# Planning, Sensing, and Control for Contact-rich Robotic Manipulation with Quasi-static Contact Models 

by
Tao Pang
Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2023
(C) Massachusetts Institute of Technology 2023. All rights reserved.

Author $\qquad$
Department of Mechanical Engineering Feb 7, 2023

Certified by .
Russ Tedrake
Professor of Electrical Engineering and Computer Science Thesis Supervisor

Accepted by $\qquad$
Nicolas Hadjiconstantinou
Department Graduate Officer

# Planning, Sensing, and Control for Contact-rich Robotic <br> Manipulation with Quasi-static Contact Models 

by<br>Tao Pang<br>Submitted to the Department of Mechanical Engineering on Feb 7, 2023, in partial fulfillment of the<br>requirements for the degree of<br>Doctor of Philosophy


#### Abstract

Human-level manipulation is not limited to the finger tips. We use our palms, arms, body and legs to make and break contact with the manipuland whenever and wherever necessary. In contrast, the vast majority of robotic manipulators are artificially divided into a small end effector that makes all the interactions, and a hefty arm that needs to always remain collision-free.

One big reason behind this unsatisfying limitation on robotic manipulators is the lack of planning algorithms that can efficiently reason about contact-rich interactions. Navigating the non-smooth landscape of contact dynamics constraints is perhaps the biggest challenge faced by model-based contact-rich planners. Existing methods either descend along the gradients of smoothed contact dynamics and get stuck in local minima, or search globally through contact mode transitions and get overwhelmed by the exponentially many modes as the problem gets more complex. Drawing lessons from the recent empirical success of reinforcement learning in contact-rich manipulation, we hypothesize that smoothing and global search are both necessary for contact-rich planners to succeed. In this thesis, we propose a model-based contact-rich planner which searches globally under the guidance of a novel contact dynamics model. The model is quasi-static, convex, differentiable and amenable to smoothing, all of which are features designed to improve the planner's efficiency. Our method can generate complex dexterous manipulation action plans with less than 1 minute of online computation on a regular desktop computer. The plans can also transfer directly to robotic hardware under certain conditions.

In addition to planning, a complete manipulation pipeline also needs contact force sensing and feedback control. Therefore, this thesis also studies the viability of estimating external contact forces from only joint torque measurements, and explores how to use feedback to keep the magnitude of contact forces bounded in an accidental collision.


Thesis Supervisor: Russ Tedrake
Title: Professor of Electrical Engineering and Computer Science

## Acknowledgments

I'd like to begin by thanking my advisor Russ. I don't think a few sentences that need to fit within half a page can describe how awesome Russ is, but I'll give it a try with some machine learning metaphors :D Russ has been an oracular reward function as I navigate the fun but treacherous landscape in the (state) space of robotics research. The hardest part of a reward function is to balance between exploration and exploitation. Russ's reward has been sparse enough that I always get to do what I'm most excited about, but also dense enough that I eventually get to put together this thesis. In addition, Russ's rigorous attitude towards research, teaching, coding, diet and bare-foot running has also made him an expert policy to imitate, not only in work, but also in life.

I'd also like to thank my committee members, Tomas and Alberto, for their valuable feedback on both research and presentation, and for pointing out the blind spots in my literature review.

I'd like to thank members of the robot locomotion group. I've received tremendous help over the years on coding, writing, refining research ideas, learning about related work, and finding good papers. I will miss the academic and non-academic discussions, the lunches and dinners, and the beers. Special thanks to my co-author Terry, who, in addition to being an awesome collaborator, kept our happiness and curiosity alive on a journey that involves banging our heads against the wall on a daily basis.

Lastly, I'd like to thank my parents, parents-in-law, and Di, my lovely wife, for their unwavering love and support during the incredible but tough journey of my PhD. Whenever I feel like I'm a RL agent stuck with a useless policy, they're there to cheer me up and remind me that there is more to life than collecting rewards.

## Contents

1 Introduction ..... 23
1.1 Difficulty of Pushing a Box with a Ball ..... 23
1.2 Related Work ..... 27
1.2.1 Modeling Contact with Complementarity Constraints ..... 28
1.2.2 Nonlinear Optimization ..... 29
1.2.3 Contact Mode Transitions ..... 30
1.2.4 Reinforcement Learning ..... 32
1.3 Contributions and Thesis Structure ..... 32
2 An Effective Contact Dynamics Model for Contact-rich Planning ..... 35
2.1 Introduction ..... 35
2.2 Background ..... 37
2.2.1 Quasi-static Models in Robotic Manipulation ..... 37
2.2.2 Convex Rigid-body Contact Dynamics ..... 39
2.3 A Convex, Quasi-static and Differentiable Contact Model ..... 40
2.3.1 Forward Dynamics ..... 41
2.3.2 Differentiability ..... 44
2.3.3 Implementation ..... 45
2.3.4 Interpreting Anitescu's Convex Relaxation ..... 46
2.3.5 Flaws of Prescribed Robot Motions as Control Input ..... 49
2.3.6 Experimental Evaluation ..... 52
2.4 Smoothing of Contact Dynamics ..... 62
2.4.1 Smoothing Schemes for Dynamical Systems ..... 62
2.4.2 Randomized Smoothing of Contact Dynamics ..... 64
2.4.3 A Smoothed Contact Dynamics Model ..... 66
2.4.4 The Smooth Contact Model as "Analytic Smoothing" ..... 68
3 Contact-Rich Manipulation Planning using Quasi-static Models ..... 73
3.1 Introduction ..... 73
3.2 Trajectory Optimization through Contact ..... 74
3.2.1 Iterative MPC with Smoothing ..... 75
3.2.2 Experiment Setup ..... 76
3.2.3 Results \& Discussion ..... 79
3.3 Local Mahalanobis Metric for RRT ..... 80
3.3.1 The Local Mahalanobis Metric ..... 81
3.3.2 Metric on Smoothed Dynamics and Unactuated Objects ..... 83
3.4 RRT through Contact ..... 86
3.4.1 Nearest Node using Local Mahalanobis Metric ..... 86
3.4.2 Dynamically Consistent Extension ..... 87
3.4.3 Contact Sampling ..... 88
3.4.4 Effectiveness of Proposed Enhancements ..... 89
3.4.5 Final Path Refinement ..... 90
3.5 Results \& Discussion ..... 91
3.5.1 Experiment Setup ..... 92
3.5.2 Results \& Discussion ..... 93
3.6 Sim2Real Transfer \& Hardware Results ..... 96
3.6.1 Experiment Setup ..... 97
3.6.2 Results \& Discussion ..... 98
4 External Contact Detection from Joint Torque Measurements ..... 103
4.1 Introduction ..... 103
4.2 Related Work ..... 105
4.3 Problem Formulation ..... 106
4.4 Detectability of contacts ..... 108
4.4.1 Detectability ..... 109
4.4.2 Quantifying "undetectability" ..... 111
4.4.3 Implications ..... 113
4.5 Contact estimation from external torque measurements ..... 113
4.5.1 Rejection sampling ..... 114
4.5.2 Rejection sampling + gradient descent (RSGD) ..... 114
4.6 Active contact discrimination ..... 117
4.7 Implementation ..... 119
4.8 Conclusion ..... 121
5 Contact-aware Control with Quasi-static Models ..... 123
5.1 Introduction ..... 123
5.2 Related Work ..... 125
5.2.1 Interaction Control ..... 125
5.2.2 Null-space Projection ..... 126
5.3 Background and Notations ..... 127
5.3.1 Constrained Inverse Dynamics Control ..... 127
5.3.2 Contact and Multibody Notations ..... 127
5.4 Quasi-static Dynamics ..... 128
5.4.1 Dynamics as Transitions between Equilibria ..... 128
5.4.2 Relationship to Null-space Projection ..... 130
5.5 QP Controller with Quasi-static Dynamics ..... 133
5.5.1 Frictionless Contacts ..... 133
5.5.2 Frictional Contacts ..... 134
5.6 Experiments ..... 137
5.6.1 Mug Placement Task (Fig. 5-4a) ..... 138
5.6.2 Mug Moving Task (Fig. 5-4b) ..... 140
5.7 Conclusion ..... 140
6 Conclusion ..... 143

## List of Figures

1-1 (a-b) An actuated, PD-controlled ball and an un-actuated box, both are constrained to slide along a rail. (a): Ball on the left of the box. (b): Ball on the right of the box. (c-d) Dynamics of the ball-box system when the ball is on the (c) left or (d) right side of the box. Red and green indicate different contact modes. The maroon dashed lines are the smoothed contact dynamics after introducing the smoothing force field.

1-2 The Allegro Hand and an oriental melon [1].

2-1 Anitescu's friction constraints (2.12) under different contact modes: (a) sticking, (b) sliding, and (c) separation. The normal and tangent of the contact frame $i$ are denoted respectively by $\mathrm{n}_{i}$ and $\mathrm{t}_{i}$. The green shaded area is the friction cone. The purple shaded area is the feasible region of $\delta \mathrm{x}_{i}$. Constraints corresponding to (2.12a) and (2.12b) are color-coded blue and red, respectively.

2-2 (a) A planar quasi-static multibody system. The red rectangles are the actuated gripper fingers, $q^{\mathrm{a}}=\left[x_{l}, x_{r}, y\right]$ and $u=\left[u_{l}, u_{r}, u_{y}\right]$. The gray sphere is the un-actuated manipuland, $q^{\mathrm{u}}=\left[x_{c}, y_{c}\right] . r=0.1 \mathrm{~m}$. The normal contact forces associated with contact pair 1 and 2 are denoted respectively by $\lambda_{\mathrm{n}_{1}}$ and $\lambda_{\mathrm{n}_{2}}$. (b) A grasping command violating the non-penetration constraint.

2-3 Free-body diagrams of the left finger when it is (a) away from the sphere, and (b) in contact with the sphere.

2-4 Bond graphs of a 1-DOF robot pushing against the wall. In (a), the robot is modeled as admittance (a prescribed motion). In (b), the robot is modeled as impedance.

2-5 Commanded and actual left finger positions for the system defined in Fig. 2-2, as simulated by LCP and the proposed CQDC dynamics (2.10). Recall that $x_{l}$ and $u_{l}$ are respectively the commanded and actual $x$-coordinate of the left finger; $y_{l}$ and $u_{y}$ are respectively the commanded and actual $y$-coordinate of both fingers. The right finger is not shown, as the motions and forces of the left and right fingers are symmetric about the $y$-axis. The green dashed vertical lines indicate contact mode changes: separation to sticking at $t=0.03 \mathrm{~s}$; sticking to sliding at $t=0.13 \mathrm{~s} ;$ sliding to sticking at $t=0.23 \mathrm{~s}$.

2-6 Contact force and signed distance at contact 1 and 3 of the grippersphere system defined in Fig. 2-2.

2-7 Starting (a) and ending (b) configurations of the reference joint-angle trajectory $q_{\text {ref }}(\cdot)$.

2-8 Mean tracking error $\Delta\left(q_{\mathrm{CQDC} / \mathrm{SAP}}, q_{\mathrm{ref}}\right)$ for trajectories simulated using the CQDC dynamics or SAP at different simulation step sizes $h$. . . 57

2-9 KUKA IIWA robot stacking cubes. The red-and-grey cube has a quadrant colored grey to indicate its orientation. The red-and-grey cube starts off on the ground (a) and is placed on a stack of cubes and rotated by 90 degrees (b).

2-10 Angular and translational mean error of the pose of the red-and-grey box, $\Delta\left(q_{\mathrm{CQDC} / \mathrm{SAP}}^{\mathrm{u}}, q_{\mathrm{GT}}^{\mathrm{u}}\right)$, at different step sizes $h$. As shown in Fig. 2-9, the box is picked up, rotated, and placed on the stack of cubes. The ground truth trajectory, $q_{\mathrm{GT}}^{\mathrm{u}}$, is defined as the trajectory generated by SAP with $h=5 \times 10^{-5}$ S. In the computation of the mean error $\Delta(\cdot, \cdot)$, the distance function $d(\cdot, \cdot)$ is the Euclidean 2-norm for translation, and the absolute difference in angle for rotation.

2-11 Final configurations of the box stacking task shown in Fig. 2-9, simulated using SAP (a-c) and CQDC (d-f).

2-12 Trajectories of the red cube in Fig. 2-9, including the ground truth (GT) and the ones generated using the CQDC dynamics with different step sizes $h . x, y$ and $z$ are the cube's center of mass coordinates in world frame. Angle comes from the axis-angle representation of the cube's orientation relative to world frame. Here the angle plot represents a rotation about the world $y$-axis by 90 degrees. The larger $y$ error at $h=0.5 \mathrm{~s}$ is caused by the gripper clipping the corner of the red-and-grey box when the gripper retreats from the box after placing it on the top of the stack. Due to the large $h$, the gripper's motion is interpolated between an upward segment and a horizontal segment. This error is visible by the difference of a few pixels between Fig. 2-11e and Fig. 2-11f.

2-13 Figure for Example 1, where quasi-dynamics of motion is interpreted as a projection operator. (a) Illustration of the system. (b) Distribution of $q+w$ (green) and $f(q+w)$ (pink). Note that the samples for which $q+w_{i}<0$ have been projected onto the surface into a delta function, and the expectation of the pink distribution lies on the right side of $q$, creating a stochastic force field effect. (c) CQDC dynamics and its randomized smoothed version. (d) Gradients of the CQDC dynamics obtained with first-order randomized smoothing.

2-14 (a) A system consisting of an actuated cart constrained to slide on a frictionless surface, and a wall occupying $q^{a} \leq 0$. The actuator has stiffness $k_{\mathrm{a}}$. (b) A system consisting of an un-actuated cart constrained to slide on a frictionless surface, and a ball actuated along both the $x$ and $y$ axes. The ball can touch the top surface of the cart with a frictional contact. (c) Randomized and analytic smoothing of (a). Randomized smoothing, shown in green, is done with a Gaussian kernel with different variances $\sigma$. Analytic smoothing, shown in magenta, is done with different log-barrier weights $\kappa$. (d) Density functions of the Gaussian kernels (green) and the elliptical distributions used for analytic smoothing (magenta). (e) Randomized and analytic smoothing of (b). We plot $q_{+}^{\mathrm{u}}$ against $u_{x}$ for a fixed $u_{y}$ that is inside the cart. The linear region in the plot corresponds to sticking contact, and the flat regions to sliding. . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

3-1 Performance of iMPC with different smoothing schemes: analytic, randomized (first-order), randomized zero-order, and exact (no smoothing). For each method, the solid line represents the mean over five runs, and the shaded region represents the standard deviation.

3-3 (a) Two different sublevel sets $\mathcal{R}_{\rho, \varepsilon, \gamma}^{u}$, represented as ellipsoids, shown in the space of $q^{u}$, with $\varepsilon=1$, and $\gamma=10^{-6}$. The ellipsoid centers are shifted to the origin for easy comparison. Red ellipsoid: $\mathcal{R}_{\rho, \varepsilon, \gamma}^{u}$ for the system configuration in Fig. 3-3b; blue ellipsoid: $\mathcal{R}_{\rho, \varepsilon, \gamma}^{u}$ for the configuration in Fig. 3-3c. Points $b_{1}, c_{1}$ are where ellipsoids' major axes intersect their boundaries. Points $b_{2}, c_{2}$ are points along the minor axes of the ellipsoids, and satisfy $\left\|b_{1}\right\|=\left\|b_{2}\right\|$ and $\left\|c_{1}\right\|=\left\|c_{2}\right\|$, where the norm is based on the standard Euclidean metric. (b) The solid robots and objects represent the $\bar{q}$ at which the $\operatorname{red} \mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$ in (a) is computed. The straight line on the puck indicates its orientation. The dashed dark red puck corresponds to the configuration $b_{1}$, and pink to $b_{2}$. Note that $b_{1}$ is easier to each than $b_{2}$. (c) Similar to Fig. 3-3b, the dashed dark blue puck correspond to $c_{1}$, and light blue to $c_{2}$. It is also easier to reach $c_{1}$ than $c_{2}$. (d) The volume of $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$ shrinks as the fingers get further away from the puck. The ellipsoids on the right are color-coded to match the robot configurations on the left. Note that the blue ellipsoid is barely visible. . . . . . . . . . . . . . . . . . . . .

3-4 (a-d) RRT trees, shown in the space of $q^{\mathrm{u}}$, at different iterations of a complete run of the enhanced RRT for the PlanarHandFixedY system. The contours are the sub-level sets of the local Mahalanobis metric of the nodes. The path from start $\left(q_{\text {init }}\right)$ to goal $\left(q_{\text {goal }}\right)$ is highlighted in red in the final tree of (d). (e-h) Visualization of RRT trees with the same number of nodes (50) but grown with different methods. (e) Tree grown with our algorithm; (f) without contact sampling; (g) using a globally uniform weighted Euclidean metric; (h) using exact gradients without smoothing. Note that our method achieves the best coverage of the space of $q^{\mathrm{u}}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90

3-5 Tasks for RRT. Similar to Fig. 3-2b, the thicker frame denotes the goal, and the thinner frame the initial configuration of the object.

3-6 Planning performance for the tasks in Fig. 3-5. Results include running RRT with the enhancements proposed in Sec. 3.4 using the three smoothing schemes from Sec. 2.4.1, as well as the three ablation studies
proposed in Sec. 3.5.1.

3-7 Hardware for the liwaBimanual setup, where the goal is to rotate the bucket by $180^{\circ}$. The left and right pictures correspond to the initial state and the final state after the open-loop plan execution. The lines between motion capture markers are connected to illustrate the change of pose in the bucket. Readers are encouraged to watch the accompanying video for the full execution.

3-8 Plots for sim2real performance of our CQDC dynamics, evaluated on the plans of Sec.3.5. First Two Columns: Scatter plot of mean error $\Delta$ vs. path length for position (first column) and orientation (second column). Each dot in the plot represents one segment trajectory. Last Two Columns: Box plot for the normalized error $\bar{\Delta}$ for positions (third column) and orientation (fourth column). Note that the mean in orange corresponds to the slope of the graph in the first two columns. Finally, we note that the AllegroHandDoor (AHD) example only consists of orientation, and has no position plot. Readers are highly encouraged to watch the accompanying video for the qualitative behavior of the actual segment trajectories in this plot

4-1 For the true contact shown in (a), which generates little torque about joint 6 and 7, the two contacts in (b) and (c), represented by red cylinders, create almost identical torque measurements as (a).

4-2 A contact force on Link 6 of the KUKA IIWA robot. The orange mesh is the surface of the link. The triad represents the body frame of the link. (a) second-order friction cone. (b) Polyhedral approximation of the friction cone with $n_{d}=4$.

4-3 The residual $l(\cdot)$ for 20000 sampled points on link 5 and 6 of the IIWA arm. The true contact position and direction is indicated by the red line. Multiple global minima of $l(\cdot)$ can be found on the robot's surface $\mathcal{S}$.

4-4 Full detectability of (a) link 6 of the IIWA arm and (b) "forearm" link of the UR5 arm. Out of the 20000 samples generated on each link, $2.19 \%$ and $32.58 \%$ of the samples on (a) and (b) are fully detectable, respectively. Fully detectable samples are shown in cyan, others in black. The meshes of the links are shown in (c) and (d). Points $A, B$ and $D$ are examples of different levels of detectability, which will be defined in Sec. 4.4.2.

4-5 $\quad \eta_{\epsilon}(\cdot)$ of IIWA link 6 and UR 5 forearm.

4-6 Finding elements of $P_{\epsilon}$ on link 5 and link 6 with $\epsilon=0.005$ using rejection sampling. A force of 10 N is applied to the robot along the green line. Enclosed by magenta boxes, the colored squares represent accepted samples. (a) Dense P: 491 out of 20000 samples are accepted. (b) Sparse P: 24 out of 1000 samples are accepted.

4-7 Running RSGD on the same robot and contact configuration as Fig. 4-6. Step 1: 175 out of 1000 samples are accepted with $\delta=0.1$, which are shown as small squares color-coded by their residual values $l$. Step 2: 155 of the 175 samples in $P_{\delta}$ converge after running Algorithm 1, which are shown as large squares. Every $p_{C} \in P_{\delta}$ is connected by a line to the corresponding element in $P_{\delta}^{\star}$ to which it converges.

4-8 (a) Distribution of residual $l(\cdot)$ of the $P_{\delta}$ and $P_{\delta}^{*}$ from Fig. 4-7. (b) A gradient descent run on link 6 of IIWA. Algorithm 4 starts at $p_{C} \in \mathcal{S}$ and converges to $p_{\bar{C}^{\star}}$. Red lines represent the path taken by gradient descent. White translucent disks represent local tangent planes.

4-9 (a) Two possible contact positions to disambiguate. The centers of the small red spheres are coincident with the candidate contact positions. (b) Solving (4.13) finds a motion that pulls away from the contact on link 6 and pushes into the contact on link 5.

4-10 Run-time breakdown of a typical iteration of RSGD.

5-1 As transparent obstacles are almost invisible to depth sensors, a collisionfree motion planner, with the goal to pick up the mug and unaware of the transparent tray, plans a trajectory that crushes the egg along the way. The crushed egg is highlighted in the red box. Our controller is able to keep the egg intact even when the reference trajectory would crush it.

5-2 Quasi-static dynamics of a 2D, 2-link robot arm. The arm starts at $q^{l}$ (black) and is commanded to go to $q_{\mathrm{cmd}}^{l+1}$ (red). The virtual spring connecting $q_{\mathrm{cmd}}^{l+1}$ to $q^{l}$ pulls the robot towards $q_{\mathrm{cmd}}^{l+1}$, but the robot eventually stabilizes to $q^{l+1}$ (green) due to contact constraints. At $l+1$, the arm makes two contact with two obstacles at $C_{1}$ and $C_{2}$ with contact forces $\lambda_{1}$ and $\lambda_{2}$

5-3 (a): The reference trajectory brings the robot into contact at $C$. Due to friction, the contact normal $n$ and the contact force direction $u$ are different. Therefore, the commanded velocity at $C$ can separate from the obstacle even when the angle between $u$ and ${ }^{W} v_{\mathrm{cmd}}^{C}$ is greater than $\pi / 2$. (b): Definitions of $\Delta q_{\mathrm{ref}}, \Delta q_{\mathrm{cmd}}$ and $\Delta q$. The range and null space of the projection $\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q+}} \mathbf{J}_{u}$ are $N\left(\mathbf{J}_{u}\right)$ and $R\left(\mathbf{J}_{u}^{\mathbf{K}_{q}+}\right)$, respectively. .

