POLITECNICO
MILANO 1863

School of Industrial and Information Engineering
Master of Science in Automation and Control Engineering

Contact-Implicit Trajectory Optimization for Dynamic Object Manipulation

Master's Thesis of:
Jean-Pierre Sleiman, 10563966

Advisor:
Prof. Paolo Rocco

Co-Advisor:
Prof. Marco Hutter

Academic Year 2017/2018

"It is by logic that we prove, but by intuition that we discover. To know how to criticize is good, to know how to create is better."

– *Henri Poincaré*

Acknowledgements

First of all, I would like to thank my Professors at Politecnico di Milano of the Automation and Control Engineering program as they equipped me with an arsenal of tools that were vital in the development of this work. More specifically, Professor Paolo Rocco, who in the coursework of "Control of Industrial Robots", provided fundamental theoretical concepts underlying my work, and was also my advisor for the thesis.

I would like to also express gratitude to Professor Marco Hutter who hosted me in the Robotics Systems Lab and was my co-advisor at ETH Zürich.

I am thankful to Jan Carius, Ruben Grandia, and Martin Wermelinger who were my mentors at ETH. Their continuous guidance and assistance was extremely helpful throughout this experience.

Most importantly, I am immensely grateful to my family and Mariana for their nurturing love and unparalleled support throughout every step of my journey.

Table of Contents

List of Figures

Abstract

In many robot planning problems, especially those involving locomotion and manipulation, some non-smooth dynamics arise due to contact events, frictional forces, or impacts. Designing optimal controllers to deal with such applications, by relying on classical direct methods for trajectory optimization (TO), could certainly be problematic for the underlying gradient-based optimization solver. One way to incorporate such non-smooth phenomena within our TO formulation is through the contact-implicit optimization (CIO) approach, which along with the multi-phase method, is a prominent trajectory optimization scheme intended to tackle systems with hybrid dynamics. Unlike its counterpart, the contact-implicit method optimizes through contact events, without the need for an a priori definition of a full contact schedule. In this thesis work, a previously developed version of a CIO approach that relies on the description of multi-contact dynamics in terms of a linear complementarity problem (LCP)-solution, accompanied by certain modifications for improved dynamic feasibility, is considered. A resulting mathematical program with complementarity constraints (MPCC) is formulated in order to solve several dynamic object manipulation tasks with a 6-degree of freedom robot. Results are given in terms of a visualization of the optimization's output, and a simulation of the controlled system's evolution. Finally, experimental results are also presented for a manipulation task that involves the manipulator dynamically pushing a block into a desired position that is outside of the robot's workspace, while also taking into account an appropriate dry friction model.

Keywords — *Contact-Implicit Optimization*, *Trajectory Optimization*, *Non-Linear Optimal Control*, *Non-Smooth Contact Dynamics*, *Robotic Manipulation*, *Dynamic Object Manipulation*

Sommario

In molti problemi di pianificazione dei robot, in particolare quelli che coinvolgono la locomozione e la manipolazione, si manifestano alcune dinamiche non regolari a causa di eventi di contatto, forze di attrito o impatti. La progettazione di controllori per gestire tali applicazioni, basandosi su metodi diretti classici per l'ottimizzazione della traiettoria (TO), potrebbe essere problematica nel caso in cui il problema di ottimizzazione sia risolto utilizzando un metodo basato sul gradiente. Un modo per incorporare tali fenomeni non regolari all'interno della nostra formulazione TO è attraverso l'approccio di ottimizzazione implicita del contatto (CIO), che, insieme al metodo multifase, è uno schema promettente di ottimizzazione della traiettoria sviluppato per affrontare sistemi con dinamica ibrida. A differenza della sua controparte, il metodo implicito del contatto è in grado di risolvere il problema senza conoscere a priori la successione degli eventi di contatto. In questo lavoro di tesi, si estende una versione precedentemente sviluppata di un approccio CIO che si basa sulla descrizione della dinamica multi-contatto come soluzione del problema di complementarietÃ lineare (LCP), apportando alcune modifiche per una migliore fattibilità dinamica. Ne risulta un programma matematico con vincoli di complementarità (MPCC), che è stato implementato per risolvere diversi compiti di manipolazione di oggetti dinamici con un robot a 6 gradi di libertà. I risultati sono forniti in termini di visualizzazione dell'output dell'ottimizzazione e di simulazione dell'evoluzione del sistema controllato. Infine, vengono presentati i risultati sperimentali per un compito di manipolazione nel quale il manipolatore spinge un blocco in una posizione desiderata situata al di fuori dell'area di lavoro del robot, tenendo conto di un appropriato modello di attrito a secco.

Parola chiave — *Ottimizzazione Implicita del Contatto*, *Ottimizzazione della Traiettoria*, *Controllo Ottimale Non-Lineare*, *Dinamiche Non-Regolari del Contatto*, *Manipolazione Robotica*, *Manipolazione Dinamica di Oggetti*

List of Symbols and Abbreviations

x	State Variables
q	Generalized Positions
u	Generalized Velocities
$\dot{u}$	Generalized Accelerations
u^-, u^+	Pre- and Post-Impact Generalized Velocities
τ	Control Inputs
Ω	Angular Velocities
$\dot{\Omega}$	Angular Accelerations
$\phi(q)$	Gap Function
λ	Contact Force
Λ	Contact Impulse
P_N	Normal Percussions
$\mathcal{N}_C$	Normal Cone Corresponding to the convex set C
ϵ_N	Newton's Coefficient of Restitution
γ	Relative Separation Velocity
$\dot{\gamma}$	Relative Separation Acceleration
$J_N(q)$	Normal Contact Jacobian
I	Inertia Tensor
$M(q)$	Inertia Matrix
$b(q, \dot{q})$	Coriolis and Centrifugal Terms
$g(q)$	Gravitational Terms
$n(q, \dot{q})$	Nonlinear Effects
I_{closed}	Set of Closed Contacts
Z	All Decision Variables
z	All Decision Variables per Stage
z_{opt}	Optimal Solution
h	Time-Step Size
N	Total Number of Stages
T	Time Horizon
F_f	Frictional Force
F_s	Static Friction Force
μ_s	Coefficient of Static Friction

V^O	Vector Expressed in Frame O
R_B^A	Rotation Matrix Giving the Orientation of Frame B w.r.t Frame A
LCP	Linear Complementarity Problem
DOF	Degrees of Freedom
EOM	Equation of Motion
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
LQR	Linear Quadratic Requlator
MPCC	Mathematical Program with Complimentarity Constraints
NMPC	Nonlinear Model Predictive Control
OC	Optimal Control
TO	Trajectory Optimization
NLOC	Nonlinear Optimal Control
CIO	Contact-Implicit Optimization
NLP	Non-linear Program
IP	Interior-Point Algorithm
SQP	Sequential Quadratic Programming
ROS	Robot Operating System
Z.O.H	Zero Order Hold
ANYpulator	A 6 DOF Robot-Manipulator Built in the Robotic Systems Lab at ETH
ANYpulator-Object Problem	A Dynamic Manipulation Task Performed by ANYpulator on the Object

Chapter 1

Introduction

1.1 Aim and Scope

Throughout the past decade, there has been an ongoing Robot Revolution that keeps on growing at a highly significant pace. A major ingredient in this uprising, is the shift in attention that was mostly directed towards industrial robot applications, – such as pick-and-place, drilling, welding, and many other manufacturing processes – to tasks involving a more dexterous interaction between the robot and its environment, such as those in the field of Locomotion and Dynamic Manipulation. However, the design of motion plans and control policies within the realm of such applications is still considered a particularly young field of research.

The associated challenges are heavily connected to the presence of unavoidable phenomena such as unilateral contacts, friction, and impact events, which render the underlying dynamics non-smooth and discontinuous. In order to deal with such inconveniences, a crucial starting point would be to have a reliable mathematical model that fairly describes the system's evolution over time, as is the case for any control design problem. After that, a problem-dependent control technique is utilized as a tool for deriving a proper control law.

In the case of controlling smooth, nonlinear, under-actuated robots, state-of-the-art techniques heavily rely on Trajectory Optimization (TO) methods, which turn out to be remarkably convenient and favorable when compared with other approaches; as they exploit the power of numerical optimization for systematically formulating and solving such problems (while also respecting input and state constraints implicitly). Building on this, extensions of traditional TO methods have been made to incorporate Hybrid systems as well. These can be classified into two main categories:

The first one is the *Multi-Phase approach*, which in brief terms, aims to identify the points at which the discrete transitions occur (those are triggered by contact events in our case), based on an a priori knowledge of the mode sequence; then the trajectories leading to and emerging from the discrete events are optimized separately. The second one is the mode-invariant method (also referred to as the *Contact-Implicit Approach* when the non-smoothness is a result of contact dynamics); with this formulation, a pre-specified mode schedule is not required anymore, but rather comes out as a result of the optimization itself which searches for an optimal trajectory *through the contacts.* However, multi-phase methods tend to produce higher-accuracy optimal trajectories that are more compatible with the dynamics of the system and thus are more dynamically feasible than those obtained from the contact-implicit approach.

For the sake of narrowing down the scope, this thesis work – which was carried out in the Robotic Systems Lab (RSL) at ETH Zürich – will make use of a variation on the contact-implicit optimization (CIO) formulation proposed by Posa et al. [39], in order to ultimately solve a number of dynamic object manipulation problems. Such applications belong to the family of graspless (also referred to as nonprehensile) manipulation, and they generally include tasks such as pushing, sliding, tumbling, pivoting and so on [25]. It is important to differentiate between dynamic manipulation, and quasi-static nonprehensile manipulation, where the latter involves relatively slow motions such that the inertial effects can be considered negligible and the robot is assumed to be in contact with the object for the whole time horizon. Whereas, in the former, these assumptions do not necessarily hold, meaning that it is possible for the robot to lose contact with the object during the desired manipulation period. This, in turn, makes it mandatory for one to reason about the dynamics induced by the environment on the uncontrollable object, before planning for an optimal robot motion that moves it from an initial state to a final desired one.

To be more specific, the modified CIO approach will be used along with a stabilizing linear feedback law, to control a 6-DOF robot referred to as ANYpulator – developed at the Robotic Systems Lab – in order to dynamically push a block from an initial configuration to a final assigned position that could be outside of the robot's workspace. The method will

be tested in simulation as well as on the real system. Other 2D preliminary examples (visualizations only) and 3D preparatory examples (simulations only) will be tackled throughout the development of the thesis. To elaborate, the dynamic equations of motion for ANYpulator will be derived using the *MATLAB Symbolic Math Toolbox*, and the finite-dimensional optimization problem will be formulated on *MATLAB*, but under the framework of *FORCES Pro*, which is a commercial tool aimed at generating highly customized and efficient optimization solvers (in the form of C code) based on an efficient implementation of the interior point algorithm [15] [51]. Finally, a C++ class structure will be developed along with the use of libraries and tools provided by the *Robot Operating System (ROS)* middleware, to be able to test the manipulation tasks within the *Gazebo Simulation Environment* as well as on the real system.

From this work, it will be demonstrated that our adopted method achieves optimal state-trajectories that are dynamically feasible, in the sense that they can be easily stabilized and tracked with a fairly simple linear feedback law, added to the optimal input sequence – given by the CIO program output – as a feedforward term. Moreover, the resulting motion plan is shown to adhere to the imposed contact conditions and results in realistic contact forces. These will indeed be the crucial ingredients for successfully attaining the desired manipulation-task goal, without the need for any sort of post-optimization modifications. As a result, shifting from a numerical implementation through simulation, onto an experimental real-time implementation on the real setup, turns out to be adequately straightforward. It is the first time that such a method is applied in practice on a dynamic object manipulation task.

1.2 Outline

Chapters 2 and 3 provide the reader with a theoretical build-up towards the adopted contact-invariant trajectory optimization approach, while also motivating the use of this methodology for optimal control problems involving hybrid system dynamics. More specifically, Chapter 2 gives an overview about the modelling of contact dynamics, before moving on to discussing the most prominent numerical simulation schemes applied to systems where set-valued force laws arise due to unilateral hard contacts,

and frictional forces: namely the *event-driven methods* and *time-stepping techniques*. It is discussed also how these set-valued laws can be resolved either by relying on the augmented Lagrangian approach or by solving a linear complementarity problem (LCP).

Chapter 3 motivates the introduction of trajectory optimization techniques, and explores some of the underlying fundamental concepts while elaborating mostly on the direct methods. In Chapter 4, a general inspection is made on related work concerning the *Multi-Phase method* and the different versions of *Contact-Implicit Optimization* approaches adopted in the robotics and computer graphics communities. Moreover, the original CIO version that initiated this work is elaborated upon, before moving on to analyze and explain about modified CIO approach that was eventually adopted in the thesis within the *FORCES Pro* framework. Two preliminary 2D examples are also briefly investigated in this chapter.

Then we move to Chapter 5, where the 6-DOF robot's equations of motion are derived, and some general considerations required for building the CIO program, aimed at solving the ANYpulator-Object problem, are presented. Furthermore, two 3D preparatory examples are tackled and simulation results are shown; that is done before concluding the chapter with an exploration of an NMPC (nonlinear model predictive control) method. Finally, the central dynamic object manipulation application (ANYpulator-Block Problem) is dealt with thoroughly in Chapter 6; a Coulomb friction model is incorporated into our optimal control formulation, and both simulations, as well as experimental results, are provided and assessed carefully. The thesis work is concluded in Chapter 7.

Chapter 2

Non-Smooth Contact Dynamics

2.1 Modeling of Contact Dynamics

It is undeniable that the analysis and modeling of frictional or frictionless contact phenomena between deformable solid bodies are best captured in the realm of solid mechanics. However, this framework generally yields a set of partial-differential equations to be solved analytically or numerically with finite element methods; and this could become problematic when high-accuracy of the resulting contact model is not a main concern, such as in dynamic simulation and control applications, where efficiency (a compromise between accuracy and speed) is a primary aspect. In such cases, a suitable dynamic modelling approach for the proper description of the overall system behavior, could be given by either one of two ways: A *soft contact model* or a *hard contact model* (see Figure 2.1)

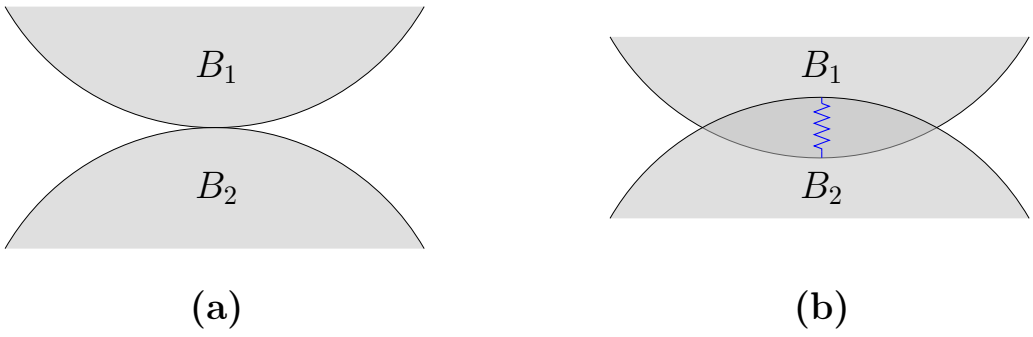


Figure 2.1: **(a)** Hard-Contact Model **(b)** Soft-Contact Model

With the first one (also referred to as a regularization approach), it is assumed that the bodies are deformable, but instead of relying on a distributed-parameter system model, a lumped-parameter system is considered [46] (*i.e.* Rigid body along with a set of virtual springs on its boundary, that would continuously act on any other potentially contacting body). The advantage here is that a single-valued force law is obtained, meaning that

it should be enough to fully describe the system's evolution by numerically integrating a set of smooth ordinary differential equations (ODE). The apparent downside is related to the fact that the resulting ODE's tend to be very stiff (due to the high-frequency oscillations induced by the high-stiffness virtual springs), and this could certainly lead to a high computational burden (or to numerical instability in case an inappropriate integration scheme was chosen). Other misfortunes associated with a soft contact model are: The permittance of unphysical penetration between contacting bodies and the unavoidable difficulty that comes with tuning the parameters of the virtual passive elements.

The second method relies on the multi-rigid-body contact formulation, where a non-interpenetration "law" governs the system's evolution immediately before, immediately after, and during contact. To elaborate, similar to how *bilateral constraints* (two-sided, equalities) impose a relationship between the states $q(.)$ and $u(.)$ (whether a set of minimal or non-minimal coordinates was chosen), hard-contact *unilateral constraints* (one-sided, inequalities) add a restriction on the states that would prevent penetration between any two bodies. As a consequence, the generalized positions are only required to be *absolutely continuous* (which is a stronger notion than continuity) functions of time, while the velocities are functions of *bounded variation* (*i.e.* They admit a countable set of discontinuities as well as an existing right and left limit $\forall\ t$). In fact, jumps in velocities arise in so-called *impact events*, anytime two bodies meet with a relative velocity that is inconsistent with the non-penetration condition; and such abrupt changes can only be caused by non-finite, impulsive forces (accelerations are undefined at such time instants).

Unlike bilateral constraints which yield a set of Differential Algebraic Equations (DAE) when added to the equations of motion, unilateral contacts are characterized by a set-valued force law (the same goes for friction modelling, where the friction force would be acting in the tangent contact plane), and this is indeed where the *non-smoothness* of the problem originates from. Also in the case of possible impacts, an additional set-valued impact law has to be included as well.

2.2 Simulation of Non-Smooth Dynamics

When it comes to numerical simulation, the considerations typically made on accuracy, stability, and convergence for basic integration methods (for instance the Runge-Kutta schemes), only apply when the ODE at hand (an autonomous system is assumed for now just for the sake of simplicity and conciseness but without any loss of generality), of the form:

$$\dot{x}(t) = f(x(t)) \tag{2.1}$$

has a continuous forcing function $f(x(t))$ (more precisely, it should be Lipschitz continuous). Otherwise, if it was discontinuous and bounded (as shown in Figure 2.2), then one could easily see how the evaluation of an integration scheme fails, by looking at the Taylor series expansion about a point of interest t_k (the expansion is used when deriving the order of the method through the local truncation error-expression):

$$x(t_{k+1}) = x(t_k) + h\dot{x}(t_k) + \frac{1}{2}h^2\ddot{x}(t_k) + \mathcal{O}(h^3)$$

The above Taylor polynomial approximation undoubtedly leads to inaccurate depictions of the real solution in a neighborhood of the discontinuity, as the second and higher order derivatives of $x(.)$ are not bounded.
Such difficulties made the design of numerical methods for non-smooth dynamical systems a field of study on its own; and the literature on that is essentially divided between two approaches:

- The *event-driven* scheme [44]: This consists of using any of the high-order integration methods for ODE's or DAE's, while at the same time, implementing an event-detection step. Every time an event is detected (by keeping track of the status of some switching function), the problem is re-initialized and the integration process is resumed. An accurate detection of the discontinuity at $\bar{t}$ (see Figure 2.2) is key to guaranteeing an overall order similar to that already associated with the numerical scheme (had it been applied to a smooth set of ODE's). The drawback of this approach is in the difficulty of achieving the detection-accuracy requirement, especially in cases involving multiple discrete events.

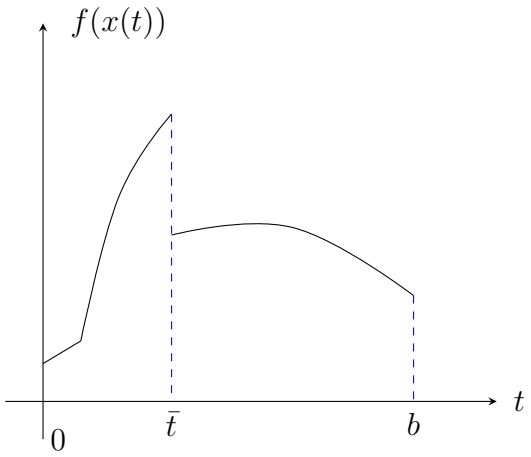


Figure 2.2: Non-Smooth Discontinuous Dynamics

– The *time-stepping* (or *event-capturing*) scheme: This involves the full discretization of the non-smooth system dynamics (by discretizing the *equality of measures* as will be shown later) in addition to the incorporation of the set-valued laws. The discretization is independent from the switching points; so the method is able to accommodate a large number of contact/discrete events without the need for any detection plan. However, on the downside, this also means that the accuracy attained by the use of high-order integrators is now lost, and the best one could generally achieve here is a local truncation error of $\mathcal{O}(h)$ (similar to that given by *Forward or Backward Euler*) regardless of the order of the consistent discretization method applied. The most prominent time-stepping scheme is that given by J.J. Moreau [31].

In the case of non-smoothness, without any consideration of possible impact events (for now), the set-valued force laws can be written in terms of *normal cone inclusions*:

Definition 2.2.1. *A normal cone is given by the following set:*

$$\mathcal{N}_C(x) = \{y \mid y^T(\hat{x} - x) \leq 0; \forall \hat{x} \in C; x \in C\} \tag{2.2}$$

where C is any convex set.

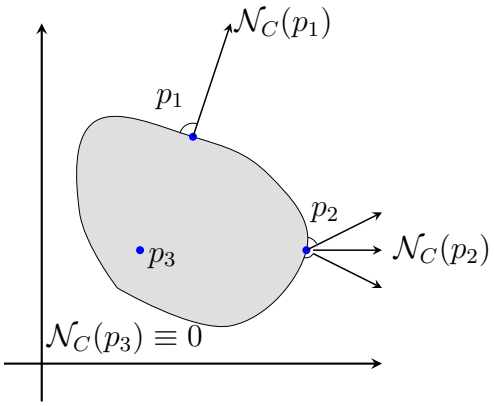


Figure 2.3: Normal Cone [47]

The interpretation of this set definition is that it contains all vectors y making an obtuse angle (as shown in Figure 2.3) with all other vectors $(\hat{x} - x)$ for $x \in C$, $\forall \hat{x} \in C$ (for more details on convex analysis see [41]).

To illustrate, consider a generalized-position-dependent gap function $\phi(q)$ which can be represented by a signed-distance function between two potentially contacting bodies ($\phi(q) < 0$ indicates that the objects are in penetration mode). The following normal cone inclusion for the contact force λ (assuming a single contact point: λ, $\phi(q) \in \mathbb{R}$):

$$-\lambda \in \mathcal{N}_{\mathbb{R}_0^+}(\phi(q)) \tag{2.3}$$

is equivalent to having no contact force when the objects are not in contact, and an arbitrary pushing-force when the distance between them is closed, *i.e.*

$$\begin{aligned} \phi(q) > 0 \;&\Rightarrow\; \lambda = 0 \\ \phi(q) = 0 \;&\Rightarrow\; \lambda \geq 0 \end{aligned} \tag{2.4}$$

Another set-valued force law can be given on the acceleration level, with the relative (separation) acceleration denoted by $\dot{\gamma}$. This is actually the one needed to be jointly considered along with the dynamic equations of motion when solving for the full non-smooth dynamics (except in the case of time-stepping schemes which require the set-valued impulse law on the velocity level instead, as will be thoroughly discussed later):

$$\phi(q) = 0 : \qquad -\dot{\gamma} \in \mathcal{N}_{\mathbb{R}_0^+}(\lambda) \tag{2.5}$$

This implies that during an already closed contact, the contact remains closed in the case of a non-vanishing contact force, and may or may not open otherwise. In the multi-contact case, the three relationships above hold element-wise.

Such normal cone inclusions can be dealt with through an *augmented Lagrangian* approach, where the saddle point conditions for the corresponding augmented Lagrangian function yield a set of non-linear equations, specifically a proximal point problem, that can be solved with Jacobi or Gauss Siedel like iterative schemes *(JORprox or SORprox)* [47].

Alternatively, one could take a different direction, by writing the inclusions as a *linear complementarity problem (LCP)* (or non-linear complementarity problem that can be solved using a sequence of LCP's) [45]. A comprehensive introduction for linear complementarity problems can be found in [9].

Definition 2.2.2. *Given two vectors w, $z \in \mathbb{R}^n$, a complementarity condition between them is given by the following:*

$$0 \leq w \perp z \geq 0 \quad \Longleftrightarrow \quad \begin{cases} w \geq 0 \\ z \geq 0 \\ w^T z = 0 \end{cases} \tag{2.6}$$

Definition 2.2.3. *Given two vectors w, $z(w) \in \mathbb{R}^n$ s.t. $z = Mw + d$, and solving a linear complementarity problem denoted by LCP(M,d) in this case, returns the pair (w,z) that satisfies the equality as well as the complementarity condition between w and $Mw + d$*

One could clearly show from the combination of the general rigid-body system dynamic equations (in joint-space coordinates) and the set-valued contact force law, that the multi-contact dynamic problem can be formulated as an LCP whose solution resolves the contact force vector λ and its complement, the relative acceleration $\dot{\gamma}$ between the contacting bodies (see [17] for an extended proof and for more details on LCP solution-existence):

$$\begin{cases} find & \lambda, \dot{\gamma} \\ s.t. & \dot{\gamma} = M\lambda + d \\ & \dot{\gamma} \geq 0 \\ & \lambda \geq 0 \\ & \dot{\gamma}^T \lambda = 0 \end{cases} \tag{2.7}$$

$$\Updownarrow$$

$$\begin{cases} \underset{\lambda}{argmin} & \frac{1}{2}\lambda^T M \lambda + \lambda^T d \\ s.t. & \lambda \geq 0 \end{cases} \tag{2.8}$$

Notice the similarity between the complementarity condition in (2.7) and the normal cone inclusion in (2.5). On the other hand, the equivalence between (2.7) and (2.8) can be seen by writing down the first-order necessary Karush-Kuhn-Tucker (KKT) conditions for optimality of the inequality constrained optimization problem given in (2.8):

$$\nabla_\lambda \mathcal{L}(\lambda, \mu) = \nabla_\lambda \left(\frac{1}{2}\lambda^T M \lambda + \lambda^T d - \mu\lambda \right) = 0 \tag{2.9}$$

$$\Rightarrow \quad \mu = M\lambda + d \qquad \textit{(Gradient of the Lagrangian function)}$$

$$\mu \geq 0 \qquad \textit{(Lagrange multipliers)} \tag{2.10}$$

$$\mu^T \lambda = 0 \qquad \textit{(Complementarity slackness condition)} \tag{2.11}$$

where the Lagrange multiplier μ plays the same role as the relative acceleration $\dot{\gamma}$.

This ability to transform LCP's into a Quadratic Program (QP) is very convenient, as robust QP solvers are widely available. Other ways for solving LCP's, such as Lemke's method, are also discussed in [9].

It is important to note that both the *augmented Lagrangian* and the *linear complementarity* approach could be used with either one of the two aforementioned numerical integration schemes for non-smooth systems.

Now when impulsive forces come into play due to possible impact events, as mentioned before, an additional impact law is required. Generally, two impact models are proposed in the literature: *Poisson's impact law* and *Newton's restitution law*. In this thesis, only the second one is dealt with extensively (refer to Figure 2.4 for a graphical representation of this law):

$$\phi(q) = 0 : \qquad -(\gamma_N^+ + \epsilon_N \gamma_N^-) \in \mathcal{N}_{\mathbb{R}_0^+}(\Lambda_N) \tag{2.12}$$

Meaning that for a closed contact, the following holds:

$$\begin{aligned} &\textbf{\textit{Contact transmits Impulse:}} && \Lambda_N > 0 \quad \Rightarrow \quad \gamma_N^+ = -\epsilon_N \gamma_N^- \\ &\textbf{\textit{Superfluous Contact:}} && \Lambda_N = 0 \quad \Rightarrow \quad \gamma_N^+ \geq -\epsilon_N \gamma_N^- \end{aligned} \tag{2.13}$$

where Λ_N is the impulsive force, γ_N^+/γ_N^- are the post/pre-impact relative velocities and ϵ_N is Newton's coefficient of restitution.

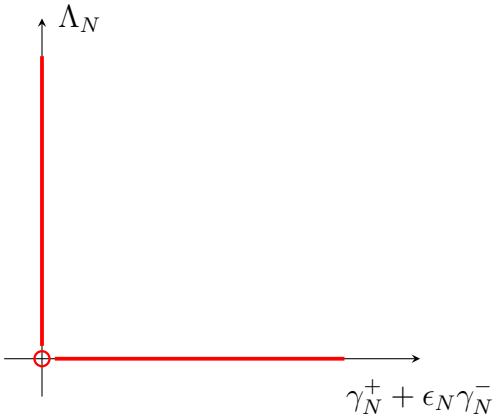


Figure 2.4: Newton's Impact Law

The first case covers a majority of the situations, and what it entails is basically the reversal of the post-impact relative velocity with respect to the one at pre-impact, along with a dampening effect given by the Newton coefficient. It is assumed from now on that $\epsilon_N = 0$ in order to ensure a totally inelastic impact event (no rebound).

The second case typically occurs in multi-contact situations, when a contact does not participate in the impact, and thus making it safe to assume that the corresponding contact was not present at this instant (it is unnecessary and its absence does not change the state evolution). See for instance the Rocking Rod example in [37].

To quickly demonstrate how this impact law can be used in the context of event-driven schemes, consider the computation of the post-impact joint velocities for a manipulator (which are used in the re-initialization step of the problem). Starting from the Impulse-Momentum equation for a multi-rigid-body system (frictionless contact is assumed)

$$M(q)(u^+ - u^-) = J_N^T(q)\Lambda_N \tag{2.14}$$

where M(q) is the generalized mass matrix, and $J_N(q)$ *the normal Jacobian which is given by*

$$J_N^T(q) = \frac{\partial \phi(q)}{\partial q} \tag{2.15}$$

and adding to that, Newton's restitution law for a participating contact,

$$\gamma_N^+ = \frac{d\phi(q)}{dt}^+ = J_N(q)u^+ = 0 \tag{2.16}$$

One could derive the expression for u^+ in terms of u^- and other known quantities as follows:

- *Multiply both sides of (2.14) by* $J_N(q)M(q)^{-1}$ *from the left*
- *This yields an expression for* Λ_N *in terms of* u^-

$$\Lambda_N = -\left(J_N M^{-1} J_N^T\right)^{-1} J_N \; u^- \tag{2.17}$$

- *Replace (2.17) in (2.14) to obtain the final expression*

$$u^+ = \left(I - M^{-1} J_N^T \left(J_N M^{-1} J_N^T\right)^{-1} J_N\right) u^- \tag{2.18}$$

As for the integration of Newton's impact law together with time-stepping schemes, this turns out to be highly convenient, due to the observation that

the time-stepping technique does not differentiate between finite forces and impulsive forces. That's because only the integral of the contact force function is evaluated over every individual time-step, and not its instantaneous value. This, in turn, avoids the "infinitely large" quantities resulting from impact events, no matter how small the chosen integration step was (think of the integral of a Dirac Delta distribution). As a consequence, the main variables of interest in the integration scheme become the velocities and the impulses instead of accelerations and forces.

As mentioned previously while introducing the time-stepping approach, the discretization of the non-smooth dynamics is carried out by discretizing the *equality of measures*. What was not mentioned is the fact that this is truly needed only in the presence of impulsive forces since in their absence, forces and accelerations are already well-defined and therefore discretizing the equations of motion should be enough.

In the following, it is shown how one could arrive at an equality of measures starting from the EOM and the impact equation; before moving on to the application of a time-stepping scheme (namely Moreau's suggested plan [31]) for the discretization procedure.

$$M(q)\dot{u} + n(q,u) = \sum_i J_{N_i}^T \lambda_{N_i} \tag{2.19}$$

$$M(q)(u^+ - u^-) = \sum_i J_{N_i}^T \Lambda_{N_i} \tag{2.20}$$

where n(q,u) is the vector of generalized Coriolis, centrifugal and gravitational effects (could also include viscous friction terms), the summation is carried out over the set of all possible contacts.

Now we multiply the equations of motion by dt, the impact equation by $d\eta$ (a Dirac point measure for a velocity discontinuity) and sum them up to acquire the equality of measures

$$M(q)du + h(q,u)dt = \sum_i J_{N_i}^T dP_{N_i} \tag{2.21}$$

with

$$du = \underbrace{\dot{u}\,dt}_{Lebesgue\ integrable\ term} + \underbrace{(u^+ - u^-)d\eta}_{atomic\ term} \tag{2.22}$$

$$dP_{N_i} \quad = \quad \underbrace{\lambda_{N_i}\, dt}_{Lebesgue\ integrable\ term} \quad + \quad \underbrace{\Lambda_{N_i} d\eta}_{atomic\ term} \tag{2.23}$$

note that $\int d\eta = 1$ *at the time of a discontinuity and is zero elsewhere*

Integrating the equality of measures between the initial time of an interval and the final time, we get

$$\int\limits_{[t_k,t_{k+1}]} M(q)du + \int\limits_{t_k}^{t_{k+1}} h(q,u)dt = \int\limits_{[t_k,t_{k+1}]} \sum_i J_{N_i}^T dP_{N_i} \tag{2.24}$$

The trick now is to approximate the integrals with an appropriate quadrature rule (M(q) and h(q,u) are assumed to be non-varying with respect to the positions q over the whole interval). The difference mainly lies in where one decides to fix the integrands. So in general one has

$$\int\limits_{[t_k,t_{k+1}]} M(q)du \approx M(q_{k+\alpha})(u_{k+1} - u_k) \tag{2.25}$$

And,

$$\int\limits_{[t_k,t_{k+1}]} h(q,u)dt \approx h(q_{k+\alpha}, u_k)\Delta t \tag{2.26}$$

where $\alpha \in$ *[0,1]*

The famous Moreau time-stepping scheme uses $\alpha = 0.5$, and so this results in the following concluding relationships that form our *non-smooth, frictionless, impulsive contact problem:*

$$\begin{cases} M(q_{k+0.5})(u_{k+1} - u_k) + h(q_{k+0.5}, u_k)\Delta t = \sum\limits_i J_{N_i}^T P_{N_i} \\ \\ -(\gamma_{Ni_{k+1}} + \epsilon_N \gamma_{Ni_k}) \in \mathcal{N}_{\mathbb{R}_0^+}(P_{N_i}) \end{cases} \tag{2.27}$$

where $q_{k+0.5} = q_k + \frac{1}{2} u_k \Delta t$ *and where* i *belongs to the set of indices corresponding to those contacts which are closed at the middle of the time-step:* $i \in \mathcal{I}_{closed} = \{i \mid \phi_i(q_{k+0.5}) \leq 0\}$

The system (2.27) can be solved using one of the methods discussed previously – using the augmented Lagrangian approach or using the system's formulation as a complementarity problem. After resolving the so-called *normal Percussions* P_{Ni} as well as the generalized velocities at t_{k+1}, one could proceed by computing the generalized positions as such:

$$q_{k+1} = q_{k+0.5} + \frac{1}{2} u_{k+1} \Delta t \tag{2.28}$$

A final note would be that even though frictionless contact was assumed so far, adding friction to the problem would not lead to major changes in the formulation (an additional set-valued law is required, along with the addition of the tangential Percussions to the equality of measures). The reason for not incorporating friction in the above demonstrations is due to the fact that the tangential components of the contact forces/impulses do not play a significant role in the applications that are dealt with in this thesis, but are rather dominated by the normal terms.

As mentioned in the outline section of Chapter 1, the concepts introduced in this chapter and those that will be discussed in the upcoming one, will be very useful for forming a fairly clear and complete picture of the developments that will be done in Chapter 4 when demonstrating the central approach underlying the thesis.

Chapter 3

Trajectory Optimization

3.1 Optimal Control Problem Motivation

The use of linear control methods, whether it was for stabilizing unstable systems at an equilibrium/trajectory (regularization/tracking) or for attaining a desired performance in terms of speed and robustness, has been extremely widespread within the control community to an extent that in most industrial applications, it is generally enough to apply a PID-type control law plus a linear feedforward term along with some minor tweaks. This is indeed true in some applications which involve non-linear dynamics as well (the desired equilibrium is stabilized locally, with a certain region of attraction). In fact, even in the case where strong non-linearities are present, a very famous state-feedback non-linear control approach would be to linearize the dynamics by brute-force through the *feedback-linearization* technique (sometimes referred to as *inverse dynamics* in the robotics community), and then to close another outer-loop that controls the resulting linear system with a PID-like scheme. However, one issue with feedback-linearization typically arises when the dynamical system at hand is under-actuated; that is due to the possibility that the system would be only partially-feedback-linearizable thereby leaving behind some *hidden, zero-dynamics* that could render the full closed-loop system unstable (see [22] for more details). Another more generic problem with the aforementioned methods is related to the observation that such control laws usually require an unnecessary, and yet unavoidable, high control effort. These are some of the justifications for the importance of introducing *Trajectory Optimization* methods, also referred to as *Optimal Control (OC)* methods.

Trajectory Optimization (TO) is a very popular and powerful framework that is adopted in multiple engineering domains; such as in chemical plants, in aerospace applications, in robotic applications... This control synthesis

approach is based on the formulation of a control problem in terms of an optimization one; and its beauty lies in its ability to handle a very broad spectrum of problems (*i.e.* linear/non-linear plants, deterministic/stochastic systems, continuous/discrete systems, presence of state and input constraints...). Whereas its strength lies in its systematic structure which mainly relies on numerical solutions rather than mathematically-tedious analytical ones, meaning that its effectiveness is purely based on the validity of the problem formulation and on the proper use of an efficient *Non-linear Programming (NLP)* solver.

