} \in \mathbb{R}^{n_b \times n_b}$.

In a constrained task where the end-effector applies a desired force f_d on a surface, the external force applied is

$$\lambda = -\overline{\Omega} f_d. \tag{3.2}$$

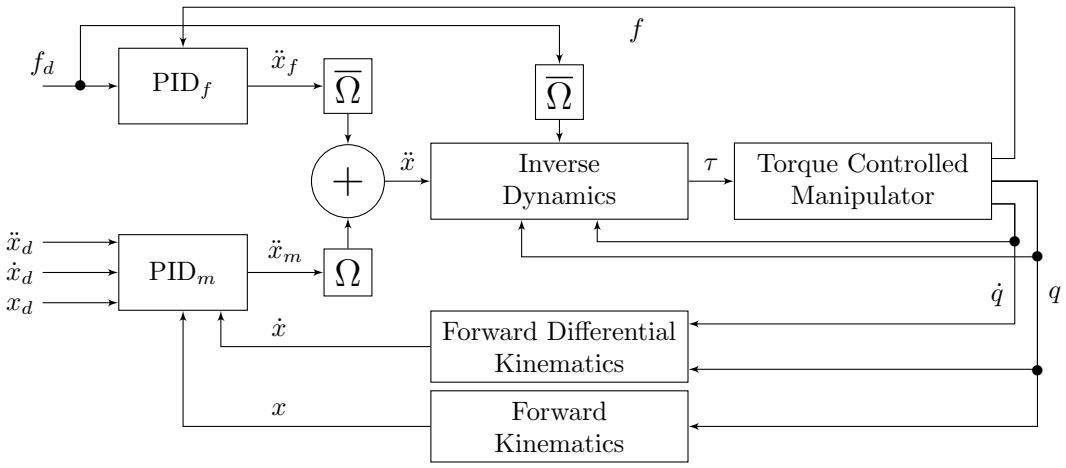


Figure 3.2: Diagram of a hybrid position/force controller for simultaneous motion and force control of a robot manipulator in the operational space, using a torque controlled robotic manipulator, i.e. we command the torques of each of the joints of the manipulator.

By substituting (3.1) and (3.2) in the constrained inverse dynamics (2.15), and in turn substituting (2.15) in the generalized force solution (2.18), for $\tau_\varepsilon = 0$, we get

$$\tau = A^T(q)\left(M_x(q)(\Omega\ddot{x}_m + \overline{\Omega}\ddot{x}_f) + \overline{\Omega}f_d\right) + A^T(q)h_x(q,\dot{q}), \tag{3.3}$$

which computes the manipulator joint torques for a task with simultaneous force and position control, and where A represents the robot manipulator Jacobian, which is a specific type of task-constraint Jacobian matrix. Figure 3.2 illustrates schematically the controller in Equation (3.3). The Manipulator block corresponds to a real physical torque controlled robot manipulator. We assume access to the forward kinematics model — $x = \phi(q)$ — as well as the manipulator's Jacobian — $\dot{x} = A(q)\dot{q}$. There are two Proportional-Integral-Derivative (PID) controllers regulating the end-effector position x and force f to achieve the respective desired values x_d and f_d. The control signal resulting from the PID controllers $\ddot{x}$ determines the joint torques τ, computed using the inverse dynamics 2.15. One exemplary application of the operational hybrid position/force control approach is in haptic teleoperation based on contact force control [126], which is particularly useful when employed in remote environments, such as deep-sea manipulation [28].

3.2.3 Manipulation Interaction Tasks

Concerning interaction between a manipulator and objects, there are other works that concentrate on exploring and tracking unknown surface geometries and properties. Ganesh et al. [57] propose a novel and unified control paradigm encompassing position, force, and impedance control, where they estimate the texture of the surface

using information of the level of compensation of the arm impedance, for maintaining a constant pressure on the tracking surface. Jamisola et al. [74] addresses the exploration of unknown surface with discontinuities, for the case of a two dimensional task space. Their work presents two methods addressing the problem of exploring discontinuous surfaces. The first method uses a relative weighting, dynamically adjusted according to the sensed torque, of force and position controllers simultaneously active in both directions. The second method implements a hybrid position/force controller, with force control on the axis normal to the surface and position control on the axis perpendicular to the surface, where the axis constantly rotates according to the sensed torques.

Back et al. [13] develop a new sensing device — a two dimensional robot finger — endowed with force sensing. This system is able to estimate the geometry and friction properties of several smooth surfaces. Once again, the study is limited to a two dimensional environment. Bosheng et al. [25] estimates the environment damping and stiffness parameters during interaction between robot and environment, by making use of the Recursive Least Squares algorithm in an adaptive impedance control method, during a task of tracing a planar surface with arbitrary inclination and varying damping and stiffness. While these two previous works estimate the material properties of environment constraint, there are other works that focus on the estimation of the constraint geometry itself, through the motion of the robot [24, 157].

On the application of cleaning robots, Hess et al. [62] propose a method for using redundant manipulators for cleaning complex 3D surfaces. The paper addresses the problem of path coverage planning constrained to a surface by using the manipulator null space to minimize the joints movement. However, this strategy leads to an erratic end-effector trajectory unlike typical human cleaning movements. Normal human cleaning movements tend to resemble movements like a raster scan or a spiral where the dirt goes from the inner to the outer side of the surface. Leidner et al. [85] use a raster scan trajectory for wiping a window, leading to a more human like movement. For control, they also use the selection matrix approach discussed in the previous section. Another way of obtaining natural trajectories in the operational space is through demonstration. Kormushev et al. [81] uses a humanoid to clean a vertical board, where the trajectory and the required forces are acquired through direct human demonstration. The robot uses a hybrid position/force controller to apply the learned trajectories and forces. As the robot is a redundant system, it uses the null space for keeping balance.

Polishing manipulators perform a similar job to cleaning manipulators — both have to sweep through a surface while maintaining a specific contact pressure. Nagata et al. [109] present a position/force controller implementation for a polishing robot, where they provide a CAD model of the surface, allowing the use of a position feed forward controller in conjunction with a compliant feedback controller. The control of the polishing force serves the purpose of improving the final quality of the polished surface. Some works, such as [72, 73, 115], show the applicability of the

hybrid force/motion control in tasks such as polishing a smooth unknown curved surface, using the operational space formulation from [77]. In [72], the end-effector is able to apply a roughly constant force (10 ± 3N), while subject to perturbations such as moving the position of the surface and the manipulator's mobile base. For the process of cleaning a train cab front, we shall use a similar control approach used in the polishing manipulator work, with the adaptation that the output commands are joint velocities rather than joint torques.

3.3 Operational Space Control with Constraints

Relation with Stack of Jacobian Matrices' Approach

This section delves into the equivalence between the Operational Space Controllers with Constraints independently proposed by De Sapio and Khatib [34] and Mistry and Righetti [101]. To do so, we start by recalling the task force in (2.40) resulting from deriving that specific task dynamics component from the task space dynamics (2.16) for a stack of Jacobian matrices

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}. \tag{3.4}$$

We can then rewrite (2.40) as

$$f_2 = M_2 \left(\ddot{x}_2 - \dot{A}_2 \dot{q} - A_2 \overline{A}_1 (\ddot{x}_1 - \dot{A}_1 \dot{q}) + A_2 M^{-1} P_{M1}^{\top} h \right) - \lambda_2. \tag{3.5}$$

Here we take the idea for splitting a Jacobian matrix into a stack of two Jacobian matrices from [34, 35]. However, De Sapio et al. [35] solely propose this split concept without fully deriving the expression (3.5), which is valid for $\text{rank}(A) = \text{rank}(A_1) + \text{rank}(A_2)$. Moreover, De Sapio et al. [35] propose this stacking for the specific case that A_1 is a constraint Jacobian (rigid constraint), and A_2 is the Jacobian of the robot. In this thesis, the definition of task-based constraints unifies these two categories and makes them belong to the same category of task-based constraints. But let's consider the proposal of De Sapio et al. [35]. For that specific case scenario, we will have $\ddot{x}_1 = 0$ for a rigid constraint and $\lambda_2 = 0$ for a motion-task. Therefore, (3.5) becomes

$$f_2 = M_2 \left(\ddot{x}_2 - \dot{A}_2 \dot{q} + A_2 \overline{A}_1 \dot{A}_1 \dot{q} + A_2 M^{-1} P_{M1}^{\top} h \right), \tag{3.6}$$

which corresponds to the motion-task control, whereas its homologous f_1 would correspond to the constraint forces.

Regarding the approach based on the PD, using the equalities $P_M M^{-1} = M^{-1} P_M^\top = M_c^{-1} P$ and $\overline{A} = M_c^{-1} R A^\dagger$, we see that for $R = I$, (3.6) becomes

$$f_2 = M_2 \left(\ddot{x}_2 + A_2 M_{c1}^{-1} P_1 h - (\dot{A}_2 - A_2 M_{c1}^{-1} A_1^\dagger \dot{A}_1) \dot{q} \right) \tag{3.7}$$

which is exactly the same as the operational space dynamics with constraints found in [101]. We can find practical implementations of this controller in [117]. This result shows that a framework of multiple Task-based Constraints generalizes the task space plus constraint formulations from De Sapio and Khatib [34] and Mistry and Righetti [101] and, furthermore, allow us to reason about them as equivalent. Note that both Park and Khatib [125] and Mistry and Righetti [101] propose equations for the partial dynamically consistent Jacobian $A_2^\#$ and the partial task-space inertia matrix M_2, but using different formulations: the first based on the inertia-weighted projection matrix P_M and the inertia matrix M, i.e. $M_2 = \left(A_2 P_{M1} M^{-1} A_2^\top \right)^{-1}$ and $A_2^\# = P_{M1} M^{-1} A_2^\top M_2$; and the second based on the orthogonal projection matrix P and the constraint inertia matrix M_c, i.e. $M_2 = \left(A_2 M_{c1}^{-1} P_1 A_2^\top \right)^{-1}$ and $A_2^{\#\top} = M_2 A_2 M_{c1}^{-1} P_1$.

Relation with the Selection Matrix approach

We can find a second equivalence of the control law in (3.5) with the Generalized Task Specification Matrices approach proposed by Khatib [77], which is another approach widely used in the literature. By analysing the task acceleration computation (3.1) from the hybrid position/force control approach for simultaneous motion and force control [77], we see that despite $\ddot{x}_m$ and $\ddot{x}_f$ having the same dimensionality of $\ddot{x}$, the selection matrices Ω and $\overline{\Omega}$ only pick some of their elements. In the original formulation though [77], Ω and $\overline{\Omega}$, called the generalized task specification matrices, are in general non-diagonal because they are the result of a rotation transformation of the actual selection matrices that are diagonal. This happens because Khatib [77] uses a Jacobian of the robot which maps the configuration velocities to the operational space velocities relative to a global frame and, therefore, the direction of the motion or force tasks might require a rotation of the global frame representation. This thesis work, in line with the concept of always defining generic tasks-based constraints, makes use of Jacobian matrices mapping directly to the task space of interest, i.e. the task Jacobian already encodes the rotation rather than the selection matrices encoding such transformation. Therefore, Ω and $\overline{\Omega}$ are always diagonal matrices with diagonal elements being either 0 or 1.

The reason why this distinction in definition of the selection matrices is important is that for the case where they are always diagonal matrices, then all they do is to pick specific elements of the vectors x_m and x_f, ignoring all the other ones. Therefore, we can always permute the elements of x_m and x_f (permuting the respective matrices such as Ω,$\overline{\Omega}$,A) such that

$$\ddot{x} = \Omega \ddot{x}_m + \overline{\Omega} \ddot{x}_f = \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix}, \tag{3.8}$$

where $\ddot{x}_1$ contains the acceleration selected elements resulting from the force control component, and $\ddot{x}_2$ contains all the acceleration selected elements resulting from the position control component. Similarly, the desired forces (3.2) would become

$$\lambda = -\overline{\Omega} f_d = \begin{bmatrix} 0 \\ \lambda_2 \end{bmatrix}, \tag{3.9}$$

where λ_2 contains the selected elements from the desired force $-f_d$. We can then easily see by inspection, that the task-space force in the torque commands Expression (3.3) resulting from Khatib [77]'s simultaneous position/force control approach, corresponds to the task-space force (2.39) resulting from the stack of Jacobian matrices, for which we use the task-space acceleration (3.8) and the task-space external force (3.9). In this case, the acceleration $\ddot{x}_1$ is the result of the force PID compensation and the acceleration $\ddot{x}_2$ is the result of the position PID compensation in the intended directions. This result shows that the multi Task-based Constraint approach, consisting in simply stacking the Jacobian matrices for each task, can also generalize the simultaneous position/force controllers based on selection matrices (as the one from the control diagram in Figure 3.2).

3.4 Surface Tracing with a Kinematic Robotic Manipulator

This section details the control strategy to simultaneously adjust the end-effector orientation, the interaction force, and the end-effector motion tangential to the target surface, for the application of using a redundant robotic manipulator for wiping a surface. This involves combining force and position information in a single local reference frame and specifying the force and position control axis. Then, the resolved motion rate control — standard inverse differential kinematics solution — translates the desired end-effector motion to the robot arm joint velocity commands. The block diagram in Figure 3.3 summarizes the main components and transformations described in this section.

Wiping Motion Description

When wiping a surface we can distinguish two distinct tasks: (i) aligning the wiping tool with the surface and ensuring contact; (ii) moving the wiping tool along the surface with a specified pattern of motion. The first task of aligning the wiping tool would in principle require a good model of the surface geometry and its position relative to the robot. However, it is impractical to precisely position a large object such as a train, or to obtain a good scan of its geometry in an outside environment

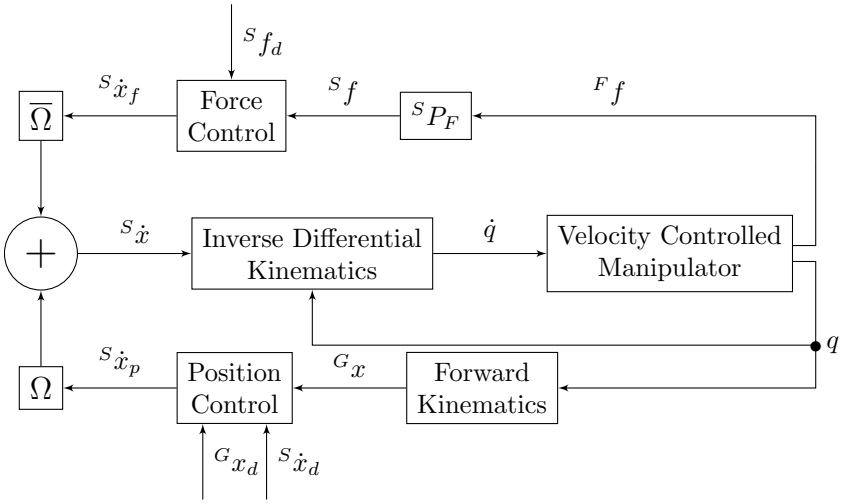


Figure 3.3: Simultaneous force and position control diagram, using resolved motion rate control, i.e. we directly command the velocities of the joints of the robotic manipulator.

subject to varying lighting conditions. An alternative is to use force information to ensure the contact and alignment requirements.

Figure 3.4 illustrates an end-effector wiping a curved surface, with its main components, their respective reference frames, and possible interaction forces and torque. If the surface geometry is unknown, the robot can use force feedback to align the tool, given that when in contact with the surface, a misalignment between the wiping tool and the surface results in a torque at the interaction point. Therefore, the robot can align the end-effector by simply minimizing this interaction torque, and guarantee a given contact pressure by controlling the normal contact force. While aligning the tool, the robot also has to perform a wiping motion along the surface, suggesting that it has to simultaneously control these interaction forces and torques and the wiping motion.

Specification of Control Directions

The simultaneous position and force control, according to the Operational Space Formulation [77], requires specifying the direction axis in the operational space (i.e. end-effector task space) in which we control force or position independently. So, for example, let u be the control input in the task space. The control u results from combining the desired motion u_m and desired force interaction with the environment u_f. However, we must ensure that the motion component is unaffected by the force component, and vice versa. In the Operational Space Formulation, this is accomplished by specifying the generalized task matrices Ω and $\overline{\Omega}$, so that $u = \Omega u_m + \overline{\Omega} u_f$, where $\overline{\Omega} = I - \Omega$ and I is the identity matrix with the appropriate dimension.

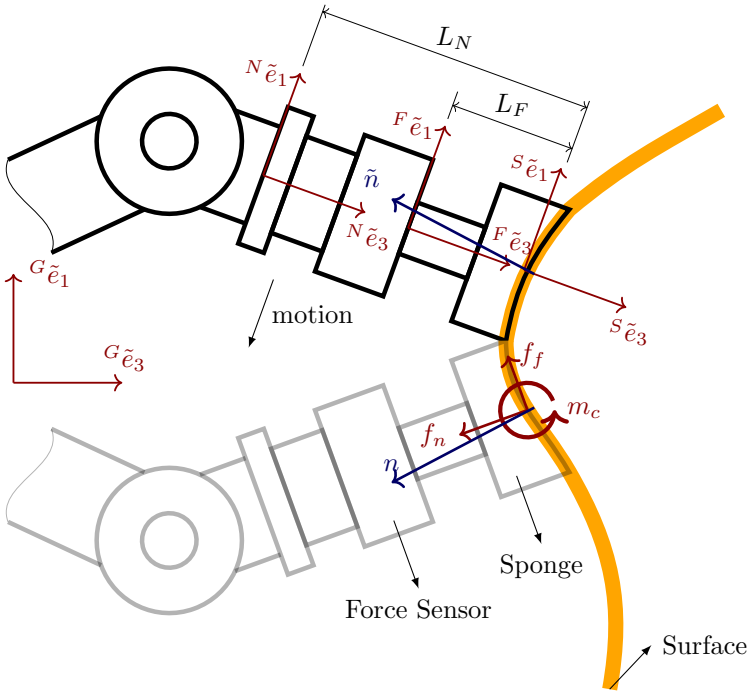


Figure 3.4: A two dimensional illustration of a robot end-effector interacting with a curved surface, portraying two particular configurations of the robot arm — with different opacity. The end-effector wiping tool includes a force/torque sensor and a soft interface material (sponge). The four coordinates systems represented are: the global coordinate system G; the standard end-effector local coordinate system N; the force sensor coordinate system F; and the surface contact coordinate system S. The interaction of the wiping tool and the surface causes a friction force f_f, a normal force f_n, and a contact torque m_c, where the arrows indicate the respective directions of the values. The alignment of the tool with the surface normal $\tilde{n}$ is achieved by minimizing the contact torque m_c, and the contact ensured by controlling the normal force f_n.

The form of the matrix Ω depends on the particular task the robot executes and on the chosen reference frame. For the wiping motion, in order to simplify the construction of this matrix, we can specify the control input in a local end-effector frame as, for instance, ${}^{S}u$ (S is the reference frame with origin in the contact point and aligned with the wiping tool as shown in Figure 3.4). Then, the generalized task specification matrix becomes

$$\Omega = \text{diag}(1, 1, 0, 0, 0, 1), \tag{3.10}$$

meaning that the position control occurs in the $\tilde{e}_1$ and $\tilde{e}_2$ end-effector local axis for translation (tangential directions to the surface for a fully aligned tool), and in the $\tilde{e}_3$ axis for rotation. Note that the axis $\tilde{e}_2$ is missing from the illustration in the Figure 3.4 due to it being a two dimensional illustration and $\tilde{e}_2$ being perpendicular to the illustrated axes $\tilde{e}_1$ and $\tilde{e}_3$, i.e. $\tilde{e}_2$ is perpendicular to the page. The force control occurs in the $\tilde{e}_3$ end-effector local axis for linear force (perpendicular direction to the surface for a fully aligned tool), and around the $\tilde{e}_1$ and $\tilde{e}_2$ end-effector local axis for torques (torques are set to zero to guarantee that the end-effector is always perpendicular to the surface). Here we assume a 6 dimensional operational space, composed of 3 linear positions/forces and 3 angular positions/torques.

Resolved Motion Rate Control

In the Operational Space Formulation [77] for simultaneous motion and force control of robot manipulators, the task space control input u corresponds to the end-effector acceleration, converted to joint torques through direct dynamics. However, even though lab robots are often torque controlled, many industrial robots lack such torque control capabilities, either due to hardware limitations, such as the joints lacking torque sensors, or simply because velocity controllers are more widely available and integrated in servo motors. Moreover, the computation of the joint torques requires the dynamic model of the robot manipulator, which for industrial applications can be difficult to obtain. For instance, a robot for washing the train front panels would require the flow of water and detergent through flexible pipes to the wiping tool, that might be external to the rigid links. Therefore, for this application, it would be more viable to directly control the joint velocities instead of the joint torques.

Once we define the task space control input u to be the end-effector velocity ${}^{S}\dot{x} \in \mathbb{R}^6$ (expressed in the local frame S), a particular solution for the joint velocities $\dot{q} \in \mathbb{R}^n$, for a robot with n joints, is simply

$$\dot{q} = A^{\dagger}\, {}^{S}\dot{x}, \tag{3.11}$$

where $q \in \mathbb{R}^n$ are measured joint position values and $A^{\dagger} \in \mathbb{R}^{n_q \times 6}$ is the pseudo inverse of the manipulator Jacobian A. For a manipulator that has more Degrees of Freedom than the number of task space variables ($n_q > 6$), the pseudo inverse

Jacobian corresponds to the mapping that minimizes the square of the joint velocities' euclidean norm $\|\dot{q}\|^2$. For the case that $n_q = 6$, the pseudo inverse degenerates into the inverse, and the solution 3.11 for $\dot{q}$ is unique. Whitney [170] and Umetani and Yoshida [165] refer to this type control of the task velocities through a differential kinematics solution (3.11) as Resolved Motion Rate Control, which is a solution widely adopted in many robotic approaches [47, 88].

In summary, Resolved Motion Rate Control consists of: (i) defining the desired end-effector velocities; (ii) computing the corresponding joint velocities using a generalized inverse of the robot Jacobian; (iii) and sending the resulting joint velocities to the motor controllers. In order to specify the end-effector position instead of the end-effector velocity, we just need to differentiate the desired position trajectory.

Admittance Control

Knowing how to control the end-effector velocity or position through joint velocity control, the next step is to compute the end-effector displacement in a certain direction to produce a desired force. As we perform compliant interaction with the surface through a sponge, we can set a proportional control on the force giving the resulting end-effector velocity as

$$^{S}\dot{x}_f = K_p(^{S}f_d - ^{S}f), \tag{3.12}$$

where $^{S}f_d$ is the desired generalized force vector, ^{S}f is the measured force, and K_P is a diagonal matrix of the gains applied in each axis. For the case of the sweeping robot that only controls the force in the end-effector $\tilde{e}_3$ direction and torques in the $\tilde{e}_1$ and $\tilde{e}_2$ directions, then

$$K_p = \text{diag}(0, 0, K_z, K_{xy}, K_{xy}, 0). \tag{3.13}$$

Reference frames

The force/torque sensor returns the measurements with respect to a frame F, different from the standard end-effector local frame N, which in turn differs from the frame of interest S, with its origin at the contact point between the wiping tool and the surface. Therefore, before any control considerations we address the transformations of the measurements into the reference frame of interest.

We assume to have access to the robot forward kinematics model ϕ which, given q, returns the end-effector position and orientation in the form of: the frame N origin Cartesian coordinates $^{G}x_N \in \mathbb{R}^3$; and the rotation matrix $^{G}R_N \in \mathbb{R}^{3\times3}$, relative to the global frame G, as

$$(^{G}x_N, ^{G}R_N) = \phi(q). \tag{3.14}$$

Then, given the position of the contact point relative to N (${}^{N}x_S \in \mathbb{R}^3$), its coordinates relative to the global frame are simply given by

$$ {}^{G}x_S = {}^{G}x_N + {}^{G}R_N \, {}^{N}x_S. \tag{3.15} $$

Due to the distance between the contact point and the force/torque sensor, the contact forces/torques are inaccessible and, therefore, we must estimate them from the measured ones. Consider the example from Figure 3.4. We are interested in minimizing the contact torque m_c, which is the torque around the axis ${}^{S}\tilde{e}_2$ (${}^{S}\tau_2 = m_c$). However, due to the distance L_F between the contact point and the force/torque sensor, its reading ${}^{F}\tau_y$ is a sum of the contact torque m_c and the effect of the force f_f applied to the contact point. Therefore, we can estimate the contact torque at the contact point as $m_c = {}^{F}\tau_y + L_F f_f$. More generally, given a generalized force

$$ {}^{F}f = \begin{bmatrix} {}^{F}f_1 & {}^{F}f_2 & {}^{F}f_3 & {}^{F}\tau_1 & {}^{F}\tau_2 & {}^{F}\tau_3 \end{bmatrix}^T, \tag{3.16} $$

obtaining the forces acting on the end-effector contact point consists in applying the relation

$$ {}^{S}f = {}^{S}T_F \, {}^{F}f. \tag{3.17} $$

where ${}^{S}T_F$ is a transformation matrix given by

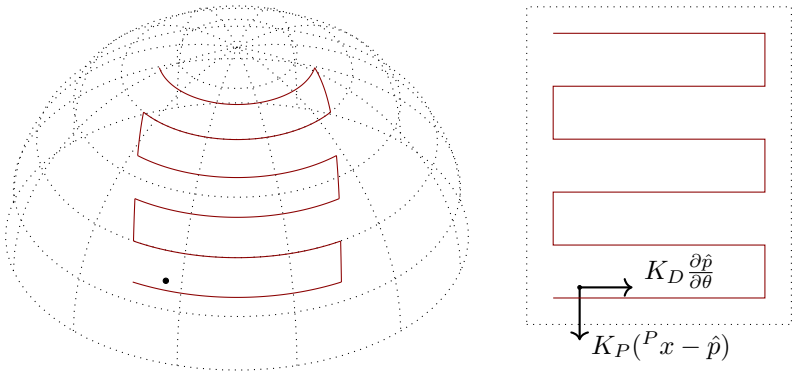
$$ {}^{S}T_F = \begin{bmatrix} I_3 & 0 \\ \left[{}^{S}x_F \right]_\times & I_3 \end{bmatrix}, \tag{3.18} $$

and ${}^{N}x_F$ is the position vector of the origin of the coordinate system F relative to the coordinate system S. The operator $[]_\times$ represents the transformation of a vector in a skew-symmetric matrix.

Trajectory Tracking

When assuming that the surface is unknown, one of the main questions is: how can we specify a trajectory? Obviously, we are unable to specify an exact trajectory on the surface without knowing its geometry, but even then we would like to cover a region following a specific pattern of motion.

The solution we propose is specifying only a planar movement rather than specifying the whole 3D trajectory. Then, the actual trajectory would be a projection of this planar trajectory onto the surface. A set of assumptions are that: the dimensions of the planar trajectory are smaller than the outer dimensions of the 3D surface; the gap between the the paths that traverse the surface is small relative to the cleaning tool size (so the tool covers all the surface area even for the most inclined surfaces); and the robotic manipulator can reach the projected positions. Figure 3.5a shows the result of projecting a raster scan in Figure 3.5b onto a spherical surface.



(a) Illustration of the projection of a vertical raster scan onto a spherical surface.

(b) Example of a planar trajectory - a raster scan.

Figure 3.5: Illustration of the result of projecting a 2D path onto a given surface geometry, resulting in a 3D path. Point indicating a possible projection of the end-effector position while tracking a desired trajectory. Vectors indicating the computed projected end-effector velocity.

By describing the planar trajectory as a parametric function $p(\theta) \in \mathbb{R}^2$, we simply define the commanded velocity ${}^P\dot{x} \in \mathbb{R}^2$ in the planar space as proportional to the derivative of this function

$$ {}^P\dot{x} = K_D \frac{dp(\theta)}{d\theta}\Big|_{\theta=\theta_i}, \tag{3.19} $$

where P stands for the planar trajectory space. By definition, the norm of the parametric function derivative is always 1 for every value of θ_i. Then we project this velocity onto the surface as

$$ {}^S\dot{x} = {}^S T_P \, {}^P\dot{x}, \tag{3.20} $$

where ${}^S T_P \in \mathbb{R}^{6\times 2}$ is a matrix that projects the planar trajectory to the local frame S.

If, while sweeping, the end-effector drifts from the path, we have to compensate this deviation. Figure 3.5a shows an example of a possible end-effector drift that, the end-effector position (black dot) no longer coincides with the projected trajectory. Given that the surface geometry is unknown, we will compute the deviation error in the projection plane P.

We first project the current end-effector position in the global frame ${}^G p_S$ to the plane of the planar trajectory, obtaining ${}^P x$. Figure 3.5b shows the respective representation of the ${}^P x$ next to the planar trajectory. Then, we compute

$$ \hat{\theta}_i = \arg\min_{\theta} \| {}^P x - p(\theta) \|, \tag{3.21} $$

which minimizes the distance between the current position and the trajectory. By plugging the estimated parameter $\hat{\theta}_i$ in the parametric function

$$ \hat{p} = p(\hat{\theta}), \tag{3.22} $$

we obtain the closest trajectory position.

The desired velocity in the planar space P results from adding the following two terms as

$$^{P}\dot{x} = K_D \frac{dp}{d\theta} + K_P(^{P}x - \hat{p}), \tag{3.23}$$

where one corrects the error between the current position and the trajectory and the other is proportional to the derivative of the parametric function, so the end-effector keeps moving forward while correcting the position. Equation (3.20) transforms the result from (3.23) to the local surface frame S. Finally, if we wish the tangential velocity of the end-effector to the surface to be constant and equal to $C_v\,\mathrm{m\,s^{-1}}$, then we still need to normalize it as

$$^{S}\dot{x}_p = \frac{\Omega\, ^{S}\dot{x}}{\|\Omega\, ^{S}\dot{x}\|} C_v. \tag{3.24}$$

The advantage of using a parametric representation of the planar trajectory is that this isolates the implementation of different trajectories by simply specifying the two functions — $p(\theta)$ and $\frac{\partial p(\theta)}{\partial \theta}$. Changing the desired motion pattern implies changing the parametric function and its derivative.

3.5 Case Studies

3.5.1 Verifying the Equivalence of Task Controllers with Constraints

The goal of this first case study is to verify the equivalence between different task controllers with constraints present in the literature. In order to do so, let us consider the planar serial robot arm with four links, illustrated in Figure 3.6, which for a given configuration q_j is in perfect contact and alignment with a flat surface. We then define two tasks for the robot to accomplish: a contact/force task consisting of keeping contact and alignment with the surface; and a motion-task responsible for the end-effector movement tangential to the surface.

We define the contact/force task coordinates as $x_1^\top = \begin{bmatrix} \mathrm{x} & \psi \end{bmatrix}$ and the corresponding task commanded forces as $f_1^\top = \begin{bmatrix} f_\mathrm{x} & f_\psi \end{bmatrix}$, where x is the position coordinate of the axis perpendicular to the surface and $\psi = \sum_i^4 q_{j[i]}$ is the global orientation of the end-effector, coinciding with the orientation of the surface. The motion-task coordinate is then $x_2 = \mathrm{y}$ and the corresponding motion-task commanded force is $f_2 = f_\mathrm{y}$, where y is the position coordinate of the axis tangential to the surface. The corresponding motion and contact task Jacobian matrices are then

$$A_1 = \begin{bmatrix} \frac{\partial \mathrm{x}}{\partial q} \\ \frac{\partial \psi}{\partial q} \end{bmatrix} \text{ and } A_2 = \frac{\partial \mathrm{y}}{\partial q}.$$

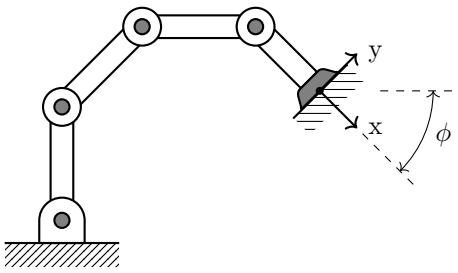


Figure 3.6: Planar serial robot arm with four links in a configuration $q_j^\top = \begin{bmatrix} 90° & -45° & -45° & -45° \end{bmatrix}$.

More concretely now, the goal of this experiment is to use the different controllers for motion-task with constraints found in the literature to compute the motion-task force control f_{y}, given a desired tangential motion acceleration $\ddot{\mathrm{y}}_d$ and desired contact force λ_d. These requirements result in the following desired task accelerations and forces

$$\ddot{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; \ddot{x}_2 = \ddot{\mathrm{y}}_d, \; \lambda_1 = \begin{bmatrix} \lambda_d \\ 0 \end{bmatrix}, \text{ and } \lambda_2 = 0.$$

We now overview four alternative ways of computing the motion-task force commands f_{y} and categorize them, just for the purposes of this experiment, as controllers (A) , (B) , (C) , and (D) , whose application will result in the computation of the respective motion-task force commands $f_{\mathrm{y}A}$, $f_{\mathrm{y}B}$, $f_{\mathrm{y}C}$, and $f_{\mathrm{y}D}$, respectively.

(A) : The first controller requires simply stacking the task Jacobian matrices $A^\top = \begin{bmatrix} A_1^\top & A_2^\top \end{bmatrix}$, as proposed in Section 2.4, which implies stacking the task position vector $x^\top = \begin{bmatrix} x_1^\top & x_2^\top \end{bmatrix}$ and the task force commands vector $f^\top = \begin{bmatrix} f_1^\top & f_2^\top \end{bmatrix}$. This entails computing the task-space dynamics according to the Equation (2.16), repeated here for convenience

$$\boxed{f = M_x \ddot{x} + h_x - \lambda},$$

for the extended task acceleration $\ddot{x}^\top = \begin{bmatrix} \ddot{x}_1^\top & \ddot{x}_2^\top \end{bmatrix}$ and forces $\lambda^\top = \begin{bmatrix} \lambda_1^\top & \lambda_2^\top \end{bmatrix}$. The motion-task force command $f_{\mathrm{y}A}$ corresponds to the last component of f.

(B) : The second controller corresponds to Equation 3.6, which is a result of the multiple Task-based Constraints derivation in Section 2.4 followed by applying the conditions for task-motion with rigid constraints proposed by De Sapio and Khatib [34] and discussed in Section 3.3. The motion-task force command $f_{\mathrm{y}A}$ corresponds then to f_2, where

$$\boxed{f_2 = M_2 \left(\ddot{x}_2 - \dot{A}_2 \dot{q} + A_2 \overline{A}_1 \dot{A}_1 \dot{q} + A_2 M^{-1} P_{M1}^\top h \right)}.$$

(C) : The third controller is the operational space dynamics with constraints proposed by Mistry and Righetti [101], shown in Equation 3.7, which we show to be

algebraically equivalent to the controller (B), in Section 3.3. We have again that f_{yC} simply corresponds to f_2 computed as

$$\boxed{f_2 = M_2\left(\ddot{x}_2 + A_2 M_{c1}^{-1} P_1 h - (\dot{A}_2 - A_2 M_{c1}^{-1} A_1^{\dagger}\dot{A}_1)\dot{q}\right)}$$

(D) : Finally, the fourth controller is based on the selection matrix method, proposed by Khatib [77], where we redefine the task coordinates vector and the task commanded forces as

$$x' = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \psi \end{bmatrix} \text{ and } f' = \begin{bmatrix} f_{\mathrm{x}} \\ f_{\mathrm{y}} \\ f_{\psi} \end{bmatrix},$$

given that in this context it is more customary to stack the coordinates starting with Cartesian positions and then placing the orientations. For the task of moving tangentially to the surface we then define the desired motion and force accelerations and the desired contact forces as

$$\ddot{x}'_m = \begin{bmatrix} 0 \\ \ddot{\mathrm{y}}_d \\ 0 \end{bmatrix}, \ \ddot{x}'_f = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{ and } f'_d = \begin{bmatrix} -\lambda_d \\ 0 \\ 0 \end{bmatrix},$$

and the generalized task specification matrices as

$$\Omega = \mathrm{diag}(0, 1, 0) \text{ and } \overline{\Omega} = \mathrm{diag}(1, 0, 1).$$

The computation of the motion-task force command consists in applying the Expression

$$\boxed{f' = M_{x'}(\Omega\ddot{x}'_m + \overline{\Omega}\ddot{x}'_f) + \overline{\Omega} f'_d + h_{x'}},$$

discussed in Subsection 3.2.2. The motion-task force command f_{yD} corresponds then to the second component of f'.

To verify the equivalence between the previously presented controllers we generated a dataset $\{(q_j, \dot{q}_j, \ddot{x}_{dj}, \lambda_j)\}_{j=1}^{\mathcal{J}}$ of $\mathcal{J} = 1000$ different scenarios by uniformly sampling the robot configurations $q_{j[1]} \in [-\pi/2, \pi/2]$ and $q_{j[2..4]} \in [-2\pi/3, 2\pi/3]$ rad and the robot configuration velocities $\dot{q}_{j[i]} \in [-0.2, 0.2]\,\mathrm{rad\,s^{-1}}$ and desired task space acceleration $\ddot{x}_{dj} \in [-0.2, 0.2]\,\mathrm{m\,s^{-2}}$ and contact force $\lambda_{dj} \in [5, 10]\,\mathrm{N}$. Similarly to the experiment in Subsection 2.3.3, the robot in Figure 3.6 has identical links with equidistant centre of mass from the joints, and with length, mass, and inertia of 1 m, 1 kg, and 0.1 $\mathrm{kg\,m^2}$, respectively. We used Corke's [33] MatLab® toolbox for computing M, h, and the robot's Jacobian.

We then computed the motion-task force commands for all the 1000 samples using the four controllers detailed above. Using the controller (A) as reference we computed

the Mean Square Error (MSE) over the 1000 samples of the result of each controller with respect to (A) . The results were

$$\mathrm{MSE}(f_{yA}, f_{yB}) = 2.9644 \times 10^{-26}\,\mathrm{N}^2,$$

$$\mathrm{MSE}(f_{yA}, f_{yC}) = 2.8273 \times 10^{-25}\,\mathrm{N}^2, \text{ and}$$

$$\mathrm{MSE}(f_{yA}, f_{yD}) = 4.9790 \times 10^{-27}\,\mathrm{N}^2,$$

indicating that here is almost no difference between the obtained force commands with these four differently formulated controllers. In other words, as Sections 2.4 and 3.3 show the algebraic equivalence of the four controllers in study, the computation using the different controllers lead to identical results with very small deviation between each other.

3.5.2 Wiping a Non-flat Surface

For the second case study we implemented the simultaneous position/force control approach for kinematic robots using one of the Baxter arms from Rethink Robotics®, shown in Figure 3.7. Baxter features 7 DoF arms. The joints are physically compliant by integrated springs, [55]. This feature makes the robot safe to operate because if it hits a person or object, the springs will retract before damaging the object or hurting the person. However, this features also makes interaction movements more unstable, as it vibrates in certain movements, especially when making contact with objects, by encountering some resistance. Nevertheless, we were still able to use this platform for constrained motions. The goal of this case study is to experimentally validate the feasibility of using a simultaneous position/force control approach on a kinematic robot for wiping a surface of unknown geometry, without any pretence of being an exhaustive study on the performance of the proposed controller. We perform single trials using the physical robot, reporting the respective tracking errors.

For reliably obtaining the forces and torques applied by the Baxter's end-effector we used the Gamma F/T sensor from ATI industrial automation, shown in Figure 3.8b. This sensor measures forces and torques along the three Cartesian axis - 6 values in total. The sensor has a resolution of: 1/40 N for f_1 and f_2; 1/20 N for f_3; and 1/800 N m for the torques. The single-axis overloads are: ±1200 N for f_1 and f_2; ±4100 N for f_3; ±79 N m for τ_1 and τ_2; and ±82 N m for τ_3.

Given that the goal is to study the behaviour of the system when interacting with a smooth and curved surface, we used a surface with varying slope to test the system ability to adapt, as shown in Figure 3.9a. In order to have a compliant contact between the end-effector and the surface, the interface material should be soft, therefore, we attached a sponge to the tip of the end-effector, emulating the cleaning tool.