5-4 (a): mug placement task. (b): mug moving task. Contacts are highlighted in red boxes. Blue arrows denote the direction of end effector velocity. For both tasks, the photographs on the left show where the real robot makes contact while executing the task. With collision geometry disabled, the simulation frames on the right show how much penetration would happen if the original trajectory were strictly followed. Videos of the real robot executing the tasks are included in the attachment.

5-5 mug placement task. $x, y, z$ components and the norm of the contact force $f_{W}^{C}$. Top: the baseline controller without contact force upper bounds is shown in the top plot. Bottom: the contact-aware controller (5.24) with a modified end effector tracking objective. In both plots, the horizontal red dashed line represents $f_{\text {threshold }}$ and the green dashed line $\lambda_{\text {max }}$.

5-6 mug placement task. Left: position tracking error. Right: orientation tracking error.

5-7 Comparison between our QP controller and null-space projection-based control in the mug moving task. Top: contact force norm $\left\|f_{W}^{C}\right\|$. The red dashed line represents $f_{\text {threshold }}$ and the green dashed line $\lambda_{\max }$. Both controllers can keep $\left\|f_{W}^{C}\right\|$ near $\lambda_{\max }$ when the robot is in contact. Bottom: robot joint velocity norm $\left\|v_{q}\right\|$ during contact separation (from $t=10 s$ to $t=14 s$ ). Note the velocity spike in the null-space projection controller. In both plots, the solid lines are the mean of 10 runs; the shaded regions around the lines represent the maximum and minimum values of all runs.

## List of Tables

3.1 Minimum cost and running time achieved by different methods. All methods are ran for 10 iterations across 5 trials. The methods names are abbreviations from the legend of Fig. 3-1.
3.2 Running time achieved by different methods in seconds. Every trial was run for 1000 iterations. We choose to not display running time for the ablation options since they are slight variations of Analytic with comparable running times

## Chapter 1

## Introduction

In 1988, Salisbury et al. from the MIT AI lab envisioned contact-rich robotic manipulators that "employ all the available manipulation surfaces of the robot to act upon and sense the environment", in a way similar to how humans interact with the environment using their limbs and torso [2]. This was in stark contrast with the prevailing practice in robotic manipulation back then, where environmental interaction is limited "at the hand of the arm", while the arm itself needs to carefully avoid contacts. Unfortunately, real-world deployment of contact-rich robotic manipulators still remains elusive even after more than 30 years. In most contemporary commercial applications of robotic manipulation, such as in warehouses or on factory floors, the robotic manipulator is still divided into the touching hand and the collisionavoiding arm [3]. The dichotomy between the hand and the arm is also prominent in robotics research: although increasingly effective algorithms for grasping objects with the hand $[4, \S 17]$ and avoiding obstacles with the arm $[5,6]$ have been developed over the decades, a reliable, generalizable and interpretable solution to human-like contact-rich manipulation still remains to be found.

### 1.1 Difficulty of Pushing a Box with a Ball

The difficulty of contact-rich manipulation planning is rooted in the non-smooth nature of contact dynamics. In this section, we hope to illustrate this difficulty


Figure 1-1: (a-b) An actuated, PD-controlled ball and an un-actuated box, both are constrained to slide along a rail. (a): Ball on the left of the box. (b): Ball on the right of the box. (c-d) Dynamics of the ball-box system when the ball is on the (c) left or (d) right side of the box. Red and green indicate different contact modes. The maroon dashed lines are the smoothed contact dynamics after introducing the smoothing force field.
using a simple task that involves just one contact: pushing a box with a ball in 1D (1-Dimension). As shown in Fig. 1-1, the system consists of an un-actuated box which slides along a rail with sufficient damping, and an actuated ball controlled by a Proportional-Derivative (PD) controller. The position of the box is denoted by $q^{\mathrm{u}} \in \mathbb{R}$. The actual and commanded positions of the ball are denoted respectively by $q^{\mathrm{a}} \in \mathbb{R}$ and $u \in \mathbb{R}$. The ball can push on the left (Fig. 1-1a) or right (Fig. 1-1b) surface of the box.

Starting from the system configuration in Fig. 1-1a, let us consider the task of pushing the box to a goal configuration $q_{\text {goal }}^{\mathrm{u}}$ on the right of the box $\left(q_{\text {goal }}^{\mathrm{u}}>q^{\mathrm{u}}\right)$. To find an action $u$ that accomplishes the task, we can formulate an optimal control problem that penalizes the distance between the goal and the box position after the action $u$ is taken:

$$
\begin{equation*}
\underset{u}{\operatorname{minimize}} \frac{1}{2}\left(q_{+}^{\mathrm{u}}-q_{\text {goal }}^{\mathrm{u}}\right)^{2}, \tag{1.1}
\end{equation*}
$$

where $q_{+}^{\mathrm{u}}$ is the steady-state position of the box after the ball is commanded to position $u$. Note that $q_{+}^{\mathrm{u}}$ depends on $q^{\mathrm{u}}, q^{\mathrm{a}}$ and $u$. For instance, if the ball starts on the left of the box ( $q^{\mathrm{a}}<q^{\mathrm{u}}$, Fig. 1-1a), any $u$ to the left of the box ( $u<q^{\mathrm{u}}$ ) will not change the box's position, as the ball will not touch the box. In contrast, the box will follow the ball to the commanded $u$ when the ball makes contact with the box $\left(u \geq q^{\mathrm{u}}\right)$. The dependence of $q_{+}^{\mathrm{u}}$ on $\left(q^{\mathrm{u}}, q^{\mathrm{a}}, u\right)$ is commonly called the dynamics of the system, and is plotted in Fig. 1-1c-d.

Gradient descent is a common way to minimize the cost (1.1). Starting with an initial guess of $\left(q^{\mathrm{u}}, q^{\mathrm{a}}, u\right)$ (without loss of generality, we assume the ball is on the left of the box in the initial guess), gradient descent iteratively improves $u$ by taking steps opposite to the cost's gradient w.r.t. $u$ :

$$
\begin{equation*}
\frac{\partial}{\partial u}\left(\frac{1}{2}\left(q_{+}^{\mathrm{u}}-q_{\mathrm{goal}}^{\mathrm{u}}\right)^{2}\right)=\left(q_{+}^{\mathrm{u}}-q_{\mathrm{goal}}^{\mathrm{u}}\right) \frac{\partial q_{+}^{\mathrm{u}}}{\partial u} \tag{1.2}
\end{equation*}
$$

However, $\frac{\partial q_{+}^{u}}{\partial u}$ is 0 unless the ball is touching the box, as shown in Fig. 1-1c-d. This means that if the box and ball are not in contact in the initial guess, the gradient $\frac{\partial q_{+}^{\mathrm{u}}}{\partial u}$ tells us nothing about how $u$ can be improved in order to get $q_{+}^{\mathrm{u}}$ closer to the goal $q_{\text {goal }}^{\mathrm{u}}$.

To combat the zero gradient problem, we can smooth the contact dynamics by introducing a force field between the box and the ball: the ball can push the box without touching it, and the magnitude of the force is inversely proportional to the distance between them. This force field will make $\frac{\partial q_{+}^{u}}{\partial u}$ positive even when the box and ball are not touching (the dashed lines in Fig. 1-1c-d), thereby informing gradient descent the direction in which $u$ needs to be improved. To make sure that the actual non-smooth dynamics is satisfied at convergence, the amount of smoothing can be iteratively reduced during the gradient steps.

However, smoothing is not the panacea in contact-rich planning. For example, what if we want to push the box to the left ( $q_{\text {goal }}^{\mathrm{u}}<q^{\mathrm{u}}$ ), with an initial guess where the ball is also on the left of the box (Fig. 1-1a)? Following the gradient would move the ball further and further away from the box until $\frac{\partial q_{+}^{u}}{\partial u}$ becomes so small that
gradient descent converges. This is an example of gradient descent getting stuck in a bad local minimum. Bad local minima can be avoided to some extent with better initial guesses: in the box-ball system, starting gradient descent with the ball on the right of the box (as in Fig. 1-1b) would yield the correct $u$. However, for more complex problems, good initial guesses are rarely handed to us on a silver platter [7].

To avoid bad local minima without good initial guesses, we need a more global search strategy that explores beyond the gradient (which greedily decreases the cost). A common way to organize the search is to divide the system dynamics into modes based on which contacts are active. For the box-ball system, there are four modes: (i) left contact, (ii) left no contact, (iii) right contact and (iv) right no contact. Each mode corresponds to a linear piece in the dynamics (Fig. 1-1c-d), which is easy to analyze. In this case, the search for the optimal action $u$ can be done by solving (1.1) for every mode, and then picking the action with the lowest cost among all modes.

But the number of modes can grow out of control quickly. For a system with $n_{\mathrm{c}}$ rigid bodies, there are $\binom{n_{c}}{2}$ pairs of bodies, or contact pairs, that can be in contact. Each contact pair can be in separation or in contact. If there is friction, "in contact" can be further divided into sticking, sliding left and sliding right, with even more sliding directions if the system is 3D. Other factors, such as the number of faces in a body, can also contribute the the number of modes. Although the four modes for the ball-box system seems manageable, the number of modes can be astronomical for more complex systems. For instance, the system in Fig. 1-2, consisting of a dexterous hand and a rigid object, has $6^{\binom{20}{2}}=6^{190}$ contact modes ${ }^{1}$. Although this number can be reduced into the trillions with a more efficient counting scheme such as [8], the reduced number of modes is still too much to handle for algorithms that reason explicitly about modes.

[^0]

Figure 1-2: The Allegro Hand and an oriental melon [1].

### 1.2 Related Work

In Section 1.1, we formulated a simple pushing task (1.1) as an optimal control problem $[9, \S 10]$, and identified the non-smoothness in contact dynamics constraints as a key source of trouble in solving the problem. We have also illustrated the two major categories of model-based solutions towards the non-smoothness and their drawbacks: (i) nonlinear optimization using smoothed gradients of contact dynamics, which frequently get stuck in local minima, and (ii) considering contact mode transitions explicitly, which scales poorly due to the exponential explosion of the number of modes.

In this subsection, after a brief review of how contacts are modeled numerically and how such modeling leads to non-smoothness, we will give a more detailed account on model-based contact-rich planning/control algorithms based on (i) nonlinear optimization and (ii) sequencing contact mode transitions. We will also discuss what model-based methods can learn from the success of applying RL to contact-rich manipulation, which has created the most impressive contact-rich manipulation demos (so far) on robotic hardware.

### 1.2.1 Modeling Contact with Complementarity Constraints

For a pair of rigid bodies, the contact force between them and the distance between them need to satisfy some intuitive constraints: (i) the contact force cannot push the bodies apart, (ii) the distance between the two bodies need to be non-negative, and (iii) non-zero contact force only exists when the two bodies are touching.

Complementarity constraints are commonly used to mathematically represent the constraints above. In addition, the Coulomb friction law which governs the magnitude of friction forces, and the maximum dissipation principle which governs the direction of friction force during sliding, can also be represented with additional complementarity constraints [10, 11].

From a numerical optimization perspective, complementarity constraints are challenging to handle as they have no interior and violate constraint qualifications which are needed by gradient-based methods to converge [12, §11]. As a result, in contact dynamics simulation, specialized algorithms such as pivoting methods or projected Gauss-Seidel [13] have been developed to efficiently handle the non-smoothness introduced by complementarity constraints. Instead of following the gradient of nonsmooth constraints, these algorithms conduct a discrete search over which sides of the complementarity constraints are active: they start with a guess of the active sides and iteratively improves the guess until an iteration limit is reached, much like how the simplex algorithm searches for the optimal active set. Such algorithms are ideal if real-timeness is more important than physical accuracy, such as in video games: the algorithm can be terminated prematurely without satisfying all constraints.

In contrast, contact-rich planning has different priorities: satisfaction of physics constraints is essential; and the planner has to look ahead more than one step (usually tens or even hundreds of steps) at a time. Therefore, a different set of techniques for handling complementarity constraints in the context of planning have been developed, and will be detailed in the next two sub-sections.

### 1.2.2 Nonlinear Optimization

An optimal control problem with contact dynamics constraints can be solved as a generic nonlinear program [14]. A typical gradient-based solution algorithm starts with an initial guess of the optimal trajectory, and iterates between (i) linearizing the cost and dynamics constraints around the current trajectory, (ii) updating the current trajectory in a direction that locally minimizes the cost $[15,16]$.

As a consequence of the non-smoothness of contact dynamics, its linearization constructed from its gradient is no longer a good local approximation. Optimizing the cost locally with a bad understanding of the constraint landscape will lead to poor convergence behavior.

Smoothing of contact constraints, therefore, has become an essential ingredient for good convergence when solving contact-rich planning problems. There are different ways in which smoothing can be introduced. Posa et al. and Manchester et al. include the complementarity constraints directly as constraints of the optimal control problem. The complementarity constraints are relaxed to have non-empty interior [17, 18], thereby satisfying constraint qualifications [19, §12]. Tassa et al. solve the optimal control problem with iterative LQR (iLQR), and imposes the dynamics constraints using rollouts [20]. Their linearization of the dynamics is smoothed by regularizing the contact forces [21]. Lastly, by penalizing constraint violation in the cost, Mordatch et al. smooths contact constraints by not always strictly enforcing them and their associated non-smoothness [22]. Importantly, smoothing introduces some form of aphysical behavior into contact dynamics (such as the "force-at-a-distance" effect in Posa's relaxation [17] and MuJoCo [23]), and therefore needs to be iteratively decreased in order to respect the true dynamics constraint when the solution algorithm converges.

By "blending" adjacent contact modes (e.g. Fig. 1-1c-d), smoothing extends the region where linearization is valid, thereby improving the performance of nonlinear optimization algorithms that rely heavily on linearization of dynamics constraints. With the help of smoothing, nonlinear optimization can find trajectories for complex
systems such as humanoids on monkey bars [24]. However, nonlinear optimization frequently gets stuck in local minima due to the inherent non-convexity of many contact-rich planning problems, and thus requires non-trivial initial guesses.

### 1.2.3 Contact Mode Transitions

Another approach to handle the non-smoothness in contact dynamics is to "chop up" the dynamics into smooth pieces (contact modes). The "chopping" makes the dynamics within modes relatively straightforward, and shifts the difficulty of the planning problem to the transitions among modes.

Mixed-Integer Programming (MIP) [25] naturally transcribes the smooth dynamics within modes and the discrete transitions between modes using continuous and integer variables, respectively [26]. For simple systems (e.g. planar linear humanoid between walls [27]), it is possible to discretize the entire state space and ask MIP solvers for a globally optimal plan. Global MIP formulations of contact-rich planning problems have very non-convex optimization landscapes, making them difficult for nonlinear-optimization-based solvers [26].

Solving MIPs can be time-consuming and often too slow for using the planner at real-time rates as a Model-Predictive Controller (MPC). As a result, various techniques have been developed to accelerate solving MIPs in the context of contact-rich planning. Marcucci et al. warm-start new MIPs using optimal solutions of similar problems [28]. Deits et al. learned value functions for a humanoid push-recovery task, making MIPs smaller by reducing the planning horizon [29]. Aydinoglu and Posa developed a specialized MIP solver which is based on the alternating direction method of multipliers (ADMM) and can solve small contact-rich planning problems at real-time rates [30].

Despite these efforts to improve solver efficiency, discretizing the entire state space for even moderately large systems (e.g. pushing a box in a plane) will lead to too many contact modes for exiting methods find a solution within a reasonable amount of time (e.g. a couple of minutes). As a result, global optimality is usually given up in exchange for keeping the size of the planning problem manageable. For instance,
considering only contact modes around a nominal trajectory instead of the entire state space can effectively reduce problem size [31,32]. However, by restricting the planner to search around a nominal trajectory, the ability to globally search the state space, a key feature to escape the local minima which plague methods based on nonlinear optimization, is lost.

Is it possible to retain the ability of global search without incurring the cost of global mode enumeration? If we are willing to give up on global optimality, sampling contact modes, instead of enumerating them, is a good solution. Similar to classical Sampling-Based Motion Planning (SBMP) methods in the configuration space [33], SBMP through contact builds a graph in the state space. From a node in the graph and its corresponding subgoal, the graph can be grown by randomly picking a contact mode and moving towards a subgoal as much as possible without causing a mode switch [34, 35].

SBMP with sampled mode transitions (e.g. [34]) can handle moderately complex systems such as 3D peg-hole insertion with point fingers. It has the best scalability among all planning methods that consider mode transitions. However, the number of contact modes in more complex tasks, such as dexterous in-hand manipulation, will stagnate the search even if mode transitions are sampled instead of enumerated. This suggests that contact-rich manipulation planning may need a method that is fundamentally different from searching through mode switches.

Recently, Marcucci et al. has developed graph of convex sets (GCS), a novel MIP formulation for optimal control problems that offers tight convex relaxations to the original MIPs [36]. Similar to other MIP formulations, GCS does not scale well if the convex sets it searches through are derived from mode enumeration. However, with an appropriate decomposition of the state space that does not directly rely on contact modes, GCS could efficiently search through contact dynamics globally without giving up optimality.

### 1.2.4 Reinforcement Learning

Recent advances in deep RL have created impressive and convincing dexterous manipulation demos where a humanoid robotic hand in-hand re-orients a cube [37,38]. A combination of domain randomization [39] and Proximal Policy Optimization (PPO) [40], the algorithm for training the policy is conceptually straightforward. Yet the complexity demonstrated in the learned policies far exceed the capabilities of the best model-based contact-rich planning methods.

Although training an RL policy requires heavy offline computation, and the learned policy often lacks interpretability and the ability to generalize to new tasks, it is undeniable that RL has solved challenging problems with which model-based methods have struggled. Understanding the reasons behind RL's success from a model-based perspective will help us improve the performance of model-based methods.

Recent works have attributed the success of RL to its stochastic nature, where the contact modes are abstracted by a process of sampling and averaging [41, 42]. This process, termed "randomized smoothing", has a similar smoothing effect on contact dynamics as the various model-based smoothing schemes introduced in Sec. 1.2.2. Another key factor behind the empirical success of RL could lie in its goal of performing global optimization [42], which nonlinear optimization fails to do, and methods based on mode transitions can do only at limited scale.

This hints at a possible strategy of enhancing model-based contact-rich planning methods: if we could combine model-based smoothing with an efficient global search scheme, we might be able to achieve RL-level performance without incurring RL-level computational cost.

### 1.3 Contributions and Thesis Structure

Smoothing and global search are both essential to the success of contact-rich planning, but as far as we know, no existing method does both. This thesis proposes a contactrich planning algorithm that searches globally under the guidance of a smoothed contact model. The proposed method can, within 1-minute of wall-clock time on a
regular desktop computer, generate contact-rich plans for complex robotic systems such as dexterous hands.

At the heart of the model-based planning approach we took lies the question: what is the right model for contact-rich manipulation? In Chapter 2, we present a novel formulation of rigid-body contact dynamics that is quasi-static, convex, differentiable and amenable to smoothing, all of which are features chosen to improve the efficiency of the planner.

In Chapter 3, we present the global contact-rich planner that utilizes the proposed quasi-static contact model and combines SBMP with trajectory optimization. Efficient global exploration is carried out with a kino-dynamic sampling-based motion planner that searches with large step sizes near the contact manifold. By abstracting contact modes away with a smoothed version of the contact dynamics, the sampling-based planner can effectively explore the state space without suffering from the exponential explosion of contact modes. Once a coarse path is found by the sampling-based planner, the path is refined by a trajectory optimizer with finer step sizes to improve its quality and physical accuracy.

The challenges facing contact-rich manipulation extend far beyond planning. Once a plan is generated, interaction forces between the robot and the objects need to be sensed and estimated. The estimated forces, together with other estimated states, are then fed to a stabilizing controller that is aware of making and breaking contacts. Therefore, this thesis also explores (i) external contact force estimation from joint torque measurements (Chapter 4), and (ii) reconciling tracking trajectory and bounding contact forces using a inverse-dynamics controller in the simple case of interacting with a static environment (Chapter 5). Interestingly, the methods in both Chapter 4 and 5 benefit from the simplifying quasi-static assumption, highlighting the utility of quasi-static dynamics in various components of the robotic manipulation pipeline.

## Chapter 2

## An Effective Contact Dynamics Model for Contact-rich Planning

### 2.1 Introduction

We believe an effective model for contact-rich manipulation planning needs to be (i) numerically robust and (ii) differentiable, so that local linearizations can be reliably computed for planning. Most importantly, the model should be able to (iii) predict the long-term behavior of the system, so that the planner can look far ahead while taking few steps. Last but not least, the model should be (iv) amenable to smoothing in order to provide more informative locally linear models across contact modes.

In this chapter, we present a contact dynamics model that satisfies all of the requirements above. Combining these ingredients enables our planner to reliably generate complex contact-rich plans that have so far been considered challenging for existing model-based approaches.

Specifically, we propose a convex implicit time-stepping contact model (Sec. 2.3.1). Among several existing schemes to convexify frictional contact constraints (Sec. 2.2.1), we adopt Anitescu's convex relaxation [43] which in practice only introduces mild nonphysical behavior [44,45]. The convexity provides a clear numerical advantage over the traditional Linear Complementarity Problem (LCP) formulation [10, 11]. Moreover, the derivatives of the dynamics with respect to the current state and control action
can be readily computed using sensitivity analysis for convex conic programs [46] (Sec. 2.3.2). In addition, we will illustrate how Anitescu's convex relaxation compares to the LCP formulation, and why it introduces a non-physical "boundary layer" during sliding (Sec. 2.3.4).