The simplest form of a finite-horizon Optimal Control Problem formulation is given by the following system:

$$\begin{cases} \underset{\tau(.)}{\boldsymbol{min}} \quad J(x(.),\tau(.),t) = \underbrace{m(x(T))}_{Mayer\ term} + \underbrace{\int_0^T L(\tau(t),x(t))dt}_{Lagrange\ term} \\ \boldsymbol{s.t.} \quad x(0) = x_0 \\ \qquad\quad \dot{x}(t) = f(x(t),\tau(t)) \\ \forall t \in [0,T] \end{cases} \tag{3.1}$$

where $J(x(.),\tau(.),t)$ is the total cost formed by summing up a terminal cost with a path-integral cost, $\tau(.)$ is the control input, and where f is assumed to be a continuously differentiable function with respect to its arguments (needed to derive the optimality conditions)

3.2 Dynamic Programming and Indirect Methods

Different optimal control methods exist for approaching such a problem, of which the main ones are highlighted in the diagram of Figure 3.1.

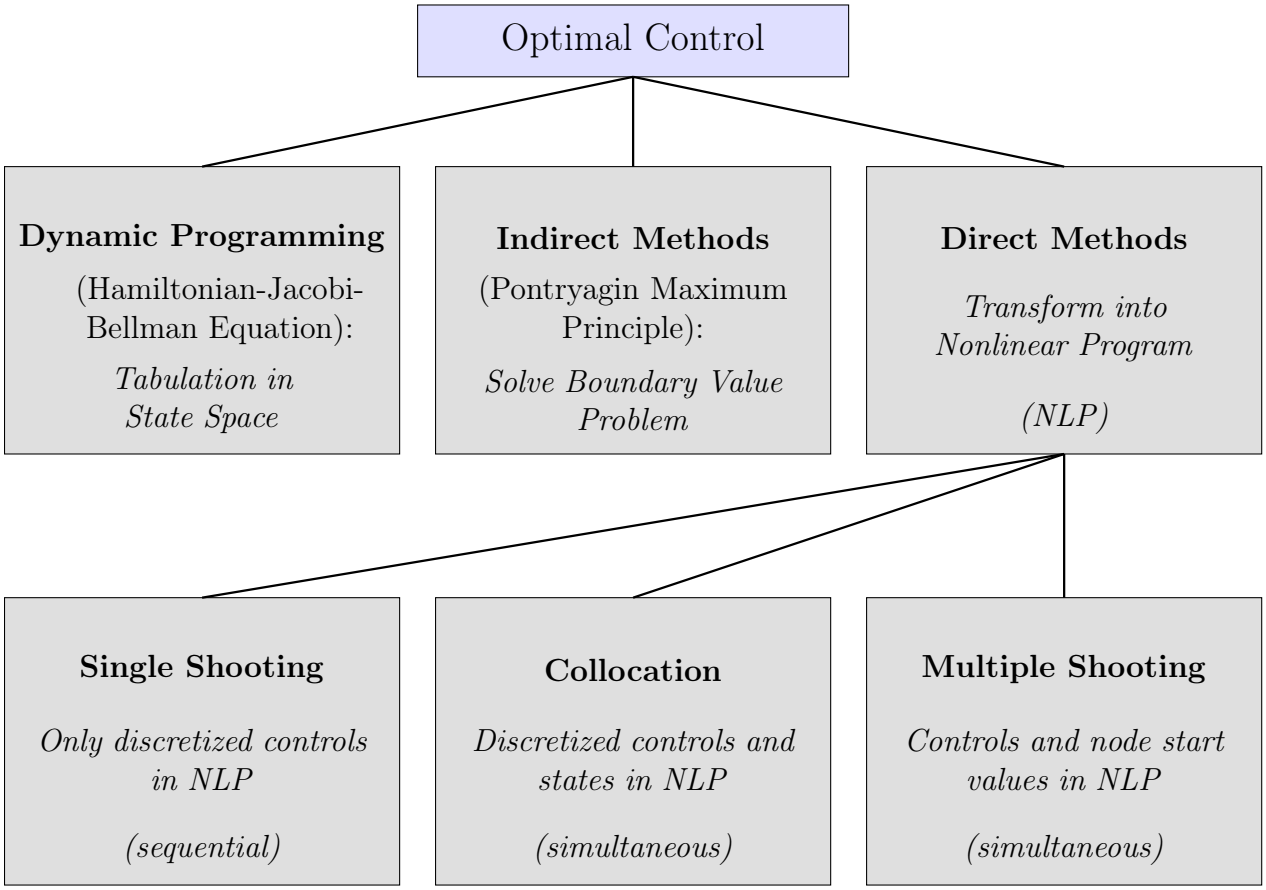


Figure 3.1: The Optimal Control Family Tree [12]

To begin with, it is worth mentioning an essential principle underlying a significant part of the upcoming analysis, and that is the *Dynamic Programming Principle*, also called *Bellman's Principle of Optimality*:

"From any point of an optimal trajectory, the remaining trajectory is optimal for the corresponding problem over the remaining number of stages, or time interval, initiated at that point" [26].

Ultimately, the goal is to obtain a globally optimal, stabilizing control policy $\tau_{opt} = \kappa(x, t)$ (notice here the dependence on the states). In other words, we would like to obtain at best, a closed form analytical solution

for the control law that stabilizes a desired equilibrium or trajectory, while also globally minimizing the cost functional (thus, for example, minimizing energy). One such law can be derived (by building upon the principle mentioned above) for a special class of problems involving a quadratic cost, accompanied by a linear time-invariant (LTI) system. The resulting controller is actually well-known in the control literature, and it is the linear-quadratic-regulator (LQR). However, such closed-form solutions are typically hard to derive; and it is quite important to understand what underlies this claim, before moving on to explain the remedy for such a complication.

Let us assume first that we loosen the requirement of an analytical expression for the globally optimal policy, by keeping only the global optimality part, but while using a set of discrete states (belonging to a finite state space), a set of discrete inputs (belonging to a finite action space) and of course a set of discrete times. In that case, what one has to basically solve is a *graph-search problem* and this can be done with the *dynamic programming algorithm*, or with other optimal "shortest-path" methods such as *Dijkstra's algorithm, A* search...* Clearly, the downside lies in the "curse of dimensionality" as such an approach holds a computational complexity that grows exponentially with respect to the number of states, so it would be best utilized when the state-space dimension is typically less than 3 [49].

Now if we go back to the original continuous-time problem formulation introduced in (3.1), while also referring to Bellman's principle of optimality, one could show after some extensive derivations (refer to a text on advanced control such as [26]) that the *sufficient conditions* for optimality are given by the following set of equations:

$$\begin{cases} -\dfrac{\delta J_{opt}(x,t)}{\delta t} = \underset{\tau(.)}{\boldsymbol{min}} \;\; L(x,\tau) + \dfrac{\delta J_{opt}(x,t)}{\delta x} f(x,\tau) \\ \\ J_{opt}(x,T) = m(x) \end{cases} \tag{3.2}$$

These equations are called the *Hamilton-Jacobi-Bellman (HJB)* equations of optimal control. The steps to solve them basically include the derivation of the optimal control law

$$\begin{aligned}\tau_{opt}(t) &= \underset{\tau(.)}{\boldsymbol{argmin}} \;\; L(x,\tau) + \frac{\delta J_{opt}(x,t)}{\delta x} f(x,\tau) \\ &= \kappa\left(x, \frac{\delta J_{opt}(x,t)}{\delta x}\right)\end{aligned} \tag{3.3}$$

then one could replace this expression in (3.2) to eventually obtain a *non-linear partial differential equation (PDE)*, with the boundary condition given in (3.2). It should be clear by now, why it is almost impossible to obtain this closed-form analytical optimal policy that we strive for in most generic OC problems (LQR is one example where the resulting PDE turns out to be solvable).

One way to avoid this inconvenience is through the so-called *indirect methods*, where the *necessary conditions* for optimality of the problem defined in (3.1) (*Pontryagin's minimum principle)* can be reproduced from the HJB equations (or the Euler-Lagrange equations) with a few mathematical manipulations

Define the adjoint (costate) variables $\nu(t)$ *as such:*

$$\nu(t) = \left(\frac{\delta J(x,t)}{\delta x}\right)^T \tag{3.4}$$

Define the Hamiltonian function $H(x,\tau,\nu)$

$$H(x,\tau,\nu) = L(x,\tau) + \nu^T f(x,\tau) \tag{3.5}$$

Pontryagin's Minimum Principle:

$$\begin{cases} \dot{x}(t) = \nabla_\nu H(x,\tau,\nu) & \textit{dynamic equations} \\ \dot{\nu}(t) = \nabla_x H(x,\tau,\nu) & \textit{adjoint equations} \\ 0 \quad\;\; = \nabla_\tau H(x,\tau,\nu) & \textit{control equations} \\ x(0) = x_0 & \textit{initial conditions} \\ \nu(T) = \nabla_x m(x(T)) & \textit{terminal conditions} \end{cases} \tag{3.6}$$

The system given by (3.6) is what is referred to as a *two-point boundary value problem (TPBVP)*, which is different and more difficult to solve when

compared to an *initial-value problem (IVP)*, but still simpler to numerically deal with than nonlinear PDE's. As a matter of fact, any of the numerical methods that are typically used to solve an OC problem with the direct approach (see Figure 3.1) can also be used for the indirect problem, but the context in which they are applied differs. To clarify, the indirect method **optimizes** then **discretizes** with these numerical techniques, while the direct one **discretizes** then **optimizes** with these techniques. Therefore, one could reasonably ask: Why the introduction of direct methods?

The answer is simply because they can find optimal solutions more efficiently than their counterparts (for reasons related to the difficulty of simulating both the dynamic and adjoint equations forward in time as one of them is usually unstable if the other is stable); and also due to the fact that indirect methods find it more difficult to deal with path inequalities. The only advantage of the indirect approach over the direct one is that when it successfully converges to a solution, one knows for sure that it is indeed globally optimal; that is unlike direct methods which typically converge to a local optimum (for more explanation see [4] [12]). An important final remark before moving on to the direct approach would be that since our control policy is being computed with an offline numerical program, what we ultimately get is an optimal, state-independent, control time-sequence; meaning that we have obtained an open-loop control law, and that is certainly not enough for robustly controlling our system. Therefore, either a stabilizing feedback term is added, or the OC problem is solved online (such as in *Model Predictive Control*); and this can only work if the adopted methods are "fast-enough" (this is also why direct methods highly thrive in the control community).

3.3 Direct Methods

As formerly mentioned, these methods initially *discretize* the infinite-dimensional optimization problem thereby turning it into a finite-dimensional problem, or non-linear program (NLP) (this process is called *transcription*), and then they *optimize* the resulting structure. Let us now consider a more general Optimal Control formulation than that given in (3.1):

$$\begin{cases} \underset{\tau(.)}{\boldsymbol{min}} & J(x(.),\tau(.),t) = m(x(T)) + \int_0^T L(\tau(t),x(t))dt \\ \boldsymbol{s.t.} & \dot{x}(t) = f(x(t),\tau(t)) \\ & h_{min} \leq h(x(t),\tau(t)) \leq h_{max} \\ & g(x(t),\tau(t)) = 0 \\ \forall t \in [0,T] \end{cases} \tag{3.7}$$

Note that boundary conditions as well as upper and lower bounds on the states/inputs can also be considered from (3.7)

- ***Shooting Method:*** One way to approach this generic problem is by discretizing the control inputs $\tau(.)$ throughout the whole time horizon. Then to use, for instance, a piecewise function for the controls (given by an initial guess) while integrating the dynamics forward in time (shooting). Then an NLP containing as decision variables $Z = [\tau_1 \ \ \tau_2 \ \ ... \ \ \tau_N]$ is solved such that the cost function is minimized, the constraints on the inputs are satisfied, and the boundary conditions on the states are attained. The resulting optimization problem looks as follows:

$$\begin{cases} \underset{\tau_1,...,\tau_N}{\boldsymbol{min}} & \sum_{k=1}^{N} F(\tau_k) \\ \boldsymbol{s.t.} & x_N = c(x_0,\tau_k) \\ & h_{min} \leq h(\tau_k) \leq h_{max} \\ & g(\tau_k) = 0 \\ \forall\ k = 1,...,N \end{cases} \tag{3.8}$$

where $c(x_0, \tau_k)$ is a state-transition function that basically represents any type of integration scheme.

One could immediately notice how the intermediate states were eliminated from the formulation, and that's simply because we do not use them as decision variables so we have no access to them. This is obviously a downside as we can no longer impose any cost, bounds, or path constraints on the state variables. Furthermore, it turns out that the resulting NLP is hard to solve and so does not converge to a solution easily.

– ***Multiple-Shooting Method:*** A variation of the above method is given by applying the same underlying idea, but instead of discretizing the inputs only, we also add the states as optimization variables. Therefore, instead of simulating (shooting) over the whole time horizon, this method allows shorter fragmented simulations that last as long as the transcription interval. Finally, all the different simulated fragments are eventually brought together as a result of constraint satisfaction from the optimization formulation given below:

$$
\textit{Let } z_k = \begin{bmatrix} \tau_k \\ x_k \end{bmatrix} \qquad \begin{cases} \underset{z_1,...,z_N}{\boldsymbol{min}} \quad m(z_N) + \sum_{k=1}^{N-1} F(z_k) & \\ \\ \boldsymbol{s.t.} \quad z_{k+1} = c(z_k) & \forall\ k = 1, ..., N-1 \\ \quad\ \ h_{min} \leq h(z_k) \leq h_{max} & \forall\ k = 1, ..., N \\ \quad\ \ g(z_k) = 0 & \forall\ k = 1, ..., N \end{cases} \tag{3.9}
$$

This certainly turns out to be a larger optimization problem than that given by the simple shooting method; however, it converges to a local optimum much more efficiently. Refer to Figure 3.2 for a comparison between both shooting techniques.

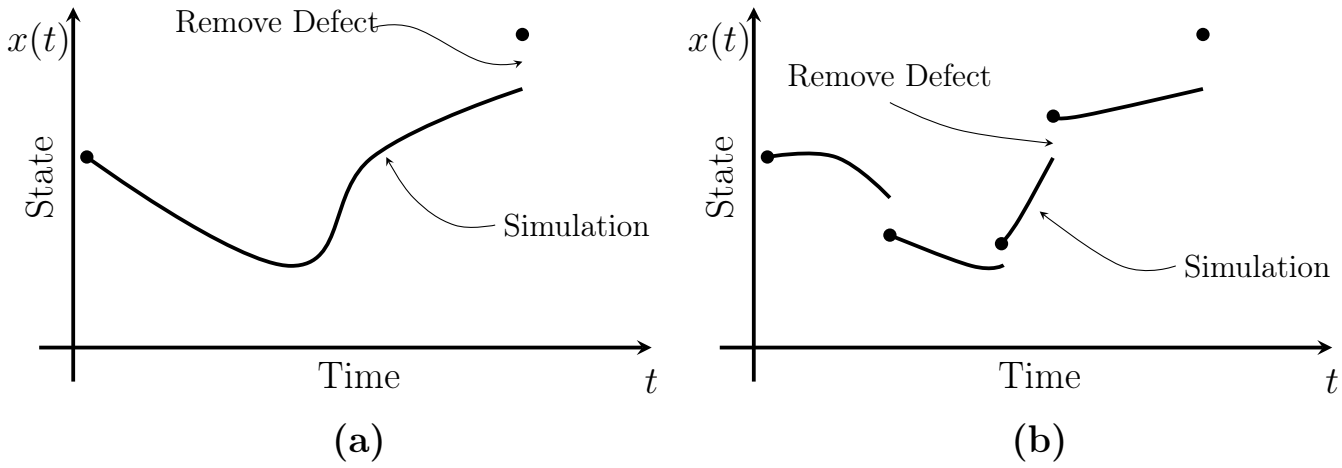


Figure 3.2: **(a)** Shooting Method **(b)** Multiple-Shooting Method [21]

– ***Direct Collocation Method:*** This method is similar to the multiple-shooting method in that it includes the states along with the inputs as optimization variables. Nevertheless, it differs from it in the way the dynamic constraints are treated: Instead of using an embedded integrator within the problem formulation (which is quite an advantage for multiple-shooting methods since they can make use of adaptive integration schemes), the dynamics are satisfied by relying on piece-wise polynomial approximations of the state trajectories, and then requiring that the time derivative of the state-polynomial evaluated at a specific *collocation point* matches with the value of the forcing function $f(x(t), \tau(t))$ at that point (this is referred to as the *collocation constraint*) [10]. To demonstrate, suppose the states at the different *knot points* (those which constitute the decision variables vector) are interpolated by cubic Hermite splines (see [20] for more details and for some example applications of the method):

$$x_k(\beta) = a_k\beta^3 + b_k\beta^2 + c_k\beta + d_k \tag{3.10}$$

where $\beta \in [0, h]$*, and where* $h = t_{k+1} - t_k$ *is the time-step between knot points. The boundary conditions needed to compute the polynomial coefficients are given by:*

$$\begin{aligned} x_k(0) &= x_k \\ x_k(h) &= x_{k+1} \\ \dot{x}_k(0) &= f_k \\ \dot{x}_k(h) &= f_{k+1} \end{aligned} \tag{3.11}$$

In addition to that, now we can require that the collocation constraint holds at the middle of the time-step (as shown in Figure 3.3):

$$x_k(h/2) = \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}(f_k - f_{k+1}) \tag{3.12}$$

$$\dot{x}_k(h/2) = \frac{3}{2h}(-x_k + x_{k+1}) - \frac{1}{4}(f_k + f_{k+1}) \tag{3.13}$$

$$\tau_k(h/2) = \tau_k \quad \textit{(zero-order hold)} \tag{3.14}$$

$$\Rightarrow \quad \dot{x}_k(h/2) = f(x_k(h/2), \tau_k(h/2)) \ \textit{(collocation constraint)} \tag{3.15}$$

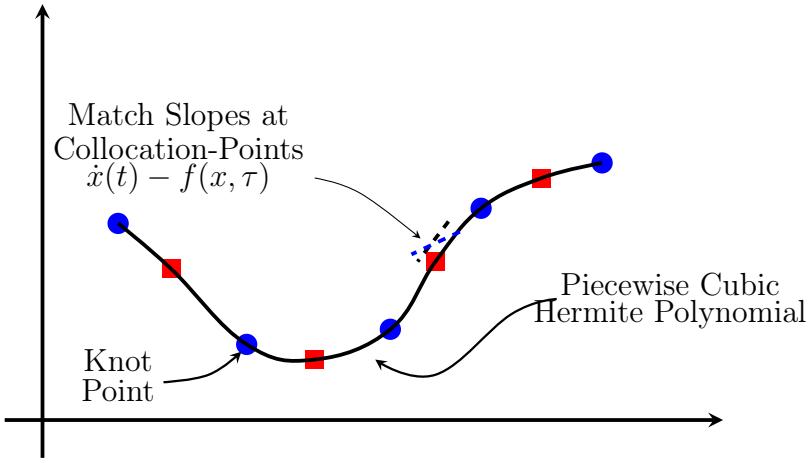


Figure 3.3: Direct Collocation Method [21]

Chapter 4

Contact-Implicit Optimization (CIO)

4.1 Related Work and the CIO Approach

Now that we have laid down the main foundations preceding this thesis's central idea, we can begin to investigate how the concepts introduced in Chapters 2 & 3 come together to tackle the problem of applying trajectory optimization on non-smooth dynamical systems (or *Hybrid systems*). It is important to recall first, that the analytical results on optimal control that have been unfolded so far, require the forcing function $f(x(t), \tau(t))$ to be sufficiently smooth; as for the numerical results, these rely on nonlinear programs or optimization algorithms that are typically gradient-based in nature, meaning that smoothness of the underlying functions is generally expected by the solver itself. Therefore, one reasonable line of thought that goes along the indirect approach, would be to repeat the derivation of Pontryagin's Minimum Principle with an assumption of non-smoothness imposed on the dynamics; but this is certainly a highly mathematically-involved process, and is outside the scope of this thesis. Another idea, which is much more intuitive and elegant, is one that goes along the lines of the direct methods for trajectory optimization and thus transcribes (discretizes) before optimizing, but the difference here is in how the non-smooth dynamics are integrated and incorporated into the resulting program. In fact, the primary integration schemes used for simulating hybrid dynamics have been already introduced extensively in Chapter 2, and in parallel to those, there exists two main classes of TO methods for hybrid systems as well.

One is the *Multi-Phase approach*, which is basically analogous to the event-driven simulation scheme, and it entails an a priori knowledge of the continuous phases (or *modes*) as well as their sequence of occurrence over the whole time horizon. The switching times for the discrete transitions are

not known a priori but are given as a result of the optimization, meaning that the start and end times of each mode are provided as additional optimization variables in the formulation (which follows from direct collocation or one of the shooting methods), while also imposing extra constraints that would ensure a proper connection among the multiple phases. Researchers such as (Patterson and Rao [36]) have developed a general-purpose MATLAB program (GPOPS-II) that utilizes the multi-phase strategy as a framework for solving hybrid optimal control problems. Other successful applications are repeatedly encountered in the legged-robotics community such as for bipedal robot locomotion in (Westervelt et al. [50]), and for quadruped locomotion where an illustration of the method applied for two different contact configurations is shown in Figure 4.1, see (Pardo et al. [35]).

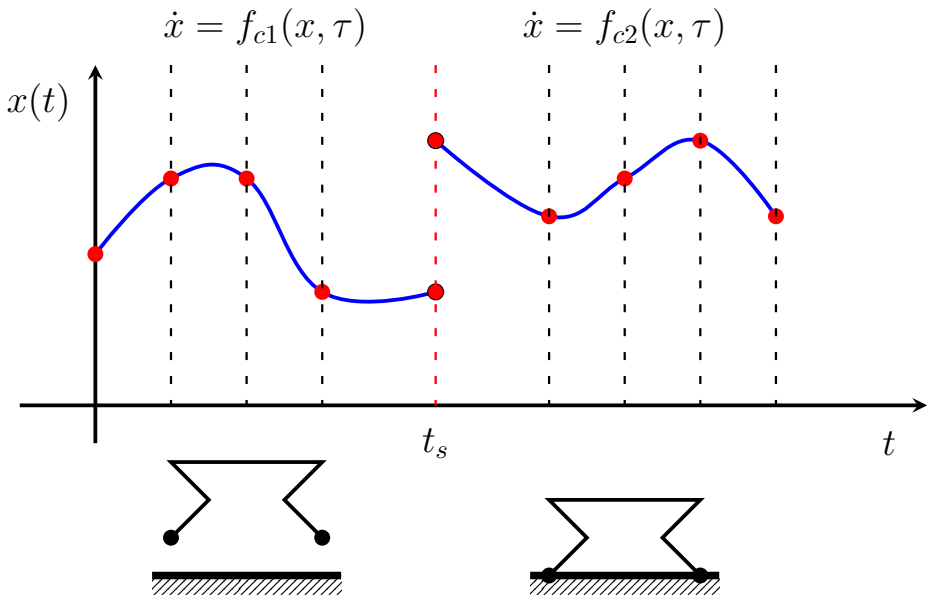


Figure 4.1: Multi-phase approach applied for two different contact configurations also allowing for impacts [35]

Complications with the use of a predefined event-schedule, that separates our TO problem into distinct continuous modes, begins to arise as the number of discrete transitions increases (for example in the case of finitely-many contact constraints between several rigid bodies). That is simply due to the exponential growth of the possible multi-phase sequences with respect to these transitions (this idea is depicted in Figure 4.2).

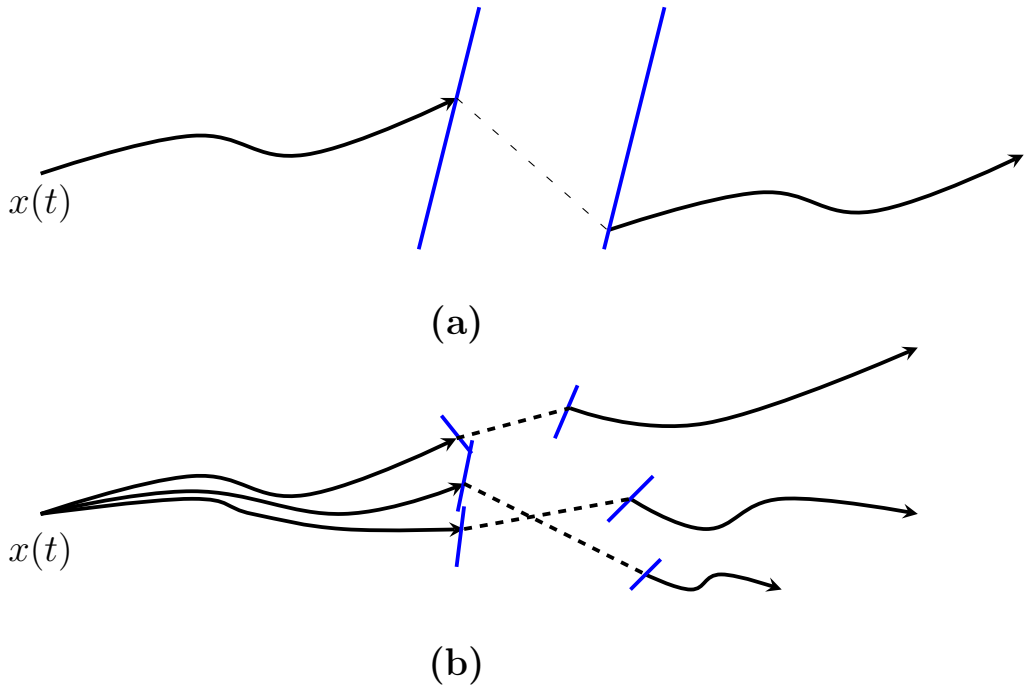


Figure 4.2: **(a)** Simple mode-sequence **(b)** Combinatorial explosion of possible discrete transitions [39]

A remedy to this issue could be found in the *Mode-Invariant approach* (which is also referred to in the literature as the *Contact-Invariant*, *Contact-Implicit*, or *Through-Contact* approach in cases where the non-smoothness arises from contact constraints). This is in fact an extrapolation of the *time-stepping* simulation scheme (introduced in Chapter 2), for accommodating trajectory optimization problems involving hybrid dynamics. The fundamental idea behind it is that instead of having a predefined mode sequence along with inter-phase constraints, discrete transitions are not given any special treatment; hence, the contact constraints are enforced at every single knot-point. This is also capable of incorporating discontinuous jumps in the states (such as in velocities during impact events) by treating the whole problem on an impulse level (refer to the part on the *equality of measures* in Section 2.2), thereby not differentiating between finite forces and impulsive forces. One should keep in mind that similar to multi-phase schemes, the contact-implicit approach can be implemented along with a variety of underlying design choices; meaning that one could flexibly choose any of the direct TO methods, with either a soft-contact model (that can be continuous, or discontinuous in the case of engaging-disengaging the passive elements and in the case of non-smooth dampening terms) or a hard-contact model; and in case the latter was chosen, one could decide to

deal with the set-valued force laws (or normal-cone inclusions) discussed in Chapter 2 by either using the augmented-Lagrangian approach or by solving a complementarity problem. Moreover, one could decide to add the contact forces into the vector of decision variables or to resolve them separately from the optimization iterations. However, the different permutations are certainly not random and thus need to be thought of carefully before arriving at a proper final combination (for example, using a state-dependent soft-contact model while also adding extra variables representing the contact forces as optimization parameters is obviously unjustifiable). Some remarkable work has been recently done in the computer graphics and the robotics communities on finding optimal trajectories under the framework of contact-invariant optimization:

For instance, (Mordatch et al. [29] [30] [28]) use a contact-invariant approach for producing physics-based animations of anthropomorphic, contact-rich tasks like walking, climbing, crawling, moving objects and hand manipulation...Although some variations of their original CIO approach was implemented, there was a common fundamental treatment of the contact constraints by imposing them through penalty terms added to the objective function (soft-constraints rather than hard-constraints); and this relaxation, while it certainly aids the gradient-based solver in efficiently converging to a local minimum, it also tolerates non-physical behavior caused by allowing non-vanishing contact forces to act at a distance. Another soft CIO formulation is achieved in (Tassa and Todorov [48]) by a Differential-Dynamic-Programming (DDP) optimal control method, which is similar to direct-shooting in that it does a full forward simulation of the dynamics but instead of solving for the optimal control sequence with an NLP solver, it does that through Riccati-based solvers while working backwards iteratively; and while performing the required forward simulations, the hard unilateral contact constraints and friction are dealt with by solving a *Stochastic Linear Complementarity Problem* rather than an LCP. With this, they get rid of the discontinuous dynamics and are left with a continuously differentiable system (which is a poorer description of the contact dynamics), while retrieving an optimal controller that is also robust against uncertainties. On the other hand, (Carius et al. [6]) rely on the augmented-Lagrangian approach, specifically the Gauss-Seidel method combined with a proximal point projection (SORprox), in order to solve the set-valued

force laws arising from the hard frictional contacts. This is performed as part of a Moreau time-stepping scheme (see Section 2.2) which is used to simulate the non-smooth dynamics within an iterative-LQR (iLQR) optimal control plan (works similarly to the DDP method). Similarly, (Neunert et al. [32]) have developed a nonlinear model predictive control (NMPC) algorithm that functions by solving, online and at high rates, the nonlinear OC problem using a generalization of the iLQR method. The NMPC scheme was implemented on a quadruped, while adopting a contact-implicit TO formulation, along with a soft-contact model (that includes smoothing elements). Problems arising from the use of such a model were already discussed in Section 2.1.

This thesis work is based upon the contact-implicit optimization approach described by (Posa et al. [39]). Their method is particularly motivated by the Stewart and Trinkle time-stepping scheme [45], where multi-contact dynamics are formulated by a linear complementarity problem (LCP) (given by a blend of complementarity constraints and simple Euler integration steps), and then these dynamics are simulated by solving the LCP at each grid point. However, in the case of trajectory optimization, they reasoned that instead of having to internally solve an LCP within a nonlinear program, one could make use of the simultaneous nature of direct collocation (also applies for multiple-shooting) by incorporating the contact forces $\lambda(t) \in \mathbb{R}^{n_\lambda}$ as optimization variables along with the control inputs $\tau(t) \in \mathbb{R}^{n_\tau}$ and the states $x(t) = [\, q(t) \quad u(t) \,]^T \in \mathbb{R}^{(n_q+n_u)}$; while additionally imposing the unilateral hard-contact constraints through nonlinear equality and inequality constraints included in the OC problem formulation in the form of complementarity conditions. Then as a result of transcription, we are left with solving a so-called *Mathematical Program with Complementarity Constraints (MPCC)*. The transition from the infinite-dimensional optimization problem to the finite-dimensional MPCC is presented below:

– **Hybrid Optimal Control Problem (excluding impacts):**

We exclude impacts in this formulation in order to avoid dealing with the impact equations (given in (2.20)) separately from the equations of motion, and then include them back again into the finite-

dimensional optimization formulation after the supposed discretization of the equality of measures (which embody both the impact equations and the EOM's). This procedure was already demonstrated while explaining how time-stepping schemes operate in Section 2.2.

We start by noting that the general ODE expression $\dot{x} = f(x(t), \tau(t), \lambda(t))$ can be specifically replaced by the under-actuated rigid body dynamic equations involving frictionless contacts, in order to have a formulation that is directed towards solving the applications tackled in this thesis

$$M(q)\dot{u} + b(q,u) + g(q) = S^T\tau + J_N(q)^T\lambda \tag{4.1}$$

where $M(q) \in \mathbb{R}^{(n_q+n_u)\times(n_q+n_u)}$ *is the generalized inertia matrix,* $b(q,u) \in \mathbb{R}^{(n_q+n_u)}$ *contains Coriolis and centrifugal terms (could also contain viscous friction terms),* $g(q) \in \mathbb{R}^{(n_q+n_u)}$ *represents the gravitational effects, and* $S \in \mathbb{R}^{n_\tau\times(n_q+n_u)}$ *is the selection matrix which maps input torques to generalized forces. Finally,* $J_N(q) \in \mathbb{R}^{n_\lambda\times(n_q+n_u)}$ *(given by the formula in (2.15)) is the normal Jacobian which is used to map the normal contact forces to generalized forces.*

It is worth noting that the symbol u is used in the place of $\dot{q}$ for the sake of generality, since in the case where q consists, for example, of a set of Euler angles for representing 3D orientations, then there exists a linear mapping (that is not given by the identity matrix) between the angular velocities and the time-derivatives of these quantities. Moreover, the generalized positions and velocities are not given similar dimensions since this would not be true in the case of a non-minimal representation of 3D orientations (for instance with quaternions). However, for the sake of this work, it is safe to drop these assumptions as they do not occur in any of the upcoming applications.

Now in addition to the equations of motions, we also have to add the following complementarity condition:

$$0 \leq \phi(q) \perp \lambda \geq 0 \tag{4.2}$$

$\phi(q) \in \mathbb{R}^{n_\lambda}$ *was already introduced in Chapter 2 as the signed-distance function between the potentially contacting bodies.* Three rules are ensured by the above condition: There can be no penetration among different objects, contact forces can only push and not pull, and there can be no contact forces acting at a distance. The resulting infinite-dimensional optimization problem is given as follows:

$$
\begin{cases}
\underset{\tau(.),x(.),\lambda(.)}{\boldsymbol{min}} \quad m(x(T)) + \int_0^T L(\tau(t), x(t), \lambda(t))dt \\
\\
\boldsymbol{s.t.} \quad M(q)\ddot{q} + b(q,\dot{q}) + g(q) = S^T\tau + J_N(q)^T\lambda \\
\\
\boxed{\begin{array}{l} \phi(q(t)) \geq 0 \\ \\ \lambda(t) \geq 0 \\ \\ \phi(q(t))^T\lambda(t) = 0 \end{array}} \quad \textbf{\textit{Complementarity Constraints}} \\
\\
x(0) = x_{init} \\
x(T) = x_{final} \\
\\
x_{lb} \leq x(t) \leq x_{ub} \\
\tau_{lb} \leq \tau(t) \leq \tau_{ub}
\end{cases}
\tag{4.3}
$$

– **Mathematical Program with Complementarity Constraints (including impacts):**

Now following the same line of reasoning as that used to discretize the equality of measures in Equations (2.24)-(2.26), one could obtain a transcribed system that also incorporates impact events by considering a first-order Euler-type time-stepping scheme (explicit, semi-implicit, or implicit). In [39], they use an implicit-Euler method (since the state at the next time step is already available in the optimization

program) in order to guarantee numerical stability (which is essential when simulating stiff ode's). The interpretation of that in terms of collocation constraints, is the introduction of an equality between the slope of the linear-state trajectory and the value of the dynamics at the point ending the time-step (the knot points at the end of all time-steps also happen to be collocation points in this case):

$$\frac{x_{k+1} - x_k}{h} = f(x_{k+1}, \tau_{k+1}, \lambda_{k+1}) \tag{4.4}$$

Therefore, the resulting finite-dimensional optimization problem turns out to be as follows:

$$Let\ z_k = \begin{bmatrix} \tau_k \\ x_k \\ \lambda_k \end{bmatrix}$$

$$\begin{cases} \underset{z_1,...,z_N}{\boldsymbol{min}} \quad m(z_N) + \sum_{k=1}^{N-1} F(z_k) \\ \boldsymbol{s.t.} \quad q_{k+1} = q_k + h\dot{q}_{k+1} \qquad \forall\ k = 1, ..., N-1 \\ \qquad \dot{q}_{k+1} = \dot{q}_k + h \cdot M_{k+1}^{-1} \left(-b_{k+1} - g_{k+1} + S^T \tau_{k+1} + J_{N_{k+1}}^T \lambda_{k+1} \right) \\ \qquad \phi(q_k) \geq 0 \qquad \forall\ k = 1, ..., N \\ \qquad \lambda_k \geq 0 \\ \qquad \phi(q_k)^T \lambda_k = 0 \\ \\ \qquad x_1 = x_{init} \\ \qquad x_N = x_{final} \\ \\ \qquad x_{lb} \leq x_k \leq x_{ub} \qquad \forall\ k = 1, ..., N \\ \qquad \tau_{lb} \leq \tau_k \leq \tau_{ub} \end{cases} \tag{4.5}$$

Note that if the integral cost in (4.3) is replaced with Riemann sums in (4.5) then $F(z_k) = h \cdot L(\tau(t_k), x(t_k), \lambda(t_k))$ where h is the time-step length.

To conclude this section, it is worth mentioning that *mode-invariant* approaches are destined to find discrete transitions only at the assigned grid points; and consequently, this leads to a local truncation error that is proportional to the time-step size h in the neighborhood of these transitions (although some recent work was done on deriving a second-order variational time-stepping scheme that could be used in a CIO setting [27], but there is certainly a significant tradeoff between the method's order and the computational cost). Whereas *multi-phase* methods do not have this as a restriction, since the switching times are found as a result of the optimization. In other words, for cases where an a priori knowledge of the mode-schedule is available, it is more reasonable to use a multi-phase method in order to attain a higher accuracy solution.