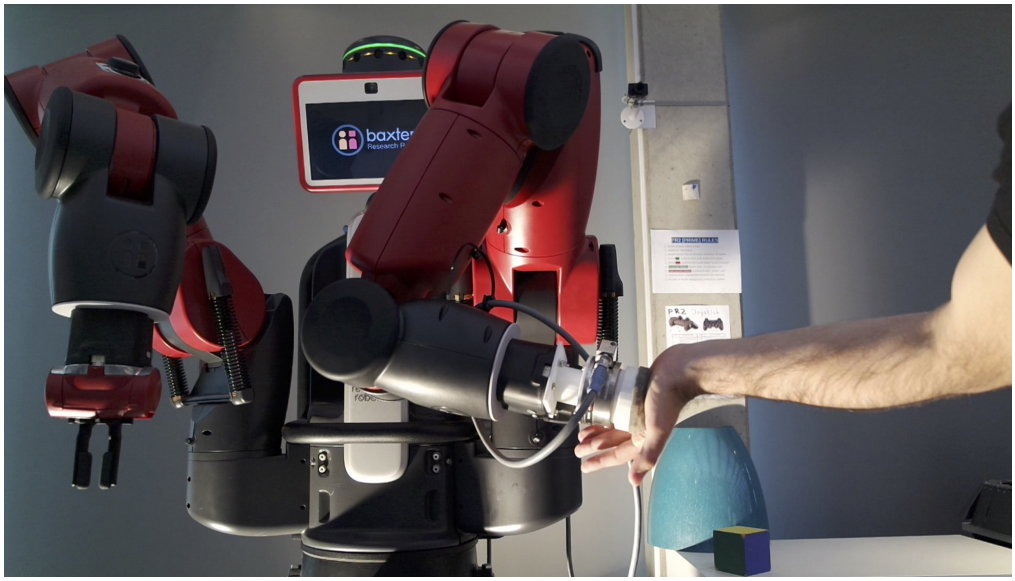
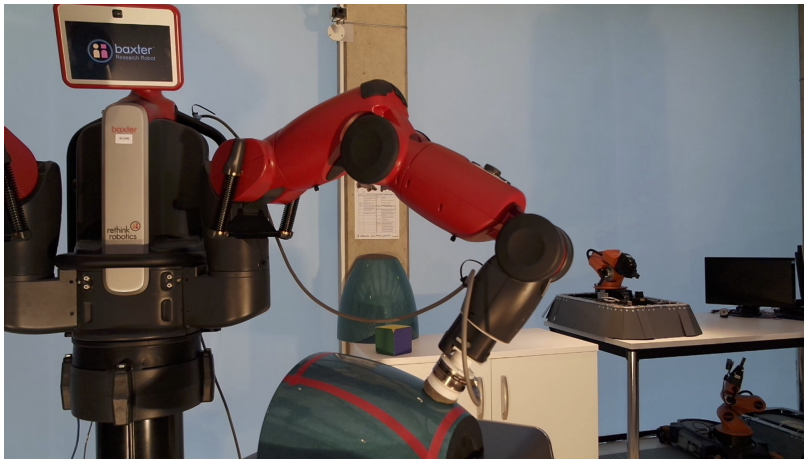
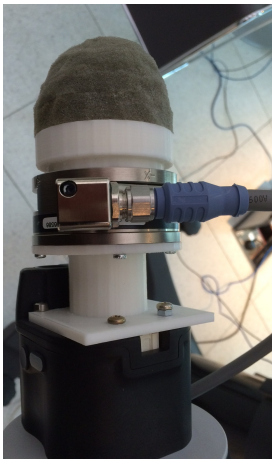


Figure 3.7: Baxter orienting the end-effector to be perpendicular to the hand.

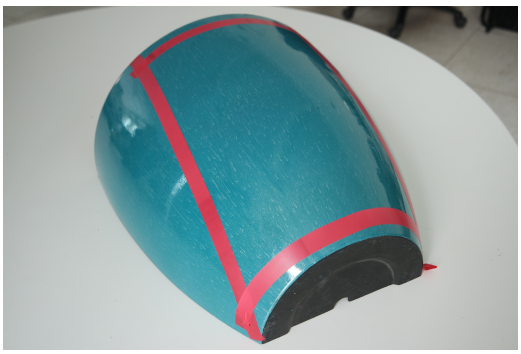


(a) Baxter robot wiping a curved surface.



(b) Force sensor.

Figure 3.8: Baxter robot and detail of the end-effector used in the experiment.



(a) Surface used in the experiments.

(b) Sponge used as compliant material to interact with the surface.

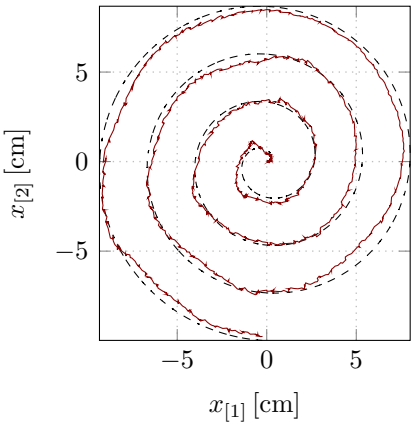
Figure 3.9: Interface elements: the surface and a compliant tip (sponge).

Figure 3.9b shows the sponge used to contact the surface. This extra compliance allows to accommodate some error in the positioning of the end-effector. However, this material brings the same effect as the springs in the Baxter joints or any other elastic material: reduced stiffness of the structure, adding some oscillation to the movements. To attach the sensor to the Baxter and the sponge to the sensor, we designed and 3D printed two plastic attachments. Figure 3.8a shows a general view of the Baxter with the sensor and sponge attached, and Figure 3.8b shows a more detailed view of the end-effector.

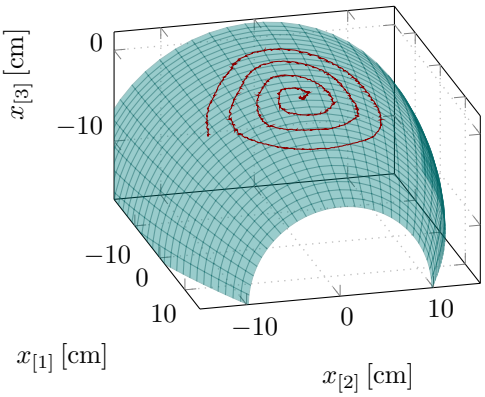
When observing the manual cleaning process of the train cab fronts, one of the simpler movements widely repeated resembled a spiral motion, as we can see in Figure 3.1a. Hence, we started by using a spiral as the motion pattern. For the use case of having a specialized brush tool endowed with a rotational motion, — which will be the case in the following subsection — it is reasonable to propose a raster scan motion instead, for covering the surface, which we also tested.

Figure 3.10 shows the position and force measurements corresponding to the spiral motion. Figures 3.11 and 3.12 show the position and force measurements corresponding to the raster scan motions with turning diameters of 3 and 6cm, respectively. Each figure is composed of three sub-figures with: (a) the projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_2$ plane and the two dimensional reference path; (b) the three dimensional end-effector trajectory on top of the model of the surface; and (c) the unfiltered 3rd component of the measured force, which should correspond to the force perpendicular to the surface, when the wiping tool is perfectly aligned.

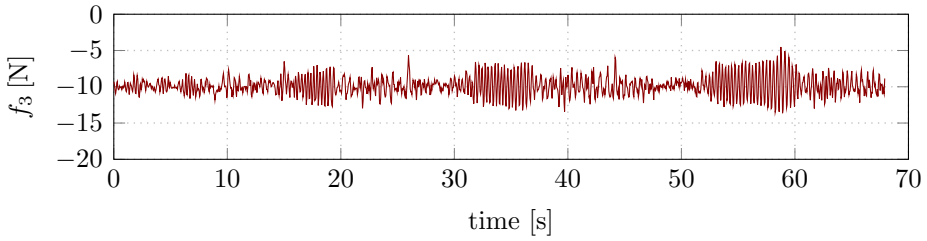
Note that the reference raster scans shown in Figure 3.11 and 3.12 have the parallel lines connected by semi-circles instead of straight lines as in the example of Figure 3.5. The requirement of using a smooth curve as a motion pattern comes from the chosen parametric representation for trajectory tracking, where we use the derivative of this curve to indicate the direction where the end-effector should move. Therefore, this



(a) Projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_2$ plane (solid line) and two dimensional reference path (dashed line).

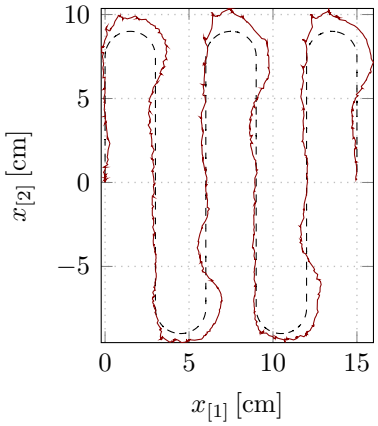


(b) Three dimensional end-effector trajectory overlaid on the surface.

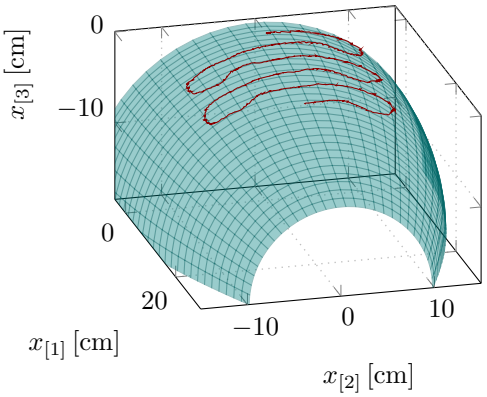


(c) Measurement of 3rd force component. When wiping tool is perfectly aligned with the surface, this component corresponds to the force perpendicular to the surface.

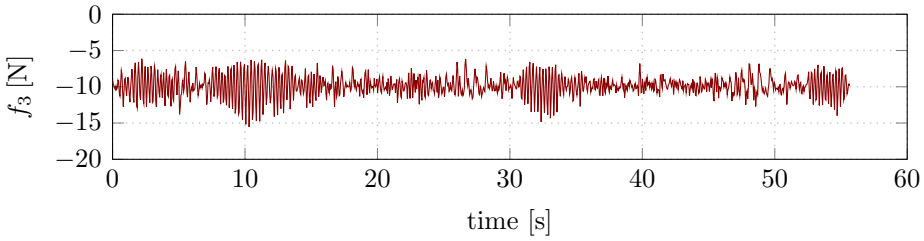
Figure 3.10: Position and force measurements corresponding to the Baxter experiment and the spiral motion, for a tangential velocity of $C_v = 0.02\,\mathrm{m\,s^{-1}}$ and a normal force set point of $f_{d3} = -10\,\mathrm{N}$.



(a) Projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_2$ plane (solid line) and two dimensional reference path (dashed line).

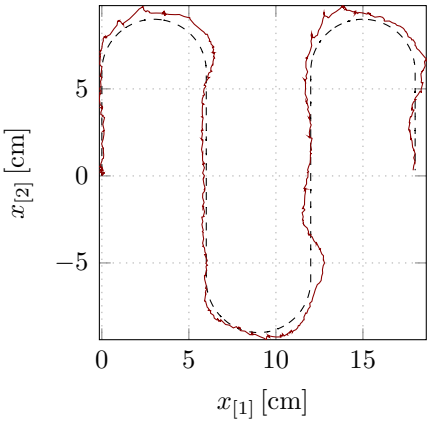


(b) Three dimensional end-effector trajectory overlaid on the surface.

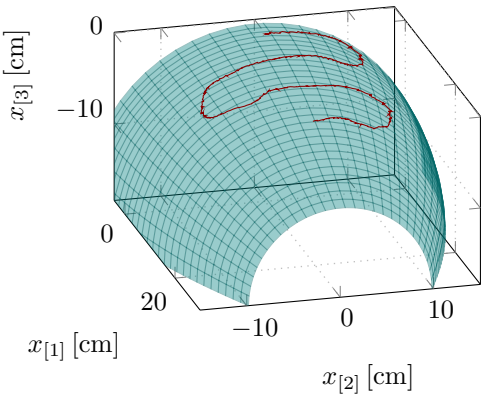


(c) Measurement of 3rd force component. When wiping tool is perfectly aligned with the surface, this component corresponds to the force perpendicular to the surface.

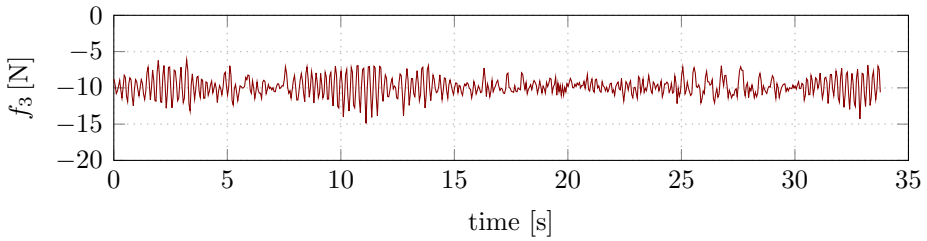
Figure 3.11: Position and force measurements corresponding to the Baxter experiment and the raster scan motion with turning diameter of 3 cm, for a tangential velocity of $C_v = 0.02\,\mathrm{m\,s^{-1}}$ and a normal force set point of $f_{d3} = -10\,\mathrm{N}$.



(a) Projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_2$ plane (solid line) and two dimensional reference path (dashed line).



(b) Three dimensional end-effector trajectory overlaid on the surface.



(c) Measurement of 3rd force component. When wiping tool is perfectly aligned with the surface, this component corresponds to the force perpendicular to the surface.

Figure 3.12: Position and force measurements corresponding to the Baxter experiment and the raster scan motion with turning diameter of 6 cm, for a tangential velocity of $C_v = 0.02\,\mathrm{m\,s^{-1}}$ and a normal force set point of $f_{d3} = -10\,\mathrm{N}$.

derivative has to be well defined in all points of the trajectory. Furthermore, we are currently using gradient descent for computing (3.21), which again requires access to the derivative of the reference curve.

To achieve smooth motions we experimentally (through trial-and-error) obtained the following control gains: $K_z = -0.03\,\mathrm{m\,s^{-1}\,N^{-1}}$; $K_{xy} = -7.0\,\mathrm{rad\,N^{-1}\,s^{-2}}$; $K_D = 1.0$; and $K_P = 100.0\,\mathrm{Hz}$, for the Baxter robot and the cleaning setup in Figure 3.8a, where K_z and K_{xy} are the proportional gains for the admitance control in (3.12) and K_P and K_D are the proportional and derivative gains of the trajectory tracking control in (3.23). We also selected the set points $C_v = 0.02\,\mathrm{m\,s^{-1}}$ and $f_{d3} = -10\,\mathrm{N}$ to give a visually smooth motion. For example, the motion's smoothness would degrade when using too high or too low contact forces. This happens because the orientation mechanism depends on measuring the tangential torque so to adapt the end-effector orientation to the surface's normal. If the force applied to the surface is small, then the reading of the tangential torque measurements will be small, becoming at some point indistinct from the signal noise. Therefore, the end-effector needs to apply sufficient pressure so that the torque resulting from the misalignment is greater than the noise. However, if the force applied to the surface is too large, then the motion becomes too sticky due to the neglected effects of friction. For instance, for a f_{d3} of $-5\,\mathrm{N}$ the robot would misalign and start losing contact with the surface and for a f_{d3} of $-15\,\mathrm{N}$ the wiping motion would become quite sticky and unstable. Note that the determination of the set of parameters that would lead to a better cleaning of the surface was out of the scope of this work.

Despite being unclear from the three dimensional plots, as the end-effector transverses throughout the surface, the surface inclination changes significantly in relatively short distances – which is having variations of up to 90° from the left to the right sides of the surface in the $\tilde{e}_2$ direction. Therefore, the robot has to cope with large variations of its end-effector orientation for being able to correctly align itself with the surface. When unable to perfectly align the end-effector with the surface normal we observe that: we are no longer controlling the force normal to the surface (Figure 3.4 shows the tool slightly misaligned to help visualize this effect); and the distance L_f between the surface and the force control sensor changes, becoming different from the constant value we use in the transformation computations. As a result, the velocity commands are no longer fully tangential to the surface, having a component that is either against it, increasing the friction, or away from it, making the end-effector slip out of the intended path and increasing the tracking error. Additionally, the requirement for the end-effector to be perpendicular at all times to a surface with such wide variations of inclination coupled with the large lenght of the end-effector tool highly reduces the workspace of the robot.

Furthermore, the proportional controller for maintaining the force level might be amplifying the sensor noise, given that the interface material (the sponge) introduces some unknown compliance/dynamics into the system. This could explain the increase in the standard variation of the 3rd component of the force from 0.35 N, when

Table 3.1: Mean μ and standard deviation σ of the tracking error ε and the 3rd component of the measured force f_3, for the wiping experiment with the Baxter and functional prototype.

	μ_ε [mm]	σ_ε [mm]	μ_{f_3} [N]	σ_{f_3} [N]
Baxter – spiral	2.44	1.64	−9.87	1.35
Baxter – raster scan 3 cm	3.92	3.36	−9.87	1.45
Baxter – raster scan 6 cm	3.06	2.20	−9.85	1.35
Prototype – raster scan 1.5 cm	2.79	1.84	−4.34	1.64
Prototype – raster scan 3 cm	1.78	1.78	−3.95	1.20

not in contact, to 1.35 N for the spiral motion and 1.45 N and 1.35 N for the raster scan motion with lower and larger curvature radius, respectively. A more careful tuning of the control gains as in [172] could attenuate the force variance. However, the larger force standard deviation for the motions with narrower turns and higher tracking error suggests that the performance of a controlled quantity depends both on its respective control gains and the performance of the other controlled quantities, such as the position and the tangential torques. Table 3.1 summarizes the mean μ and the standard deviation σ for both the tracking error ε and the 3rd component of the measured force f_3, for all the previous experiments and the experiments of the following subsection. The tracking error is defined here as $\varepsilon = \|{}^P x - \hat{p}\|$, i.e. the Euclidean norm of the distance between the projected sample ${}^P x$ and its corresponding closest position $\hat{p}$ on the reference trajectory.

3.5.3 Automation of Train Cab Front Cleaning

This case study is a feasibility study that looks at employing the position/force control approach discussed in this chapter to the cleaning of the trains' fronts. The current cleaning process of the trains' exterior includes mechanized washers, as shown in Figure 3.13, that successfully wash the trains' side panels but fail to clean the train cab front nose and body-end panels between carriages. Due to the train cab front being non-flat and having complex shapes, specialized automatic washers tend to be fairly large, complex, and expensive. Moreover, such machines lack some flexibility and are, in general, unsuitable for different train cab front geometries.

As a result, train cleaning operations still comprise depot workers manually washing the front train panels. The resulting health and safety concerns include: workers operating in non-ergonomic postures and subject to bad weather conditions; and exposure to a highly humid environment close to 25kV overhead lines and 750V rails. Hence, the emergence of safety regulations such as the orange "wash line" marking the limit for manual washing. Figure 3.1 shows two examples of trains

(a) Train washer for fuel powered trains. Wembley depot, London.

(b) Train washer for electric trains. Willesden depot, London.

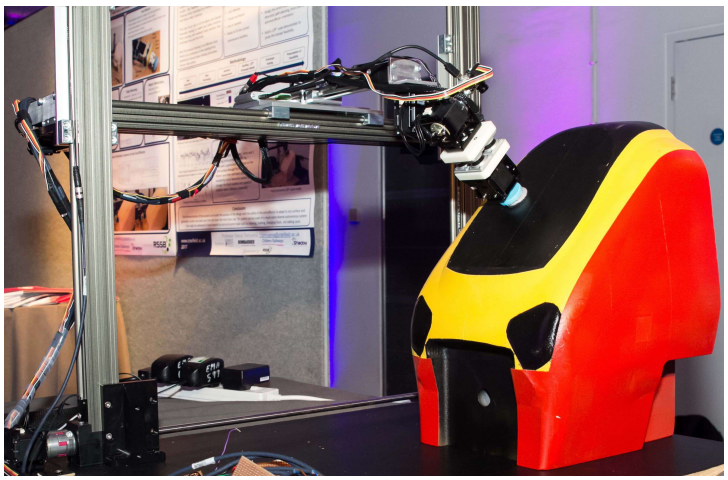
Figure 3.13: Examples of train mechanic washers from two train maintenance depots. London, 9th of June of 2016.

being manually cleaned.

We validated the same control and path tracking strategy, presented in Section 3.4 and validated with the Baxter robot in the previous case study, using a purposely designed Cab Front Cleaning Robot functional prototype. This robot is flexible to accommodate different train cab front geometries, which is a significant advantage compared to many commercially available cleaning solutions. Moreover, the detection of the surface geometry through force sensing provides a robust solution for the outdoor environments, where the cleaning process typically takes place. This prototype came to life through a partnership project between Heriot-Watt University, Cranfield University, Bombardier Transportation, Chiltern Railways, and Shadow Robot Company, entitled "Cab Front Cleaning Robot", awarded by the Railway Safety and Standards Board (RSSB), via the Rail Research UK Association (RRUKA). This project aimed at designing and building a scaled functional prototype of a robotic manipulator specifically adapted to the task of cleaning the front part of a train cab.

Figure 3.14a shows the final scaled functional robot prototype wiping the surface of a 1/8 scaled train cab front, using the strategy described in this chapter. This is a 5 DoF robot with a rotating brush as the end-effector, Dynamixel AX12 servo motors at the 5 joints, and a OPTOFORCE HEX-58-RE force/torque sensor connected to the brush motor (Figure 3.14b). The contribution from the work developed throughout this thesis to the project included: the implementation of the control and path tracking strategy for sweeping the scaled train cab front; and the implementation of both the low level controllers for the prototype servo motors as well as the communication between the low level controllers with a ROS interface.

In order to better cover the train front surface we used a raster scan motion instead of a spiral. The brush at the end-effector rotates along its 3rd local axis, performing the cleaning motion similar to the spiral movement. The rotational motion of the



(a) Scaled functional prototype wiping a scaled train cab front using a force/torque sensor to detect the surface slope.

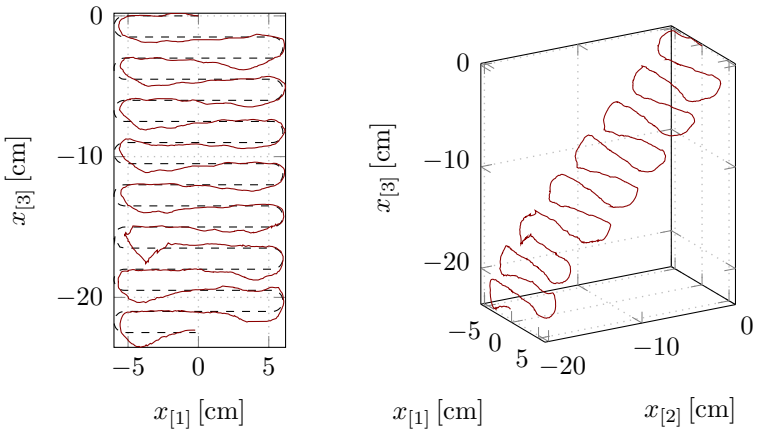
(b) Wiping brush and the force/torque sensor.

Figure 3.14: Cab Front Cleaning scaled robot prototype and detail of the end-effector used in the experiment.

brush also helped reducing the friction, leading to a more fluid motion.

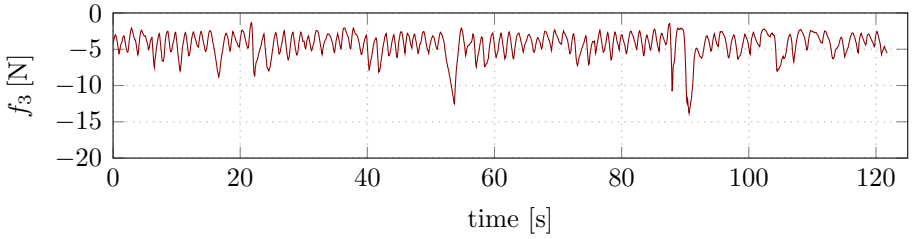
Figure 3.15 and 3.16 show the position and force measurements corresponding to the Cab Front Cleaning Robot prototype wiping the scaled train cab front using a raster scan motion pattern with diameters of 1.5 and 3 cm, respectively. As in the case of the Baxter figures, each figure is composed of three sub-figures with: (a) the projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_3$ plane and the reference two dimensional path; (b) the three dimensional end-effector trajectory; (c) and the unfiltered 3rd component of the measured force, which should correspond to the force perpendicular to the surface, for a perfectly aligned wiping tool. To achieve a smooth motion, we experimentally (through trial-and-error) obtained the following controller gains: $K_z = -0.1\,\mathrm{m\,s^{-1}\,N^{-1}}$; $K_{xy} = -3.0\,\mathrm{rad\,N^{-1}\,s^{-2}}$; $K_D = 1.0$; and $K_P = 50.0\,\mathrm{Hz}$, where K_z and K_{xy} are the proportional gains for the admitance control in (3.12) and K_P and K_D are the proportional and derivative gains of the trajectory tracking control in (3.23). To give a visually smooth motion, we selected the set points $C_v = 0.025\,\mathrm{m\,s^{-1}}$ and $f_{d3} = -4\,\mathrm{N}$.

Table 3.1 summarizes the mean μ and the standard deviation σ for both the tracking error ε and the 3rd component of the measured force f_3, for both experiments with the Baxter and the functional prototype, corresponding to the trajectories plotted in the Figures 3.10, 3.11, and 3.12 for the Baxter experiments and the Figures 3.15 and 3.16 for the prototype experiments. We can verify that for the raster scan with larger radius and the spiral, i.e., the patterns with smoother change of motion, both the tracking error mean and standard deviation and the force standard deviation are lower.



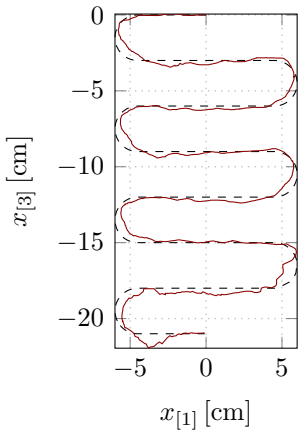
(a) Projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_3$ plane (solid line) and reference two dimensional path (dashed line).

(b) Three dimensional end-effector trajectory.

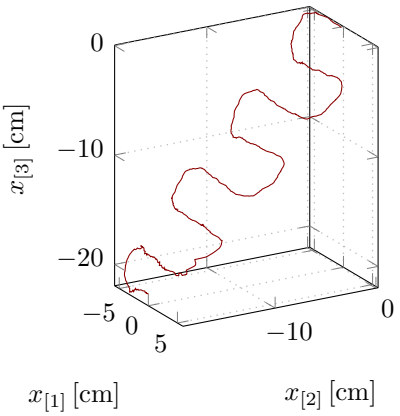


(c) Measurement of 3rd force component. When wiping tool is perfectly aligned with the surface, this component corresponds to the force perpendicular to the surface.

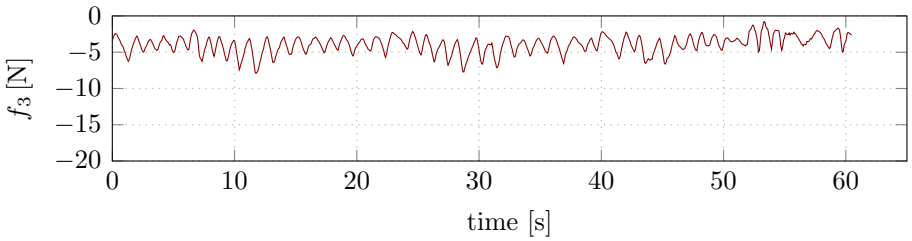
Figure 3.15: Position and force measurements corresponding to the functional prototype experiment and the raster scan motion with turning diameter of 1.5 cm, for a tangential velocity of $C_v = 0.025\,\mathrm{m\,s^{-1}}$ and a normal force set point of $f_{d3} = -4\,\mathrm{N}$.



(a) Projection of the end-effector trajectory on the $\tilde{e}_1 - \tilde{e}_3$ plane (solid line) and reference two dimensional path (dashed line).



(b) Three dimensional end-effector trajectory.



(c) Measurement of 3rd force component. When wiping tool is perfectly aligned with the surface, this component corresponds to the force perpendicular to the surface.

Figure 3.16: Position and force measurements corresponding to the functional prototype experiment and the raster scan motion with turning diameter of 3 cm, for a tangential velocity of $C_v = 0.025\,\mathrm{m\,s^{-1}}$ and a normal force set point of $f_{d3} = -4\,\mathrm{N}$.

3.6 Discussion

This chapter contains two key contributions: the first is the discovery of the equivalence between three main approaches for the simultaneous motion and force control in the operational space for torque commanded (dynamic) robotic manipulators; and the second is the experimental validation of the simultaneous motion and force controller based on the selection matrices approach, in a simpler velocity commanded (kinematic) robotic manipulator setting for a relevant industrial application — train cab front cleaning operation. Regarding the first contribution, we showed that we can derive the operational space controller with constraints, proposed by Mistry and Righetti [101], by stacking the constraint and the robot Jacobian matrices and then substituting that expanded Jacobian in the task-space dynamic equations, approach proposed by De Sapio and Khatib [34]. Moreover, we showed that the selection matrices approach (or originally denominated as Generalized Task Specification Matrices) for the simultaneous motion and force control in the operational space, proposed by Khatib [77], also corresponds to the approach of stacking the Jacobian matrices for each of the force and motion-tasks and then pass it through the task-space dynamics. In essence, these equivalence becomes obvious when we consider that for the constraint directions we perform force control and for the unconstrained directions we perform motion/position control. This last equivalence should readily apply to the simpler kinematic controlled robots case.

The choice of using the selection matrices method for the two case studies presented, rather than using a stack of task/constraint Jacobian matrices, was purely because these experiments preceded the discovery of the equivalence between these two approaches. The approach of using selection matrices remotes back to Raibert and Craig [135]'s work, which means it is a much older and widely used approach. It is also quite intuitive to think in this way of selecting specific spatial directions for force or position control. These first case studies were useful to trigger our interest, leading to the research of the different methods for simultaneous position and force control of robotic manipulators.

The major limitations of the contributions from this chapter are: the lack of experiments for simultaneous position/force control with a torque controlled robotic arm, and subsequent comparison of the results with the ones obtained from the velocity controlled arms; and the need for further comparisons between the three control approaches for simultaneous motion and force control — despite being analytically equivalent, these methods might perform differently from a numerical point of view, i.e speed of computation and numerical stability. The benefits of torque controlled versus velocity/position controlled robots is still an ongoing debate on the robotics community, and most probably an application dependent problem. On one hand, torque controlled robots can provide compliance, which is an important property when dealing with contacts and interactions, but on the other hand, they

require good models of the interaction and their own dynamics, which are often inaccurate [110] or inaccessible. Many industrial robots simply lack torque control capabilities, which requires specialized joint motors. Another consideration is the need for real-time computations, where torque control loops commonly run at much higher sampling frequency compared to velocity control loops [138]. Regarding the numerical comparison, even though this is a worth pursuing experiment, in our experience, for the typical dimensionality of single arm manipulators (6-7DoF), the numerical stability and computational speed differences between different controllers tends to be unimportant, when factoring the current computational power and low machine errors. Finally, this work only addresses the simultaneous position and force control of a robotic manipulator once it is already in contact with a curved surface. The making and breaking of contacts is in itself another challenging problem in the robotics domain, both from a control and planning perspective [133, 139, 140], which is out of the scope of this work. In our robotic experiments, we evaded this complex problem by simply implementing a contact phase where the robot slowly approaches the surface, avoiding any force spike due to impact. Once the force sensor would sense a contact force, the robot would initialize the motion along the surface.

Regarding the case study of wiping/cleaning the train's exterior surfaces, we validated the feasibility of the application of our simultaneous position and force control strategy to this type of industrial tasks. Given that train cleaning solutions need to handle a wide heterogeneity of train shapes, one of our questions was whether the use of force sensing could aid an autonomous cleaning system to be more flexible and adaptable to unknown or uncertain surface geometries. We addressed that question by considering the extreme case scenario where we have access to minimal information about the shape — only assuming that the surface is smooth and has known boundaries — and can only rely on the measurement of the interaction force. As we can see from the results, the robots successfully adapt their wiping tool orientation while traversing the curved surfaces, albeit at a relatively slow pace. Therefore, we believe that any viable/commercial industrial solution for the autonomous cleaning of curved surfaces will benefit from the integration of interaction force information with other modes of perception, such as vision, for improving their overall operation speed and reliability. The evaluation of criteria for cleaning effectiveness, such as time or cleanliness, and subsequent comparison with the current human manual cleaning process were out of the scope of this work. Finally, the "Cab Front Cleaning Robot" project, which motivated the study of our control approach, culminated in the development of a scaled functional prototype which we used to verify our technique and later led to a followup Innovate UK funded project — consisting of a Knowledge Transfer Partnership (KTP) between Cranfield University and Garrandale Ltd, a UK based company which designs and manufactures various systems for the UK railway industry – with the goal of designing and developing a full-scale prototype.

Chapter 4

Learning Generalizable Constrained Policies by Demonstration

"An idea is always a generalization, and generalization is a property of thinking. To generalize means to think."

Georg Wilhelm Friedrich Hegel

This chapter introduces and reviews a family of methods for learning control policies from constrained demonstrations. It presents a novel Constraint-aware Policy Learning (CaPL) method for learning underlying/unconstrained policies that generalize across different constraints/tasks. The method consists of a two step process that includes estimating/learning the task-constraint and using that estimation for learning the unconstrained policy. The chapter contains two simulation case studies for comparing policy learning methods and a case study with real hardware, using the KUKA LWR 3, for the validation of the CaPL generalization capabilities. It also goes through a discussion on constraint similarity analysis and theoretical analysis of closed-form solutions for estimation of different task-based constraints.

4.1 Introduction

An important function of the human upper body is to perform various force interactive tasks when moving in contact with the external environment [116, 158]. Examples of common everyday tasks constrained by the physical environment are opening a door, pulling out a drawer and turning a steering wheel [65, 116]. It seems plausible to assume that humans exploit the constrained nature of those tasks and the redundancy of their joint angles and muscle forces, when adopting fundamentally different control strategies for physically constrained and unconstrained motions [116, 158]. However, when replicating such control strategies with robotic manipulators, the traditional dynamic modelling and control approaches, covered by

the previous chapters, might fail to fully capture some nuanced complexities of the motions that humans employ.

A growing number of researchers in the field of robotics seem to believe that the answer for mimicking such complex behaviours lies in adopting some form of learning techniques, i.e capturing those motions from data rather than implementing purposely designed controllers. Indeed, robot learning research has become quite popular, with countless works published in all main robotics' venues and more recently even with dedicated conferences to the topic. A term that encapsulates this idea of mimicking complex behaviors through observation is Robot Programming by Demonstration (PbD), a topic of robot learning that aims at researching and implementing user-friendly training interfaces for teaching robots to perform specific tasks by non-specialized humans, automating the manual programming of robots [18, 146]. Other terms widely used in the literature are Robot Learning from Demonstration (LfD) [17, 147] and Imitation Learning [18, 146].

The engineering-oriented or computational research approaches to Imitation Learning typically focus on the development of appropriate representations — what are appropriate control policy functions? (i.e. *what to imitate?*) — and respective learning algorithms — how to learn the parameters of a chosen control policy? (i.e. *how to imitate?*) [18, 146]. Questions such as *when to imitate?* and *whom to imitate?* remain largely unexplored [18]. Given observations of the states and actions of a demonstrator and given a control policy function, the aim of PbD is to directly learn the policy parameters, which encode the demonstrated behaviour or trajectory, using some supervised learning method [19, 146]. However, the problem with a naive Direct Policy Learning (DPL) method is that slightly different conditions from the ones used during demonstration, e.g. different environment and/or positioning of the robot, can result in a faulty repetition/replaying of the learnt policy.

Many works attempt to address this issue of lack of robustness and generalization by exploring various policy encodings — essentially addressing the *what to imitate?* question [18]. One prominent example are the Dynamic Movement Primitives (DMPs) [70, 71, 149], which is a line of research that proposes policies combining a nonlinear component, learnt from demonstration, with a point or a limit cycle attractor. The nonlinear component is responsible for reproducing the demonstrator behaviour whereas the attractor component ensures the stability and generalization of the policy. The DMP ability of generalizing to different trajectory goals, scales and speeds, is particularly useful for reproducing learnt trajectories whilst avoiding obstacles [70, 71]. Other examples of methods for learning generalizable and/or robust policies from demonstration are, for instance, Probabilistic Movement Primitives (ProMPs) [120, 121] and Learning Dynamical Systems with Gaussian Mixture Models (GMMs) [76].

Despite addressing some of the challenges regarding the lack of generalization, DPL methods fundamentally lack any explicit awareness of the notion of constrained mo-

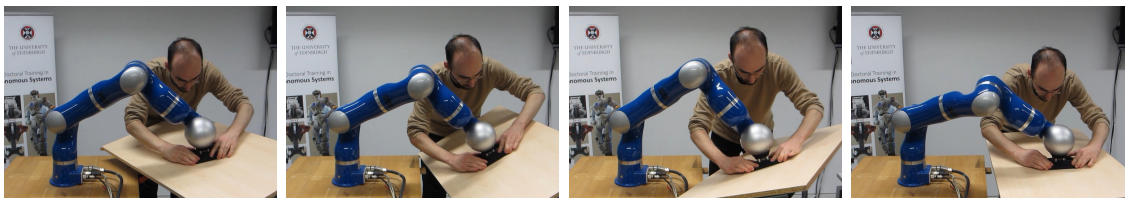
(a) (b) (c)

Figure 4.1: Manual cleaning/wiping of an electric train, illustrating in a concrete real application scenario the ability of humans to: perform constrained manipulation motions in contact; adapt the same underlying wiping motion to different parts of a complex shaped surface. Willesden depot, London (2016).

tions. Generally speaking, DPL methods either learn a free/unconstrained motion trajectory or implicitly integrate the constraints in the learnt trajectory, i.e. the constraint is invariant across the demonstrations and tests [63]. For example, by changing the point of attraction or scaling of a DMP, one might be able to avoid an obstacle, such as a table surface, absent in the demonstrations. However, what if that trajectory is meant to be in contact with the table surface, and the table geometry or positioning changes during the replay of the learnt motion. In that case, the table constraints the whole motion and, therefore, the control policy needs somehow to encode the interference of the table in order to adapt the full motion to variations of the table positioning. An example of a concrete real application scenario is the manual cleaning/wiping of the front panel of a train. Figure 4.1 shows an operator successfully adapting a given wiping motion — what we call the underlying motion — to various parts of the train surface with different geometries.

More than being able to adapt a learnt motion to varying constraints, we should be able to handle learning those motions under varying constraints. Let's consider the case where the demonstrations themselves contain motions subject to different environment constraints. Figure 4.2 exemplifies the process of teaching a robot to execute circular wiping motions on a table, for different table positions and orientations. In this case, the challenge is broader than *how to adapt or generalize the learnt control policy?* and goes back to *how to imitate?* when there is variability of the data, resulting from several demonstrations subject to different constraints. Howard [63] devoted his thesis to the problem of learning control policies from constrained motions, focusing on developing learning methods that are able to capture an underlying control policy — by assumption invariant across demonstrations — subject to a specific class of constraints on the motion — by assumption variant across demonstrations [67].

The goal of this chapter is to address the challenge of exploiting the constrained



(a) (b) (c) (d)

Figure 4.2: Example of using Programming by Demonstration for teaching a robot to execute circular wiping motions on a table, for a variety of different table positions and inclinations.

nature of certain tasks for learning generalizable control policies, by reviewing and expanding on the work of Howard [63]. The task-based constraint constitutes a useful abstraction for achieving generalization of underlying motions across different tasks and constraints, by splitting the space of control actions into task-space and null-space components.

4.2 Background

The goal of this section is to review the main research literature on learning control policies from constrained motion. This section introduces the nomenclature and the key concepts for the exploration of learning constraint-aware policies by demonstration, covering from Direct Policy Learning to learning constraint-consistent and null-space policies.