For long-term predictability, we adopt the quasi-static assumption widely used in robotic manipulation [32,34, 47, 48]. A quasi-static model "sees" further than its second-order counterpart by ignoring transient dynamics and focuses on transitions between steady-states. Furthermore, quasi-static models are simpler: not only do they have half as many states, they also do not need parameters to describe damping, as damping is modeled by throwing away kinetic energy at every time step.

In our quasi-static model, robots are modeled as an impedance source. However, throughout the long history of quasi-static models in robotic manipulation planning (Sec. 2.2.1), robots have often been modeled as an admittance source. This can lead to an ill-defined behavior where the output cannot be uniquely determined by the input, thereby limiting the utility of quasi-static models in contact-rich planning. In Sec. 2.3.5, we will elaborate on the limitations of the admittance modeling choice, and how impedance can resolve these problems.

The proposed quasi-static model is validated by running the same input trajectories in a high-fidelity second-order simulator (Sec. 2.3.6). Our results show that our model is able to approximate the second-order dynamics well if the system considered is highly damped and dominated by frictional forces. Furthermore, in contact-rich scenarios, our model can keep the simulation error small at larger step sizes than the second-order simulator.

Lastly, to leverage the benefits of smoothing (Sec. 2.4.1) in contact-rich planning, we show that in addition to the standard randomized smoothing schemes (Sec. 2.4.2), our contact model can be smoothed out using a log-barrier relaxation (Sec. 2.4.3). In this relaxation scheme, the hard contact constraints are softly enforced by a logbarrier function, a common technique used in the interior-point method for convex programs [49, §11] [50]. We further show that the gradients of the smoothed contact model can be easily computed with the implicit function theorem.

### 2.2 Background

In this section, we will review how quasi-static dynamical models have been applied in manipulation planning, and highlight the flaw of modeling robots as an admittance source (Sec. 2.2.1). We will also discuss the features of and trade-offs between various convex relaxations of rigid-body contact models (Sec. 2.2.2).

### 2.2.1 Quasi-static Models in Robotic Manipulation

Quasi-static models, which previous works have used extensively for robotic manipulation planning and control [32,34,47,48,51], simplify Newtonian dynamics by removing terms related to velocity and acceleration, and focusing on contact forces at the core of robot-object interactions. Although the model cannot describe highly-dynamics behaviors such as spinning a pen between fingers [22], it holds up for a wide variety of manipulation tasks, including many that involve dexterous hands [37].

Quasi-static models also have a long history in robotic manipulation. Planar pushing is perhaps one of the earliest problems where quasi-static models are applied. Pioneering work by Mason and Lynch focuses on predicting the motion of an object supported on a horizontal surface without knowledge of the pressure distribution between the object and the surface [52,53]. Later work incorporates a detailed pressure distribution $[54,55]$ and stochasticity [56] into the quasi-static planar pushing model. Such models have been effectively employed in a Model Predictive Control (MPC) framework which enables an object pushed by a single point contact to track complex planar trajectories [31]. In addition to planar pushing, quasi-static models have also been applied in simple, planar instances of dexterous manipulation [57,58].

Halving the number of states by removing velocity is one of the most recognized advantages of quasi-static systems [47, §10.1]. However, by focusing on transitions between static equilibria and ignoring transients induced by damping and acceleration, a quasi-static model can look further into the future while taking fewer steps than its second-order counterpart, thereby reducing the planning horizon. We believe this temporal aspect of quasi-static models' advantage is important but often
overlooked in the literature. In practice, many trajectory optimization formulations for manipulation enforce the second-order Newtonian dynamics [59, 60]. Although modeling the transients allows the discovery of more dynamic behaviors, we believe, especially in the manipulation setting, that the added computational complexity due to long planning horizon oftentimes outweighs the benefits.

Despite their popularity, many popular variants of quasi-static models, such as [31,52,53,56], have a serious flaw: the next state cannot be always be uniquely determined from the current state and control input. In quasi-static models, the control input is usually defined as the position commands of the robots. Many quasistatic models also assume that the position commands can be perfectly executed as prescribed robot motions. Under this assumption, contact forces and object motions can be ambiguous for the same control input. This flaw does not manifest in simple tasks such as planar pushing, where robot motion does uniquely determine object motion. On the other hand, it can be crippling in more complex tasks such as grasping (more details in Sec. 2.3.5).

It is possible to circumvent the ambiguity by including contact forces as additional states of the dynamics model [57,61]. However, reasoning about both positions and forces compels the planner to handle the complementarity constraints between forces and distances, thereby limiting the algorithms the planner can employ.

A more fundamental remedy to this flaw is to account for the unmodeled compliance in the system, thereby allowing commanded and actual robot positions to be different. For instance, Harm and Posa added compliance at the robot joints by modeling the robot as velocity-controlled with finite gains [62]; in previous work we added compliance at the contact points [63]. However, these remedies do not always reflect the reality on robotic hardware, which often consists of rigid objects and impedance-controlled robots. Moreover, [63] is too computationally expensive, and [62] is developed specifically for manipulating planar objects supported on a tabletop.

In Sec. 2.3.5, we will discuss in more detail the flaws of commanding prescribed robot motions, and how our quasi-static formulation resolves it in a computationally-
efficient and hardware-consistent manner.

### 2.2.2 Convex Rigid-body Contact Dynamics

Without friction, quasi-static dynamics with frictionless unilateral contacts can be written as the KKT optimality conditions of a convex Quadratic Program (QP) that minimizes the system's potential energy subject to non-penetration constraints [44]. The complementarity between forces and distances (Sec 1.2.1) are captured by the complementarity slackness between primal constraints and dual variables. However, the Coulomb friction law and the maximum dissipation principle, both of which are needed to model frictional contacts, introduce additional complementarity constraints that can no longer be accommodated by a convex program. Consequently, contact dynamics with Coulomb friction is formulated as an LCP, which is solved by specialized non-convex solvers [10,11].

Anitescu restored convexity in contact dynamics by replacing the linear complementarity constraints with cone complementarity constraints. The price of convexity is that the sliding velocity is no longer restricted to the tangent plane defined by the contact normal $[13,43]$. The resulting program is a convex QP with discretized friction cones, or a Second-Order Cone Program (SOCP) with circular friction cones. Both formulations can be readily solved with commercial convex solvers such as [64].

Anitescu's convex relaxation exactly reproduces Coulomb friction when the contact is sticking, and injects a non-physical "boundary layer" between relatively-sliding objects. A big advantage of Anitescu's relaxation is that the extent of relaxation is controlled by only one hyperparameter: the step size. As the step size converges to 0 , the boundary layer also vanishes [43]. The step size therefore can serve as a knob that allows the planner to trade the dynamics model's predictive power for physical accuracy. We will discuss this non-physical behavior in more detail in Sec. 2.3.4.

The popular MuJoCo simulator $[21,65]$ solves a convex program that closely resembles the dual of Anitescu's formulation [9, Appendix B]. Moreover, Mujoco also adds regularization on contact forces to the objective function in order to improve numerical conditioning and make the dynamics invertible. However, the regulariza-
tion introduces non-physical behaviors such as "force at a distance" that are harder to control as they are induced by many hyper-parameters [23].

Recently, Castro et al. developed an unconstrained convex formulation of contact dynamics [45] that combines many of the nice features from MuJoCo [21] and Anitescu's convex relaxation [43]. For instance, Castro's primal formulation is similar to Anitescu's; the contact force regularization and the removal of friction cone constraints using a projection function are inspired by similar techniques from MuJoCo. Castro's formulation is designed from ground up for efficient and accurate physics simulation. For instance, it provides physical interpretations and units to the contact force regularization parameters that are originally proposed by MuJoCo to improve numerical conditioning.

### 2.3 A Convex, Quasi-static and Differentiable Contact Model

In this section, we present the Convex, Quasi-static, Differentiable Contact (CQDC) model which will be used for the contact-rich planning tasks in Chapter 3. Specifically, we will start with the equations of motion of quasi-static contact dynamics, and massage it into the KKT optimality conditions of a convex optimization program (Sec. 2.3.1). We will then describe how to take the (discontinuous) derivatives of the proposed contact dynamics (Sec. 2.3.2), and how the forward dynamics and gradient computation are implemented (Sec. 2.3.3). In addition, we will illustrate how Anitescu's convex relaxation introduces a non-physical "boundary layer" during sliding (Sec. 2.3.4). We will also dive more deeply into the common flaw in quasistatic dynamical models brought up in Sec. 2.2, and explain how our quasi-static model resolves it (Sec. 2.3.5). Lastly, with two examples of common manipulation tasks, we show that the proposed quasi-static model and an established second-order simulator (Drake [66]) generate very similar state trajectories for the same action sequence (Sec. 2.3.6).

### 2.3.1 Forward Dynamics

Despite the numerical advantage of convex contact dynamics formulations [21, 43, 45], some formulations can produce highly inaccurate contact forces with poorly-chosen hyperparameters [23]. Our formulation adopts Anitescu's convex relaxation of the Coulomb friction constraints [43]. Anitescu's convex relaxation is equivalent to common LCP formulations [10] in non-sliding contacts or in separation, and introduces a non-physical yet mild "boundary layer" effect between relatively-sliding objects [44, 45]. Not only does Anitescu's convex friction model have the simulation step size $h$ as the only hyperparameter, the "boundary layer" also disappears as $h \rightarrow 0$. The step size therefore can serve as a knob that allows the planner to trade the dynamics model's predictive power for physical accuracy.

A generic discrete-time dynamical system is written as

$$
\begin{equation*}
x_{+}=f(x, u) \tag{2.1}
\end{equation*}
$$

where $x \in \mathbb{R}^{n}$ is the state, $u \in \mathbb{R}^{m}$ the control input, and $f: \mathbb{R}^{n} \times \mathbb{R}^{m} \rightarrow \mathbb{R}^{n}$ the system dynamics. In the robotic manipulation setting, a system can be divided into $n_{\mathrm{a}}$ actuated Degree of Freedoms (DOFs), which correspond to the robots, and $n_{\mathrm{u}}$ unactuated DOFs, which correspond to the objects. The configurations of objects and robots are denoted respectively by $q^{u} \in \mathbb{R}^{n_{u}}$ and $q^{\mathrm{a}} \in \mathbb{R}^{n_{\mathrm{a}}}$. The system state is defined by $x:=q:=\left(q^{\mathrm{u}}, q^{\mathrm{a}}\right)$. We model the robots as impedances [67], which in the quasi-static setting reduces to springs with a diagonal stiffness matrix $\mathbf{K}_{\mathrm{a}} \in \mathbb{R}^{n_{\mathrm{a}} \times n_{\mathrm{a}}}$. Accordingly, the input $u \in \mathbb{R}^{n_{a}}$ is defined by the commanded positions of the robots' joints, which can also be interpreted as the equilibrium positions of the springs.

The discretized quasi-static equations of motion are

$$
\begin{align*}
h \mathbf{K}_{\mathrm{a}}\left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}-u\right) & =h \tau^{\mathrm{a}}+\sum_{i=1}^{n_{\mathrm{c}}}\left(\mathbf{J}_{\mathrm{a}_{i}}\right)^{\top} \lambda_{i},  \tag{2.2a}\\
\left(\frac{\epsilon}{h} \mathbf{M}_{\mathrm{u}}\right) \delta q^{\mathrm{u}} & =h \tau^{\mathrm{u}}+\sum_{i=1}^{n_{\mathrm{c}}}\left(\mathbf{J}_{\mathrm{u}_{i}}\right)^{\top} \lambda_{i}, \tag{2.2b}
\end{align*}
$$

where $h \in \mathbb{R}_{++}$is the step size in seconds; $\mathbf{M}_{\mathbf{u}} \in \mathbb{R}^{n_{\mathrm{u}} \times n_{\mathrm{u}}}$ is the mass matrix of the objects; $\epsilon \in \mathbb{R}_{+}$is a small regularization constant; the change in system configuration from the current to the next step is $\delta q:=\left(\delta q^{\mathrm{u}}, \delta q^{\mathrm{a}}\right) ; \tau^{\mathrm{a}} \in \mathbb{R}^{n_{\mathrm{a}}}$ and $\tau^{\mathrm{u}} \in \mathbb{R}^{n_{\mathrm{u}}}$ are non-contact external torques (e.g. due to gravity) for robots and objects. There are $n_{\mathrm{c}}$ contact pairs at the current time step. For the $i$-th contact, $\lambda_{i} \in \mathbb{R}^{3}$ is the contact impulse; $\mathbf{J}_{\mathrm{a}_{i}} \in \mathbb{R}^{3 \times n_{\mathrm{a}}}$ and $\mathbf{J}_{\mathrm{u}_{i}} \in \mathbb{R}^{3 \times n_{\mathrm{u}}}$ are the contact Jacobians [68] of the robots and the objects, respectively. For 3D systems, it is common that the rotational displacement and the angular velocity are different and related by a linear map. As the technique to deal with this difference is standard (e.g. [45, Section II]), we leave it out here for brevity and notational simplicity.

Equation (2.2a) states that the robot joint positions' deviation from their commanded values $u$, as a result of external impulses, is proportional to the stiffness $\mathbf{K}_{\mathrm{a}}$. Equation (2.2b) can have different physical interpretations depending on the value of the regularization constant $\epsilon$. If $\epsilon=0,(2.2 b)$ is the force (impulse) balance of the objects. This corresponds to the exact quasi-static formulation in [44]. If $\epsilon=1$, (2.2b) is the standard second-order rigid body equations of motion (expressed in impulse-momentum form) for a system with 0 velocity at the current time step. This corresponds to Mason's classical definition of quasi-dynamic systems [47]. Note that for any $\epsilon$, the momentum gained at every step due to external impulses is discarded at the next time step, which is characteristic of the highly damped behavior typical in quasi-static systems.

In our physical experiments in later sections where objects are light, we find that the dynamics is closer to being exactly quasi-static $(\epsilon=0)$ than being classically quasi-dynamic $(\epsilon=1)$. Therefore, we pick $\epsilon$ to be as small as possible without causing numerical issues. A rule of thumb is to set $\epsilon$ such that the largest eigenvalue of $\left(\epsilon \mathbf{M}_{\mathbf{u}} / h\right)$ is one tenth of the smallest eigenvalue of $\left(h \mathbf{K}_{\mathrm{a}}\right)$ (see the definition of $\mathbf{Q}$ in (2.6c)).

In addition, the Coulomb friction model requires the contact impulses $\lambda_{i}$ to stay inside the friction cone, and the relative sliding velocities to satisfy the maximum dissipation principle [10]. To enforce these constraints, we first introduce some additional
notation. The contact Jacobian for contact $i$ is

$$
\mathbf{J}_{i}:=\left[\mathbf{J}_{\mathrm{u}_{i}}, \mathbf{J}_{\mathrm{a}_{i}}\right]:=\left[\begin{array}{l}
\mathbf{J}_{\mathrm{n}_{i}}  \tag{2.3}\\
\mathbf{J}_{\mathrm{t}_{i}}
\end{array}\right] \in \mathbb{R}^{3 \times\left(n_{\mathrm{u}}+n_{\mathrm{a}}\right)},
$$

where $\mathbf{J}_{\mathrm{n}_{i}} \in \mathbb{R}^{1 \times\left(n_{\mathrm{u}}+n_{\mathrm{a}}\right)}$ maps the generalized velocity of the system to the normal contact velocity, and $\mathbf{J}_{\mathrm{t}_{i}} \in \mathbb{R}^{2 \times\left(n_{\mathrm{u}}+n_{\mathrm{a}}\right)}$ to the tangent velocities. Next, the friction cone at contact $i, \mathcal{K}_{i}$, and its dual cone $\mathcal{K}_{i}^{\star}$, are denoted by

$$
\begin{align*}
\mathcal{K}_{i} & :=\left\{\lambda_{i}=\left(\lambda_{\mathrm{n}_{i}}, \lambda_{\mathrm{t}_{i}}\right) \in \mathbb{R}^{3} \mid \mu_{i} \lambda_{\mathrm{n}_{i}} \geq \sqrt{\lambda_{\mathrm{t}_{i}}^{\top} \lambda_{\mathrm{t}_{i}}}\right\},  \tag{2.4a}\\
\mathcal{K}_{i}^{\star} & :=\left\{v_{i}=\left(v_{\mathrm{n}_{i}}, v_{\mathrm{t}_{i}}\right) \in \mathbb{R}^{3} \mid v_{\mathrm{n}_{i}} \geq \mu_{i} \sqrt{v_{\mathrm{t}_{i}}^{\top} v_{\mathrm{t}_{i}}}\right\}, \tag{2.4b}
\end{align*}
$$

where $\mu_{i}$ is the friction coefficient; the dual variable $v_{i}$ can also be interpreted as the relative contact velocity for contact $i$; the subscripts $(\cdot)_{\mathrm{n}}$ and $(\cdot)_{\mathrm{t}}$ indicate respectively the normal and tangential components.

With this notation, Anitescu's frictional contact constraints can be written as:

$$
\begin{align*}
v_{i}:=\mathbf{J}_{i} \delta q+\left[\begin{array}{c}
\phi_{i} \\
0_{2}
\end{array}\right] & \in \mathcal{K}_{i}^{\star},  \tag{2.5a}\\
\lambda_{i} & \in \mathcal{K}_{i},  \tag{2.5b}\\
v_{i}^{\top} \lambda_{i} & =0, \tag{2.5c}
\end{align*}
$$

where $\phi_{i} \in \mathbb{R}$ is the signed distance for contact $i$ at the current time step. These constraints enforce the Coulomb friction model exactly when a contact is sticking (not sliding) and in separation. In sliding, these constraints enforce maximum dissipation, but adds a small gap between the two relatively-sliding objects. Anitescu showed that this gap converges to 0 as $h \rightarrow 0$ [43].

Remarkably, this implies that the quasi-static equations of motion (2.2), together with the friction constraints (2.5), are the KKT optimality conditions [49, §5.9] of the
following convex Second-Order Cone Program (SOCP) [43]:

$$
\begin{align*}
& \min _{\delta q} . \frac{1}{2} \delta q^{\top} \mathbf{Q} \delta q+b^{\top} \delta q, \text { subject to }  \tag{2.6a}\\
& \mathbf{J}_{i} \delta q+\left[\begin{array}{c}
\phi_{i} \\
0_{2}
\end{array}\right] \in \mathcal{K}_{i}^{\star}, \forall i \in\left\{1 \ldots n_{\mathrm{c}}\right\} \text {, where }  \tag{2.6b}\\
& \mathbf{Q}:=\left[\begin{array}{cc}
\epsilon \mathbf{M}_{\mathrm{u}} / h & 0 \\
0 & h \mathbf{K}_{\mathrm{a}}
\end{array}\right], b:=-h\left[\begin{array}{c}
\tau^{\mathrm{u}} \\
\mathbf{K}_{\mathrm{a}}\left(u-q^{\mathrm{a}}\right)+\tau^{\mathrm{a}}
\end{array}\right] \tag{2.6c}
\end{align*}
$$

whose primal and dual solutions, $\delta q^{\star}$ and $\lambda^{\star}:=\left(\lambda_{1}^{\star}, \ldots, \lambda_{n_{\mathrm{c}}}^{\star}\right)$, can be obtained by conic solvers such as $[64,69]$. Note that when $\epsilon=0$, which corresponds to strictly enforcing force balance on the object, the objective (2.6a) is positive semi-definite, and thus (2.6) may have multiple solutions.

The SOCP (2.6) reduces to a QP when there is no friction or if the friction cone can be described by linear constraints. The second-order friction cones (2.4) can be equivalently represented as linear constraints in the planar case, or be approximated with polyhedral cones [10] in the 3D case. However, this approximation introduces non-physical anisotropic behaviors [50,70], which is why (2.4) is preferred.

### 2.3.2 Differentiability

We illustrate how to compute the derivatives of the system configuration at the next step, $q_{+}$, with respect to the current $q$ and $u$. Let us express the forward contact dynamics in the standard dynamical system form (2.1):

$$
\begin{equation*}
q_{+}=f(q, u)=q+\delta q^{\star}(q, u) \tag{2.7}
\end{equation*}
$$

where $\delta q^{\star}$ is the solution to (2.6). Taking the derivatives of (2.7) yields

$$
\begin{equation*}
\mathbf{A}=\frac{\partial f}{\partial q}=\mathbf{I}+\frac{\partial \delta q^{\star}}{\partial q}, \mathbf{B}=\frac{\partial f}{\partial u}=\frac{\partial \delta q^{\star}}{\partial u} \tag{2.8}
\end{equation*}
$$

where $\partial \delta q^{\star} / \partial q$ and $\partial \delta q^{\star} / \partial u$ can be expanded using the chain rule into:

$$
\begin{align*}
\frac{\partial \delta q^{\star}}{\partial q} & =\frac{\partial \delta q^{\star}}{\partial b} \frac{\partial b}{\partial q}+\frac{\partial \delta q^{\star}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial q}+\sum_{i=1}^{n_{\mathrm{c}}} \frac{\partial \delta q^{\star}}{\partial \mathbf{J}_{i}} \frac{\partial \mathbf{J}_{i}}{\partial q}+\frac{\partial \delta q^{\star}}{\partial \phi_{i}} \frac{\partial \phi_{i}}{\partial q}  \tag{2.9a}\\
\frac{\partial \delta q^{\star}}{\partial u} & =\frac{\partial \delta q^{\star}}{\partial b} \frac{\partial b}{\partial u} \tag{2.9b}
\end{align*}
$$

Similar to other differentiable simulators based on implicit time-stepping [50, 71], we compute the derivatives of the solution $\delta q^{\star}$ with respect to the problem data $\left(\mathbf{Q}, b, \mathbf{J}_{i}, \phi_{i}\right)$ by applying the implicit function theorem to the KKT optimality conditions of the convex program (2.6) [46]. Then, the derivatives of $\left(\mathbf{Q}, b, \mathbf{J}_{i}, \phi_{i}\right)$ with respect to $q$ and $u$ can be straightforwardly computed using either automatic differentiation or a more specialized method that takes advantage of the structure of rigid body systems [72].