In this thesis, the dynamic object manipulation tasks that will be presented later, could have been formulated and solved within the multi-phase framework, but the goal was to implement the discussed CIO approach along with some modifications (given in Section 4.3) in order to allow the NLP to discover the mode sequence by itself and optimize through the different modes. This also turns out to be more interesting when dry friction is involved (in Chapter 6) since a larger combination of possible events exists in this case. Furthermore, it was important to ensure the dynamic feasibility of our method by evaluating whether the resulting optimal trajectories could be easily stabilized and tracked with a simple linear feedback control law (unlike Posa's work on trajectory stabilization in [38] [40], as will be further elaborated upon in Section 4.3).

4.2 Preliminary 2D Example: Finger Rolls Ellipse

The above contact-invariant optimization method was initially tested on a couple of elementary 2D examples before shifting to the central problem of this thesis *(a 6-DOF robot dynamically displacing a 3D block to a desired position)*; one of which consisted of a fully-actuated two-link (ellipse-shaped) finger rotating an unactuated but damped 1-DOF ellipse (this problem also appears in [39] [48]). In this Chapter, it is not intended to make a detailed demonstration of the TO formulation for the provided examples, but rather to use them purely for going over certain points.

The assigned task was to displace the free ellipse, starting at rest, from an angle of $q_{3_1} = 0$ to a final angle of $q_{3_N} = -\frac{\pi}{2}$ and zero final velocity over a time horizon of $T = 2s$. The objective function was given by

$$\sum_{k=1}^{N-1} F(z_k) = \sum_{k=1}^{N-1} h\left(u_k^T u_k\right) \tag{4.6}$$

in order to minimize the total energy input.

Although not necessary for our forthcoming discussion, but it would be an interesting exercise to show how the minimum distance function between E_2 and E_3 (see Figure 4.3) was obtained in this example application:

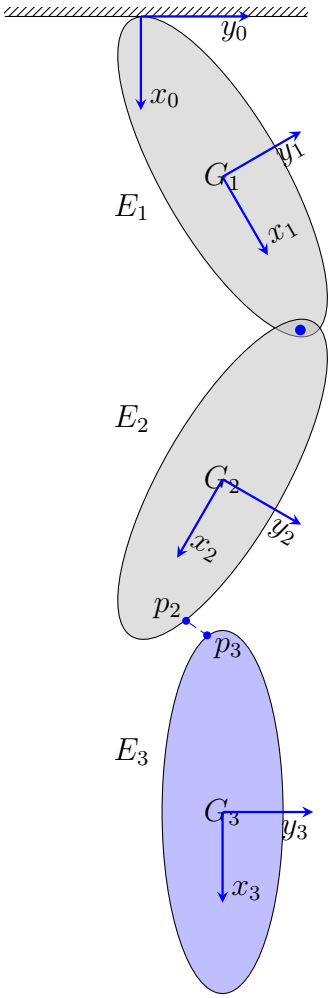


Figure 4.3: Graphical illustration of a two-link finger rotating the unactuated ellipse at the bottom

We start by parameterizing the contour of an ellipse with respect to its geometric center using the parameter φ as such:

$$x(\varphi) = r(\varphi)\cos(\varphi)$$
$$y(\varphi) = r(\varphi)\sin(\varphi)$$

$$s.t. \quad \left(\frac{x(\varphi)}{a}\right)^2 + \left(\frac{y(\varphi)}{b}\right)^2 = 1$$

Solving the above system yields an expression for the coordinates of an arbitrary point P lying on the ellipse contour with respect to $\{G\}$, the center's reference frame.

$$v_{GP}^{G} = \begin{bmatrix} x^G(\varphi) \\ y^G(\varphi) \end{bmatrix} = \frac{ab}{\sqrt{b^2\cos^2(\varphi) + a^2\sin^2(\varphi)}} \begin{bmatrix} \cos(\varphi) \\ \sin(\varphi) \end{bmatrix} \tag{4.7}$$

Therefore, the expression of a vector from a point P_2 on E_2 to a point P_3 on E_3, expressed in the fixed world frame {0}, is given as follows:

$$\begin{aligned} v_{P_2P_3}^{0}(q_1, q_2, q_3, \varphi_2, \varphi_3) &= v_{P_3}^{0} - v_{P_2}^{0} \\ &= \left(v_{G_3}^{0} + R_{G_3}^{0} v_{G_3P_3}^{G_3}\right) - \left(v_{G_2}^{0} + R_{G_2}^{0} v_{G_2P_2}^{G_2}\right) \end{aligned} \tag{4.8}$$

where the unknown quantities appearing in the above equation can be obtained from a straightforward kinematic analysis of the system (R_G^0 is a rotation matrix representing the orientation of frame {G} with respect to frame {0}).

Finally, the minimum distance function $\phi(q)$ is retrieved from the evaluation of the objective function at the solution of an unconstrained non-convex optimization problem.

$$\underset{\varphi_2,\varphi_3}{\boldsymbol{min}} \;\; \|v_{P_2P_3}^{0}\|_2 \tag{4.9}$$

A closed-form expression could not be attained, so this NLP had to be embedded and solved internally within the parent optimization problem.

One important side note is that $\phi(q)$ has to be a signed distance function, otherwise the non-penetration inequality constraint $\phi(q) \geq 0$ would not make sense. It is clear that the minimum distance resulting from (4.9) is always a non-negative quantity, even during penetration. Therefore, to satisfy this constraint, an additional condition was placed which requires the gradients (as unit vectors) of the elliptic level curves $g_2(x, y) = 0$ and $g_3(x, y) = 0$, at the optimal solution $(\varphi_2^*, \varphi_3^*)$, to have opposing directions *(This relation can be derived from the Karush-Kuhn-Tucker conditions of a convex-constrained minimization problem [8]).* That is to say;

Given

$$g(x(\varphi), y(\varphi)) = \left(\frac{x(\varphi)}{a}\right)^2 + \left(\frac{y(\varphi)}{b}\right)^2 - 1$$

Then,

$$\nabla^G g(\varphi^*) = \begin{bmatrix} \dfrac{2x(\varphi^*)}{a^2} \\ \\ \dfrac{2y(\varphi^*)}{b^2} \end{bmatrix} \tag{4.10}$$

So the extra non-penetration condition is,

$$R^0_{G_2}\left(\frac{\nabla^{G_2} g_2(\varphi_2^*)}{||\nabla^{G_2} g_2(\varphi_2^*)||_2}\right) + R^0_{G_3}\left(\frac{\nabla^{G_3} g_3(\varphi_3^*)}{||\nabla^{G_3} g_3(\varphi_3^*)||_2}\right) = 0 \tag{4.11}$$

As a matter of fact, it would have been possible to use all the conditions derived in [8] to correctly impose the complementarity constraints without having to solve an inner nonlinear optimization problem. Nonetheless, to do that, one has to increase the size of the parent NLP by adding φ_2 and φ_3 at every grid-point as decision variables.

The problem was formulated on *MATLAB* where it was solved with the nonlinear programming solver *FMINCON*, using the Interior-Point Method as an underlying optimization algorithm. Other algorithms were also tested, such as Sequential-Quadratic-Programming (SQP) which was leveraged by Posa et al. [39] in their work; but the interior-point approach produced better results. The solver took around 45 minutes for 11 grid points, and it kept on iterating until the maximum number of function evaluations (3000) was reached without converging to an optimum, but ending up much better than how it started (most importantly in terms of satisfying the nonlinear equality and inequality constraints up to a certain threshold). The task was achieved as a result of a contact event purely discovered by the solver without any prior knowledge of the different modes (as can be seen in Figures 4.4 and 4.5).

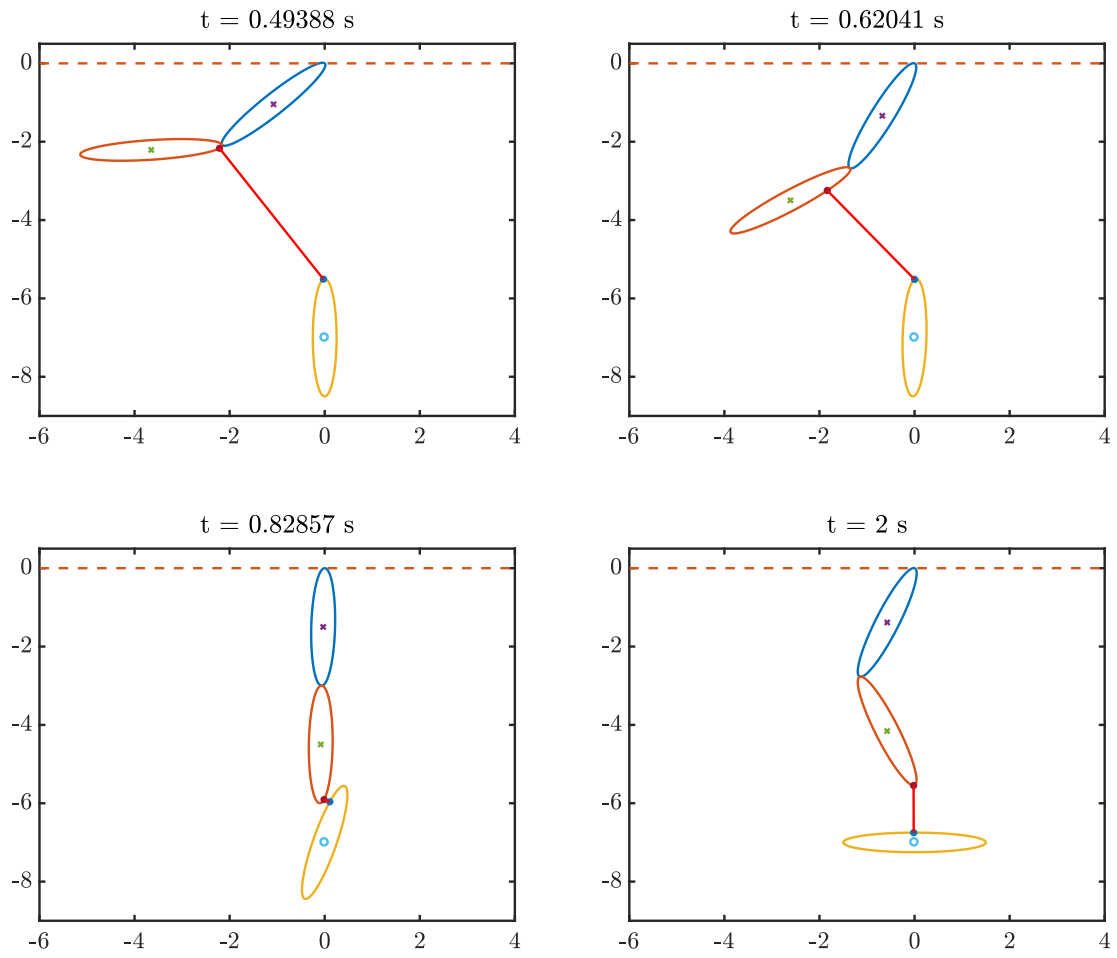


Figure 4.4: Finger rotating ellipse *MATLAB* animation

One could also notice from the plots that the contact force over the whole time-step can be non-zero if and only if the distance function is zero at the end of the interval. Meaning that the force starts to act possibly even before the gap has been closed; this is not an issue for relatively small time-steps, but in this case it turned out to be problematic since the chosen step-length was a bit large ($h = 0.2s$) *(because otherwise, for a fixed time horizon, achieving a smaller step-length requires a higher number of knot points which in turn increases the size of the problem to be solved).* This led to the rotation of the lower ellipse, 0.2 seconds before it was actually hit by the finger. As for the non-penetration constraint, it was successfully satisfied by adding Equation (4.11) into the original CIO formulation.

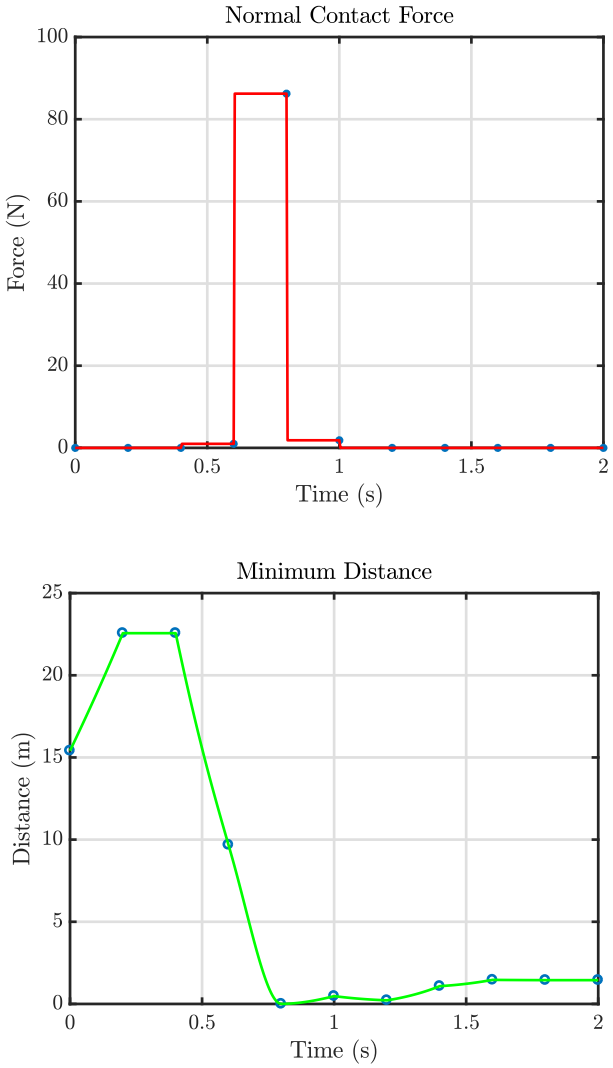


Figure 4.5: Minimum distance between the contacting ellipses *(bottom)* and the generated normal contact force between them *(top)*

4.3 A Modified CIO Approach

In this section, we discuss the final version of the contact-invariant approach that will be utilized from now on throughout the rest of the thesis. But first, it is of paramount importance to introduce the framework within which the finite-time nonlinear optimal control problem will be formulated. Instead of relying on general-purpose NLP solvers such as *MATLAB's FMINCON*, we make use of a commercial tool called *FORCES Pro* [15]. Motivated by the rising need to solve OC problems in an online fashion (for Model Predictive Control applications), this web-based service generates highly customized and efficient optimization solvers which can be deployed on all embedded platforms (the fact that the generated code is customized for one particular optimization problem structure makes it smaller and thus easier to solve efficiently). More specifically, we use the *High-Level Interface* of *FORCES Pro* (*FORCES NLP*) which basically deals with non-linear, non-convex finite-dimensional optimization programs holding the following generic structure:

$$
\begin{cases}
\underset{z_1,\dots,z_N}{\boldsymbol{min}} \quad F_f(z_N, p_N) + \sum_{k=1}^{N-1} F(z_k, p_k) \\
\\
\boldsymbol{s.t.} \quad E_k\ z_{k+1} = c(z_k, p_k) \qquad \forall\ k = 1, \dots, N-1 \\
\\
\qquad S_1\ z_1 = z_{init} \\
\qquad S_N\ z_N = z_{final} \\
\\
\qquad \underline{z}_k \leq z_k \leq \bar{z}_k \qquad \forall\ k = 1, \dots, N \\
\qquad \underline{h}_k \leq h(z_k, p_k) \leq \bar{h}_k \qquad \forall\ k = 1, \dots, N
\end{cases}
\tag{4.12}
$$

where E_k, S_1, S_N are selection matrices, and where p_k represent real-time data that could possibly change after already generating the code (they are assigned a blue color to indicate that they are not treated as optimization variables in the problem).

Furthermore, $c(z_k, p_k)$ is a state-transition function which maps the states from stage k to stage $k+1$; it could be given as a result of any desired inte-

gration scheme; but to be more general, this function could also consist of other nonlinear equality constraints, which is actually necessary for including our non-penetration condition. One immediately realizes that solving this multi-staged system resembles solving a continuous-time nonlinear optimal control problem using the multiple-shooting method. However, in this thesis, (4.12) will be used in the context of a contact-implicit optimization formulation; meaning that a high-order integration method would surely not be the right way to go (since as mentioned before, generally, the best one could anyway achieve with any *Runge-Kutta* scheme is an order of $\mathcal{O}(h)$). Also from now on, unlike what was used in Sections 4.1 and 4.2 (the implicit-Euler method), we only resort to an explicit or semi-implicit Euler scheme (intermediary integration steps could also be made within every time-step h), as the variable at the next stage (z_{k+1}) is not accessible anymore by the state-transition function $c(z_k, p_k)$.

The above formulation can be written down on *MATLAB* and then a customized (problem-specific) C code (which is library-free and avoids any dynamic memory allocation) will be automatically generated. If anything had to be changed from inside the formulation, then the code has to be re-generated, unless the change was in one of these quantities: z_{init}, z_{final}, p_k, or z_0 (the initial guess). The underlying optimization algorithm is an interior-point method with adaptive barrier rules and approximated Hessians. Its effectiveness is mainly a result of an efficient linear-system solver that exploits the multi-stage structure presented in (4.12) in order to solve the resulting Karush-Kuhn-Tucker (KKT) system. This enables *FORCES NLP* to achieve significantly low computation times and thus makes it faster than the state-of-the-art interior point solver *IPOPT* by up to an order of magnitude (refer to [51] for more details).

Another fundamental matter to be discussed, is to state what we expect from the CIO approach in this work, and what we intend to achieve with it, compared to how it was utilized by Posa et al. Referring to [38] [40], one could clearly infer that the proposed contact-invariant method was only used in order to discover the various contact modes and discrete transitions that would eventually be necessary to formulate a multi-phase hybrid problem. Then the latter was solved with a technique they developed for dealing with trajectory optimization regarding constrained dynamical sys-

tems *(DIRCON)*, which is a third-order integration scheme that extends from the previously-explained Hermite-Simpson direct collocation method *(DIRCOL)*. The reason for that is essentially related to the difficulty associated with stabilizing and tracking the optimal trajectories resulting from CIO, especially when dealing with under-actuated systems. Furthermore, an extension of the linear-quadratic-regulator (LQR) was used on the now-accurate optimal trajectory to synthesize a cost-to-go function, which was then minimized (along with additional constraints), online, by a QP solver (as illustrated in Figure 4.6).

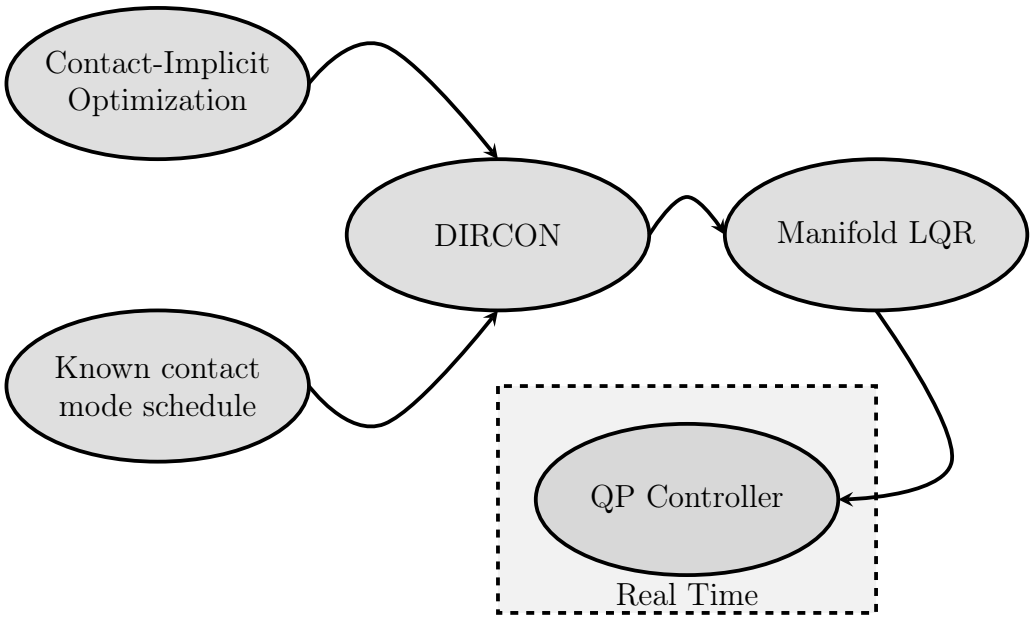


Figure 4.6: Block diagram depicting a full picture of the control plan *(trajectory optimization and trajectory stabilization)* as used by Posa et al. [38]

Alternatively, what was desired in this thesis can be illustrated in Figure 4.7. That is, to purely rely on CIO for obtaining an optimal trajectory that could be directly tracked with a simple linear feedback control-law plus a feedforward term, thus achieving the required task at hand without having to resort to the more accurate multi-phase method; because otherwise, we could have used it in the first place, and the CIO formulation would be deemed useless for our applications.

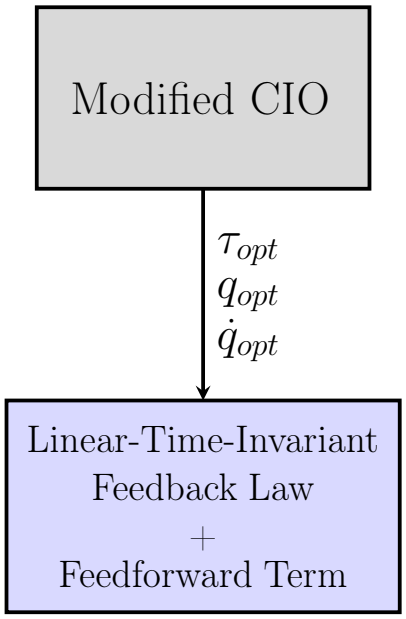


Figure 4.7: Block diagram depicting a full picture of the control plan *(trajectory optimization and trajectory stabilization)* as used in this thesis

In other words, we solve, offline, a finite-dimensional optimization problem of the form (4.12), and then use $\tau_{opt}(t)$, $x_{opt}(t)$, and $\dot{x}_{opt}(t)$ in the following control policy:

$$\tau_i(t) = \underbrace{\tau_{opt_i}(t)}_{feedforward\ term} + \underbrace{k_{p_i}\left(q_{opt_i}(t) - q_i(t)\right) + k_{d_i}\left(\dot{q}_{opt_i}(t) - \dot{q}_i(t)\right)}_{feedback\ term} \quad (4.13)$$

where the index (i) refers to an element from the set of the actuated degrees of freedom, k_{p_i} and k_{d_i} are the scalar feedback gains; and where $\tau_{opt}(t)$ is obtained by applying a zero-order-hold on the sequence τ_{opt_k}, while $q_{opt}(t)$ and $\dot{q}_{opt}(t)$ by linearly interpolating the grid-point-values q_{opt_k} and $\dot{q}_{opt_k}$ respectively, as shown below:

$$\begin{cases} \forall\ k = 1, ..., N-1; \quad t \in [0, T]: \\ \\ (k-1)\cdot h \ \leq\ t\ \leq\ k\cdot h \\ \qquad \Downarrow \\ \tau_{opt}(t) = \tau_{opt_k} \end{cases} \quad (4.14)$$

Now recalling that $x(t) = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix}$;

$$\begin{cases} \forall\ k = 1, ..., N-1; \quad t \in [0, T]: \\ \\ (k-1) \cdot h \ \leq \ t \ \leq \ k \cdot h \\ \qquad\qquad \Downarrow \\ x_{opt}(t) = x_{opt_k} + \left(\dfrac{x_{opt_{k+1}} - x_{opt_k}}{h} \right) \cdot (t - (k-1) \cdot h) \end{cases} \tag{4.15}$$

Now most importantly, the CIO technique that will be used in the upcoming tasks has to take into account the physical feasibility of the occurring contact events. That is because we basically have no control over the improvement of the integration scheme's accuracy except by making the step-size as small as possible (and as explained previously, there is a trade-off between reducing the step-length and increasing the size of the optimization problem to be solved). Therefore, we mainly reason on the frictionless hard-contact dynamics and the resulting unilateral constraints that are imposed in our program as complementarity conditions:

We start the analysis by noting from the state-transition function of (4.12), $c(z_k, p_k)$, that the only way the contact forces λ act over the whole interval is as a result of a zero-order-hold on the values λ_k at the start of the interval (similar to the inputs τ_k). This in turn leads to a different indexing than that appearing in (4.5), hence there is a slight change in the interpretation of the contact conditions. To elaborate, it was previously stated in Section 4.2 that: "The contact force over the whole time-step can be non-zero if and only if the distance function is zero at the **end** of the interval"; whereas now, the condition holds if and only if the gap is closed at the **start** of the interval. However there is still no guarantee that the contact event $\phi(q) = 0$ is held over the entirety of the time-step, even though the contact force keeps acting unjustifiably at a non-zero distance and only vanishes at the end of it. Moreover, the complementarity conditions, which are purely

imposed on a position-level, only ensure that the resulting percussions P_N (note that percussions and contact forces are proportional in time-stepping schemes, and in this case are related by $P_{N_k} = h \cdot \lambda_k$, so they will sometimes be used interchangeably) are large enough to prevent penetration in the next step, but there is certainly no upper limit on how big this value can be (indicating that it is quite arbitrary and only directed towards satisfying the position-based conditions). To illustrate, one could think of a rigid body grazing another body tangentially; so at this moment, $\phi(q) = 0$, and even if there is no chance of penetration, the contact force could still be made arbitrarily large as a result of the optimization, since this would satisfy the complementarity conditions anyway. This straightforward realization is highly significant and crucial for improving the method's dynamic feasibility; and to tackle this issue, we refer to some of the expressions introduced in Chapter 1 while presenting Moreau's time-stepping scheme:

Particularly, we recall the normal cone inclusion given in Equation (2.27) relating the percussions to the pre and post relative velocities, as well as Equations (2.12) and (2.13) which would allow us to interpret the contact event (including impact) on the impulse-velocity level, in terms of a complementarity condition over each time-step:

$$\phi_i(q_k) = 0 : \qquad 0 \leq P_{Ni_k} \perp \left(\gamma_{Ni_{k+1}} + \epsilon_N \gamma_{Ni_k}\right) \geq 0 \tag{4.16}$$

In other words, whenever a contact is closed, the generated percussion (which is a combination of the impulse due to impact and the impulse obtained by integrating the finite contact force acting over the whole time-step) has to respect Newton's restitution law. Moreover, noting that the case of a non-participating contact (which happens in some multi-contact situations) is non-occurring in any of our example applications, we can arrive at a final condition that would be added to the optimization formulation in order to ameliorate dynamic feasibility:

$$P_{Ni_k} \cdot \left(\gamma_{Ni_{k+1}} + \epsilon_N \gamma_{Ni_k}\right) = 0 \tag{4.17}$$

or equivalently,

$$\lambda_{i_k} \cdot \left(\gamma_{Ni_{k+1}} + \epsilon_N \gamma_{Ni_k}\right) = 0 \tag{4.18}$$

Note that the above expression is not equivalent to $\lambda_k^T \cdot \left(\gamma_{N_{k+1}} + \epsilon_N \gamma_{N_k}\right) = 0$ as we do not impose that $\left(\gamma_{N_{k+1}} + \epsilon_N \gamma_{N_k}\right) \geq 0$. One should also note that the relative velocity at the end of the time step $\gamma_{Ni_{k+1}}$ depends on the states q_{k+1} and $\dot{q}_{k+1}$ which are not accessible by the nonlinear equality constraint present in (4.12).

Now assuming a single potential contact, and replacing the notation γ_N with γ, we obtain the following final form of the equality constraint to be added to our formulation:

$$\textit{Given the integration scheme} \quad \begin{bmatrix} q_{k+1} \\ \dot{q}_{k+1} \end{bmatrix} = \begin{bmatrix} c_1(x_k, \tau_k, \lambda_k) \\ c_2(x_k, \tau_k, \lambda_k) \end{bmatrix}$$

$$\begin{aligned} \lambda_k \cdot (\gamma_{k+1} + \epsilon_N \gamma_k) &= 0 \\ \lambda_k \cdot (J(q_{k+1})\dot{q}_{k+1} + \epsilon_N J(q_k)\dot{q}_k) &= 0 \\ \lambda_k \cdot (J(c_1(x_k, \tau_k, \lambda_k)) \cdot c_2(x_k, \tau_k, \lambda_k) + \epsilon_N J(q_k)\dot{q}_k) &= 0 \end{aligned} \tag{4.19}$$

Finally, it is important to put this constraint equation into words, for the case of a purely inelastic collision ($\epsilon_N = 0$): Whenever the contact force is allowed to be non-zero due to a vanishing gap function, λ should hold a value that not only guarantees a pushing effect and prevents penetration, but that also tries to ensure (along with the current states and control inputs) a zero relative velocity at the end of the corresponding time-step. Consequently, the force is now constrained to fulfill this condition, which makes it non-arbitrary; in addition to that, the contact is now held over the whole interval, making the presence of a constant force continuously acting over it (due to the z.o.h) certainly more justifiable. This in turn led to a notable and undeniable improvement in terms of imposing dynamic feasibility, as will be seen and further discussed throughout the development of this thesis.

4.4 Preliminary 2D Example: Leg Kicks Ball

We briefly describe the application of the modified contact-implicit approach on another 2D toy-example, while using *FORCES NLP* to define and solve our optimal control problem. The system is composed of a planar leg with two actuated degrees of freedom (revolute joints at the hip and at the knee) and one free DOF corresponding to the vertical movement of the body attached to the hip. The desired task at hand can be described as follows:

Starting at rest from a given initial configuration, the robotic leg is supposed to kick a ball such that it reaches a pre-specified position at the end of the time horizon ($T = 1.1$ s) with an arbitrary final velocity (there is no stopping force acting on the ball, so once it starts rolling, it cannot be decelerated). Moreover, the leg has to do that whilst respecting certain upper and lower bounds on the states as well as the joint torques. All this is done while striving to minimize energy consumption and of course achieving dynamic feasibility by satisfying the equations of motion (which were integrated with a simple explicit-Euler scheme), as well as the aforementioned contact conditions. To ensure that the kinematic and dynamic parameters considered when deriving the EOM's are realistic, the robotic system *Capler-Leg* [19] was used as a reference in our example. The signed distance function with respect to the ground is simply the height of the foot end-effector, while that corresponding to the ball will be dealt with in terms of a generalized distance function in Chapter 5. To make this multi-contact problem more interesting, the ground plane was assumed to be frictionless and the robot's foot was required to keep a zero tangential velocity while it's in a closed contact situation with the floor. This resulted in a motion plan that involves the Capler-Leg jumping several times before and after making contact with the ball as shown in Figures 4.8 and 4.9; this was necessary in order to avoid violating the no-slip condition in the presence of a slippery surface (absence of stiction). One could also notice from the plots, the satisfaction of the complementarity conditions (no forces acting at a non-zero distance, a positive contact force and no penetration).

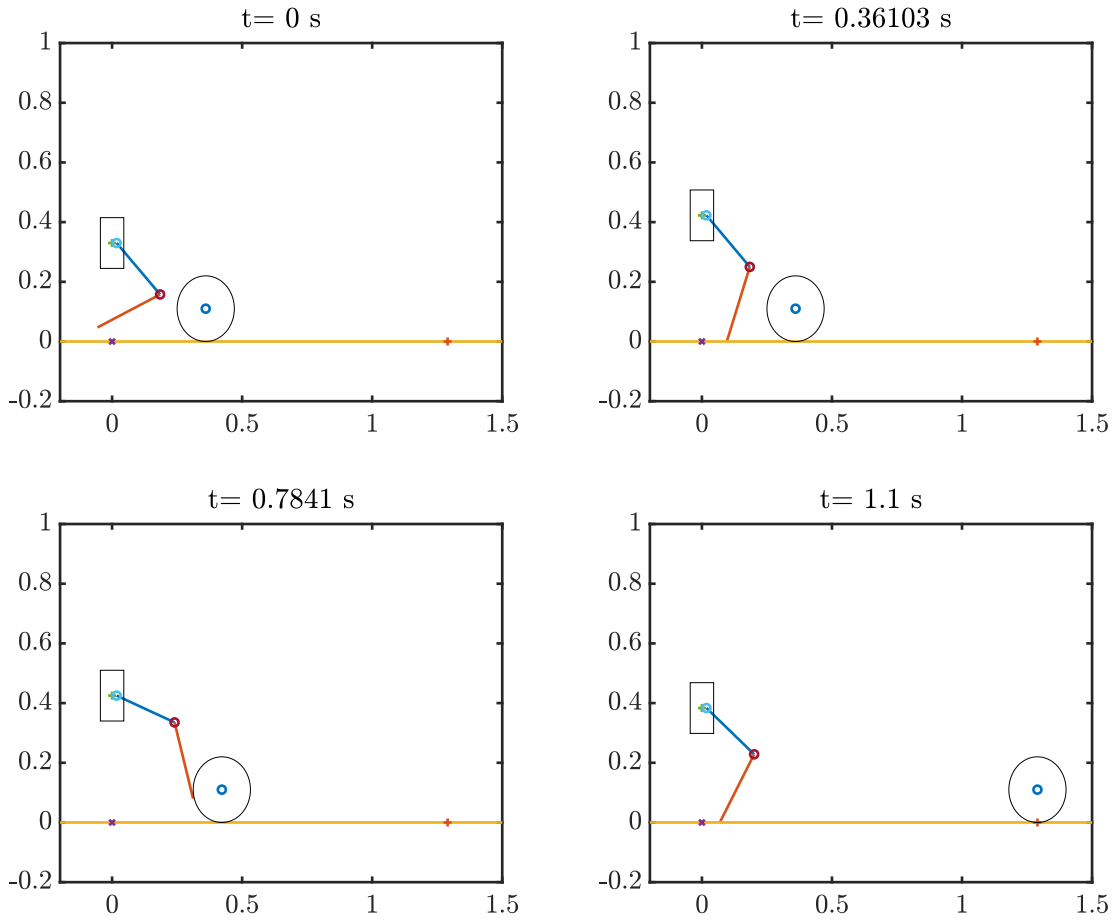


Figure 4.8: Leg kicks ball *MATLAB* animation

A distinction between the original CIO formulation and the modified one will not be made here, but rather later with the more advanced applications, where simulations (and not just visualizations as in this example) are added to the picture. Finally, it is worth noting that the optimization program was solved within 100 milliseconds (500 iterations) after initializing the solver with the right initial guess! The solver was called in a loop multiple times, each time with a random guess, slightly-perturbed from the user-defined one (given by a linear interpolation of the boundary variables). Therefore, all in all, it took around 1.5 to 4 seconds before converging to an optimal solution, for a problem definition consisting of 40 stages ($N = 40$ grid points).