4.2.1 Direct Policy Learning

Schaal et al. [146] formalize the problem of motor control as finding a task-specific control policy function $\boldsymbol{\pi}(\cdot)$, generically written as

$$u(t) \triangleq \boldsymbol{\pi}(s(t), t; \beta), \tag{4.1}$$

which selects appropriate motor commands or actions $u(\cdot)$ for all the actuators of a moving system as a function of its internal and the environment state, encapsulated by $s(\cdot)$, and for a given time t. Equation (4.1) represents a non-autonomous policy, i.e. explicitly dependent on t. From now on, let us only consider the subset of autonomous policies

$$u(t) \triangleq \boldsymbol{\pi}(s(t); \beta), \tag{4.2}$$

i.e. policies explicitly independent from t. Finally, β represents the control policy's set of open parameters. In the context of this chapter, the terms policy and control policy are interchangeable.

There are two key elements to the problem of motor control: choosing an appropriate control policy function $\boldsymbol{\pi}(\cdot)$ — policy selection [32]; and finding suitable values for the set of parameters β — parameter estimation. Generally speaking, Programming by Demonstration refers to the problem of parameter estimation, i.e. finding $\hat{\beta}$ given a policy function $\boldsymbol{\pi}(\cdot)$. Therefore, unless stated otherwise, writing the learnt policy as $\hat{\boldsymbol{\pi}}(\cdot)$ essentially means $\boldsymbol{\pi}(\cdot;\hat{\beta})$.

Crucial to Imitation Learning is the existence of an evaluation criterion, which quantifies how similar are the demonstrated and learnt trajectories/behaviours. Defining ε such that it captures the task goal is, in general, a nontrivial problem. Even in biologically inspired works, discovering what are the metrics learners use when imitating remains a challenge [146]. In computational approaches to Imitation Learning, this evaluation criterion will typically be an error metric $\varepsilon(\cdot)$, function of the policy parameters β, defined for a given policy function $\boldsymbol{\pi}(\cdot)$, state and action trajectories $s(t)$ and $u(t)$, respectively, with $t \in [0, T] \subset \mathbb{R}$. When obvious from the context, $\varepsilon(w)$ will replace the more extended version $\varepsilon(w; \boldsymbol{\pi}, s, u)$.

Imitation by Direct Policy Learning then simply consists in learning an appropriate control policy directly by supervised learning, i.e. assuming observable and identifiable u and s, the problem consists in finding $\beta \in \mathcal{B}$ that minimizes a given cost ε, or more formally

$$\hat{\beta} \in \arg\min_{\beta \in \mathcal{B}} \varepsilon(\beta), \tag{4.3}$$

As evaluation criteria, it is quite common to use a squared error of the learnt actions — i.e the square of the Euclidean distance in the actions space — over the time horizon of the observations [146]. Let's then define the normalized Direct Policy Error (nDPE) of an estimated policy $\hat{\boldsymbol{\pi}}$ as

$$\varepsilon_{\text{nDPE}}(\hat{\boldsymbol{\pi}}; s, u) \triangleq \frac{1}{T} \int_0^T \left\| u(t) - \hat{\boldsymbol{\pi}}(s(t)) \right\|^2 dt, \tag{4.4}$$

using the *given* $\hat{\boldsymbol{\pi}}$ notation to compactly represent *given* $\boldsymbol{\pi}(.)$ *and* $\hat{\beta}$.

In general, we lack access to the state and action trajectories, $s(\cdot)$ and $u(\cdot)$, and can only observe data samples. Let's represent a sequence of $\mathcal{K}$ states as the set $\{s_k\}_{k=1}^{\mathcal{K}} = \{s_1, s_2, \ldots, s_{\mathcal{K}}\}$ and a sequence of $\mathcal{K}$ controls or actions as the set $\{u_k\}_{k=1}^{\mathcal{K}} = \{u_1, u_2, \ldots, u_{\mathcal{K}}\}$, or in a short notation as $\{s_k\}$ and $\{u_k\}$, respectively. The sequence $\{s_k\}$ corresponds to the application $s_k = s(t_k)$ for all time instances t_k with $k \in [1, \mathcal{K}] \subset \mathbb{Z}$, such that $0 \leq t_1 < \cdots < t_k < \cdots < t_{\mathcal{K}} \leq T$, and likewise for the sequence of controls $\{u_k\}$. A dataset for training policies contains all pairs of observed states and actions

$$\mathcal{X} = \{(s_k, u_k)\}_{k=1}^{\mathcal{K}} = \{(s_1, u_1), (s_2, u_2), \ldots, (s_{\mathcal{K}}, u_{\mathcal{K}})\}. \tag{4.5}$$

We then redefine the normalized Direct Policy Error for the sampled data case as

$$\overline{\varepsilon}_{\text{nDPE}}(\hat{\boldsymbol{\pi}}; \mathcal{X}) \triangleq \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \|u_k - \hat{\boldsymbol{\pi}}(s_k)\|^2, \tag{4.6}$$

by approximating (4.4) through a Riemann sum with a constant sampling interval Δt, such that $T = \mathcal{K}\Delta t$. Now, depending on the choice of $\boldsymbol{\pi}$, we can devise different methods for minimizing (4.6).

4.2.2 Receptive Field Weighted Regression

A possible model for a global policy $\boldsymbol{\pi}$ is to use a weighted combination of M local models, such that

$$\boldsymbol{\pi}(s) = \frac{\sum_{m=1}^{M} \omega_m(s)\boldsymbol{\pi}_m(s)}{\sum_{m=1}^{M} \omega_m(s)} = \sum_{m=1}^{M} \overline{\omega}_m(s)\boldsymbol{\pi}_m(s), \tag{4.7}$$

where

$$\overline{\omega}_m(s) = \frac{\omega_m(s)}{\omega(s)} \tag{4.8}$$

are the importance weightings of each model, with $\omega(s) = \sum_{m=1}^{M} \omega_m(s)$ and $\omega_m(s) = \mathrm{e}^{-\frac{1}{2}(s-c_m)^\top \Sigma_m (s-c_m)}$ being a receptive field, with center c_m, and a positive definite variance matrix Σ_m, defining the location and shape of the receptive field [10, 148]. Applying or learning such a model will usually include some either manual or automatic process for choosing/finding the centers and variances of the receptive fields which, as we shall see, is typically a separate process from learning the local models $\boldsymbol{\pi}_m$ themselves.

Local Least Squares

One of the properties of the weighted combination of local models is the partition of a large set of model parameters β into M subsets of local model parameters β_m, such that $\beta = [\beta_1; \ldots; \beta_M]$. This partition allows finding an upper bound to the error metric of the global model (4.6) based on the following summation of error metrics of each individual local model

$$\overline{\varepsilon}_{\text{nDPE}}(\beta) \leq \sum_{m=1}^{M} \overline{\varepsilon}_{\text{nDPE}_m}(\beta_m), \tag{4.9}$$

with

$$\overline{\varepsilon}_{\text{nDPE}_m}(\beta_m) = \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \overline{\omega}_m^2(s_k) \|u_k - \boldsymbol{\pi}_m(s_k; \beta_m)\|^2. \tag{4.10}$$

The proof of the result (4.9) is in Appendix B.1. We can use this property to simplify the learning of the global policy by minimizing the upper bound (4.9) instead of (4.6), which translates into performing M simpler minimizations rather than a single complex one.

By choosing local models $\boldsymbol{\pi}_m$ that are linear on a set of given feature functions $\psi_{\boldsymbol{\pi}}$ as

$$\boldsymbol{\pi}_m(s) = \psi_{\boldsymbol{\pi}}(s)\beta_m, \tag{4.11}$$

the local error metrics become

$$\overline{\varepsilon}_{\mathrm{nDPE}_m}(\beta_m) = \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \overline{\omega}_m^2(s_k) \|u_k - \psi_{\boldsymbol{\pi}}(s_k)\beta_m\|^2, \tag{4.12}$$

which we can minimize via Weighted Least Squares (WLS), as

$$\hat{\beta}_m = \arg\min_{\beta_m} \overline{\varepsilon}_{\mathrm{nDPE}_m}(\beta_m) = \left(\Psi_{\boldsymbol{\pi}}^\top \mathcal{W}_m \Psi_{\boldsymbol{\pi}}\right)^{-1} (\mathcal{W}_m \Psi_{\boldsymbol{\pi}})^\top \mathcal{U}_\parallel, \tag{4.13}$$

with

$$\mathcal{U}_\parallel = \begin{bmatrix} u_1 \\ \vdots \\ u_{\mathcal{K}} \end{bmatrix}, \quad \Psi_{\boldsymbol{\pi}} = \begin{bmatrix} \psi_{\boldsymbol{\pi}}(s_1) \\ \vdots \\ \psi_{\boldsymbol{\pi}}(s_{\mathcal{K}}) \end{bmatrix} \quad \text{and} \quad \mathcal{W}_m = \mathrm{diag}\left(\overline{\omega}_m^2(s_1) \otimes I_{n_u}, \ldots, \overline{\omega}_m^2(s_{\mathcal{K}}) \otimes I_{n_u}\right),$$

where $\otimes$ denotes the Kronecker product operator, I_{n_u} an $n_u \times n_u$ identity matrix, and n_u the dimensionality of the space of controls $u_k \in \mathbb{R}^{n_u}$ [10]. To avoid storing large matrices such as $\mathcal{W}_m$, note that

$$\mathcal{W}_m \Psi_{\boldsymbol{\pi}} = \begin{bmatrix} \overline{\omega}_m^2(s_1)\psi_{\boldsymbol{\pi}}(s_1) \\ \vdots \\ \overline{\omega}_m^2(s_{\mathcal{K}})\psi_{\boldsymbol{\pi}}(s_{\mathcal{K}}) \end{bmatrix}. \tag{4.14}$$

We will refer to the estimation of the parameters β_m via (4.13) as local Least Squares (LS), because it performs a Least Squares operation for each local model.

Global Least Squares

The choice of local linear models (4.11) also allows for a closed-form solution of the optimal parameters

$$\hat{\beta} = \arg\min_{\beta} \overline{\varepsilon}_{\mathrm{nDPE}}(\beta) = \Psi_{\mathcal{W}\boldsymbol{\pi}}^\dagger \mathcal{U}_\parallel, \tag{4.15}$$

with

$$\Psi_{\mathcal{W}\boldsymbol{\pi}} = \begin{bmatrix} \mathcal{W}^\top(s_1) \otimes \psi_{\boldsymbol{\pi}}(s_1) \\ \vdots \\ \mathcal{W}^\top(s_{\mathcal{K}}) \otimes \psi_{\boldsymbol{\pi}}(s_{\mathcal{K}}) \end{bmatrix}, \tag{4.16}$$

$$\mathcal{U}_{\|} = \begin{bmatrix} u_1 \\ \vdots \\ u_{\mathcal{K}} \end{bmatrix} \quad \text{and} \quad \mathcal{W}(s) = \begin{bmatrix} \overline{\omega}_1(s) \\ \vdots \\ \overline{\omega}_M(s) \end{bmatrix},$$

where † denotes the pseudo-inverse of a matrix, that directly minimizes the error metric (4.6), rather than its upper bound (4.12). The proof of the result (4.15) is in Appendix B.2. We will refer to the estimation of the total set of parameters β via (4.15) as global Least Squares (LS), because it performs a single Least Squares operation for all local models. Note that we can efficiently implement the series of Kronecker products in (4.16), by using a single KhatriRao product [80, 93], which is part of the MatLab® tensor toolbox library.

4.2.3 Modelling Constraints

The central assumption of Howard's [63] and this chapter's line of work for learning constrained motions is that the controls u are subject to an equality constraint written in the form

$$A(\cdot)u = b(\cdot), \tag{4.17}$$

where $A(\cdot) \in \mathbb{R}^{n_{\text{ts}} \times n_u}$ is a constraint matrix and $b(\cdot) \in \mathbb{R}^{n_{\text{ts}}}$ is a task control/policy vector, and they might be constant or functions of time t and/or state s.

The solution to (4.17) that enables the decomposition of the controls u into two orthogonal components, while simultaneously minimizing $\|Au - b\|^2$, is

$$u = \boldsymbol{\pi}(\cdot) \triangleq \underbrace{A^{\dagger} b}_{{}^{\text{ts}}u} + \underbrace{P\, {}^{\text{u}}\boldsymbol{\pi}(\cdot)}_{{}^{\text{ns}}u}, \tag{4.18}$$

where $P = I_{n_u} - A^{\dagger}A$ is an orthogonal projection matrix and $A^{\dagger}$ is the Moore-Penrose generalized inverse of A. The solution (4.18) conveniently decomposes the control u into a task-space component ${}^{\text{ts}}u$, which is responsible for a given task motion, and a null-space component ${}^{\text{ns}}u$, which is by definition neutral to the task space. Consider, for instance, a velocity controlled redundant robotic manipulator, where $u = \dot{q}$ would be the commanded joint velocities, $b = \dot{x}$ the desired end-effector velocity and A the Jacobian of the manipulator. For this kinematic control example, the task space component could be tracking a desired trajectory with the end-effector, while the null space component could be driving the manipulator's joint angles towards a comfortable configuration [159].

A simple way of embedding the constraint assumption from (4.17) is to assume that all our policies of interest are constrained policies, that follow the structure of (4.18). Howard et al. [65, 69] introduce this structure for learning constrained motions for the case where $b = 0$ and, later on, Towell et al. [159] consider the case

for which $b \neq 0$. They also unravel the notion of unconstrained policy ${}^{\mathrm{u}}\boldsymbol{\pi}$, which outputs unconstrained control actions

$$ {}^{\mathrm{u}}u(t) = {}^{\mathrm{u}}\boldsymbol{\pi}(s(t); \beta_u), \tag{4.19} $$

that when multiplied by the projection matrix P originate the null-space component of the control actions

$$ {}^{\mathrm{ns}}u = P\, {}^{\mathrm{u}}u. \tag{4.20} $$

This unconstrained policy terminology naturally raises the question of what does *to be unconstrained* exactly mean? In fact, its definition (4.19) is identical to the one we used for a generic autonomous control policy (4.2). In this work, a constrained policy will specifically refer to policies explicitly modelling constraints according to (4.18), whereas an unconstrained policy will refer, in general, to any other policy with no explicit encoding for constraints. From that perspective, any policy learned through DPL is an unconstrained policy, because even if the learning data includes control actions generated under a given constraint, the policy will be unaware of that, essentially acting as an unconstrained policy. From that reasoning, one approach for achieving generalizable policies is, rather than learning the constrained policies directly, learning an unconstrained policy instead and plugging it in (4.18), allowing its' application across different task-based constraints [10]. In this context, it is common to refer to this unconstrained policy/motion as the underlying policy/motion, given that that's the underlying motion intended for generalization. Towell et al. [159] and Armesto et al. [12] also often use the term null-space policy referring to ${}^{\mathrm{u}}\boldsymbol{\pi}$, given that both its application and learning intimately tie with the concept of null-space. Note, however, that the output of ${}^{\mathrm{u}}\boldsymbol{\pi}(t) \in \mathbb{R}^{n_u}$ itself lies in a space larger than the null-space $\mathcal{N}(A) \in \mathbb{R}^{n_u}$, indicating that it might be a misleading term. In this description we make the distinction between underlying or unconstrained policies ${}^{\mathrm{u}}\pi$ and null-space policies ${}^{\mathrm{ns}}\pi$.

4.2.4 Learning from Constrained Policies

Having introduced the concept of constrained motions/policies, the question we can pose now is how to learn from them? The most direct approach would be to attempt to learn them through a Direct Policy Learning method. However, this direct imitation usually generates policies unable to adapt to slightly different goals, and simply repeats the observed action patterns without any knowledge on how to adapt them to new/unseen contexts [146]. If the goal is to achieve some generalization capabilities, there needs to be an essential component that remains unchanged across various demonstrations [18]. Therefore, it is imperative to establish what that essential component is, which lead us to one of the main assumptions about learning constrained policies. A key assumption of Howard's [63] and this chapter's line of work is that

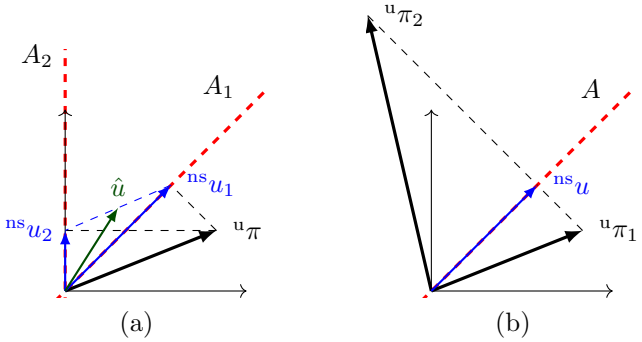


Figure 4.3: Illustration of the effects of constraints on the generation of control actions under an underlying policy $^{\text{u}}\boldsymbol{\pi}$ and the problems resulting from learning such policies through DPL methods, being: (a) averaging, the problem that the result of *averaging* actions generated under multiple different constraints A_1 and A_2 differs from the ground truth underlying policy; and (b) indetermination, the problem that multiple different policies can explain the same constrained control action.

all the training control actions result from executing the same underlying policy $^{\text{u}}\boldsymbol{\pi}$ under various different constraints and tasks, A and b.

Consider a given dataset containing samples as pairs of states and actions $\mathcal{X} = \{(s_k, u_k)\}_{k=1}^{\mathcal{K}}$. Howard et al. [67, 68, 69] generate those samples from the same underlying policy $^{\text{u}}\boldsymbol{\pi}$ subject to randomly varying constraints A — these works assume $b = 0$. This assumes that the varying constraints fully justify the variability of the data, rather than the noise associated with the underlying policy, as it is typically the case for DPL. Towell et al. [159] and Armesto et al. [10, 12], on the other hand, consider the case that a given dataset $\mathcal{X}$ contains a set of $\mathcal{J}$ sub-datasets $\{\mathcal{X}_j\}_{j=1}^{\mathcal{J}}$. Each sub-dataset contains a different task/constraint with samples of pairs of observations $\mathcal{X}_j = \{(s_{ij}, u_{ij})\}_{i=1}^{\mathcal{I}_j}$. The total number of samples is, therefore, $\mathcal{K} = \sum_{j=1}^{\mathcal{J}} \mathcal{I}_j$. If all the sub-datasets contain the same number of samples, then the total number of samples is simply $\mathcal{K} = \mathcal{I}\mathcal{J}$. This chapter follows the case of having separate sub-datasets for different tasks and constraints. However, the assumption that the same underlying policy $^{\text{u}}\boldsymbol{\pi}$ generates all samples from the dataset remains valid.

The main motivation for learning from constrained policies is to avoid the need for multiple policy models by learning an underlying policy that generalizes over constraints [65]. However, there are other motivations, such as addressing the problems of averaging and indetermination, illustrated in Figure 4.3.

Averaging and Indetermination problems with DPL

The typical assumption when learning policies through regression is that observations are noisy. For example, for the particular case of Gaussian noise, minimizing (4.6)

even corresponds to maximising the expected value of the observations [19]. But what if there is some unmodelled event, such as varying constraints as we assume in this work, that justifies the variability in the control actions better than simply added random noise. In that case, applying a DPL method, usually consisting of some regression, will result in estimates of the control actions that essentially average out the observed actions rather than finding the ground truth unconstrained policy. Consider the example illustrated in Figure 4.3a where two different constraints A_1 and A_2 lead to the observed actions ${}^{\text{ns}}u_1$ and ${}^{\text{ns}}u_2$, respectively, for some state s. Assuming noise as the reason for variability in the observed actions leads to an estimated action $\hat{u}$ which averages the observations and, therefore, differs from the ground truth actions ${}^{\text{ns}}u = (I_{n_u} - A^{\dagger}A)\,{}^{\text{u}}\boldsymbol{\pi}(s)$, for the case where $b = 0$.

Figure 4.3b illustrates another issue with disregarding the notion of constraints when assuming that those influence the generation of the observations. This is the problem of indetermination, where two different policies ${}^{\text{u}}\boldsymbol{\pi}_1$ and ${}^{\text{u}}\boldsymbol{\pi}_2$, and indeed an infinite number of different policies, can justify the observed control action ${}^{\text{ns}}u$. Howard [63] refers to this problem as the *degeneracy* problem. This issue motivates the reasoning that for capturing underlying unconstrained policies, the dataset must contain a rich set of demonstrations generated under a variety of different constraints. Take the extreme case scenario where the dataset contains demonstrations only exposing a single constraint. In that case, it is impossible to fully recover an underlying policy, meaning that the learnt policy implicitly encodes the constraints present during the demonstrations being, therefore, unable to generalize to other contexts, as it is our goal.

4.2.5 Metrics for Evaluating Performance

A crucial concept in imitation learning is the determination of appropriate evaluation metrics [18]. With that in mind, Howard et al. [67] define the Unconstrained Policy Error (UPE) of the estimated underlying policy ${}^{\text{u}}\hat{\boldsymbol{\pi}}$ for a given dataset $\mathcal{X}_{\text{u}} = \{(s_k, {}^{\text{u}}u_k)\}$ as

$$\overline{\varepsilon}_{\text{UPE}}({}^{\text{u}}\hat{\boldsymbol{\pi}}; \mathcal{X}_{\text{u}}) \triangleq \sum_{k=1}^{\mathcal{K}} \| {}^{\text{u}}u_k - {}^{\text{u}}\hat{\boldsymbol{\pi}}(s_k)\|^2, \tag{4.21}$$

and the Constrained Policy Error (CPE) of the estimated underlying policy ${}^{\text{u}}\hat{\boldsymbol{\pi}}$ for a given orthogonal projection P and a given dataset $\mathcal{X}_{\text{ns}} = \{(s_k, {}^{\text{ns}}u_k)\}$ as

$$\overline{\varepsilon}_{\text{CPE}}({}^{\text{u}}\hat{\boldsymbol{\pi}}; P, \mathcal{X}_{\text{ns}}) \triangleq \sum_{k=1}^{\mathcal{K}} \| {}^{\text{ns}}u_k - P(s_k)\,{}^{\text{u}}\hat{\boldsymbol{\pi}}(s_k)\|^2. \tag{4.22}$$

Howard et al. [66] use normalized versions of (4.21) and (4.22), by defining $\overline{\varepsilon}_{\text{n...}} = \overline{\varepsilon}_{\text{...}}/\mathcal{K}\sigma_{\pi}^2$. The following definitions of normalized Unconstrained Policy Error (nUPE)

$$\overline{\varepsilon}_{\text{nUPE}}({}^{\text{u}}\hat{\boldsymbol{\pi}}; \mathcal{X}_{\text{u}}) \triangleq \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \| {}^{\text{u}}u_k - {}^{\text{u}}\hat{\boldsymbol{\pi}}(s_k)\|^2, \tag{4.23}$$

and normalized Constrained Policy Error (nCPE)

$$\overline{\varepsilon}_{\text{nCPE}}({}^{\text{u}}\hat{\boldsymbol{\pi}}; P, \mathcal{X}_{\text{ns}}) \triangleq \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \| {}^{\text{ns}}u_k - P\,{}^{\text{u}}\hat{\boldsymbol{\pi}}(s_k)\|^2, \tag{4.24}$$

drop the normalization factor σ_{π}^2, because it is constant when both learning and comparing learned policies across different methods, assuming a given ground truth policy. Note that the definition of normalized Unconstrained Policy Error (nUPE) (4.23) is essentially identical to the definition of normalized Direct Policy Error (nDPE) (4.6). Only the input policy function and input dataset differ.

4.2.6 Learning Constraint-consistent Policies

Howard et al.'s [64] original approach for learning from constrained demonstrations assumes a potential-based unconstrained policy

$${}^{\text{u}}\boldsymbol{\pi}(s) = -\nabla_s \psi_{\text{u}}(s), \tag{4.25}$$

where ψ_{u} is some scalar potential function and ∇ is the gradient operator. Rather than learning ${}^{\text{u}}\boldsymbol{\pi}$, Howard et al. [64] learn ψ_{u} instead. By assuming that the control actions only contain a null-space component, i.e. ${}^{\text{ts}}u = 0$, and assuming kinematic policies, i.e. $u = \dot{s}$, it results that

$$u(t) = {}^{\text{ns}}u(t) = \dot{s}(t) = -P\nabla_s \psi_{\text{u}}(s(t)). \tag{4.26}$$

Assuming a set of $\mathcal{J}$ sub-datasets $\{\mathcal{X}_j\}_{j=1}^{\mathcal{J}}$, each sub-dataset $\mathcal{X}_j = \{(s_{ij})\}_{i=1}^{\mathcal{I}_j}$ representing a different trajectory subject to a different constraint A_j, Howard et al. [65, 66] learn ψ_{u} through a three step process, consisting in: first, obtaining estimates of ψ_{u} using Euler integration on (4.26), obtaining a dataset formed of the tuples $(s_{ij}, \hat{\psi}_{\text{u}_{ij}})$; second, aligning all the trajectories so that for similar states s, across different trajectories, the values of ψ_{u} are similar, obtaining a dataset formed of the tuples $(s_{ij}, \hat{\psi}_{\text{u}_{ij}} + \Delta_j)$, with Δ_j being the alignment shift for each trajectory; and finally, learning a model ψ_{u} through Locally Weighted Projection Regression (LWPR) [166], using the newly formed dataset.

Learning potential-based unconstrained policies poses quite strong assumptions which, ideally, we would like to relax. Relaxing one of those assumptions — the form of the policy itself (potential-based) — leads to the problem of how to learn generic

unconstrained policies. The difficulty is that assuming unobservable unconstrained actions ${}^{\mathrm{u}}u$ and unknown constraints P renders both metrics presented in the previous section, respectively the UPE and the CPE, unusable for learning purposes. Essentially, those metrics will only be suitable for evaluation purposes in artificial/made-up examples with either known ground truth unconstrained policy, for UPE, or known constraints, for CPE. Given that the unknown constraint P, by assumption, disables the use of CPE (4.22) for estimating ${}^{\mathrm{u}}\boldsymbol{\pi}$, Howard et al. [69] propose to approximate $P \approx \tilde{u}\tilde{u}^\top$, with $\tilde{u} = u/\|u\|$, and define an new metric, the Constraint Consistent Policy Error (CCPE) as

$$\overline{\varepsilon}_{\mathrm{CCPE}}({}^{\mathrm{u}}\hat{\boldsymbol{\pi}}; \mathcal{X}_{\mathrm{ns}}) \triangleq \sum_{k=1}^{\mathcal{K}} \| {}^{\mathrm{ns}}u_k - {}^{\mathrm{ns}}\tilde{u}_k \, {}^{\mathrm{ns}}\tilde{u}_k^\top \, {}^{\mathrm{u}}\hat{\boldsymbol{\pi}}(s_k)\|^2, \tag{4.27}$$

which is possible to minimize.

Howard et al. [67] propose the Constraint Consistent Learning (CCL) method, which consists in minimizing (4.27), hence, assuming that the projection for any given observation is explicitly unknown, in which case we can lump all the observations in a single full dataset $\mathcal{X}$, without any need for splitting them according to different trajectories. Furthermore, Howard et al. [67] establish that

$$\overline{\varepsilon}_{\mathrm{CCPE}} \leq \overline{\varepsilon}_{\mathrm{CPE}} \leq \overline{\varepsilon}_{\mathrm{UPE}} \tag{4.28}$$

and recognize that, ideally, we would like to minimize an upper bound rather than a lower bound. However, it is for the moment unclear how to derive such an upper bound given the assumptions of learning from constrained data. Nevertheless, the CCL approach is able to reconstruct an underlying unconstrained policy, by consolidating observations from different constraints. By modelling the unconstrained policy ${}^{\mathrm{u}}\boldsymbol{\pi}$ as a weighted combination of M locally linear models

$${}^{\mathrm{u}}\boldsymbol{\pi}(s) = \sum_{m=1}^{M} \overline{\omega}_m(s) \psi_{\boldsymbol{\pi}_u}(s) \beta_{u_m},$$

obtaining the set of parameters $\beta_u = [\beta_{u_1}; \ldots; \beta_{u_M}]$ that minimize (4.27) or its decoupled upper bound (based on the inequality (4.9)), is simply a matter of rewriting the feature functions in (4.15) and (4.13) solutions, respectively, as

$$\psi_{\boldsymbol{\pi}}(s, u) = \frac{uu^\top}{u^\top u} \psi_{\boldsymbol{\pi}_u}(s). \tag{4.29}$$

Later on, Howard et al. [69] argued that only minimizing the Constraint Consistent Policy Error (4.27) results in a ${}^{\mathrm{u}}\boldsymbol{\pi}$ that explains any variations in the observations exclusively as variations in the underlying constraints, instead of variations in the policy itself, resulting in poor performance for the cases where the motions are truly

unconstrained or there are similar constraints between observations. By hypothesizing that this might be a manifestation of learning a policy based on a lower bound error metric, Howard et al. [69] propose a Robust Constraint Consistent Learning (Robust CCL) approach for learning ${}^{\mathrm{u}}\boldsymbol{\pi}$ based on a two-stage optimisation of

$$\bar{\varepsilon}_{\mathrm{DPE}}({}^{\mathrm{u}}\hat{\boldsymbol{\pi}}; \mathcal{X}_{\mathrm{ns}}) + \alpha \bar{\varepsilon}_{\mathrm{CCPE}}({}^{\mathrm{u}}\hat{\boldsymbol{\pi}}; \mathcal{X}_{\mathrm{ns}}), \tag{4.30}$$

which combines two risk functionals: the DPE metric typically associated with learning unconstrained policies from unconstrained observations; and the CCPE associated with learning constraint consistent policies, where α is a weighting factor that reflects the prior belief on whether the data contains different constraints.

4.2.7 Handling Task-Space Component

A strong assumption when applying CCL is that the null-space controls ${}^{\mathrm{ns}}u$ are observable, which is true only if either the task space controls ${}^{\mathrm{ts}}u$ are null, i.e $b = 0$, or if we have a process for disambiguating the task component and the null-space component from the observed controls u. Towell et al. [159] are the first to address this question of handling control actions containing a task space component, i.e. the full problem of learning unconstrained policies with controls subject to $Au = b$. Towell et al. [159] propose a two-step approach, with a first step consisting in learning a null-space policy for each sub-dataset $\mathcal{X}_j$, assuming each sub-dataset contains a single task/constraint, and a second step consisting in learning the unconstrained policy, which by assumption is the same across all sub-datasets, using estimates of the null-space controls obtained through the previously learnt null-space policies.

By using the property that $A^{\dagger}P = 0$, we obtain the null-space component of the observed controls as ${}^{\mathrm{ns}}u = Pu$. We could then define a Null Space Policy Error (NSPE) metric as

$$\bar{\varepsilon}_{\mathrm{NSPE}}({}^{\mathrm{ns}}\boldsymbol{\pi}_j; P_j, \mathcal{X}_j) \triangleq \sum_{i=1}^{\mathcal{I}_j} \|P_j u_i - {}^{\mathrm{ns}}\boldsymbol{\pi}_j(s_i)\|^2, \tag{4.31}$$

that evaluates the null-space policy ${}^{\mathrm{ns}}\boldsymbol{\pi}_j$ for a given sub-dataset $\mathcal{X}_j = \{(s_{ij}, u_{ij})\}_{i=1}^{\mathcal{I}_j}$. However, we are unable to use directly (4.31) for learning null-space policies because, by assumption, the projection matrix P_j of each sub-dataset is unknown. Similar to [67]'s approach, Towell et al. [159] propose to approximate $P_j \approx \tilde{P}_j = {}^{\mathrm{ns}}\tilde{\boldsymbol{\pi}}_j^{\top}\,{}^{\mathrm{ns}}\tilde{\boldsymbol{\pi}}_j$, with ${}^{\mathrm{ns}}\tilde{\boldsymbol{\pi}}_j = {}^{\mathrm{ns}}\boldsymbol{\pi}_j / \|{}^{\mathrm{ns}}\boldsymbol{\pi}_j\|$, and for each sub-dataset find ${}^{\mathrm{ns}}\hat{\boldsymbol{\pi}}_j$ that minimizes $\bar{\varepsilon}_{\mathrm{NSPE}}({}^{\mathrm{ns}}\boldsymbol{\pi}_j; \tilde{P}_j, \mathcal{X}_j)$. Because $\bar{\varepsilon}_{\mathrm{NSPE}}$ becomes nonlinear on the parameters to estimate, even when using a linear policy, Towell et al. [159] use a numerical optimization solver, based on the Levenberg-Marquardt algorithm [104], to estimate the null-space policy.

After learning a null-space policy for each sub-dataset, it is possible to generate sub-datasets with the null-space component of the controls $\mathcal{X}_{j\text{ns}} = \{(s_{ij}, {}^{\text{ns}}u_{ij})\}_{i=1}^{\mathcal{I}_j}$ out of the original sub-datasets $\mathcal{X}_j$. Then lumping all sub-datasets together $\mathcal{X}_{\text{ns}} = \{\mathcal{X}_{j\text{ns}}\}$ allows using CCL for learning the unconstrained policy ${}^{\text{u}}\boldsymbol{\pi}$.

4.2.8 Learning Null-Space Projections

As discussed previously, the unknown constraint assumption renders the CPE unfit as a learning metric, being only useful for evaluation purposes. Howard et al. [67] derive a lower bound of CPE, useful for learning unconstrained policies, by approximating the projection matrix P with a 1 dimensional (1-D) projection for each observed control action. A different approach would be to estimate P, instead, and then use its estimate for minimizing CPE when learning the unconstrained policy. This approach will typically assume that, given $\mathcal{J}$ sub-datasets, each sub-dataset $\mathcal{X}_j$ only contains observations from a single constraint.

Lin et al. [89] consider the problem of learning the null-space projection for constrained motions and define

$$\overline{\varepsilon}_{\text{POE}}(\hat{A}, \{{}^{\text{ns}}u_k\}) \triangleq \sum_{k=1}^{\mathcal{K}} \| {}^{\text{ns}}u_k - \hat{P}\, {}^{\text{ns}}u_k\|^2 \tag{4.32}$$

$$= \sum_{k=1}^{\mathcal{K}} \|\hat{A}^{\dagger}\hat{A}\, {}^{\text{ns}}u_k\|^2 = \sum_{k=1}^{\mathcal{K}} {}^{\text{ns}}u_k^{\top}\hat{A}^{\dagger}\hat{A}\, {}^{\text{ns}}u_k, \tag{4.33}$$

as the Projected Observation Error (POE) metric. When referring to learning or estimating the null-space projection matrix $\hat{P}$, actually means learning the constraint matrix $\hat{A}$ and then computing the respective orthogonal projection matrix. In that case, one approach for learning the null-space projection simply consists in finding $\hat{A}$ that minimizes the POE.

Lin et al. [89] start by representing the constraint matrix as

$$A \triangleq \begin{bmatrix} A_1 \\ \vdots \\ A_{n_{\text{ts}}} \end{bmatrix}, \tag{4.34}$$

where $A_i = [A_{i[1]}, \ldots, A_{i[n_u]}] \in \mathbb{R}^{n_u}$ is a row vector corresponding to the ith constraint in the observations, such that $A_i \perp A_j$ for all $i \neq j$ in order to guarantee that any new added constraint A_i leaves the previous $i-1$ constraints unaffected. Note that $A \in \mathbb{R}^{n_{\text{ts}} \times n_u}$, where n_{ts} is the number of constraints and n_u is the dimensionality of the control space. Furthermore, Lin et al. [89] model each individual constraint A_i as unit vectors represented according to the dimension n_u as:

- $A_i = [\cos\theta_{i[1]}, \sin\theta_{i[1]}] \in \mathbb{R}^2$;
- $A_i = [\cos\theta_{i[1]}, \sin\theta_{i[1]}\cos\theta_{i[2]}, \sin\theta_{i[1]}\sin\theta_{i[2]}] \in \mathbb{R}^3$;
- $A_i = [\cos\theta_{i[1]}, \sin\theta_{i[1]}\cos\theta_{i[2]}, \sin\theta_{i[1]}\sin\theta_{i[2]}\cos\theta_{i[3]}, \sin\theta_{i[1]}\sin\theta_{i[2]}\sin\theta_{i[3]}] \in \mathbb{R}^4$;

and so forward for $n_u > 4$. Therefore, the problem of estimating A_i shifts to the problem of estimating $\theta_i = \{\theta_{i[1]}, \ldots, \theta_{i[n_u-1]}\}$, which consists in finding the set of parameters $\{\theta_i\}_{i=1}^{n_{\text{ts}}}$ that minimize (4.32). Assuming a single constraint, i.e. $n_{\text{ts}} = 1$, Lin et al. [89] simply propose to sample $\theta_{1[i]} \in [0, \pi]$ and to pick the corresponding A_i resulting in minimum POE. For multiple constraints, i.e. $n_{\text{ts}} > 1$, Lin et al. [89] propose an iterative approach consisting in: finding the first constraint $\hat{A}_1$ through sampling, as described before; finding the second and successive constraints through sampling as well, but subject to $\hat{A}_j \perp \hat{A}_i \, \forall i < j$, when choosing the minimizer. As a stopping criteria, we either know a priori the total number of constraints to estimate or stop when adding a new estimated constraint results in higher POE.

Later on, Lin et al. [90] consider the estimation of the constraint matrix A for the full case of $Au = b$, i.e. controls containing task-space component. Their approach consists in using the first step optimization proposed by Towell et al. [159] to split control actions u into their task-space component ${}^{\text{ts}}u$ and null-space component ${}^{\text{ns}}u$, by learning null-space policies. Then, Lin et al. [90] minimize a redefined POE

$$\sum_{k=1}^{K} \left(\| {}^{\text{ns}}\hat{u}_k - P \, {}^{\text{ns}}\hat{u}_k \|^2 + \| P \, {}^{\text{ts}}\hat{u}_k \|^2 \right), \tag{4.35}$$

with a nonlinear optimization solver, using the Levenberg-Marquardt algorithm [104], while keeping the same θ parameterization for the constraint matrix from [89], previously described.

Manavalan and Howard [97] point out the scalability issues of the sampling-based estimation in [89] and rewrite the POE as

$$\bar{\varepsilon}_{\text{POE}}(\hat{A}, \{{}^{\text{ns}}u_k\}) = \sum_{k=1}^{\mathcal{K}} {}^{\text{ns}}u_k^\top \hat{A}^\dagger \hat{A} \, {}^{\text{ns}}u_k = \text{Tr}\left(\mathcal{U}^\top \hat{A}^\dagger \hat{A} \mathcal{U} \right), \tag{4.36}$$

with $\mathcal{U} = [u_1, \ldots, u_{\mathcal{K}}]$ and $\text{Tr}(\cdot)$ representing the trace of a matrix. Manavalan and Howard [97] then derive the gradient of (4.36), obtaining

$$\nabla_\theta \bar{\varepsilon}_{\text{POE}} = A \mathcal{U} \mathcal{U}^\top \frac{\partial A}{\partial \theta}, \tag{4.37}$$

allowing for the use of gradient descent optimization methods for estimating the θ parameters.

As the constraint parameterization from [89] assumes that the constraints are linear in the control space, Lin et al. [90] also consider the case where constraints are nonlinear in the control space, given the linearization mapping

$$A(s) \triangleq \beta_A \psi_A(s), \tag{4.38}$$

and propose the appropriate modifications to its estimation. However, by a change of variables we could keep the exact same estimation process for the case where the constraints are linear in the control space. Let

$$y_A(s,u) = \psi_A(s)u, \tag{4.39}$$

then finding A such that $Au = 0$ turns into finding β_A such that $\beta_A y_A = 0$, which corresponds to minimizing $\bar{\varepsilon}_{\text{POE}}(\beta_A, \{{}^{\text{ns}}y_{Ak}\})$, with ${}^{\text{ns}}y_{Ak} = \psi_A(s_k)\,{}^{\text{ns}}u_k$ (where for now we ignore the task component b). In the very same year, Armesto et al. [9] proposed an identical description of the constraint matrix A as in (4.38), representing the matrix of parameters

$$\beta_A \triangleq \begin{bmatrix} \beta_{A1} \\ \vdots \\ \beta_{An_{\text{ts}}} \end{bmatrix}, \tag{4.40}$$

where β_{Ai} with $i \in [1, n_A]$ are orthonormal row vectors, i.e. β_A is a semi-orthogonal matrix, and n_A is the number of feature functions $\psi_A(s) \in \mathbb{R}^{n_A \times n_u}$. Note that the representation of β_A in (4.40) is identical to the representation of A in (4.34), which indicates that estimating A assuming a constraint linear in the control space is just a degenerate case for when ψ_A is the identity.