As the derivatives (2.9) are discontinuous functions of $(q, u)$ [41], they are not a good local approximation of the CQDC dynamics unless multiple gradients are computed for different state-action pairs and averaged using the first-order randomized smoothing scheme (Sec. 2.4.1).

### 2.3.3 Implementation

As all existing rigid-body differentiable simulators, to the best of our knowledge, assume second-order Newtonian dynamics, it is difficult and inefficient to modify them to support our CQDC dynamics formulation. Therefore, we implemented the proposed dynamics formulation using the Drake robotics toolbox [66]. Although our implementation is not heavily optimized, it is adequate for computing contact-rich plans within a reasonable amount of wall-clock time (a few minutes).

For the forward dynamics (Sec. 2.3.1), we use Drake's MultibodyPlant and SceneGraph for collision detection and the computation of the object mass matrix $\mathbf{M}_{u}$, contact Jacobians $\mathbf{J}_{i}$ and signed distances $\phi_{i}$. The SOCP (2.6) can then be constructed with MathematicalProgram and solved with a third-party solver of our choice.

For the dynamics derivatives (Sec. 2.3.2), we have a custom implementation for
differentiating through the KKT optimality conditions of an SOCP using Eigen's [73] linear solvers, allowing us to efficiently compute the partial derivatives of $\delta q^{\star}$ with respect to $\left(\mathbf{Q}, b, \mathbf{J}_{i}, \phi_{i}\right)$ from primal-dual solutions $\left(\delta q^{\star}, \lambda^{\star}\right)$ given by third-party conic solvers. As for the partials of $\left(\mathbf{Q}, b, \mathbf{J}_{i}, \phi_{i}\right)$ with respect to $q$, we note that $b$ is a linear function of $q ; \partial \phi_{i} / \partial q=\mathbf{J}_{\mathbf{n}_{i}} ; \partial \mathbf{J}_{i} / \partial q$ and $\partial \mathbf{Q} / \partial q$ are computed with Drake's forward-mode automatic differentiation.

Finally, in order to avoid discontinuities coming from collision detection, we curate our system models so that every contact pair is either sphere-sphere, sphere-box, or sphere-cylinder, which means the contact points and normals change smoothly with the system configuration $q$ [13]. For example, in Fig. 3-5, both the box-shaped fingers of the Allegro hand and the box-shaped manipuland in the Allegro Plate system are represented as arrays of inscribing spheres for the purpose of collision detection. We note that this limitation can be alleviated with smoothing over collision geometries [74, 75].

### 2.3.4 Interpreting Anitescu's Convex Relaxation

In this subsection, we will explain for planar (2D) systems how complementarity constraints in Anitescu's relaxation behave in different contact modes, namely sticking, sliding and separation. We will also illustrate the cause of the "boundary layer" artifact during sliding.

When both robots and objects are constrained in a plane, the CQDC dynamics (2.6) simplifies to the following QP:

$$
\begin{align*}
\min _{\delta q} . & \frac{1}{2} \delta q^{\top} \mathbf{Q} \delta q+b^{\top} \delta q, \text { subject to }  \tag{2.10a}\\
& \left(\mathbf{J}_{\mathbf{n}_{i}}+\mu_{i} \mathbf{J}_{\mathrm{t}_{i}}\right) \delta q+\phi_{i} \geq 0, i \in\left\{1 \ldots n_{\mathrm{c}}\right\},  \tag{2.10b}\\
& \left(\mathbf{J}_{\mathrm{n}_{i}}-\mu_{i} \mathbf{J}_{\mathrm{t}_{i}}\right) \delta q+\phi_{i} \geq 0, i \in\left\{1 \ldots n_{\mathrm{c}}\right\}, \tag{2.10c}
\end{align*}
$$

where the contact Jacobian $\mathbf{J}_{\mathrm{t}_{i}}$ has only one row instead of two, and the conic contact constraint (2.6b) reduces to two inequality constraints (2.10b) and (2.10c). The KKT
conditions of (2.10) include the following complementarity constraints for the $i$-th contact pair:

$$
\begin{align*}
& 0 \leq \beta_{i 1} \perp\left(\mathbf{J}_{\mathrm{n}_{i}}+\mu_{i} \mathbf{J}_{\mathrm{t}_{i}}\right) \delta q+\phi_{i} \geq 0  \tag{2.11a}\\
& 0 \leq \beta_{i 2} \perp\left(\mathbf{J}_{\mathrm{n}_{i}}-\mu_{i} \mathbf{J}_{\mathrm{t}_{i}}\right) \delta q+\phi_{i} \geq 0 \tag{2.11b}
\end{align*}
$$

where the Lagrange multipliers $\beta_{i 1}$ and $\beta_{i 2}$ are no longer the normal and tangential component of the contact impulses. Instead, they are the components of the contact impulses along the extreme rays of the friction cone (Fig. 2-1a).

As the interplay between contact impulses $\left(\beta_{i 1}, \beta_{i 2}\right)$ and non-penetration constraints (RHS of (2.11)) are best shown in the contact frame of contact pair $i$, we define the relative Cartesian translation of contact pair $i$ during the time step as $\delta \mathrm{x}_{i}:=\left[\delta \mathrm{n}_{i}, \delta \mathrm{t}_{i}\right] \in \mathbb{R}^{2}$, where $\delta \mathrm{n}_{i}:=\mathbf{J}_{\mathrm{n}_{i}} \delta q \in \mathbb{R}$ and $\delta \mathrm{t}_{i}:=\mathbf{J}_{\mathrm{t}_{i}} \delta q \in \mathbb{R}$ are the normal and tangential components of $\delta \mathrm{x}_{i}$ respectively. These quantities are illustrated in Fig. 2 -1c. With this notation, (2.11) can be written as

$$
\begin{align*}
& 0 \leq \beta_{i 1} \perp \delta \mathrm{n}_{i}+\mu_{i} \delta \mathrm{t}_{i}+\phi_{i} \geq 0  \tag{2.12a}\\
& 0 \leq \beta_{i 2} \perp \delta \mathrm{n}_{i}-\mu_{i} \delta \mathrm{t}_{i}+\phi_{i} \geq 0 \tag{2.12b}
\end{align*}
$$

Note that (i) the feasible region of $\delta \mathrm{x}_{i}$ is defined by the RHS of (2.12a) and (2.12b); (ii) both boundaries of the feasible region of $\delta x_{i}$ (the blue and red dashed lines in Fig. 2-1) intersect the $\mathrm{n}_{i}$-axis at $-\phi_{i}$, which is non-positive; (iii) the normal and tangential contact force impulses are given respectively by $\lambda_{\mathrm{n}_{i}}=\beta_{i 1}+\beta_{i 2}$ and $\lambda_{\mathrm{t}_{i}}=\mu_{i}\left(\beta_{i 1}-\beta_{i 2}\right)$.

## Sticking (Fig. 2-1a)

In a sticking contact, $\delta \mathrm{x}_{i}=0, \phi_{i}=0$ and the contact force is inside the friction cone. The conditions for sticking, $\delta \mathrm{x}_{i}=0$ and $\phi_{i}=0$, imply that $\delta \mathrm{n}_{i}+\mu_{i} \delta \mathrm{t}_{i}+\phi_{i}=0$ and $\delta \mathbf{n}_{i}-\mu_{i} \delta \mathrm{t}_{i}+\phi_{i}=0$, i.e. the RHS of (2.12a) and (2.12b) are active. Therefore, both $\beta_{1 i}$ and $\beta_{1 i}$ can be positive, allowing any contact impulse inside the friction cone. In this case, Anitescu's constraints are identical to Coulomb's friction law.


Figure 2-1: Anitescu's friction constraints (2.12) under different contact modes: (a) sticking, (b) sliding, and (c) separation. The normal and tangent of the contact frame $i$ are denoted respectively by $n_{i}$ and $t_{i}$. The green shaded area is the friction cone. The purple shaded area is the feasible region of $\delta \mathrm{x}_{i}$. Constraints corresponding to (2.12a) and (2.12b) are color-coded blue and red, respectively.

## Sliding (Fig. 2-1b)

In a sliding contact, the contact force is on the boundary of the friction cone, and the relative displacement $\delta \mathrm{x}_{i}$ is horizontal and opposing the friction force. Without loss of generality, we can assume $\beta_{i 2}>0$ and $\beta_{i 1}=0$. Hence the RHS of (2.12b) is active, i.e. $\delta \mathrm{x}_{i}$ is constrained to the red dashed line defined by $\delta \mathrm{n}_{i}-\mu_{i} \delta \mathrm{t}_{i}+\phi_{i}=0$. Therefore, $\phi_{i}=\mu_{i} \delta \mathrm{t}_{i}>0$ per the definition of sliding ( $\delta \mathrm{n}_{i}=0$ and $\delta \mathrm{t}_{i}>0$ ). This is the source of the non-physical behavior of Anitescu's friction constraints: when one body is sliding relative to the other, the body slides at a positive distance away from the other body instead of on its surface. In other words, to achieve pure sliding, there must be some separation between the two relative-sliding objects.

## Separation (Fig. 2-1c)

Separation indicates that there is no contact force, i.e. $\beta_{1 i}=\beta_{2 i}=0$. Hence $\delta \mathbf{x}_{i}$ can take any value in the feasible region defined by the RHS of (2.12a) and (2.12b). For moderate $\mu_{i}$ and reasonably large $\phi_{i}$, the feasible region is large enough to accommodate a wide range of velocities.

### 2.3.5 Flaws of Prescribed Robot Motions as Control Input

In Sec. 2.3.1, we model robots as impedance sources, but not all quasi-static models have made this modeling choice. As discussed earlier in Sec. 2.2.1, many existing quasi-static models assume the control input $u$ prescribes robot motions (admittance source) $[31,52,53,56]$, which can be sufficient for planning but insufficient for simulation. In this subsection, we will illustrate the flaws of modeling robot as prescribed motion for simulation from two complementary perspectives: (i) an example of a parallel-jaw gripper lifting up a sphere, and (ii) bond graphs.

## Parallel-jaw Gripper Lifting Up a Sphere

The gripper-ball system shown in Fig. 2-2a has three actuated DOFs: $x_{l}$ and $x_{r}$ are the translation of the left and right gripper fingers along the $x$-axis, respectively; $y$ is


Figure 2-2: (a) A planar quasi-static multibody system. The red rectangles are the actuated gripper fingers, $q^{\mathrm{a}}=\left[x_{l}, x_{r}, y\right]$ and $u=\left[u_{l}, u_{r}, u_{y}\right]$. The gray sphere is the un-actuated manipuland, $q^{\mathrm{u}}=\left[x_{c}, y_{c}\right] . r=0.1 \mathrm{~m}$. The normal contact forces associated with contact pair 1 and 2 are denoted respectively by $\lambda_{n_{1}}$ and $\lambda_{n_{2}}$. (b) A grasping command violating the non-penetration constraint.
the translation of both fingers along the $y$-axis. The two un-actuated DOFs, $x_{c}$ and $y_{c}$, are the $x$ and $y$ translation of the sphere.

As a consequence of modeling robots as prescribed motions, the robot force balance condition (2.2a) is replaced by

$$
\begin{equation*}
q^{\mathrm{a}}+\delta q^{\mathrm{a}}=u \tag{2.13}
\end{equation*}
$$

Although seemingly plausible, the simplistic robot dynamics (2.13) is an ill-posed dynamical system: (i) it is possible to command a $u$ that violates non-penetration constraints; (ii) when two bodies are in contact, the contact force between them can be under-determined [62,63]. Instead of belonging to a niche set of contrived corner cases, such issues arise naturally and frequently in even the simplest robotic manipulation tasks. Consider the example in Fig. 2-2a, where the fingers are commanded to grasp the sphere. For PD-controlled grippers, it is common to command a small amount of penetration to establish contact forces. However, such commands would penetrate the object, as shown in Fig. 2-2b. On the other hand, even if the fingers are commanded to "graze" the sphere ( $\phi_{1}=\phi_{2}=0$ ), which respects non-penetration constraints, any non-negative contact force $\lambda_{n_{1}}$ and $\lambda_{n_{2}}$ would satisfy (2.2b). This is problematic if the fingers are also commanded to move up: small contact forces would leave the sphere on the ground, but large contact forces would generate enough friction to lift up the sphere. Modeling control inputs as robot motions simply cannot determine whether

(b)


Figure 2-3: Free-body diagrams of the left finger when it is (a) away from the sphere, and (b) in contact with the sphere.
the sphere moves with the hand or stays on the table.
Our impedance formulation (2.2a) resolves the ill-posedness of (2.13) by effectively connecting $q^{\text {a }}$ and $u$ using springs with zero rest lengths. To illustrate how the spring helps, we focus on $x_{l}$, the prismatic joint of left finger in the planar grasping example in Fig. 2-2a. Force balance for $x_{l}$ is given by

$$
\begin{equation*}
k\left(u_{l}-x_{l}\right)+\lambda_{\mathrm{n}_{1}}=0 \tag{2.14}
\end{equation*}
$$

where $k$ is the stiffness of the spring, $k\left(u_{l}-x_{l}\right)$ is the spring force acting on the left finger, and $f_{n_{1}}$ the force from contact with the sphere.

When the left finger is not in the vicinity of the sphere (Fig. 2-3a), we have $\lambda_{\mathrm{n}_{1}}=0$, which together with (2.14) implies that $x_{l}=u_{l}$. Therefore, in the absence of contact, adding the spring has the same effect as (2.13). On the other hand, when the left finger is commanded to squeeze the sphere (Fig. 2-3b), $x_{l}$ and $u_{l}$ are different due to the non-penetration constraint. The spring force $k\left(x_{l}-u_{l}\right)$ is balanced by the contact force $\lambda_{\mathrm{n}_{1}}$.

The addition of the spring resolves both issues with (2.13): (i) feasibility of the non-penetration constraint is retained by allowing $x_{l}$ to be different from its commanded value $u_{l}$; (ii) the magnitude of the contact force is also uniquely determined by the difference between $x_{l}$ and $u_{l}$.

Although adding springs between $q^{\mathrm{a}}$ and $u$ may seem arbitrary, it is equivalent to
modeling the actuators as impedances [67]. For instance, the closed-loop dynamics of the KUKA iiwa arm in joint-impedance mode is:

$$
\begin{equation*}
\mathbf{M}(q) \ddot{q}+\left(\mathbf{D}_{q}+\mathbf{C}(q, \dot{q})\right) \dot{q}+\mathbf{K}_{q}(q-u)=\tau_{\mathrm{ext}} \tag{2.15}
\end{equation*}
$$

where $q$ is the joint angles of the robot arm, $\mathbf{M}(q)$ the mass matrix, $\mathbf{C}(q, \dot{q})$ the Coriolis force, $\mathbf{K}_{q}$ the diagonal joint stiffness matrix, $\mathbf{D}_{q}$ the diagonal damping matrix and $\tau_{\text {ext }}$ the joint torque generated by external contact [76]. Discarding terms related to velocity and acceleration, the second-order dynamics (2.15) becomes

$$
\begin{equation*}
\mathbf{K}_{q}(q-u)=\tau_{\mathrm{ext}} \tag{2.16}
\end{equation*}
$$

which can be interpreted as the joint space version of (2.14).

## Bond Graph

Bond graph is a graphical tool for describing energy flows between components of a dynamical system [77]. In his seminal work on impedance control, Hogan justified modeling robots as impedances using bond graph analysis [78].

Fig. 2-4 shows bond graphs of a 1-DOF quasi-static robot pushing against a wall when the robot is modeled as prescribed motions (admittance) vs. as impedance. Both the robot and the wall are modeled as flow sources. In Fig. 2-4a the flows from both sources are constrained to be equal by the 1 junction, whereas in Fig. 2-4b the flows can be different due to the spring. The flaw of modeling robots as prescribed motions is revealed clearly as a syntax error in the bond graph. In Fig. 2-4a, the 1 junction has two strong bonds, both of which dictate the flow. However, the syntax of 1 junction only allows one strong bond per junction.

### 2.3.6 Experimental Evaluation

We want to understand the extent to which simulation accuracy is affected by the "boundary layer" due to Anitescu's convex relaxation of the Coulomb friction con-


Figure 2-4: Bond graphs of a 1-DOF robot pushing against the wall. In (a), the robot is modeled as admittance (a prescribed motion). In (b), the robot is modeled as impedance.
straint, which has been illustrated in Sec. 2.3.4. By comparing the proposed CQDC dynamics and a quasi-static LCP formulation on the 2D parallel gripper system using a trajectory that involves multiple contact mode changes, we show that the difference between the two friction formulations is small for typical manipulation tasks.

We also want to ensure that quasi-static simulations stay close to their secondorder counterparts for typical manipulation tasks. To this end, we compare the CQDC dynamics against a high-fidelity second-order simulator, Drake's MultibodyPlant (MBP). To make the comparison as fair as possible, we turn on the semi-analytic primal (SAP) solver [45] in MBP. Similar to CQDC, SAP uses Anistecu's convex relaxation of friction constraints [43], and supports implicit integration for PD controllers ${ }^{1}$. Our comparison shows that both CQDC and SAP are stable at large integration step sizes (at least 0.5 s ), but CQDC seems to be better at respecting non-penetration constraints as the step size gets larger, possibly because CQDC enforces non-penetration constraints directly on positions rather than on velocities.

[^1]
## 2D Parallel Gripper

We study the "boundary layer" effect of the CQDC dynamics by comparing it against an LCP-based quasi-static formulation [44] which enforces the same object and robot equations of motion in (2.2), but models friction using linear complementarity constraints [79] instead of the cone complementarity constraints (2.5).

The comparison is done on the 2D grasping example introduced in Fig. 2-2 using a gripper trajectory that induces multiple contact mode changes between the fingers and the sphere. As the system is planar, CQDC's SOCP formulation (2.6) reduces to the QP formulation (2.10). We use the QP formulation for the comparison in this example.

The simulation time step $h$ is set to 0.01 s ; the weight of the sphere is 10 N ; the stiffness for all actuated DOFs is $1000 \mathrm{~N} / \mathrm{m}$ and a friction coefficient of 0.5 is used for all contacts. The regularization constant $\epsilon$ in (2.2b) is set to 0 . We will use $c_{\mathrm{n}_{i}}:=\lambda_{\mathrm{n}_{i}} / h$ and $c_{\mathrm{f}_{i}}:=\lambda_{\mathrm{t}_{i}} / h$ to denote the normal and tangent components of the contact forces.

As shown in Fig. 2-5, the grippers start 0.006 m away from the surface of the sphere. They are first commanded to translate horizontally, touching the sphere at $t=0.03 \mathrm{~s}$. The grippers continue to squeeze the object until $t=0.08 \mathrm{~s}$. Accordingly, $c_{\mathrm{n}_{1}}$ grows from 0 N to 10 N , while $\phi_{1}$ stays at 0 . As shown in Fig. 2-6, this behavior is reproduced by both the LCP formulation and the CQDC dynamics (2.10).

The grippers are then commanded to pull the ball downward into the ground. Although initially resisted by friction, the downward commands eventually overcome friction and slipping between the ball and the fingers starts at $t=0.13 \mathrm{~s}$. In the LCP simulation, $c_{\mathrm{n}_{1}}, c_{\mathrm{f}_{1}}$ and $\phi_{1}$ remain constant despite the contact mode transition from sticking to sliding. In the CQDC simulation, however, as sliding starts at $t=0.13 \mathrm{~s}$, a small increase in $\phi_{1}$ is observed, which, as explained in Section. 2.3.4, is needed by sliding under Anitescu's friction constraints. In both the LCP and CQDC simulations, $c_{\mathrm{n}_{3}}$ grows with the friction $c_{\mathrm{f}_{1}}$ in order to keep the sphere in force balance.

The downward commands stop at $t=0.24 \mathrm{~s}$, and the contact mode switches back


Figure 2-5: Commanded and actual left finger positions for the system defined in Fig. $2-2$, as simulated by LCP and the proposed CQDC dynamics (2.10). Recall that $x_{l}$ and $u_{l}$ are respectively the commanded and actual $x$-coordinate of the left finger; $y_{l}$ and $u_{y}$ are respectively the commanded and actual $y$-coordinate of both fingers. The right finger is not shown, as the motions and forces of the left and right fingers are symmetric about the $y$-axis. The green dashed vertical lines indicate contact mode changes: separation to sticking at $t=0.03 \mathrm{~s}$; sticking to sliding at $t=0.13 \mathrm{~s}$; sliding to sticking at $t=0.23 \mathrm{~s}$.
to sticking from sliding. Once again, $c_{\mathrm{n}_{1}}, c_{\mathrm{f}_{1}}$ and $\phi_{1}$ remain constant in the LCP formulation. In contrast, the "boundary layer" created by sliding in the CQDC dynamics disappears as sliding stops.

## Iiwa Trajectory Tracking

Our first comparison between CQDC and SAP involves tracking a reference jointangle trajectory $q_{\mathrm{ref}}(\cdot)$ on a gravity-compensated and PD-controlled iiwa arm. The task only involves the smooth closed-loop dynamics of the arm (2.15), and does not involve any external contact. The starting and ending configurations of the reference trajectory are shown in Fig. 2-7a and Fig. 2-7b, respectively. We interpolate between the two configurations using a cubic spline, with maximum velocity in the middle and zero velocity at the two ends. We will denote the trajectory simulated by the CQDC dynamics as $q_{\mathrm{CQDC}}(\cdot)$, and SAP by $q_{\mathrm{SAP}}(\cdot)$.

To evaluate the performance of the two simulators, we first define the mean error


Figure 2-6: Contact force and signed distance at contact 1 and 3 of the gripper-sphere system defined in Fig. 2-2.


Figure 2-7: Starting (a) and ending (b) configurations of the reference joint-angle trajectory $q_{\text {ref }}(\cdot)$.
$\Delta(\cdot, \cdot)$ between two joint-angle trajectories $q_{1}(\cdot)$ and $q_{2}(\cdot)$ as

$$
\begin{equation*}
\Delta\left(q_{1}, q_{2}\right):=\frac{1}{T} \int_{0}^{T} d\left(q_{1}(t), q_{2}(t)\right) \mathrm{d} t \tag{2.17}
\end{equation*}
$$

where $T$ denotes the duration of the trajectories; $d(\cdot, \cdot)$ is the distance metric for the space of joint angles, which is simply the Euclidean 2-norm.