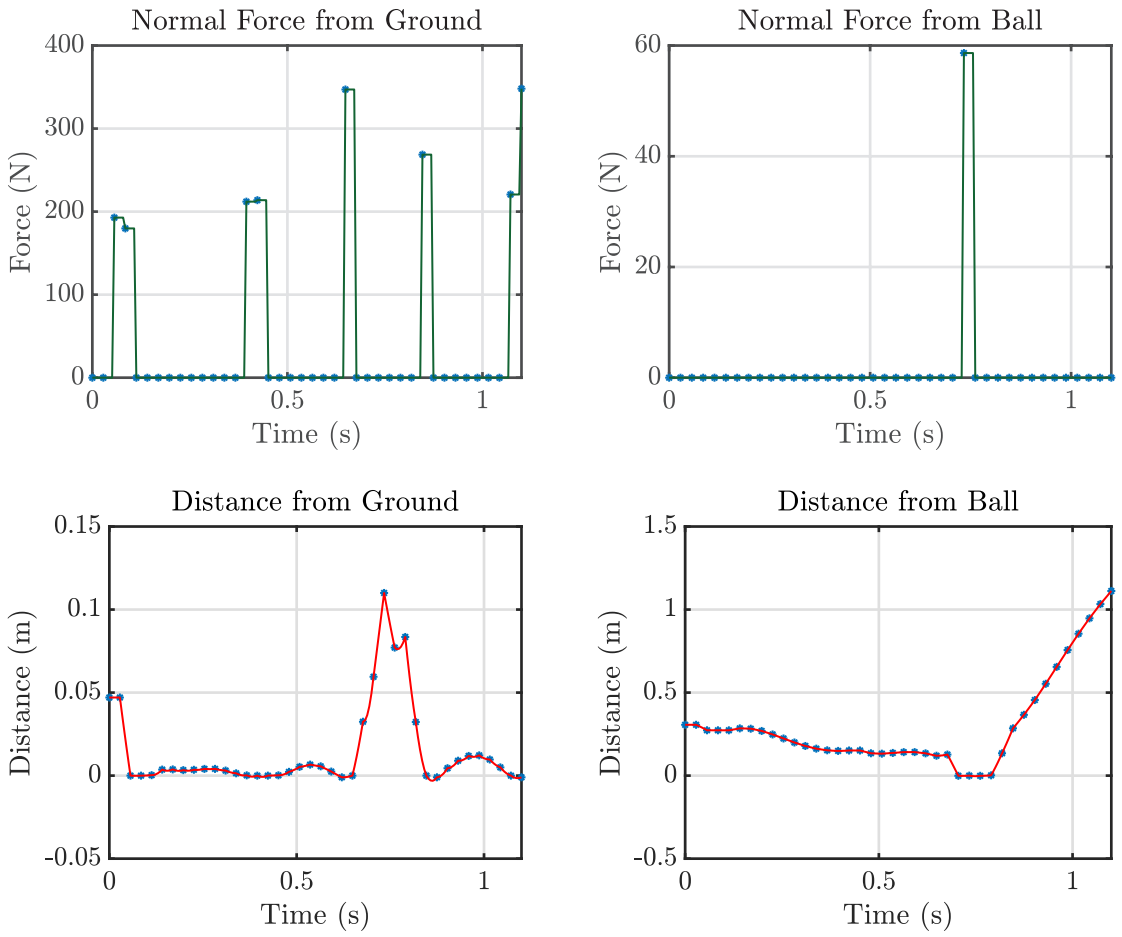


Figure 4.9: Minimum distance between the foot and the ground *(bottom left)* and the generated normal contact force between them *(top left)*, as well as the minimum distance between the foot and the ball *(bottom right)* and the generated normal contact force between them *(top right)*

Chapter 5

Dynamic Object Manipulation by 6-DOF ANYpulator

5.1 ANYpulator Equations of Motion

ANYpulator [5] [7] (presented in Figure 5.1 as the robot arm attached to the mobile platform) is a 6 degree of freedom robot manipulator that was designed and built by the Robotic Systems Lab (RSL) at ETH Zurich. It consists of the following links: *L_0-Base (fixed), L_1-Shoulder, L_2-Arm, L_3-Forearm, L_4-$Wrist_1$, L_5-$Wrist_2$, and L_6-$Wrist_3$ (end-effector link)*; as well as 6 joints responsible for: *J_1-Shoulder Rotation, J_2-Shoulder Flexion/Extension, J_3-Elbow Flexion/Extension, J_4-Wrist Flexion/Extension, J_5-Wrist Deviation, and J_6-Wrist Pronation.*

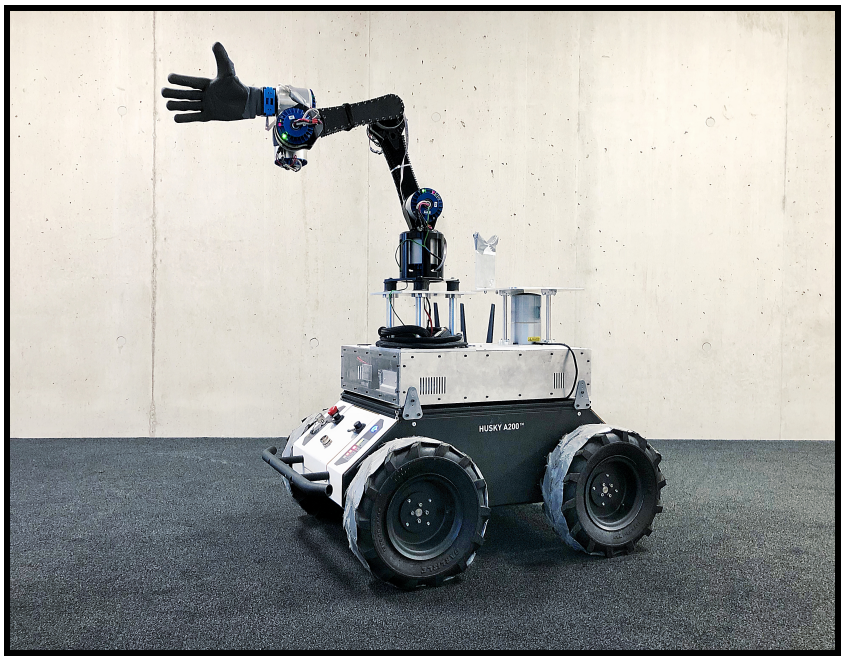


Figure 5.1: ANYpulator robotic arm mounted on top of a Husky mobile platform [7]

The robot comprises a set of robust and mechanically compliant series-elastic actuators (ANYdrive, also developed at RSL, is shown in Figure 5.2) that feature precise position and torque control, high impact robustness, and temporary energy storage; this, in turn, allows for highly dynamic motions and provides the robot with inherent interaction safety [1].

Figure 5.2: Series-elastic actuators, ANYdrive, constituting the joints of ANYpulator [1]

ANYpulator will be used within all the three upcoming dynamic object manipulation tasks, one of which will be tested experimentally. Tackling these problems with the CIO approach essentially requires a proper derivation of the robot and object dynamic equations of motion. In this section, we only do that for the robot, and leave the objects' EOM's for their corresponding parts.

We start by referring to Figure 5.3 while noting that the x-axes are indicated in red, the y-axes in green and the z-axes in blue. For the sake of this demonstration, assume the reference frames are labeled from $\{0\}$ till $\{6\}$ going from the bottom left to the top right of the image, and labeling the last frame (which is attached to the added end-effector) to the far right of the image with $\{\mathrm{E}\}$. Also, as we are not dealing with a mobile robot situation, the base frame $\{0\}$ can be also referred to as the inertial world frame $\{\mathrm{I}\}$.

We define the generalized position vector $q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^T$ and the generalized velocity vector $\dot{q} = \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & \dot{q}_3 & \dot{q}_4 & \dot{q}_5 & \dot{q}_6 \end{bmatrix}^T$.

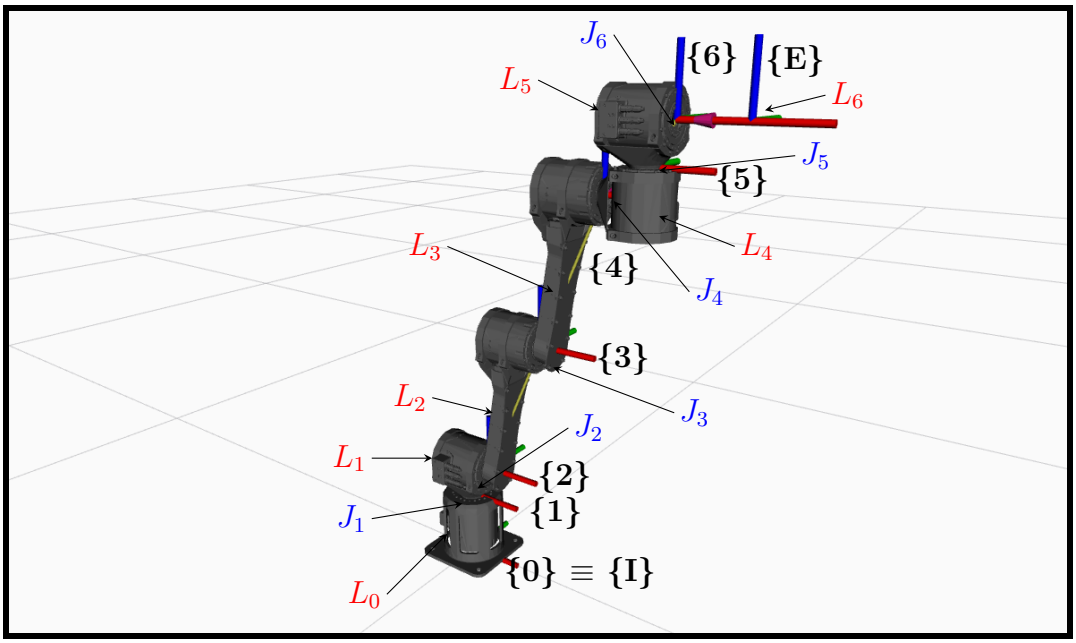


Figure 5.3: Rviz visualization showing ANYpulator along with the sequence of joints, links, and joint-attached reference frames

The position-dependent kinematics of the robot are given by the following homogeneous transformations:

$$T_1^0 = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & \\ \sin(q_1) & \cos(q_1) & 0 & r_{01}^0 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2^1 = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & \cos(q_2) & -\sin(q_2) & r_{12}^1 \\ 0 & \sin(q_2) & \cos(q_2) & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & \cos(q_3) & -\sin(q_3) & r_{23}^2 \\ 0 & \sin(q_3) & \cos(q_3) & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_4^3 = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & \cos(q_4) & -\sin(q_4) & r_{34}^3 \\ 0 & \sin(q_4) & \cos(q_4) & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} \cos(q_5) & -\sin(q_5) & 0 & \\ \sin(q_5) & \cos(q_5) & 0 & r_{45}^4 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_6^5 = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & \cos(q_6) & -\sin(q_6) & r_{56}^5 \\ 0 & \sin(q_6) & \cos(q_6) & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_E^6 = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & r_{6E}^6 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $r_{i_1 i_2}^{i_1}$ represents the position vector directed from the origin of frame $\{i_1\}$ to frame $\{i_2\}$ expressed in frame $\{i_1\}$. In this case the position vectors are independent from the generalized positions, and they can be obtained by referring to the URDF (Unified Robot Description Format) file of the robot (which actually includes all the kinematic and dynamic properties of the robot, obtained from the corresponding CAD model). Now to obtain the homogeneous transformation for any frame {k} *(k=1,...,6)* with respect to the world frame {I} we use the formula below:

$$T_k^I = \prod_{i=1}^{k} T_i^{i-1} \tag{5.1}$$

and

$$T_E^I = T_6^I T_E^6 \tag{5.2}$$

The velocity-dependent kinematics (differential kinematics) are given by the following geometrical Jacobians corresponding to links k expressed in frame {I}:

$$J^{(k)}(q) = \begin{bmatrix} J_P^{(k)} \\ J_O^{(k)} \end{bmatrix} \tag{5.3}$$

where

$$J_P^{(k)} = \frac{\partial r_{s_k}^I(q)}{\partial q} \tag{5.4}$$

such that the position vector $r_{s_k}^I$ directed from the origin of frame {I} to the center of mass of link k expressed in frame {I}, is computed by extracting the first three elements of the vector below:

$$\tilde{r}_{s_k}^I = T_k^I \cdot \begin{bmatrix} r_{ks_k}^k \\ 1 \end{bmatrix} \tag{5.5}$$

$r^k_{ks_k}$ *is also obtained from the URDF file of the robot.*

Also

$$J_O^{(k)} = \begin{bmatrix} R_1^I \cdot \hat{d}_1^1 & \dots & R_k^I \cdot \hat{d}_k^k & 0 & \dots & 0 \end{bmatrix}_{3\times 6} \tag{5.6}$$

where R_k^I *is a rotation matrix which can be extracted from* T_k^I*, and* $\hat{d}_k^k$ *is a unit vector representing the axis of rotation of joint* k *in frame* $\{k\}$*:*

$$\hat{d}_1^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \hat{d}_2^2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \hat{d}_3^3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{d}_4^4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \hat{d}_5^5 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \hat{d}_6^6 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Moreover, in order to map the joint velocities to the linear velocity of frame {E}'s origin we have that

$$J_P^{(E)} = \frac{\partial r_E^I(q)}{\partial q} \tag{5.7}$$

whereas

$$J_O^{(E)} = J_O^{(6)} \tag{5.8}$$

Finally, the *Projected Newton-Euler* formulation [18], which is essentially a combination between the classical *Newton-Euler* approach and the *Euler-Lagrange* formulation, is used to compute the system mass matrix, Coriolis and centrifugal terms, as well as gravitational terms, appearing in the following EOM expressed in generalized coordinates:

$$M(q)\ddot{q} + b(q, \dot{q}) + g(q) = \tau \tag{5.9}$$

where

$$M(q) = \sum_{k=1}^{6} \left(J_P^{(k)}\right)^T m_k \left(J_P^{(k)}\right) + \left(J_O^{(k)}\right)^T \left(R_k^I\right) I_{s_k}^k \left(R_k^I\right)^T \left(J_O^{(k)}\right) \tag{5.10}$$

$$b(q,\dot{q}) = \sum_{k=1}^{6} \left(J_P^{(k)}\right)^T m_k \left(\dot{J}_P^{(k)}\right) \dot{q} + \left(J_O^{(k)}\right)^T \left(R_k^I\right) I_{s_k}^k \left(R_k^I\right)^T \left(\dot{J}_O^{(k)}\right) \dot{q} + \left(J_O^{(k)}\right)^T \left(J_O^{(k)}\dot{q}\right) \times \left(\left(R_k^I\right) I_{s_k}^k \left(R_k^I\right)^T J_O^{(k)}\dot{q}\right) \tag{5.11}$$

$$g(q) = \left(\sum_{k=1}^{6} -m_k \; g^T J_P^{(k)}\right)^T \tag{5.12}$$

where the link mass m_k, and the link inertia tensor $I_{s_k}^k$ (about the center of mass expressed with respect to frame {k}) are obtained from the URDF file. Also, $g^T = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}$ and the time-derivatives of the jacobians are obtained by filling the matrices element by element as such: (let's call the jacobian matrices A and their time derivatives $\dot{A}$)

$$\dot{A}_{ij} = \frac{\partial A_{ij}}{\partial q} \cdot \dot{q} \tag{5.13}$$

All these expressions were obtained with the help of the *MATLAB Symbolic Math Toolbox*, by setting the vectors q and $\dot{q}$ as symbolic variables in the program.

5.2 ANYpulator-Object CIO Formulation: General Considerations

In this section, the CIO formulation commonalities among the upcoming application examples are indicated, while keeping the problem-dependent specifics for later. We refer to Equation (4.12) in order to build our finite-time optimal control problem. *Note that the subscripts 'r' and 'o' will be used to differentiate between quantities related to the robot, and the object respectively.*

– The **number of stages** used was $N = 40$ with a **time-horizon** of $T = 1.5$ s hence leaving us with a **time-step** size of $h = \dfrac{T}{N-1} = 0.0385$ s

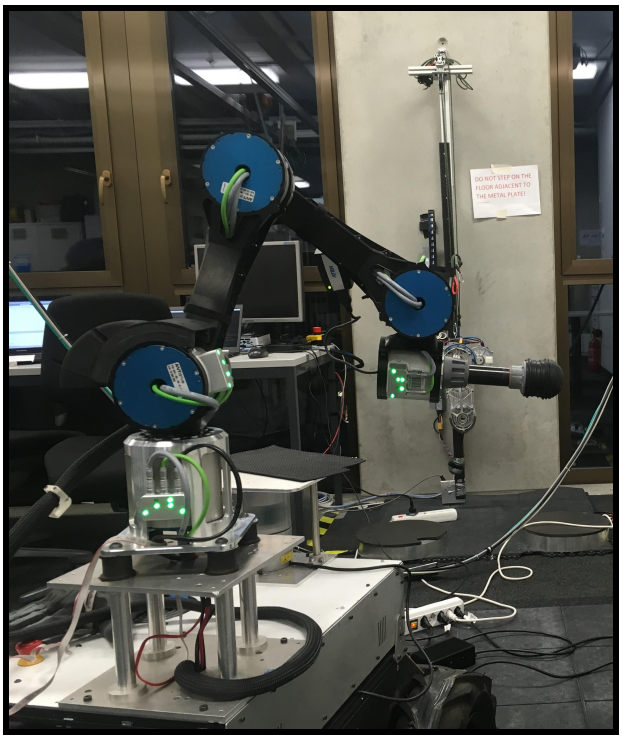


Figure 5.4: ANYpulator with a custom-built end-effector that is used in all ANYpulator-Object manipulation problems

– A common, **custom-built end-effector** was used for all the object-manipulation problems. The specialized tool basically constitutes of a cylindrical rod with a spherical head, as shown in Figure 5.4. Therefore, it was quite clear that the rotation of the last joint would have no contribution at all to the resulting solutions of the applica-

tions at hand. Therefore, a **5-DOF dynamic model** was extracted from Equation (5.9) by setting $q_6 = 0$ and $\dot{q}_6 = 0$ while also getting rid of the sixth equation. The jacobians presented in the previous section were also altered accordingly. This helped reduce the optimization program significantly by removing $3N = 120$ optimization variables.

– The vector of **optimization variables** is given by $z = \begin{bmatrix}\tau & x & \lambda\end{bmatrix}^T$ where $\tau \in \mathbb{R}^5$, $\lambda \in \mathbb{R}$, and $x = \begin{bmatrix}q & \dot{q}\end{bmatrix}^T = \begin{bmatrix}q_r & q_o & \dot{q}_r & \dot{q}_o\end{bmatrix}^T \in \mathbb{R}^{(5+n_{q_o})\times(5+n_{q_o})}$

– **Upper and lower bounds** were imposed on the robot state variables and torques by referring to the *URDF* file. The maximum joint velocity was set to $\dot{q}_{r_{max}} = 12\ rad/s$ and the maximum allowable torque was set to $\tau_{max} = 30\ N.m$. In addition to that, it was set that $\lambda_k > 0$ (one of the complementarity conditions).

– A **common objective function** was used that aims at minimizing energy and also puts a cost on the wrist joints velocities (because otherwise, it was noticed that an unnecessary movement of the two wrist joints occurs, as they do not contribute much to the total energy expenditure). Moreover, the torques and joint velocities were normalized as they do not have the same units. Therefore, the cost function is given as follows:

$$\sum_{k=1}^{N} h \cdot \left(\frac{\tau_k^T R \tau_k}{\tau_{max}^2} + \frac{\dot{q}_{r_k}^T Q \dot{q}_{r_k}}{\dot{q}_{max}^2} \right) \tag{5.14}$$

$$where\ R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad and \quad Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

– Two nonlinear **inequality constraints** were constantly used in the NLP formulation. The first one is the non-penetration condition ($\phi(q_k) \geq 0$), while the second one imposes that the height of the tool frame's origin (*i.e.* The origin of frame {E}) relative to the base frame is always greater than 5 cm $\left(z_E^I(q_{r_k}) = \begin{bmatrix}0 & 0 & 1\end{bmatrix} \cdot r_E^I(q_{r_k}) \geq 0.05\right)$

– The fully-actuated dynamics of the robot were concatenated with the non-actuated dynamics of the object to form the following **under-actuated system dynamic equations** (which are essentially equivalent to those introduced previously in Equation (4.1)):

$$\begin{aligned}\ddot{q} &= \begin{bmatrix}\ddot{q}_r \\ \ddot{q}_o\end{bmatrix} \\ &= \begin{bmatrix}M_r(q_r) & 0_{5\times n_{q_o}} \\ 0_{n_{q_o}\times 5} & M_o(q_o)\end{bmatrix}^{-1} \left(S^T\tau + \begin{bmatrix}b_r(q_r,\dot{q}_r) \\ b_o(q_o,\dot{q}_o)\end{bmatrix} + \begin{bmatrix}g_r(q_r) \\ g_o(q_o)\end{bmatrix} + J_N^T\lambda \right)\end{aligned} \tag{5.15}$$

Furthermore, in order to integrate the dynamics, a **state-transition function** was derived on the basis of a *Semi-Implicit Euler Scheme*. Similar to the classical explicit Euler, the latter is a first-order integrator; however, it additionally holds inherent energy-conservation properties that make it much more reliable. Therefore, the nonlinear equality constraints responsible for satisfying the dynamics are given as follows:

$$\begin{cases}\dot{q}_{k+1} = \dot{q}_k + h\cdot\ddot{q}_k \\ q_{k+1} = q_k + h\cdot\dot{q}_{k+1}\end{cases} \tag{5.16}$$

– Other **common equality constraints** were introduced to the formulation: One ensures that no contact forces are able to act at a distance ($\lambda_k \cdot \phi(q_k) = 0$) and the other one makes sure that Newton's restitution law is satisfied whenever a non-zero contact force (or impulse) exists ($\lambda_k \cdot J_N(q_{k+1}) \cdot \dot{q}_{k+1} = 0$).

– In order to reduce the **computational load** while solving the CIO program, the **gradients** of the objective function and the nonlinear inequality constraints were provided analytically (those are not the final expressions used in the program since more inequality constraints will be added later in the problem-specific discussions, but we present them at this level only, just for the sake of demonstration). The way analytical gradients are provided in the *FORCES Pro* framework, is by filling in the elements corresponding to the variables of a single stage only. This means that the size of the derived gradient will be of size equal to the length of z_k and not N times this length.

$$\textit{Gradient of Objective:} \quad \begin{bmatrix} \dfrac{2hR\tau_k}{\tau_{max}^2} \\ 0_{n_q \times 1} \\ \dfrac{2hQ\dot{q}_{r_k}}{\dot{q}_{max}^2} \\ 0_{n_{\dot{q}o} \times 1} \\ 0 \end{bmatrix} \tag{5.17}$$

Gradient of Inequality Constraints:

$$\nabla_{z_k} \begin{bmatrix} \phi_k \\ z_{E_k}^I \end{bmatrix} = \begin{bmatrix} 0_{1\times 5} & J_{N_k} & 0_{1\times n_{\dot{q}}} & 0 \\ 0_{1\times 5} & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} J_{P_k}^{(E)} & 0_{1\times n_{qo}} & 0_{1\times n_{\dot{q}}} & 0 \end{bmatrix} \tag{5.18}$$

As for the nonlinear equality constraints, their gradients were not provided analytically as this procedure is quite intricate and involved (due to the presence of the dynamic equations of motion among the constraints).

However, due to *FORCES Pro's* integration with *CasADi* [2] [3], an open-source package for generating derivatives through automatic (or algorithmic) differentiation, the required gradients can still be computed efficiently (when compared to numerical or symbolic differentiation). Concerning the hessian expressions of the objective function and constraints, those are provided neither analytically nor through CasADi. That is because *FORCES NLP* does not rely on exact Hessians within its Newton-iterations, but rather approximates them using Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates [33] [51].

– For the **initialization** of the NLP solver, instead of simply relying on a linear interpolation of the boundary variables as an initial guess (which is altered in a loop by adding a slight randomness to it), the solution from a previous non-optimal trial run was used to initialize the next one (this is referred to as warm-starting the solver). *Warm-starting* an inter-point algorithm is generally not so effective as it is for other optimization algorithms, such as Sequential-Quadratic-Programming (SQP). Nonetheless, it still enabled faster convergence in our case. This issue will be brought up again in this thesis when dealing with a non-linear model predictive control (NMPC) approach for the ANYpulator-Ball problem.

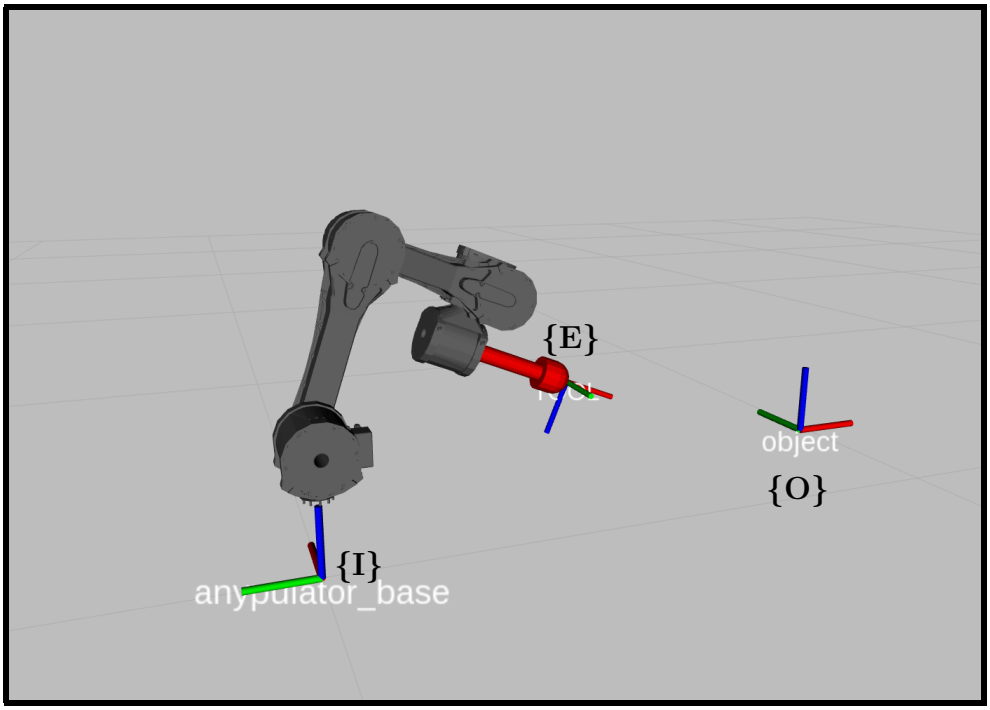


Figure 5.5: Rviz image showing the tool frame $\{E\}$, base frame $\{I\}$, and object frame $\{O\}$

– A **generalized signed distance function** was introduced for the ANYpulator-Object problem, independently from the object being manipulated. Additional geometry-dependent constraints are then added to the optimization accordingly. To elaborate, we start by referring to the image provided in Figure 5.5; it should be noted that at this configuration, the object is considered to have an angle (about

the z-axis) of zero. This choice of frames was made mainly because it was more convenient to place the objects on this side of the robot (towards the negative y-axis of the base frame) while experimenting on the real system. The idea behind our choice of the distance function $\phi(q)$ is based on taking the projection of the position vector r_{EO}^{O} onto the x-axis of the object frame {O}. In that way, it is assumed that the object frame is attached to a virtual plane (that can be translated along the x or y-axes, or rotated about the z-axis, and that separates the robot on one side from the object on the other side), and the gap function is therefore defined as the minimum signed distance between a point and a plane (going through the plane renders the gap function negative thus indicating penetration). However, since our object is not really an infinitely large plane, then some additional constraints will have to be added (in later sections) to achieve the desired behavior. For now, we demonstrate how the expression for $\phi(q)$ is derived:

$$\begin{aligned} r_{EO}^{O} &= R_{I}^{O}\ r_{EO}^{I} \\ &= \left(R_{O}^{I}\right)^{T} \cdot \left(r_{O}^{I} - r_{E}^{I}\right) \\ &= \begin{bmatrix} \sin(\theta_o) & -\cos(\theta_o) & 0 \\ \cos(\theta_o) & \sin(\theta_o) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o^I - x_E^I \\ y_o^I - y_E^I \\ z_o^I - z_E^I \end{bmatrix} \end{aligned} \tag{5.19}$$

Taking the x-component of the above vector yields:

$$\begin{aligned} \phi(q) &= (x_o^I - x_E^I)\sin(\theta_o) - (y_o^I - y_E^I)\cos(\theta_o) \\ &= (x_o(q_o) - x_E(q_r))\sin(\theta_o(q_o)) - (y_o(q_o) - y_E(q_r))\cos(\theta_o(q_o)) \end{aligned} \tag{5.20}$$

It should be noted that x_o, y_o, and θ_o do not always necessarily depend on the object generalized coordinates (as will be seen in the upcoming examples), but here it is assumed to be the case in order to be as general as possible.

Now from this signed distance function, we can obtain the normal Ja-

cobian $\left(J_N(q) = \dfrac{\partial \phi}{\partial q}\right)$ that is required in our CIO formulation:

$$J_N^T(q) = \begin{bmatrix} -\left(\begin{bmatrix}1 & 0 & 0\end{bmatrix} J_P^{(E)}\right)^T \sin(\theta_o) + \left(\begin{bmatrix}0 & 1 & 0\end{bmatrix} J_P^{(E)}\right)^T \cos(\theta_o) \\ \sin(\theta_o) \\ -\cos(\theta_o) \\ (x_o - x_E)\cos(\theta_o) + (y_o - y_E)\sin(\theta_o) \end{bmatrix} \tag{5.21}$$

– The results provided up till now in this thesis work have been purely given by either plotting the optimization output or by kinematically visualizing it. Nevertheless, there is still no guarantee that such trajectories are actually feasible (in the sense that they can be easily stabilized and tracked with the control law introduced in Equation (4.13)); and even if they are, it is also not certain that the contact event that was predicted by our CIO approach is valid enough to attain the desired task at hand. Therefore, **simulating our system** is an unavoidable and essential step to pursue before moving to the real setup. All our simulations were performed within the *Gazebo* robot-simulator environment [23] (except for the NMPC example which was done on *Simulink*). The utilized physics engine underlying Gazebo was the Open Dynamics Engine (ODE) which consists of a rigid body dynamics simulation engine in addition to a collision detection engine. A C++ executable was written in order to send commands to the robot joints in terms of feed-forward torques, reference joint positions, and reference joint velocities; while the PD gains (or PID gains but with a very small integral term) were tuned manually starting from a set of previously-assigned gains for the ANYdrives (the gains used for the Gazebo simulations are: $K_p = [20 \;\; 50 \;\; 50 \;\; 50 \;\; 50]^T$ and $K_d = [10 \;\; 15 \;\; 15 \;\; 1 \;\; 1]^T$).

5.3 Preparatory Example Applications

5.3.1 ANYpulator-Door Problem

5.3.1.1 Problem-Formulation Specifics

In this application, we setup a contact-implicit optimization program which aims at discovering optimal trajectories that would allow ANYpulator to open a door dynamically. To begin with, we describe the door as a rectangular object with these properties: $length\ (l) = 200$ cm, $width\ (w) = 80$ cm, $depth\ (d) = 5$ cm, $mass\ (m) = 15$ kg, and $damping\ coefficient\ (c) = 3$ N.m.s/rad. It is attached to a hinge as depicted in Figure 5.6.

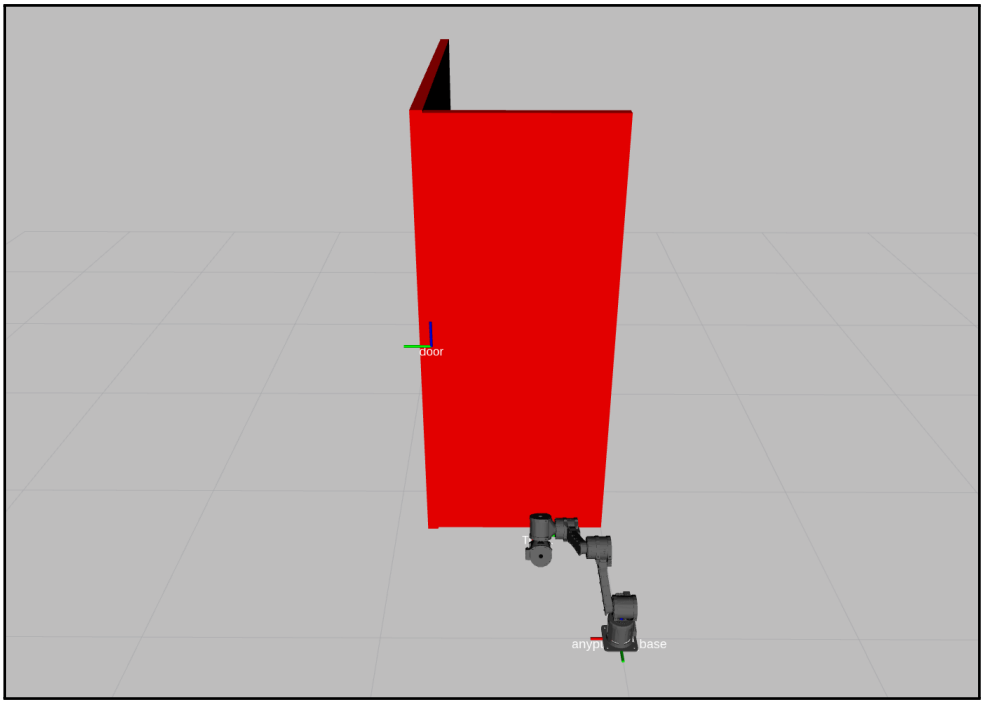


Figure 5.6: Rviz image showing the door and its hinge with respect to the robot

In this case, the manipulated object is a 1-DOF system whose equation of motion is provided as follows: (the robot equations are left out in this section for the sake of conciseness, as they are already given in Section 5.1)

Noting that $x_o = x_{hinge}$, $y_o = y_{hinge}$, and $\theta_o = q_o$ (which is the only generalized coordinate), then we only extract the last row from Equation (5.21) to account for the generalized torque caused by the normal contact force

on the door. Therefore,

$$I_{door}\ddot{\theta}_o = -c\,\dot{\theta}_o + \lambda \cdot (x_{hinge} - x_E(q_r))\cos(\theta_o) + \lambda \cdot (y_{hinge} - y_E(q_r))\sin(\theta_o) \tag{5.22}$$

where I_{door} is the moment of inertia of the rectangular door with respect to the z-axis passing through its hinge. So by the parallel axis theorem, we have that:

$$I_{door} = \frac{1}{12}m(d^2 + w^2) + m\left(\frac{w}{2}\right)^2 \tag{5.23}$$

Now to tell the program that the robot will be making contact with the given door rather than with an infinitely large plane, an additional inequality constraint was introduced:

$$\lambda \cdot \psi(q) \cdot (w - \psi(q)) \geq 0 \tag{5.24}$$

where $\psi(q)$ is the projection of the position vector r^O_{EO} onto the y-axis of the object frame {O} (see Figure 5.5), so it can be retrieved from Equation (5.19)*:*

$$\psi(q) = (x_o - x_E)\cos(\theta_o) + (y_o - y_E)\sin(\theta_o) \tag{5.25}$$

The inequality in (5.24) guarantees that a contact force is generated only if the tool of the robot lies within the door's width.

Finally, the robot and the door were given a starting configuration (as shown in Figure 5.6), with all initial generalized velocities set to zero (as well as the final robot joint velocities). As for the door coordinates at the end of the time horizon, they were chosen in such a way that the desired task is achieved (sufficiently opening the door with a specified final velocity). It is important to note that doing the latter by simply adding the proper final boundary (equality) conditions is problematic, not only for this problem but also for the other applications, as the solver almost never converges to an optimal solution. That is because, roughly speaking, having to satisfy this specific combination of dynamic equations with a final "exact" position at an "exact" velocity, given a fixed time horizon, makes the problem somehow stringent and strict. A good way to deal with such an issue would be to relax the boundary equality constraints and replace

them with upper and lower bounds as such:

$$\begin{cases} \forall\ k = 1, ..., N-1: \\ 0 \leq \theta_{o_k} \leq \dfrac{\pi}{2} - 0.1 \\ -2\pi \leq \dot{\theta}_{o_k} \leq 2\pi \\ \\ \dfrac{\pi}{2} - 0.6 \leq \theta_{o_N} \leq \dfrac{\pi}{2} - 0.1 \\ 0 \leq \dot{\theta}_{o_N} \leq 0.5 \end{cases}$$

5.3.1.2 Results: Optimization and Simulation

As one might expect from an NLP solver, several optimal solutions were returned from various initial guesses; however here we focus on one of them. In this particular case, the program terminates after 1279 iterations, which takes around 1.5 seconds to attain the optimality conditions up to the following tolerances on the residuals:

$$\begin{cases} ||\nabla_z \mathcal{L}||_\infty = 10^{-3} \quad \textit{(stationarity residuals)} \\ ||c||_\infty = 10^{-5} \quad \textit{(equality residuals)} \\ ||h + s||_\infty = 10^{-5} \quad \textit{(inequality residuals)} \\ ||\mathcal{U}s - \kappa||_\infty = 10^{-5} \quad \textit{(complementarity slackness residuals)} \end{cases}$$

where $\mathcal{L}$ is the Lagrangian function, c represents the equality constraints, while h and s represent the inequality constraints and the added slack variables respectively. Moreover, $\mathcal{U}$ is a diagonal matrix containing the inequality Lagrange multipliers and κ represents the barrier parameter (as a vector). Refer to [33] [51] *for more details.*

The plots in Figure 5.7 demonstrate what occurs throughout the interior-point (IP) algorithm's iterations. First of all, one could notice how the objective starts at a value of zero, and that is because of our initialization choice, which gives zero-input torques and zero-joint-velocities as part of the initial guess to the program. Secondly, one could also clearly see that in contrast to unconstrained optimization iterations, not all search directions

tend to reduce the value of the primary objective function, since another concern of the solver would be to keep the decision variables inside the feasibility region. Indeed, an equivalence relationship could be shown between the primal-dual IP-method and the minimization of an augmented cost that consists of the primary objective plus a logarithmic-barrier function that is dominant when the barrier parameter κ is relatively large (thus keeping the variables away from the boundary of the feasibility region), but eventually becomes insignificant as κ decreases (thus truly minimizing the original cost function) [33].