Armesto et al. [9] define the Constraint Space Error (CSE) as

$$\bar{\varepsilon}_{\text{CSE}}(A, \{{}^{\text{ns}}u_k\}) \triangleq \sum_{k=1}^{\mathcal{K}} \|A\,{}^{\text{ns}}u_k\|^2 = \sum_{k=1}^{\mathcal{K}} {}^{\text{ns}}u_k^\top A^\top A\,{}^{\text{ns}}u_k, \tag{4.41}$$

and use it to estimate the constraint matrix A. Essentially, Armesto et al. [9] proposes to estimate the constraint matrix A by minimizing (4.41), which is an error metric defined in the task/constraint space, rather than minimizing (4.32), which is an error metric defined in the null-space. By assuming (4.38) and using (4.39), we get

$$\bar{\varepsilon}_{\text{CSE}}(A, \{{}^{\text{ns}}u_k\}) = \bar{\varepsilon}_{\text{CSE}}(\beta_A, \{{}^{\text{ns}}y_{Ak}\}) = \sum_{k=1}^{\mathcal{K}} {}^{\text{ns}}y_{Ak}^\top \beta_A^\top \beta_A\,{}^{\text{ns}}y_{Ak} \tag{4.42}$$

$$= \beta_{\|A}^\top \underbrace{\sum_{k=1}^{\mathcal{K}} \left((I_{n_{\text{ts}}} \otimes y_{Ak}^\top)^\top (I_{n_{\text{ts}}} \otimes y_{Ak}^\top) \right)}_{\triangleq \mathcal{D}_y} \beta_{\|A}, \tag{4.43}$$

with $\beta_{\|A} = \text{vec}(\beta_A^\top)$. It turns out that through simple algebraic manipulation we can write $\mathcal{D}_y = I_{n_{\text{ts}}} \otimes \left(\mathcal{Y}_A \mathcal{Y}_A^\top\right)$, with $\mathcal{Y}_A = [y_{A1}, \ldots, y_{A\mathcal{K}}]$. Armesto et al. [9] then reformulate the estimation of the parameters of the constraint matrix as a standard quadratic optimization problem

$$\begin{aligned} \hat{\beta}_{\|A} = \arg\min_{\beta_{\|A}} \quad & \beta_{\|A} \mathcal{D}_y \beta_{\|A}^\top \\ s.t. \quad & \beta_A \beta_A^\top - I_{n_{\text{ts}}} = 0, \end{aligned} \tag{4.44}$$

with $\beta_A^\top = \text{vec}^{-1}_{n_A \times n_{\text{ts}}}(\beta_{\|A})$ being the inverse operation of $\text{vec}(\cdot)$ for a given matrix dimension $n_A \times n_{\text{ts}}$. Here, we greatly simplified the representation of the cost and equality condition in (4.44) relative to the original formulation [9] by using the Kronecker $\otimes$ and the $\text{vec}(\cdot)$ operators. The equality condition in (4.44) guarantees that the resulting β_A remains semi-orthogonal.

Ortenzi et al. [119] identified that the problem of finding A such that $Au = 0$ corresponds to the problem of finding the solution of an homogeneous system, which has an existing known solution based on Singular Value Decomposition (SVD). Applying SVD to the matrix of observed actions $\mathcal{U}$, results in

$$\mathcal{U} = U_{\mathcal{U}} S_{\mathcal{U}} V_{\mathcal{U}}^\top, \tag{4.45}$$

where the columns of $U_{\mathcal{U}}$ and $V_{\mathcal{U}}$ are the left-singular vectors and right-singular vectors of $\mathcal{U}$, respectively, and $S_{\mathcal{U}}$ is a rectangular diagonal matrix, whose elements of the diagonal contain the singular values of $\mathcal{U}$ in decreasing order. Both $U_{\mathcal{U}}$ and $V_{\mathcal{U}}$ are orthonormal matrices. For a given number of constraints n_{ts}, then the rows of A simply correspond to the last columns of $U_{\mathcal{U}}$, and for unknown number of constraints then A correspond to the last columns of $U_{\mathcal{U}}$ corresponding to the singular values smaller than a given threshold value. The next section will expand on using SVD for estimating constraints when looking at the method proposed in [10].

4.2.9 Summary

This section reviewed the main relevant literature on works addressing the problem of learning unconstrained policies from constrained observations. Table 4.1 categorizes the aforementioned works according to the type of linear constraints they handle, i.e. with or without task-space b, and the specific functions they learn. It is interesting to observe the historical evolution of the methods for learning constrained policies, from originally only being able to estimate the unconstrained policy ${}^{\text{u}}\boldsymbol{\pi}$ such that $Au = 0$, to the more recent methods also concerned with learning the constraints themselves. The last row of the Table 4.1 introduces two additional works that are the topic of the following section.

4.3 Learning Constraint-aware Policies

This section will carry on with the exploration of methods for learning unconstrained policies, that by assumption are invariant across a given set of $\mathcal{J}$ demonstrations. Like in previous works, for instance as in [159], we assume that each demonstration contains observations of motions performed under different contexts, i.e different constraints A and task policies b. We also focus here on learning those constraints, as in [89, 90], with the addition that we will learn the task policy as well. However,

Table 4.1: Categorization of publications addressing the problem of learning unconstrained policies ${}^{\mathrm{u}}\boldsymbol{\pi}$ from constrained observations u. This table categorizes the different works according to the type of constraints they handle, with or without task-space b, and the particular functions they learn, namely a potential function ψ_{u} for which ${}^{\mathrm{u}}\boldsymbol{\pi}(s) = -\nabla_s \psi_{\mathrm{u}}(s)$, the constraint matrix A and the task-space controller b.

	Handling Task-space		Learning			
	$Au = 0$	$Au = b$	$\psi_{\mathrm{u}}(\cdot)$	${}^{\mathrm{u}}\boldsymbol{\pi}(\cdot)$	$A(\cdot)$	$b(\cdot)$
Howard et al. [64, 65, 66] (2006-2008)	✓		✓			
Howard et al. [67, 68, 69] (2009)	✓			✓		
Towell et al. [159] (2010)	✓	✓		✓		
Lin et al. [89] (2015)	✓			✓	✓	
Ortenzi et al. [119] (2016) Manavalan and Howard [97] (2017)	✓				✓	
Lin et al. [90] (2017)	✓	✓			✓	
Armesto et al. [9] (2017)	✓			✓	✓	
Armesto et al. [10, 12] (2017-2018)	✓	✓		✓	✓	✓

in contrast with the previous literature as [89, 90], we aim to learn the constraints so that we can decouple the task-space from the null-space components of the observed actions u, and ultimately improve the learning of the unconstrained policy itself. Other strands of work aim at learning constraints with the goal of employing them directly in trajectory optimization methods [127]. This section will focus on two major contributions to the literature on learning constrained motions: a closed-form solution for the estimation of the constraint and control task; and the introduction of the Constraint-aware Policy Learning (CaPL) method that decomposes the learning from constrained motions into a two step process comprising first the estimation of the constraints and task controls, and second the estimation of the unconstrained policy.

4.3.1 Closed-form Constraint Estimation

We start by expressing A and b as a linear combination of some feature functions as

$$A(s) \triangleq \beta_A \psi_A(s) \tag{4.46}$$

$$b(s) \triangleq \beta_b \psi_b(s), \tag{4.47}$$

where $\beta_A \in \mathbb{R}^{n_{\text{ts}} \times n_A}$ and $\beta_b \in \mathbb{R}^{n_{\text{ts}} \times n_b}$ are constant matrices containing the parameters of the constraint and task, and $\psi_A(s) \in \mathbb{R}^{n_A \times n_u}$ and $\psi_b(s) \in \mathbb{R}^{n_b}$ are, by assumption, some known feature functions. What (4.46) and (4.47) implicitly assume is that we know in which space the constraints live, i.e. the space where they are linear. Even though, this is indeed quite a strong assumption, we could easily argue that the problem of estimating a non-linear constraint surface that explains a finite dataset is fundamentally ill-posed, given that there exists an infinite number of non-linear mappings that could explain the observed data. Therefore, in this type of problem we require some form of prior knowledge, in this case the feature functions, in addition to the data-set [20, 21].

By replacing (4.46) and (4.47) in (4.17), we get

$$A(s)u = b(s) \Leftrightarrow \beta_A \psi_A(s) u = \beta_b \psi_b(s) \Leftrightarrow \underbrace{\begin{bmatrix}\beta_A & \beta_b\end{bmatrix}}_{\triangleq \beta_{Ab}} \underbrace{\begin{bmatrix}\psi_A(s)u \\ -\psi_b(s)\end{bmatrix}}_{\triangleq y_{Ab}(s,u)} = 0, \tag{4.48}$$

resulting in a simple expression

$$\beta_{Ab} y_{Ab}(s, u) = 0, \tag{4.49}$$

similar to the one used for estimating the constraint for the case where $Au = 0$, discussed in the previous section. One of the known and simpler solutions for estimating β_{Ab} is the application of the SVD method.

The Singular Value Decomposition (SVD) Solution

A closed-form solution for the estimate of the parameters β_{Ab} in (4.49), proposed by Armesto et al. [10], is

$$\hat{\beta}_{Ab}^{\top} = \begin{bmatrix}\hat{\beta}_A & \hat{\beta}_b\end{bmatrix}^{\top} = U_{\mathcal{Y}_{Ab}} S_{0I}, \tag{4.50}$$

where $S_{0I}^{\top} = \begin{bmatrix}0 & I_{n_{\text{ts}}}\end{bmatrix}$ is a selection matrix which extracts the last n_{ts} columns of $U_{\mathcal{Y}_{Ab}}$, and where $U_{\mathcal{Y}_{Ab}}$ is the matrix of the left-singular vectors resulting from the Singular Value Decomposition (SVD)

$$U_{\mathcal{Y}_{Ab}} S_{\mathcal{Y}_{Ab}} V_{\mathcal{Y}_{Ab}}^{\top} = \mathcal{Y}_{Ab}, \tag{4.51}$$

with

$$\mathcal{Y}_{Ab} = \begin{bmatrix} y_{Ab}(s_1, u_1), \ldots, y_{Ab}(s_{\mathcal{K}}, u_{\mathcal{K}}) \end{bmatrix} \tag{4.52}$$

being the matrix containing the evaluation of y_{Ab} for a given dataset $\mathcal{X} = \{(s_k, u_k)\}_{k=1}^{\mathcal{K}}$.

It is important, however, to relate the solution in (4.50) with the underlying optimization problem it is solving. For that, we shall redefine the Constraint Space Error

(CSE), defined in (4.41), to include task controller b as

$$\overline{\varepsilon}_{\text{CSE}}(A, b; \{u_k\}). \triangleq \sum_{k=1}^{\mathcal{K}} \|Au_k - b\|^2 \tag{4.53}$$

Using (4.48), we can rewrite (4.53) as

$$\overline{\varepsilon}_{\text{CSE}}(A, b; \{u_k\}) = \overline{\varepsilon}_{\text{CSE}}(\beta_{Ab}, \{y_{Ab_k}\}) = \sum_{k=1}^{\mathcal{K}} \|\beta_{Ab} y_{Ab_k}\|^2. \tag{4.54}$$

In the Appendix B.4 we show that the solution (4.50) is the result of the optimization

$$\begin{aligned} \hat{\beta}_{Ab} = \arg\min_{\beta_{Ab}} \quad & \overline{\varepsilon}_{\text{CSE}}(\beta_{Ab}, \{y_{Ab_k}\}) \\ s.t. \quad & \beta_{Ab}\beta_{Ab}^\top = I_{n_{\text{ts}}}. \end{aligned} \tag{4.55}$$

It is interesting to note that the SVD solution (4.50) results in β_{Ab} with orthonormal rows whereas the optimization (4.44) from the previous work [9], which handles constraints of the type $Au = 0$, results in β_A with orthonormal rows. We will further explore this detail and its implications in the following subsection.

Note that, by definition, $S_{\mathcal{Y}_{\text{Ab}}}$'s diagonal contains the single values of $\mathcal{Y}_{\text{Ab}}$ in descending order and, hence, the last n_{ts} columns of $U_{\mathcal{Y}_{\text{Ab}}}$, in (4.50), are the left single vectors of $\mathcal{Y}_{\text{Ab}}$ corresponding to its lowest single values. If $\mathcal{Y}_{\text{Ab}}$ has exactly n_{ts} null single values, then the closed form solution (4.50) simply becomes

$$\hat{\beta}_{\text{Ab}}^\top = \mathcal{N}(\mathcal{Y}_{\text{Ab}}^\top), \tag{4.56}$$

where $\mathcal{N}(\mathcal{Y}_{\text{Ab}}^\top)$ returns a null space base for the matrix $\mathcal{Y}_{\text{Ab}}^\top$, i.e. the set of vectors in the domain of the linear mapping $\mathcal{Y}_{\text{Ab}}^\top$ that map to zero — $\mathcal{Y}_{\text{Ab}}^\top \hat{\beta}_{\text{Ab}}^\top = 0$. However, because in practice data contains noise and, therefore, there is an imperfect match between modelled constraint and the observations, the lowest single values of the data matrix $\mathcal{Y}_{\text{Ab}}$ will often be marginally greater than zero, rendering (4.56) unusable in practice. The insight, however, that the solution to our constraint estimation corresponds to finding the null space of the data matrix, provide us a useful method for finding the number of constraints n_{ts} automatically rather than pre specifying it. We can set the number of constraints n_{ts} to the number of single values of $\mathcal{Y}_{Ab}$ that are zero or approximately zero, given some threshold.

In Appendix B.3 we also prove that a solution to (4.55) corresponds to the transpose of the matrix of the right eigenvectors of $\mathcal{Y}_{\text{Ab}}\mathcal{Y}_{\text{Ab}}^\top$, corresponding to its lowest eigenvalues. Therefore, another way of obtaining $\hat{\beta}_{\text{Ab}}$ is through an Eigenvalue Decomposition (EVD) of $\mathcal{Y}_{\text{Ab}}\mathcal{Y}_{\text{Ab}}^\top$. However, we will prefer to use SVD given it that it is a numerically better conditioned operation than EVD [103].

4.3.2 Constrained Policy Estimation Decomposition

After presenting a method for estimating constraints given datasets of constrained control observations, in [10], we proposed a method for learning the unconstrained policy ${}^{\mathrm{u}}\boldsymbol{\pi}$ composed of a two stage process: given $\mathcal{J}$ sub-datasets $\{\mathcal{X}_j\}$, each containing observations coming from different constraints and control tasks, we 1. estimate the constraint and control task for each sub-dataset, by minimizing the CSE; and then 2. agglomerate all the observations in a single large dataset, and learn the unconstrained policy ${}^{\mathrm{u}}\boldsymbol{\pi}$ by minimizing the CPE using the estimated projection $\hat{P}$ for each sub-dataset. By modelling the unconstrained policy ${}^{\mathrm{u}}\boldsymbol{\pi}$ as a weighted combination of M locally linear models

$$ {}^{\mathrm{u}}\boldsymbol{\pi}(s) = \sum_{m=1}^{M} \overline{\omega}_m(s)\psi_{\boldsymbol{\pi}_u}(s)\beta_{u_m}, $$

obtaining the set of parameters $\beta_u = [\beta_{u_1}; \ldots; \beta_{u_M}]$ that minimize (4.27) or its decoupled upper bound (based on the inequality (4.9)), is simply a matter of rewriting the feature functions in (4.15) and (4.13) solutions, respectively, as

$$ \psi_{\boldsymbol{\pi}j}(s) = \hat{P}_j(s)\psi_{\boldsymbol{\pi}_u}(s). \tag{4.57} $$

We originally proposed splitting the estimation of the unconstrained policy into a two stage optimization because, intuitively, the estimation of the constraint would aid the estimation of the unconstrained policy. Later on, in [12], we formalized this learning decomposition as a special case of directly learning a policy defined as (4.18), in the following result:

Lemma 4.3.1. For a policy $\boldsymbol{\pi}(t) \triangleq A(t)^{\dagger}b(t) + P(t)\,{}^{\mathrm{u}}\boldsymbol{\pi}(t)$, if the rows of A are orthogonal, i.e. A is semi-orthogonal, then we can express the Direct Policy Error (DPE) of $\boldsymbol{\pi}$

$$ \varepsilon_{\mathrm{DPE}}(u; \boldsymbol{\pi}) \triangleq \int_0^T \|u(t) - \boldsymbol{\pi}(t)\|^2 dt. $$

as the sum

$$ \varepsilon_{\mathrm{DPE}}(u; A, b, {}^{\mathrm{u}}\boldsymbol{\pi}) = \varepsilon_{\mathrm{CSE}}(u; A, b) + \varepsilon_{\mathrm{CPE}}(u; P, {}^{\mathrm{u}}\boldsymbol{\pi}), \tag{4.58} $$

where

$$ \varepsilon_{\mathrm{CSE}}(u; A, b) \triangleq \int_0^T \|A(t)u(t) - b(t)\|^2 dt, $$

and

$$ \varepsilon_{\mathrm{CPE}}(u; P, {}^{\mathrm{u}}\boldsymbol{\pi}) \triangleq \int_0^T \|P(t)(u(t) - {}^{\mathrm{u}}\boldsymbol{\pi}(t))\|^2 dt. $$

Appendix A.5 contains the proof of the Lemma 4.3.1. It immediately follows that this result is also valid for the error metrics defined for observed samples rather than continuous trajectories. Therefore, we also have

$$ \overline{\varepsilon}_{\mathrm{DPE}}(\mathcal{X}; A, b, {}^{\mathrm{u}}\boldsymbol{\pi}) = \overline{\varepsilon}_{\mathrm{CSE}}(\mathcal{X}; A, b) + \overline{\varepsilon}_{\mathrm{CPE}}(\mathcal{X}; P, {}^{\mathrm{u}}\boldsymbol{\pi}), \tag{4.59} $$

where

$$\overline{\varepsilon}_{\text{DPE}}(\mathcal{X}; A, b, {}^{\text{u}}\boldsymbol{\pi}) \triangleq \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \|u_k - A^{\dagger}(s_k)b(s_k) - P(s_k)\,{}^{\text{u}}\boldsymbol{\pi}(s_k)\|^2, \tag{4.60}$$

and we defined the CSE and the CPE in (4.53) and (4.22), respectively.

This result supports the sequential optimization originally proposed in [10], in the sense that the two step optimization minimizes separately the first and second terms of the error in (4.59). However, because both $\overline{\varepsilon}_{\text{CSE}}$ and $\overline{\varepsilon}_{\text{CSE}}$ depend on the constraint matrix A, such sequential optimization still fails to fully decouple the estimation of the parameters from the constraint and from the unconstrained policy. In [10], we first estimate the constraint parameters for each sub-dataset by minimizing $\overline{\varepsilon}_{\text{CSE}}$ and then estimate the unconstrained policy parameters by minimizing $\overline{\varepsilon}_{\text{CSE}}$, where we set the constraint A to the respective estimate $\hat{A}$. Therefore, minimizing the sum of $\overline{\varepsilon}_{\text{CSE}}$ and $\overline{\varepsilon}_{\text{CSE}}$ is different from minimizing them sequentially. However, when estimating the unconstrained policy using the second term in (4.59), we use the data coming from several trajectories/demonstrations rather than a single one, hopefully addressing the challenge of indetermination when estimating unconstrained policies, discussed in the Subsection 4.2.4. Furthermore, if we wish that the result in (4.59) remains valid, then we need to introduce the semi-orthogonal condition for A when estimating it. For the case where $Au = 0$ and A is a constant matrix, then the SVD estimation method, corresponding to the optimization in (4.55) where we simply have that $A = \beta_{Ab}$ and $u = y_{Ab}$, already assures such condition. For all other cases, i.e. $b \neq 0$ and/or $A(s) = \beta_A \psi_A(s)$, then the previously proposed SVD constraint estimation approach no longer assures the orthogonality condition for A.

The Generalized Eigenvalue Decomposition (GEVD) Solution

For a constraint matrix modelled as a linear combination of a set of given/pre-specified non-linear feature functions $A(s) = \beta_A \psi_A(s)$, it will be virtually impossible to guarantee the orthogonality condition for every state s of a given trajectory/demonstration. In [12] we proposed, instead, to try to meet this orthogonality criteria on average, i.e

$$\frac{1}{T} \int_0^T A(t)A(t)^{\top} dt = I_{n_{\text{ts}}},$$

for the case of a continuous trajectory or

$$\frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} A(s_k)A(s_k)^{\top} = I_{n_{\text{ts}}}, \tag{4.61}$$

for a set of $\mathcal{K}$ samples along a trajectory. We can then rewrite (4.61) as

$$\begin{aligned}
\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} A(s_k)A(s_k)^\top &= \frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} \beta_A \psi_A(s_k)\psi_A(s_k)^\top \beta_A^\top \\
&= \beta_A \left(\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} \psi_A(s_k)\psi_A(s_k)^\top\right)\beta_A^\top \\
&= \underbrace{\begin{bmatrix}\beta_A & \beta_b\end{bmatrix}}_{\triangleq \beta_{Ab}} \left(\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} \underbrace{\begin{bmatrix}\psi_A(s_k)\\ 0\end{bmatrix}}_{\triangleq z_{Ab}(s_k)} \begin{bmatrix}\psi_A(s_k)^\top & 0\end{bmatrix}\right)\begin{bmatrix}\beta_A^\top \\ \beta_b^\top\end{bmatrix} \\
&= \beta_{Ab}\left(\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} z_{Ab}(s_k)z_{Ab}(s_k)^\top\right)\beta_{Ab}^\top \\
&= \beta_{Ab}\underbrace{\begin{bmatrix}\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}}\psi_A(s_k)\psi_A(s_k)^\top & 0\\ 0 & 0\end{bmatrix}}_{\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^\top}\beta_{Ab}^\top
\end{aligned} \tag{4.62}$$

with

$$\mathcal{Z}_{Ab} = \frac{1}{\sqrt{\mathcal{K}}}\left[z_{Ab}(s_1), \ldots, z_{Ab}(s_{\mathcal{K}})\right] \tag{4.63}$$

being the matrix containing the evaluation of z_{Ab} for a given dataset $\mathcal{X} = \{(s_k, u_k)\}_{k=1}^{\mathcal{K}}$.

We can then rewrite the optimization (4.55) with the update of the orthogonality on average condition (4.61), resulting in

$$\begin{aligned}
\hat{\beta}_{Ab} = \arg\min_{\beta_{Ab}} \quad & \sum_{k=1}^{\mathcal{K}} \|\beta_{Ab} y_{Ab}(s_k, u_k)\|^2 \\
s.t. \quad & \beta_{Ab}\left(\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} z_{Ab}(s_k)z_{Ab}(s_k)^\top\right)\beta_{Ab}^\top = I_{n_{\text{ts}}}.
\end{aligned} \tag{4.64}$$

Armesto et al. [12] propose that the generalized eigenvector corresponding to the smallest generalized eigenvalue of the matrix pair $(\mathcal{Y}_{Ab}\mathcal{Y}_{Ab}^\top, \mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^\top)$, obtained through a Generalized Eigenvalue Decomposition (GEVD), is a solution to (4.64).

The proof of the lemma presented in [12] proposing the GEVD as a solution to (4.64), assumes that $n_{\text{ts}} = 1$, i.e. $I_{n_{\text{ts}}} = 1$, which corresponds to a single constraint. Furthermore, even though that proof rightfully concludes that, for such case, the minimizer

of (4.64) must be a generalized eigenvector, it fails to clearly establish that such a vector corresponds indeed to the smallest generalized eigenvalue. Appendix A.6 extends the result of [12] to the case that $n_{\text{ts}} \geq 1$, proving that the transpose of the matrix of right generalized eigenvectors of the pair $(\mathcal{Y}_{Ab}\mathcal{Y}_{Ab}^{\top}, \mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top})$ corresponding to its smallest generalized eigenvalues, with $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ positive definite, is a solution to the optimization problem (4.64). Furthermore, Appendix A.6 shows that the corresponding minimum of (4.64) is the sum of the n_{ts} lowest generalized eigenvalues of the pair $(\mathcal{Y}_{Ab}\mathcal{Y}_{Ab}^{\top}, \mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top})$.

The result from Appendix A.6 deserves some more attention and discussion, because it introduces a crucial condition missed in our original work [12], which is that $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ be positive definite. By rewriting (4.64) as (Appendix A.6)

$$\begin{aligned} \hat{\beta}_{Ab} = \arg\min_{\beta_{Ab}} \quad & \text{Tr}(\beta \mathcal{Y}_{Ab}\mathcal{Y}_{Ab}^{\top}\beta^{\top}) \\ s.t. \quad & \beta_{Ab}\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}\beta_{Ab}^{\top} = I_{n_{\text{ts}}}, \end{aligned} \tag{4.65}$$

then Sameh and Wisniewski [145] also proved that the minimum for this problem corresponds to the sum of the smallest n_{ts} generalized eigenvalues, for $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ positive definite. Later on, Liang et al. [87] proved the same result, but for $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ positive semi-definite, deeming the non-singular assumption for $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ unnecessary for finding the minimum of (4.65). However, Appendix A.6 additionally provides a minimizer to (4.65) given $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ positive definite, and is unclear if such a result is a valid minimizer under the more relaxed positive semi-definite assumption, even though the minimum result is valid.

The reason why the positive definitiveness condition is so crucial becomes immediately apparent when we verify, by analying (4.62), that in fact $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top}$ is a positive semi-definite matrix. Therefore, we lack any proof or theoretical result to claim that the generalized eigenvector corresponding to the smallest generalized eigenvalue of the matrix pair $(\mathcal{Y}_{Ab}\mathcal{Y}_{Ab}^{\top}, \mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^{\top})$, obtained trough a GEVD, is a solution to (4.64). We could claim, that it is a solution to the following problem

$$\begin{aligned} \hat{\beta}_{A} = \arg\min_{\beta_{A}} \quad & \sum_{k=1}^{\mathcal{K}} \|\beta_{A} y_{A}(s_k, u_k)\|^2 \\ s.t. \quad & \beta_{A} \left(\frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \psi_{A}(s_k)\psi_{A}(s_k)^{\top} \right) \beta_{A}^{\top} = I_{n_{\text{ts}}}. \end{aligned} \tag{4.66}$$

where $y_{A}(s, u) = \psi_{A}(s)u$ and which we can rewrite as

$$\begin{aligned} \hat{\beta}_{A} = \arg\min_{\beta_{A}} \quad & \text{Tr}(\beta \mathcal{Y}_{A}\mathcal{Y}_{A}^{\top}\beta^{\top}) \\ s.t. \quad & \beta_{A}\mathcal{Z}_{A}\mathcal{Z}_{A}^{\top}\beta_{A}^{\top} = I_{n_{\text{ts}}}, \end{aligned} \tag{4.67}$$

where

$$\mathcal{Y}_A = \left[y_A(s_1), \ldots, y_A(s_\mathcal{K})\right]$$

and

$$\mathcal{Z}_A = \frac{1}{\sqrt{\mathcal{K}}}\left[\psi_A(s_1), \ldots, \psi_A(s_\mathcal{K})\right]$$

are matrices containing the evaluation of y_A and ψ_A, respectively, for a given dataset. In this case, we can choose ψ_A such that $\mathcal{Z}_A\mathcal{Z}_A^\top$ is positive definite and, therefore, we can use GEVD to obtain a minimizer of (4.66). However, this brings us back to the assumption that $Au = 0$. In fact, we verified in numerous experiments posterior to the work in [12], that using the GEVD to estimate β_A would lead to solutions that would indeed satisfy the condition of orthogonality on average, whereas when using it to estimate β_{Ab} often would not.

The Generalized Singular Value Decomposition (GSVD) Solution

To address the limitation of the need for positive definite $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^\top$, let's consider the Generalized Singular Value Decomposition (GSVD) of the pair $(\mathcal{Y}_{Ab}^\top, \mathcal{Z}_{Ab}^\top)$ [58, 175], as

$$\mathcal{Y}_{Ab}^\top = U_{\mathcal{Y}_{Ab}}\Sigma_{\mathcal{Y}_{Ab}}X_{(\mathcal{Y}_{Ab},\mathcal{Z}_{Ab})} \tag{4.68}$$

$$\mathcal{Z}_{Ab}^\top = V_{\mathcal{Z}_{Ab}}\Sigma_{\mathcal{Z}_{Ab}}X_{(\mathcal{Y}_{Ab},\mathcal{Z}_{Ab})}, \tag{4.69}$$

where $U_{\mathcal{Y}_{Ab}} \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ and $V_{\mathcal{Y}_{Ab}} \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ are orthogonal matrices, $\Sigma_{\mathcal{Y}_{Ab}} \in \mathbb{R}^{\mathcal{K}\times n_{Ab}}$ and $\Sigma_{\mathcal{Z}_{Ab}} \in \mathbb{R}^{\mathcal{K}\times n_{Ab}}$ are rectangular diagonal matrices whose diagonal elements range from 0 to 1 ordered in ascending and descending order, respectively, for which

$$\Sigma_{\mathcal{Y}_{Ab}}^\top\Sigma_{\mathcal{Y}_{Ab}} + \Sigma_{\mathcal{Z}_{Ab}}^\top\Sigma_{\mathcal{Z}_{Ab}} = I_{n_{Ab}},$$

and $X_{(\mathcal{Y}_{Ab},\mathcal{Z}_{Ab})} \in \mathbb{R}^{n_{Ab}\times n_{Ab}}$ is an invertible matrix. Appendix (A.7) shows that

$$\hat{\beta}_{Ab}^\top = X_{(\mathcal{Y}_{Ab},\mathcal{Z}_{Ab})}^{-1}S_{I0}, \tag{4.70}$$

where $S_{I0}^\top = \begin{bmatrix} I_{n_{\text{ts}}} & 0_{n_{\text{ts}}\times(n_{Ab}-n_{\text{ts}})}\end{bmatrix} \in \mathbb{R}^{n_{\text{ts}}\times n_{Ab}}$ is a selection matrix which extracts the first n_{ts} columns of $X_{(\mathcal{Y}_{Ab},\mathcal{Z}_{Ab})}^{-1}$, is a solution to the optimization problem in (4.64), given that the first n_{ts} elements of the diagonal of $\Sigma_{\mathcal{Z}_{Ab}}$ are unitary.

At first, it might seem that the condition that the first n_{ts} elements of the diagonal of $\Sigma_{\mathcal{Z}_{Ab}}$ be unitary is quite restrictive. Perhaps, even more restrictive than the condition that $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^\top$ be positive definite. In practice, though, we failed to encounter any situation where the number of unitary elements of the diagonal of $\Sigma_{\mathcal{Z}_{Ab}}$ would be smaller than the pre-specified number of constraints n_{ts}. Whereas, considering the existence of the task controller b always leads to $\mathcal{Z}_{Ab}\mathcal{Z}_{Ab}^\top$ non positive definite, as shown in (4.62), making the application of GEVD unsuitable. Furthermore, if there

are cases for which the number of unitary elements of the diagonal of $\Sigma_{\mathcal{Z}_{Ab}}$ are larger than the pre-specified number of constraints n_{ts}, then we can potentially use that information to adjust the n_{ts}. Listing 4.1 contains a snippet of MATLAB code exemplifying how to use SVD, GEVD, and GSVD to estimate the set of parameters β_{Ab} given the matrices $\mathcal{Y}_{Ab}$ and $\mathcal{Z}_{Ab}$ defined in (4.52) and (4.63), respectively.

Listing 4.1: MATLAB code for estimating β_{Ab} using SVD, GEVD, and GSVD.

```
%% INPUT: data matrices Y_Ab and Z_Ab; number of constraints n_ts
% Singular Value Decomposition for estimating beta_Ab
[Usvd,Ssvd,Vsvd]=svd(Y_Ab');
beta_hat_svd = Vsvd(:,(end-n_ts):end);
% Generalized Eigenvalue Decomposition for estimating beta_Ab
[Vgev,Dgev] = eig(Y_Ab*Y_Ab',Z_Ab*Z_Ab');
beta_hat_gev = Vgev(:,1:n_ts);
% Generalized Singular Value Decomposition for estimating beta_Ab
[Ugsvd,Vgsvd,Xgsvd,Cgsvd,Sgsvd] = gsvd(Y_Ab',Z_Ab');
if abs(Sgsvd(n_ts,n_ts) - 1.0) < eps_tol
    Xgsvd_inv = inv(Xgsvd');
    beta_hat_gsvd = Xgsvd_inv(:,1:n_ts);
else; error('GSVD invalid');
end
```

4.4 Case Studies

This section will present a number of different case studies contemplating the application of some of the approaches discussed thus far. Starting with a two dimensional (2D) example for a more careful analysis moving to higher dimensional examples, such as learning a wiping motion policy with a 7 Degrees of Freedom (DoF) robotic manipulator, and its execution using a KUKA LWR 3 arm.

4.4.1 Learning a 2D Policy

This subsection will focus on a simple two dimensional example, identical to the one used in [67]. Let's then consider the limit cycle attractor system represented in polar coordinates as

$$\dot{r} = r(\rho - r^2), \qquad \dot{\theta} = \omega$$

where $\rho = -0.5\,\text{m}^2$ and $\omega = 1.0\,\text{rad}\,\text{s}^{-1}$, for which the Cartesian state space coordinates becomes

$$s = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}.$$

We assume the control actions as being the time derivative of the state $u = \dot{s}$, subject to the equality constraint $Au = 0$, where

$$A = \begin{bmatrix} \cos\alpha & \sin\alpha \end{bmatrix}. \tag{4.71}$$

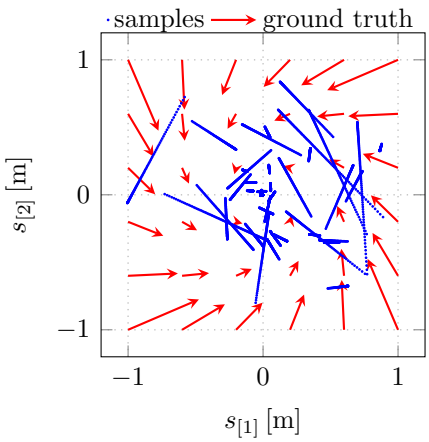


Figure 4.4: Vector field of the ground truth control actions in a 6×6 grid, in red, overlaid with 40 state trajectories generated under the constraint $Au = 0$, in blue, belonging to one of the datasets used for training.

We generated 40 datasets, each dataset containing 50 sub-datasets, where each sub-dataset consists of a single trajectory. Out of the 50 sub-datasets, we use the first 40 for training ($\mathcal{J} = 40$), reserving the last 10 for evaluation. Each trajectory contains 100 samples ($\mathcal{I} = 100$), corresponding to a 2 s simulation with a sampling rate of 50 Hz, of pairs of states and controls (s_{ji}, u_{ji}). Every single trajectory has a different initial condition uniformly sampled from the domains $\theta_{j1} \in [0, 2\pi]$ rad and $r_{j1} \in [0, 1]$ m, and a different underlying constraint, written as (4.71), sampled from the domain $\alpha_j \in [0, 2\pi]$ rad. Figure 4.4 shows the plot of the sequence of states of one of the datasets containing 40 trajectories used for training, along with the quiver plot of the ground truth unconstrained control actions in a 6×6 grid.

For the unconstrained policy we used a weighted combination of 16 local models, using receptive fields as importance weighting as defined in Expression (4.7), with centres uniformly distributed, according to Figure 4.5a, and using a diagonal variance matrix with diagonal elements of 0.01. For estimating the parameters of the unconstrained policy, we used both the local and global Least Squares (LS) methods described in the Subsection 4.2.2.

In this example we compare three of the previously discussed methods for learning the unconstrained policy. The Direct Policy Learning (DPL), discussed in Subsection 4.2.1, the Constraint Consistent Learning (CCL), introduced in the Subsection 4.2.6, and finally the Constraint-aware Policy Learning (CaPL) from Section 4.3, based on SVD. The goal being that by refining the level of constraint awareness in the learning method, the overall results will improve. For instance, by looking at Figure 4.4, which shows a quiver plot of the ground truth unconstrained controls and the unconstrained policy outputs for the same grid for both the DPL and CCL trained with 20 constrained trajectories, we can immediately see that the CCL ap-

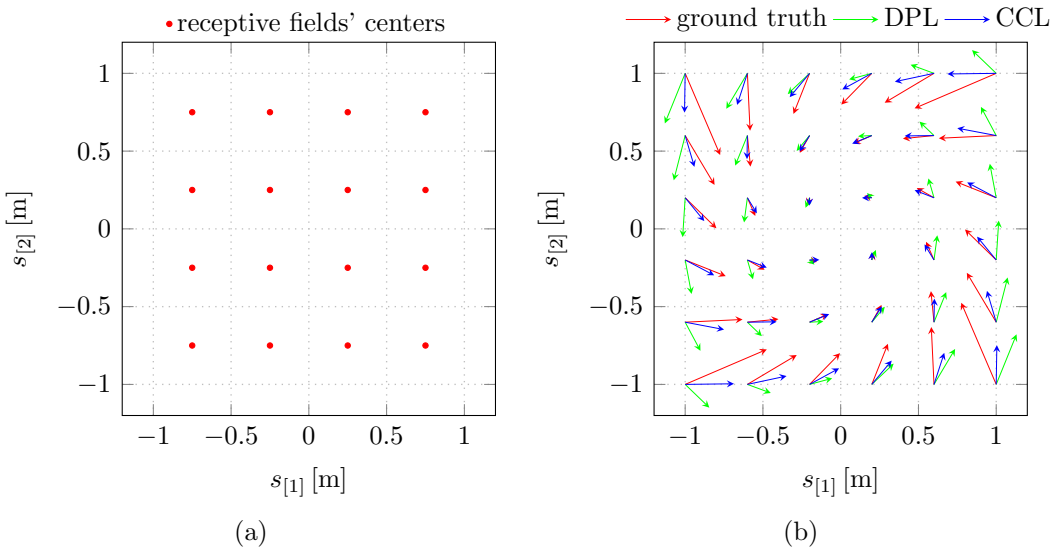


Figure 4.5: State space of the 2D example with (a) the unconstrained policy receptive fields' centers location; and (b) a 6×6 grid quiver plot of the ground truth control actions and the output of the unconstrained policies learned through DPL and CCL, trained only using 20 trajectories out of the total 40 trajectories in the dataset. Note that, for illustration purposes, for the arrows in (b) we use a scale for the DPL policy output which is $9\times$ the scale used for the ground truth and the CCL policy output.

proximates the ground truth unconstrained actions better than the naive DPL, which lacks any notion of constrained observations in its synthesis. Note, that when using the SVD method from the Subsection 4.3.1, we have that $\psi_A = I_2$ and $\beta_b = 0$ and, therefore, $A = \beta_A$.

The first experiment looks at how those methods perform for different levels of training data. In this experiment, we estimate the parameters using a subset from 5 to the full 40 training trajectories, and compute the Unconstrained Policy Error (UPE) and the Constrained Policy Error (CPE) for the 10 testing trajectories. Note that even though we have both the constrained and unconstrained ground truth controls for both the training and testing sub-datasets, for training purposes we only make use of the constrained control actions. Note also that we add a very small level of 50dB signal-to-noise ratio of noise to the training constrained observations. We perform this experiment for all the 40 datasets, recording the errors for each of them. Figure 4.6 shows the median together with the 10th and the 90th percentiles of the UPE and CPE for the experiment of comparing the three learning methods when training with a partial dataset, and using both the local and global LS estimators, discussed in Subsection 4.2.2.