For different simulation step sizes $h$, the mean errors between simulated and reference trajectories, i.e. $\Delta\left(q_{\mathrm{CQDC} / \mathrm{SAP}}, q_{\mathrm{ref}}\right)$, are shown in Fig. 2-8. The error increases with $h$, but remains small even for very large $h(h=0.5 \mathrm{~s})$. In addition, as CQDC has no inertia, it tends to be slightly better at tracking the reference, especially as $h$ gets smaller.


Figure 2-8: Mean tracking error $\Delta\left(q_{\mathrm{CQDC} / \mathrm{SAP}}, q_{\mathrm{ref}}\right)$ for trajectories simulated using the CQDC dynamics or SAP at different simulation step sizes $h$.


Figure 2-9: KUKA IIWA robot stacking cubes. The red-and-grey cube has a quadrant colored grey to indicate its orientation. The red-and-grey cube starts off on the ground (a) and is placed on a stack of cubes and rotated by 90 degrees (b).

## Box Stacking with Iiwa

Our second comparison between CQDC and SAP focuses on a complex 3D system with many contacts, as shown in Fig. 2-9. The system has 10 cubes and 69 DOFs. The task is the robot picking up the red-and-grey cube and placing it on the stack. There is on average 56 contacts per time step during the execution of the task.

Firstly, although both CQDC and SAP are stable for large step sizes (at least $h=0.5 \mathrm{~s}$ ), CQDC is accurate for all step sizes, whereas the accuracy of SAP starts to degrade significantly as $h$ crosses a threshold. Similar to the iiwa trajectory tracking example, we compare trajectories of the red-and-grey cube using the mean error defined by (2.17). Instead of comparing against a reference trajectory, we define the ground truth trajectory of the box, $q_{\mathrm{GT}}^{\mathrm{u}}(\cdot)$, as the trajectory generated by SAP using $h=5 \times 10^{-5} \mathrm{~s}$. As shown in Fig. 2-10, the mean error of the box pose between CQDC and the ground truth, $\Delta\left(q_{\mathrm{CQDC}}^{\mathrm{u}}, q_{\mathrm{GT}}^{\mathrm{u}}\right)$, remains small up to $h=0.1 \mathrm{~s}$ and is still moderate at $h=0.5 \mathrm{~s}$, indicating that a large $h$ can be used for planning without impacting accuracy. In contrast, $\Delta\left(q_{\mathrm{SAP}}^{\mathrm{u}}, q_{\mathrm{GT}}^{\mathrm{u}}\right)$ becomes large for $h \geq 0.01 \mathrm{~s}$.

The reason behind the larger error of SAP at larger step sizes is shown in Fig. 2-11. With SAP, it appears that contacts become softer as $h$ increases, eventually leading to boxes sinking into each other and the ground.

Good simulation accuracy by the CQDC dynamics at large $h$ is also evident in the


Figure 2-10: Angular and translational mean error of the pose of the red-and-grey box, $\Delta\left(q_{\mathrm{CQDC} / \mathrm{SAP}}^{\mathrm{u}}, q_{\mathrm{GT}}^{\mathrm{u}}\right)$, at different step sizes $h$. As shown in Fig. 2-9, the box is picked up, rotated, and placed on the stack of cubes. The ground truth trajectory, $q_{\mathrm{GT}}^{\mathrm{u}}$, is defined as the trajectory generated by SAP with $h=5 \times 10^{-5} \mathrm{~s}$. In the computation of the mean error $\Delta(\cdot, \cdot)$, the distance function $d(\cdot, \cdot)$ is the Euclidean 2-norm for translation, and the absolute difference in angle for rotation.


Figure 2-11: Final configurations of the box stacking task shown in Fig. 2-9, simulated using SAP (a-c) and CQDC (d-f).
pose trajectory of the box, which is shown in Fig. 2-12. The translational trajectories are almost identical to the ground truth for all $h$ except $h=0.5 \mathrm{~s}$, but even at $h=0.5 \mathrm{~s}$ the error is just slightly more than 1 cm . There is also no significant deviation of the object orientation from the ground truth.

Lastly, the CQDC dynamics (2.6) scales well with the number of DOFs and contacts. The mean time of solving SOCP (2.6) using Gurobi [80] for the box stacking task in Fig. 2-9 is less than 10 ms on a Mac mini with Intel i7-8700B CPU and 64GB of RAM.


Figure 2-12: Trajectories of the red cube in Fig. 2-9, including the ground truth (GT) and the ones generated using the CQDC dynamics with different step sizes $h . x, y$ and $z$ are the cube's center of mass coordinates in world frame. Angle comes from the axisangle representation of the cube's orientation relative to world frame. Here the angle plot represents a rotation about the world $y$-axis by 90 degrees. The larger $y$ error at $h=0.5 \mathrm{~s}$ is caused by the gripper clipping the corner of the red-and-grey box when the gripper retreats from the box after placing it on the top of the stack. Due to the large $h$, the gripper's motion is interpolated between an upward segment and a horizontal segment. This error is visible by the difference of a few pixels between Fig. 2-11e and Fig. 2-11f.

### 2.4 Smoothing of Contact Dynamics

This section discusses how the proposed CQDC dynamics (Sec. 2.3) can be smoothed, so that we can efficiently make good local linear approximations that are useful for planning. We will begin with a summary of the randomized and analytic smoothing schemes proposed in [42, Section II] (Sec. 2.4.1). We will then discuss how the proposed quasi-static dynamics model readily supports both randomized (Sec. 2.4.2) and analytic smoothing (Sec. 2.4.3), and how the two schemes are related (Sec. 2.4.4).

### 2.4.1 Smoothing Schemes for Dynamical Systems

This subsection summarizes three different smoothing schemes for dynamical systems of the form $x_{+}=f(x, u)$ (2.1). The smoothing schemes are (i) analytic smoothing, (ii) first-order randomized smoothing and (iii) zeroth-order randomized smoothing.

Although the three schemes all compute the same quantity, they require different levels of access of the dynamics $f(\cdot, \cdot)$. Analytic smoothing utilizes the structure of $f(\cdot, \cdot)$ to efficiently compute a smoothed version of it, and therefore requires white-box access to $f(\cdot, \cdot)$. In contrast, randomized smoothing schemes achieve smoothing via sampling: first-order randomized smoothing needs black-box access to $f(\cdot, \cdot), \frac{\partial f}{\partial x}(\cdot, \cdot)$ and $\frac{\partial f}{\partial u}(\cdot, \cdot)$, whereas zeroth-order randomized smoothing only needs black-box access to $f(\cdot, \cdot)$.

We use $(\mathbf{A}(\bar{x}, \bar{u}), \mathbf{B}(\bar{x}, \bar{u}), c(\bar{x}, \bar{u}))$ to parameterize the linear system $\bar{f}$ that best describes $f$ around some nominal point $(\bar{x}, \bar{u})$,

$$
\begin{equation*}
\bar{f}(x, u)=\mathbf{A}(\bar{x}, \bar{u}) \underbrace{(x-\bar{x})}_{=: \delta x}+\mathbf{B}(\bar{x}, \bar{u}) \underbrace{(u-\bar{u})}_{=: \delta u}+c(\bar{x}, \bar{u}) . \tag{2.18}
\end{equation*}
$$

For dynamical systems, the Taylor expansion requires the system Jacobian,

$$
\begin{align*}
\mathbf{A}(\bar{x}, \bar{u}) & =\left.\frac{\partial}{\partial x} f(x, u)\right|_{x=\bar{x}, u=\bar{u}} \\
\mathbf{B}(\bar{x}, \bar{u}) & =\left.\frac{\partial}{\partial u} f(x, u)\right|_{x=\bar{x}, u=\bar{u}}  \tag{2.19}\\
c(\bar{x}, \bar{u}) & =f(\bar{x}, \bar{u}) .
\end{align*}
$$

The smooth surrogate of $f(x, u)$ can be defined as

$$
\begin{equation*}
f_{\rho}(x, u):=\mathbb{E}_{w \sim \rho}\left[f\left(x+w^{x}, u+w^{u}\right)\right], \tag{2.20}
\end{equation*}
$$

where $\rho$ is a probability distribution, $w^{x}$ is the component of $w$ that corresponds to $x$, and $w^{u}$ is defined similarly. The model parameters of the locally linear model $\bar{f}_{\rho}$ are given as follows:

$$
\begin{align*}
\mathbf{A}_{\rho}(\bar{x}, \bar{u}) & =\left.\frac{\partial}{\partial x} \mathbb{E}_{w \sim \rho}\left[f\left(x+w^{x}, u+w^{u}\right)\right]\right|_{x=\bar{x}, u=\bar{u}} \\
\mathbf{B}_{\rho}(\bar{x}, \bar{u}) & =\left.\frac{\partial}{\partial u} \mathbb{E}_{w \sim \rho}\left[f\left(x+w^{x}, u+w^{u}\right)\right]\right|_{x=\bar{x}, u=\bar{u}}  \tag{2.21}\\
c_{\rho}(\bar{x}, \bar{u}) & =\mathbb{E}_{w \sim \rho}\left[f\left(\bar{x}+w^{x}, \bar{u}+w^{u}\right)\right] .
\end{align*}
$$

In the remaining sections, we shorthand the notation and refer to the model parameters as a matrix instead of a function for compactness (e.g. $\mathbf{A}_{\rho}$ instead of $\mathbf{A}_{\rho}(\bar{x}, \bar{u})$ ).

## Analytic Smoothing

Analytic smoothing is done by explicit computation of $f_{\rho}$ as defined in (2.20). In general, it is difficult to compute $f_{\rho}$ for arbitrary functions $f$ and smoothing kernels $\rho$. In Sec. 2.4.3, we introduce a method for computing $f_{\rho}$ when $f$ is the CQDC dynamics.

## First-Order Randomized Smoothing

We use the following estimators for first-order randomized smoothing of dynamical systems,

$$
\begin{align*}
\mathbf{A}_{\rho} & \approx \frac{1}{N} \sum_{i=1}^{N}\left[\frac{\partial}{\partial x} f\left(\bar{x}+w_{i}^{x}, \bar{u}+w_{i}^{u}\right]\right. & w_{i} & \sim \rho  \tag{2.22a}\\
\mathbf{B}_{\rho} & \left.\approx \frac{1}{N} \sum_{i=1}^{N}\left[\frac{\partial}{\partial u} f\left(\bar{x}+w_{i}^{x}, \bar{u}+w_{i}^{u}\right)\right]\right] & w_{i} & \sim \rho  \tag{2.22b}\\
c_{\rho} & \left.\approx \frac{1}{N} \sum_{i=1}^{N}\left[f\left(\bar{x}+w_{i}^{x}, \bar{u}+w_{i}^{u}\right)\right]\right] & w_{i} & \sim \rho \tag{2.22c}
\end{align*}
$$

where $\partial f / \partial x$, and $\partial f / \partial u$ are Jacobians of the dynamics.

## Zeroth-Order Randomized Smoothing

We estimate the gradients using least-squares when $\rho$ is Gaussian:

$$
\begin{equation*}
\mathbf{A}_{\rho}, \mathbf{B}_{\rho}=\underset{\mathbf{A}, \mathbf{B}}{\operatorname{argmin}} \sum_{i=1}^{N}\left\|f\left(\bar{x}+w_{i}^{x}, \bar{u}+w_{i}^{u}\right)-\mathbf{A} w_{i}^{x}-\mathbf{B} w_{i}^{u}-c_{\rho}\right\|_{2}^{2} \tag{2.23}
\end{equation*}
$$

where $c_{\rho}$ is computed with (2.22c).

### 2.4.2 Randomized Smoothing of Contact Dynamics

We follow the method in Sec. 2.4.1 in order to perform randomized smoothing of the contact dynamics. For first-order randomized smoothing, we utilize (2.22) with access to gradients of the contact dynamics obtained in Sec.2.3.2. We similarly do zero-order randomized smoothing using (2.23). As the state consists of only the system configuration $q$ for quasi-static systems, we assume we compute a local model around some nominal state-input pair $(\bar{q}, \bar{u})$ from here onward.

However, there is a caveat in the randomized smoothing scheme: if the sampling distribution $\rho$ has infinite support, the sampled state $\bar{q}+w_{i}^{q}$ could violate the nonpenetration constraint for rigid-bodies, i.e. $\phi\left(\bar{q}+w_{i}^{q}\right)<0$ for some $i$.

Although reasoning about the dynamics $f(q, u)$ with an infeasible (penetrating) $q$ may seem ill-posed [81], we can define the dynamics from an infeasible $q$ as the projection of $q$ to the "nearest" point in the feasible (non-penetrating) set. The notion of nearest can be defined in terms of the work required to move the system configuration by $\delta q$, which is precisely the quadratic cost (2.6a) (divided by the step size $h$ ). This projection problem can then be written as

$$
\begin{align*}
\min _{\delta q} . & \frac{1}{2} \delta q^{\top} \mathbf{Q} \delta q+b^{\top} \delta q, \text { subject to }  \tag{2.24a}\\
& \phi_{i}(q+\delta q) \geq 0, i \in\left\{1 \ldots n_{\mathrm{c}}\right\} \tag{2.24b}
\end{align*}
$$

where ( 2.24 b ) is the non-linear non-penetration constraint.
While the projection in (2.24) is difficult to solve in general, we can linearize the constraint $(2.24 b)$ in order to locally approximate the problem as a QP. When the
constraint is linearized, the problem remarkably becomes equivalent to the frictionless special case of the CQDC dynamics (2.6). In other words, projection is simply another interpretation of what the CQDC dynamics does within the penetrating regime, and no other explicit treatment is required for projection other than the evaluation of CQDC dynamics. In practice, due to the local nature of linearization, we use a low-variance distribution $\rho$ to sample states, though higher variance can be used for inputs.

When samples within the penetrating regime are projected onto the boundary of the feasible set and then averaged, the expected value of such a distribution creates a stochastic force field that pushes the object away from feasible set's boundary. We illustrate this phenomenon through a simple example.

Example 1. (The Stochastic Force Field) Consider the dynamics of an unactuated 1D block with a wall occupying $q \leq 0$ (Fig. 2-13a), such that the physical dynamics is identity if the block is in a non-penetrating configuration, $f(q)=q$ if $q \geq 0$. The dynamics within the penetrating regime is not well-defined physically; yet, applying the quasi-dynamic equations of motions (2.24) to this system gives

$$
\begin{align*}
\min _{\delta q} & \frac{1}{2} m(\delta q)^{2}, \text { subject to }  \tag{2.25a}\\
& q+\delta q \geq 0 \tag{2.25b}
\end{align*}
$$

which has the following solution,

$$
f(q)=q+\delta q= \begin{cases}q & \text { if } q \geq 0 \text { (no penetration) }  \tag{2.26}\\ 0 & \text { else (penetration) }\end{cases}
$$

One interpretation of (2.26) is that configurations inside the wall gets projected onto the wall. By taking an average according to such dynamics, the expectation pushes the box away from the wall as illustrated in Fig. 2-13, which creates a stochastic force field. Note that with this interpretation, the gradients are also well-defined within the penetrating regime - an infinitesimal change of position in the penetrating


Figure 2-13: Figure for Example 1, where quasi-dynamics of motion is interpreted as a projection operator. (a) Illustration of the system. (b) Distribution of $q+w$ (green) and $f(q+w)$ (pink). Note that the samples for which $q+w_{i}<0$ have been projected onto the surface into a delta function, and the expectation of the pink distribution lies on the right side of $q$, creating a stochastic force field effect. (c) CQDC dynamics and its randomized smoothed version. (d) Gradients of the CQDC dynamics obtained with firstorder randomized smoothing.
configuration does not have any effect on the location of projection in this example, thus $\partial f(q) / \partial q=0$ if $q<0$. For more complex geometries, the location of projection changes due to changes in the surface, which we connect back to the presence of the $\partial \mathbf{J} / \partial q$ and $\partial \phi / \partial q$ terms in (2.9a).

### 2.4.3 A Smoothed Contact Dynamics Model

Both randomized smoothing schemes only involve repeatedly solving the CQDC dynamics (2.6) and/or its gradient (2.8). In contrast, analytic smoothing (Sec.2.4.1) of the CQDC dynamics (2.6) is not as straightforward, since the solution to an optimization problem does not give easy access to explicit forms from which we can analytically design smooth surrogates.

To perform analytic smoothing, we can convert the hard constraints in a convex program into costs using a penalty method. Specifically, we convert the constrained program (2.6) into an unconstrained convex program using the log-barrier function,
a common technique in the interior-point method for convex conic programs, with weight $\kappa$ :

$$
\begin{equation*}
\min _{\delta q} \frac{1}{2} \delta q^{\top} \mathbf{Q} \delta q+b^{\top} \delta q-\frac{1}{\kappa} \sum_{i=1}^{n_{\mathrm{c}}} \log \left[\frac{\left(\mathbf{J}_{\mathrm{n}_{i}} \delta q+\phi_{i}\right)^{2}}{\mu_{i}^{2}}-\left(\mathbf{J}_{\mathrm{t}_{i}} \delta q\right)^{\top} \mathbf{J}_{\mathrm{t}_{i}} \delta q\right], \tag{2.27}
\end{equation*}
$$

whose solution converges to the solution of (2.6) as $\kappa \rightarrow \infty$ [49, $\S 11.3$ and §11.6].

From a contact mechanics perspective, the log-barrier formulation is motivated by relaxing the strict complementary slackness condition of the CQDC dynamics to allow contact forces at a distance. In fact, the optimality conditions of (2.27) are almost identical to the KKT conditions of (2.6). The only difference is that complementary slackness (2.5c) is relaxed from $v_{i}^{\top} \lambda_{i}=0$ to $v_{i}^{\top} \lambda_{i}=\kappa^{-1}$ by the log barrier formulation ( $v_{i}$ is defined in (2.5a)).

The log-barrier term in (2.27) can be interpreted as the potential of a force field whose strength is inversely proportional to the distance to the boundary of the constraint [49, p.567]. For moderate values of $\kappa$, constraints can exert forces even though they are not active, achieving a smoothing effect similar to the "force-at-a-distance" relaxation of complementarity constraints, which are commonly used in planning through contact methods such as $[17,50]$. We will further illustrate this similarity in Example 2.

From $\delta q^{\star}$, the solution to the smoothed dynamics (2.27), we can directly compute the smoothed gradient $\mathbf{A}_{\rho}$ and $\mathbf{B}_{\rho}$ in the same way as (2.8). Once again, the derivatives of $\delta q^{\star}$ with respect to $q$ and $u$ are computed by applying the implicit function theorem to the optimality condition of (2.27), which only consists of the stationarity condition due to the absence of conic constraints.

To solve (2.27), we implemented an in-house solver using Newton's method [49, §9.5], and find that our out-of-the-textbook implementation works robustly and reliably for all numerical experiments in Sec. 3.2 and 3.5.

### 2.4.4 The Smooth Contact Model as "Analytic Smoothing"

In Sec. 2.4.1, we saw that randomized and analytic smoothing can be interpreted as different methods of computing an equivalent quantity. Here, we show with examples that (i) for simple systems, we can derive the sampling distribution $\rho$ needed to obtain the smoothing given by the log-barrier-based analytic smoothing scheme; (ii) for more complex systems, randomized smoothing schemes using a Gaussian $\rho$ and the log-barrier-based analytic smoothing scheme result in qualitatively-similar smoothed dynamics.

Example 2. (Equivalence of Smoothing Schemes) We start with the 1D frictionless system in Fig. 2-14a, whose dynamics is an instance of the frictionless CQDC dynamics (2.6):

$$
\begin{align*}
\min _{\delta q^{\mathrm{a}}} & \frac{1}{2} h k_{a}\left(\delta q^{\mathrm{a}}\right)^{2}-h k_{\mathrm{a}}\left(u-q^{\mathrm{a}}\right) \delta q^{\mathrm{a}}, \text { subject to }  \tag{2.28}\\
& q^{\mathrm{a}}+\delta q^{\mathrm{a}} \geq 0
\end{align*}
$$

The KKT conditions of (2.28) are also the equations of motion of the system:

$$
\begin{align*}
h k_{\mathrm{a}}\left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}-u\right) & =\lambda,  \tag{2.29a}\\
0 \leq\left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}\right) & \perp \lambda \geq 0, \tag{2.29b}
\end{align*}
$$

which has the explicit solution

$$
q_{+}^{\mathrm{a}}=q^{\mathrm{a}}+\delta q^{\mathrm{a}}= \begin{cases}u & \text { if } u \geq 0(\text { no contact })  \tag{2.30}\\ 0 & \text { else (contact) }\end{cases}
$$

Also as an instance of $(2.27)$, (2.28) can be smoothed analytically by converting the constraint into a cost using the log-barrier function:

$$
\begin{equation*}
\min _{\delta q^{\mathrm{a}}} . \frac{1}{2} h k_{a}\left(\delta q^{\mathrm{a}}\right)^{2}-h k_{\mathrm{a}}\left(u-q^{\mathrm{a}}\right) \delta q^{\mathrm{a}}-\frac{1}{\kappa} \log \left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}\right), \tag{2.31}
\end{equation*}
$$

whose optimality condition is obtained by setting the gradient of the smoothed cost
to 0 , yielding the equations of motion of the smoothed system:

$$
\begin{equation*}
h k_{\mathrm{a}}\left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}-u\right)=\frac{1}{\kappa}\left(\frac{1}{q^{\mathrm{a}}+\delta q^{\mathrm{a}}}\right) . \tag{2.32}
\end{equation*}
$$

The right-hand side of (2.32) can be interpreted as an impulse whose magnitude is inversely proportional to the distance to the wall. Calling this impulse $\beta$, we note that

$$
\begin{equation*}
\beta\left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}\right)=\kappa^{-1} \tag{2.33}
\end{equation*}
$$

which is analogous to the common bilinear relaxation to the complementarity constraint for contact $[17,50]^{2}$.