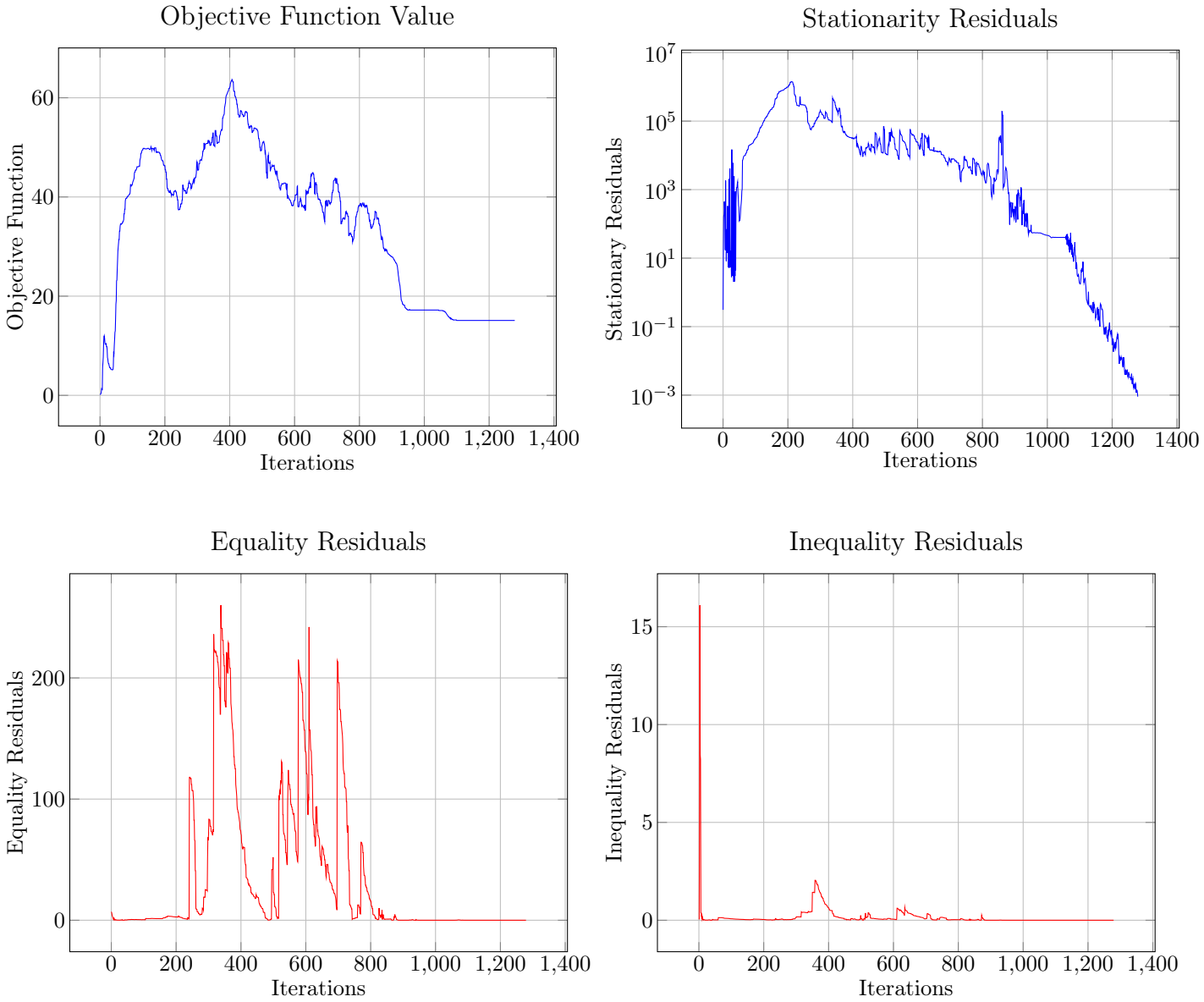


Figure 5.7: FORCES NLP printed output

Another thing that could be noticed from the information outputted by the solver, is that the equality and inequality residuals tend to satisfy the desired tolerances earlier (around 150 iterations before) than the satisfaction of the stationarity tolerances. This idea could be exploited to aid with faster convergence results by slightly increasing the tolerance on the stationarity residuals.

In Figure 5.8, one could see the transition that the solution, corresponding to the input torques of joints 3 and 4, goes through before reaching the local minimum. By looking also at the normal contact force plot in Figure 5.9, it is apparent how the jumps in the torque values at around $t = 0.2$ s relate to the generated contact force between the robot end-effector and the door (when the gap function $\phi(q)$ vanishes). Furthermore, it is clear how the distance between the tool frame and the hinge ($\psi(q)$) respects the requirement that the tool should lie within the door's width during a contact event (in addition to having its height greater than 5 cm at all times).

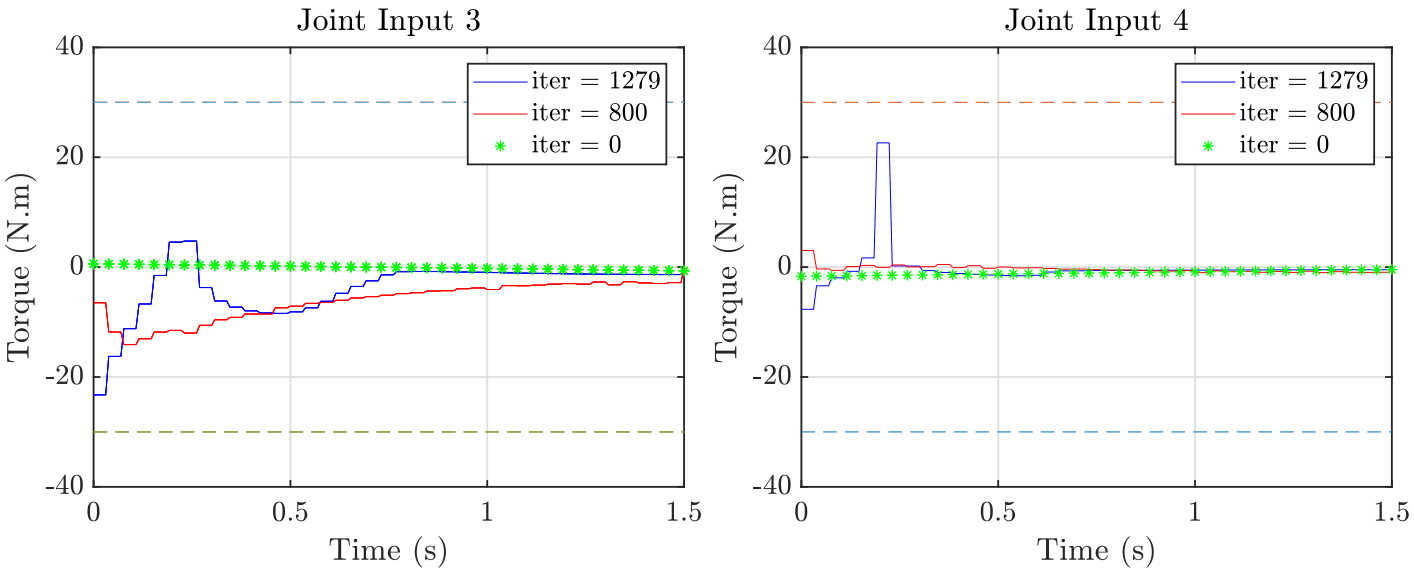


Figure 5.8: Evolution of the input trajectories for joints 3 and 4 over the solver's iterations (0, 800, and 1279)

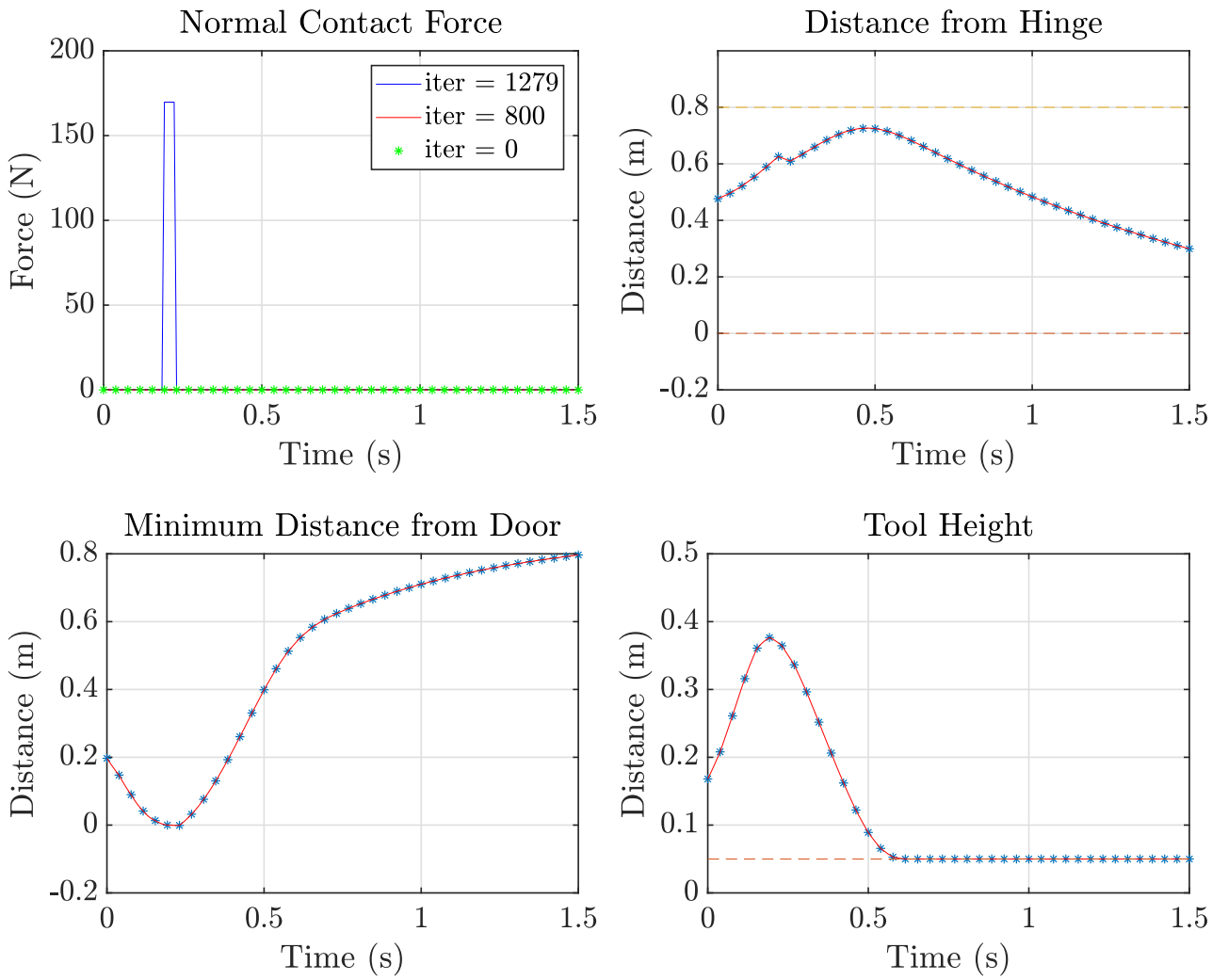


Figure 5.9: Plots indicating the satisfaction of the contact constraints in the ANYpulator-Door problem

Finally, Figure 5.10 shows a Gazebo-Simulation (the visualization was done on Rviz while Gazebo published the current states of the system) where the optimal trajectories were successfully tracked by the manipulator, and the task was properly achieved as a result. Simulation plots are not provided here but will be shown in the other examples, in order to avoid repeating similar ideas and discussions.

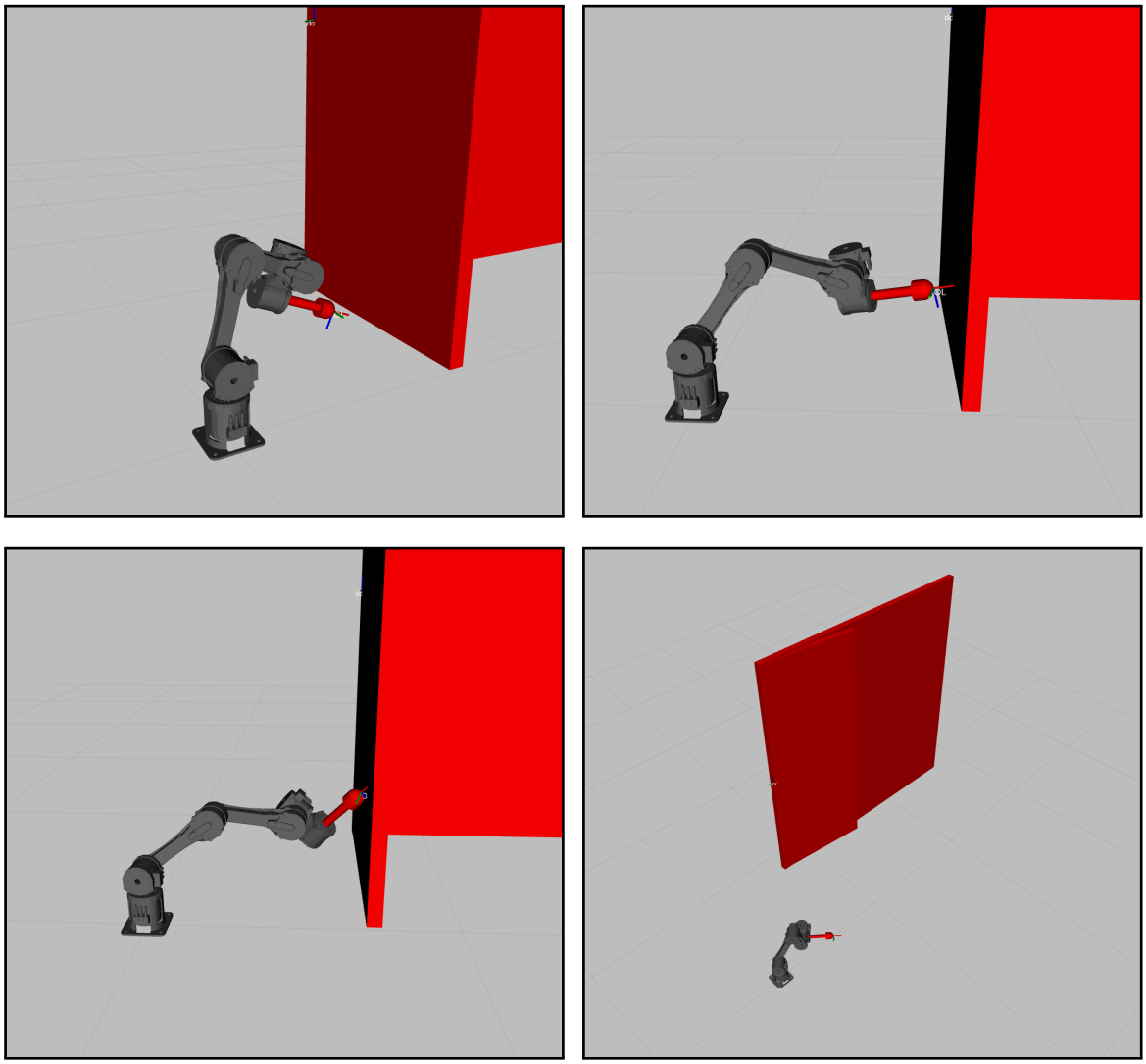


Figure 5.10: Snapshots from a simulation of the assigned manipulation task (Robot pushes door to open it beyond a certain threshold with a relatively low final velocity)

5.3.2 ANYpulator-Ball Problem

5.3.2.1 Problem-Formulation Specifics

In this example application, ANYpulator is required to push a ball with *radius* (R_{ball}) = 11 cm and *mass* (m) = 0.43 kg, along a desired direction, while reaching a goal position in the plane at the end of the time horizon (therefore the final ball velocity doesn't really matter, and an arbitrary bounded value could be assigned to it).

We start by referring to Figure 5.11 in order to derive the equations of motion describing a ball rolling on a flat plane.

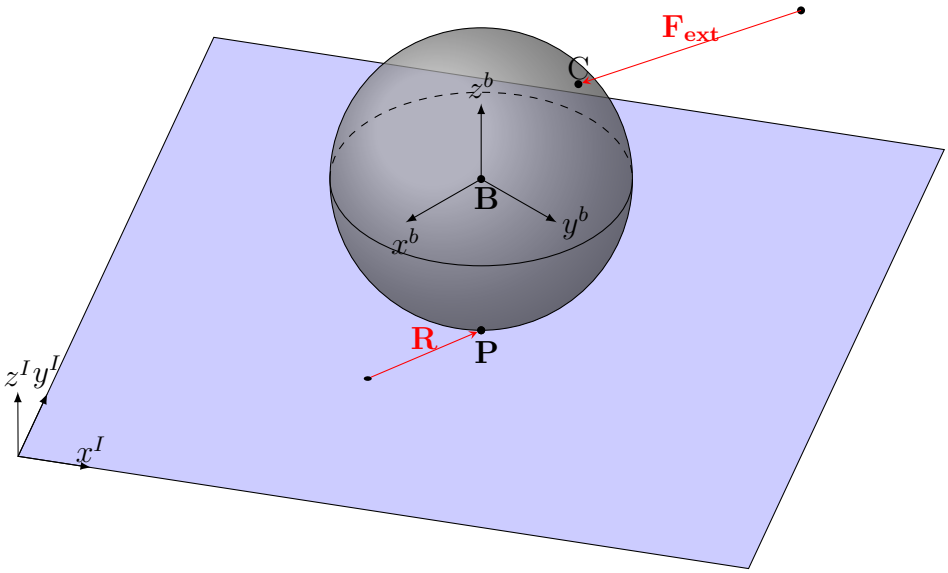


Figure 5.11: Sketch of a 3D ball rolling on a flat plane

Assuming no-slip conditions ($v_P^I = 0$), we have that:

$$v_b^I = \Omega_b^I \times r_{pb}^I \tag{5.26}$$

Basically, we would like to integrate the above equation in order to get a minimal set of generalized coordinates which can be used in the Euler-Lagrange formulation for deriving the EOM's. However, this does not apply here since the above constraint is a non-holonomic constraint (or non-integrable). What can be done in such cases is to use the method of Lagrange multipliers within the Euler-Lagrange equations, to deal with the

non-holonomic constraints. But this approach turns out to be mathematically tedious for this problem, which is why we resort to the *Newton-Euler* formulation instead:

Note that all vectors will be expressed with respect to the inertial frame {I} unless specified otherwise

$$\begin{cases} F_{ext} + R + mg = m\dot{v}_b \\ \tau_{ext} + r_{bp} \times R + r_{bc} \times F_{ext} = I_b\dot{\Omega}_b + \Omega_b \times (I_b\Omega_b) \end{cases} \tag{5.27}$$

where I_b is the inertia tensor of the ball expressed in the inertial reference frame.

This quantity is not constant, hence we replace it with an equivalent expression that includes a constant inertia tensor instead (this is done by equating the rotational kinetic energies of the ball expressed in the inertial frame and the ball frame).

Consequently we have that,

$$I_b^I = I_b = R_b^I I_b^b R_I^b \tag{5.28}$$

such that

$$I_b^b = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} \frac{2}{5}mR_{ball}^2 & 0 & 0 \\ 0 & \frac{2}{5}mR_{ball}^2 & 0 \\ 0 & 0 & \frac{2}{5}mR_{ball}^2 \end{bmatrix} \tag{5.29}$$

Therefore, the second equation in (5.27) becomes

$$\tau_{ext} + r_{bp} \times R + r_{bc} \times F_{ext} = I\dot{\Omega}_b \tag{5.30}$$

Replacing the expression for R in the above equation, one obtains the following

$$\tau_{ext} + r_{bp} \times (-F_{ext} - mg + m\dot{v}_b) + r_{bc} \times F_{ext} = I\dot{\Omega}_b \tag{5.31}$$

Since there are no external torques acting on the ball in our case, and the gravitational vector g is parallel to the vector r_{bp} while r_{bc} is parallel to F_{ext} (only in our case because we assume that a normal force will be acting on the ball and we neglect the forces tangential to the surface), then we are left with the following equation:

$$\Rightarrow \quad -r_{bp} \times F_{ext} + m\ r_{bp} \times \dot{v}_b = I\dot{\Omega}_b \tag{5.32}$$

Now by deriving Equation (5.26) with respect to time, and replacing the expression for $\dot{\Omega}_b$ in there from (5.32), we get:

$$\dot{v}_b = \frac{1}{I}(m\ r_{bp} \times \dot{v}_b - r_{bp} \times F_{ext}) \times r_{pb} \tag{5.33}$$

Noting that $r_{bp} = \begin{bmatrix} 0 & 0 & -R_{ball} \end{bmatrix}^T$ at all times, we solve the above equation accordingly,

$$\Rightarrow \begin{cases} \dot{v}_{b_x} = \dfrac{5}{7m} F_{ext_x} \\ \\ \dot{v}_{b_y} = \dfrac{5}{7m} F_{ext_y} \end{cases} \tag{5.34}$$

Therefore, the ball generalized coordinates are given by $x_o = q_{o_1}$, and $y_o = q_{o_2}$ (note that those coordinates no longer indicate the position of the ball COM, but rather the position of the ball-point belonging to the separating plane), while $\theta_o = \theta_{desired}$ which is the user-specified desired direction along which the ball should travel. Then the final resulting EOM's turn out to be as follows:

$$\begin{cases} \ddot{x}_o = \dfrac{5}{7m} \left(\sin\left(\theta_{desired}\right) \cdot \lambda\right) \\ \\ \ddot{y}_o = \dfrac{5}{7m} \left(-\cos\left(\theta_{desired}\right) \cdot \lambda\right) \end{cases} \tag{5.35}$$

Now to ensure that the robot's tool would make contact with the spherical ball (rather than with any point on the plane that is tangent to it), the

following equality conditions were added to the program:

$$\begin{cases} \lambda \cdot \psi(q) = 0 \\ \\ \lambda \cdot \left(z_E^I(q) - R_{ball} \right) = 0 \end{cases} \tag{5.36}$$

The above conditions actually constrain the manipulator to hit the ball at a single possible point only.

5.3.2.2 Results: Optimization and Simulation

The problem was formulated such that the ball had to travel a displacement of 40 cm at an angle of $\theta_{desired} = 0$ rad and another one at an angle of $\theta_{desired} = -\frac{\pi}{6}$ rad, over the full time horizon ($T = 1.5$ s) such that the final velocity is less than 1 m/s. One would expect that given the low inertia of the ball and the short distance it is required to travel over T, along with the absence of any decelerating forces, the generated contact force is supposedly low. In fact, this turns out to be the case as can be seen in Figure 5.12 from the force plot (this is a good indicator of the dynamic feasibility of our approach). Moreover, the satisfaction of the complementarity conditions can be deduced from the other two plots in the figure. On a parallel note, the solver was able to converge to the two optimal solutions in 365 ms (387 iterations for the case of $\theta_{desired} = 0$ rad) and in 182 ms (208 iterations for the case of $\theta_{desired} = -\frac{\pi}{6}$ rad). This is quite fast for such a highly nonlinear optimization program: not fast enough to be solved in real-time within an NMPC scheme, but fast enough to at least test an NMPC controller efficiently in simulation (without having to wait for long periods of time for the simulation to terminate).

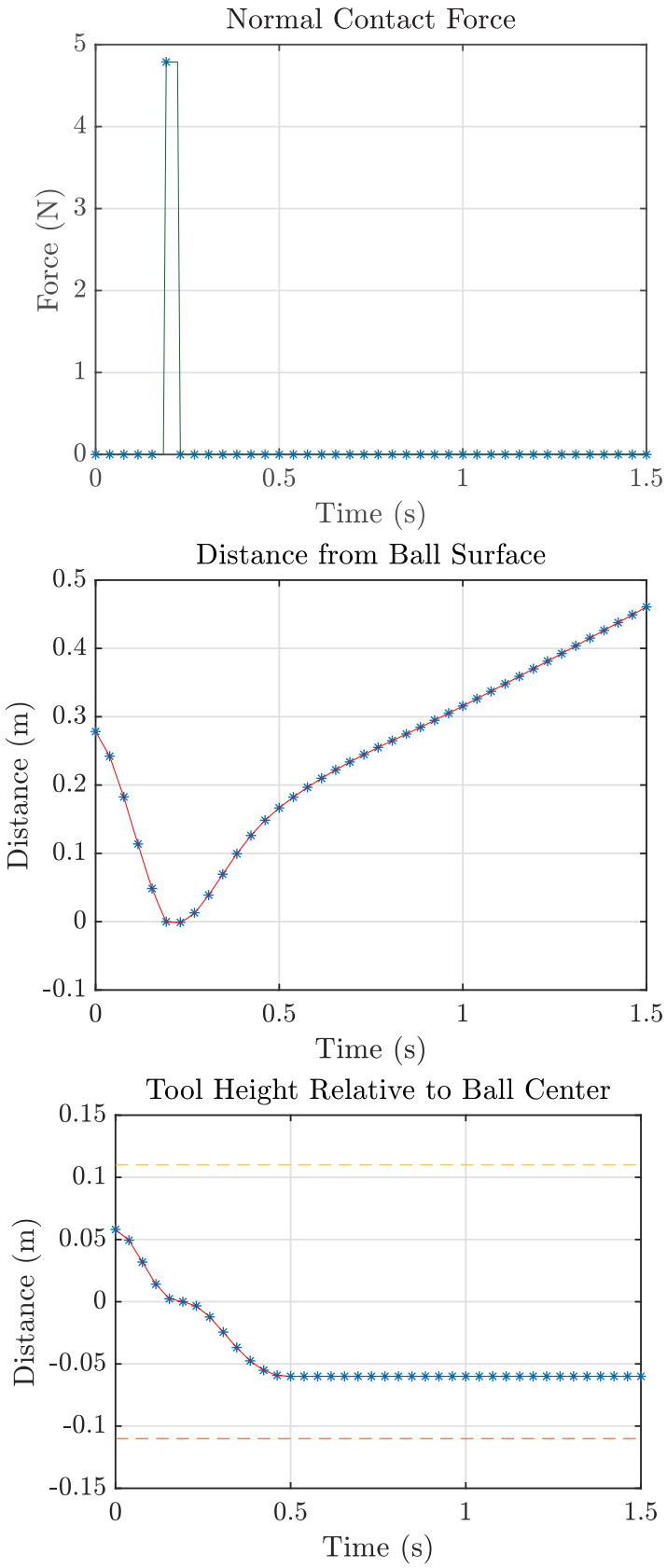


Figure 5.12: Optimization results indicating the satisfaction of the contact conditions

As for the simulation results, Figures 5.13 and 5.14 show how the two dynamic manipulation tasks are well-satisfied for different desired travel-directions. While Figure 5.15 reveals how well the joint position-reference trajectories (obtained from the solution of the CIO problem) are tracked with our simple linear-feedback control law added to the feed-forward term given by the optimal torque trajectories. It is not so clear from these plots when the contact event takes place, and how it affects the tracking process. This is because, as mentioned before, the generated contact force is quite small when compared to the other applications. This effect will be encountered more clearly when working with the ANYpulator-Block problem in the next chapter.

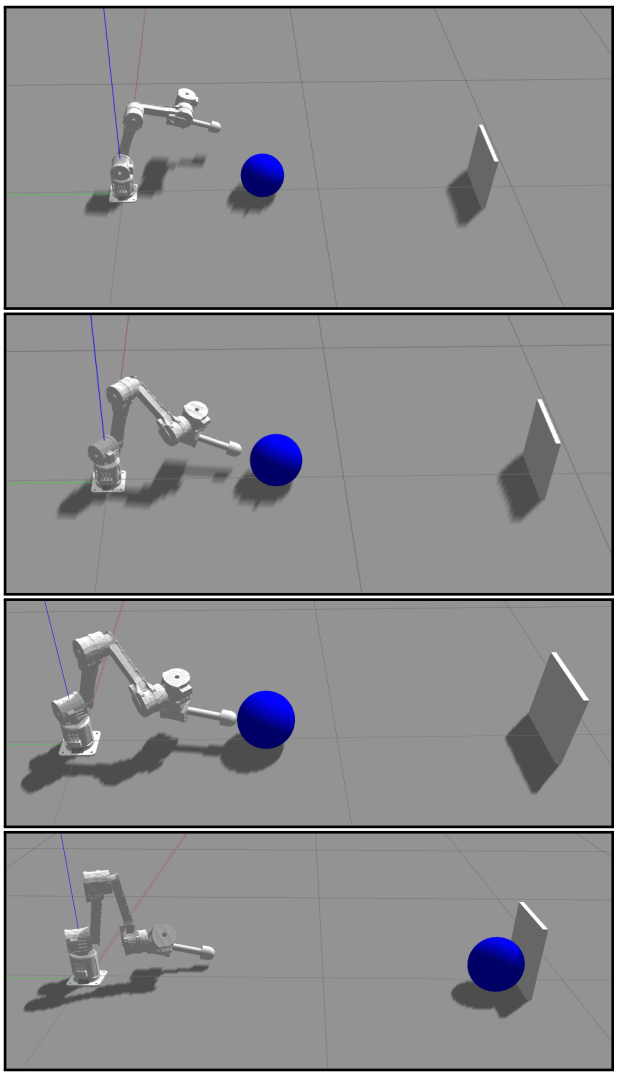


Figure 5.13: Snapshots from a simulation of the assigned manipulation task (Robot pushes ball to a final position in a desired direction of 0 degrees)

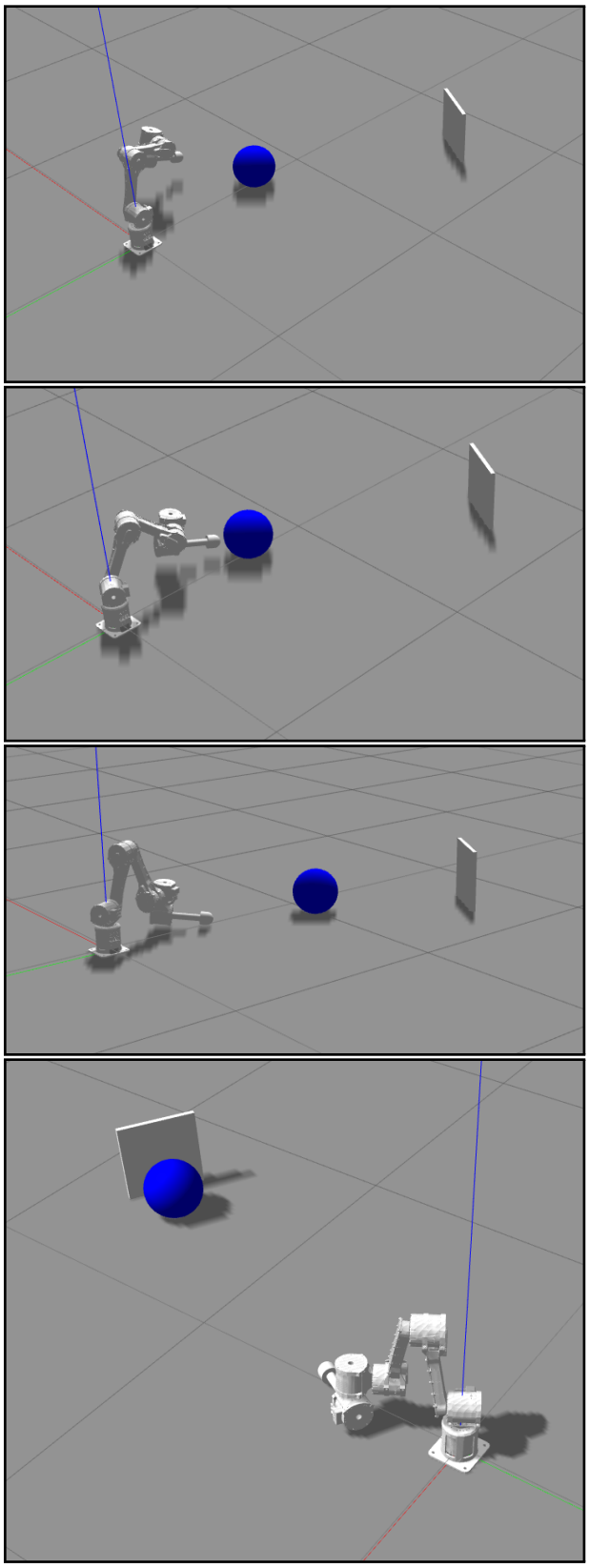


Figure 5.14: Snapshots from a simulation of the assigned manipulation task (Robot pushes ball to a final position in a desired direction of -30 degrees)

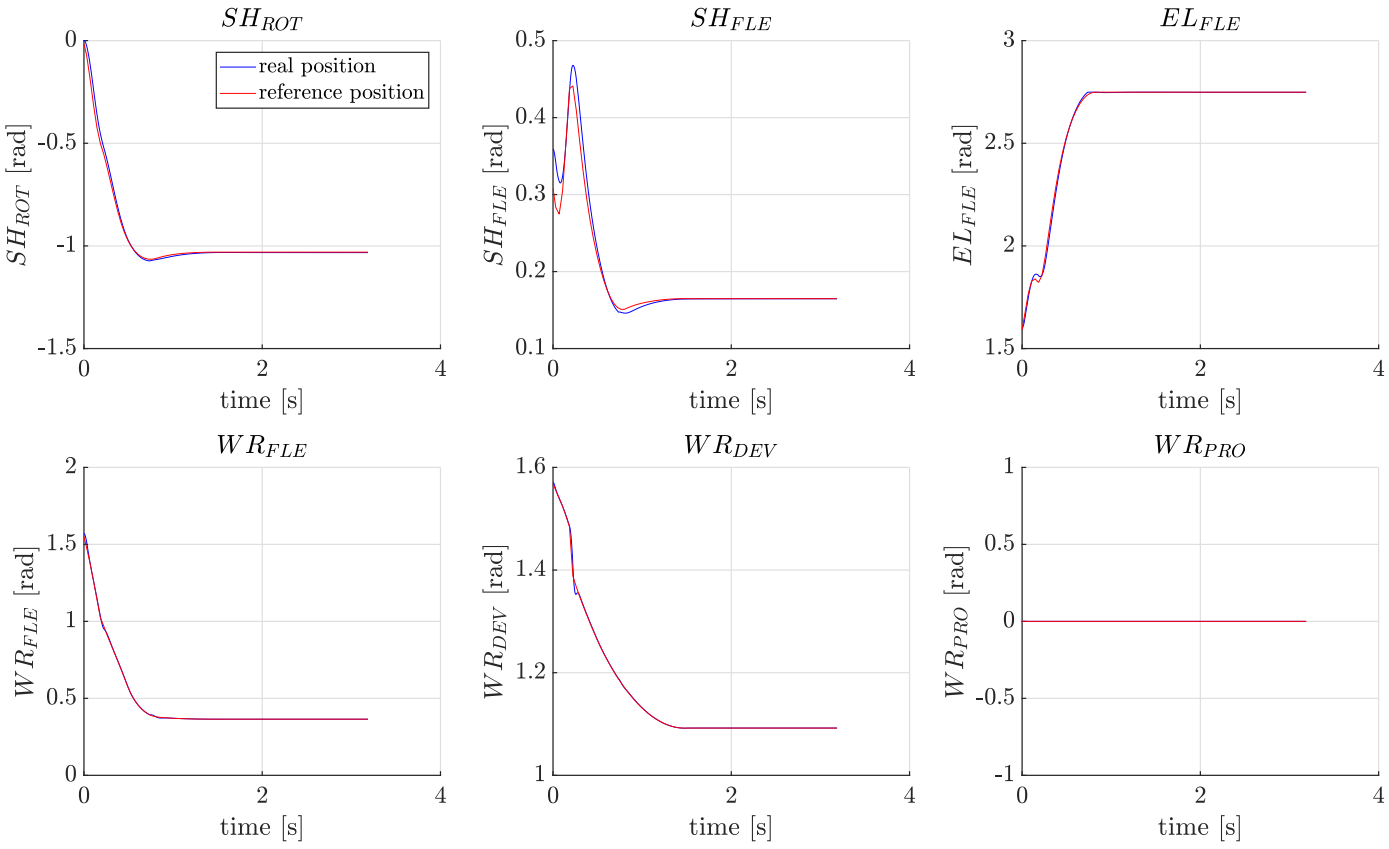


Figure 5.15: Simulation plots showing the tracking of optimal-reference joint positions

5.4 Exploring a Non-linear Model Predictive Control Approach

Non-Linear Model Predictive Control (NMPC) is a feedback control technique that basically solves the non-linear optimal control (NLOC) problem presented in (3.7) repeatedly, online, then only applies the first control input (*i.e.* τ_1) from the entire input sequence obtained over the full horizon from each NLOC solution. This control technique has been extremely prominent and impactful in the process industry throughout the last three decades. Its power lies in the same reasons mentioned in Section 3.1 while motivating the introduction of optimal control methods (more specifically direct methods), plus the added benefit of retrieving an optimal input trajectory that is also stabilizing against uncertainties and disturbances (nominal stability, as well as robust stability results, can be found in [11]). However, having to continuously solve an NLP within the domain of real-time applications comes with an immediate drawback which is related to the large time-delays imposed by the optimization algorithm's convergence time. That is not a serious issue when the evolution of the plant is inherently governed by relatively slow dynamics and the desired control behavior has a low bandwidth (such as in the case of chemical plants). But for applications involving fast dynamic responses, such as those tackled in this thesis, the use of a computationally efficient NMPC scheme is key for its success.

One remedy to the control degradation caused by this expected computational burden would be to turn our NLP into a quadratic-program (QP) by taking a quadratic approximation of the objective function and a linear approximation of the nonlinear constraints. The system dynamics are linearized around the current prediction of the optimal trajectories (given by a combination of the fed-back states and the last optimal solution obtained), which yields a linear-time-varying (LTV) system instead (see for example [24]). Doing this is indeed very helpful from a computational perspective, as highly efficient and fast convex QP solvers can attain convergence times in fractions of a millisecond [16]. Another approach, proposed by Diehl et al. in [13] (also see [14]), is the Real-Time Iteration (RTI) scheme, in which the finite-dimensional nonlinear optimization problem is solved with the sequential-quadratic-programming (SQP) algorithm. However,

instead of waiting for the solver to return a local minimum, the problem is solved sub-optimally by performing a single SQP iteration per sampling step, while initializing the current program with the last returned solution. As a result, the feedback delay is significantly reduced, and a locally optimal solution is gradually attained.