We verify that for the local LS, the DPL method results in high error regardless of the number of trajectories used for training, which essentially means it completely

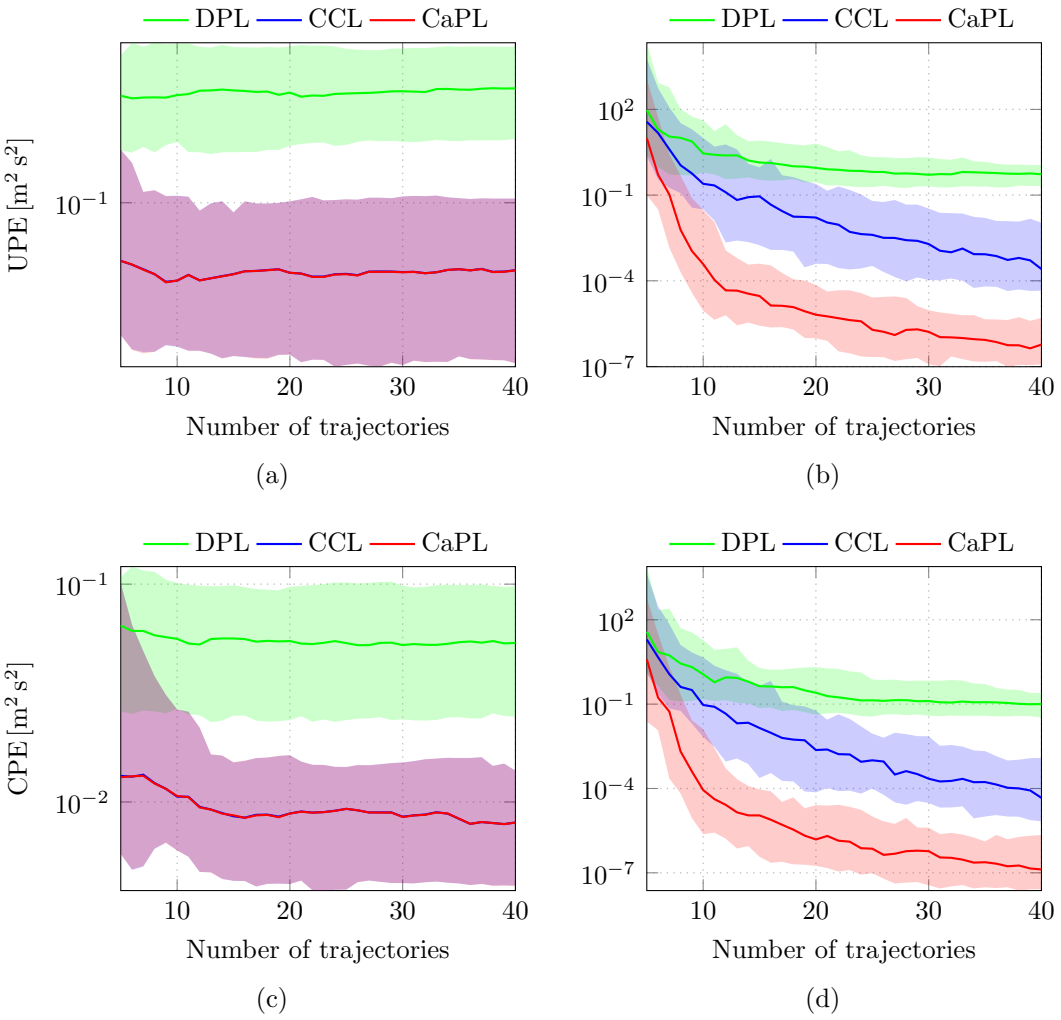


Figure 4.6: Unconstrained and constrained policy error evolution when increasing the number of trajectories used for training the unconstrained policy, with: (a) UPE for local LS; (b) UPE for global LS; (c) CPE for local LS; (d) CPE for global LS. The shaded area corresponds to the errors between the 10th and the 90th percentiles, i.e 80% of the 40 computed error curves fall within this area, and the line indicates the median value of the errors across the 40 datasets tested.

fails at learning anything. Interestingly enough, the CCL and the CaPL methods perform equally well reducing the error to the minimum level with about 15 training trajectories. For the global LS, all the methods improve the results with the increase of the number of training trajectories. However, the DPL stagnates much quicker at an higher error, and in this case the CaPL outperforms the CCL.

One intriguing result from the previous experiment is that for the local LS estimator, the CCL and CaPL perform equally well. This happens because at its core, CCL approximates the constraint projection matrix by a one dimensional projection using each observation. Therefore, for this experiment, in the absence of any noise, this approximation will exactly match the ground truth. A second experiment tests the robustness of the learning methods with regards to added levels of noise. In this second experiment, we train the policies using the full 40 trajectories but with varying levels of added noise, ranging from 50 to 10 dB of signal-to-noise ratio (with lower levels of signal-to-noise ratio corresponding to higher levels of noise) [136]. Similarly to the previous experiment, we compute the UPE and the CPE based on 10 testing trajectories, unseen during the training phase. Figure 4.7 shows the median together with the 10th and the 90th percentiles of the UPE and CPE for the experiment of comparing the three learning methods when adding increasing levels of noise to the observations, and using both the local and global LS estimators.

We can then verify that for the local LS, the CCL and the CaPL perform equally well only for very low levels of noise. For increasing levels of noise the CCL errors increase to close to the errors obtained by DPL. This shows that, in the presence of noise, there is a benefit of explicitly estimating the constraint when learning the unconstrained policy, at least for the case where the constraint matrix is constant, i.e. the constraint lives in the same space as the control actions. Interestingly enough, for the global LS case, the CaPL outperforms the CCL even for very low levels of noise and also, contrary to what happens with the local LS, its performance degrades with the increased levels of noise. This result is an indication that there are two contributions for the estimation error of the unconstrained policy based on the constrained observations: one, is the method for estimating the constraint; and the other is the method for estimating the unconstrained policy itself. That can explain the difference in errors between local and global LS when using the same method for the constraint estimation. Finally, it's worth noting that varying levels of noise hardily impact the high errors resulting from DPL, which shows that this method fails to capture the underlying unconstrained policy from constrained data.

Obviously, adding more complexity to a learning method comes with its own costs. One of those costs is the increased computation time. Figure 4.8 shows the median together with the 10th and 90th percentiles of the computation time for training the unconstrained policy for the three learning methods in comparison and for the first experiment, i.e the evolution of the computation time with the number of trajectories used for learning. Note that in the case of the CaPL, this includes both estimating the constraints and training the unconstrained policy. Note as well that

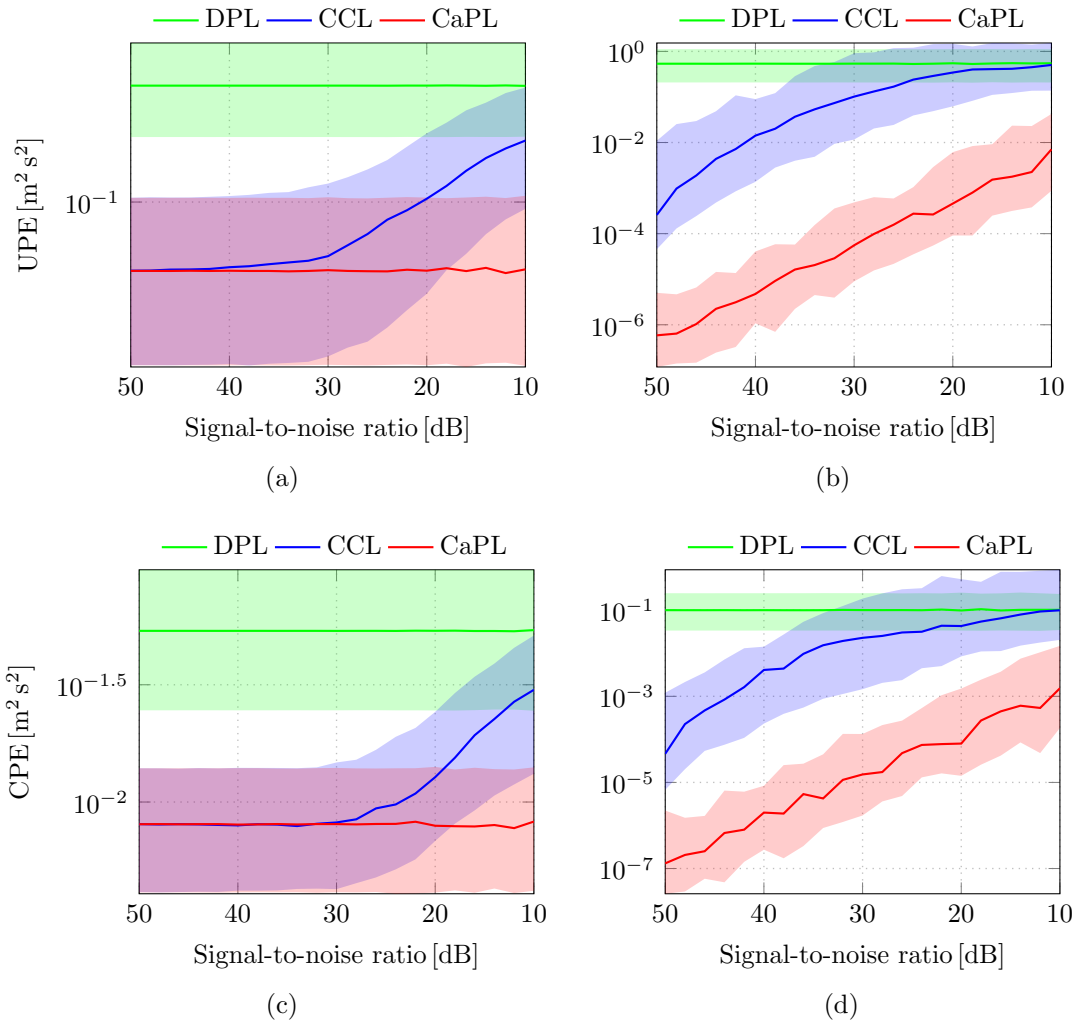


Figure 4.7: Unconstrained and constrained policy error evolution when increasing the level of noise of the control actions used for training the unconstrained policy, with: (a) UPE for local LS; (b) UPE for global LS; (c) CPE for local LS; (d) CPE for global LS. Note that lower signal-to-noise ratio in dB corresponds to higher levels of noise, where in the limit 0 decibel corresponds to a ratio of 1:1 which indicates the same level of background noise as the actual signal. The shaded area corresponds to the errors between the 10th and the 90th percentiles, i.e 80% of the 40 computed error curves fall within this area, and the line indicates the median value of the errors across the 40 datasets tested.

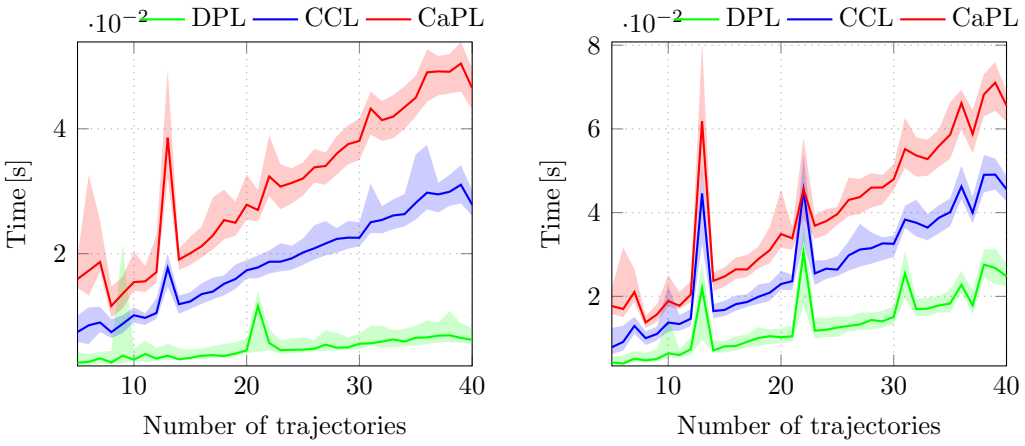


Figure 4.8: Computation time evolution when increasing the number of trajectories used for training the unconstrained policy, with: (a) local LS; (b) global LS; The shaded area corresponds to the errors between the 10th and the 90th percentiles, i.e 80% of the 40 computed error curves fall within this area, and the line indicates the median value of the errors across the 40 datasets tested.

the time computation comes from the standard MatLab® functions *tic* and *toc*, without extra care for repeatability and interference from other processes in the operating system, besides shutting down all unnecessary applications and the internet connection. Nevertheless, Figure 4.8 clearly shows an overall tendency of increased computation time with the number of trajectories used for learning and that CaPL takes longer to compute than CCL, which in turn takes longer than the simple DPL.

4.4.2 Learning a 2D Policy with Task Component

This subsection considers the same limit cycle attractor system presented in the previous subsection, with the difference that we now subject the control actions $u = \dot{s}$ to the task-based constraint $Au = b$, i.e. b is nonzero, and with A given by (4.71) (the same constraint matrix expression used previously). The generation of the dataset is also identical to the one in the previous subsection, with the exception of the existence of an additional parameter for the task component, being sampled from the domain $b_j \in [-0.3, 0.3]\ \mathrm{m\,s^{-1}}$. Figure 4.9 shows the plot of the sequence of states of one of the datasets containing 40 trajectories used for training, along with the quiver plot of the ground truth unconstrained control actions in a 6×6 grid. Note that the state trajectories now exhibit some curvature as a result of the task component b being nonzero, which contrasts with the rectilinear state trajectories from the previous experiment, shown in Figure 4.4.

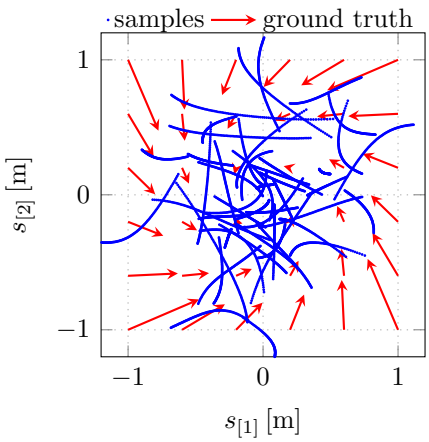


Figure 4.9: Vector field of the ground truth control actions in a 6×6 grid, in red, overlaid with 40 state trajectories generated under the constraint $Au = b$, in blue, belonging to one of the datasets used for training.

In this example we compare the Constraint Consistent Learning (CCL) method, presented in Subsection 4.2.6, with the Constraint-aware Policy Learning (CaPL) method, of Section 4.3. The goal is to show that CaPL is able to uncover/learn the unconstrained policy when we subject its output control actions to task-based constraints. Note that when using the closed-form solutions based either on the SVD decomposition, described in Subsection 4.3.1, or the GSVD decomposition, discussed in Subsection 4.3.2, for this particular example we have $\psi_A = I_2$ and $\psi_b = 1$ and, therefore, $A = \beta_A$ and $b = \beta_b$, where $\beta_{Ab} = \begin{bmatrix}\beta_A & \beta_b\end{bmatrix}$ is the set of task-based constraint parameters.

This experiment uses the same parameterization for the unconstrained policy described in the previous subsection. However, this time we only apply the global Least Squares (LS) method, presented in Subsection 4.2.2, for estimating its parameters.

We carry out the same two experiments as in the previous subsection. The first experiment looks at how the methods perform when estimating the policy parameters using a different number of training sub datasets, more specifically we train the policy using a subset of the available 40 training trajectories, varying the size of the subset from 5 to 40. The second experiment looks at the response of the learning methods with regards to the added levels of noise, i.e. we train the policies using the full 40 trajectories but with varying levels of added noise, ranging from 50 to 10 dB of signal-to-noise ratio (with lower levels of signal-to-noise ratio corresponding to higher levels of noise) [136]. In both experiments we compute the Unconstrained Policy Error (UPE) for the remaining 10 testing trajectories. Note that even though we have both the constrained and unconstrained ground truth controls for both the training and testing sub-datasets, for training purposes we only make use of the constrained control actions. Figure 4.10 shows the evolution of the median together with the

10th and the 90th percentiles of the UPE for both the experiments of learning with partial dataset and increasing levels of noise on the observations. Figure 4.11 shows the computation times for the experiment of learning with the partial dataset. Note that in this subsection we test the CaPL method using both the previously described algebraic decompositions SVD and GSVD.

As expected, the CCL method is unable to learn the unconstrained policy in the presence of task-based constraints, displaying large UPE regardless of the number of training trajectories and the added level of noise, as shown in Figure 4.10. On the other hand, the CaPL method successfully recovers the unconstrained policy in the presence of task-based constraints and displays a typical behaviour of improved performance for higher number of training trajectories and lower level of added noise. Note that regarding the estimation of the unconstrained policy the CaPL SVD and GSVD variants lead to the same results, hence Figure 4.10 shows a single curve for CaPL.

Given that one of the contributions of the CaPL method is the ability to estimate the component b, we also evaluated the evolution of the Mean Square Error (MSE) of $|\hat{b}|$ for the two experiments described above, i.e. increased number of training trajectories and levels of noise, as shown in Figure 4.12. Figure 4.12b shows that for low levels of noise the GSVD solution outperforms the SVD solution. The reason we use $|b|$ instead of b in the MSE computation is that when estimating A and b such that $Au = b$, if both $\hat{A}$ and $\hat{b}$ return with opposite signs to the ground truth, they still satisfy the constraint equation.

4.4.3 Learning Cartesian Circular Trajectories

This Subsection will focus on a simple three dimensional example, with the goal of contrasting the Direct Policy Learning (DPL) and the Constraint-aware Policy Learning (CaPL) regarding their generalization capabilities for a strikingly small dataset with only two training trajectories. Consider a particle moving in a three dimensional Cartesian space at constant speed - norm of the velocity vector - and at constant distance of one meter from the origin. When restricting the motion of this particle to a plane intersecting the origin, the resulting trajectory is a circumference centred at the origin. Our aim is to learn this circular motion for any other plane intersecting the origin, provided a set of trajectories of the particle constrained to different planes. We captured two particular trajectories of this particle when constrained to move in two planes with an inclination of 60° with the state/position coordinate $s_{[2]}$ axis, as shown in Figure 4.13.

For this problem we define the state $s \in \mathbb{R}^3$ as the Cartesian position of the particle and the actions $u \in \mathbb{R}^3$ as the particle velocity. Each sub-dataset has 500 data points that correspond to a full revolution with duration of 5 seconds — Figure 4.13 plot uses one fifth of the total number of training samples. We generate the sub-datasets

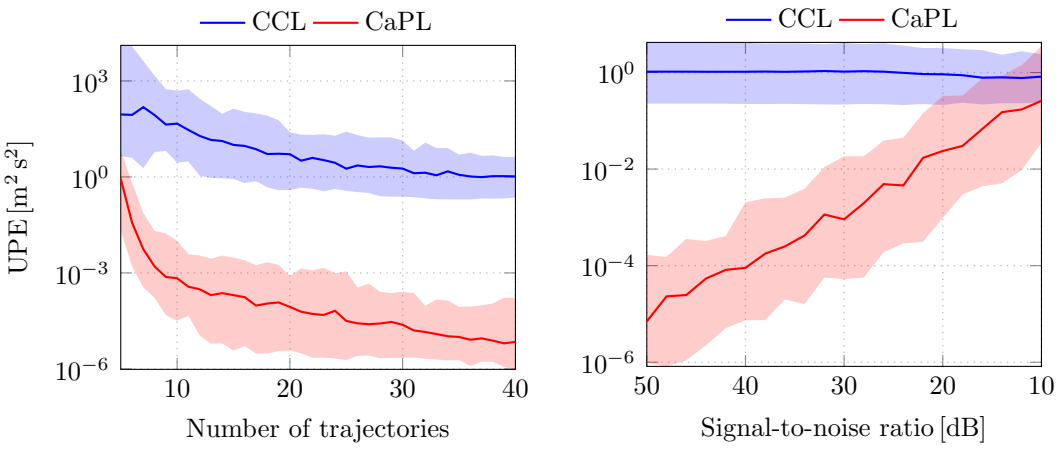


Figure 4.10: Unconstrained policy error evolution when (a) increasing the number of trajectories and (b) increasing the level of noise, for global LS. Note that lower signal-to-noise ratio in dB corresponds to higher levels of noise, where in the limit 0 decibel corresponds to a ratio of 1:1 which indicates the same level of background noise as the actual signal. The shaded area corresponds to the errors between the 10th and the 90th percentiles, i.e 80% of the 40 computed error curves fall within this area, and the line indicates the median value of the errors across the 40 datasets tested.

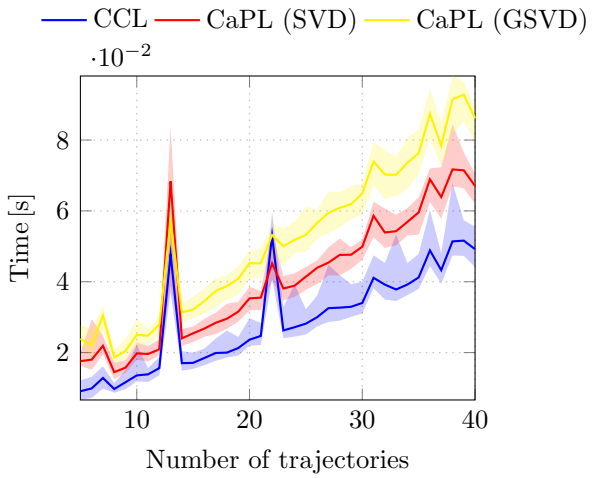


Figure 4.11: Computation time evolution when increasing the number of trajectories used for training the unconstrained policy, with global LS. The shaded area corresponds to the errors between the 10th and the 90th percentiles, i.e 80% of the 40 computed error curves fall within this area, and the line indicates the median value of the errors across the 40 datasets tested.

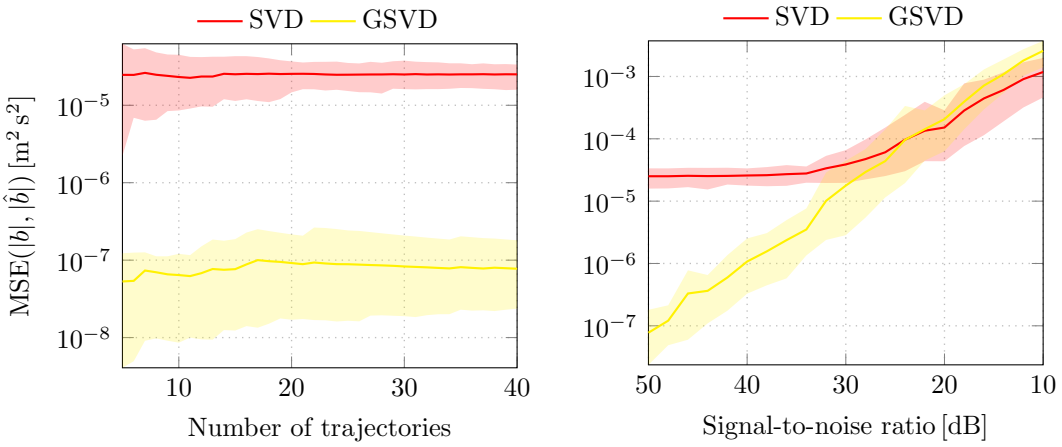


Figure 4.12: Mean Square Error (MSE) evolution of the estimated task component $|\hat{b}|$ when (a) increasing the number of trajectories and (b) increasing the level of noise, for global LS. Note that lower signal-to-noise ratio in dB corresponds to higher levels of noise, where in the limit 0 decibel corresponds to a ratio of 1:1 which indicates the same level of background noise as the actual signal. The shaded area corresponds to the errors between the 10th and the 90th percentiles, i.e 80% of the 40 computed error curves fall within this area, and the line indicates the median value of the errors across the 40 datasets tested.

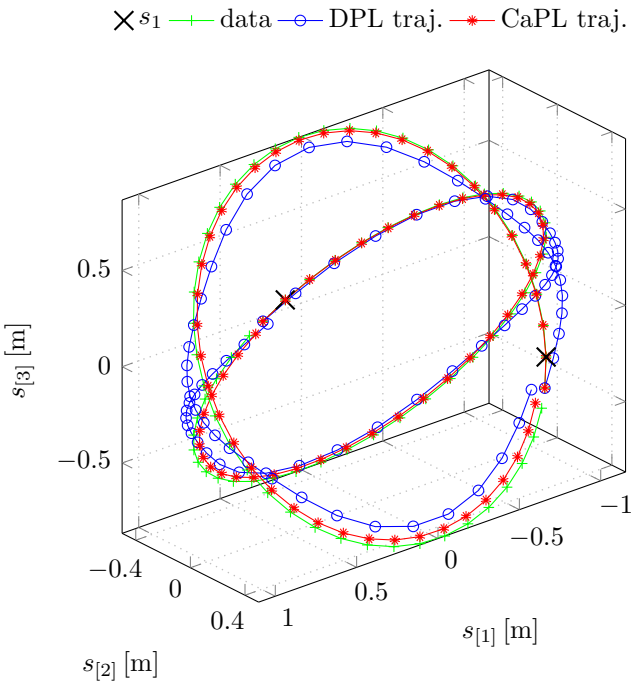


Figure 4.13: Two circular trajectories of a particle moving constrained to two different two dimensional planes in a three dimensional Cartesian space. Plot of the training data and the result of replaying two policies learned through Direct Policy Learning (DPL) and Constraint-aware Policy Learning (CaPL), starting at the same initial position s_1 and subject to the same planar constraints. The constraint planes and the training circles' centers both intersect the origin, making a 60° inclination with the coordinate $s_{[2]}$ axis.

by first generating the circle trajectory and then obtaining the sample control actions as the difference of the positions divided by the constant time step.

As described in Section 4.3, CaPL consists in a two stage process: first, learning the constraint for each of the sub-datasets; and second, extracting the null space component of each of the data point actions using the estimated null space projection matrix, and learn the unconstrained policy using only the estimated null space component of the actions. Given that the constraint is linear in the state space, we define the feature functions for the constraint matrix A as a constant matrix $\psi_A = I_3$. Moreover, as the particle never leaves the constraint plane, the first task is unnecessary and therefore $b = 0$, corresponding to the case where there is only a null space component of the actions and no task component. Given the noiseless training data, the estimated constraint parameters $\hat{\beta}_{A_1} = \begin{bmatrix} 0.0 & -0.866 & 0.5 \end{bmatrix}$ and $\hat{\beta}_{A_2} = \begin{bmatrix} 0.0 & 0.866 & 0.5 \end{bmatrix}$ exactly match the normals of the planes used in the generation of the training data. Having estimated the constraint matrix A, we can compute the estimated null space projection matrix and then compute the null space component of the training actions using (4.20). In the process of estimating the constraint matrix using SVD we can also obtain the singular values corresponding to each constraint, which are $\{14.0493, 14.0493, 4.2494 \times 10^{-14}\}$ for the first constraint and $\{14.0493, 14.0493, 3.9491 \times 10^{-14}\}$ for the second. Indeed, this shows that we can recover the dimensionality of the constraint from the number of singular values close to zero.

For the unconstrained policy we used a weighted combination of $M = 20$ local models as defined in (4.7), with local $\boldsymbol{\pi}_m$'s as in (4.11), using the following local feature function

$$\psi_{\boldsymbol{\pi}}(s) \triangleq s^\top \otimes I_3, \tag{4.72}$$

where $\otimes$ denotes the Kronecker product operator. We obtained the receptive field centers by running the K-means algorithm, using the function *kmeans* from MATLAB, on the states of the training data, and obtained the diagonal elements of the variance matrix Σ_m by computing standard deviation of the training data states and multiplying it by a scale of 0.2. The policy parameters $\hat{\beta}_{\pi_{\text{CaPL}}}$ result from minimizing the Constrained Policy Error $\overline{\varepsilon}_{\text{CPE}}$ in (4.22) using the estimated projection matrix $\hat{P}_j = I_3 - \hat{\beta}_{A_j}^\top \hat{\beta}_{A_j}$, which corresponds to a standard Least Squares (LS) problem.

For the DPL we used the same policy function used as the unconstrained policy in the CaPL. Therefore, estimating the parameters $\beta_{\pi_{\text{DPL}}}$ also corresponds to a standard LS problem using directly the control observations from the datasets, without any null-space projection transformation. When replaying the learned policies, we guarantee that the generated motion remains on the constraint plane by multipling the policy output with the corresponding null space projection matrix, as

$$u = P\,{}^{\text{u}}\boldsymbol{\pi}(s, \beta_{\boldsymbol{\pi}}). \tag{4.73}$$

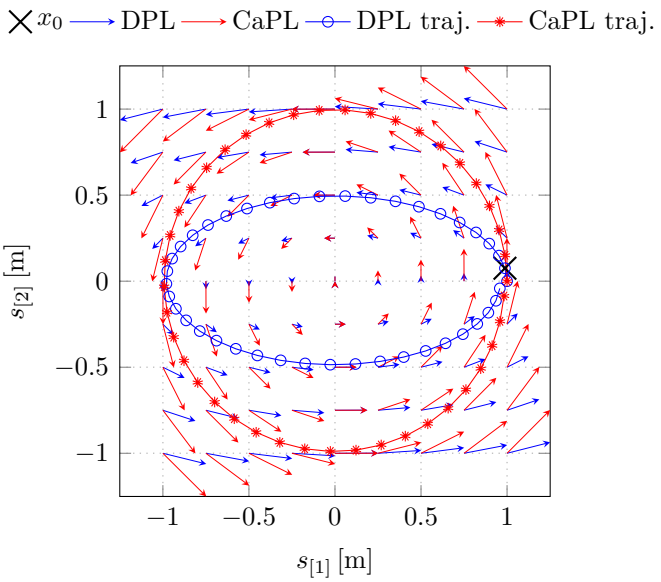


Figure 4.14: Vector field of two policies learned through Direct Policy Learning (DPL) and Constraint-aware Policy Learning (CaPL), constrained to the $s_{[1]} - s_{[2]}$ plane, and two trajectories obtained when replaying these two policies for a given initial position s_1.

For the case of the DPL we have $\beta_\pi = \beta_{\pi_{\text{DPL}}}$ and for the case of the CaPL then $\beta_\pi = \beta_{\pi_{\text{CaPL}}}$.

Figure 4.13 shows the trajectories obtained by replaying the two policies starting at the same initial positions as the training data and subject to the same constraints, showing that the CaPL approximates the training trajectories better than DPL. Note, that we tuned the number of local models and their variance so that ${}^{\text{u}}\boldsymbol{\pi}(s, \beta_{\pi_{\text{DPL}}})$ could approximate reasonably well the two training trajectories. For the case of the CaPL, actually one local model would suffice to give a similar result to the one given by the 20 local models, showing the capability of the CaPL method to capture this simple constrained motion.

However, we are interested in the ability to generalize to unseen constraints rather than just learning a set of given trajectories. Therefore, we replayed both learned policies for a different constraint from the ones used for training. Figure 4.14 shows the vector field of the two policies constrained to the horizontal $s_{[1]} - s_{[2]}$ plane, and one example of a trajectory resulting from playing the learned policies when starting at a given initial position. As shown in Figure 4.14, when playing the direct learned policy in an unseen constraint, it fails to capture the circular motion present in the demonstrations, resulting in a motion that resembles more of an average of the training trajectories than the intended circle. This results from mixing all training data when learning the policy without accounting for the different constraints that each demonstration was subject to.

4.4.4 Learning Planar-Constrained Policies

Defining the appropriate set of feature functions can be a difficult task without prior knowledge about the application. In this section, we propose to exploit the prior knowledge of the application by using Jacobian matrices of the end-effector as the main feature functions for learning both the constraint and the unconstrained policy. This will allow us to define exact models for tasks demonstrated on planar surfaces. After training the unconstrained policy, the robot can execute it on non-planar surfaces as long as there is some task control that guarantees the alignment of the end-effector with the surface (e.g. by using force-feedback). This parameterisation is useful for applications where non-planar surfaces constrain the motion of the robot such as in wiping, dusting, sweeping, scratching, writing, etc. In all these examples, we can define a task-based constraint as the minimization of the distance to the surface and the misalignment between the surface normal and the orientation of the robot's tool (see Figure 4.18). The null-space of this task would be any motion of the robot's tool on the surface, i.e., with speed of movements tangential to the surface.

Learning the constraint and the tasks controller

Let us consider a robot with some tool at its end, where x represents its three dimensional position/coordinates and the unitary vectors $\tilde{e}_1$, $\tilde{e}_2$ and $\tilde{e}_3$ represent its local reference frame. We consider training scenarios where the reference surface is flat and static, as shown in Figure 4.15. Given a normal to the surface $\tilde{n}$ that remains constant throughout the demonstration, we can define a task error using the distance of the tool to the surface and the tool's misalignment, as

$$\varepsilon_b(s(t)) \triangleq \begin{bmatrix} \tilde{n}^\top (x(s(t)) - p) \\ \tilde{n}^\top \tilde{e}_1(s(t)) \\ \tilde{n}^\top \tilde{e}_2(s(t)) \end{bmatrix}, \tag{4.74}$$

where p is any arbitrarily chosen point on the surface.

By differentiating (4.74), we get

$$\underbrace{\dot{\varepsilon}_b(s(t))}_{\triangleq b(t)} = \underbrace{\frac{\partial \varepsilon_b(s)}{\partial s}}_{\triangleq A(s(t))} \underbrace{\dot{s}(t)}_{\triangleq u(t)}, \tag{4.75}$$

where $A(\cdot)$ in this case is the analytical Jacobian of the task. In this particular problem $b(\cdot)$ becomes the dynamics of the implicit alignment controller ensuring that the error converges to zero. We assume that the demonstrator crafts $u(t)$ such that it pursues some surface approximation and alignment task, with a certain target "closed-loop dynamics" when there is an initial error.

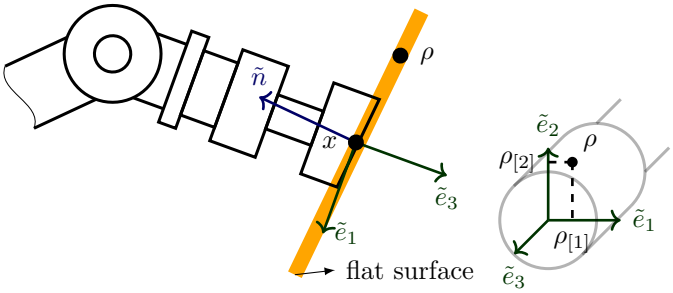


Figure 4.15: A two dimensional illustration of the robot motion on a flat surface, where $\tilde{n}$ represents the normal of the surface, $\tilde{e}_1, \tilde{e}_2, \tilde{e}_3$ are unitary axis of the tool local reference frame with origin in the contact position x, and ρ is a reference point on the surface.

From (4.75) and (4.74) we can select the following feature functions

$$\psi_A(s) \triangleq \begin{bmatrix} \partial x(s)/\partial s \\ \partial \tilde{e}_1/\partial s \\ \partial \tilde{e}_2/\partial s \end{bmatrix}, \tag{4.76}$$

$$\psi_b(s) \triangleq \begin{bmatrix} x(s) \\ \tilde{e}_1(s) \\ \tilde{e}_2(s) \\ 1 \end{bmatrix}, \tag{4.77}$$

where the above task controller would attempt to achieve a linear time-invariant stable closed loop, so the position and alignment error converge to zero. Indeed, note that

$$A(s) = \underbrace{\begin{bmatrix} \tilde{n}^\top & 0 & 0 \\ 0 & \tilde{n}^\top & 0 \\ 0 & 0 & \tilde{n}^\top \end{bmatrix}}_{\beta_A} \psi_A(s) \tag{4.78}$$

and thus, the choice of $\psi_A(s)$ corresponds to the ground-truth A expressed as the linear-in-parameter expression (4.78).

In differential kinematics [155], we can define the state of a robot by the joint positions, $s = q$, for which case the feature functions $\psi_A(q)$, corresponding to a Jacobian of the robot, are sufficient to fully describe the task of minimizing the misalignment error to the surface. However, the feature functions $\psi_b(q)$ might be insufficient to correctly characterize the alignment task, as the human operator might be acting under some non-linear controller for alignment. Note also that measurement noise and small varying distances to the surface during the demonstration will, in general, make it impossible for the CSE and the CPE in (4.53) and (4.22) to become exactly zero. As earlier commented, from the analysis of the singular values, if the number

of singular values that are significantly smaller in comparison to the rest is less than the dimension of ε_b, then we can clearly identify a situation where we need extra parametrization.

Regarding the feature functions $\psi_b(q)$, in this wiping type of applications it might be unnecessary to learn task controllers, since we can ensure contact and alignment with the surface via sensory feedback. This means that the recommendation for practical applications would be to provide demonstrations with an initial configuration already on the surface (or trimming the prior samples of the actual demonstration data) and assume $b = 0$. This assumption has computational benefits, allowing the use of the CaPL based on the SVD rather than the generalised version discussed earlier.

Learning the Unconstrained Policy

We will now propose a specialized structure for the unconstrained policy ${}^{\mathrm{u}}\boldsymbol{\pi}(\cdot)$ based on the Jacobian specific to the planar-constrained task under consideration. Incorporating problem-dependent information when building the feature functions instead of generic universal black-box feature functions will allow to us to improve accuracy and decrease the number of parameters. Indeed, some literature strongly advocates that learning control policies in the operational space is beneficial [128], hence the inclusion of the Jacobian. Recall that the task controller attempts to align the tool orientation with the surface (constraining two DoF) and maintain the contact (constraining one more DoF). This implies a task which constrains a total of three DoF of the robot. We can reasonably assume that any motion along the surface will be part of the null-space of the task controller, with the remaining DoF available.

Since the task controller constrains the tool's orientation, only the position trajectory $x(t)$ is relevant for the unconstrained policy. We choose an arbitrary reference frame $(\tilde{l}_1(\tilde{n}),\ \tilde{l}_2(\tilde{n}))$ on the surface orthogonal to the normal $\tilde{n}$, and we define the unconstrained feature functions as

$$\psi_{\boldsymbol{\pi}}(q;\tilde{n}) \triangleq \left(\begin{bmatrix} \tilde{l}_1(\tilde{n}) & 0 & 0 \\ \tilde{l}_2(\tilde{n}) & 0 & 0 \end{bmatrix} \psi_A(q) \right)^{\dagger} \psi_{2\mathrm{D}}(q), \tag{4.79}$$

which computes the tool's speed relative to this reference frame, and where $\psi_{2\mathrm{D}}$ are feature functions with a two dimensional output, used to represent the two dimensional trajectory demonstrated on the surface. The estimated parameters $\hat{\beta}_A$ will, in general, differ from the block-diagonal expression arising from (4.78), nonetheless, if the orientation error during the demonstration is reasonably small, the surface normal will be close to the tool's orientation $\tilde{e}_3(q)$ vector. So, we will neglect this error and propose the parametrization based on a modified tool's Jacobian $J_{2\mathrm{D}}$, as

$$\psi_{\boldsymbol{\pi}}(q) \triangleq \underbrace{\left(\begin{bmatrix} \tilde{l}_1(\tilde{e}_3(q)) & 0 & 0 \\ \tilde{l}_2(\tilde{e}_3(q)) & 0 & 0 \end{bmatrix} \psi_A(q) \right)^{\dagger}}_{\triangleq J_{2\mathrm{D}}^{\dagger}} \psi_{2\mathrm{D}}(q), \tag{4.80}$$

which, basically, coincides with (4.79) for small misalignment deviations during the demonstration.

For encoding the motion of the full configuration of the robot, for instance the existence of a preferred configuration, we can extend the unconstrained policy feature functions as

$$\psi_{\boldsymbol{\pi}}(q) \triangleq \begin{bmatrix} J_{2\mathrm{D}}^{\dagger}\psi_{2\mathrm{D}}(q) & (I_{n_q} - J_{2\mathrm{D}}^{\dagger}J_{2\mathrm{D}})\psi_q(q), \end{bmatrix} \tag{4.81}$$

where in the absence of any prior domain knowledge we can pick a generic set of feature functions for the preferred configuration

$$\psi_q(q) = \begin{bmatrix} q^{\top} & 1 \end{bmatrix} \otimes I_{n_q}. \tag{4.82}$$

Note that in the absence of the domain knowledge about the type of demonstrations, which in this case consists of a sliding motion along a planar surface, the unconstrained policy feature functions would have to be some set of generic functions like in (4.82), encoding no prior information about the task at hand. This would lead to the need of a higher number of local models, parameters, which in turn would require much more demonstrations for such encodings to represent the demonstrated motions.

Likewise, the choice of feature functions encoding the planar trajectory can also reflect our prior knowledge about that motion. For example, for a circular wiping motion, we can define the following feature functions

$$\psi_{2\mathrm{D}}(q) \triangleq \begin{bmatrix} \rho^{\perp}(q) & \rho(q)\left(1 - \frac{r}{\|\rho(q)\|}\right), \end{bmatrix} \tag{4.83}$$

where $\rho \in \mathbb{R}^2$ are the coordinates of the circle center (relative to the end-effector frame), $\rho^{\perp}$ is perpendicular to ρ and r is the radius. When training a model from real experimental data, we can easily extract these parameters prior to obtaining the policy parameters $\beta_{\boldsymbol{\pi}}$.