The solution to (2.32) is given by

$$
\begin{equation*}
q_{+}^{\mathrm{a}}=\frac{1}{2}\left(u+\sqrt{u^{2}+\frac{4}{\kappa h^{2} k_{\mathrm{a}}^{2}}}\right) \tag{2.34}
\end{equation*}
$$

which is equivalent to randomized smoothing of the original dynamics under the following elliptical distribution

$$
\begin{equation*}
\rho(w)=\sqrt{\frac{4 \sigma}{\left(w^{\top} \sigma w+4\right)^{3}}}, \tag{2.35}
\end{equation*}
$$

where $\sigma:=h k \kappa$.
Conversely, we hypothesize the existence of a barrier function that corresponds to the Gaussian density, though its form is prohibitive for analysis. In addition, for more complex examples involving friction, such as the system in Fig. 2-14b, obtaining an analytic expression for $\rho$ when the contact dynamics is smoothed by log barrier (2.27) is difficult. Instead, we numerically illustrate the performance of the two smoothing schemes in Fig. 2-14e. More details about this example can be found in [41].

Finally, we note that with a different smoothing scheme which relaxes complementary slackness, Howell et al. showed a similar trend for the smoothed dynamics as

[^2]

Figure 2-14: (a) A system consisting of an actuated cart constrained to slide on a frictionless surface, and a wall occupying $q^{\mathrm{a}} \leq 0$. The actuator has stiffness $k_{\mathrm{a}}$. (b) A system consisting of an un-actuated cart constrained to slide on a frictionless surface, and a ball actuated along both the $x$ and $y$ axes. The ball can touch the top surface of the cart with a frictional contact. (c) Randomized and analytic smoothing of (a). Randomized smoothing, shown in green, is done with a Gaussian kernel with different variances $\sigma$. Analytic smoothing, shown in magenta, is done with different log-barrier weights $\kappa$. (d) Density functions of the Gaussian kernels (green) and the elliptical distributions used for analytic smoothing (magenta). (e) Randomized and analytic smoothing of (b). We plot $q_{+}^{\mathrm{u}}$ against $u_{x}$ for a fixed $u_{y}$ that is inside the cart. The linear region in the plot corresponds to sticking contact, and the flat regions to sliding.
the relaxation is tightened [50, Fig. 7]. We believe this corroborates the equivalence between analytic and randomized smoothing schemes.

## Chapter 3

## Contact-Rich Manipulation Planning using Quasi-static Models

### 3.1 Introduction

In Chapter 2, we have developed the CQDC dynamics and prposed computational methods to obtain local linear models of its smooth surrogate. These tools are not specific to any algorithm: they can improve the performance of most iterative planning algorithms that rely on local approximations.

We start the section on contact-rich planning with iterative MPC (iMPC) [41], a variant of trajectory optimization, that uses the CQDC dynamics to enforce contact dynamics constraints and compute smoothed gradients. In addition to demonstrating the utility of the proposed CQDC dynamics, we also introduce iMPC as a handy refinement tool for trajectories generated by a sampling-based motion planner (SBMP). Such trajectories are well-known for making inefficient and meandering "dances" before arriving at the goal.

However, we have argued that the success of contact-rich planning depends on both smoothing and global search. Planning based on local optimization, including trajectory optimization with smoothed contact dynamics, generally results in nonconvex optimization problems where the quality of different local minima can be a make-or-break factor. Solving such problems requires non-trivial initialization and
cost tuning [7], both of which can be highly problem specific and notoriously hard to debug. This motivates us to search further for a more global approach.

In robotics, SBMP algorithms such as the Rapidly-Exploring Random Tree (RRT) [5] are widely used for global search problems, including those with kinodynamic constraints [82]. However, the success of such planners has rarely extended to contactrich settings. We find that while optimization-based methods for planning through contact have employed smoothing schemes, all existing SBMP methods for contact planning explicitly consider modes instead of smoothing them [34, 35, 48, 83, 84], as SBMP methods do not inherently require local characterizations of dynamics (i.e. gradients). Yet, previous works have shown that such local models are highly relevant for designing more efficient distance metrics during the nearest neighbor queries that respects dynamic reachability [35, 85, 86].

In this chpater, we fill in this gap by combining contact mode abstraction via smoothing, and the global search capabilities of RRT. We enable RRT to explore efficiently through contact constraints by utilizing a novel distance metric based on the local smoothed linearizations of CQDC dynamics (Sec.3.3). In addition, we further propose an efficient extension step by computing actions that expand the tree to new parts of the state space using the local linearizations (Sec.3.4). With a variety of contact-rich tasks inspired by [87] that involve both intrinsic and extrinsic dexterity [88], we show that combining smoothing with RRT achieves tractable global motion planning for highly contact-rich manipulation (Sec.3.5); and that the planned trajectories can transfer to high-fidelity second-order simulations or even robot hardware. To the best of our knowledge, our work appears to be the first to successfully combine SBMP with contact mode smoothing.

### 3.2 Trajectory Optimization through Contact

In this section, we demonstrate the efficacy of smoothed CQDC dynamics on trajectory optimization for systems with contacts. Although smoothing of hard contact constraints has been widely utilized to improve convergence [17, 50, 89], existing
methods still struggle with complex problems such as dexterous manipulation. In particular, the quality of solutions can be very sensitive to initial guesses [7]. In this context, we show that a variant of trajectory optimization, with the help of smoothed CQDC dynamics and only a trivial initial guess, can perform well even on dexterous manipulation tasks.

### 3.2.1 Iterative MPC with Smoothing

The variant of trajectory optimization algorithm used in this section, which we call iterative MPC (iMPC), is an iLQR-inspired algorithm proposed in [41]. We briefly summarize iMPC here for completeness, and illustrate how smoothing can be easily integrated into iMPC.

Consider the problem of finding an optimal sequence of inputs to track some desired state trajectory $\left\{x_{t}^{d}\right\}_{t=0}^{T}$. We need an initial guess for the nominal input trajectory $\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}$, from which the nominal state trajectory $\left\{\bar{x}_{t}\right\}_{t=0}^{T}$ can be obtained by rolling out $\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}$ from the initial state $x_{0}$. For every time $t$ (i.e. in a time-varying manner), we can create a locally linear model that approximates the dynamics, with model parameters $\left\{\mathbf{A}_{t}, \mathbf{B}_{t}, c_{t}\right\}_{t=0}^{T-1}$ (2.19). Then, finding the optimal $\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}$, subject to the locally linear model of the dynamics, can be written as a QP. We present the MPC variant of this problem that receives the initial state $\bar{x}_{j}$ at time $t=j$ and computes the optimal action for the remaining time steps,

$$
\begin{gather*}
\operatorname{MPC}\left(\bar{x}_{j}\right)=u_{j}^{\star}, \text { where }  \tag{3.1a}\\
\left\{x_{t}^{\star}\right\}_{t=j}^{T},\left\{u_{t}^{\star}\right\}_{t=j}^{T-1}=\underset{\left\{x_{t}\right\}_{t=j}^{T},\left\{u_{t}\right\}_{t=j}^{T-1}}{\operatorname{argmin}}\left\|x_{T}-x_{T}^{d}\right\|_{\mathbf{Q}_{T}}^{2}+\sum_{t=j}^{T-1}\left(\left\|x_{t}-x_{t}^{d}\right\|_{\mathbf{Q}_{t}}^{2}+\left\|u_{t}\right\|_{\mathbf{R}_{t}}^{2}\right)  \tag{3.1b}\\
\text { s.t. } x_{t+1}=\mathbf{A}_{t}\left(x_{t}-\bar{x}_{t}\right)+\mathbf{B}_{t}\left(u_{t}-\bar{u}_{t}\right)+c_{t},  \tag{3.1c}\\
\mathbf{C}_{t}^{x} x_{t} \leq d_{t}^{x}, \mathbf{C}_{t}^{u} u_{t} \leq d_{t}^{u}, \forall t \in\{j \cdots T-1\},  \tag{3.1d}\\
x_{j}=\bar{x}_{j} . \tag{3.1e}
\end{gather*}
$$

Here, $\left\{\mathbf{Q}_{t}, \mathbf{R}_{t}\right\}$ are the quadratic weights for state and input, respectively; $\mathbf{Q}_{T}$ is
the weight on the terminal state; $\left\{\mathbf{C}_{t}^{x}, d_{t}^{x}\right\}$ and $\left\{\mathbf{C}_{t}^{u}, d_{t}^{u}\right\}$ are inequality parameters on the state and input, respectively. The linear constraints (3.1e) can enforce, for instance, joint and actuation limits.

The iMPC algorithm is summarized in Alg. 1. In every outer iteration (body of the while loop starting at Line 4), iMPC solves truncated versions of (3.1) for $T-1$ times. Specifically, at inner iteration $j$ (body of the for loop starting at Line 6), we solve MPC (3.1) for the sub-problem starting at $t=j$ (Line 7), and apply $u_{j}^{\star}$ from the solution to update $\bar{x}_{j+1}$ (Line 8). In every inner iteration, we also enforce a trust region by using (3.1e) to constrain $x_{t}$ and $u_{t}$ to stay close to $\bar{x}_{t}$ and $\bar{u}_{t}$, respectively.

```
Algorithm 1: iMPC
    Input: Initial state \(x_{0}\), input trajectory guess \(\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}\);
    Output: Optimized input trajectory \(\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}\);
    \(\left\{\bar{x}_{t}\right\}_{t=0}^{T} \leftarrow\) Rollout \(f\) from \(x_{0}\) with \(\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}\);
    while not converged do
        Compute system matrices \(\left\{\mathbf{A}_{t}, \mathbf{B}_{t}, c_{t}\right\}_{t=0}^{T-1}\);
        for \(0 \leq j<T\) do
            \(\bar{u}_{j} \leftarrow \operatorname{MPC}\left(\bar{x}_{j}\right) ;\)
                \(\bar{x}_{j+1} \leftarrow f\left(\bar{x}_{j}, \bar{u}_{j}\right) ;\)
    return \(\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}\)
```

To apply smoothing to iMPC, we substitute the linearizations of smooth surrogates $\left\{\mathbf{A}_{t, \rho}, \mathbf{B}_{t, \rho}, c_{t, \rho}\right\}_{t=0}^{T-1}(2.21)$ for the first-order Taylor expansions $\left\{\mathbf{A}_{t}, \mathbf{B}_{t}, c_{t}\right\}_{t=0}^{T-1}$ (2.19). After every outer iteration, we also reduce the variance of $\rho$ (this can be done for analytic smoothing by increasing the $\log$ barrier weight $\kappa$ ), allowing the smooth surrogates $f_{\rho}$ to converge to the true CQDC dynamics $f$.

### 3.2.2 Experiment Setup

## Systems Description

We test iMPC with different smoothing schemes on two planar systems from [41] and a 3D system for in-hand rotation. We describe our systems below, and their visualization can be seen in Fig. 3-6. The three tuple after the name of each system


Figure 3-1: Performance of iMPC with different smoothing schemes: analytic, randomized (first-order), randomized zero-order, and exact (no smoothing). For each method, the solid line represents the mean over five runs, and the shaded region represents the standard deviation.
indicates $\left(n_{\mathrm{u}}, n_{\mathrm{a}}, n_{\mathrm{cg}}\right)$, where $n_{\mathrm{u}}$ is the number of unactuated DOFs, $n_{\mathrm{a}}$ the number of actuated DOFs, and $n_{\text {cg }}$ the number of collision geometries.

1. Planar Pushing, (3,2,2). A classical example of nonprehensile manipulation [53]. The goal is specified as some 2D configuration of the box.
2. Planar Hand Reorientation, $(\mathbf{3 , 4 , 1 3})$. We use a planar hand with two fingers, each with two DOFs. The goal is to change the position and orientation of the ball in a 2 D plane.
3. Allegro In-Hand Rotation, $(\mathbf{6}, \mathbf{1 6}, \mathbf{2 0})$. 3D In-hand rotation of the ball with the full model of the allegro hand [90]. The goal is specified as a rotated configuration of the ball.

## Initialization

As mentioned in Sec. 3.2.1, given an initial state $x_{0}$, we need to initialize the nominal input trajectory $\left\{\bar{u}_{t}\right\}_{t=0}^{T-1}$, where $\bar{u}_{t}$ is the commanded positions of the robots at step $t$ under the CQDC dynamics. Empirically, we find that good convergence can be achieved with a constant initialization, i.e. $\bar{u}_{0}=\bar{u}_{1}=\ldots \bar{u}_{T-1}$. Although this initialization is still prone to local minima, it is surprisingly effective when the solution can be found without global search.

For the numerical experiments in this section, we need a $\bar{u}_{0}$ that makes contact with the object. Otherwise the baseline which does not use smoothing would have zero gradients and make no progress at all.

In contrast, for iMPC with smoothing, it is sufficient to set $\bar{u}_{0}=q_{0}^{\mathrm{a}}$, as long as $q_{0}^{\mathrm{a}}$ is not "too far away" from making contact with the objects (the reason is explained in Example 4). In many practical problems, the object's initial configuration $q_{0}^{u}$ is fixed, but we are free to choose the initial robot configuration $q_{0}^{\mathrm{a}}$. In this case, we can simply calculate a $q_{0}^{\text {a }}$ that is "close" to making contact using, for example, methods that compute grasps [91].

## Hardware \& Implementation Details

The numerical experiments are run on a desktop with one AMD Threadripper 2950 CPU (16 cores, 32 threads) and 32GB of RAM. The code for iMPC using different smoothing schemes is identical except for the computation of the linearizations. For analytic smoothing and the baseline, we solve respectively the smoothed (2.27) and original (2.6) dynamics once and then apply the chain rule to get the linearization. For first-order randomized smoothing, we solve the original dynamics (2.6) and apply the chain rule for 100 samples $(N=100)$, which is parallelized on all available threads, and then average the gradients of the samples. Zeroth-order randomized smoothing simply requires parallel evaluation of the dynamics.

| Problem | PlanarPushing |  | PlanarHand |  | AllegroHand |  |
| :---: | ---: | ---: | ---: | ---: | ---: | ---: |
| Method | Cost | Time(s) | Cost | Time(s) | Cost | Time(s) |
| A. | 11.74 | 2.17 | 26.55 | 5.20 | 5.78 | 19.59 |
| RF. | $\mathbf{1 1 . 7 3}$ | 4.64 | $\mathbf{1 7 . 8 7}$ | 23.09 | 8.42 | 40.07 |
| RZ. | 12.86 | 4.61 | 18.29 | 11.93 | 28.21 | 34.05 |
| E. | 31.64 | 1.88 | 18.49 | 5.91 | 44.68 | 12.92 |

Table 3.1: Minimum cost and running time achieved by different methods. All methods are ran for 10 iterations across 5 trials. The methods names are abbreviations from the legend of Fig. 3-1.

(a) Initial configurations and goals. Each system is shown in its initial configuration $q_{0}$. The thicker frame denotes the goal while the thinner frame denotes the initial configuration of the object.

(b) Final object configuration achieved by the best runs within each of the four methods. Pink shaded denotes the goal configuration for the first two examples, while the goal configuration in the last example is marked by the pink line protruding out of the object. Colors correspond to the plots in Fig. 3-1.

Figure 3-2: Tasks and results for the trajectory optimization case study.

### 3.2.3 Results \& Discussion

In Fig. 3-1, we plot the performance of iMPC with a baseline that does not use smoothing, and the three different smoothing schemes in Sec. 2.4.1, namely analytic, randomized first-order, and randomized zeroth-order. We also summarize the running time of each method, as well as the minimum cost achieved across the iterations, in Table 3.1. Illustrations of the tasks and the results achieved by different methods are shown in Fig. 3-2. We interpret the results and discuss the relevant findings in this section.

## Exact vs. Smoothing

For PlanarPushing and AllegroRotation, the various smoothing schemes achieve much lower costs than using exact gradients. However, for PlanarHand, using the exact linearization is performant as well. This difference may be explained by the observation that the planar hand example does not go through many mode changes, while the planar pusher and the allegro hand require several mode changes to converge to the locally optimal trajectory.

## Analytic vs. Randomized Smoothing

Comparing the performance of the three smoothing schemes, the analytic and the first-order randomized smoothing perform similarly, while the zeroth-order version does not perform as well. We believe the cause lies in the high variance characteristic of the zeroth-order estimator in higher dimensions.

## Running (wall-clock) Time

While analytic smoothing only requires one evaluation of the smoothed dynamics (2.27) in order to compute $\left(\mathbf{A}_{\rho}, \mathbf{B}_{\rho}, c_{\rho}\right)$, randomized smoothing requires taking $N$ samples and averaging them, which costs $N$ times more compute-time. After parallelization, we expect randomized smoothing to be roughly $N / \xi$ times slower than analytic smoothing where $\xi$ is the number of threads. Indeed, with $N=100$ and $\xi=32$, our results show that randomized smoothing is 2 to 3 times slower than analytic smoothing.

### 3.3 Local Mahalanobis Metric for RRT

While trajectory optimization can find trajectories reaching goals that are close to the initial configuration, it is highly prone to local minima when the goal is further away (e.g. moving the box back in PlanarPushing, rotating the ball by 180 degrees in PlanarHand, AllegroHand). To solve these tasks, the planning algorithm needs to be
more global. When faced with such problems, the RRT algorithm [5] has proven to be a classical and effective method for global planning.

However, extending RRT to dynamical systems (i.e. kinodynamic RRT) has been difficult, as a distance metric between two states is hard to define. In [85], it was argued that a good distance metric for RRT would need to explicitly consider dynamic reachability in order to efficiently grow the tree. The authors further proposed Reachability-Guided RRT (RG-RRT), which had system-specific reachability metrics that was shown to be effective for smooth systems. To alleviate the limitation of being system-specific, later works have considered building such metrics based on local characteristics of the dynamics such as local linearizations [35, 86].

However, when the dynamics involves contact, such local linearizations are no longer informative, and existing approaches often tackle dynamic reachability by explicitly considering contact modes $[34,83]$. This has led to planners that scale poorly with the number of contacts. In contrast, we propose to handle the challenges brought about by contact with smoothing. We show that when combined with smoothing, the locally linear model can be used to construct an informative distance metric that is consistent with notions of reachability.

### 3.3.1 The Local Mahalanobis Metric

Consider the following problem: given the current configuration $\bar{q}$, and some queried configuration $q$, how can we formulate a distance metric $d(q ; \bar{q})$ that is consistent with reachability characteristics of the system? We propose to utilize the locally linear model around the nominal configuration $\bar{q}$, that characterizes the local response of the next system configuration $q_{+}$with respect to the movement of the actuated configurations $u$. This local model can be written as

$$
\begin{equation*}
q_{+}=\mathbf{B}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) \underbrace{\left(u-\bar{q}^{\mathrm{a}}\right)}_{\delta u}+c\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) \tag{3.2}
\end{equation*}
$$

where the notation is consistent with the CQDC dynamics formulation in Sec.2.3.1; the input $u$ is the position command to the system), and $\mathbf{B}, c$ are defined as (2.19).


Figure 3-3: (a) Two different sublevel sets $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$, represented as ellipsoids, shown in the space of $q^{\mathrm{u}}$, with $\varepsilon=1$, and $\gamma=10^{-6}$. The ellipsoid centers are shifted to the origin for easy comparison. Red ellipsoid: $\mathcal{R}_{\rho, \varepsilon, \gamma}^{u}$ for the system configuration in Fig. 3-3b; blue ellipsoid: $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$ for the configuration in Fig. 3 -3c. Points $b_{1}, c_{1}$ are where ellipsoids' major axes intersect their boundaries. Points $b_{2}, c_{2}$ are points along the minor axes of the ellipsoids, and satisfy $\left\|b_{1}\right\|=\left\|b_{2}\right\|$ and $\left\|c_{1}\right\|=\left\|c_{2}\right\|$, where the norm is based on the standard Euclidean metric. (b) The solid robots and objects represent the $\bar{q}$ at which the red $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$ in (a) is computed. The straight line on the puck indicates its orientation. The dashed dark red puck corresponds to the configuration $b_{1}$, and pink to $b_{2}$. Note that $b_{1}$ is easier to each than $b_{2}$. (c) Similar to Fig. 3-3b, the dashed dark blue puck correspond to $c_{1}$, and light blue to $c_{2}$. It is also easier to reach $c_{1}$ than $c_{2}$. (d) The volume of $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$ shrinks as the fingers get further away from the puck. The ellipsoids on the right are color-coded to match the robot configurations on the left. Note that the blue ellipsoid is barely visible.

We note that the contribution of $\mathbf{A}$ term is zero since $\delta x=0$.
Given such a local characterization, and a queried state $q$, we define the Mahalanobis metric as follows.

Definition 1. Local Mahalanobis Metric. Given a nominal configuration $\bar{q}$, and queried configuration $q$, we define the Mahalanobis distance $d_{\gamma}$ of $q$ from $\bar{q}$ as follows:

$$
\begin{align*}
d_{\gamma}(q ; \bar{q}) & :=\|q-\mu\|_{\boldsymbol{\Sigma}_{\gamma}^{-1}}=\frac{1}{2}(q-\mu)^{\top} \boldsymbol{\Sigma}_{\gamma}^{-1}(q-\mu)  \tag{3.3}\\
\boldsymbol{\Sigma}_{\gamma} & :=\mathbf{B}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) \mathbf{B}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right)^{\top}+\gamma \mathbf{I}_{n}, \mu:=c\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) .
\end{align*}
$$

The regularization $\gamma \mathbf{I}_{n}$ is added to ensure that $\boldsymbol{\Sigma}_{\gamma}$ is positive definite and the inverse $\boldsymbol{\Sigma}_{\gamma}^{-1}$ is well-defined. Note that the $\varepsilon$-sublevel set of this metric $d_{\gamma}(q ; \bar{q})$, which we denote by $\mathcal{R}_{\varepsilon, \gamma}(\bar{q})$, describes an ellipsoid that is centered at $\mu$ and has a shape matrix $\boldsymbol{\Sigma}_{\gamma}^{-1}$. We further motivate our construction of the metric by noting that this ellipsoid can be alternatively characterized (equivalent up to a regularization) by the following set,

$$
\begin{equation*}
\mathcal{R}_{\varepsilon}(\bar{q}):=\left\{\mathbf{B}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) \delta u+c\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) \mid\|\delta u\| \leq \varepsilon\right\} . \tag{3.4}
\end{equation*}
$$

This equivalence relation is exact when $\mathbf{B}$ has full row rank (i.e. the system is one-step controllable) and $\gamma=0$. On the other hand, if $\mathbf{B}$ loses rank, one of the principle axis of $\mathcal{R}_{\varepsilon}$ has a length of zero and the set becomes degenerate.