Both of the methods proposed above can be seen as approximate solutions to the original NMPC problem, and this could basically imply a sacrifice in the scheme's optimality, in addition to its aforementioned stability properties, for the sake of a reduced computational time-delay. Therefore, the overall performance of NMPC and its real-time feasibility can be enhanced by finding a good compromise between speed and solving a proper representation of the original problem to optimality. In fact, such a desire is what actually prompted the introduction of the *FORCES NLP* software package [51].

Our goal in this work is to mildly explore a *Receding Horizon* NMPC scheme – in which the time horizon is kept constant all throughout, unlike what happens in *Shrinking Horizon* NMPC – by purely relying on the optimal solution of the exact non-linear mathematical program with complementarity constraints (MPCC), returned by *FORCES Pro* at every sampling instant. We test our algorithm on the ANYpulator-Ball problem using *Simulink* as a simulation environment by relying on a fixed-step Runge-Kutta-4 integration scheme (there was no contact model included in the simulation, but it was simply assumed that the optimal force trajectory will be applied as is on both bodies). Recalling that convergence for this problem occurred within 500 iterations (around 410 ms), we set the maximum allowable number of iterations to 500 (so that the solver would not take more time in case convergence was not achieved before that). One could notice that such a computational time is not fast enough for a real-time implementation (which requires in our case a sampling rate no less than 1 kHz); nonetheless, it is still worth trying in simulation. Generally, one would expect the optimization algorithm to take this much time only for poor initial guesses, and then to become significantly faster by initializing our current program with the previous solution. However, as hinted upon earlier, warm-starting an interior-point method is not very effective (even initializing the same exact program with its previously computed op-

timal solution sometimes leads to failure in convergence!). That is related to the fact that when the optimal solution was found, the barrier parameter had a different value than the one it starts with when initializing the problem again (so it is as if a different KKT system is being solved). Moreover, the obtained optimal solution tends to be close to the boundary of the feasibility region; which means that starting our algorithm from that point would lead to very short step lengths (to stay within the feasible region) taken in the corresponding search directions, hence the search space is not explored properly. As a matter of fact, there are research papers that are solely devoted for presenting appropriate warm-starting strategies for IP methods in the context of MPC and NMPC [42] [52], but this is outside the scope of this thesis.

Let us assume that we were able to achieve a lower computational time, for example by considering a proper warm-starting technique (or any other possible improvements), to the extent that this time is somewhat close to the step-length used inside the definition of our NLP (*i.e.* $h = 38.7$ ms). In that case, we can implement an NMPC-type scheme that is described by the following generic algorithm (provided in Algorithm 5.1). A couple of points ought to be brought up concerning the suggested scheme:

– The first thing is that the CIO formulation, being solved online, needs to have both its initial and final boundary conditions updated. The former is quite obvious as we intend to obtain a state-feedback controller (so the start of the problem's horizon represents the current actual time), whereas the latter is mainly due to the fact that we are relying on a receding horizon controller (T is kept constant). To elaborate, one could think for example of what happens when a contact event is discovered at stage k starting from the current instant, and so as a result of that, the ball starts rolling at a required constant velocity that would allow it to reach the exact final desired position after T seconds. Now at the next sampling instant, the past optimal trajectory is shifted by one step, thus also leading to an event occurring after $k-1$ stages instead of k. Therefore, at the end of the time horizon, the dynamic equations of the ball would govern that it travels a longer distance than that assigned originally. This explains why a 1-step update was continuously made on the final states (before reaching the contact event).

Algorithm 5.1 *CIO-NMPC Algorithm for Dynamic Object Manipulation*

Given

- Time horizon T
- Total number of stages N
- Time-step length $h = \dfrac{T}{N-1}$
- Sampling time $\Delta t_{sampling} \approx h$
- CIO_1: Corresponding CIO program to be solved offline, with an arbitrary number of *maximum iterations*
- CIO_2: Corresponding CIO program to be solved online, with a number of *maximum iterations* chosen such that the maximum solve-time is close to h

Do Offline

- Solve CIO_1
- Save optimal sequences: $z_{opt} = [\tau_{opt}\ x_{opt}\ \lambda_{opt}]^T$
- Set $k = 0$

Repeat Online

- Get state measurement x_{meas} at $t = k\ \Delta t_{sampling}$
- Apply the control law: $\tau = \tau_{opt_1} + k_p(q_{opt_1} - q_{meas}) + k_d(\dot{q}_{opt_1} - \dot{q}_{meas})$ (this input is held over $\Delta t_{sampling}$)
- As long as contact has not been achieved yet: assign the initial boundary conditions $(S_1 z_1)$ from the available parts of x_{meas} and the remaining elements from z_{opt_1}; specify the final boundary conditions $(S_N z_N)$ by advancing the previous condition (only the states) with a single step; also assign $z_{guess} = z_{opt}$
- Otherwise if a contact event has just occurred: assign the initial boundary conditions $(S_1 z_1)$ from the available parts of x_{meas} (robot states) and choose the remaining initial and final boundary conditions in a manner that assumes that the manipulation task is already over from this point on; also assign $z_{guess} = z_{opt}$
- Solve CIO_2 (this should take around $\Delta t_{sampling}$ seconds to terminate)
- Disregard the first stage of the new optimal solution z_{opt} (or the old one in case the solver does not converge to a local minimum), add $z_{opt_{N+1}} = z_{opt_N}$, then replace z_{opt_N} with $z_{opt_{N-1}}$
- Increment k by 1

- Secondly, it should be made clear that for the sake of our dynamic manipulation problem, the measured quantities that are required to assign the initial boundary condition only include the robot-joint-states. Hence, sensory feedback on the object's states and the contact force is not needed, as it would not make sense to rely on such information since the response speed of such a dynamic pushing action is much higher than that for the model predictive controller's reaction needed to alter the contact force. In addition to that, the ball quickly goes outside of the robot's workspace when pushed, meaning that it is doomed to be uncontrollable after this moment; so there is no added benefit in measuring or even estimating its motion. Therefore, the wisest thing to do would be to rely on the information revealed to us by the preceding optimal solution. This also explains why after contact has taken place, the NLP was formulated and solved as if the task has already been achieved (the robot is incapable of controlling the object at this point)
- Finally, one should keep in mind that the NMPC control law is supposed to be stabilizing, without any need for any other feedback terms (such as the PD-term included in the algorithm); however, this holds true assuming that the sampling rates are "high enough", the NLP includes an accurate representation of the dynamics, and that it is solved up to optimality at every sampling instant.

The major results for this scheme along with a comparison between pure NMPC (only optimal torques without a PD-term), NMPC plus a PD-term, and an open-loop solution for the NLOC problem plus a PD-term, are given in the three figures below. One could see from the Solution Status plot of Figure 5.16 that the solver only fails twice to converge to an optimal solution (indicated by an Exitflag of 1). Moreover, a clear distinction can be made between the position responses in plots (b) and (d) that correspond to an NMPC scheme without and with PD-action, respectively.

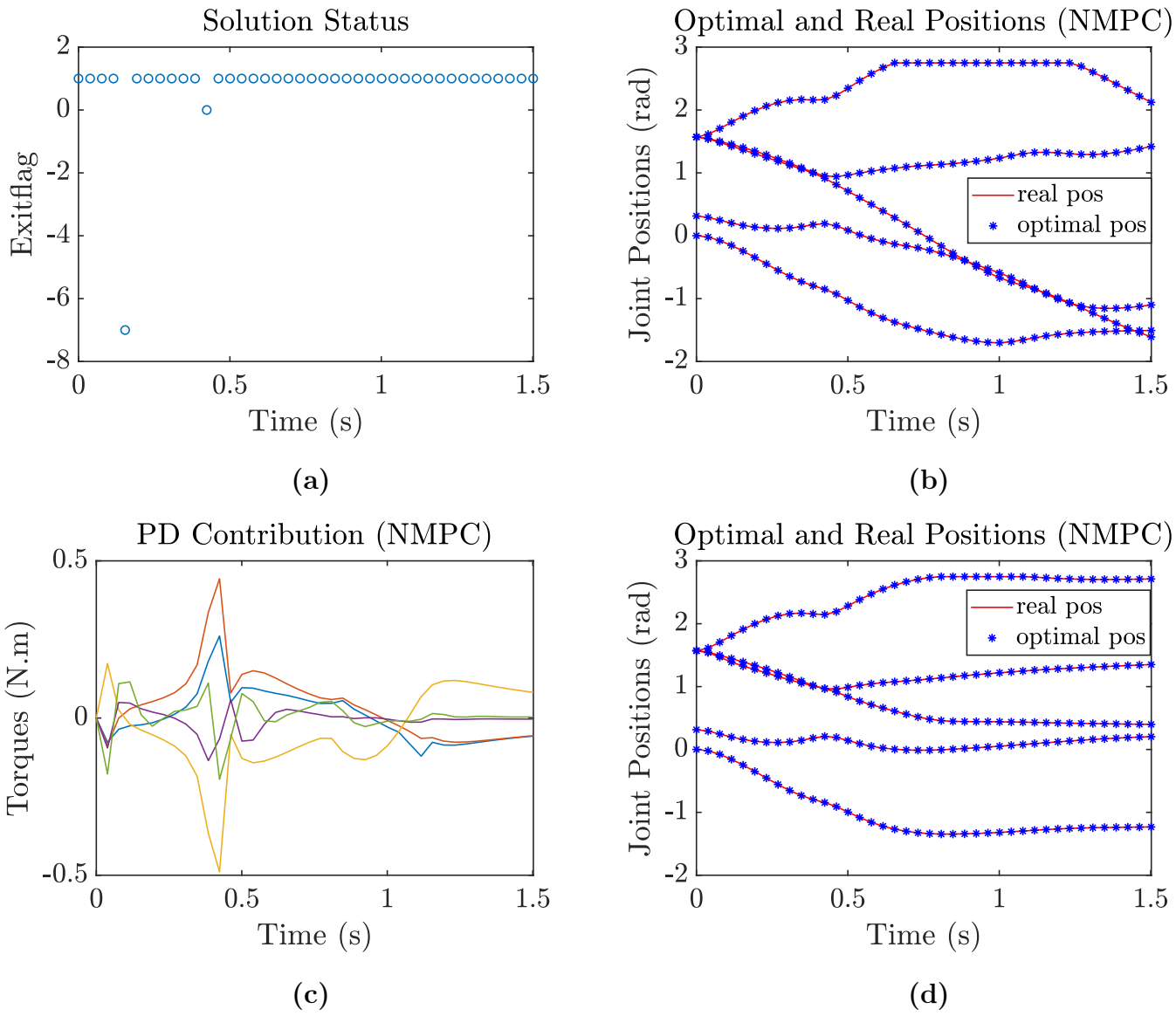


Figure 5.16: **(a)** Solver exit status (0: maximum iterations reached, 1: local minimum found, 7: infeasible problem), **(b)** Optimal and real joint positions from NMPC without any PD-compensation, **(c)** PD-term contribution in total input torques for NMPC with PD-compensation, **(d)** Optimal and real joint positions from NMPC with PD-compensation

But still, the contribution of this extra term is quite low (relatively low gains were used: $k_p = 5$ and $k_d = 0.1$) when compared to that applied to the open-loop optimal trajectory as shown in Figure 5.17 (gains of around $k_p = 20$ and $k_d = 1$ to obtain the response shown in plot (b)). In fact, computing the L_2-norms of the total input signals reveals the efficiency of the NMPC scheme (in terms of energy cost minimization) when compared to the open-loop NLOC solution.

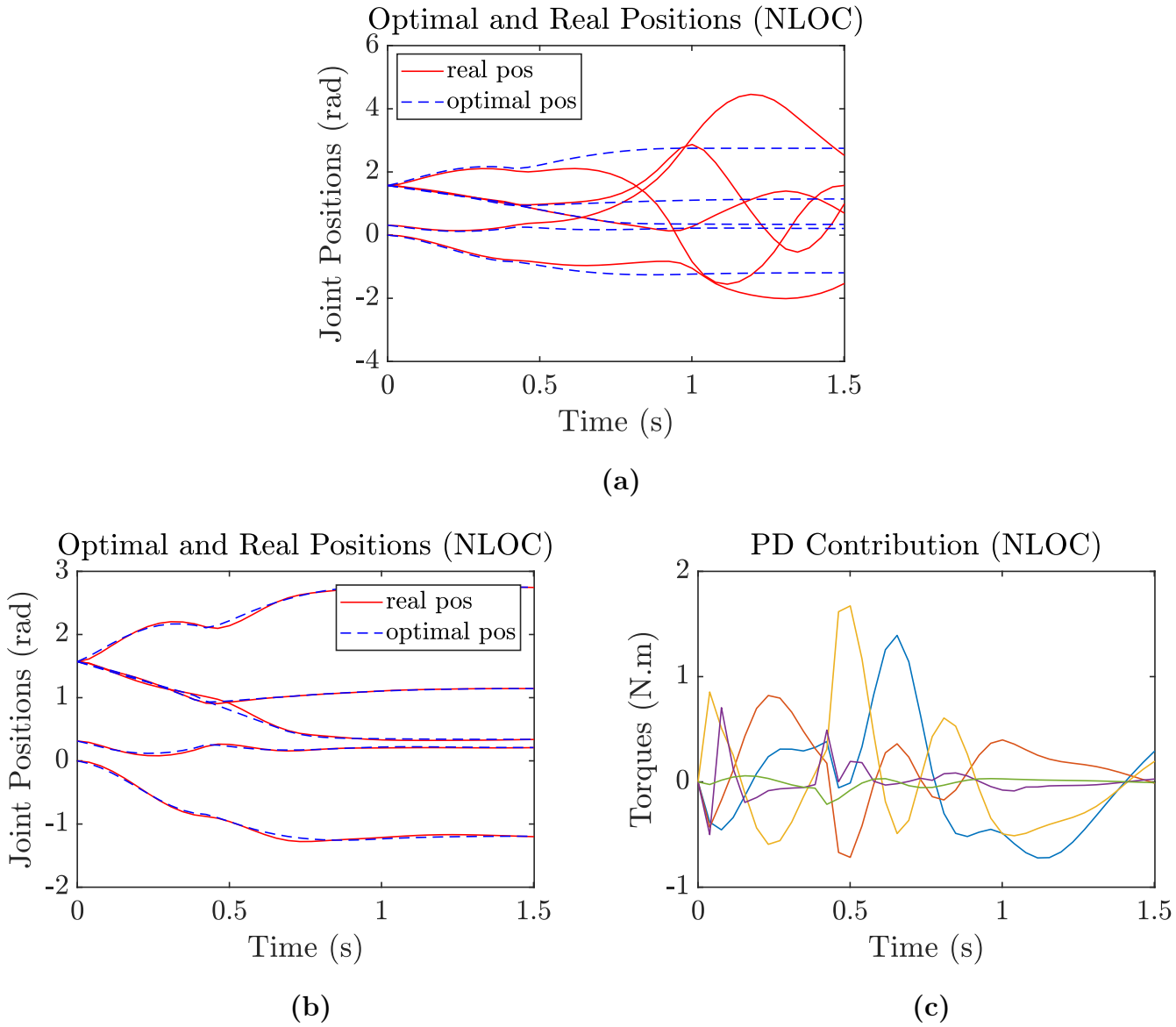


Figure 5.17: **(a)** Optimal and real joint positions from open-loop NLOC without any PD-compensation, **(b)** Optimal and real joint positions from open-loop NLOC with PD-compensation, **(c)** PD-term contribution in total input torques for open-loop NLOC with PD-compensation

Nonetheless, there are still clear similarities between the two approaches that mainly appear when looking at the resulting optimal reference-position trajectories, the optimal contact force sequence, and the resulting ball motion (see Figure 5.18) which happens to satisfy the desired requirement through the use of both controllers.

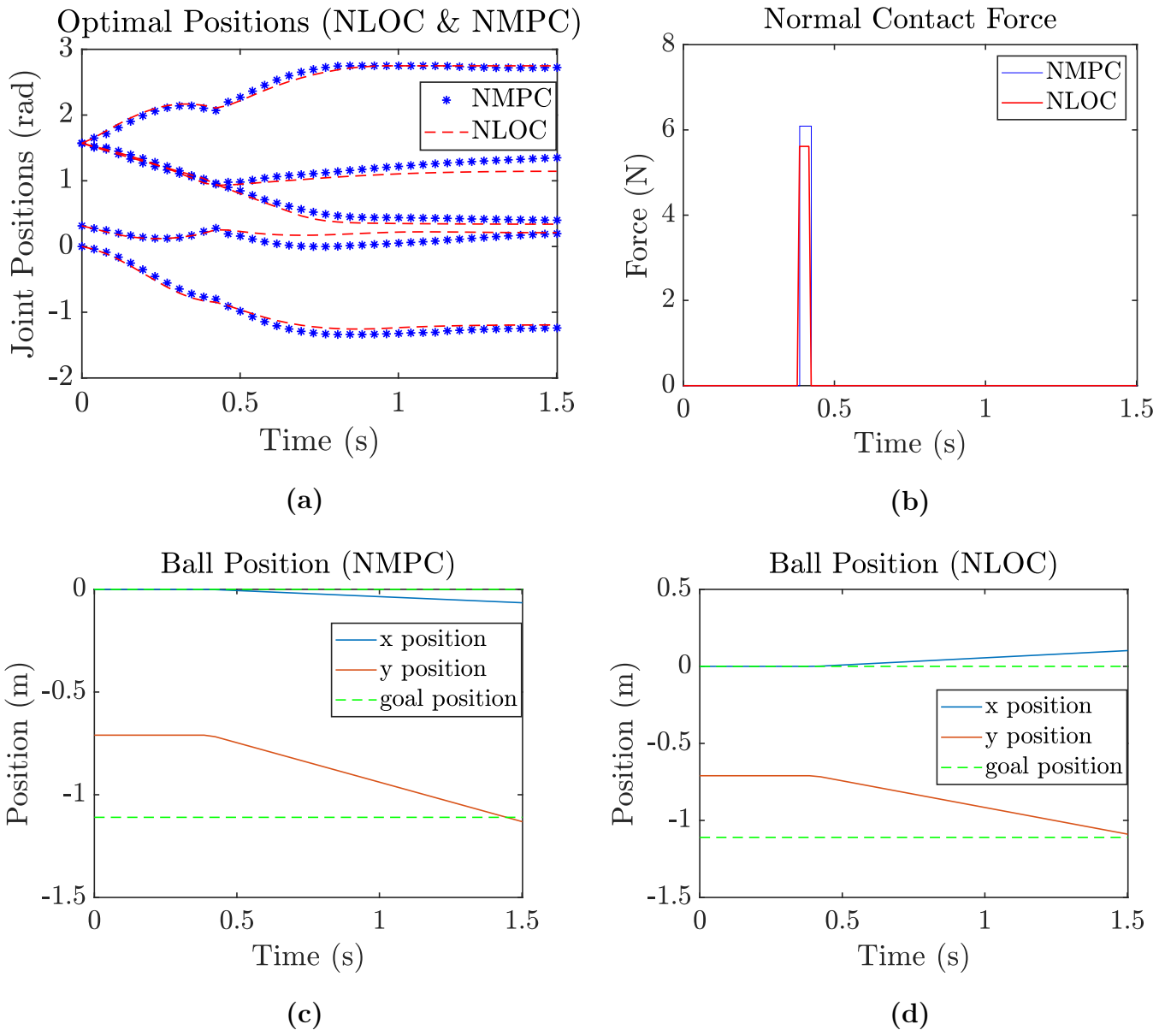


Figure 5.18: **(a)** Comparison of optimal joint positions obtained from open-loop NLOC and NMPC, **(b)** Comparison of their contact force trajectories, **(c)** Simulated ball position for NMPC, with PD-compensation, **(d)** Simulated ball position for open-loop NLOC, with PD-compensation

Chapter 6

ANYpulator-Block Problem

6.1 Problem-Formulation Specifics Including Dry Friction

In this last example, which is the central application of this thesis, ANYpulator is expected to dynamically push a block with *length* (l) = 23 cm, *width* (w) = 23 cm, *depth* (d) = 29 cm, and *mass* (m) = 1.4 kg such that it's center of mass ends up within a desired narrow position-range ($x_{desired}$ $\pm$10 cm) at the end of the time-horizon. What makes this problem interesting, is that unlike the two previous examples, dry friction plays a fundamental role in determining how the manipulation task plays out. One should note that we are not referring to the frictional forces arising in the tangential contact plane between the end-effector and the block-face, but rather to the frictional force arising from the interaction of the block with its own environment. The intricacy here is brought up by having to adopt a proper friction model, that could also be included into our contact-implicit optimization program while making sure not to overwhelm our gradient-based solver with any discontinuous or non-smooth constraints. Going into the realm of memoryless static friction models, one could clearly see the intrinsic non-smooth nature of frictional phenomena from Figure 6.1. In this work, we rely on a simple Coulomb friction model, that would lead to the following set of equations (set-valued force law) when deriving the equations of motion for the block in this problem:

Noting that $x_o = q_{o_1}$, $y_o = q_{o_2}$ (note that those coordinates do not indicate the position of the block COM, but rather the position of the block-face belonging to the separating plane), and $\theta_o = q_{o_3}$. Therefore, we extract the three last rows from Equation (5.21) to account for the generalized forces resulting from the normal contact force on the block.

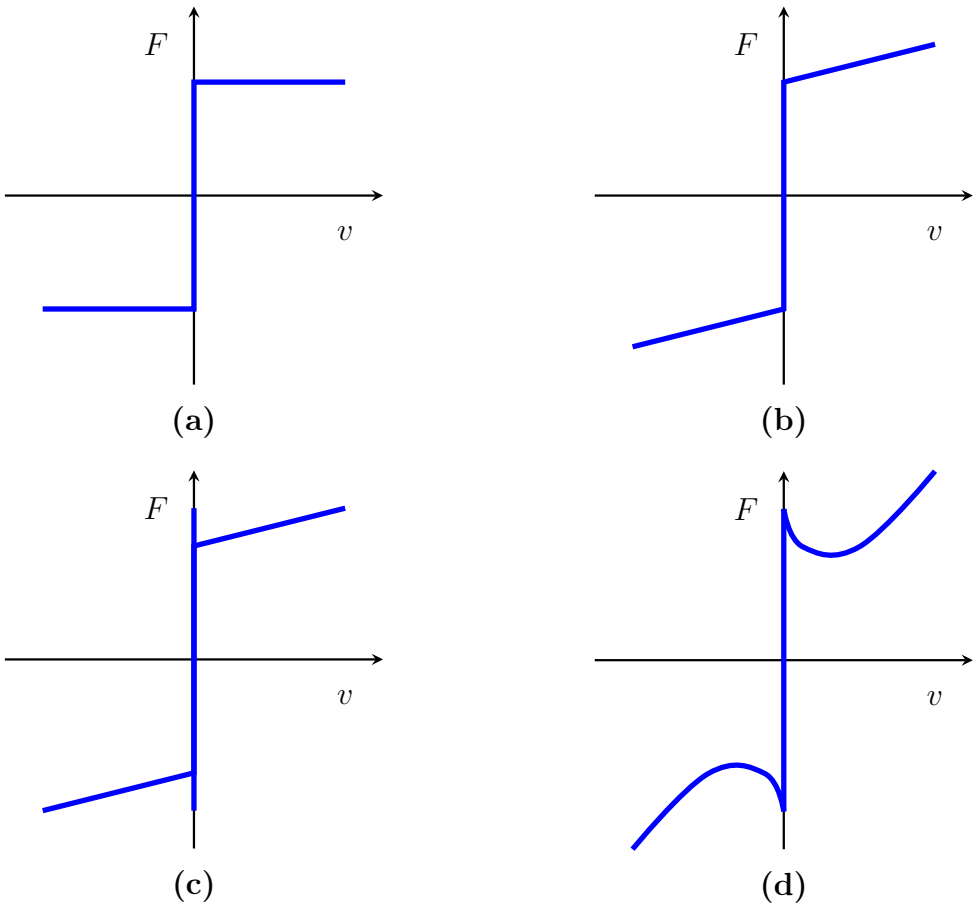


Figure 6.1: Various Static Friction Models: **(a)** Coulomb Friction, **(b)** Coulomb with Viscuous Friction, **(c)** Stiction with Coulomb and Viscous Friction, **(d)** Stribeck Effect [34]

$$m \begin{bmatrix} \ddot{q}_{o_1} \\ \ddot{q}_{o_2} \\ \ddot{q}_{o_3} \end{bmatrix} = \begin{bmatrix} \sin(q_{o_3}) \\ -\cos(q_{o_3}) \\ (q_{o_1} - x_E(q_r))\cos(q_{o_3}) + (q_{o_2} - y_E(q_r))\sin(q_{o_3}) \end{bmatrix} \cdot \lambda - \begin{bmatrix} \sin(q_{o_3}) \\ -\cos(q_{o_3}) \\ 0 \end{bmatrix} F_f \tag{6.1}$$

And

$$\begin{cases} F_f = \lambda & \text{if } v_{dir} = 0 \text{ and } \lambda < F_s \\ F_f = F_s & \text{if } v_{dir} > 0 \end{cases} \tag{6.2}$$

where F_f is the actual frictional force acting on the block, v_{dir} is a scalar

quantity that signifies the velocity along the direction of motion ($v_{dir} = \dot{q}_{o_1} \sin(q_{o_3}) - \dot{q}_{o_2} \cos(q_{o_3})$) *– which is parallel to the x-axis of the object frame – and* F_s *is the static friction force or the breakaway force and it is related to the normal force* N *by the so-called coefficient of static friction* μ_s ($F_s = \mu_s N$).

It is noteworthy to mention that even though a frictional torque does exist and acts along the q_{o_3} generalized direction, it is neglected in our problem formulation since, as will be seen later, we require that the block does not rotate at all while achieving our task. Furthermore, generally speaking, the second condition in Equation (6.2) usually takes into account the case where the velocity v_{dir} is non-zero, and so the expression for F_f would be ($F_f = sgn(v_{dir})F_s$). However, for the sake of our dynamic manipulation tasks, the scenario where $v_{dir} < 0$ is non-existent, since for any given block-orientation q_{o_3}, there is a unique irreversible direction along which the object could travel (otherwise, it would mean that the robot is also capable of pulling the object and not just pushing it).

Now in order to be able to utilize any friction model, its constituent parameters have to be determined first. In this case, we perform an identification experiment that would allow us to determine the friction coefficient μ_s corresponding to the surface interaction between our block and the table that will be used in our manipulation experiments. The required identification procedure is fairly simple and straightforward:

1. Measure the mass of the block (m)
2. Use a simple and portable load measurement device (for example a portable luggage scale)
3. Attach this device to the block and start pulling it slowly and gradually until the block starts moving (the reading on the device indicates the breakaway force F_s either directly or indirectly by displaying a mass-value instead)
4. Compute the friction coefficient with the formula $\mu_s = \dfrac{F_s}{mg}$
5. Repeat the above steps a couple of times, then take the average of all the obtained results

In our case, the static friction coefficient turned out to be relatively small ($\mu_s = 0.13$).

When wanting to incorporate the resulting friction model within a direct trajectory optimization formulation, the conditional if-statements presented in Equation (6.2) should be avoided. One way to go around that would be to smoothen the friction-velocity relationship (for example by using a sigmoid function); however, such a modification would render the problem stiff, and could, therefore, lead to significantly slow and inefficient convergence. In this application, the friction-law was encoded within the CIO formulation by the addition of the following constraints:

$$\left\{\begin{array}{l} v_{dir}\,(F_s - F_f) = 0 \\ F_s - F_f \geq 0 \\ v_{dir} \geq 0 \\ (F_s - F_f)(\lambda - F_f) = 0 \end{array}\right\} \iff 0 \leq v_{dir} \perp (F_s - F_f) \geq 0 \tag{6.3}$$

where F_f is now added to the vector of optimization variables in the optimal control problem.

A sufficiency proof is provided below:

Proposition:

If System (6.3) is true $\implies$ System (6.2) also holds true

Proof:

(a) For the case where $\lambda < F_s$ and $v_{dir} = 0$ (which is the first condition of Equation (6.2)), the satisfaction of (6.3) means that we can either have $F_f = F_s$ or $F_f = \lambda$. But having $F_f = F_s$ with $\lambda < F_s = F_f$ leads to the contradiction: $v_{dir} < 0$ (from the dynamic equations of the block given in (6.1)). Therefore, $\boxed{F_f = \lambda}$

(b) For the case where $v_{dir} > 0$ (which is the second condition of Equation (6.2)), it should hold from the first equation in Equation (6.3) that $\boxed{F_f = F_s}$

Now we discuss the remaining considerations that had to be made while building our CIO formulation for this problem:

– In order to ensure that the robot end-effector actually comes in contact with the block-face, the following nonlinear inequality conditions were added:

$$\begin{cases} \lambda \cdot (l - z_E(q_r)) \geq 0 \\ \lambda \cdot \left(\left(\frac{w}{2} \right)^2 - \psi^2(q) \right) \geq 0 \end{cases} \tag{6.4}$$

we recall that $\psi(q)$ has already been introduced in the previous chapter as the projection of the position vector r_{EO}^{O} onto the y-axis of the object frame {O} (see Figure 5.5).

System (6.4) can be interpreted as allowing a non-zero contact force to act only if the tool frame origin lies within the block's width and length. However, we would like to additionally impose that the block's rotational velocity ($\dot{q}_{o_3}$) is always zero as the desired translational motion is carried out. As a result, the optimal trajectories are expected to produce a motion plan that involves the robot contacting the block within the central line's length only (the vertical line having the same length as the block and passing through the center of its width).

– Taking into account that the final designed formulation will be implemented on the real system and not just in simulation, the accuracy of the system's dynamic evolution within the NLP was improved by decreasing the integration time step from h to $h/2$. This was not done by doubling the total number of stages, but rather by implementing two semi-implicit Euler steps within each interval. Moreover, recalling that the tangential frictional forces arising due to contact events are neglected in our problem, a condition was added to confine the tool's velocity vector along the normal direction, during a closed contact (so no sliding along the tangential contact plane is allowed). In fact, the tangential direction that is along the y-axis of the object's frame is already taken care of by Newton's restitution law; however, the one along the z-axis is not considered. Hence, we add the following nonlinear equality constraint:

$$\lambda \left([0 \;\; 0 \;\; 1] \cdot J_P^{(E)} \dot{q}_r \right) = 0 \tag{6.5}$$

– Finally, after doing all of the required modifications, it is quite inevitable to encounter a deterioration in speed and some convergence difficulties. An effective way to aid the solver was by relaxing the tolerance on the stationarity residuals (as previously discussed in the ANYpulator-Door problem of Chapter 5) and through the adopted initialization scheme. To elaborate on the latter idea, we start by recalling that the introduction of the modified CIO approach in Section 4.3 was based on adding Newton's restitution law for feasibility guarantees (this will also be verified in the next Section), but we also mentioned that this could make it more challenging for the solver to attain convergence as fast as before. This idea was tested on the current problem, and it appears that if one starts with a proper initial guess (the guess that leads to a local minimum after a number of iterations), then convergence speeds are quite similar for both formulations. However, for a poor initial guess, getting rid of the impact law slightly improves (based on observation) the total convergence speed (after solving the NLP multiple times while continuously updating the initial guess). Therefore, what was eventually done was to introduce a boolean variable as a real-time parameter (just to avoid having to re-generate the code again due a change in the formulation) in Equation (4.12), that would first "turn-off" the no-rebound condition, solve the problem, then "turn-on" the condition and solve the problem again using the previous solution as an initial guess.

6.2 Results: Optimization, Simulation and Experimentation

Just to get an idea of how large is the CIO program needed to solve this manipulation task: we obtain a problem size that is defined by 896 optimization variables, 819 equality constraints, and 1955 inequality constraints! The resulting NLP is solved in around 800 ms (300 iterations) depending on which local minimum is attained (some minima required around 2 s to be found), when the proper initial guess is used of course. For poor initial guesses, the total solving time can reach up to 1 min, but generally not more than that. As for the simulations, the same considerations are applied here as for the other previous problems, but adding to that, the Gazebo option for specifying the static coefficient of friction was set such that it emulates our real-life example (*i.e.* $\mu_s = 0.13$).

Coming to the experimentation, several test runs were performed on variations of the ANYpulator-Block problem (*i.e.* Trying different robot initial conditions, different block starting configurations and different final desired positions for the block). Here we emphasize on three experiments that were tested 5 times each, and then analyze the results and draw conclusions from them. But first we mention a couple of essential points concerning the experimental procedure:

Since the time-horizon we are working with is considerably small (the robot is expected to perform the manipulation task in $T = 1.5$ s), initial experiments involved testing the resulting optimal trajectories by scaling them with respect to time: Starting with a larger time horizon and going down gradually to a scale of 1 (this was done due to safety considerations). Moreover, the block was not introduced at this stage of the experiments, since the feasibility of the robot joint-trajectories was our mere primary concern. In order to scale our trajectories (positions, velocities, as well as torques), formulas obtained from *Dynamic Scaling* had to be considered (see [43] for more details):

$$\textit{Given } t \in [0, T], \quad \textit{let } \ \theta = k \cdot t \ \Rightarrow \theta \in [0, kT]$$

So the new scaled optimal trajectories are now referred to as $\hat{q}(\theta), \hat{q}'(\theta)$, and $\hat{\tau}(\theta)$ and can be shown to be computed as follows (for each joint):

$$\begin{cases} \hat{q}(\theta) = q_{opt}(t) \\ \hat{q}'(\theta) = \dfrac{\dot{q}_{opt}(t)}{k} \\ \hat{\tau}(\theta) = g(\hat{q}(\theta)) + \dfrac{\tau_{opt}(t) - g(q_{opt}(t))}{k^2} \end{cases} \tag{6.6}$$

where $g(.)$ is the gravitational torque applied on the joint of interest.

Eventually, with gradual down-scaling and proper manual tuning of the PID gains, we were able to achieve satisfactory tracking results using the following gains (the best gains turned out to be given by a set of previously tuned default values): $K_p = [110 \quad 120 \quad 100 \quad 80 \quad 80]^T$, $K_d = [0.2 \quad 0.1 \quad 0.05 \quad 0.15 \quad 0.05]^T$, and $K_i = [0.02 \quad 0.025 \quad 0.015 \quad 0.03 \quad 0.03]^T$.

Another issue worth considering, was the proper placement of the block with respect to the robot's base frame, because if this was not the case, then the optimal trajectories to be tracked by the robot would either miss the block and not make contact with it, or would push it at the wrong place. So to have a somewhat accurate placement of the initial block pose, its starting position and orientation were provided as a target reference for the robot's tool frame, that was tracked with an *inverse kinematics controller*. After that, the block was placed accordingly, and then the robot was commanded to go to its own starting configuration before running our CIO controller.

Before showing our experimental results, a single simulation result is given for the mere purpose of comparing the CIO approach with and without Newton's restitution law. Even though both methods appear to find the same optimal contact force sequence (needed to achieve the task of pushing the block to the desired position), it is clear from the simulations in Figures 6.4 and 6.5 that one is actually feasible and the other is not (the one with no impact law doesn't even cause the block to move, while the other one sends the block really close to the target position). This infeasibility can be well-understood by looking at the differences in the optimal torques obtained from the optimization in Figures 6.2 and 6.3 (notice the jumps that occur in the torques at the contact event in Figure 6.3).