We recorded a dataset of wiping trajectories demonstrated by a human, as shown in Figure 4.16. The dataset contains 12 trajectories, each on a surface of a different orientation (four of them shown in Figure 4.17). Each demonstration involved several circles with the tool of the robot, containing approximately 2000 data points[1] (using a sampling rate of 100Hz). We minimally cropped the demonstrated data to ensure the data contained only poses where the tool was in contact with the surface and moving along the demonstrated trajectory.

We first learned the constraint for each trajectory, by parametrising them as a linear combination of feature functions, functions of the configuration, and by applying the estimation method described in Section 4.3.1. The feature functions for each constraint matrix are the ones from (4.76). For the policy ${}^{u}\boldsymbol{\pi}$ we used 25 Locally

[1]The data included the joint positions q and the joint commands $u = \dot{q}$, obtained by simple first-order Euler differentiation of the joint positions.

Figure 4.16: Demonstration of a circular wiping trajectory on a flat surface. The demonstration was repeated on surfaces of multiple orientations.

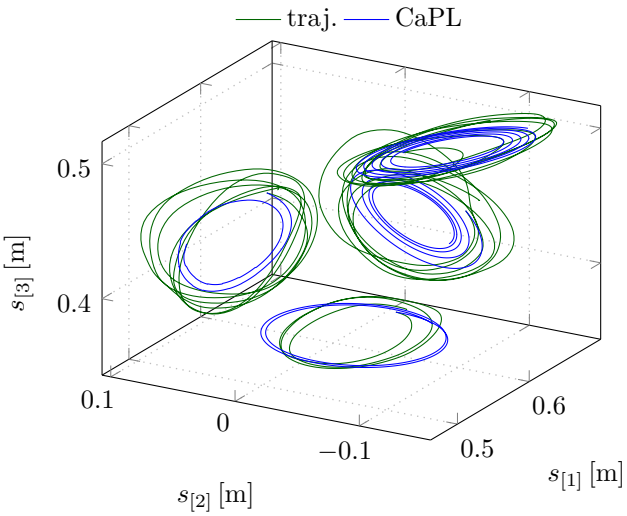


Figure 4.17: Learning by demonstration: four of the twelve wiping trajectories from human demonstration (green), and closed-loop policy validation using the respective flat surface orientation and initial position (blue).

Table 4.2: Values of the costs $\overline{\varepsilon}_{\text{DPE}}$, $\overline{\varepsilon}_{\text{CSE}}$, and $\overline{\varepsilon}_{\text{CPE}}$ for the four experimental demonstrations shown in Figure 4.17.

Dem.	$\overline{\varepsilon}_{\text{DPE}}$	$\overline{\varepsilon}_{\text{CSE}}$	$\overline{\varepsilon}_{\text{CPE}}$
1	0.0206	$0.81 \cdot 10^{-6}$	0.0199
2	0.0445	$2.36 \cdot 10^{-6}$	0.0431
4	0.0319	$2.39 \cdot 10^{-6}$	0.0302
7	0.0199	$4.43 \cdot 10^{-6}$	0.0175

Weighted Models with the same feature functions from (4.81). We then stored the resulting policy and used it, in closed-loop, together with the force-based surface alignment task that will be described in the following section.

Figure 4.17 shows the robot's end-effector trajectory corresponding to the execution of the estimated unconstrained policy for the same constraint (surface inclination) of the demonstrations, as well as the respective end-effector position corresponding to the data. The figure shows that the learnt locally-weighted model exhibits a "common" circular wiping motion across the different surface inclinations. Table 4.2 shows the result of computing the costs $\overline{\varepsilon}_{\text{DPE}}$, $\overline{\varepsilon}_{\text{CSE}}$, and $\overline{\varepsilon}_{\text{CPE}}$, according to equations (4.60), (4.53), and (4.22), respectively.

4.4.5 Task Generalization using a Force Sensor

This Subsection shows the utility of learning surface-constrained policies through generalization to a novel task. In many scenarios such as in the train cleaning application (Figure 4.1), it might be hard to obtain a precise model of the surface due to outdoors lighting conditions, different surface materials, and its dimensions. Thus, in practical applications, the constraint surface may be unknown. Therefore, we aim to redefine the surface alignment task using, for instance, a force/torque sensor.

To guarantee the alignment between the robot end-effector and the curved surface, the robot must exert some contact force on the surface and adjust the end-effector orientation to be perpendicular to that surface. As shown in Figure 4.15, this alignment corresponds to having the end-effector local $\tilde{e}_3$ axis collinear with the surface normal and the end-effector local $\tilde{e}_1$ and $\tilde{e}_2$ axes tangential to the surface. This alignment corresponds to having the minimal torque around the local $\tilde{e}_1$ and $\tilde{e}_2$ axes at the contact point, and having the contact force applied along the local $\tilde{e}_3$ axis. Therefore, we can define an alignment task error as

$$\varepsilon_f \triangleq \begin{bmatrix} f_c - f_3 \\ -m_1 \\ -m_2 \end{bmatrix}, \tag{4.84}$$

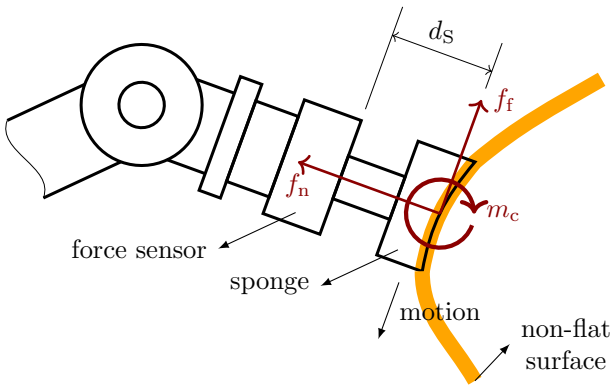


Figure 4.18: Two dimensional illustration of a robot performing a constrained task on a curved surface. The robot uses a force sensor and a soft material (sponge) mounted at the end-effector as a tool. The interaction of the wiping tool and the surface causes a friction force f_f, a normal force f_n, and a contact torque m_c. The task is to align the tool with the surface normal, by minimizing the contact torque m_c, and maintain contact by controlling the normal force f_n.

where f_3 is the measured force along the $\tilde{e}_3$ axis and f_c is the desired contact force; and m_1, and m_2 are the measured torques along the axes $\tilde{e}_1$ and $\tilde{e}_2$, respectively. By attaching a force/torque sensor at the tip of the end-effector, we can measure the contact wrench (force and torque), and by minimizing ε_f, the robot end-effector will align with the contact surface.

In this scenario, we can redefine a new constraint matrix as

$$A(q) \triangleq \begin{bmatrix} \tilde{e}_3^\top & 0 \\ 0 & \tilde{e}_1^\top \\ 0 & \tilde{e}_2^\top \end{bmatrix} J(q) \tag{4.85}$$

where $J \in \mathbb{R}^{6\times 7}$ represents a standard geometric robot Jacobian. We intentionally used a different constraint matrix (4.85) for the real-time operation (based on sensor information). This constraint replaces the purely geometric choice (4.78) during learning in order to show the generalisation capabilities to a new primary constraint/control law. Additionally, we define our task controller as $b = -K_p\varepsilon_f$, where $K_p \in \mathbb{R}^{3\times 3}$ is a diagonal matrix with its entries being tunable control gains.

In our experiments, we used the 7 DoF KUKA LWR3 robot with an ATI industrial automation Gamma F/T sensor attached at the end-effector, as shown in Figure 4.19. The force sensor retrieves a 6 dimensional wrench vector expressed in the sensor frame. Therefore, we compute the torque at the contact point by transforming the wrench by distance d_S towards the contact point. We estimated this distance empirically by pressing the tool against surfaces at different angles. We use admittance control to achieve the minimization of the force-based task error (4.84), i.e the robot compensates the end-effector position and orientation according to the wrench

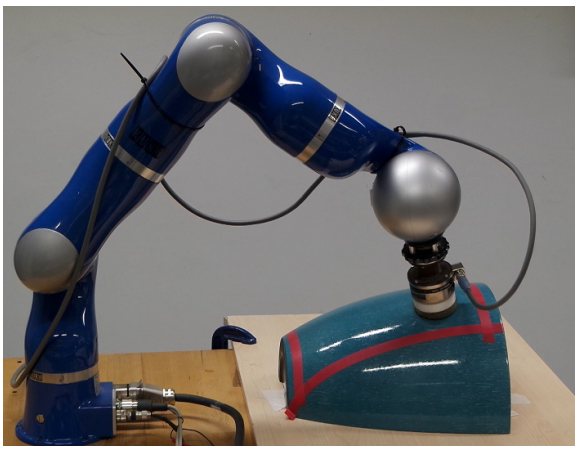


Figure 4.19: KUKA LWR 3 robotic arm equipped with a force/torque sensor wiping a curved surface.

feedback. In order to accommodate this motion when in contact with a rigid surface, we introduced a compliant material at the end-effector tip (such as a sponge).

Furthermore, we have also validated the learnt policy on a non-flat surface, as shown in Figure 4.19, demonstrating that the policy, trained from human demonstrations on flat surfaces, generalizes to both flat and curved surfaces. The resulting wiping motion is depicted in Figure 4.20. Note that we have demonstrated the wiping motion exclusively on flat surfaces, therefore, showing two aspects of generalization: (i) from a surface alignment task to a force alignment task, and (ii) from flat surfaces to a curved surface. See [11] for the video recordings of the policy generalization to a curved surface. In many practical cases, training with flat surfaces will be easier for the demonstrator (for instance to properly align the tool with the surface), resulting in a dataset with demonstrations in which $Au \approx 0$, and consequently reducing the amount of error in the task policy.

4.5 Constraint Similarity Analysis

In all experiments so far, we assumed that the demonstrator provides a set of sub-datasets $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_\mathcal{J}\}$, each of which containing samples of pairs of raw observations, that encapsulate a sufficiently diverse set of tasks/constraints, allowing us to uncover the underlying policy common to all demonstrations that is, therefore, generalizable to different task-based constraints. The aim now is to analyse how similar/distinct are these sub-datasets from one another, regarding the estimated underlying constraint, by using the same cost metric proposed for the constraint estimation, the Constraint Space Error (CSE). Moreover, we consider this analysis for the case of a single full dataset containing data originated by different constraints, in order to help us identify the transition regions. The experiment in this section

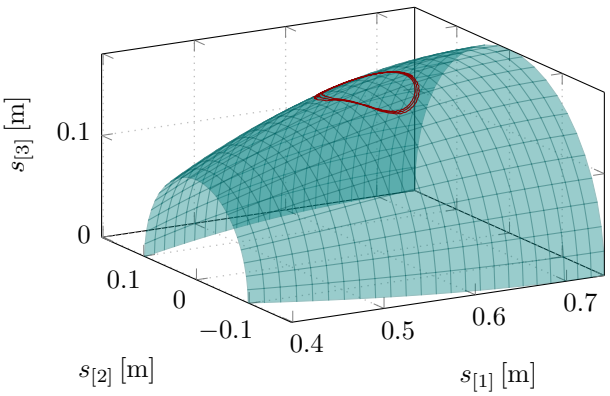


Figure 4.20: Resulting three dimensional trajectory of the robot end-effector (in red), overlaid with a model of the surface, when replaying the wiping policy trained from human demonstrations on flat surfaces (without using the force sensor). The policy generalizes to non-flat surfaces using a force-sensor based task to align the tool dynamically.

provides an additional analysis of the training data, highlighting the importance of constraint estimation when having data collected under different constraints.

Let's consider a set of sub-datasets $\{\mathcal{X}_1, \ldots, \mathcal{X}_l, \ldots, \mathcal{X}_h, \ldots, \mathcal{X}_\mathcal{J}\}$. One way of analysing the constraint similarity between two arbitrary sub-datasets, such as $\mathcal{X}_l$ and $\mathcal{X}_h$, is to estimate the task-constraint parameters $\hat{\beta}_{Ab_l}$ for the sub-dataset $\mathcal{X}_l$, by minimizing $\bar{\varepsilon}_{\text{CSE}}(\beta_{Ab_l}; \mathcal{X}_l)$, and then evaluate these estimated parameters using the other sub-dataset $\mathcal{X}_h$, as in

$$\bar{\varepsilon}_{\text{CSE}}(\hat{\beta}_{Ab_l}; \mathcal{X}_h) = \sum_{i=1}^{\mathcal{I}_h} \|A(s_{ih}; \hat{\beta}_{A_l})u_{ih} - b(s_{ih}; \hat{\beta}_{b_l}\|^2, \tag{4.86}$$

where $\mathcal{I}_h$ is the number of data pairs of observations of the hth sub-dataset. For short, we will refer to the error in (4.86) as $\bar{\varepsilon}_{\text{CSE},l,h}$, meaning the CSE of the constraint-task parameters estimated on the lth sub-dataset and evaluated on the hth sub-dataset. The value of $\bar{\varepsilon}_{\text{CSE},l,h}$ will be low for $l = h$ and high otherwise, according to the assumption that each sub-dataset was subjected to different constraints. When different constraints intersect in some region of the space, i.e., the underlying constraints are similar to one another this cost should be low reflecting this constraint similarity.

For the experimental data used in the Subsection 4.4.4, we manually selected the $\mathcal{J}$ sub-datasets, separating the full dataset into the individual sub-datasets. Figure 4.21 shows the Cartesian positions of the KUKA's end-effector for the full dataset (blue) and, overlapping, the corresponding sub-datasets, manually separated (red). We selected the initial and final indices of the data points for each sub-dataset, by visually inspecting the data. However, for larger full datasets this figure might become cluttered, making it difficult to verify even that some demonstrations correspond to very

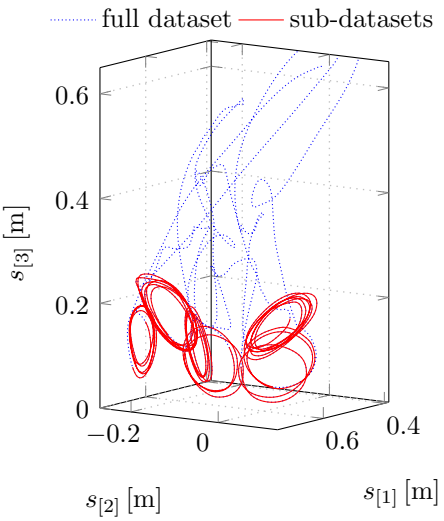


Figure 4.21: KUKA lightweight robotic arm end-effector Cartesian positions for a full unseparated dataset (blue), subject to different constraints in the form of flat surface inclinations. Overlapped is the manually separated sub-datasets, showing that a full unprocessed dataset contains transition regions with data-points that are discarded before the learning process.

similar constraints. This suggests that we could use the CSE to aid the sub-datasets' separation process.

Given an unprocessed full dataset, i.e containing demonstrations and transitions between the demonstrations, we must perform a similarity analysis for groups of data points, regardless of whether they correspond to the same constraint or not. One approach is to select a set of consecutive data points which represent a window within the full dataset. We then compute the parameters for that window l. We shift the window l across the dataset by some increment smaller than the size of the window, creating a window $l+1$ (where size refers to the number of consecutive data points). If the evaluation of the previously estimated parameters on this new window produces a low CSE, then this suggests that the data covered by these two consecutive windows is subject to the same underlying constraint.

By repeating this process for the full dataset, we then obtain a matrix such as the one in Figure 4.22. This matrix corresponds to the data shown in Figure 4.21. We have empirically chosen a window size of 400 samples (corresponding to 8 seconds for a sampling frequency of 50 Hz) and increments of 50 samples (1 second). In Figure 4.22 we also overlap boxes showing the manual separation. Note that using Figure 4.22 alone would lead to confusing at least two groups of windows (around indices 120 and 150) with demonstrations, given that they produce squares of low CSE in the matrix. Figure 4.22 indicates that the data belonging to those two groups is consistent with some constraint, which is sufficiently well modelled by the chosen combination of

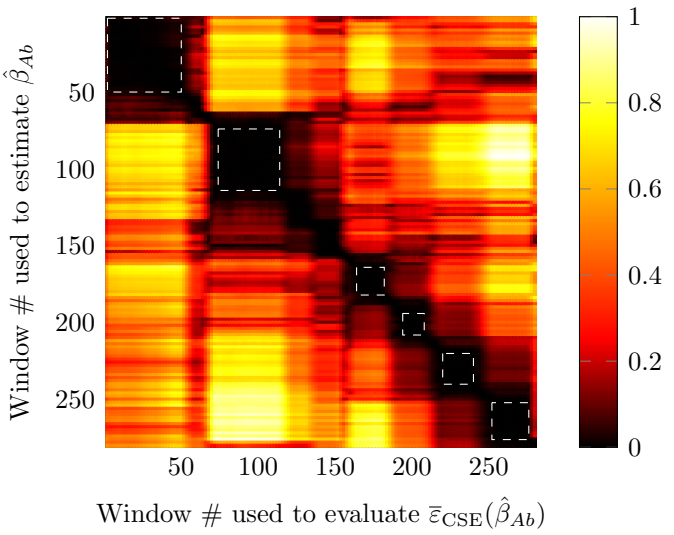


Figure 4.22: Normalized $\bar{\varepsilon}_{\mathrm{CSE},l,h}$ cost for the window h using the estimated parameters from the window l. Each window contains 400 consecutive data points from the full unseparated dataset, differing from the preceding window by 50 data points.

feature functions. For instance, if those samples correspond to a moment in time where the robot was static while changing the flat table orientation between demonstrations, then it makes sense to say that those data points are consistent with the same constraint, e.g. the same configuration of the robot. Therefore, we conclude that we must combine this metric with other application specific metrics. We might remove data points where the robot is static using pre-processing if necessary and, alternatively, we can use a tactile sensor to detect when the end tool is in contact with the surface, as in [156].

4.6 Discussion

This chapter presented a new method for learning, from demonstration, policies that lie in the null-space of a primary task, i.e. subject to some constraint. We introduce the term "Constraint-aware Policy Learning" as the reformulation of the direct policy learning method, where the policy appropriately parametrizes the constraint and the primary control task. Additionally, we discuss the conditions for which this "Constraint-aware Policy Learning" splits into two optimization problems — constraint estimation followed by unconstrained policy estimation — and propose different methods for estimating the constraint depending on those conditions.

The main advantage of this approach, compared to classic direct policy learning, is its ability to learn a policy consistent with the constraint. To demonstrate this point, we used different tasks and constraints in our experimental demonstration with the

real KUKA lightweight arm. In this case, while recording the training data, the human performs and demonstrates the task whereas, in the validation stage, we use a force-based task to adapt and align the tool to an unknown surface.

While we can use Locally Weighted Models to parameterize unconstrained policies, as discussed in this chapter, or any other more generic functions, in the example of learning a wiping motion, we choose to take advantage of our domain knowledge of this specific task, by incorporating more specialised feature functions. This decreased the number of parameters the algorithm has to learn, decreasing the required number of demonstrations. Certainly, a clever choice of feature functions can — as in our case — greatly improve the results, or even turn the learning exercise into a trivial problem. However, what this framework provides is a way of encapsulating all the specifics and domain knowledge in the chosen feature functions, rather than in the learning algorithm itself.

In order to learn a generalisable unconstrained policy, we must somehow guarantee that the training datasets provide enough variety of constraints. We provide a means of comparing the datasets regarding their underlying constraint by using the same metric used in the constraint estimation. This consists in building a similarity matrix by computing the estimation residual of a sub-dataset, using the estimated parameters from the other sub-datasets. Furthermore, besides allowing us to identify similar constraints between different sub-datasets, this similarity matrix allows us to identify different constraints within the same dataset, by running the same metric but over windows of data. This can be a valuable tool for helping identifying the beginning and end of a demonstration.

The approaches presented for learning both the tasks/constraints and the unconstrained policy easily extend to higher dimensions, as the methods presented are numerically efficient. Additionally, the methods easily extend to different applications, whether by providing generic feature functions, or feature functions specific to the intended application. The hierarchical split of tasks, where higher priority tasks are unaffected by the lower priority tasks by means of the constraint and null-space projection, is a powerful concept which has been successfully applied in the resolution of many problems, with works even learning the priority of those tasks from data [60]. Being able to learn the tasks themselves from demonstrations, in a simple and concise framework, is something that can be potentially very useful in many of those problems.

One key ingredient to the CaPL approach is the efficient way, based on either SVD or GSVD, for learning the rigid or task-based constraints. One common limitation of all methods on constraint learning, including ours and other methods on learning task manifolds for constrained manipulation [86], is the requirement to provide feature functions for the constraint matrix or constraint manifold. This implies that we know the space where the constraint lives as prior knowledge. It would be interesting to consider cases where this prior knowledge is nonexistent and how, in these

situations, learning the constraint would compare with the CCL approach, regarding the improvement of the estimation of the unconstrained policy.

The work presented here builds on more than a decade of research on learning generalizable policies from constrained observations. The most well known learning method being the Constraint Consistent Learning (CCL) from Howard et al. [67], but much work has followed both on the learning of unconstrained policies and the constraints themselves. This chapter reviews the main relevant methods in this literature and compares the DPL, CCL and our CaPL on two examples, for ease of understanding and discussion. For the two dimensional example, we also compared the use of local versus global LS methods for learning Locally Weighted Models, showing that for this type of low dimensional example the global LS significantly outperforms the local LS, without compromising much on the computational cost. However, previous publications, including our own, tend to often use the local LS instead, perhaps influenced by the incremental learning literature which, due to the lack of prior knowledge of the policy model complexity — i.e. the number of local models and variance of the receptive fields — favors using local LS methods that greatly reduce the negative interference of each local model on the other ones while learning [166]. Finally, this chapter complements the current state of the art literature with a theoretical unpublished analysis on methods for estimating constraints and task controllers, through various numerical matrix decompositions, such as EVD, SVD, GEVD and GSVD.

Chapter 5

Conclusions

"A bend in the road is not the end of the road... Unless you fail to make the turn."

Helen Keller

This chapter summarizes the main contributions of the thesis and discusses the limitations of some of the results, highlighting the open questions and future research avenues.

5.1 Summary and Contributions

Realizing motions while in contact with the environment remains an intriguing challenge in many of the robotic applications. In face of such challenge, many works — especially in the manipulation domain — focus on avoiding those contact interactions altogether, by developing various obstacle avoidance approaches [96, 112, 123]. However, the very nature of some other domains — such as legged locomotion — makes tackling this challenge unavoidable, which perhaps is a key factor why it remains such a challenging domain in robotics. Walking robots, typically, need to account for the making, maintaining and breaking of fixed position contacts, having to deal with the decision of particular location and time of contact [122, 138]. However, that still excludes the challenges regarding more dynamic contacts, such as establishing and maintaining contacts with moving objects and sliding contacts. Sliding contacts, in particular, constitutes an interesting robotics problem, because it immediately reveals its dual nature, connecting motion interaction (constrained motion) and force interaction (contact maintenance). Albeit addressed by some studies from both a force control point of view [72] and a constrained motion point of view [118], questions about how to exploit the nature of this type of contact interaction in modelling, control and learning definitely deserve more exploration.

This thesis delves specifically in the exploration of this type of constrained motions.

In particular, it studies the class of Task-based Constraints (TbCs) that can model both task motion constraints and rigid contact constraints. A TbC is an equality constraint which we can express as

$$A(\cdot)u = b(\cdot), \tag{5.1}$$

where the control actions and task-space component become, respectively, $u = \dot{q}$ and $b = \dot{x}(t)$ for a kinematic problem and $u = \ddot{q}$ and $b = \ddot{x}(t) - \dot{A}(q)\dot{q}$ for a dynamic problem. Chapter 2 uses the TbC abstraction to prove the equivalence between the forward dynamics' models derived from the Gauss Principle of Least Constraint, the Operational Space Formulation (OSF) and the Projected Dynamics (PD) approach. We obtain such results by also reformulating the PD approach, originally proposed by Aghili [1], and generalizing the dynamically consistent inverse Jacobian concept, defined by Khatib [77], to a rank deficient constraint Jacobian matrix A. Chapter 3 uses the multi TbC abstraction to show the equivalence between the operational space controllers with rigid constraints, separately proposed by De Sapio and Khatib [34] and Mistry and Righetti [101], highlighting the relation between these controllers with the selection matrix approach for hybrid position/force control [77, 99]. We also validated the selection matrix approach for simultaneous position and force control in the task of wiping curved surfaces of unknown geometry — sliding constraints — using velocity controlled robots (kinematic problem), motivated by the industrial application of cleaning the front panels of the train cabs. Finally, Chapter 4 presents a Constraint-aware Policy Learning (CaPL) method for learning control actions u subject to TbCs (5.1), which is based on a two step optimization process of estimating the constraint matrix A and task controller b for each sub-dataset, via closed-form solutions based on SVD or GSVD methods, and estimating the unconstrained policy ${}^{u}\boldsymbol{\pi}$ that is consistent with a set of different TbC's of the form (5.1). We evaluated our learning framework both on low dimensional examples, for the clarity of the analysis, and an experiment with a seven DoF robotic arm. While the low dimensional examples show the benefit of explicitly estimating the constraints when learning unconstrained policies, in comparison to a DPL method or even to CCL [67], the robot experiment shows that our closed-form solutions for estimating constraints scale well to higher dimensional systems, unlike previous sampling-based methods for estimating constraints [89].

The organization of this thesis reflects the study of a simple hypothesis in three crucial domains of robotics. The simple hypothesis is if a Task-based Constraint abstraction represents a useful mechanism of decoupling the robotic motion control policies into simpler motions, bringing us better understanding, ease of implementation and, finally, generalization capabilities across different environments. The three main domains in which we explored this question was in the modelling of the *dynamics* of robotic systems — Chapter 2 — in the simultaneous position and force *control* of robots — Chapter 3 — and, finally, in the *learning* of generalizable policies — Chapter 4. These three domains: dynamics, control and learning, are important pillars of the robotics research. Another of such pillars is trajectory optimization.

Recent work has started to address the incorporation of the dynamics of constrained multi-body system in trajectory optimization algorithms [122, 134] — in the context of legged locomotion. The question remains, will the explicit incorporation of Task-based Constraint (TbC) help advance trajectory optimization methods, by making them more stable, simpler, faster and more transferable to real robots. In this thesis I advocate for the incorporation of constraints in the modelling, control and learning of robotics' motions. One can now wonder in different ways of extending these ideas to other domains of robotics, such as planning/trajectory optimization.

5.2 Discussion and Future Directions

This research gave us new insights on current control methods and led to new learning methods for robotic motions subject to constraints. However, there are many questions left to answer. For instance, we found out the equivalence between thought to be distinct modeling and control approaches for constrained robotic motions, but we still lack thorough experimental comparison results accross them, in particular regarding to their numerical properties. Also from an experimental point of view, the task of wiping the curved surface required exhaustive parameter tuning, that we would like to mitigate by estimation of certain contact interaction properties, such as frictional contact [50] and contact impedance [44]. From a control perspective, many methods are shifting towards Quadratic Programming (QP) based approaches [75, 91, 138], which allows the incorporation of more expressive inequality constraints. The result that we can obtain many of the complicated controllers purposely derived for simultaneous position and force control from a stack of Jacobian matrices, give us an indication on why these recent optimization approaches work well. Indeed, the solution obtained through stacking different Jacobian matrices, corresponding to different tasks (whether they are force or motion tasks), simply corresponds to the solution of a convex optimization problem only using equality constraints and, hence, we can think of the QP approaches as a general way of encoding any other specific task-space controllers [82]. The TbC abstraction also allows for some hierarchical decomposition, by introducing a null-space component that always gives precedence to the task-space component. Recent works [37, 92, 95] developed quite sophisticated methods for hierarchically controlling more than two tasks. Such hierarchical structures even include inequality constraints [48, 49]. However, when systems become more complex, they also become more difficult to understand and less explainable. For example, Dietrich et al. [43] recently presented a stability analysis for a hierarchical OSF (where task motions essentially correspond to equality constraints). For more complex systems with inequality constraints, it is unclear how to obtain such stability analysis. Another use of the hierarchical decomposition of TbC, is in learning generalizable control policies, when learning from constrained observations. However, it is unclear if there are other latent space encodings and approaches, as in [22], for learning such latent spaces that might achieve the same or

better generalization capabilities. All these questions arise from our search for better performing but also more explainable control and learning methods for manipulation motions in contact.

Appendix A

Contributed Proofs/Results

This appendix provides the proofs/results that constitute novel contributions of this thesis.

A.1 Constrained Inertia Matrix with Minimum Condition Number

The goal here is to find a matrix $R^{(*)}$ such that the condition number κ of $M_c = PM + R(I_n - P)$ is minimal. By definition the condition number of a square matrix C is

$$\kappa(C) \triangleq \|C^{-1}\| \cdot \|C\|, \tag{A.1}$$

for any consistent norm [59].

Proof. From definition (A.1), we can verify that the minimum possible condition number is 1, which only happens if C is a scalar multiple of a linear isometry, i.e. a distance preserving transformation. In the Euclidean space, such transformation is given by an orthogonal matrix Q. Therefore, if we can find R such that $M_c = \mu Q$, where $\mu \in \mathbb{R}_{\neq 0}$, then that is the minimum possible condition number we can hope for. By equating M_c and μQ, we obtain

$$M_c = \mu Q \Leftrightarrow R(I_n - P) = \mu Q - PM. \tag{A.2}$$

Given that $(I_n - P)$ is non invertible, it means the equality in (A.2) is false. However, if we post-multiply $(I_n - P)$ by both sides of (A.2), we get

$$R(I_n - P) = (\mu Q - PM)(I_n - P), \tag{A.3}$$

which is true for $R = (\mu Q - PM)$, and leads to the closest result of the approximation $M_c \approx \mu Q$. The resulting constraint inertia matrix is $M_c = \mu Q(I - P) + PMP$.

To keep generality, we can still consider Q to be any square full rank matrix, thus all we have achieved so far is a rewritten M_c in terms of Q and μ instead of R. We shall now find a Q and μ that minimize $\kappa(M_c)$. Given that $(I_n - P)$ is an orthogonal projection matrix, we can always find a partial isometry Z_1 such that $(I_n - P) = Z_1 Z_1^\top$, and analogously $P = Z_2 Z_2^\top$, where $Q_p = \begin{bmatrix} Z_1 & Z_2 \end{bmatrix}$ is an orthogonal matrix. We can then rewrite M_c as

$$\begin{aligned} M_c &= \mu Q(I - P) + PMP = \mu Q Z_1 Z_1^\top + Z_2 Z_2^\top M Z_2 Z_2^\top \\ &= \underbrace{\begin{bmatrix} QZ_1 & Z_2 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} \mu I_{n-m} & 0 \\ 0 & Z_2^\top M Z_2 \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} Z_1^\top \\ Z_2^\top \end{bmatrix}}_{Q_p^\top}. \end{aligned} \tag{A.4}$$

We have that for any suitable norm

$$\|M_c\| = \|BXQ_p^\top\| = \|BX\|,$$

$$\|M_c^{-1}\| = \|Q_p X^{-1} B^{-1}\| = \|X^{-1}B^{-1}\|,$$

and replacing the previous results in (A.1), we obtain

$$\begin{aligned} \kappa(M_c) &= \|M_c^{-1}\| \cdot \|M_c\| = \|X^{-1}B^{-1}\| \cdot \|BX\| \\ &\leq \|X^{-1}\| \cdot \|X\| \cdot \|B^{-1}\| \cdot \|B\| \\ &= \kappa(X)\kappa(B). \end{aligned} \tag{A.5}$$

As B only depends on Q and X only depends on μ, we can independently find Q and μ that minimize the respective $\kappa(B)$ and $\kappa(X)$. By inspection of (A.4) we see that B is orthogonal if $Q = I_n$, in which case the inequality in (A.5) becomes an equality. Therefore, we obtain that the result that minimizes $\kappa(M_c)$ is $R^{(*)} = (\mu I_n - PM)$, for which

$$M_c^{(*)} = \mu(I_n - P) + PMP. \tag{A.6}$$

Aghili [3, 4, 5] more recently proposes an M_c in the form of Equation (A.6) and proves using a 2-norm that its condition number is minimum for $\{\varsigma_{\min}(PMP) \neq 0\} \leq \mu \leq \varsigma_{\max}(PMP)$, where ς represents the singular values of a given matrix. We have that $\{\varsigma_{\min}(PMP) \neq 0\} = \varsigma_{\min}(Z_2^\top M Z_2)$ and $\varsigma_{\max}(PMP) = \varsigma_{\max}(Z_2^\top M Z_2)$. If we use a 2-norm in (A.1), then the condition number of a matrix is given by the ratio of its singular values. Therefore, by inspection of (A.4), we see that the minimum $\kappa(M_c^{(*)})$ is $\kappa(Z_2^\top M Z_2)$. ■

A.2 Projected Forward Dynamics Equivalence

The goal is to prove that $\overline{A} = M_c^{-1} R A^\dagger$ and $P_M M^{-1} = M_c^{-1} P$ for any $R \in \mathbb{R}^{n \times n}$ such that $M_c = PM + R(I_n - P)$ is full rank, with $M \in \mathbb{S}_n^+$.

Proof. Using the MP-conditions that apply to $A^\dagger$: (i) $AA^\dagger A = A$ (ii) $A^\dagger A A^\dagger = A^\dagger$ (iii) $AA^\dagger = \left(AA^\dagger\right)^\top$ (iv) $A^\dagger A = \left(A^\dagger A\right)^\top$, and the inertia-weighted generalized inverse conditions that apply to $\overline{A}$: (a) $A\overline{A}A = A$ (b) $\overline{A}A\overline{A} = \overline{A}$ (c) $A\overline{A} = \left(A\overline{A}\right)^\top$ (d) $M\overline{A}A = \left(M\overline{A}A\right)^\top$, we start by showing the intermediary results

$$
\begin{aligned}
P M \overline{A} &= (I_n - A^\dagger A) M \overline{A} = M\overline{A} - A^\dagger A M \overline{A} \\
&\overset{\text{(b)}}{=} M\overline{A} - A^\dagger A M (\overline{A} A \overline{A}) \\
&\overset{\text{(iv)}}{=} M\overline{A} - (A^\dagger A)^\top (M\overline{A}A)^\top \overline{A} \\
&\overset{\text{(d)}}{=} M\overline{A} - (A^\dagger A)(M\overline{A}A)^\top \overline{A} = M\overline{A} - \left(\overline{A}^\top M \overline{A} (A A^\dagger A)\right)^\top \\
&\overset{\text{(i)}}{=} M\overline{A} - \left(\overline{A}^\top (M\overline{A}A)\right)^\top \\
&\overset{\text{(d)}}{=} M\overline{A} - \left(\overline{A}^\top (M\overline{A}A)^\top\right)^\top = M\overline{A} - M(\overline{A}A\overline{A}) \\
&\overset{\text{(b)}}{=} M\overline{A} - M\overline{A} = 0,
\end{aligned}
$$

and

$$
\begin{aligned}
(I_n - P)\overline{A} &= A^\dagger A \overline{A} \\
&\overset{\text{(ii)}}{=} (A^\dagger A A^\dagger) A \overline{A} \\
&\overset{\text{(c)}}{=} A^\dagger (A A^\dagger)(A\overline{A})^\top \\
&\overset{\text{(iii)}}{=} A^\dagger (A A^\dagger)^\top (A\overline{A})^\top = A^\dagger A^{\dagger\top} (A\overline{A}A)^\top \\
&\overset{\text{(a)}}{=} A^\dagger A^{\dagger\top} A^\top = A^\dagger (A A^\dagger)^\top \\
&\overset{\text{(iii)}}{=} A^\dagger A A^\dagger \overset{\text{(ii)}}{=} A^\dagger.
\end{aligned}
$$

For M_c invertible $\overline{A} = M_c^{-1} R A^\dagger \Leftrightarrow M_c \overline{A} = R A^\dagger$, and then we can show that

$$\begin{aligned} M_c \overline{A} &= (PM + R(I_n - P))\,\overline{A} \\ &= \underbrace{PM\overline{A}}_{=0} + R\,\underbrace{(I_n - P)\overline{A}}_{=A^\dagger} \\ &= R A^\dagger. \end{aligned}$$

Analogously, for $P_M M^{-1} = M_c^{-1} P \Leftrightarrow M_c P_M M^{-1} = P$, for which we can show that

$$\begin{aligned} M_c P_M M^{-1} &= (PM + R(I_n - P))\left(I_n - \overline{A}A\right) M^{-1} \\ &= \left(PM - \underbrace{PM\overline{A}}_{=0} A + R\,\underbrace{(I_n - P)}_{=A^\dagger A} - R\,\underbrace{(I_n - P)\overline{A}}_{=A^\dagger} A \right) M^{-1} \\ &= \left(PM + RA^\dagger A - RA^\dagger A\right) M^{-1} \\ &= P\,\underbrace{MM^{-1}}_{=I_n} = P. \end{aligned}$$

■

A.3 Singular Dynamically Consistent Jacobian

The goal is to prove that the inertia-weighted generalized inverse $\overline{A}$ of the rank deficient matrix A satisfies the condition (2.19), i.e.

$$AM^{-1}\left(I_{n_q} - A^\top \overline{A}^\top\right)\tau_\varepsilon = 0, \tag{A.7}$$

and, therefore, it is a dynamically consistent inverse.

Proof. Let $\overline{A}$ be the inertia-weighted generalized inverse of the rank deficient matrix A, satisfying the conditions: (a) $A\overline{A}A = A$ (b) $\overline{A}A\overline{A} = \overline{A}$ (c) $A\overline{A} = \left(A\overline{A}\right)^\top$ (d) $M\overline{A}A = \left(M\overline{A}A\right)^\top$, where (d) is equivalent to (e) $\overline{A}AM^{-1} = (\overline{A}AM^{-1})^\top$, given M symmetric positive definite [15] (Appendix B.5 shows the same equivalence

result in (B.34)). We can then show that

$$\begin{aligned} AM^{-1}(I_n - A^\top \overline{A}^\top)\tau_\varepsilon &= (AM^{-1} - A(\overline{A}AM^{-1})^\top)\tau_\varepsilon \\ &\overset{(e)}{=} (AM^{-1} - (A\overline{A}A)M^{-1})\tau_\varepsilon \\ &\overset{(a)}{=} (AM^{-1} - AM^{-1})\tau_\varepsilon \\ &= 0, \end{aligned}$$

for any $\tau_\varepsilon \in \mathbb{R}^n$, proving that $\overline{A}$ is a dynamically consistent inverse of the Jacobian matrix A. ■

A.4 Partitioned Task-space Inertia Matrix

The goal is to show that for a partitioned constraint Jacobian

$$\mathbb{R}^{m\times n} \ni A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, \tag{A.8}$$

if $\text{rank}(A) = \text{rank}(A_1) + \text{rank}(A_2)$, then we can write the task-inertia matrix

$$M_x \triangleq (AM^{-1}A^\top)^\dagger, \tag{A.9}$$

as

$$M_x = \begin{bmatrix} M_1 & -\overline{A}_1^\top A_2^\top M_2 \\ -\overline{A}_2^\top A_1^\top M_1 & M_2 \end{bmatrix} \tag{A.10}$$

$$= \begin{bmatrix} M_1 & -M_1 A_1 \overline{A}_2 \\ -M_2 A_2 \overline{A}_1 & M_2 \end{bmatrix}, \tag{A.11}$$

where

$$M_1 \triangleq \left(A_1 P_{M2} M^{-1} A_1^\top\right)^\dagger$$

$$M_2 \triangleq \left(A_2 P_{M1} M^{-1} A_2^\top\right)^\dagger,$$

with $\overline{A}_1$ and $\overline{A}_2$ being, respectively, the inertia-weighted generalized inverse of A_1 and A_2, and P_{M1} and P_{M2} being the respective projection matrices.