Finally, note that when $q-\mu \notin \operatorname{Range}(\mathbf{B})$, i.e. there is no actuation $u$ that can take the state $\bar{q}$ to the queried state $q$, the distance $d_{\gamma}(q ; \bar{q})$ is a large number dominated by the inverse of the regularization term $\gamma^{-1}$, which is consistent with the intuition that states that are harder to reach are further away.

### 3.3.2 Metric on Smoothed Dynamics and Unactuated Objects

As explained in the previous sections, the local model constructed using $\mathbf{B}$ may not be a very informative one for non-smooth systems with contact. In light of the various smoothing schemes introduced in Sec.2.4.1 to alleviate this issue, we propose a metric by utilizing the linearization of the smooth surrogate $\left(\mathbf{B}_{\rho}, c_{\rho}\right)$, as opposed to those of the original contact dynamics, $(\mathbf{B}, c)$.

Furthermore, for systems where robots interact with unactuated objects through contact, we focus on the reachability of the objects, as the robots are actuated and can easily move to a desired configuration without contact. We combine smoothing and the object-centric reachability in the following variant of the Mahalanobis metric $d_{\rho, \gamma}^{\mathrm{u}}$,

$$
\begin{align*}
d_{\rho, \gamma}^{\mathrm{u}}(q ; \bar{q}) & :=\left\|q^{\mathrm{u}}-\mu_{\rho}^{\mathrm{u}}\right\|_{\boldsymbol{\Sigma}_{\rho, \gamma}^{\mathrm{u}}}-1 \\
\boldsymbol{\Sigma}_{\rho, \gamma}^{\mathrm{u}} & :=\mathbf{B}_{\rho}^{\mathrm{u}}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) \mathbf{B}_{\rho}^{\mathrm{u}}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right)^{\top}+\gamma \mathbf{I}_{n_{\mathrm{u}}},  \tag{3.5}\\
\mu_{\rho}^{\mathrm{u}} & :=c_{\rho}^{\mathrm{u}}\left(\bar{q}, \bar{q}^{\mathrm{a}}\right) .
\end{align*}
$$

where $\mathbf{B}_{\rho}^{\mathrm{u}}$ is formed by the rows of $\mathbf{B}_{\rho}$ corresponding to the unactuated DOFs, and $c_{\rho}^{\mathrm{u}}$ is defined similarly. Finally, we define $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}(\bar{q})$ as the $\varepsilon$-sublevel set of $d_{\rho, \gamma}^{\mathrm{u}}(q ; \bar{q})$.

In the rest of this section, we give several examples that provide intuition into the local Mahalanobis metric $d_{\rho, \gamma}^{\mathrm{u}}$ and its sublevel set $\mathcal{R}_{\rho, \varepsilon, \gamma}^{\mathrm{u}}$.

Example 3. (Understanding B for Planar Systems) As shown in Sec. 2.3.4, 3.2.2), the CQDC dynamics (2.6) simplifies to a QP (2.10) for planar systems. For
reference, the QP (2.10) is reproduced below:

$$
\begin{align*}
\min _{\delta q} . & \frac{1}{2} \delta q^{\top} \mathbf{Q} \delta q+b^{\top} \delta q, \text { subject to }  \tag{3.6a}\\
& \left(\mathbf{J}_{\mathrm{n}_{i}}+\mu_{i} \mathbf{J}_{\mathrm{t}_{i}}\right) \delta q+\phi_{i} \geq 0, i \in\left\{1 \ldots n_{\mathrm{c}}\right\}  \tag{3.6b}\\
& \left(\mathbf{J}_{\mathrm{n}_{i}}-\mu_{i} \mathbf{J}_{\mathrm{t}_{i}}\right) \delta q+\phi_{i} \geq 0, i \in\left\{1 \ldots n_{\mathrm{c}}\right\} \tag{3.6c}
\end{align*}
$$

Recall that the contact Jacobian $\mathbf{J}_{\mathrm{t}_{i}}$ has only one row instead of two. We define $\mathbf{J} \in \mathbb{R}^{\left(2 n_{c}\right) \times n_{q}}$ by stacking the $\mathbf{J}_{\mathbf{n}_{i}}+\mu_{i} \mathbf{J}_{\mathbf{t}_{i}}$ and $\mathbf{J}_{\mathbf{n}_{i}}-\mu_{i} \mathbf{J}_{\mathbf{t}_{i}}$ from (3.6b) and (3.6c) into a single matrix, and partition $\mathbf{J}$ into $\mathbf{J}_{\mathbf{u}}$ and $\mathbf{J}_{\mathbf{a}}$ in a similar way as in (2.3).

More structure behind the $\mathbf{B}$ matrix (as defined in (2.8)) can be revealed with a bit of linear algebra. We can work out by hand the application of the implicit function theorem to the KKT conditions of (2.10), and the chain rule in (2.9b), to obtain an explicit expression for $\mathbf{B}$ :

$$
\begin{align*}
& \mathbf{B}=\left[\begin{array}{l}
\mathbf{B}^{\mathrm{a}} \\
\mathbf{B}^{\mathrm{u}}
\end{array}\right]=\left[\begin{array}{c}
\mathbf{I}-\left(h^{2} \mathbf{K}_{\mathrm{a}}\right)^{-1}\left(\tilde{\mathbf{J}}_{\mathrm{a}}\right)^{\top} \mathbf{P} \tilde{\mathbf{J}}_{\mathrm{a}} \\
\mathbf{M}_{\mathrm{u}}^{-1}\left(\tilde{\mathbf{J}}_{\mathrm{u}}\right)^{\top} \mathbf{P} \tilde{\mathbf{J}}_{\mathrm{a}}
\end{array}\right], \text { with }  \tag{3.7a}\\
& \mathbf{P}=\left[\tilde{\mathbf{J}}_{\mathrm{u}} \mathbf{M}_{\mathrm{u}}^{-1}\left(\tilde{\mathbf{J}}_{\mathrm{u}}\right)^{\top}+\tilde{\mathbf{J}}_{\mathrm{a}}\left(h^{2} \mathbf{K}_{\mathrm{a}}\right)^{-1}\left(\tilde{\mathbf{J}}_{\mathrm{a}}\right)^{\top}\right]^{-1} . \tag{3.7b}
\end{align*}
$$

where we assume $\tilde{\mathbf{J}}_{\mathrm{u}}$ and $\tilde{\mathbf{J}}_{\mathrm{a}}$ have full row rank. The tilde over a Jacobian indicates the sub-matrix formed by rows of the original matrix corresponding to the active constraints, i.e. contacts with non-zero contact forces.

The structure in $\mathbf{B}^{\mathrm{u}}$ explains why $\mathbf{B}_{\rho}^{\mathrm{u}}$ is a good measure of the object's reachability when there is contact. We can interpret Range $\left(\tilde{\mathbf{J}}_{\mathbf{u}}^{\top}\right)$ as achievable object motions under the specific subset of active contacts. By averaging $\mathbf{B}^{\mathbf{u}}$ computed from different contacts which can be activated from the nominal $(\bar{q}, \bar{u}), \mathbf{B}_{\rho}^{\mathrm{u}}$ summarizes possible object motions due to contact, in the form of Range $\left(\tilde{\mathbf{J}}_{\mathbf{u}}^{\top}\right)$ weighted by $\mathbf{M}_{u}$ and $\mathbf{P}$.

Furthermore, for a configuration with no active contacts, (3.7) implies that $\mathbf{B}^{\mathbf{a}}=\mathbf{I}$ and $\mathbf{B}^{\mathbf{u}}=\mathbf{0}$, as both $\tilde{\mathbf{J}}_{\mathrm{a}}$ and $\tilde{\mathbf{J}}_{\mathrm{u}}$ are empty matrices in the absence of active contacts. This has the intuitive interpretation that under a $u$ that does not lead to contacts, the robot will move to where it is commanded to, and the object will remain still.

As $\mathbf{B}_{\rho}^{\mathrm{u}}$ is the expected value of $\mathbf{B}^{\mathbf{u}}$, it follows naturally from the above observation that the local distance metric $d_{\rho, \gamma}^{u}$ tends to be dominated by the regularization $\gamma \mathbf{I}_{n_{u}}$ for a nominal configuration $\bar{q}$ where robots and objects are far from making contact. In such cases, the probability that an action $u$ sampled from a distribution $\rho$ centered at $\bar{q}^{\text {a }}$ leads to active contacts is low. As a result, in the Monte-Carlo estimation of $\mathbf{B}_{\rho}^{\mathrm{u}}$, such samples simply introduce $\mathbf{0}$ into the average, dragging the distance metric $d_{\rho, \gamma}^{u}$ towards being dominated by the regularization.

Example 4. (Metric on Planar Hand) We illustrate how the Mahalanobis metric can guide planning using the PlanarHand system first introduced in Sec. 3.2.2. As shown in Fig. 3-3, the system lives in the $x y$ plane, with gravity pointing into the paper along the negative $z$ direction. The system consists of two actuated 2-link robotic fingers and an unactuated puck which is free to translate and rotate. Each finger can interact with the ball through frictional contacts along both links.

For a given $q^{\mathrm{u}}$, the difficulty of reaching $q^{\mathrm{u}}$ from $(\bar{q}, \bar{u})$ can be measured by the local Mahalanobis metric $d_{\rho, \gamma}^{\mathrm{u}}$, whose 1-sublevel sets are shown in Fig. 3-3a as ellipsoids. Although object configurations $b_{1}$ and $b_{2}$ are equidistant to the origin under the globally-uniform Euclidean metric, $b_{1}$ is considered much closer than $b_{2}$ under the local Mahalanobis metric (red ellipsoid). Indeed, in Fig. 3-3b, reaching $b_{1}$ from the current puck configuration seems easier than reaching $b_{2}$. A similar observation can be made for the configuration in Fig. 3-3c.

In addition, the local Mahalanobis metric also varies greatly from one configuration another, as evidenced by the difference between the blue and red ellipsoids in Fig. 3-3a. This implies that a globally-uniform metric is rarely a good measure of reachability characteristics.

Lastly, the ellipsoid that corresponds to the 1-sublevel set shrinks as the nominal state gets further away from the contact manifold, as shown in Fig. 3-3d. This signifies that the configurations where the object is less accessible by the robot are naturally considered "further away" and can thus be avoided by the planner.

```
Algorithm 2: RRT
    Input: \(q_{\text {init }}, q_{\text {goal }}, K\);
    Output: \(\mathcal{T}\);
    \(\mathcal{T}=\left\{q_{\text {init }}\right\} ;\)
    for \(k=1, \ldots, K\) do
        \(q_{\text {subgoal }}=\) SampleSubgoal \((\mathrm{p}) ;\)
        \(q_{\text {nearest }}=\operatorname{Nearest}\left(q_{\text {subgoal }}\right)\);
        \(q_{\text {new }}=\operatorname{Extend}\left(q_{\text {nearest }}, q_{\text {subgoal }}\right)\);
        AddNode ( \(q_{\text {new }}\) );
        if GoalReached then
            break;
```


### 3.4 RRT through Contact

We are now ready to present our smoothing-based enhancements to the vanilla RRT algorithm, which we reproduce in Alg. 2 to establish notations for our discussion. Our method enhances RRT by incorporating (i) a reachability-aware Nearest operation based on the smoothed Mahalanobis metric on the unactuated objects $d_{\rho, \gamma}^{u}$, (ii) a fast Extend operation based on the projection of the subgoal to the range of $\mathbf{B}_{\rho}$; and (iii) a contact sampling procedure which improves the reachability of nodes added to the tree.

We denote the RRT tree as $\mathcal{T}=(\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Each node $q \in \mathcal{V}$ is simply a point in the configuration space of the system.

### 3.4.1 Nearest Node using Local Mahalanobis Metric

As illustrated in Sec.3.3, in particular by Example 4, a globally-uniform metric used by the vanilla RRT is usually a poor measure of reachability. Given a subgoal $q_{\text {subgoal }}$, if the nearest node $q_{\text {nearest }}$ is chosen under a globally-uniform metric, reaching $q_{\text {subgoal }}$ from $q_{\text {nearest }}$ may require large $u$ or even be dynamically infeasible. This will compromise RRT's ability to explore the configuration space, as trying to Extend towards a hard-to-reach $q_{\text {subgoal }}$ typically returns a child node that is close to the parent node $q_{\text {nearest }}$. In order to retain RRT's ability to efficiently explore under dynamics constraints, we use the smoothed Mahalanobis metric (3.5) instead of the usual Euclidean
metric in the Nearest step:

$$
\begin{equation*}
q_{\text {nearest }}=\operatorname{argmin}_{q \in \mathcal{V}} d_{\rho, \gamma}^{\mathrm{u}}\left(q_{\text {subgoal }} ; q\right) . \tag{3.8}
\end{equation*}
$$

### 3.4.2 Dynamically Consistent Extension

After choosing $q_{\text {nearest }}$ from the tree $\mathcal{T}$, we need an action or a sequence of actions that moves the system from $q_{\text {nearest }}$ to $q_{\text {subgoal }}$ subject to the dynamics constraint. One feasible strategy to connect $q_{\text {nearest }}$ to $q_{\text {subgoal }}$ is to solve for an input sequence $\left\{u_{t}\right\}_{t=0}^{T-1}$ using a trajectory optimization algorithm such as Alg. 1 [82]. However, the high computational cost of trajectory optimization motivates us to seek a simpler solution.

Fortunately, as a result of the farsightedness of quasi-static models, even an input sequence with $T=1$ (i.e. a single time step) can steer the system fairly far away from $q_{\text {nearest }}$. Although trajectory optimization with $T>1$ can explore a larger region around $q_{\text {nearest }}$, we find in practice that a single time step is sufficient for Extend to effectively grow the RRT tree $\mathcal{T}$.

We present the modified Extend that uses a single time step in Alg. 3. The input $u$ is computed by projecting ( $q_{\text {subgoal }}^{\mathrm{u}}-\mu_{\rho}^{\mathrm{u}}$ ) to Range $\left(\mathbf{B}_{\rho}^{\mathrm{u}}\right)$ using least squares (Line 3 ), which is significantly cheaper than solving trajectory optimization. Afterwards, we normalize the input and multiply it by some stepsize $\varepsilon$. The scaled input is then passed to the forward dynamics to obtain a new node. Crucially, we use the actual dynamics $f$ as opposed to the smooth surrogate dynamics $f_{\rho}$ (Line 4). This ensures that while the search for the next action relies on the smoothed model, the actual path is dynamically consistent under the original non-smooth contact dynamics (i.e. CQDC).

[^3]
### 3.4.3 Contact Sampling

A node $q$ where robots and objects are far from making contacts hinders the growth of the RRT tree for two reasons. First, such nodes are considered far away under the local Mahalanobis metric from most sampled $q_{\text {subgoal }}$, as the sublevel sets of their Mahalanobis distance metric have small volume (e.g. Fig. 3-3d). As a result, adding such nodes to the tree simply increases "deadweight" without improving coverage of the state space. Moreover, when such a node is chosen by Nearest, the Extend operation that follows often results in another non-contact configuration.

To reduce the number of such nodes in $\mathcal{T}$ and improve exploration during tree growth, the Extend operation is replaced, with some probability, by a new operation called ContactSample. ContactSample takes $q_{\text {nearest }}$ as input, and creates another node with a better local metric by fixing $q_{\text {nearest }}^{\mathrm{u}}$ and finding an informative $q_{\text {nearest }}^{\mathrm{a}}$ that makes contact with the object.

The ContactSample operation is essential for adequate exploration of the robot's state space, and needs to be designed differently for different robots. As an example, we briefly describe how ContactSample is implemented for the three robots shown in Fig. 3-5, which are used in our experiments in Sec. 3.5.

- PlanarPushing. The robot (the red sphere) is placed at a random point sampled on the perimeter of the box.
- PlanarHand. The robot consists of two fingers that can be treated as two-link robot arms. There are two different ways for the robot to contact the sphere: (i) enveloping grasps, where for each finger, we start with the finger straight and horizontal, and then rotate each joint towards the ball until all links are touching the ball; (ii) pinch grasps, where for each finger, we sample a point on the sphere, and solve inverse kinematics to find a finger configuration that touches the ball at the fingertip.
- AllegroHand (in all 4 systems with the hand). We start with the hand open and near the object (for AllegroDoor the object is the door knob). We then close
the hand along a randomly picked direction in the robot's joint space until the object is "grasped". The direction is generated by a weighted average of several EigenGrasps directions [92], where the weights are sampled randomly.

Contact sampling introduces non-physical behavior where the robot teleports from one configuration to another. This is not a problem when the object can sustain static equilibrium without the actuated DOFs that need to teleport. For instance, in AllegroHand, when the ball is supported by the palm, the fingers are free to move around the ball to regrasp. In contrast, if AllegroHand were facing downwards, the ball would fall under gravity if it were not secured by some of the fingers. Although this is a limitation of our current contact sampling implementation, we believe this can be resolved by a more sophisticated contact sampler which moves some of the actuated DOFs while keeping the object in static equilibrium with the rest.

### 3.4.4 Effectiveness of Proposed Enhancements

We introduce a new system with a 2 -dimensional object configuration space to illustrate the effectiveness of the proposed RRT enhancements:

- Planar Hand with fixed $y,(2,4,13)$. A simplified version of the PlanarHand system in Sec. 3.2.2. We fix the $y$-coordinate of the object, so that $q^{\mathrm{u}}=(x, \theta) \in$ $\mathbb{R}^{2}$ can be easily plotted on paper.

As shown in Fig. 3-4e, the vanilla RRT enhanced with the proposed Nearest, Extend and ContactSample achieves good coverage of the space of $q^{\mathrm{u}}$, which is crucial for RRT to adequately explore the configuration space and find a path to $q_{\text {goal }}$. In contrast, tree growth is stuck around the root without contact sampling (Fig. 3-4f) or the local metric (Fig. 3-4g).

We also illustrate how the tree grows throughout a complete run of the enhanced RRT in Fig. 3-4a-d. Even with the proposed enhancements, tree growth can get stuck at times. This is characterized by a specific type of subgraph of the tree which we call a "broom". A broom consists of one parent node with many child nodes, and is formed by repeated unsuccessful attempts to grow towards different subgoals


Figure 3-4: (a-d) RRT trees, shown in the space of $q^{u}$, at different iterations of a complete run of the enhanced RRT for the PlanarHandFixedY system. The contours are the sub-level sets of the local Mahalanobis metric of the nodes. The path from start ( $q_{\text {init }}$ ) to goal ( $q_{\text {goal }}$ ) is highlighted in red in the final tree of (d). (e-h) Visualization of RRT trees with the same number of nodes (50) but grown with different methods. (e) Tree grown with our algorithm; (f) without contact sampling; (g) using a globally uniform weighted Euclidean metric; (h) using exact gradients without smoothing. Note that our method achieves the best coverage of the space of $q^{\mathrm{u}}$.
from the same parent node. The occasional appearance of brooms is a sign that the proposed enhancements are not perfect. Nevertheless, the enhanced RRT is able to quickly branch out into empty part of the configuration space, and sufficiently cover the space as the tree grows.

### 3.4.5 Final Path Refinement

The final path returned by the RRT algorithm is visually plausible, yet suffers from two minor drawbacks: (i) RRT tends to produce randomized paths that can be shortened, and (ii) the big step size used in the Extend operation creates some nonphysical artifacts due to Anitescu's convex relaxation of the Coulomb friction model (Sec.2.3.1).

To mitigate these issues, we refine the RRT plan using trajectory optimization [84, 93] and short-cutting [94]. We first divide the RRT path into segments punctuated by ContactSample operations. We call these segments contact-rich as they


Figure 3-5: Tasks for RRT. Similar to Fig. 3-2b, the thicker frame denotes the goal, and the thinner frame the initial configuration of the object.
involve contact-based interactions between the object and the robot. We shortcut the sequence of trajectories by (i) removing consecutive ContactSample steps, and (ii) truncating each segment if there is no movement in $q^{\mathrm{u}}$. Then, for each contact-rich segment, we run trajectory optimization (Alg.1) with a smaller time step $h$, using the RRT path segment as the initial guess. This not only smooths the final path, but also ensures that each trajectory segment is more physically realistic. Finally, we connect adjacent contact-rich segments with a collision-free robot trajectory created by a collision-free RRT. We assume that the object configuration remains unchanged during the collision-free segment.

We find that combining these two strategies is effective in creating shorter and more physically realistic trajectories.

### 3.5 Results \& Discussion

In this section, we apply our algorithm on difficult 3D contact-rich manipulation problems previously only tackled by heavy offline approaches in RL [87, 95], and illustrate that we can generate plans on the order of a minute of online compute time, all on the CPU, which shows the efficacy of our method. The experiments in this section are designed to validate the following four hypotheses.

1. Using the smooth surrogate greatly improves the performance over using the exact dynamics for linearization.
2. The equivalence of smoothing schemes establishes that analytic and randomized smoothing will have similar levels of performance empirically, with analytic
smoothing showing superior computation time.
3. Using the Mahalanobis distance metric improves performance over a globally uniform distance metric.
4. Contact sampling greatly aids sample efficiency of the algorithm.

### 3.5.1 Experiment Setup

To test the efficacy of our algorithm and the above stated hypotheses, we run our algorithm to reach more challenging goals than the trajectory optimization examples in Sec. 3.2.2, as well as on 3 more contact-rich tasks on 3 new systems defined below.