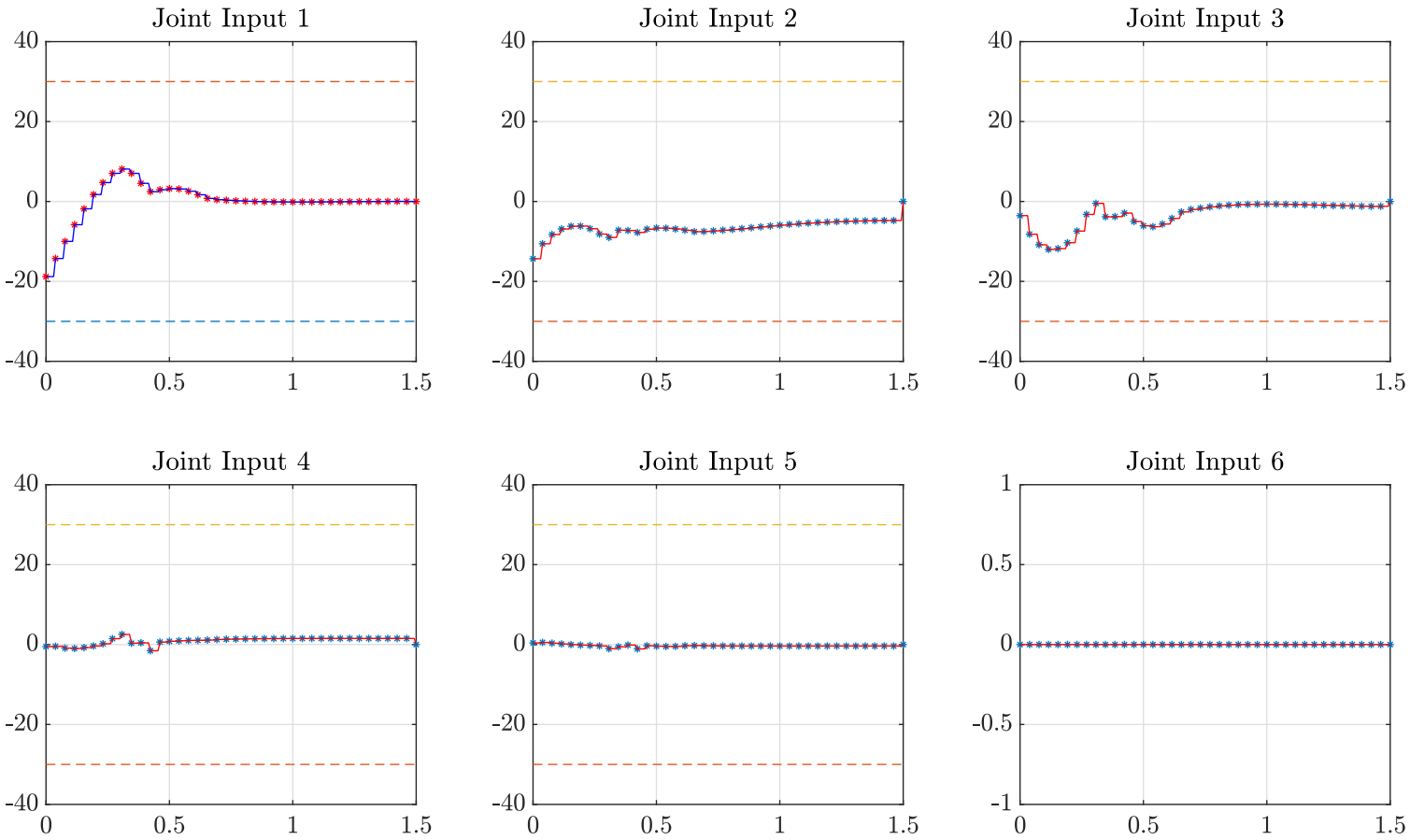


Figure 6.2: Optimal joint input torques resulting from a CIO formulation that does not include Newton's restitution law

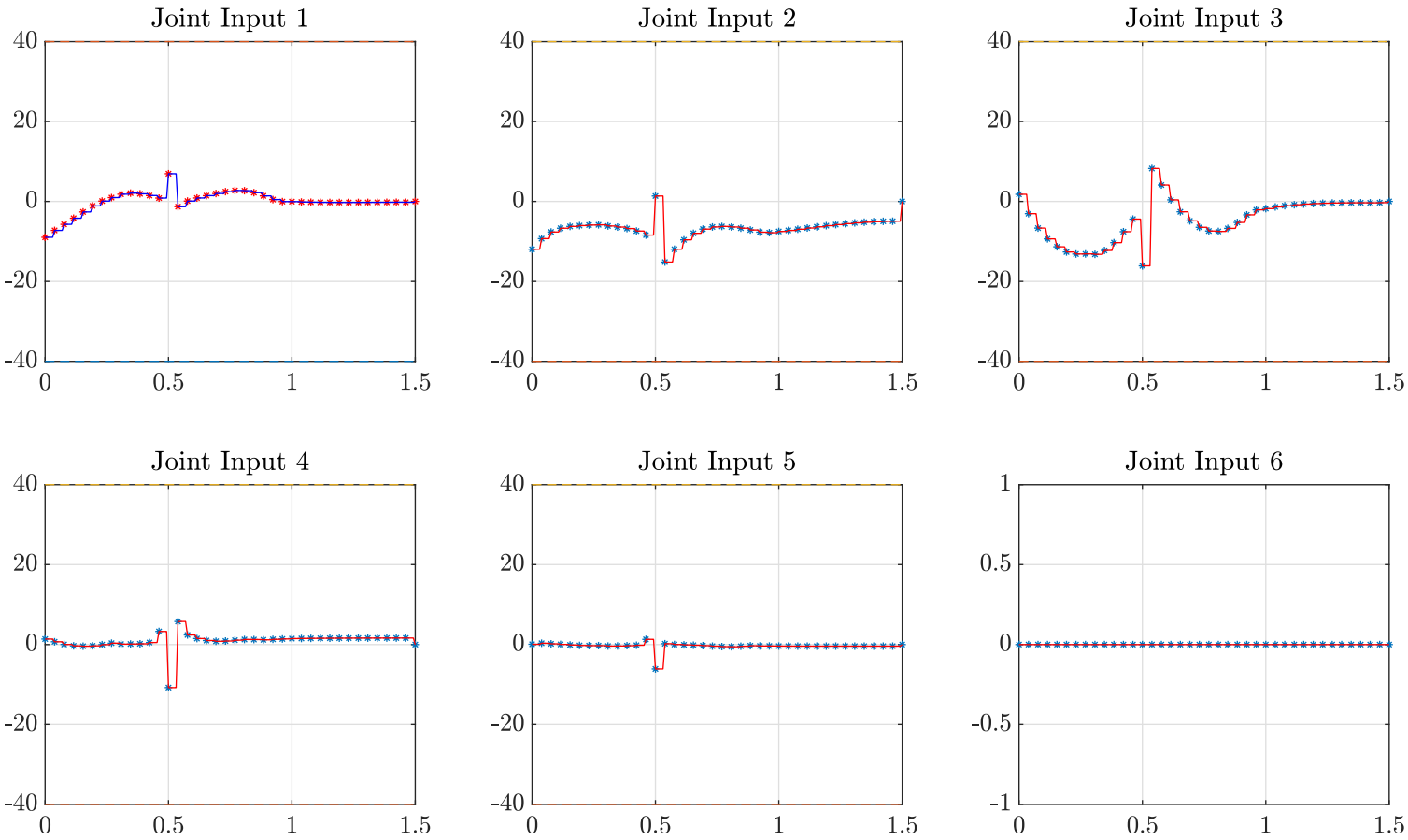


Figure 6.3: Optimal joint input torques resulting from a CIO formulation that includes Newton's restitution law

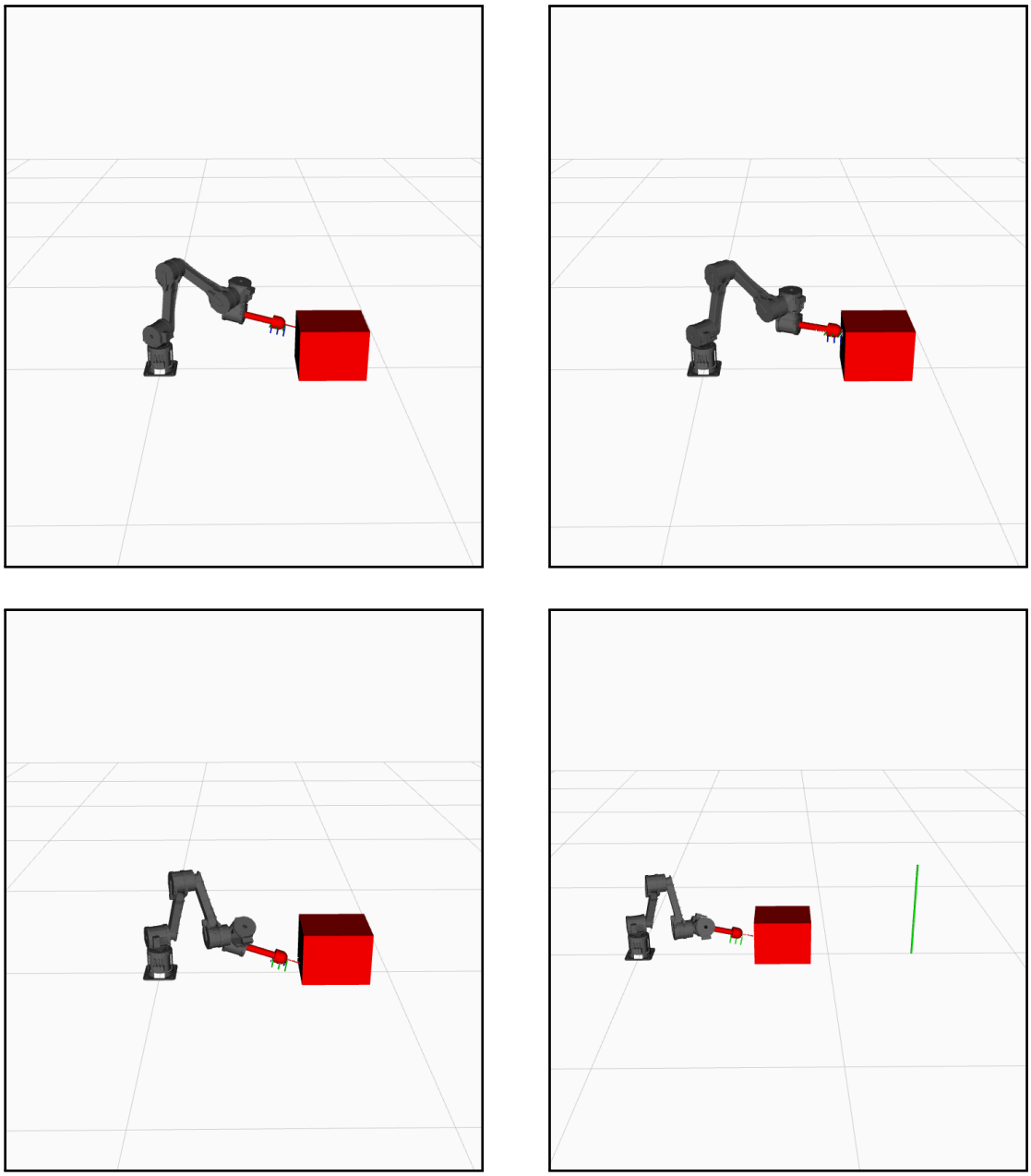


Figure 6.4: Snapshots of a simulation resulting from a CIO formulation that does not include Newton's restitution law

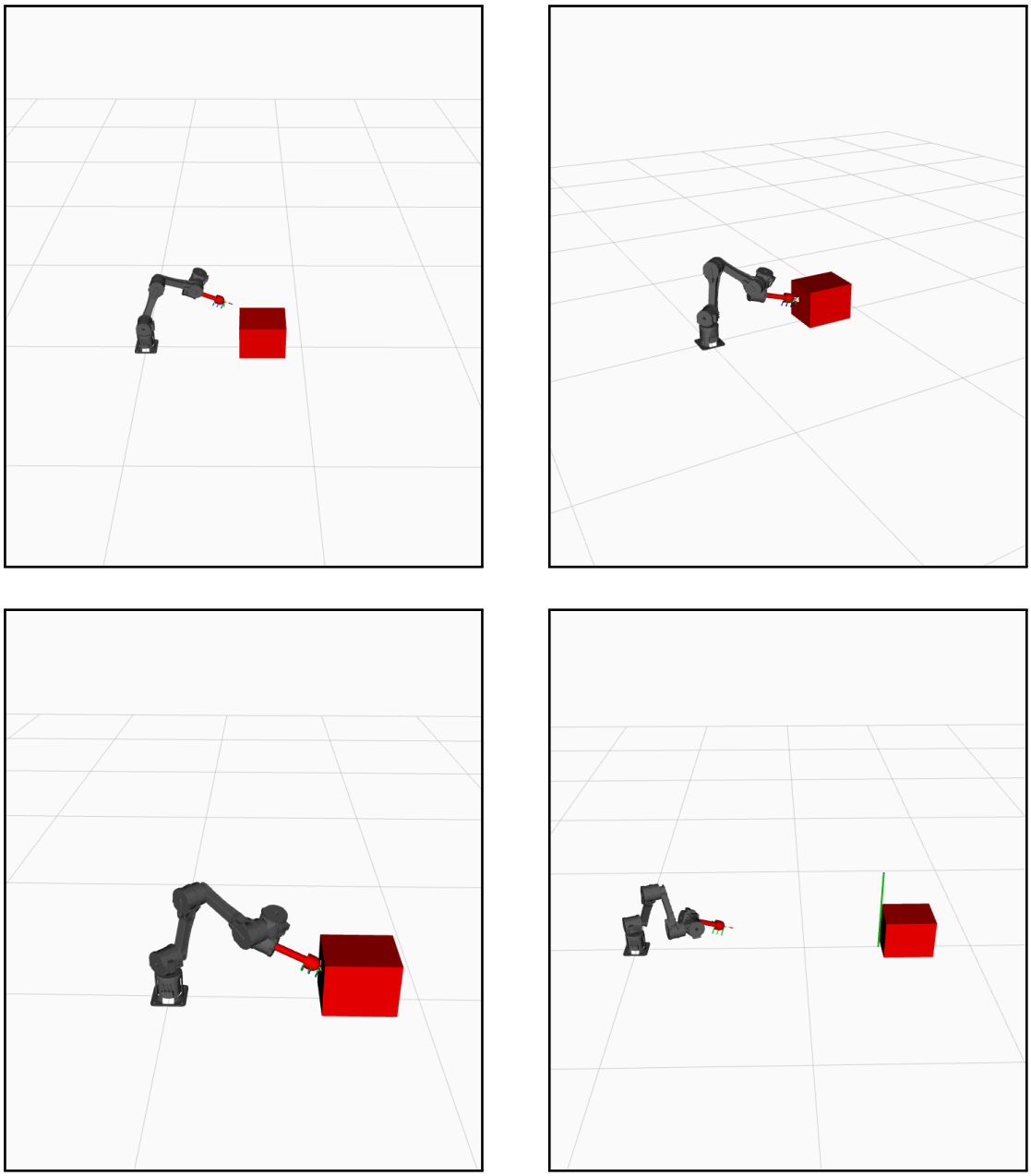


Figure 6.5: Snapshots of a simulation resulting from a CIO formulation that includes Newton's restitution law

As for the obtained experimental results, these will be distributed among three different parts, where we show, for all of them, the joint position responses, the end-effector (tool) position responses, and snapshots from their corresponding videos; in addition to that, a force/torque sensor was mounted on the robot end-effector in order to compare the contact forces generated on the real setup to the forces predicted by the optimization. The three performed experiments involve the same initial block pose and robot configuration; however, the desired displacement of the block varies among them.

In the first experiment, the expected block displacement (given from the optimization) is 0.7 m. The contact event involves a single impulsive push with a force of about 50 N acting over the whole time step (for $h = 0.0387$ s). On the other hand, the desired displacement for experiments 2 and 3 were reduced to 0.6 m and 0.4 m, respectively. Consequently, this led to a decrease in the maximum generated contact force. More interestingly, in the second experiment, we perceive a contact event that lasts for approximately $8 \cdot h = 0.3096$ s, while in the last one we encounter two separate contacts (one with a small magnitude and the other with a large one) held over a single time step each.

The reason why this information is highlighted is basically to pinpoint the richness of the CIO approach in general, as it yields contact schedules that couldn't be easily considered or planned within a multi-phase approach; and also to indicate the dynamic feasibility of the utilized method in three different settings by noting that the predicted contact sequence was indeed true as it was compatible with both the readings of the force sensor and the observation of the properly-tracked motion plan (from the videos as well as the joint and end-effector position plots). To elaborate, it could be noted from the Cartesian coordinates of the tool that the robot indeed reaches the block's initial position and tends to sustain the closed contact as much as specified by the optimal trajectories, while also maintaining a tool height greater than 5 cm throughout the full time-horizon. Specifically, recalling that the block is pushed in the negative y-direction, one could notice the different tool y-position responses among the three performed experiments. Furthermore, one should keep in mind that eventhough the force magnitudes in the normal contact force plots are not equivalent, the

resulting impulses, however, tend to be reasonably similar, as the larger forces (perceived from the experimental force/torque sensor-data) act over a smaller time-interval.

It is important to recall also that each experiment was repeated 5 times in order to demonstrate the method's repeatability and reliability (indeed this was shown to be true, as the same results as those presented below came up repeatedly, for various runs of the same experiment). A discussion on the method's accuracy is provided after the upcoming experimental results.

Experiment 1

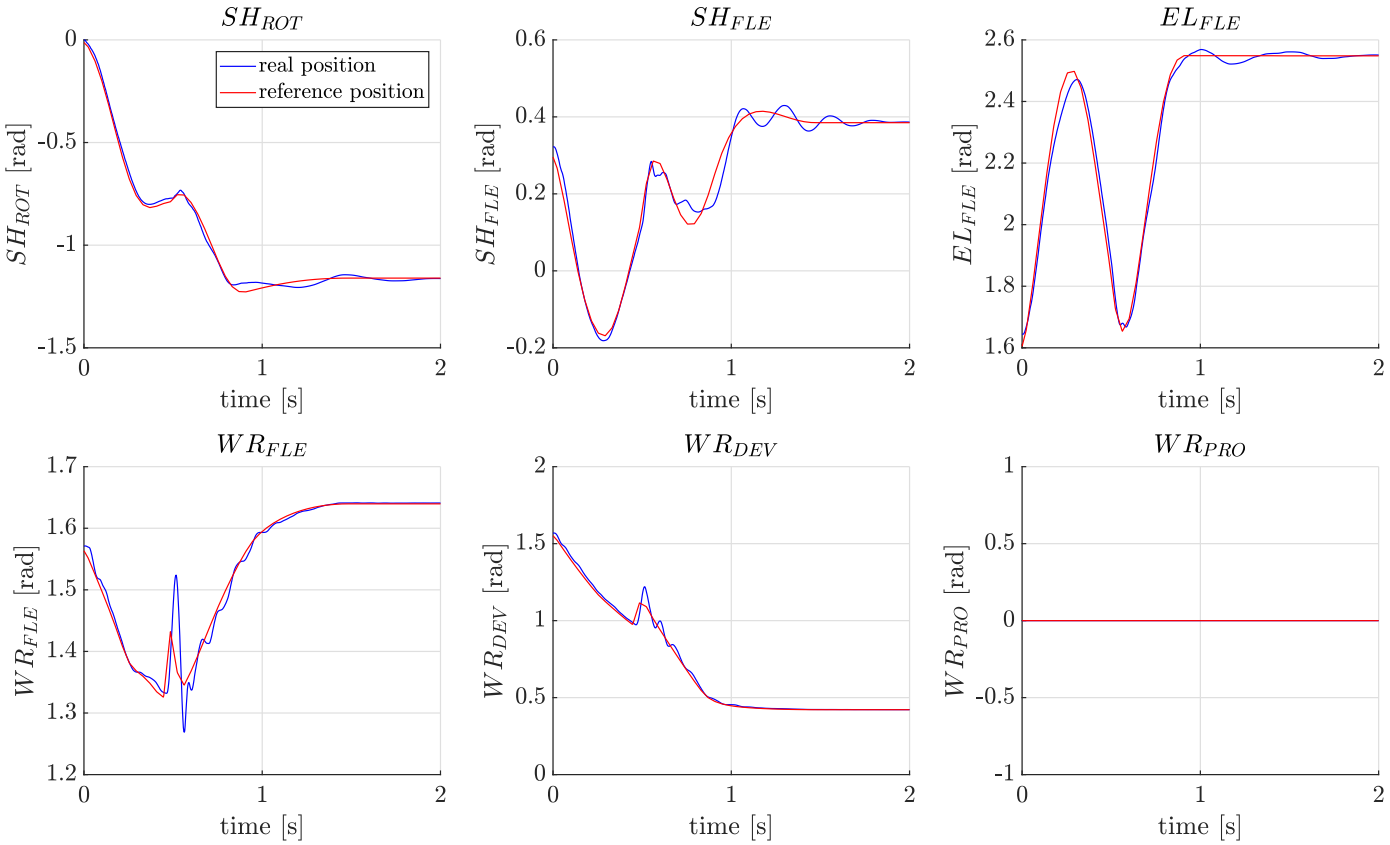


Figure 6.6: ANYpulator joint positions tracking optimal-reference trajectories (Experiment 1)

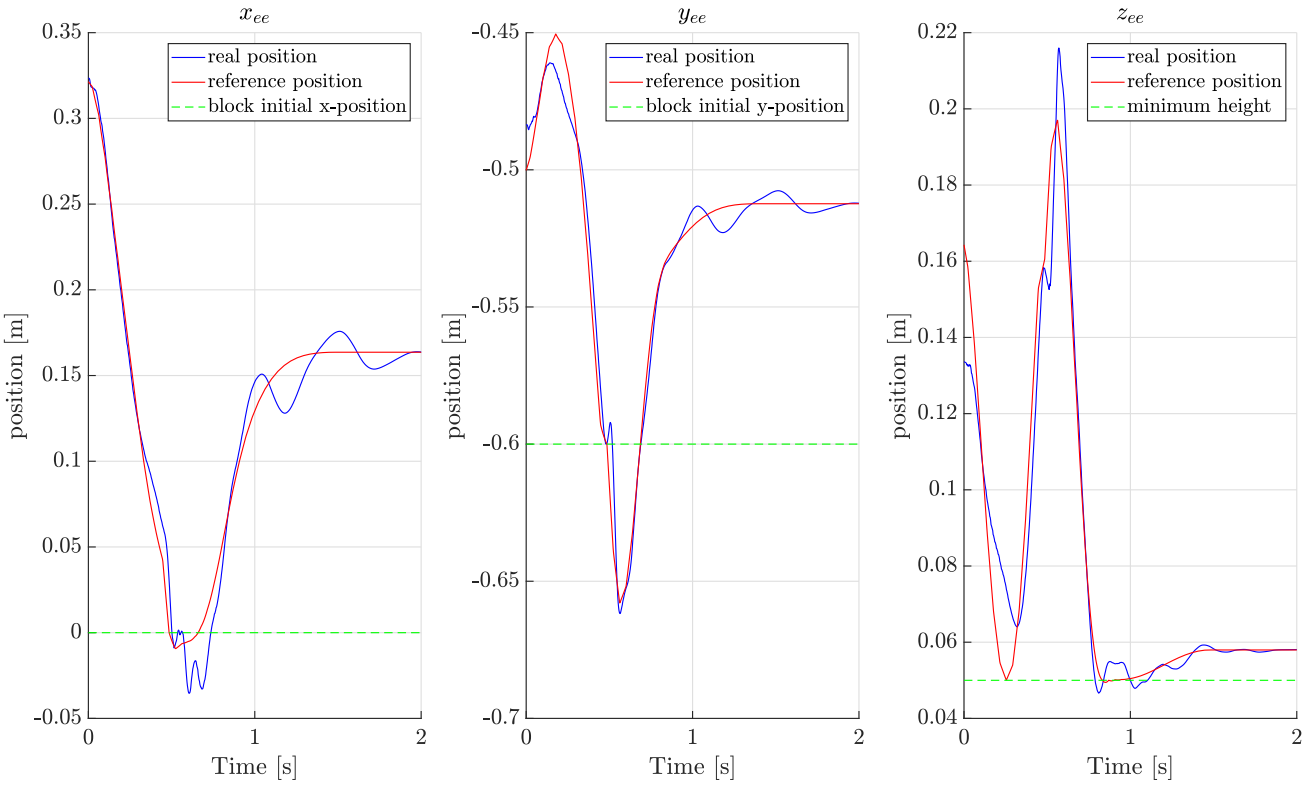
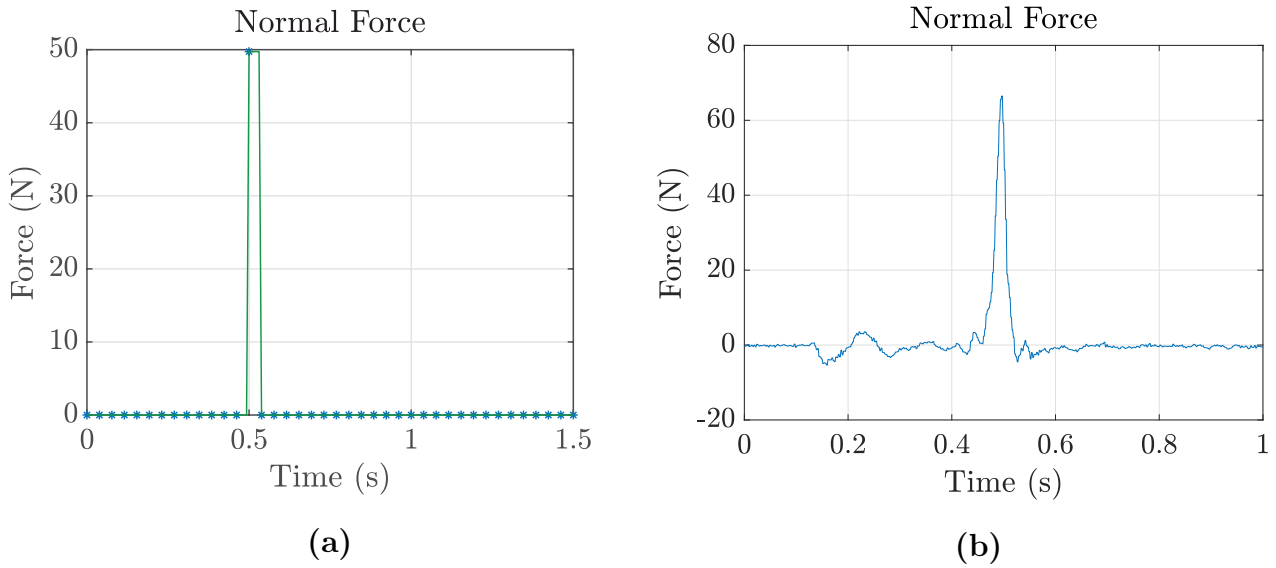


Figure 6.7: ANYpulator end-effector tool position (in cartesian coordinates), tracking optimal trajectories, given by forward kinematics (Experiment 1)



(a) **(b)**

Figure 6.8: Normal Contact Force: **(a)** Optimization Result, **(b)** Experimental Result (Experiment 1)

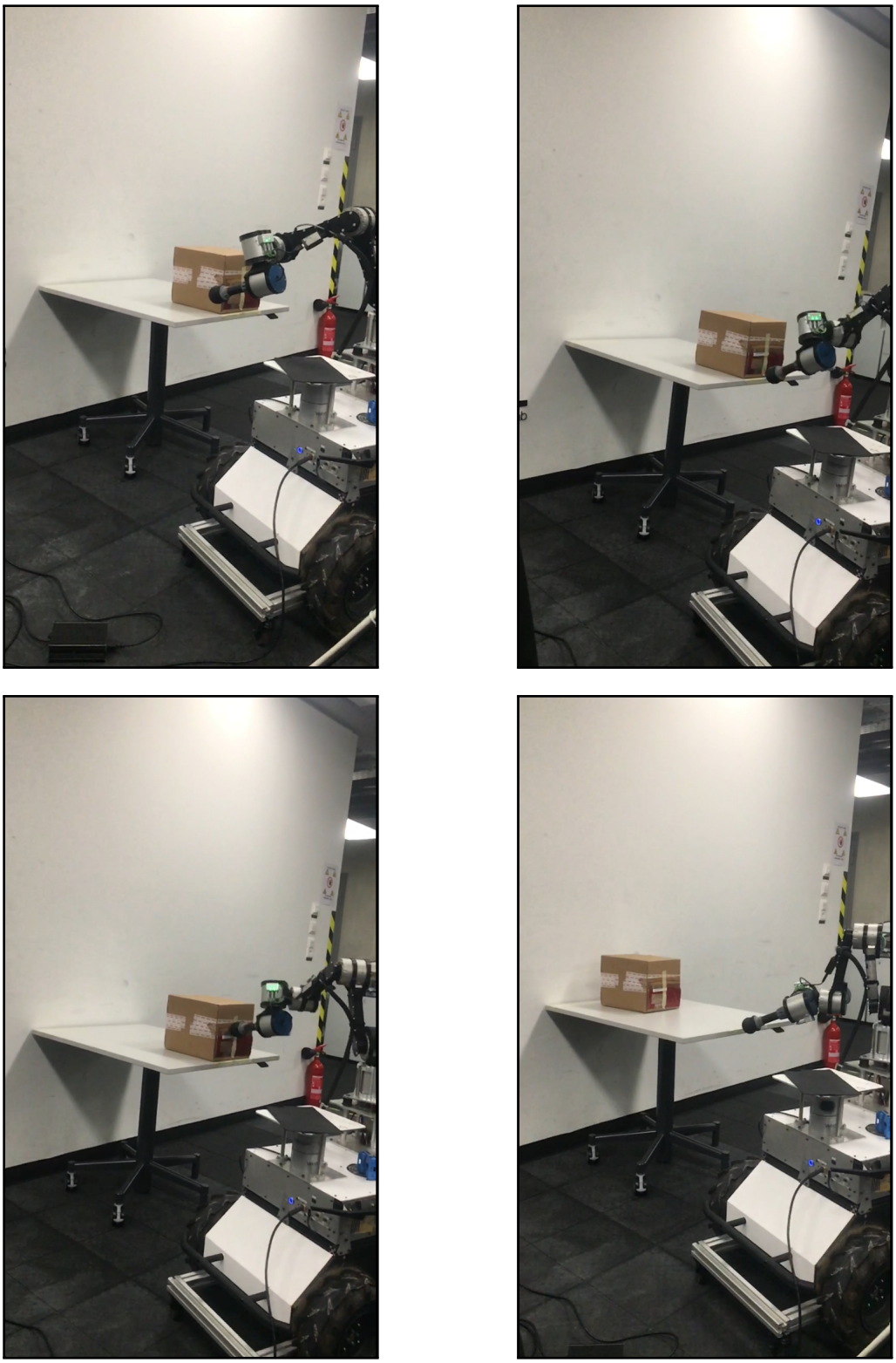


Figure 6.9: Snapshots for the manipulation task with a desired block displacement of 0.7 m (Experiment 1)

Experiment 2

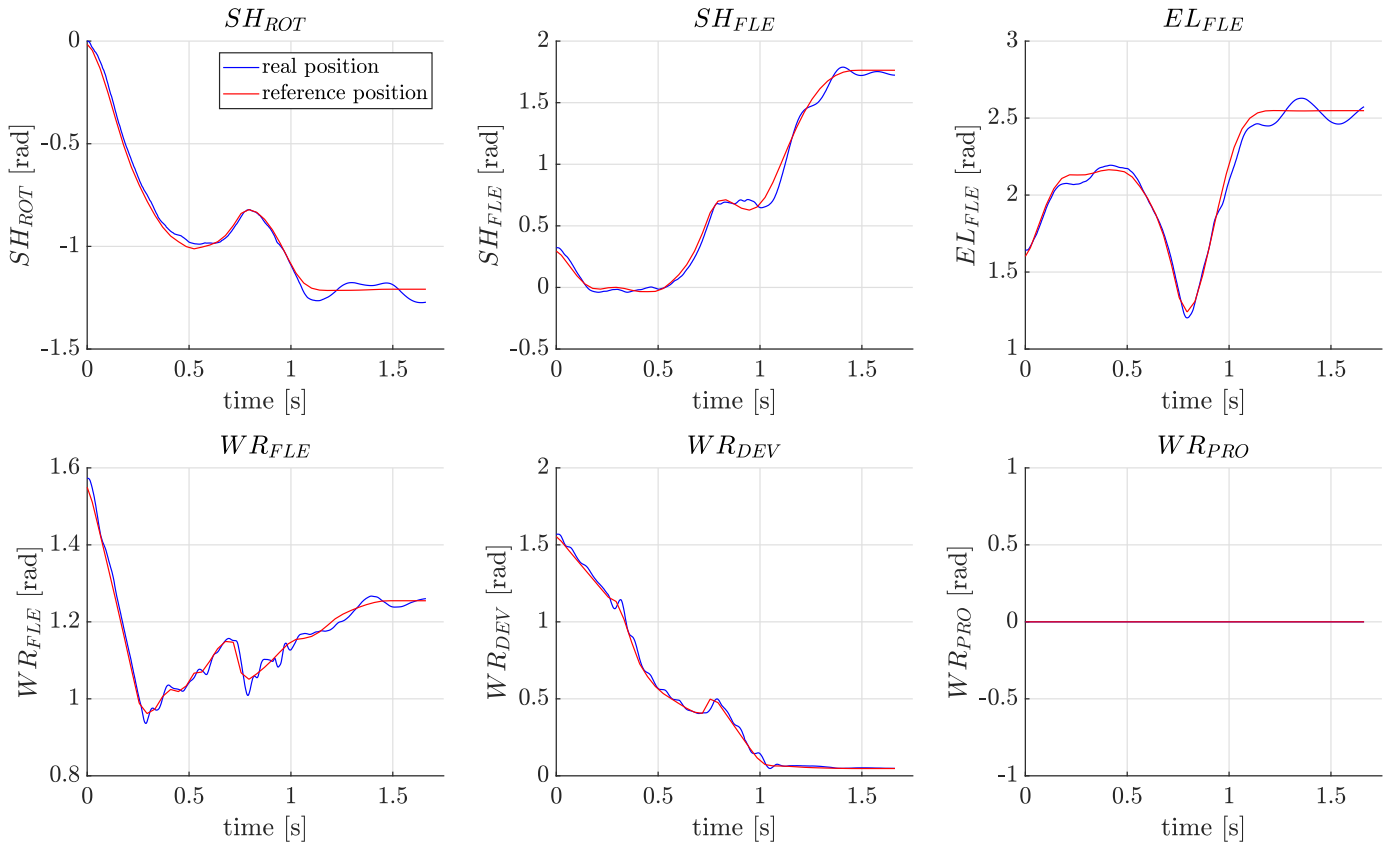


Figure 6.10: ANYpulator joint positions tracking optimal-reference trajectories (Experiment 2)

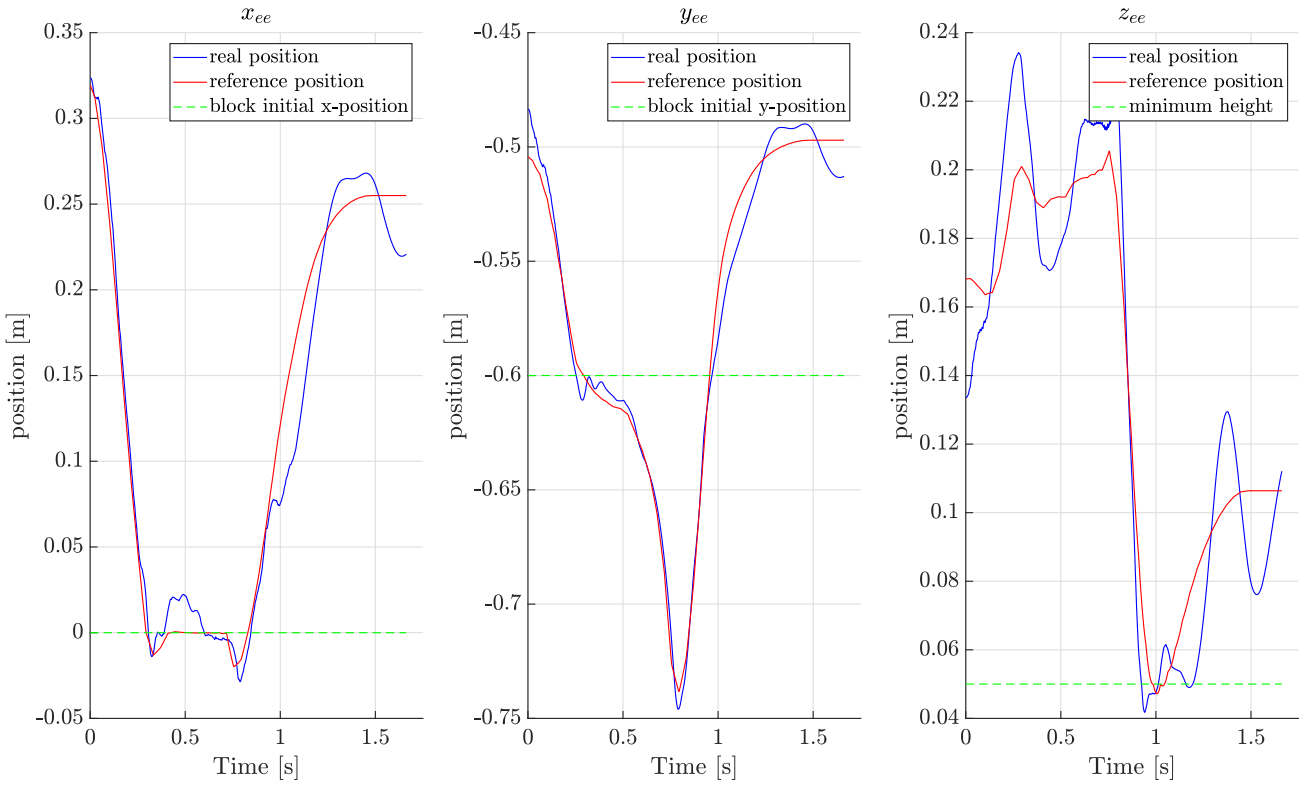


Figure 6.11: ANYpulator end-effector tool position (in cartesian coordinates), tracking optimal trajectories, given by forward kinematics (Experiment 2)