Proof. Let's start by defining the positive semi-definite matrix

$$\mathbb{S}_+^n \ni H \triangleq AM^{-1}A^\top = \begin{bmatrix} H_{11} & H_{12} \\ H_{12}^\top & H_{22} \end{bmatrix} = \begin{bmatrix} A_1M^{-1}A_1^\top & A_1M^{-1}A_2^\top \\ A_2M^{-1}A_1^\top & A_2M^{-1}A_2^\top \end{bmatrix}. \tag{A.12}$$

Our goal is then to obtain $H^\dagger = M_x$. We have that the for $\text{rank}(H) = \text{rank}(H_{11}) + \text{rank}(H_{22})$, the Moore-Penrose inverse of H is

$$H^\dagger = \begin{bmatrix} H_{11}^\dagger + H_{11}^\dagger H_{12}\Sigma_{22}^\dagger H_{12}^\top H_{11}^\dagger & -H_{11}^\dagger H_{12}\Sigma_{22}^\dagger \\ -\Sigma_{22}^\dagger H_{12}^\top H_{11}^\dagger & \Sigma_{22}^\dagger \end{bmatrix} \tag{A.13}$$

$$= \begin{bmatrix} \Sigma_{11}^\dagger & -\Sigma_{11}^\dagger H_{12}H_{22}^\dagger \\ -H_{22}^\dagger H_{12}^\top \Sigma_{11}^\dagger & H_{22}^\dagger + H_{22}^\dagger H_{12}^\top \Sigma_{11}^\dagger H_{12}H_{22} \end{bmatrix}, \tag{A.14}$$

where $\Sigma_{11} = H_{11} - H_{12}H_{22}^\dagger H_{12}^\top$ and $\Sigma_{22} = H_{22} - H_{12}^\top H_{11}^\dagger H_{12}$ [141, 160, 173].

Given that $M \in \mathbb{S}_{++}^n$, by inspection of (A.12) we immediately see that $\text{rank}(H) = \text{rank}(A)$, $\text{rank}(H_{11}) = \text{rank}(A_1)$ and $\text{rank}(H_{22}) = \text{rank}(A_2)$ and, hence,

$$\text{rank}(H) = \text{rank}(H_{11}) + \text{rank}(H_{22}) \Leftrightarrow \text{rank}(A) = \text{rank}(A_1) + \text{rank}(A_2). \tag{A.15}$$

All it's left is to compute the block elements of (A.13) and (A.14), using the equalities in (A.12) as follows

$$\begin{aligned} \Sigma_{11}^\dagger &= \left(H_{11} - H_{12}H_{22}^\dagger H_{12}^\top\right)^\dagger \\ &= \left(A_1M^{-1}A_1^\top - A_1 \underbrace{M^{-1}A_2^\top (A_2M^{-1}A_2^\top)^\dagger}_{\triangleq \overline{A}_2} A_2M^{-1}A_1^\top\right)^\dagger \\ &= \left(A_1 \underbrace{\left(I_n - \overline{A}_2A_2\right)}_{\triangleq P_{M2}} M^{-1}A_1^\top\right)^\dagger = \left(A_1P_{M2}M^{-1}A_1^\top\right)^\dagger \triangleq M_1, \end{aligned}$$

$$\begin{aligned} \Sigma_{22}^\dagger &= \left(H_{22} - H_{12}^\top H_{11}^\dagger H_{12}\right)^\dagger \\ &= \left(A_2M^{-1}A_2^\top - A_2 \underbrace{M^{-1}A_1^\top (A_1M^{-1}A_1^\top)^\dagger}_{\triangleq \overline{A}_1} A_1M^{-1}A_2^\top\right)^\dagger \\ &= \left(A_2 \underbrace{\left(I_n - \overline{A}_1A_1\right)}_{\triangleq P_{M1}} M^{-1}A_2^\top\right)^\dagger = \left(A_2P_{M1}M^{-1}A_2^\top\right)^\dagger \triangleq M_2, \end{aligned}$$

$$-\Sigma_{11}^{\dagger} H_{12} H_{22}^{\dagger} = -M_1 A_1 \underbrace{M^{-1} A_2^{\top} (A_2 M^{-1} A_2^{\top})^{\dagger}}_{=\overline{A}_2} = -M_1 A_1 \overline{A}_2,$$

$$-\Sigma_{22}^{\dagger} H_{12}^{\top} H_{11}^{\dagger} = -M_2 A_2 \underbrace{M^{-1} A_1^{\top} (A_1 M^{-1} A_1^{\top})^{\dagger}}_{=\overline{A}_1} = -M_2 A_2 \overline{A}_1.$$

■

A.5 Direct Policy Error Decomposition

Let the Constraint Space Error (CSE) be

$$\varepsilon_{\text{CSE}}(u; A, b) \triangleq \int_0^T \|A(t)u(t) - b(t)\|^2 dt, \tag{A.16}$$

where $A \in \mathbb{R}^{m \times n}$, and the Constrained Policy Error (CPE) be

$$\varepsilon_{\text{CPE}}(u; P, {}^{\text{u}}\boldsymbol{\pi}) \triangleq \int_0^T \|P(t)(u(t) - {}^{\text{u}}\boldsymbol{\pi}(t))\|^2 dt, \tag{A.17}$$

where $P = (I_n - A^{\dagger}A) \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, and the Direct Policy Learning (DPL) be

$$\varepsilon_{\text{DPL}}(u; \boldsymbol{\pi}) \triangleq \int_0^T \|u(t) - \boldsymbol{\pi}(t)\|^2 dt. \tag{A.18}$$

We aim to prove that for a policy $\boldsymbol{\pi}$ defined as

$$\boldsymbol{\pi}(t) \triangleq A(t)^{\dagger} b(t) + P(t)\, {}^{\text{u}}\boldsymbol{\pi}(t), \tag{A.19}$$

then we can decompose the rewriten DPE

$$\varepsilon_{\text{DPL}}(u; A, b, {}^{\text{u}}\boldsymbol{\pi}) \triangleq \int_0^T \|u(t) - (A(t)^{\dagger} b(t) + P(t)\, {}^{\text{u}}\boldsymbol{\pi}(t))\|^2 dt, \tag{A.20}$$

as

$$\varepsilon_{\text{DPL}}(u; A, b, {}^{\text{u}}\boldsymbol{\pi}) = \varepsilon_{\text{CSE}}(u; A, b) + \varepsilon_{\text{CPE}}(u; P, {}^{\text{u}}\boldsymbol{\pi}) \tag{A.21}$$

for A semi-orthogonal.

Proof. Let's start by summing (A.16) and (A.17), obtaining

$$\varepsilon_{\text{CSE}}(u; A, b) + \varepsilon_{\text{CPE}}(u; P, {}^{\text{u}}\boldsymbol{\pi}) \tag{A.22}$$

$$= \int_0^T \|A(t)u(t) - b(t)\|^2 dt + \int_0^T \|P(t)(u(t) - {}^{\text{u}}\boldsymbol{\pi}(t))\|^2 dt \tag{A.23}$$

$$= \int_0^T \|A(t)u(t) - b(t)\|^2 + \|P(t)(u(t) - {}^{\text{u}}\boldsymbol{\pi}(t))\|^2 dt, \tag{A.24}$$

which is equal to (A.20) if

$$\|u(t)-(A(t)^{\dagger}b(t)+P(t)\,{}^{\mathrm{u}}\boldsymbol{\pi}(t))\|^2 = \|A(t)u(t)-b(t)\|^2+\|P(t)(u(t)-{}^{\mathrm{u}}\boldsymbol{\pi}(t))\|^2, \quad \text{(A.25)}$$

$\forall t \in [0, T]$.

Let

$$Q = \begin{bmatrix} A \\ \mathcal{N}(A)^{\top} \end{bmatrix} \quad \text{(A.26)}$$

be a square matrix, where $\mathcal{N}(A)^{\top}$ is the transpose of an orthogonal basis of the right null space of A, satisfying the condition $A\mathcal{N}(A) = 0$, i.e., the rows of $\mathcal{N}(A)^{\top}$ have unitary Euclidean norm and are orthogonal to the rows of A, where here we drop the time t dependence for readability purposes.

If A is a semi-orthogonal matrix then T is an orthogonal matrix, resulting in

$$\|\alpha\|^2 = \|Q\alpha\|^2 \quad \text{(A.27)}$$

$$= \left\| \begin{bmatrix} A\alpha \\ \mathcal{N}(A)^{\top}\alpha \end{bmatrix} \right\|^2 \quad \text{(A.28)}$$

$$= \begin{bmatrix} A\alpha \\ \mathcal{N}(A)^{\top}\alpha \end{bmatrix}^{\top} \begin{bmatrix} A\alpha \\ \mathcal{N}(A)^{\top}\alpha \end{bmatrix} \quad \text{(A.29)}$$

$$= \alpha^{\top}A^{\top}A\alpha + \alpha^{\top}\mathcal{N}(A)\mathcal{N}(A)^{\top}\alpha \quad \text{(A.30)}$$

$$= \|A\alpha\|^2 + \|\mathcal{N}(A)^{\top}\alpha\|^2. \quad \text{(A.31)}$$

Let

$$\alpha = u - A^{\dagger}b - P\,{}^{\mathrm{u}}\boldsymbol{\pi} \quad \text{(A.32)}$$

where $A^{\dagger} = A^{\top}$, for A semi-orthogonal, and $P = \mathcal{N}(A)\mathcal{N}(A)^{\top}$, for any choice of null space base $\mathcal{N}(A)$, then the first term of (A.31) becomes

$$\|A\alpha\|^2 = \|Au - AA^{\dagger}b - AP\,{}^{\mathrm{u}}\boldsymbol{\pi}\|^2 \quad \text{(A.33)}$$

$$= \|Au - b\|^2, \quad \text{(A.34)}$$

where we used $AA^{\dagger} = I$ and $AP = 0$. The second term of (A.31) becomes

$$\|\mathcal{N}(A)^{\top}\alpha\|^2 = \|\mathcal{N}(A)^{\top}u - \mathcal{N}(A)^{\top}A^{\dagger}b - \mathcal{N}(A)^{\top}P\,{}^{\mathrm{u}}\boldsymbol{\pi}\|^2 \quad \text{(A.35)}$$

$$= \|\mathcal{N}(A)^{\top}(u - {}^{\mathrm{u}}\boldsymbol{\pi})\|^2, \qquad \mathcal{N}(A)^{\top}A^{\top} = 0 \quad \text{(A.36)}$$

$$= \|\mathcal{N}(A)\mathcal{N}(A)^{\top}(u - {}^{\mathrm{u}}\boldsymbol{\pi})\|^2 \quad \text{(A.37)}$$

$$= \|N(u - {}^{\mathrm{u}}\boldsymbol{\pi})\|^2. \quad \text{(A.38)}$$

By replacing (A.34) and (A.38) in (A.31), we prove that

$$\begin{aligned}\|\alpha(t)\|^2 &= \|u(t) - A(t)^\dagger b(t) - P(t)\,{}^{\mathrm{u}}\boldsymbol{\pi}(t)\|^2 \\ &= \|A(t)u(t) - b(t)\|^2 + \|P(t)\,(u(t) - {}^{\mathrm{u}}\boldsymbol{\pi}(t))\|^2\end{aligned}$$

for $A(t)$ semi-orthogonal matrix and $\forall t$.

■

A.6 Estimating parameters with GEVD

We aim to show that the transpose of the matrix of right generalized eigenvectors of the pencil $(\mathcal{Y}\mathcal{Y}^\top, \mathcal{Z}\mathcal{Z}^\top)$ corresponding to its smallest generalized eigenvalues, with $\mathcal{Z}\mathcal{Z}^\top$ positive definite, is a solution to the optimization problem

$$\begin{aligned}\mathbb{R}^{m\times n} \ni \hat{\beta} = \arg\min_{\beta} \quad & \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2. \\ s.t. \quad & \beta\left(\frac{1}{\mathcal{K}}\sum_{k=1}^{\mathcal{K}} z_k {z_k}^\top\right)\beta^\top = I_m,\end{aligned} \tag{A.39}$$

where

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Y} = \left[y_1, \ldots, y_{\mathcal{K}}\right]. \tag{A.40}$$

and

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Z} = \frac{1}{\sqrt{\mathcal{K}}}\left[z_1, \ldots, z_{\mathcal{K}}\right]. \tag{A.41}$$

Proof. Let's start by rewriting the cost in (A.39) as

$$\begin{aligned}\sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2 &= \sum_{k=1}^{\mathcal{K}} y_k^\top \beta^\top \beta y_k^\top \\ &\overset{(a)}{=} \mathrm{Tr}(\mathcal{Y}^\top \beta^\top \beta \mathcal{Y}) \\ &\overset{(b)}{=} \mathrm{Tr}(\beta \mathcal{Y}\mathcal{Y}^\top \beta^\top) \\ &\overset{(c)}{=} \|\beta\mathcal{Y}\|_F^2,\end{aligned} \tag{A.42}$$

where we get (a) by inspection; (b) using the commutativity property of the trace operator [175]; and in (c) $\|A\|_F$ represents the Frobenius norm of A [175]. By

rewriting the equality constraint in (A.39) as $\beta\mathcal{Z}\mathcal{Z}^\top\beta = I_m$, we can then rewrite the full optimization (A.39) as

$$\begin{aligned}\hat{\beta} = \arg\min_{\beta} \quad & \mathrm{Tr}(\beta\mathcal{Y}\mathcal{Y}^\top\beta^\top) \\ s.t. \quad & \beta\mathcal{Z}\mathcal{Z}^\top\beta^\top = I_m.\end{aligned} \tag{A.43}$$

The Lagrangian (Boyd and Vandenberghe [27]) for (A.39) is

$$\mathcal{L} = \mathrm{Tr}(\beta\mathcal{Y}\mathcal{Y}^\top\beta^\top) - \mathrm{Tr}(\Lambda^\top(\beta\mathcal{Z}\mathcal{Z}^\top\beta^\top - I_n)), \tag{A.44}$$

where $\Lambda \in \mathbb{R}^{m\times m}$ is a diagonal matrix whose entries are the Lagrange multipliers. Equating the partial derivatives of $\mathcal{L}$ [130] to zero give us

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\beta} =& 2\beta\mathcal{Y}\mathcal{Y}^\top - 2\Lambda^\top\beta\mathcal{Z}\mathcal{Z}^\top \overset{\text{set}}{=} 0 \\ \implies & \beta\mathcal{Y}\mathcal{Y}^\top = \Lambda^\top\beta\mathcal{Z}\mathcal{Z}^\top \\ \implies & \mathcal{Y}\mathcal{Y}^\top\beta^\top = \mathcal{Z}\mathcal{Z}^\top\beta^\top\Lambda,\end{aligned} \tag{A.45}$$

which corresponds to the generalized eigenvalue problem for the $n \times n$ matrix pair or pencil $(\mathcal{W}_\mathcal{Y}, \mathcal{W}_\mathcal{Y})$ [175], where $\mathcal{W}_\mathcal{Y} = \mathcal{Y}\mathcal{Y}^\top$ and $\mathcal{W}_\mathcal{Z} = \mathcal{Z}\mathcal{Z}^\top$ are both square symmetric matrices, and

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\Lambda} =& \beta\mathcal{Z}\mathcal{Z}^\top\beta^\top - I_n \overset{\text{set}}{=} 0 \\ \Leftrightarrow & \beta\mathcal{Z}\mathcal{Z}^\top\beta^\top = I_n.\end{aligned} \tag{A.46}$$

Consider the Generalized Eigenvalue Decomposition (GEVD) of the real symmetric matrix pencil $(\mathcal{W}_\mathcal{Y}, \mathcal{W}_\mathcal{Z})$:

$$\mathcal{W}_\mathcal{Y} U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})} = \mathcal{W}_\mathcal{Z} U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})} D_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})}, \tag{A.47}$$

where $D_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})} \in \mathbb{R}^{n\times n}$ is a diagonal matrix, containing the generalized eigenvalues of the pencil $(\mathcal{W}_\mathcal{Y}, \mathcal{W}_\mathcal{Y})$, and where $U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})} \in \mathbb{R}^{n\times n}$ is a full matrix whose columns contain the corresponding generalized eigenvectors [175]. Let's split the left hand side of (A.47) as

$$\begin{aligned}\mathcal{W}_\mathcal{Y} U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})} &= \mathcal{W}_\mathcal{Y}\begin{bmatrix} U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})1} & U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})2}\end{bmatrix} \\ &= \begin{bmatrix}\mathcal{W}_\mathcal{Y} U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})1} & \mathcal{W}_\mathcal{Y} U_{(\mathcal{W}_\mathcal{Y},\mathcal{W}_\mathcal{Z})2}\end{bmatrix},\end{aligned} \tag{A.48}$$

and the right and side as

$$\begin{aligned}\mathcal{W}_{\mathcal{Z}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})}D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})} &= \mathcal{W}_{\mathcal{Z}}\begin{bmatrix}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})1} & U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}\end{bmatrix}\begin{bmatrix}D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})1} & 0\\ 0 & D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}\end{bmatrix}\\ &= \begin{bmatrix}\mathcal{W}_{\mathcal{Z}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})1}D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})1} & \mathcal{W}_{\mathcal{Z}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}\end{bmatrix},\end{aligned} \tag{A.49}$$

with $D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})1} \in \mathbb{R}^{(n-m)\times(n-m)}$ and $D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2} \in \mathbb{R}^{m\times m}$, being also diagonal matrices.

Then, by inspection of the second terms of (A.48) and (A.49) where

$$\mathcal{W}_{\mathcal{Y}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2} = \mathcal{W}_{\mathcal{Z}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2},$$

we can immediately see that the solution

$$\beta = U^{\top}_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2} \in \mathbb{R}^{n\times m}, \tag{A.50}$$

i.e the transpose of the matrix composed by m generalized eigenvectors of the pencil $(\mathcal{W}_{\mathcal{Y}}, \mathcal{W}_{\mathcal{Z}})$, satisfies the condition (A.45). Furthermore, given

$$U^{\top}_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}\mathcal{W}_{\mathcal{Z}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2} = I_m$$

for $\mathcal{W}_{\mathcal{Z}}$ positive definite [58], then (A.50) also satisfies (A.46) and leads to

$$U^{\top}_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}\mathcal{W}_{\mathcal{Y}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2} = D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}.$$

The remaining question is which selection of generalized eigenvectors minimize the cost (A.42). We, therefore, substitute $\beta = U^{\top}_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2} \in \mathbb{R}^{n\times m}$ in (A.42), obtaining

$$\begin{aligned}\mathrm{Tr}(\beta\mathcal{Y}\mathcal{Y}^{\top}\beta^{\top}) &= \mathrm{Tr}(U^{\top}_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}\mathcal{W}_{\mathcal{Y}}U_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2})\\ &= \mathrm{Tr}(D_{(\mathcal{W}_{\mathcal{Y}},\mathcal{W}_{\mathcal{Z}})2}),\end{aligned} \tag{A.51}$$

which is minimal for $\mathcal{W}_{\mathcal{Y}2}$ containing the smallest generalized eigenvectors of the pencil $(\mathcal{W}_{\mathcal{Y}}, \mathcal{W}_{\mathcal{Z}})$, showing that the transpose of the matrix of right generalized eigenvectors of the pencil $(\mathcal{W}_{\mathcal{Y}}, \mathcal{W}_{\mathcal{Z}})$ corresponding to its smallest generalized eigenvalues, with $\mathcal{W}_{\mathcal{Z}}$ positive definite, is a solution to the optimization problem (A.43).

■

A.7 Estimating parameters with GSVD

We aim to show that

$$\hat{\beta}^{\top} = X^{-1}_{(\mathcal{Y},\mathcal{Z})}S_{I0} \tag{A.52}$$

is a solution o the optimization problem

$$\begin{aligned} \mathbb{R}^{m\times n} \ni \hat{\beta} = \arg\min_{\beta} \quad & \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2. \\ s.t. \quad & \beta \left(\frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} z_k {z_k}^\top \right) \beta^\top = I_m, \end{aligned} \tag{A.53}$$

given that the first m elements of the diagonal of $\Sigma_{\mathcal{Z}}$ are unitary, where $X_{(\mathcal{Y},\mathcal{Z})}$ and $\Sigma_{\mathcal{Z}}$ are the result of the Generalized Singular Value Decomposition (GSVD) of the pencil $(\mathcal{Y}^\top, \mathcal{Z}^\top)$ [58, 175], as

$$\mathcal{Y}^\top = U_{\mathcal{Y}} \Sigma_{\mathcal{Y}} X_{(\mathcal{Y},\mathcal{Z})} \tag{A.54}$$

$$\mathcal{Z}^\top = V_{\mathcal{Z}} \Sigma_{\mathcal{Z}} X_{(\mathcal{Y},\mathcal{Z})}, \tag{A.55}$$

with

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Y} = \left[y_1, \ldots, y_{\mathcal{K}} \right]. \tag{A.56}$$

and

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Z} = \frac{1}{\sqrt{\mathcal{K}}} \left[z_1, \ldots, z_{\mathcal{K}} \right], \tag{A.57}$$

and where $S_{I0}^\top = \begin{bmatrix} I_m & 0_{m\times(n-m)} \end{bmatrix} \in \mathbb{R}^{m\times n}$ is a selection matrix, which extracts the first m columns of $X_{(\mathcal{Y},\mathcal{Z})}^{-1}$.

Proof. Let's start by rewriting the cost in (A.53) as

$$\begin{aligned} \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2 &= \sum_{k=1}^{\mathcal{K}} y_k^\top \beta^\top \beta y_k^\top \\ &\overset{(a)}{=} \mathrm{Tr}(\mathcal{Y}^\top \beta^\top \beta \mathcal{Y}) \\ &\overset{(b)}{=} \|\mathcal{Y}^\top \beta^\top\|_F^2, \end{aligned}$$

where we get (a) by inspection; and in (b) $\|\cdot\|_F$ represents the Frobenius norm [175]. By rewriting the equality constraint in (A.53) as $\beta \mathcal{Z}\mathcal{Z}^\top \beta = I_m$, we can then rewrite the full optimization (A.53) as

$$\begin{aligned} \hat{\beta} = \arg\min_{\beta} \quad & \|\mathcal{Y}^\top \beta^\top\|_F^2 \\ s.t. \quad & \beta \mathcal{Z}\mathcal{Z}^\top \beta^\top = I_m. \end{aligned} \tag{A.58}$$

Let's now consider the Generalized Singular Value Decomposition (GSVD) of the matrix pencil $(\mathcal{Y}^\top, \mathcal{Z}^\top)$

$$\mathcal{Y}^\top = U_\mathcal{Y} \Sigma_\mathcal{Y} X_{(\mathcal{Y},\mathcal{Z})} \tag{A.59}$$

$$\mathcal{Z}^\top = V_\mathcal{Z} \Sigma_\mathcal{Z} X_{(\mathcal{Y},\mathcal{Z})} \tag{A.60}$$

where $U_\mathcal{Y} \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ and $V_\mathcal{Y} \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ are orthogonal matrices, $\Sigma_\mathcal{Y} \in \mathbb{R}^{\mathcal{K}\times n}$ and $\Sigma_\mathcal{Z} \in \mathbb{R}^{\mathcal{K}\times n}$ are rectangular diagonal matrices whose diagonal elements range from 0 to 1 ordered in ascending and descending order, respectively, for which $\Sigma_\mathcal{Y}^\top\Sigma_\mathcal{Y} + \Sigma_\mathcal{Z}^\top\Sigma_\mathcal{Z} = I_n$, and $X_{(\mathcal{Y},\mathcal{Z})} \in \mathbb{R}^{n\times n}$ is an invertible matrix. By replacing (A.59) in the cost of (A.58), we get

$$\begin{aligned}
\|\mathcal{Y}^\top\beta^\top\|_F^2 &= \|U_\mathcal{Y}\Sigma_\mathcal{Y}X_{(\mathcal{Y},\mathcal{Z})}\beta^\top\|_F^2 \\
&\overset{(a)}{=} \|\Sigma_\mathcal{Y}X_{(\mathcal{Y},\mathcal{Z})}\beta^\top\|_F^2 \\
&\overset{(b)}{=} \|\Sigma_\mathcal{Y}\alpha^\top\|_F^2 \\
&= \mathrm{Tr}(\alpha\Sigma_\mathcal{Y}^\top\Sigma_\mathcal{Y}\alpha^\top),
\end{aligned} \tag{A.61}$$

where in (a) the norm is invariant to the orthogonal matrix $U_\mathcal{Y}$; in (b) $\alpha \triangleq \beta X_{(\mathcal{Y},\mathcal{Z})}^\top$, and by replacing (A.60) in the equality condition of (A.58), we get

$$\begin{aligned}
\beta\mathcal{Z}\mathcal{Z}^\top\beta^\top &= \beta X_{(\mathcal{Y},\mathcal{Z})}^\top\Sigma_\mathcal{Z}^\top V_\mathcal{Z}^\top V_\mathcal{Z}\Sigma_\mathcal{Z}X_{(\mathcal{Y},\mathcal{Z})}\beta^\top \\
&= \beta X_{(\mathcal{Y},\mathcal{Z})}^\top\Sigma_\mathcal{Z}^\top\Sigma_\mathcal{Z}X_{(\mathcal{Y},\mathcal{Z})}\beta^\top \\
&= \alpha\Sigma_\mathcal{Z}^\top\Sigma_\mathcal{Z}\alpha^\top.
\end{aligned} \tag{A.62}$$

Replacing (A.61) and (A.62) in (A.58) leads to the following optimization problem

$$\begin{aligned}
\hat{\beta} = \arg\min_\beta \quad & \mathrm{Tr}(\alpha\Sigma_\mathcal{Y}^\top\Sigma_\mathcal{Y}\alpha^\top) \\
s.t. \quad & \alpha\Sigma_\mathcal{Z}^\top\Sigma_\mathcal{Z}\alpha^\top = I_m.
\end{aligned} \tag{A.63}$$

The optimization problem (A.63) has a known minimum which is the sum of the smallest m eigenvalues of $\Sigma_\mathcal{Y}^\top\Sigma_\mathcal{Y}$ [87], which in this case are simply the first $\Sigma_\mathcal{Y}^\top\Sigma_\mathcal{Y}$ elements of its diagonal. Let's propose the following minimizer:

$$\hat{\alpha} = S_{I0}^\top \triangleq \begin{bmatrix} I_m & 0_{m\times(n-m)} \end{bmatrix} \in \mathbb{R}^{m\times n}. \tag{A.64}$$

We easily verify that as long as the first m elements of the diagonal of $\Sigma_\mathcal{Z}$ are identical to 1, then (A.64) satisfies the equality condition in (A.63) and yields to the minimum cost in (A.63). Therefore, the result

$$\hat{\beta}^\top = X_{(\mathcal{Y},\mathcal{Z})}^{-1}\hat{\alpha}^\top = X_{(\mathcal{Y},\mathcal{Z})}^{-1}S_{I0}$$

is a minimizer for the optimization problem (A.53), given that the first m diagonal elements of $\Sigma_{\mathcal{Z}}$ are unitary.

■

Appendix B

Supplementary Proofs/Results

This appendix provides some supplementary proofs/results developed in the context of this thesis with the aim of improving the understanding of some of the material exposed, but fall outside of the novel contributions of the thesis.

B.1 Receptive Field Weighted Regression error cost decoupling

Let

$$\overline{\varepsilon}_{\text{nDPE}}(\beta) \triangleq \frac{1}{K}\sum_{k=1}^{K}\|u_k - \pi(s_k;\beta)\|^2 \tag{B.1}$$

be an error metric for the policy π with parameters β. We aim to prove that, for a policy

$$\pi(s;\beta) \triangleq \frac{\sum_{m=1}^{M}\omega_m(s)\pi_m(s;\beta_m)}{\sum_{m=1}^{M}\omega_m(s)} = \sum_{m=1}^{M}\overline{\omega}_m(s)\pi_m(s;\beta_m), \tag{B.2}$$

defined as a weighted combination of M local models π_m, where

$$\overline{\omega}_m(s) = \frac{\omega_m(s)}{\omega(s)} \tag{B.3}$$

are the importance weightings of each local model and

$$\omega(s) = \sum_{m=1}^{M}\omega_m(s), \tag{B.4}$$

we can write an upper bound to the error metric (B.1) for the global policy as a summation of decoupled error metrics of each local model, as

$$\overline{\varepsilon}_{\text{nDPE}}(\beta) \leq \sum_{m=1}^{M}\overline{\varepsilon}_{\text{nDPE}_m}(\beta_m), \tag{B.5}$$

with

$$\overline{\varepsilon}_{\mathrm{nDPE}_m}(\beta_m) = \frac{1}{K}\sum_{k=1}^{K} \overline{\omega}_m^2(s_k)\|u_k - \pi_m(s_k, \beta_m)\|^2, \tag{B.6}$$

where $\beta = \{\beta_1, \ldots, \beta_M\}$.

Proof. Lets start by replacing (B.3) in (B.2), obtaining

$$\omega(s)\pi(s;\beta) = \sum_{m=1}^{M} \omega_m(s)\pi_m(s;\beta_m). \tag{B.7}$$

From (B.1) we obtain that

$$\begin{aligned}
\overline{\varepsilon}_{\mathrm{nDPE}}(\beta) &\triangleq \frac{1}{K}\sum_{k=1}^{K} \|u_k - \pi(s_k;\beta)\|^2 \\
&= \frac{1}{K}\sum_{k=1}^{K} \frac{\omega^2(s_k)}{\omega^2(s_k)}\|u_k - \pi(s_k;\beta)\|^2 \\
&\overset{(a)}{=} \frac{1}{K}\sum_{k=1}^{K} \frac{1}{\omega^2(s_k)}\|\omega(s_k)u_k - \omega(s_k)\pi(s_k;\beta)\|^2 \\
&\overset{(b)}{=} \frac{1}{K}\sum_{k=1}^{K} \frac{1}{\omega^2(s_k)}\left\|\left(\sum_{m=1}^{M}\omega_m(s_k)\right)u_k - \sum_{m=1}^{M}\left(\omega_m(s_k)\pi_m(s_k;\beta_m)\right)\right\|^2 \\
&\overset{(c)}{=} \frac{1}{K}\sum_{k=1}^{K} \frac{1}{\omega^2(s_k)}\left\|\sum_{m=1}^{M}\omega_m(s_k)\left(u_k - \pi_m(s_k;\beta_m)\right)\right\|^2 \\
&\overset{(d)}{\leq} \frac{1}{K}\sum_{k=1}^{K} \frac{1}{\omega^2(s_k)}\sum_{m=1}^{M}\|\omega_m(s_k)\left(u_k - \pi_m(s_k;\beta_m)\right)\|^2 \\
&\overset{(e)}{=} \frac{1}{K}\sum_{k=1}^{K} \frac{1}{\omega^2(s_k)}\sum_{m=1}^{M}\omega_m^2(s_k)\|u_k - \pi_m(s_k;\beta_m)\|^2 \\
&\overset{(f)}{=} \frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M} \frac{\omega_m^2(s_k)}{\omega^2(s_k)}\|u_k - \pi_m(s_k;\beta_m)\|^2 \\
&\overset{(g)}{=} \frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M} \overline{\omega}_m^2(s_k)\|u_k - \pi_m(s_k;\beta_m)\|^2
\end{aligned}$$

$$\overset{(c)}{=} \sum_{m=1}^{M} \frac{1}{K} \sum_{k=1}^{K} \overline{\omega}_m^2(s_k) \|u_k - \pi_m(s_k; \beta_m)\|^2$$

$$= \sum_{m=1}^{M} \overline{\varepsilon}_{\text{nDPE}_m}(\beta_m),$$

(a) using the absolutely homogeneous property of any norm and for $\omega(\cdot) \geq 0$;
(b) using (B.4) and (B.7);
(c) using the distributivity, commutativity and associativity properties of the summation;
(d) using the triangle inequality;
(e) using the absolutely homogeneous properties of any norm and for $\omega_m(\cdot) \geq 0$;
(f) using the distributivity property of the summation;
(g) using (B.3).

■

B.2 Regression of a weighted combination of locally linear models

Let

$$\overline{\varepsilon}_{\text{nDPE}}(\beta) \triangleq \frac{1}{K} \sum_{k=1}^{K} \|u_k - \pi(s_k; \beta)\|^2 \tag{B.8}$$

be an error metric for the policy π with parameters β. We aim to prove that, for a policy

$$\pi(s; \beta) \triangleq \sum_{m=1}^{M} \overline{\omega}_m(s) \pi_m(s; \beta_m), \tag{B.9}$$

defined as a weighted combination of M local linear models $\pi_m(s) = \phi_\pi(s)\beta_m$, where $\overline{\omega}_m$ are the importance weightings of each local model, the vector of parameters $\beta = [\beta_1; \ldots; \beta_M]$ that minimizes the error (B.8) is

$$\hat{\beta} = \arg\min_{\beta} \overline{\varepsilon}_{\text{nDPE}}(\beta) = \Phi_{\mathcal{W}\pi}^{\dagger} \mathcal{U}, \tag{B.10}$$

with

$$\Phi_{\mathcal{W}\pi} = \begin{bmatrix} \mathcal{W}^\top(s_1) \otimes \phi_\pi(s_1) \\ \vdots \\ \mathcal{W}^\top(s_\mathcal{K}) \otimes \phi_\pi(s_\mathcal{K}) \end{bmatrix} \quad \text{and} \quad \mathcal{W}(s) = \begin{bmatrix} \overline{\omega}_1(s) \\ \vdots \\ \overline{\omega}_M(s) \end{bmatrix},$$

where † denotes the pseudo-inverse of a matrix and $\otimes$ denotes the Kronecker product operator.

Proof. Let's start by rewriting π as single linear model as,

$$\pi(s;\beta) = \sum_{m=1}^{M} \overline{\omega}_m(s)\phi_\pi(s)\beta_m = \phi_\pi(s) \sum_{m=1}^{M} \beta_m \overline{\omega}_m(s) \tag{B.11}$$

$$= \phi_\pi(s) \underbrace{\begin{bmatrix}\beta_1, & \cdots, & \beta_M\end{bmatrix}}_{\triangleq B} \underbrace{\begin{bmatrix}\overline{\omega}_1(s) \\ \vdots \\ \overline{\omega}_M(s)\end{bmatrix}}_{\triangleq \mathcal{W}(s)} \tag{B.12}$$

$$= \phi_\pi(s) B \mathcal{W}(s) \tag{B.13}$$

$$\overset{\text{(a)}}{=} \operatorname{vec}\left(\phi_\pi(s) B \mathcal{W}(s)\right) \tag{B.14}$$

$$\overset{\text{(b)}}{=} \underbrace{\left(\mathcal{W}^\top(s) \otimes \phi_\pi(s)\right)}_{\triangleq \phi_{\mathcal{W}\pi}(s)} \underbrace{\operatorname{vec}(B)}_{\triangleq \beta} = \phi_{\mathcal{W}\pi}(s)\beta, \tag{B.15}$$

where $\operatorname{vec}(B)$ denotes the vectorization of the matrix B, formed by stacking the columns of B into a single column vector, and where

(a) the vectorization of a vector is the vector itself;
(b) using the connection between the Kronecker product and vectorization [114].

Replacing (B.15) in (B.8) results in

$$\overline{\varepsilon}_{\text{nDPE}}(\beta) = \frac{1}{K} \sum_{k=1}^{K} \|u_k - \phi_{\mathcal{W}\pi}(s_k)\beta\|^2, \tag{B.16}$$

for which there is a known minimizer [19, Section 3.1.5]

$$\hat{\beta} = \arg\min_{\beta} \overline{\varepsilon}_{\text{nDPE}}(\beta) = \Phi_{\mathcal{W}\pi}^{\dagger} \mathcal{U}, \tag{B.17}$$

with

$$\Phi_{\mathcal{W}\pi} = \begin{bmatrix}\phi_{\mathcal{W}\pi}(s_1) \\ \vdots \\ \phi_{\mathcal{W}\pi}(s_\mathcal{K})\end{bmatrix} = \begin{bmatrix}\mathcal{W}^\top(s_1) \otimes \phi_\pi(s_1) \\ \vdots \\ \mathcal{W}^\top(s_\mathcal{K}) \otimes \phi_\pi(s_\mathcal{K})\end{bmatrix} \quad \text{and} \quad \mathcal{U} = \begin{bmatrix}u_1 \\ \vdots \\ u_\mathcal{K}\end{bmatrix}.$$

■

B.3 Estimating parameters with EVD

We aim to show that the transpose of the left eigenvectors of $\mathcal{Y}\mathcal{Y}^\top$ corresponding to its smallest eigenvalues is a solution to the optimization problem

$$\begin{aligned} \mathbb{R}^{m\times n} \ni \hat{\beta} = \arg\min_{\beta} \quad & \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2. \\ s.t. \quad & \beta\beta^\top = I_m, \end{aligned} \tag{B.18}$$

with

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Y} = \left[y_1, \ldots, y_{\mathcal{K}}\right].$$

Proof. Let's start by rewriting the cost in (B.18) as

$$\begin{aligned} \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2 &= \sum_{k=1}^{\mathcal{K}} y_k^\top \beta^\top \beta y_k^\top \\ &\overset{(a)}{=} \mathrm{Tr}(\mathcal{Y}^\top \beta^\top \beta \mathcal{Y}) \\ &\overset{(b)}{=} \mathrm{Tr}(\beta \mathcal{Y}\mathcal{Y}^\top \beta^\top) \\ &\overset{(c)}{=} \|\beta\mathcal{Y}\|_F^2, \end{aligned} \tag{B.19}$$

where we get (a) by inspection; (b) using the commutativity property of the trace operator [175]; and in (c) $\|A\|_F$ represents the Frobenius norm of A [175]. The Lagrangian (Boyd and Vandenberghe [27]) for (B.18) is

$$\mathcal{L} = \mathrm{Tr}(\beta\mathcal{Y}\mathcal{Y}^\top\beta^\top) - \mathrm{Tr}(\Lambda^\top(\beta\beta^\top - I_n)), \tag{B.20}$$

where $\Lambda \in \mathbb{R}^{m\times m}$ is a diagonal matrix whose entries are the Lagrange multipliers. Equating the partial derivatives of $\mathcal{L}$ [130] to zero give us

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\beta} &= 2\beta\mathcal{Y}\mathcal{Y}^\top - 2\Lambda^\top\beta \overset{\text{set}}{=} 0 \\ &\implies \beta\mathcal{Y}\mathcal{Y}^\top = \Lambda^\top\beta \\ &\implies \mathcal{Y}\mathcal{Y}^\top\beta^\top = \beta^\top\Lambda, \end{aligned} \tag{B.21}$$

which corresponds to the eigenvalue problem for $\mathcal{W}_\mathcal{Y}$, where $\mathcal{W}_\mathcal{Y} = \mathcal{Y}\mathcal{Y}^\top$ is a square symmetric matrix [175], and

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \Lambda} =& \beta\beta^\top - I_n \overset{\text{set}}{=} 0 \\ \Leftrightarrow& \beta\beta^\top = I_n. \end{aligned} \tag{B.22}$$

Consider the Eigenvalue Decomposition (EVD) of $\mathcal{W}_\mathcal{Y}$:

$$\mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}} = U_{\mathcal{W}_\mathcal{Y}} D_{\mathcal{W}_\mathcal{Y}}, \tag{B.23}$$

where $D_{\mathcal{W}_\mathcal{Y}} \in \mathbb{R}^{n\times n}$ is a diagonal matrix, whose elements contain the eigenvalues of $\mathcal{W}_\mathcal{Y}$, and where $U_{\mathcal{W}_\mathcal{Y}} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix whose columns are the corresponding eigenvectors [58]. Let's split (B.23) as

$$\begin{aligned} &\mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}} = U_{\mathcal{W}_\mathcal{Y}} D_{\mathcal{W}_\mathcal{Y}} \\ \Leftrightarrow\ &\mathcal{W}_\mathcal{Y} \begin{bmatrix} U_{\mathcal{W}_\mathcal{Y}1} & U_{\mathcal{W}_\mathcal{Y}2} \end{bmatrix} = \begin{bmatrix} U_{\mathcal{W}_\mathcal{Y}1} & U_{\mathcal{W}_\mathcal{Y}2} \end{bmatrix} \begin{bmatrix} D_{\mathcal{W}_\mathcal{Y}1} & 0 \\ 0 & D_{\mathcal{W}_\mathcal{Y}2} \end{bmatrix} \\ \Leftrightarrow\ &\begin{bmatrix} \mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}1} & \mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}2} \end{bmatrix} = \begin{bmatrix} U_{\mathcal{W}_\mathcal{Y}1} D_{\mathcal{W}_\mathcal{Y}1} & U_{\mathcal{W}_\mathcal{Y}2} D_{\mathcal{W}_\mathcal{Y}2} \end{bmatrix}, \end{aligned} \tag{B.24}$$

with $D_{\mathcal{W}_\mathcal{Y}1} \in \mathbb{R}^{(n-m)\times(n-m)}$ and $D_{\mathcal{W}_\mathcal{Y}1} \in \mathbb{R}^{m\times m}$. Then, by inspection of the second term of (B.24) where $\mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}2} = U_{\mathcal{W}_\mathcal{Y}2} D_{\mathcal{W}_\mathcal{Y}2}$, we can immediately see that the solution

$$\beta = U_{\mathcal{W}_\mathcal{Y}2}^\top \in \mathbb{R}^{n\times m}, \tag{B.25}$$

i.e. the transpose of the matrix composed by m eigenvectors of $\mathcal{W}_\mathcal{Y}$, satisfies the condition (B.21). Furthermore, given that $U_{\mathcal{W}_\mathcal{Y}2} U_{\mathcal{W}_\mathcal{Y}2}^\top = I_m$, then (B.25) also satisfies (B.22) and leads to

$$\mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}2} = U_{\mathcal{W}_\mathcal{Y}2} D_{\mathcal{W}_\mathcal{Y}2} \Leftrightarrow U_{\mathcal{W}_\mathcal{Y}2}^\top \mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}2} = D_{\mathcal{W}_\mathcal{Y}2}.$$

The remaining question is which selection of eigenvectors minimize the cost (B.19). We, therefore, substitute $\beta = U_{\mathcal{W}_\mathcal{Y}2}^\top \in \mathbb{R}^{n\times m}$ in (B.19), obtaining

$$\begin{aligned} \text{Tr}(\beta \mathcal{Y}\mathcal{Y}^\top \beta^\top) &= \text{Tr}(U_{\mathcal{W}_\mathcal{Y}2}^\top \mathcal{W}_\mathcal{Y} U_{\mathcal{W}_\mathcal{Y}2}) \\ &= \text{Tr}(D_{\mathcal{W}_\mathcal{Y}2}), \end{aligned}$$

which is minimal for $\mathcal{W}_{\mathcal{Y}2}$ containing the smallest eigenvectors of $\mathcal{W}_\mathcal{Y}$, showing that the transpose of the matrix of right eigenvectors of $\mathcal{W}_\mathcal{Y}$ corresponding to the smallest eigenvalues of $\mathcal{W}_\mathcal{Y}$ is a solution to (B.18).

■

B.4 Estimating parameters with SVD

We aim to show that

$$\hat{\beta}^\top = U_\mathcal{Y} S_{0I} \tag{B.26}$$

is a solution to the optimization problem

$$\begin{aligned} \mathbb{R}^{m\times n} \ni \hat{\beta} = \arg\min_{\beta} \quad & \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2 \\ s.t. \quad & \beta\beta^\top = I_m, \end{aligned} \tag{B.27}$$

where $U_\mathcal{Y}$ is the matrix of the left-singular vectors resulting from the Singular Value Decomposition (SVD) of $\mathcal{Y}$, with

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Y} = \left[y_1, \ldots, y_\mathcal{K}\right],$$

and where $S_{0I}^\top = \begin{bmatrix} 0_{m\times(n-m)} & I_m \end{bmatrix} \in \mathbb{R}^{m\times n}$ is a selection matrix, which extracts the last m columns of $U_\mathcal{Y}$.

Proof. Let's start by rewriting the cost in (B.18) as

$$\begin{aligned} \sum_{k=1}^{\mathcal{K}} \|\beta y_k\|^2 &= \sum_{k=1}^{\mathcal{K}} y_k^\top \beta^\top \beta y_k^\top \\ &\overset{(a)}{=} \mathrm{Tr}(\mathcal{Y}^\top \beta^\top \beta \mathcal{Y}) \\ &\overset{(b)}{=} \|\mathcal{Y}^\top \beta^\top\|_F^2, \end{aligned} \tag{B.28}$$

where we get (a) by inspection; and in (b) $\|\cdot\|_F$ represents the Frobenius norm [175].

Let's now consider the Singular Value Decomposition (SVD)

$$\mathbb{R}^{n\times\mathcal{K}} \ni \mathcal{Y} = U_\mathcal{Y} \Sigma_\mathcal{Y} V_\mathcal{Y}^\top, \tag{B.29}$$

where $\Sigma_\mathcal{Y} \in \mathbb{R}^{n\times\mathcal{K}}$ is a rectangular diagonal matrix whose diagonal entries contain the singular values of $\mathcal{Y}$, and $U_\mathcal{Y} \in \mathbb{R}^{n\times n}$ and $V_\mathcal{Y} \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ are orthogonal matrices composed, respectively, of the left-single vectors and right-single vectors of $\mathcal{Y}$ [175]. We can then replace (B.29) in (B.28), obtaining

$$\begin{aligned} \|\mathcal{Y}^\top \beta^\top\|_F^2 &= \|V_\mathcal{Y} \Sigma_\mathcal{Y}^\top U_\mathcal{Y}^\top \beta^\top\|_F^2 \\ &\overset{(c)}{=} \|\Sigma_\mathcal{Y}^\top U_\mathcal{Y}^\top \beta^\top\|_F^2 \\ &\overset{(d)}{=} \|\Sigma_\mathcal{Y}^\top \alpha^\top\|_F^2 \\ &\overset{(e)}{=} \mathrm{Tr}(\alpha \Sigma_\mathcal{Y}^2 \alpha^\top) \end{aligned} \tag{B.30}$$

where in (c) the norm is invariant to the orthogonal matrix $V_{\mathcal{Y}}$; in (d) $\alpha \triangleq \beta U_{\mathcal{Y}}$; and in (e) $\Sigma_{\mathcal{Y}}^2 \triangleq \Sigma_{\mathcal{Y}}\Sigma_{\mathcal{Y}}^\top \in \mathbb{R}^{n\times n}$ is a square diagonal matrix containing the square of the singular values of $\mathcal{Y}$, in descending order. We also have that

$$\beta\beta^\top = \beta U_{\mathcal{Y}} U_{\mathcal{Y}}^\top \beta^\top = \alpha\alpha^\top = I_m. \tag{B.31}$$

By substituting the rewritten condition (B.31) and cost (B.30) in the optimization (B.27), we can redefine it as

$$\begin{aligned} \mathbb{R}^{m\times n} \ni \hat{\alpha} = \arg\min_{\alpha} \quad & \mathrm{Tr}(\alpha\Sigma_{\mathcal{Y}}^2\alpha^\top) \\ s.t. \quad & \alpha\alpha^\top = I_m. \end{aligned} \tag{B.32}$$

The optimization problem (B.32) has a known minimum which is the sum of the smallest m eigenvalues of $\Sigma_{\mathcal{Y}}^2$ [87, 145], which in this case are simply the last m elements of its diagonal. Let's then propose the following minimizer:

$$\hat{\alpha} = S_{0I}^\top \triangleq \begin{bmatrix} 0_{m\times(n-m)} & I_m \end{bmatrix} \in \mathbb{R}^{m\times n}. \tag{B.33}$$

We can immediately see that (B.33) satisfies the equality condition in (B.32) and yields to the minimum cost in (B.32). Therefore, the result

$$\hat{\beta}^\top = U_{\mathcal{Y}}\hat{\alpha}^\top = U_{\mathcal{Y}}S_{0I}$$

is a minimizer for the optimization problem (B.27).

■

B.5 Inertia-Weighted Generalized Inverses and Projection Equalities

Let $\overline{A}$ be the inertia-weighted generalized inverse of the rank deficient matrix A, satisfying the conditions: (a) $A\overline{A}A = A$ (b) $\overline{A}A\overline{A} = \overline{A}$ (c) $A\overline{A} = \left(A\overline{A}\right)^\top$ (d) $M\overline{A}A = \left(M\overline{A}A\right)^\top$, given $M \in \mathbb{S}_{++}^n$, and let $P_M \triangleq (I_n - \overline{A}A)$. This appendix derives some useful equalities resulting from the above definition of weighted-inverse matrix:

$$M\overline{A}A = \left(M\overline{A}A\right)^\top \Leftrightarrow \overline{A}AM^{-1} = \left(\overline{A}AM^{-1}\right)^\top, \tag{B.34}$$

$$\begin{aligned} P_M M^{-1} &= M^{-1} - \overline{A}AM^{-1} \\ &\overset{\text{B.34}}{=} M^{-1} - \left(\overline{A}AM^{-1}\right)^\top \\ &= M^{-1} - M^{-1}\left(\overline{A}A\right)^\top \\ &= M^{-1}P_M^\top. \end{aligned} \tag{B.35}$$

Bibliography

[1] Farhad Aghili. Inverse and direct dynamics of constrained multibody systems based on orthogonal decomposition of generalized force. In *IEEE International Conference on Robotics and Automation, ICRA*, 2003. doi:10.1109/ROBOT.2003.1242217.

[2] Farhad Aghili. A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: Applications to control and simulation. *IEEE Transactions on Robotics*, 21(5):834–849, 2005. doi:10.1109/TRO.2005.851380.

[3] Farhad Aghili. Projection-based modeling and control of mechanical systems using non-minimum set of coordinates. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2015. doi:10.1109/IROS.2015.7353815.

[4] Farhad Aghili. Non-minimal order model of mechanical systems with redundant constraints for simulations and controls. *IEEE Transactions on Automatic Control*, 61(5):1350 – 1355, 2016. doi:10.1109/TAC.2015.2463632.

[5] Farhad Aghili. Modeling and analysis of multiple impacts in multibody systems under unilateral and bilateral constrains based on linear projection operators. *Multibody System Dynamics*, 46:41–62, 2019. doi:10.1007/s11044-018-09658-w.

[6] Farhad Aghili and Jean Claude Piedbœuf. Simulation of motion of constrained multibody systems based on projection operator. *Multibody System Dynamics*, 10(1):3–16, 2003. doi:10.1023/A:1024584323751.

[7] Farhad Aghili and Chun Yi Su. Control of constrained robots subject to unilateral contacts and friction cone constraints. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2016-June, pages 2347–2352. IEEE, 2016. doi:10.1109/ICRA.2016.7487385.

[8] Gianluca Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25(5):985 – 994, 2009. doi:10.1109/TRO.2009.2017135.

[9] Leopoldo Armesto, Jorren Bosga, Vladimir Ivan, and Sethu Vijayakumar. Efficient learning of constraints and generic null space policies. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1520–1526, 2017. doi:10.1109/ICRA.2017.7989181.

[10] Leopoldo Armesto, João Moura, Vladimir Ivan, Antonio Salas, and Sethu Vijayakumar. Learning constrained generalizable policies by demonstration. In *Robotics: Science and Systems (RSS)*, 2017. doi:10.15607/RSS.2017.XIII.036.

[11] Leopoldo Armesto, João Moura, Vladimir Ivan, Antonio Salas, and Sethu Vijayakumar. Learning constrained generalizable policies by demonstration, 2017. URL `https://youtu.be/4Jo7q1_Frrc`. Accessed on 29 June 2020.

[12] Leopoldo Armesto, João Moura, Vladimir Ivan, Mustafa Suphi Erden, Antonio Sala, and Sethu Vijayakumar. Constraint-aware learning of policies by demonstration. *The International Journal of Robotics Research*, 37(13-14):1673–1689, 2018. doi:10.1177/0278364918784354.

[13] Junghwan Back, João Bimbo, Yohan Noh, Lakmal Seneviratne, Kaspar Althoefer, and Hongbin Liu. Control a contact sensing finger for surface haptic exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. doi:10.1109/ICRA.2014.6907251.

[14] P. Baerlochcr and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 1998. doi:10.1109/IROS.1998.724639.

[15] K. S. Banerjee, C. Radhakrishna Rao, and Sujit Kumar Mitra. Generalized inverse of matrices and its applications. In *Berkeley Symposium on Mathematical Statistics and Probability*, 1973. doi:10.2307/1266840.

[16] Antonio Bicchi, J. Kenneth Salisbury, and David L. Brock. Contact sensing from force measurements. *The International Journal of Robotics Research*, 12 (3):249–262, 1993. doi:10.1177/027836499301200304.

[17] Aude Billard and Roland Siegwart. Robot learning from demonstration. *Robotics and Autonomous Systems*, 47(2-3):65–67, 2004. doi:10.1016/j.robot.2004.03.001.

[18] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer Handbook of Robotics*, pages 1371–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-30301-5_60.

[19] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

[20] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Developments of the generative topographic mapping. *Neurocomputing*, 21(1-3): 203–224, 1998. doi:10.1016/S0925-2312(98)00043-5.

[21] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998. doi:10.1162/089976698300017953.

[22] Sebastian Bitzer and Sethu Vijayakumar. Latent spaces for dynamic movement primitives. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2009. doi:10.1109/ICHR.2009.5379530.

[23] Wojciech Blajer. A geometric unification of constrained system dynamics. *Multibody System Dynamics*, 1:3–21, 1997. doi:10.1023/A:1009759106323.

[24] M. Blauer and P. Belanger. State and parameter estimation for robotic manipulators using force measurements. *IEEE Transactions on Automatic Control*, 32(12):1055–1066, 1987. doi:10.1109/TAC.1987.1104524.

[25] Ye Bosheng, Song Bao, Li Zhengyi, Xiong Shuo, and Tang Xiaoqi. A study of force and position tracking control for robot contact with an arbitrarily inclined plane. *International Journal of Advanced Robotic Systems*, 10(1), 2012. doi:10.5772/55086.

[26] Karim Bouyarmane and Abderrahmane Kheddar. On weight-prioritized multitask control of humanoid robots. *IEEE Transactions on Automatic Control*, 63(6):1632 – 1647, 2018. doi:10.1109/TAC.2017.2752085.

[27] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.

[28] Gerald Brantner and Oussama Khatib. Controlling ocean one: Human–robot collaboration for deep-sea manipulation. *Journal of Field Robotics*, 2020. doi:10.1002/rob.21960.

[29] Herman Bruyninckx and Oussama Khatib. Gauss' principle and the dynamics of redundant and constrained manipulators. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000. doi:10.1109/ROBOT.2000.846414.

[30] Pasquale Chiacchio, Stefano Chiaverini, Lorenzo Sciavicco, and Bruno Siciliano. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *The International Journal of Robotics Research*, 10(4):410–425, 1991. doi:10.1177/027836499101000409.

[31] Stefano Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398 – 410, 1997. doi:10.1109/70.585902.

[32] Jefferson A. Coelho, Elizeth G. Araujo, Manfred Huber, and Roderic A. Grupen. Dynamical categories and control policy selection. In *IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 459–464, 1998. doi:10.1109/isic.1998.713705.

[33] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB, Second Edition*. Springer Publishing Company, Incorporated, 2017. doi:10.1007/978-3-319-54413-7.

[34] Vincent De Sapio and Oussama Khatib. Operational space control of multibody systems with explicit holonomic constraints. In *IEEE International Conference on Robotics and Automation, ICRA*, 2005. doi:10.1109/ROBOT.2005.1570562.

[35] Vincent De Sapio, Oussama Khatib, and Scott Delp. Task-level approaches for the control of constrained multibody systems. *Multibody System Dynamics*, 16 (1):73–102, 2006. doi:10.1007/s11044-006-9017-3.

[36] Niels Dehio. *Prioritized Multi-Objective Robot Control*. PhD thesis, Technical University Braunschweig, 2018.

[37] Niels Dehio and Jochen J Steil. Dynamically-consistent generalized hierarchical control. In *International Conference on Robotics and Automation, ICRA*, 2019. doi:10.1109/ICRA.2019.8793553.

[38] Niels Dehio, Joshua Smith, Dennis Leroy Wigand, Guiyang Xin, Hsiu-chin Lin, Jochen J Steil, and Michael Mistry. Modeling and control of multi-arm and multi-leg robots: Compensating for object dynamics during grasping. In *IEEE International Conference on Robotics and Automation, ICRA*, 2018. doi:10.1109/ICRA.2018.8462872.

[39] Andrea Del, Francesco Nori, Giorgio Metta, and Lorenzo Natale. Prioritized motion – force control of constrained fully-actuated robots : "task space inverse dynamics ". *Robotics and Autonomous Systems*, 63:150–157, 2015. doi:10.1016/j.robot.2014.08.016.

[40] Alexander Dietrich, Alin Albu-Schäffer, and Gerd Hirzinger. On continuous null space projections for torque-based, hierarchical, multi-objective manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2978–2985. IEEE, 2012. doi:10.1109/ICRA.2012.6224571.

[41] Alexander Dietrich, Christian Ott, and Alin Albu-Schäffer. An overview of null space projections for redundant, torque-controlled robots. *The International Journal of Robotics Research*, 34(11):1385–1400, 2015. doi:10.1177/0278364914566516.

[42] Alexander Dietrich, Christian Ott, and Stefano Stramigioli. Passivation of

projection-based null space compliance control via energy tanks. *IEEE Robotics and Automation Letters*, 1(1):184–191, 2016. doi:10.1109/LRA.2015.2512937.

[43] Alexander Dietrich, Christian Ott, and Jaeheung Park. The hierarchical operational space formulation: Stability analysis for the regulation case. *IEEE Robotics and Automation Letters*, 3(2):1–1, 2018. doi:10.1109/LRA.2018.2792154.

[44] N. Diolaiti, C. Melchiorri, and S. Stramigioli. Contact impedance estimation for robotic systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2004. doi:10.1109/iros.2004.1389790.

[45] Keith L. Doty, Claudio Melchiorri, and Claudio Bonivento. A theory of generalized inverses applied to robotics. *The International Journal of Robotics Research*, 12(1):1–19, 1993. doi:10.1177/027836499301200101.

[46] 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. doi:10.1002/rob.4620070202.

[47] James D. English and Anthony A. Maciejewski. On the implementation of velocity control for kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 30(3):233–237, 2000. doi:10.1109/3468.844350.

[48] Adrien Escande, Nicolas Mansard, and Pierre Brice Wieber. Fast resolution of hierarchized inverse kinematics with inequality constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. doi:10.1109/ROBOT.2010.5509953.

[49] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014. doi:10.1177/0278364914521306.

[50] Nima Fazeli, Roman Kolbert, Russ Tedrake, and Alberto Rodriguez. Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact. *International Journal of Robotics Research*, 36(13-14):1437–1454, 2017. doi:10.1177/0278364917698749.

[51] Roy Featherstone. An empirical study of the joint space inertia matrix. *The International Journal of Robotics Research*, 23(9):859–871, 2004. doi:10.1177/0278364904044869.

[52] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, Berlin, Heidelberg, 2008. doi:10.1007/978-1-4899-7560-7.

[53] Roy Featherstone. Exploiting sparsity in operational-space dynam-

ics. *International Journal of Robotics Research*, 29(10):1353–1368, 2010. doi:10.1177/0278364909357644.

[54] Roy Featherstone and Oussama Khatib. Load independence of the dynamically consistent inverse of the jacobian matrix. *The International Journal of Robotics Research*, 16(2):168–170, 1997. doi:10.1177/027836499701600203.

[55] Cliff Fitzgerald. Developing baxter. In *IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2013. doi:10.1109/TePRA.2013.6556344.

[56] Paulo Flores and Hamid Lankarani. *Contact Force Models for Multibody Dynamics*. Springer, 01 2016.

[57] Gowrishankar Ganesh, Nathanael Jarrassé, Sami Haddadin, Alin Albu-Schaeffer, and Etienne Burdet. A versatile biomimetic controller for contact tooling and haptic exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012. doi:10.1109/ICRA.2012.6225057.

[58] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[59] Gene H. (Gene Howard) Golub. *Matrix computations*. The Johns Hopkins University Press, Baltimore, Md., fourth edition, 2013.

[60] Sovannara Hak, Nicolas Mansard, Olivier Stasse, and Jean Paul Laumond. Reverse control for humanoid robot task recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(6):1524–1537, 2012. doi:10.1109/TSMCB.2012.2193614.

[61] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016. doi:10.1007/s10514-015-9476-6.

[62] Jurgen Hess, Gian Diego Tipaldi, and Wolfram Burgard. Null space optimization for effective coverage of 3d surfaces using redundant manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1923–1928. IEEE, 2012. doi:10.1109/IROS.2012.6385960.

[63] Matthew Howard. *Learning Control Policies from Constrained Motion*. PhD thesis, University of Edinburgh, 2009.

[64] Matthew Howard, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Learning utility surfaces for movement selection. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2006. doi:10.1109/ROBIO.2006.340168.

[65] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and

Sethu Vijayakumar. Learning potential-based policies from constrained motion. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2008. doi:10.1109/ICHR.2008.4755977.

[66] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Behaviour generation in humanoids by learning potential-based policies from constrained motion. *Applied Bionics and Biomechanics*, 5 (4):195–211, 2008. doi:10.1080/11762320902789830.

[67] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. A novel method for learning policies from variable constraint data. *Autonomous Robots*, 27(2):105–121, 2009. doi:10.1007/s10514-009-9129-8.

[68] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Robust constraint-consistent learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4629–4636, 2009. doi:10.1109/IROS.2009.5354663.

[69] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. A novel method for learning policies from constrained motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009. doi:10.1109/robot.2009.5152335.

[70] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1547–1554, 2002.

[71] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors, 2013.

[72] Rodrigo Jamisola, Marcelo H. Ang, Denny Oetomo, Oussama Khatib, Tao Ming, and Ser Yong Lim. The operational space formulation implementation to aircraft canopy polishing using a mobile manipulator. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002. doi:10.1109/ROBOT.2002.1013393.

[73] Rodrigo S. Jamisola, Denny N. Oetomo, Marcelo H. Ang, Oussama Khatib, Tao Ming Lim, and Ser Yong Lim. Compliant motion using a mobile manipulator: An operational space formulation approach to aircraft canopy polishing. *Advanced Robotics*, 19(5):613–634, 2005. doi:10.1163/156855305323383820.

[74] Rodrigo S. Jamisola, Petar Kormushev, Antonio Bicchi, and Darwin G. Caldwell. Haptic exploration of unknown surfaces with discontinuities. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1255–1260. IEEE, 2014. doi:10.1109/IROS.2014.6942718.

[75] Oussama Kanoun, Florent Lamiraux, and Pierre Brice Wieber. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Transactions on Robotics*, 27(4):785 – 792, 2011. doi:10.1109/TRO.2011.2142450.

[76] Mohammad Khansari-Zadeh and Aude Billard. Learning stable non-linear dynamical systems with gaussian mixture models. *Transactions on Robotics*, 27(5):943–957, 2011. doi:10.1109/TRO.2011.2159412.

[77] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. doi:10.1109/JRA.1987.1087068.

[78] Oussama Khatib. Inertial properties in robotic manipulation: An object-level framework. *The International Journal of Robotics Research*, 14(1):19–36, 1995. doi:10.1177/027836499501400103.

[79] Oussama Khatib and Joel Burdick. Motion and force control of robot manipulators. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1986.

[80] CG Khatri and C Rao. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhyā: The Indian Journal of Statistics, Series A*, 30(2):167–180, 1968.

[81] Petar Kormushev, Dragomir N. Nenchev, Sylvain Calinon, and Darwin G. Caldwell. Upper-body kinesthetic teaching of a free-standing humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. doi:10.1109/ICRA.2011.5979537.

[82] Martin De Lasa and Aaron Hertzmann. Prioritized optimization for task-space control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009. doi:10.1109/IROS.2009.5354341.

[83] André Laulusa and Olivier A. Bauchau. Review of classical approaches for constraint enforcement in multibody systems. *Journal of Computational and Nonlinear Dynamics*, 3, 2008. doi:10.1115/1.2803257.

[84] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[85] Daniel Leidner, Alexander Dietrich, Florian Schmidt, Christoph Borst, and Alin Albu-Schäffer. Object-centered hybrid reasoning for whole-body mobile manipulation. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2014. doi:10.1109/ICRA.2014.6907099.

[86] Miao Li, Kenji Tahara, and Aude Billard. Learning task manifolds for constrained object manipulation. *Autonomous Robots*, 42(1):159–174, 2018. doi:10.1007/s10514-017-9643-z.

[87] Xin Liang, Ren Cang Li, and Zhaojun Bai. Trace minimization principles for positive semi-definite pencils. *Linear Algebra and Its Applications*, 2013. doi:10.1016/j.laa.2012.12.003.

[88] Alain Liegeois. Automatic supervisory control of the configuration and behavior of multibody mecanisms. *IEEE Transactions on Systems, Man and Cybernetics*, 7(12):868–871, 1977. doi:10.1109/tsmc.1977.4309644.

[89] Hsiu-Chin Lin, Matthew Howard, and Sethu Vijayakumar. Learning null space projections. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2613–2619, 2015. doi:10.1109/ICRA.2015.7139551.

[90] Hsiu-Chin Lin, Prabhakar Ray, and Matthew Howard. Learning task constraints in operational space formulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 309–315. IEEE, 2017. doi:10.1109/ICRA.2017.7989039.

[91] Hsiu-Chin Lin, Joshua Smith, Keyhan Kouhkiloui Babarahmati, Niels Dehio, and Michael Mistry. A projected inverse dynamics approach for dual-arm cartesian impedance control. In *IEEE International Conference on Robotics and Automation, ICRA*, 2018. doi:10.1109/ICRA.2018.8461202.

[92] Mingxing Liu, Yang Tan, and Vincent Padois. Generalized hierarchical control. *Autonomous Robots*, 40(1):17–31, 2016. doi:10.1007/s10514-015-9436-1.

[93] S Liu and G Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *International Journal of Information and systems sciences*, 4(1): 160–177, 2008.

[94] Ewald Lutscher and Gordon Cheng. Constrained manipulation in unstructured environment utilizing hierarchical task specification for indirect force controlled robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3471–3476. IEEE, 2014. doi:10.1109/ICRA.2014.6907359.

[95] Ewald Lutscher, Emmanuel C. Dean-Leon, and Gordon Cheng. Hierarchical force and positioning task specification for indirect force controlled robots. *IEEE Transactions on Robotics*, 34(1):280–286, 2018. doi:10.1109/TRO.2017.2765674.

[96] Anthony A. Maciejewski and Charles A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985. doi:10.1177/027836498500400308.

[97] Jeevan Manavalan and Matthew Howard. Learning null space projections fast. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2017.

[98] Giacomo Marani, Jinhyun Kim, Junku Yuh, and Wan K. Chung. Algorithmic singularities avoidance in task-priority based controller for redundant manipulators. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2003. doi:10.1109/IROS.2003.1249709.

[99] Matthew T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 11(6): 418–432, 1981. doi:10.1109/TSMC.1981.4308708.

[100] James K. Mills and Andrew A. Goldenberg. Force and position control of manipulators during constrained motion tasks. *IEEE Transactions on Robotics and Automation*, 5(1):30 – 46, 1989. doi:10.1109/70.88015.

[101] Michael Mistry and Ludovic Righetti. Operational space control of constrained and underactuated systems. *Robotics: Science and Systems, RSS*, 2011. doi:10.15607/RSS.2011.VII.031.

[102] Michael Mistry, Jun Nakanishi, and Stefan Schaal. Task space control with prioritization for balance and locomotion. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007. doi:10.1109/IROS.2007.4399595.

[103] Cleve B. Moler. Numerical Computing with Matlab. In *Numerical Computing with MATLAB, Revised Reprint*, chapter 0. SIAM, Philadelphia, 2008. doi:10.1137/1.9780898717952.fm.

[104] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116. Springer, Berlin, Heidelberg, 1978. doi:10.1007/bfb0067700.

[105] João Moura and Mustafa Suphi Erden. Formulation of a control and path planning approach for a cab front cleaning robot. In *Procedia CIRP*, volume 59, pages 67–71, 2017. doi:10.1016/j.procir.2016.09.024.

[106] João Moura, William McColl, Gerard Taykaldiranian, Tetsuo Tomiyama, and Mustafa Suphi Erden. Automation of train cab front cleaning with a robot manipulator. *IEEE Robotics and Automation Letters*, 3(4):3058 – 3065, 2018. doi:10.1109/LRA.2018.2849591.

[107] João Moura, Vladimir Ivan, Mustafa Suphi Erden, and Sethu Vijayakumar. Equivalence of the projected forward dynamics and the dynamically consistent inverse solution. In *Robotics: Science and Systems (RSS)*, 2019. doi:10.15607/rss.2019.xv.036.

[108] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., USA, 1st edition, 1994.

[109] Fusaomi Nagata, Tetsuo Hase, Zenku Haga, Masaaki Omoto, and Keigo Watanabe. Cad/cam-based position/force controller for a mold polishing robot. *Mechatronics*, 17(4-5):207–216, 2007. doi:10.1016/j.mechatronics.2007.01.003.

[110] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008. doi:10.1177/0278364908091463.

[111] Mehrzad Namvar and Farhad Aghili. Adaptive force-motion control of coordinated robots interacting with geometrically unknown environments. *IEEE Transactions on Robotics*, 21(4):678–694, 2005. doi:10.1109/TRO.2004.842346.

[112] Bojan Nemec, Leon Zlajpah, and Damir Omrčen. Comparison of null-space and minimal null-space control algorithms. *Robotica*, 25(5):511–520, 2007. doi:10.1017/S0263574707003402.

[113] Dragomir N. Nenchev, Atsushi Konno, and Teppei Tsujita. *Humanoid robots: modeling and control*. Butterworth-Heinemann, Oxford, OX, 2019. doi:10.1016/C2015-0-01877-9.

[114] H. Neudecker. Some theorems on matrix differentiation with special reference to kronecker matrix products. *Journal of the American Statistical Association*, 64(327):953–963, 1969. doi:10.2307/2283476.

[115] Denny Oetomo, Marcelo H. Ang, Rodrigo Jamisola, and Oussama Khatib. Integration of torque controlled arm with velocity controlled base for mobile manipulation. In *Romansy 14*. Springer, Vienna, 2002. doi:10.1007/978-3-7091-2552-6_22.

[116] Ken Ohta, Mikhail M. Svinin, Zhiwei Luo, Shigeyuki Hosoe, and Rafael Laboissière. Optimal trajectory formation of constrained human arm reaching movements. *Biological Cybernetics*, 91:23–36, 2004. doi:10.1007/s00422-004-0491-5.

[117] Valerio Ortenzi, Maxime Adjigble, Jeffrey A. Kuo, Rustam Stolkin, and Michael Mistry. An experimental study of robot control during environmental contacts based on projected operational space dynamics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015. doi:10.1109/HUMANOIDS.2014.7041392.

[118] Valerio Ortenzi, Rustam Stolkin, Jeffrey A. Kuo, and Michael Mistry. Projected inverse dynamics control and optimal control for robots in contact with the environment: A comparison. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. doi:10.1109/IROS.2015.7353942.

[119] Valerio Ortenzi, Hsiu-Chin Lin, Morteza Azad, Rustam Stolkin, Jeffrey A. Kuo, and Michael Mistry. Kinematics-based estimation of contact constraints using only proprioception. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1304–1311. IEEE, 2016. doi:10.1109/HUMANOIDS.2016.7803438.

[120] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives. *Neural Information Processing Systems*, pages 1–9, 2013.

[121] Alexandros Paraschos, Elmar Rueckert, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives under unknown system dynamics. *Advanced Robotics*, 32(6):297–310, 2018. doi:10.1080/01691864.2018.1437674.

[122] Diego Pardo, Michael Neunert, Alexander W Winkler, Ruben Grandia, and Jonas Buchli. Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion. In *Robotics, Science and Systems (RSS)*, 2017. doi:10.15607/RSS.2017.XIII.042.

[123] Dae Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2008. doi:10.1109/ICHR.2008.4755937.

[124] J. Park, Y. Choi, Wan Kyun Chung, and Y. Youm. Multiple tasks kinematics using weighted pseudo-inverse for kinematically redundant manipulators. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2001. doi:10.1109/robot.2001.933249.

[125] Jaeheung Park and Oussama Khatib. Contact consistent control framework for humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2006. doi:10.1109/ROBOT.2006.1641993.

[126] Jaeheung Park and Oussama Khatib. A haptic teleoperation approach based on contact force control. In *International Journal of Robotics Research*, 2006. doi:10.1177/0278364906065385.

[127] Claudia Perez-D'Arpino and Julie A. Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4058–4065. IEEE, 2017. doi:10.1109/ICRA.2017.7989466.

[128] Jan Peters and Stefan Schaal. Learning to control in operational space. *The International Journal of Robotics Research*, 27(2):197–212, 2008. doi:10.1177/0278364907087548.

[129] Jan Peters, Michael Mistry, Firdaus Udwadia, Rick Cory, Jun Nakanishi, and Stefan Schaal. A unifying methodology for the control of robotic systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005. doi:10.1109/IROS.2005.1545516.

[130] K. B. Petersen and M. S. Pedersen. The matrix cookbook, October 2008. URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`.

[131] Robert Platt, Muhammad Abdallah, and Charles Wampler. Multiple-priority impedance control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. doi:10.1109/ICRA.2011.5980228.

[132] Robert Platt, Muhammad Abdallah, and Charles Wampler. Multi-priority cartesian impedance control. In *Robotics: Science and Systems (RSS)*, 2011.

[133] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research*, 33(1):69–81, 2014. doi:10.1177/0278364913506757.

[134] Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016. doi:10.1109/ICRA.2016.7487270.

[135] M. H. Raibert and J. J. Craig. Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):126, 1981. doi:10.1115/1.3139652.

[136] Jitendra R. Raol. *Multi-Sensor Data Fusion with MATLAB*. CRC Press, Inc., USA, 1st edition, 2009. doi:10.1201/9781439800058.

[137] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Inverse dynamics control of floating-base robots with external constraints: A unified view. In *IEEE International Conference on Robotics and Automation, ICRA*, 2011. doi:10.1109/ICRA.2011.5980156.

[138] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *International Journal of Robotics Research*, 32(3):280–298, 2013. doi:10.1177/0278364912469821.

[139] Mark Rijnen, Eric De Mooij, Silvio Traversaro, Francesco Nori, Nathan Van De Wouw, Alessandro Saccon, and Henk Nijmeijer. Control of humanoid robot motions with impacts: Numerical experiments with reference spreading control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. doi:10.1109/ICRA.2017.7989472.

[140] Mark Rijnen, Alessandro Saccon, and Henk Nijmeijer. Reference spreading: Tracking performance for impact trajectories of a 1dof setup. *IEEE Transactions on Control Systems Technology*, 28(3):1124–1131, 2020. doi:10.1109/TCST.2019.2898953.

[141] Charles A. Rohde. Generalized inverses of partitioned matrices. *Journal of the Society for Industrial and Applied Mathematics*, 13(4):146–153, 1964. doi:10.13624/j.cnki.issn.1001-7445.2012.06.031.

[142] L Saab, N Mansard, F Keith, J-y Fourquet, and P Soueres. Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011. doi:10.1109/ICRA.2011.5980384.

[143] Hamid Sadeghian, Luigi Villani, Mehdi Keshmiri, and Bruno Siciliano. Dynamic multi-priority control in redundant robotic systems. *Robotica*, 31(7): 1155–1167, 2013. doi:10.1017/S0263574713000416.

[144] J. Kenneth Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *IEEE Conference on Decision and Control (CDC)*, 1980.

[145] Ahmed H. Sameh and John A. Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM Journal on Numerical Analysis*, 1982. doi:10.1137/0719089.

[146] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431):537–547, 2003. doi:10.1098/rstb.2002.1258.

[147] Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1997. doi:10.1007/978-1-4419-1428-6_4646.

[148] Stefan Schaal and Christopher G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998. doi:10.1162/089976698300016963.

[149] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. *Robotics Research*, 15(2005):1–10, 2005. doi:10.1007/11008941_60.

[150] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4):505–518, 2005. doi:10.1142/S0219843605000594.

[151] Luis Sentis and Oussama Khatib. Control of free-floating humanoid robots through task prioritization. In *IEEE International Conference on Robotics and Automation, ICRA*, 2005. doi:10.1109/ROBOT.2005.1570361.

[152] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE International Conference on Robotics and Automation, ICRA*, 2006. doi:10.1109/ROBOT.2006.1642100.

[153] Bruno Siciliano and Jean-Jacques E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *International Conference on Advanced Robotics (ICAR)*, 1991. doi:10.1109/icar.1991.240390.

[154] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo.

Robotics: Modelling, Planning and Control. Springer Publishing Company, Incorporated, 2008. doi:10.1007/978-1-84628-642-1.

[155] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008. doi:10.1007/978-1-84628-642-1.

[156] Guru Subramani, Michael Gleicher, and Michael Zinn. Recognizing geometric constraints in human demonstrations using force and position signals. *IEEE Robotics and Automation Letters*, 3(2):1252–1259, 2018. doi:10.1109/LRA.2018.2795648.

[157] Akio Sudou. Dynamic hybrid position/force control of robot manipulators—online estimation of unknown constraint. *IEEE Transactions on Robotics and Automation*, 9(2):220–226, 1993. doi:10.1109/70.238286.

[158] M. Svinin, T. Odashima, S. Ohno, Z. W. Luo, and S. Hosoe. An analysis of reaching movements in manipulation of constrained dynamic objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005. doi:10.1109/IROS.2005.1545252.

[159] Chris Towell, Matthew Howard, and Sethu Vijayakumar. Learning nullspace policies. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2010. doi:10.1109/IROS.2010.5650663.

[160] Götz Trenkler, Bernhard Schipp, Heinz Neudecker, and Shuangzhe Liu. Generalized inverses of partitioned matrices. *Econometric Theory*, 9(3):530–533, 1993.

[161] Fan Chung Tseng, Zheng Dong Ma., and Gregory M. Hulbert. Efficient numerical solution of constrained multibody dynamics systems. *Computer Methods in Applied Mechanics and Engineering*, 192(3-4):439–472, 2003. doi:10.1016/S0045-7825(02)00521-2.

[162] F. E. Udwadia and R. E. Kalaba. What is the general form of the explicit equations of motion for constrained mechanical systems? *Journal of Applied Mechanics, Transactions ASME*, 69(3):335–339, 2002. doi:10.1115/1.1459071.

[163] Firdaus Udwadia. A new perspective on the tracking control of nonlinear structural and mechanical systems. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2035):1783–1800, 2003. doi:10.1098/rspa.2002.1062.

[164] Firdaus E. Udwadia and Robert E. Kalaba. *Analytical Dynamics: A New Approach*. Cambridge University Press, 1996. doi:10.1017/CBO9780511665479.

[165] Yoji Umetani and Kazuya Yoshida. Resolved motion rate control of space manipulators with generalized jacobian matrix. *IEEE Transactions on Robotics and Automation*, 5(3):303–314, 1989. doi:10.1109/70.34766.

[166] Sethu Vijayakumar, Aaron D'Souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005. doi:10.1162/089976605774320557.

[167] Luigi Villani and Joris De Schutter. Force control. In *Springer Handbook of Robotics*, pages 195–220. Springer International Publishing, 2016. doi:10.1007/978-3-319-32552-1_9.

[168] Miomir Vukobratovic, Dragoljub Surdilovic, Yury Ekalo, and Dusko Katic. *Dynamics and Robust Control of Robot-Environment Interaction*. WORLD SCIENTIFIC, 2009. doi:10.1142/7017.

[169] Patrick M. Wensing, Luther R. Palmer, and David E. Orin. Efficient recursive dynamics algorithms for operational-space control with application to legged locomotion. *Autonomous Robots*, 38(4):363–381, 2015. doi:10.1007/s10514-015-9420-9.

[170] Daniel E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53, 1969. doi:10.1109/TMMS.1969.299896.

[171] Daniel E. Whitney. Force feedback control of manipulator fine motions. *Journal of Dynamic Systems, Measurement and Control, Transactions*, 99(2):91–97, 1977. doi:10.1115/1.3427095.

[172] Tomasz Winiarski and Adam Woźniak. Indirect force control development procedure. *Robotica*, 31(03):465–478, 2013. doi:10.1017/S0263574712000446.

[173] Haruo Yanai, Kei Takeuchi, and Yoshio Takane. *Projection Matrices, Generalized Inverse Matrices, and Singular Value Decomposition*. Springer New York, 2011. doi:10.1007/978-1-4419-9887-3.

[174] Chenguang Yang, Zhijun Li, and Etienne Burdet. Human like learning algorithm for simultaneous force control and haptic identification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013. doi:10.1109/IROS.2013.6696429.

[175] Xian-Da Zhang. *Matrix Analysis and Applications*. Cambridge University Press, 2017. doi:10.1017/9781108277587.