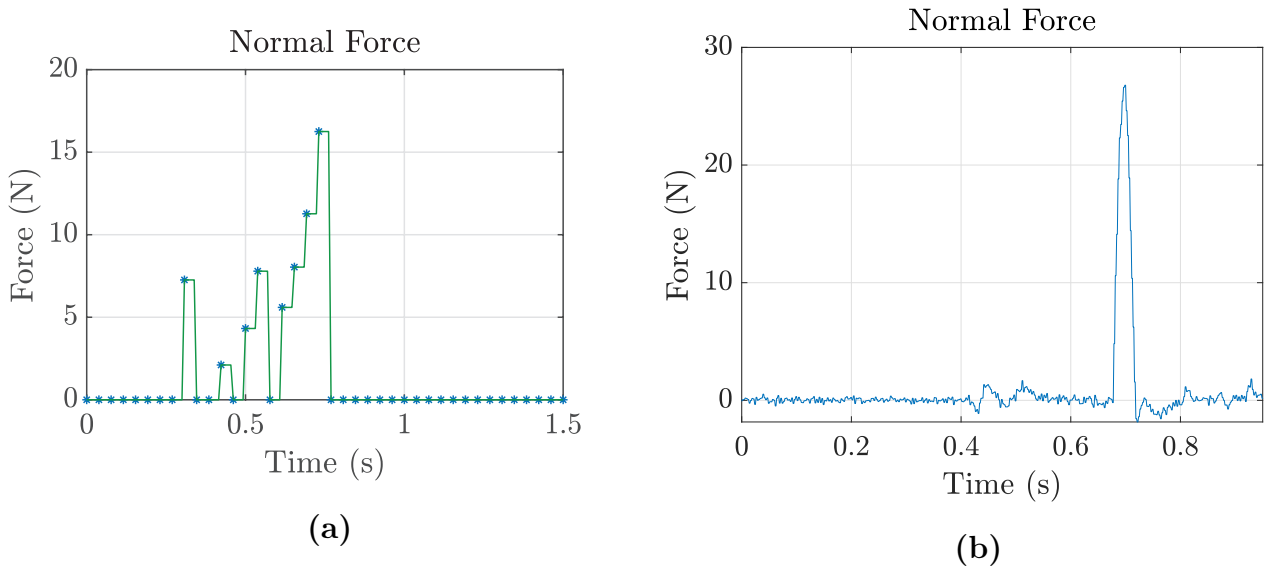


Figure 6.12: Normal Contact Force: **(a)** Optimization Result, **(b)** Experimental Result (Experiment 2)

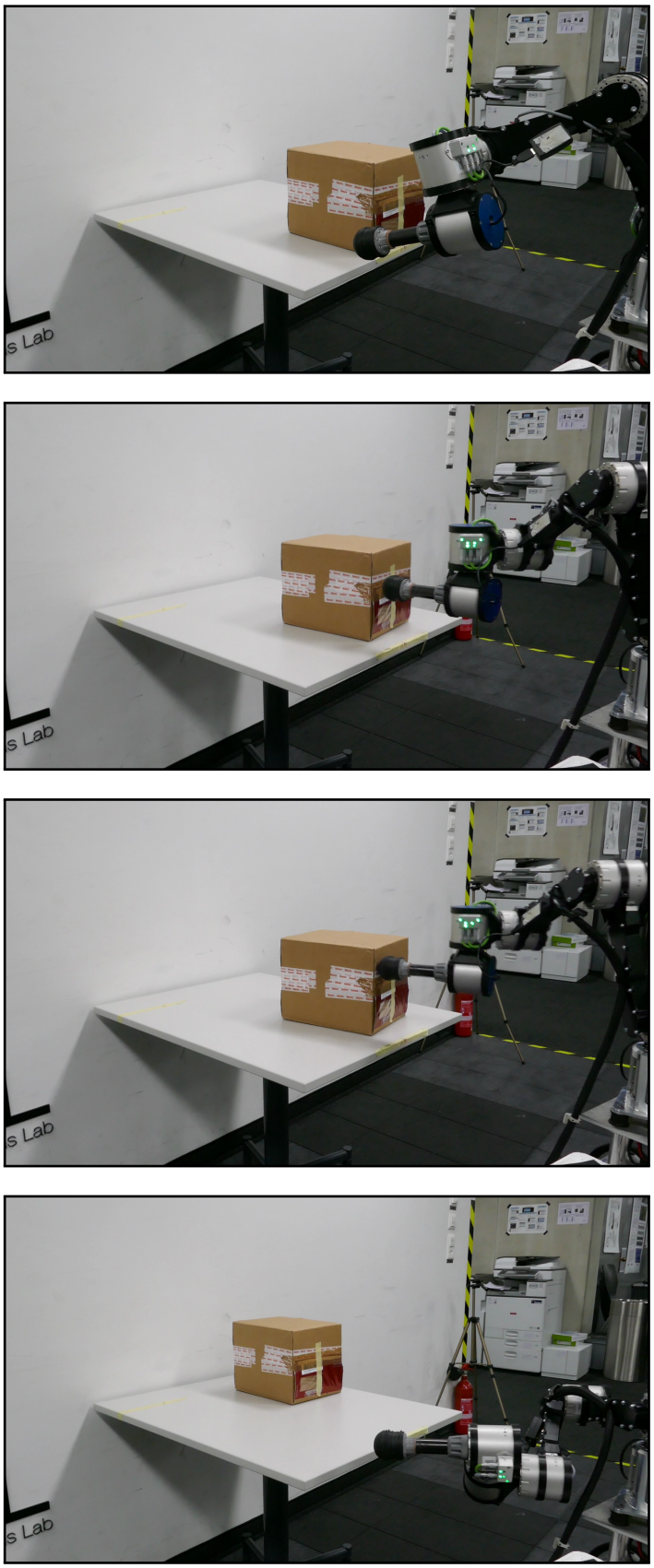


Figure 6.13: Snapshots for the manipulation task with a desired block displacement of 0.6 m (Experiment 2)

Experiment 3

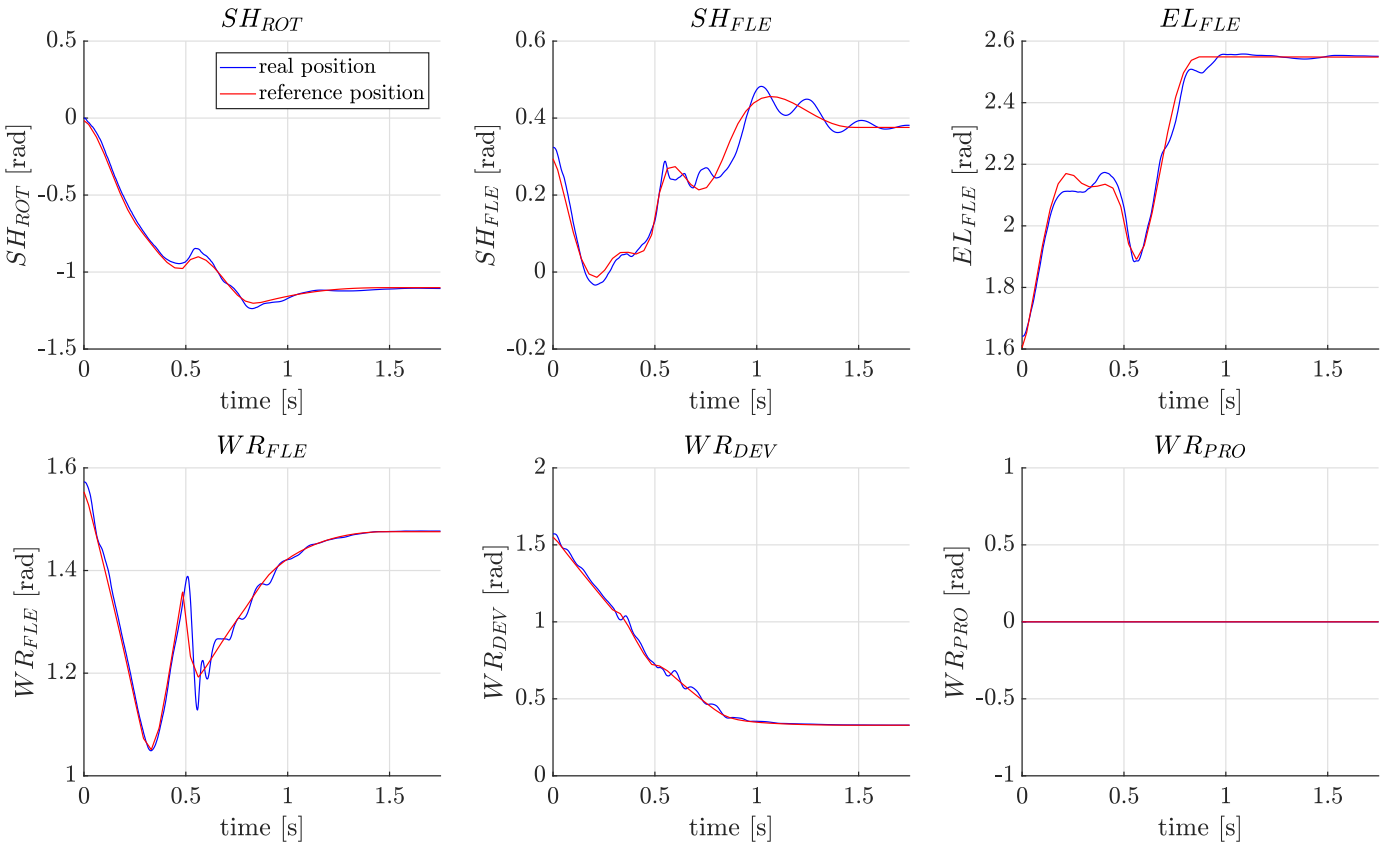


Figure 6.14: ANYpulator joint positions, tracking optimal-reference trajectories (Experiment 3)

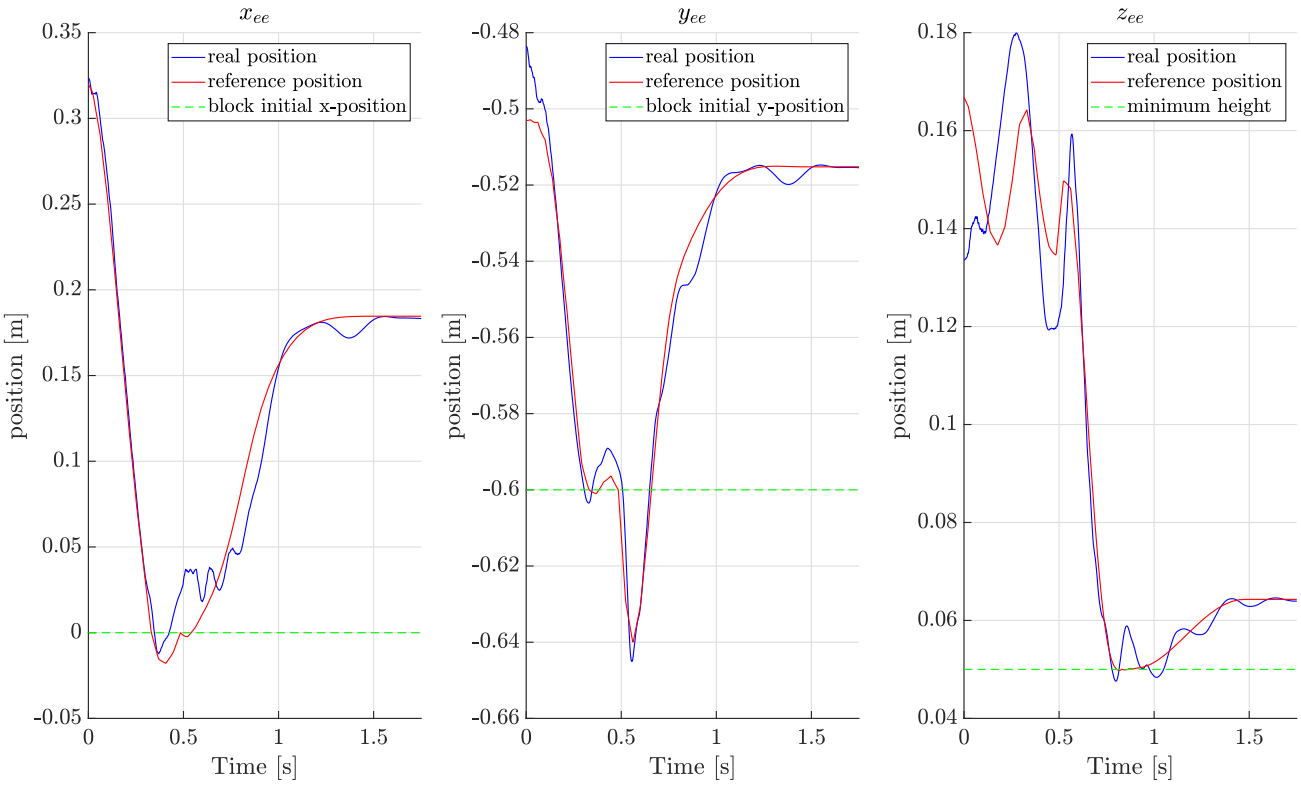


Figure 6.15: ANYpulator end-effector tool position (in cartesian coordinates), tracking optimal trajectories, given by forward kinematics (Experiment 3)

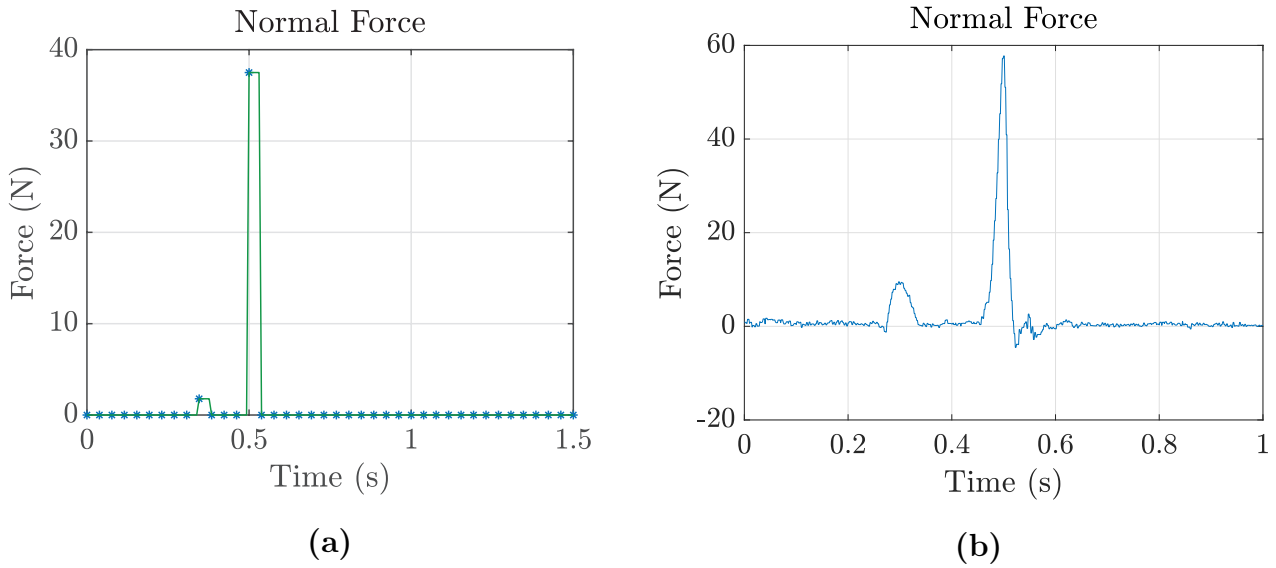


Figure 6.16: Normal Contact Force: **(a)** Optimization Result, **(b)** Experimental Result (Experiment 3)

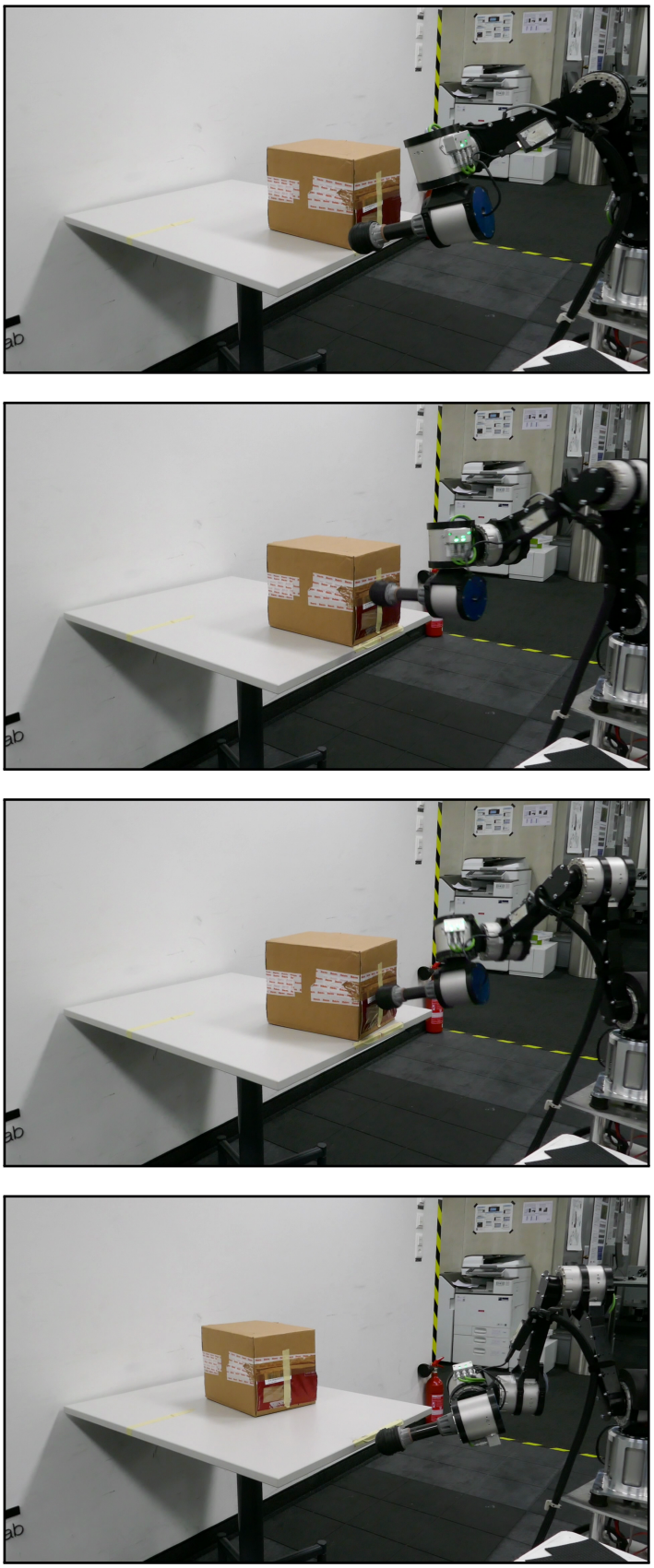


Figure 6.17: Snapshots for the manipulation task with a desired block displacement of 0.4 m (Experiment 3)

Richness, feasibility, and repeatability of the adopted CIO approach have been brought up in our discussion on the available experimental results. Now to add an accuracy-based evaluation into the picture, we refer to the plot shown in Figure 6.18, which indicates the final error (in absolute value) between the real block position and the final position that was anticipated as a result of the optimization. First of all, it was noticed that in all of our experiments, the block always travels a distance that is less than the expected one. This could be heavily related to the inaccuracy in our dynamic model in general, and specifically our impact model (which was assumed to be purely inelastic) and the assumed static friction model (in this case it seems that we have underestimated the frictional effects).

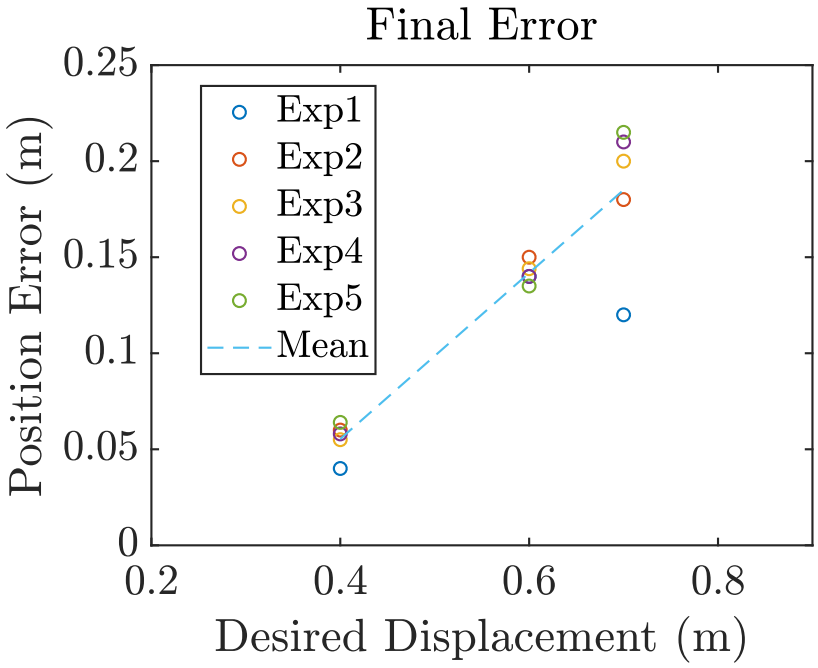


Figure 6.18: Observed error between the block's final position and the desired one, throughout the three experiments (performed 5 times each)

Secondly, a trend could be fairly noticed and elaborated upon: It is quite clear that the higher is the desired block displacement, the more erroneous is our manipulation-task satisfaction. This could be partly related to the fact that the higher the demanded displacement, the larger the generated contact force; and since there could be a slight deviation from the required contact location (the central line of the box), then this would lead to the block's rotation along with its translation. This, in turn, would make it lose a part of its total kinetic energy, thus having less energy left to go

through the whole required translational motion. Moreover, the neglected frictional forces acting in the tangent contact plane may as well lead to the block's undesired rotation. In case the contact force did not turn out to be larger, but started acting at an earlier time (to attain a farther away position at the end of the time horizon) this would also increase inaccuracy due to the greater accumulated integration error. A final comment to be mentioned concerning the method's accuracy is that some local minimums are in fact much worse than others (in the sense that they are not nearly as accurate as their counterparts). Most of the times, this turns out as a result of the poor modeling of the contacting bodies (which assumes in our case that it happens between a point and a plane), whereas in reality, the sides of the end-effector, and not just its tool-frame's origin, also can come into contact with potentially contacting bodies; and this is not taken into account in our formulation. That is why it is essential to check the resulting local minimum by visualizing its trajectories and then simulating them, before thinking of applying them to the real system setup.

Chapter 7

Conclusion

To conclude, the aim of this work was to tackle robot dynamic object-manipulation tasks by relying on a contact-implicit optimization (CIO) approach, that essentially incorporates a hard contact model into the formulation of a direct trajectory optimization scheme. The resulting contact forces were resolved, along with the state and control input trajectories, through the use of a multiple-shooting method, within the *FORCES Pro* framework. More specifically, the state-transition function was given as a result of a time-stepping scheme, which treats the problem of non-smooth dynamic simulation on a velocity-impulse level, thus also allowing for the treatment of impact events with ease. As for the set-valued force laws, resulting from the non-smooth contact dynamics, these were dealt with by the addition of complementarity conditions that needed to be satisfied at every stage. In addition to that, the dynamic feasibility of the optimal trajectories was further enhanced by incorporating Newton's impact law into the CIO formulation. Consequently, we were left with a mathematical program with complementarity constraints (MPCC) that was efficiently solved with a primal-dual interior point algorithm. The obtained control policy is in open-loop form, as it is the outcome of solving a nonlinear optimal control problem once, offline. Hence, a control law that combines the optimal input sequence with a PD-feedback term was utilized instead, in order to stabilize the desired motion plan.

A theoretical motivation has been made for the adoption of this methodology in controlling the hybrid system dynamics governing our applications. First, different modelling techniques and simulation schemes for non-smooth contact dynamics have been presented. The key-points that were addressed in this part reveal the benefits of resorting to a hard contact model and a time-stepping (event-capturing) integration scheme instead of a soft model and an event-driven scheme, respectively. This was done

while also considering the difficulties and downsides that come along such a choice, one of which was having to accept the fact that time-stepping techniques can only yield a local truncation error of $\mathcal{O}(h)$, regardless of the order of the underlying Runge-Kutta scheme. It was also seen that the choice of a hard contact model gives rise to set-valued force laws, rather than smooth ordinary differential equations, due to the unilateral contact constraints. In related works, this matter is generally resolved by solving a linear complementarity problem (LCP) – which was the basis behind the adopted CIO formulation – or through the augmented Lagrangian approach.

Secondly, a survey of the different optimal control methods was made. We leveraged the use of direct methods in trajectory optimization (such as single shooting, direct collocation, and multiple shooting), over dynamic programming or indirect approaches. Therefore, what we end up doing is a discretization of the optimal control problem – which can also be seen as an infinite-dimensional optimization problem – thus transforming it into a finite-dimensional optimization program that was solved with the *FORCES NLP* solver. Eventually, the presented concepts were then extended to accommodate problems involving optimal control of hybrid systems. Accordingly, the multi-phase approach was introduced as the control analog of event-driven simulation techniques – and thus requires an a priori knowledge of the full mode sequence – while the contact-invariant (more generally, mode-invariant) approach as the analog of time-stepping simulation schemes – and thus optimizes through the contacts without the need for a pre-specified contact schedule. We have also shown that it is possible to have several variations of the CIO formulation, but we constrained this work to the implementation of the one described at the beginning of our conclusion.

This method was tested on several manipulation tasks, all involving dynamic pushing as their main contact event. Visualizations of low-dimensional preliminary 2D examples were provided as a result of the optimal trajectories outputted from the nonlinear program, mostly to verify the satisfaction of the contact constraints at every stage, and the boundary conditions (through which the task is defined). Simulations were performed for the higher-dimensional 3D applications involving object manipulation by the

6-DOF ANYpulator robot. The dynamic feasibility of the state trajectories as well as the generated contact forces, was ensured in all scenarios, in addition to verifying the satisfaction of the complementarity conditions and boundary conditions, by the presence of a powerful rigid body dynamics simulation engine and a collision detection engine (Open Dynamics Engine). On the other hand, a nonlinear model predictive control (NMPC) scheme, that does not require any sensory feedback on the contact force or the object states, was explored and implemented to solve the ANYpulator-Ball problem. *Simulink* simulation results were presented that suggest a notable similarity in the contact schedule resulting from solving the nonlinear optimal control problem in open-loop versus in closed loop. However, while NMPC has shown to be more energy efficient than the open-loop scheme, it is still a hypothetical implementation that assumes reasonably higher convergence speeds (for the NLP solver); and therefore cannot be applied in practice so far.

Finally, a contact-implicit optimization program has been formulated, solved, and tested experimentally for the problem in which ANYpulator is required to push a block onto a desired position that is outside of the robot's reach. In this part, we have demonstrated how the Coulomb static friction model was identified and then incorporated into our CIO formulation by adding the friction force as an extra optimization variable and satisfying the frictional set-valued force law with complementarity conditions. The corresponding results displayed satisfactory tracking of joint and end-effector tool reference positions. Furthermore, a force/torque sensor was mounted on the tool to retrieve the normal contact force and compare it to the optimal contact force sequence; these produced results that are indeed compatible. This information, along with the observation of the real-time implementation of the manipulation task, have suggested the richness (in terms of discovering various contact schedules), dynamic feasibility, and repeatability of this approach; while its task-satisfaction accuracy has been shown to be adequate, but increases with the assigned desired displacement of the block. This increased inaccuracy was speculated to be a possible result of the adopted friction model, the unmodelled frictional forces arising in the contact plane, the assumption of a purely inelastic collision, accumulated integration error, the point-plane distance assumption when computing the minimum signed gap function between the tool and the block, and the lat-

eral offset between the tool and the block centerline during a closed contact, that may cause a loss of translational kinetic energy due to the undesirable rotation of the block.

There are several directions for future work and considerations, one of which encompasses the improvement of the current CIO approach by moving away from the typical Runge-Kutta schemes – which when applied within the time-stepping framework, yield a maximum order proportional to the step-size – and exploring higher-order time-stepping techniques instead, in order to improve dynamic feasibility and hence, the method's accuracy. For applications that involve frictional forces arising between the environment and the object (as in the ANYpulator-Block problem), alternative friction models could be chosen, while finding an appropriate balance between modeling accuracy and complexity. Frictional forces arising in the tangential contact plane could also be incorporated into the formulation; in fact, they should be included for manipulation tasks where the contact force direction is no longer dominated by the normal direction. Furthermore, the proposed non-linear model predictive control scheme could be tested in real-time on other application problems that can be solved within the required computational time. In addition to that, a deeper exploration of NMPC approaches could also be made, for instance by relying on a shrinking-horizon concept rather than the already-used receding-horizon; or by solving our contact-implicit optimization problem, online, with an SQP algorithm, while following the steps constituting the *real-time iteration* scheme. On the other hand, it would also be interesting to try out different variations of the contact-invariant method, applied to similar dynamic manipulation tasks, to eventually examine their differences and conclude on which one would turn out to be superior for such applications. Lastly, it is worth mentioning that the approach presented in this thesis work is expected to be less successful when applied to tasks involving highly-underactuated dynamics – such as in locomotion tasks, where also contacts are generally sustained for longer periods of time – because of the difficulty associated with stabilizing the optimal trajectories in these cases. However, it would still be compelling to test it within such domains and improve it accordingly. One could think of implementing the method on more useful applications, such as, for instance, a legged robot (quadruped or biped), that has to use one of its limbs to open a door by pushing it dynamically; one could also move away

from graspless, nonprehensile manipulation and tackle problems involving robotic hands operating with objects through grasping. Those are just a few examples of many other future potential robotic tasks to be accomplished within contact-rich settings.

Bibliography

[1] ANYdrive. `http://www.rsl.ethz.ch/robots-media/actuators/anydrive.html`. Accessed: 21-11-2018.

[2] CasADi. `www.casadi.org`. Accessed: 24-11-2018.

[3] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A Software Framework for Nonlinear Optimization and Optimal Control. *Mathematical Programming Computation*, In Press, 2018.

[4] J. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, second edition, 2010.

[5] Karen Bodie, C Dario Bellicoso, and Marco Hutter. ANYpulator: Design and Control of a Safe Robotic Arm. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1119–1125. IEEE, 2016.

[6] J. Carius, R. Ranftl, V. Koltun, and M. Hutter. Trajectory Optimization With Implicit Hard Contacts. *IEEE Robotics and Automation Letters*, 3(4):3316–3323, Oct 2018.

[7] Jan Carius, Martin Wermelinger, Balasubramanian Rajasekaran, Kai Holtmann, and Marco Hutter. Deployment of an Autonomous Mobile Manipulator at MBZIRC. *Journal of Field Robotics*, 2018.

[8] Nilanjan Chakraborty, Stephen Berard, Srinivas Akella, and Jeff Trinkle. An Implicit Time-Stepping Method for Multibody Systems with Intermittent Contact. In *In Robotics Science and Systems*, 2007.

[9] Richard Cottle, Jong-Shi Pang, and Richard E Stone. *The Linear Complementarity Problem* . Philadelphia Society for Industrial and Applied Mathematics, siam ed., [classics ed.] edition, 2009. Originally published: Boston : Academic Press, c1992. Includes bibliographical references and index.

[10] Hargraves CR and Paris SW. Direct Trajectory Optimization Using Nonlinear Programming and Collocation. *Journal of Guidance, Control and Dynamics*, 10(4):338–342, 1987.

[11] G. De Nicolao, L. Magni, and R. Scattolini. Stability and Robustness of Nonlinear Receding Horizon Control. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, pages 3–22, Basel, 2000. Birkhäuser Basel.

[12] Moritz Diehl. Lecture notes on Optimal Control and Estimation, July 24, 2014.

[13] Moritz Diehl, H Georg Bock, Johannes P Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-Time Optimization and Nonlinear Model Predictive Control of Processes Governed by Differential-Algebraic Equations. *Journal of Process Control*, 12(4):577–585, 2002.

[14] Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

[15] Alexander Domahidi and Juan Jerez. FORCES Professional. embotech GmbH (`http://embotech.com/FORCES-Pro`), July 2014.

[16] Alexander Domahidi, Aldo U Zgraggen, Melanie N Zeilinger, Manfred Morari, and Colin N Jones. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 668–674. IEEE, 2012.

[17] Roy Featherstone. *Robot Dynamics Algorithm*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[18] Marco Hutter. Robot Dynamics Lecture Notes, January 6, 2017.

[19] Jemin Hwangbo, Vassilios Tsounis, Hendrik Kolvenbach, and Marco Hutter. Cable-driven Actuation for Highly Dynamic Robotic Systems. *arXiv preprint arXiv:1806.10632*, 2018.

[20] Matthew Kelly. An Introduction to Trajectory Optimization: How To Do Your Own Direct Collocation. *SIAM Review*, 59(4):849–904, 2017.

[21] Matthew P Kelly. Transcription Methods for Trajectory Optimization. *Tutorial, Cornell University, Feb*, 2015.

[22] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[23] Nathan P Koenig and Andrew Howard. Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator.

[24] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-Based Autonomous Racing of 1:43 Scale RC Cars. *Optimal Control Applications and Methods*, 36(5):628–647.

[25] Y. Maeda, T. Nakamura, and T. Arai. Motion Planning of Robot Fingertips for Graspless Manipulation. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pages 2951–2956 Vol.3, April 2004.

[26] L. Magni and R. Scattolini. *Advanced and Multivariable Control*. Pitagora, 2014.

[27] Zachary Manchester and Scott Kuindersma. Variational Contact-Implicit Trajectory Optimization. In *International Symposium on Robotics Research (ISRR)*, Puerto Varas, Chile, 2017.

[28] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant Optimization for Hand Manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 137–144, Goslar Germany, Germany, 2012. Eurographics Association.

[29] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of Complex Behaviors Through Contact-invariant Optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, July 2012.

[30] Igor Mordatch, Jack M. Wang, Emanuel Todorov, and Vladlen Koltun. Animating Human Lower Limbs Using Contact-invariant Optimization. *ACM Trans. Graph.*, 32(6):203:1–203:8, November 2013.

[31] Panagiotopoulos P.D. (eds) Moreau J.J. Unilateral Contact and Dry Friction in Finite Freedom Dynamics, International Centre for Me-

chanical Sciences (Courses and Lectures), Springer, Vienna. *Nonsmooth Mechanics and Applications*, 302, 1988.

[32] Neunert, Michael and Stäuble, Markus and Giftthaler, Markus and Bellicoso, Dario and Carius, Jan and Gehring, Christian and Hutter, Marco and Buchli, Jonas. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters*, PP, 12 2017.

[33] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[34] H. Olsson, K.J. Ãström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky. Friction Models and Friction Compensation. *European Journal of Control*, 4(3):176 – 195, 1998.

[35] Diego Pardo, Michael Neunert, Alexander Winkler, Ruben Grandia, and Jonas Buchli. Hybrid Direct Collocation and Control in the Constraint-Consistent Subspace for Dynamic Legged Robot Locomotion. July 2017.

[36] Michael A. Patterson and Anil V. Rao. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Trans. Math. Softw.*, 41(1):1:1–1:37, October 2014.

[37] Friedrich Pfeiffer and Ch Glocker. Glocker, C.: Multibody Dynamics with Unilateral Contacts. Wiley, New York. 01 1996.

[38] Michael Posa. *Optimization for Control and Planning of Multi-Contact Dynamic Motion*. PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2017.

[39] Michael Posa, Cecilia Cantu, and Russ Tedrake. A Direct Method for Trajectory Optimization of Rigid Bodies through Contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[40] Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and Stabilization of Trajectories for Constrained Dynamical Systems. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016. IEEE, IEEE.

[41] R. Tyrrell Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970.

[42] Amir Shahzad and Paul J. Goulart. A New Hot-Start Interior-Point Method for Model Predictive Control. *IFAC Proceedings Volumes*, 44(1):2470 – 2475, 2011. 18th IFAC World Congress.

[43] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, 2010.

[44] Georgios Stavroulakis. Multibody Dynamics with Unilateral Contacts by Friedrich Pfeiffer and Christoph Glocker, Wiley, New York, 1996. *Shock and Vibration*, 5:371–372, 01 1998.

[45] D. Stewart and J. C. Trinkle. An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 162–169 vol.1, April 2000.

[46] W.J. Stronge, R James, and Bahram Ravani. Oblique Impact with Friction and Tangential Compliance. *Philosophical Transactions of The Royal Society A: Mathematical, Physical and Engineering Sciences*, 359:2447–2465, 12 2001.

[47] Christian Studer and Christoph Glocker. Solving Normal Cone Inclusion Problems in Contact Mechanics by Iterative Methods. *Journal of System Design and Dynamics*, 1(3):458–467, 2007.

[48] Yuval Tassa and Emanuel Todorov. Stochastic Complementarity for Local Control of Discontinuous Dynamics. In *Robotics: Science and Systems*, 2010.

[49] Russell Tedrake. 6.832 Underactuated Robotics, Spring 2009.

[50] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, and et al. Feedback Control of Dynamic Bipedal Robot Locomotion, 2007.

[51] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. FORCES NLP: An Efficient Implementation of Interior-Point Methods for Multistage

Nonlinear Nonconvex Programs. *International Journal of Control*, 0(0):1–17, 2017.

[52] Andrea Zanelli, Rien Quirynen, Juan Jerez, and Moritz Diehl. A Homotopy-based Nonlinear Interior-Point Method for NMPC. *IFAC-PapersOnLine*, 50:13188–13193, 07 2017.