1. Pen Placement $(\mathbf{6}, \mathbf{1 9}, \mathbf{2 4})$. The robot hand needs to both translate and rotate the pen [87] to the desired configuration.
2. Plate Pickup $(\mathbf{6}, \mathbf{1 9}, \mathbf{4 2})$. The robot has to exploit the external contact between the plate and the wall [34], showing extrinsic dexterity [88].
3. Door Opening $(\mathbf{2 , 1 9 , 2 2})$ [87] involves reasoning about a constrained system, where the handle must be rotated first before the door can be pushed open.

The definition of the number tuples is identical to Sec.3.2.2.
The contact-rich planning tasks are illustrated in Fig. 3-5. We design the tasks so that solving any of them with a single run of trajectory optimization is expected to fail due to their difficulty and the resulting non-convexity of the problem.

To compare our algorithm with different baselines, we rate the quality of planners using two metrics:

1. Iteration vs. Minimum distance to goal. we measure the distance between the goal and the tree, defined by $\min _{q \in \mathcal{V}}\left\|q^{\mathrm{u}}-q_{\text {goal }}^{\mathrm{u}}\right\|$ for every iteration. A successful planning algorithm would eventually reach the vicinity of the goal asymptotically, driving this metric to zero.
2. Iteration vs. Packing Ratio. To characterize the exploration performance of RRT, we do a Monte-Carlo estimation of the packing ratio, which is defined as the volume of the space occupied by the reachability ellipses, divided by the total volume of some workspace limit for the unactuated objects. The workspace limit is the set from which subgoals are sampled when running RRT. More formally, we define the numerator as

$$
\begin{equation*}
V_{\text {reachable }}=\operatorname{vol}\left(\left\{q^{\mathrm{u}} \mid \min _{\bar{q} \in \mathcal{V}} d_{\rho, \gamma}^{\mathrm{u}}(q ; \bar{q}) \leq \eta\right\}\right) \tag{3.9}
\end{equation*}
$$

where $\eta$ is some threshold on the distance metric. The Monte-Carlo estimate of $V_{\text {reachable }} / V_{\text {workspace }}$ can be computed by drawing $N$ samples within the workspace and counting how many of them belong to $V_{\text {reachable }}$. A good planner should asymptotically reach a ratio of 1 if the system can reach all points in the workspace.

For both metrics, the performance of our RRT algorithms averaged over 5 runs is plotted in Fig. 3-6 for different smoothing schemes, as well as ablations of different choices we made for the algorithm. In particular, we run three variations of our algorithm that misses a crucial ingredient.

1. Exact. We replace the linearization of the smooth surrogate $\mathbf{B}_{\rho}$ with the exact linearization $\mathbf{B}$, used for both extension and metric computation.
2. NoContact. We do not allow contact sampling (Sec.3.4.3) in this variant of the algorithm.
3. Global. Instead of the local Mahalanobis metric, we use a globally uniform metric during the Nearest step of the algorithm. For our experiments, we use a carefully-chosen weighted Euclidean norm.

### 3.5.2 Results \& Discussion

We plot the results of our experiments in Fig. 3-6, and display the running time of our algorithm using the three different smoothing schemes in Table 3.2. We discuss


Figure 3-6: Planning performance for the tasks in Fig. 3-5. Results include running RRT with the enhancements proposed in Sec. 3.4 using the three smoothing schemes from Sec. 2.4.1, as well as the three ablation studies proposed in Sec. 3.5.1.
some of our findings from the experiment, in the context of our hypotheses in the beginning of this section.

| Method | PPushing | PHand | AHand | APlate | APen | ADoor |
| :---: | ---: | ---: | ---: | ---: | ---: | ---: |
| A. | 3.25 | 10.32 | 31.01 | 117.16 | 24.84 | 12.42 |
| RF. | 7.50 | 21.34 | 80.12 | 161.64 | 86.12 | 34.55 |
| RZ. | 7.40 | 21.00 | 82.00 | 168.11 | 81.23 | 33.80 |

Table 3.2: Running time achieved by different methods in seconds. Every trial was run for 1000 iterations. We choose to not display running time for the ablation options since they are slight variations of Analytic with comparable running times.

## Smoothing vs. Exact

Throughout all experiments, we saw that using the exact linearization to compute the distance metric and extension results in much worse performance compared to any of the smoothing schemes, which supports our hypothesis that mode smoothing is necessary in order to solve many of the tasks.

## Analytic vs. Randomized Smoothing

For most of the tasks, we saw no meaningful difference between analytic and randomized smoothing schemes in terms of both how fast the goal is reached and the packing ratio. This empirically supports our theory that the two smoothing schemes are equivalent methods to compute local models of surrogate dynamics. The running time in Table 3.2, however, shows that analytic smoothing results in faster computation time as it does not require taking multiple samples.

Astute readers might have noticed that in the AllegroPlate results (Fig. 3-6b), the packing ratio curve of analytic smoothing lags behind those of both randomized smoothing schemes by a few hundred iterations. First of all, it takes only a few seconds to generate the several hundred samples, so this lag in practice is barely noticeable.

Nevertheless, we believe the small amount of lag can be attributed to hyperparameter tuning. For a given amount of smoothing, there is a sweet spot for the step size $h$ : if $h$ is too small, RRT makes little progress; if $h$ is too large, RRT
takes steps beyond the locality where the smoothed linearization is valid. As the sampling distribution corresponding to a specific $\kappa$ in analytic smoothing is difficult to determine in general, we usually independently pick the variance for randomized smoothing and the $\kappa$ for analytic smoothing, which means the sizes of the valid regions of the smoothed linearizations under different smoothing schemes can be different. Ideally, we should pick the best $h$ for every smoothing scheme, but in practice we found that an $h$ between 0.1 s and 0.2 s works reasonably well for all smoothing schemes on all systems. Coming back to the AllegroPlate example, we believe fine-tuning $h$ for analytic smoothing can remove the lag, but the benefit of doing so is marginal.

## Global vs. Mahalanobis Metric

Despite reasonable efforts to choose good weights for the weighted Euclidean norm, we consistently observed that the globally uniform weighted Euclidean metric resulted in much worse performance compared to the local Mahalanobis metric. This supports our hypothesis, and the findings of [85] that kino-dynamic RRT in general greatly benefits from guiding tree growth with reachability information.

## Effect of Contact Sampling

For some of the tasks (e.g. AllegroPlate, AllegroPen), contact sampling was not necessary. However, for examples that require resetting the actuator into a completely different configuration to make progress (e.g. PlanarPushing, AllegroHand, PlanarHand), contact sampling greatly improves the planner's performance.

### 3.6 Sim2Real Transfer \& Hardware Results

Although our planner successfully plans through our CQDC dynamics model, we further investigate if the plans can successfully transfer to real experiments. For this purpose, we run the obtained plans from Sec.3.5 in open-loop on a higher fidelity simulator Drake [66], as well as an actual hardware setting. These experiments further shed light on the efficacy and the limitations of our proposed method.

### 3.6.1 Experiment Setup

## Open-Loop Plan Transfer

Our plan consists of state and action sequences, i.e. lists of "knot" points, that are consistent with the CQDC dynamics. We first divide this plan into individual segments $\left(\left\{q_{k, \operatorname{sim}}\right\}_{k=0}^{K},\left\{u_{k}\right\}_{k=0}^{K-1}\right)$, punctuated by the ContactSample operation.

We convert the knot points $\left(\left\{q_{k, \operatorname{sim}}\right\}_{k=0}^{K},\left\{u_{k}\right\}_{k=0}^{K-1}\right)$ into state and action trajectories $q_{\mathrm{sim}}:[0, T] \rightarrow \mathbb{R}^{n_{\mathrm{u}}+n_{\mathrm{a}}}$ and $u:[0, T] \rightarrow \mathbb{R}^{n_{\mathrm{a}}}$ using first-order hold. Here $T$ denotes the duration of the trajectories in seconds. Specifically, we connect adjacent knot points with linear interpolation for positions and joint angles, or spherical linear interpolation (Slerp) for 3D orientations. The duration of each linear piece is computed such that the robots move at a small and constant speed. Lastly, rolling out $u(\cdot)$ on the real dynamics gives $q_{\text {real }}:[0, T] \rightarrow \mathbb{R}^{n_{\mathrm{u}}+n_{\mathrm{a}}}$, which is compared against $q_{\text {sim }}(\cdot)$ to evaluate the $\operatorname{sim} 2 \mathrm{r}$

## Evaluation Metrics

To evaluate the performance of sim2real transfer, we first define the mean error $\Delta(\cdot, \cdot)$ between the two trajectories $q_{\mathrm{sim}}^{\mathrm{u}}(\cdot)$ and $q_{\mathrm{real}}^{\mathrm{u}}(\cdot)$ as

$$
\begin{equation*}
\Delta\left(q_{\mathrm{sim}}^{\mathrm{u}}, q_{\mathrm{real}}^{\mathrm{u}}\right):=\frac{1}{T} \int_{0}^{T} d\left(q_{\mathrm{sim}}^{\mathrm{u}}(t), q_{\mathrm{real}}^{\mathrm{u}}(t)\right) \mathrm{d} t \tag{3.10}
\end{equation*}
$$

where $d(\cdot, \cdot)$ is the Euclidean 2-norm for position (in meters), and the absolute change in angle for orientation (in radians). Note that for 3D, this change of angle is welldefined in the axis-angle representation.

In addition, we expect that the metric $\Delta$ will depend on how much movement is inside the reference trajectory of the plan. To account for this scaling, we normalize $\Delta$ by dividing it by the length of the trajectory in the original plan, and denote the normalized error as $\bar{\Delta}$ :

$$
\begin{equation*}
\bar{\Delta}\left(q_{\mathrm{sim}}^{\mathrm{u}}, q_{\mathrm{real}}^{\mathrm{u}}\right):=\frac{\Delta\left(q_{\mathrm{sim}}^{\mathrm{u}}, q_{\mathrm{real}}^{\mathrm{u}}\right)}{L\left(q_{\mathrm{sim}}^{\mathrm{u}}\right)}, \tag{3.11}
\end{equation*}
$$

where the denominator computes the path length of $q_{\mathrm{sim}}^{\mathrm{u}}$ :

$$
\begin{equation*}
L\left(q_{\mathrm{sim}}^{\mathrm{u}}\right):=\int_{0}^{T}\left\|\dot{q}_{\mathrm{sim}}^{\mathrm{u}}(t)\right\|_{2} \mathrm{~d} t=\sum_{k=0}^{K-1} d\left(q_{k+1, \mathrm{sim}}^{\mathrm{u}}, q_{k, \mathrm{sim}}^{\mathrm{u}}\right) \tag{3.12}
\end{equation*}
$$

This normalization also takes into account the inherent scales of the system, and makes $\bar{\Delta}(\cdot, \cdot)$ a dimensionless quantity. For each system in Fig. 3-5, we obtain at least 10 segments and evaluate our error metrics.

## Simulation Setup

We transfer the examples of Fig.3-5 into Drake [66], which utilizes a full secondorder dynamics model with error-controlled integration, as well as a sophisticated and realistic contact model [96]. The collision geometries, robot controller stiffness and coefficients of friction are kept consistent between the CQDC dynamics and Drake.

## Hardware Setup

To verify results on actual hardware, we create a variant of the PlanarHand environment, where the object is replaced by a bucket, and 2 Kuka iiwa arms are used for the actuators. We name this environment liwaBimanual. We utilize a motion capture system to estimate the state of the bucket in order to compare the two trajectories of $q_{\mathrm{sim}}^{\mathrm{u}}(\cdot)$ and $q_{\text {real }}^{\mathrm{u}}(\cdot)$. Our setup is illustrated in Fig. 3-7.

### 3.6.2 Results \& Discussion

We plot the results of our experiments in Fig.3-8. While 2D systems such as PlanarPushing, PlanarHand, and liwaBimanual display low error and good sim2real transfer, 3D systems such as AllegroHand, AllegroPlate, AllegroPen and AllegroDoor show larger error. To better understand the discrepancy of sim2real performance on different systems, we visualized trajectories from all systems by overlaying $q_{\text {real }}^{\mathrm{u}}$ on top of $q_{\text {sim }}$ (some of these visualizations are shown in the accompanying video).

From the video, it is clear that on all systems, there exists a persistent phase


Figure 3-7: Hardware for the liwaBimanual setup, where the goal is to rotate the bucket by $180^{\circ}$. The left and right pictures correspond to the initial state and the final state after the open-loop plan execution. The lines between motion capture markers are connected to illustrate the change of pose in the bucket. Readers are encouraged to watch the accompanying video for the full execution.


Figure 3-8: Plots for sim2real performance of our CQDC dynamics, evaluated on the plans of Sec.3.5. First Two Columns: Scatter plot of mean error $\Delta$ vs. path length for position (first column) and orientation (second column). Each dot in the plot represents one segment trajectory. Last Two Columns: Box plot for the normalized error $\bar{\Delta}$ for positions (third column) and orientation (fourth column). Note that the mean in orange corresponds to the slope of the graph in the first two columns. Finally, we note that the AllegroHandDoor (AHD) example only consists of orientation, and has no position plot. Readers are highly encouraged to watch the accompanying video for the qualitative behavior of the actual segment trajectories in this plot.
difference between $q_{\text {real }}^{\mathrm{u}}$ and $q_{\mathrm{sim}}^{\mathrm{u}}: q_{\text {real }}^{\mathrm{u}}$ tends to lag $q_{\mathrm{sim}}^{\mathrm{u}}$ when the robot accelerates, and lead when the robot decelerates. This is not surprising, as the CQDC dynamics that generates $q_{\mathrm{sim}}^{\mathrm{u}}$ is inherently a first-order system, whereas $q_{\mathrm{real}}^{\mathrm{u}}$ is generated from second-order dynamics. On trajectory segments with good sim2real performance, the phase gap only results in harmless oscillations of $q_{\text {real }}^{\mathrm{u}}$ around $q_{\mathrm{sim}}^{\mathrm{u}}$. In these cases, we believe that the good sim2real performance validates our contact model. However, the phase gap can have more serious consequences, which we will discuss next.

## Violation of Quasi-static Assumption

The quasi-static assumption implies that objects are quickly stopped by damping when they are not "pushed around" by the robot. This is true on 2 D systems, as the friction patch between the object and its supporting surface is persistent and always provides enough damping to bring the object to still.

However, the necessary damping to uphold the quasi-static assumption does not always exist on 3D systems. For instance, in AllegroHand and AllegroHandPen, the point contact between the object and the palm provides very little damping. Most of the damping comes from the finger joints when the fingers are opposing the object's motion. Therefore, when the grasp on the object "leaves an opening", the object can roll quite far from the planned trajectory or even off the palm.

## Missed Contacts

Due to the non-smooth nature of contact dynamics, small discrepancies in object trajectory caused by the phase gap can lead to the robot completely missing contacts with the object. For example, we observed that in AllegroHandPlate or AllegroHandDoor, some grasps that were valid under the CQDC dynamics no longer succeeded in holding the object in place in Drake. The consequence of these failed grasps is that plates are dropped on the table in AllegroHandPlate, and door handles are missed in AllegroHandDoor.

## Necessity of Stabilization \& Robustification

These results tell us that the plan suggested by the CQDC dynamics might give a high-level direction, but its open-loop execution may not succeed under second-order dynamics with high velocities and low amount of damping. We believe that tracking this high-level plan requires low-level feedback controllers that can stabilize to the plan, and actively enforce the closed-loop system to be quasi-static. We also believe that the high-level planer can benefit from robustness objectives such as encouraging grasps that are considered good under classical grasping metrics. Such grasps will increase the control authority of the robot over the object, thereby providing sufficient damping and decreasing the chances of dropping the object. We leave these as promising directions for future work.

## Chapter 4

## External Contact Detection from Joint Torque Measurements

### 4.1 Introduction

No longer confined to factory-floor workcells repeating painstakingly hand-coded trajectories, robot arms today have been tasked with increasingly open-ended assignments such as "put this shoe on the shelf" or "load the dish washer", where the robots need to operate in unknown, unstructured environments potentially populated by humans. Naturally, the safety of such operations hinges upon the robot's ability to reliably handle unplanned collisions between any part of itself and the environment.

The ultimate sensor for collision detection is perhaps a sensitive tactile skin covering the entire surface of the robot [97,98]. However, such skins are rarely seen outside research labs, as they are usually expensive and prone to wear and tear. On the other hand, joint-level proprioceptive torque sensors are mature, robust and becoming more common in robot arms designed for human-robot interactions [99, 100].

Several techniques have been developed to estimate both the external contact force and its location on the whole robot using only proprioceptive torque sensors [101-104]. However, due to the sparse nature of proprioceptive measurements (one torque measurement per link), estimation of contact force and location from only joint torque has obvious limitations. For a typical serial robot arm with 7 links, when link


Figure 4-1: For the true contact shown in (a), which generates little torque about joint 6 and 7 , the two contacts in (b) and (c), represented by red cylinders, create almost identical torque measurements as (a).
$i$ (numbered from the base) of the arm is in contact, there are $i$ torque measurements available, and $i \leq 6$ if the contact is not on the end effector. The fact that the contact is pushing the robot gives two additional constraints: the force being in the friction cone and the position being on the robot's surface. In contrast, a contact position has 3 independent components to estimate, and a contact wrench has 6 . As a result of this deficiency in available measurements, the literature on joint-torque-based contact estimation commonly assumes that (i) there exists at most one external contact, and (ii) the contact generates negligible moments at the contact point $[101,104]$.

However, even within the boundary set by these simplifying assumptions, joint-torque-based contact estimation is still limited by the loss of detectability of contacts from joint torque measurements, which we will discuss more formally in Section. 4.4. Intuitively, as shown in Fig. 4-1, it is likely that multiple contact positions and forces (Fig. 4-1b and 4-1c) create almost identical torque measurements, making them impossible to distinguish by looking at joint torque alone. Although this failure mode has been observed in existing work [103], a thorough analysis on how often joint-torque-based contact estimators fail and what can be done to mitigate such failures appears to be absent from the literature.

In this chapter, we show quantitatively that two distinct contacts generating almost identical joint torque measurements is far from a 0 -probability event. Moreover,
this probability can get alarmingly high for links whose geometry is "concentrated" around their joint axes, such as links of the IIWA arm. Therefore, we believe that solving for a single contact estimate from joint torque measurements is an inadequate problem formulation. Instead we propose to estimate the set of possible contact positions consistent with the measurements. Elements of this set can be cast as the global optimal solution of a nonlinear optimization problem, which is difficult to solve directly. Nevertheless, by combining rejection sampling and gradient descent on manifolds, we propose an estimator that searches for local minima of the nonlinear optimization problem, and provide an efficient implementation capable of running at real-time rates. In practice, the proposed estimator usually finds all local minima on the links it searches. In addition, given the set of possible contact positions, we also propose an active contact discrimination strategy that falsifies spurious contact positions by slightly moving the robot.

### 4.2 Related Work

Although raw measurements from joint torque sensors include gravitational and inertial effects, the torque generated by external contacts, also known as the residual torque, can be extracted from them using external torque observers [101]. Having become an integral part in many robot arms' firmware [99, 100], such observers can update residual torque estimates at hundreds of Hz , providing the foundation for all proprioceptive contact estimation methods.

The contact estimator by Haddadin et al. [101] first determines the link in contact as the last link with non-zero residual torque. It then solves for the line of action of the contact force from a system of linear equations relating torque measurements to the contact wrench. Finally, if the contact geometry of the link is convex, intersecting the line with the link will give two potential contact points, one for pulling and the other for pushing. As contact forces almost always push, the contact point can thus be uniquely determined. However, solving for the line of force action needs at least 6 torque measurements and a full-rank contact Jacobian, which implies this method
does not produce any outcome on links more proximal to the base than link 6 or when the robot is close to singular. Moreover, when a link is not convex (e.g. link 5 in Fig. 4-1), the line of force action may intersect the link at more than two locations, making it impossible to uniquely determine the contact point. Last but not least, when the joint torque signals are noisy, which they always are, it is likely that the computed line of action does not intersect with the link geometry at all, making it impossible to determine the contact location [105].

Methods based on the Markov Chain Monte Carlo (MCMC) methodology [102, 104] can theoretically work on any link and with rank-deficient contact Jacobians, although estimation accuracy typically degrades on links too close to the base, or when the robot is close to singular. The degradation is not a limitation of the methods themselves, but a result of the loss of detectability of contacts from joint torque measurements. Both [102] and [104] use random walk on the robot's surface as the proposal distribution, and evaluate the likelihood of samples using the L2 norm of the difference between the measured joint torque and the joint torque created by the sample. The difference is that [102] assumes frictional contacts whereas [104] assumes that contacts are frictionless. The biggest drawback of MCMC methods is that they typically converge to only one local minima of the likelihood function, and are oblivious of other local minima when they exist.

Proprioceptive contact estimator based on machine learning has also been explored. Zwiener et al. discretize a robot's surface into finitely many patches, and train a classification network to predict the patch in contact from joint torque [103]. Although the contact classifier works well on both the training and validation data sets, its limitations and failure modes are difficult to analyze; its ability to generalize beyond the data set it is trained on is also hard to gauge.

### 4.3 Problem Formulation

For given joint angles $q \in \mathbb{R}^{n_{q}}$ and residual joint torque $\tau_{\text {ext }} \in \mathbb{R}^{n_{q}}$ created by one external contact at point $C^{\star}$, the problem is to find $p_{C^{\star}} \in \mathbb{R}^{3}$, the coordinate of
contact point $C^{\star}$, and the contact force $f_{C^{\star}}$. In other words, we would like to solve for $p_{C^{\star}}$ and $f_{C^{\star}}$ from the following equation:

$$
\begin{equation*}
\tau_{\mathrm{ext}}=\mathbf{J}_{C}\left(q, p_{C}\right)^{\top} f_{C}, \tag{4.1}
\end{equation*}
$$

where $C$ is a generic point on the robot's surface; $\mathbf{J}_{C}\left(q, p_{C}\right) \in \mathbb{R}^{3 \times n_{q}}$ the contact Jacobian that maps joint velocity $\dot{q}$ to the velocity of $C$.

As shown in Fig. 4-2a, the contact force $f_{C}$ needs to stay inside the friction cone at $C$ : $\left\|f_{C_{f}}\right\| \leq \mu\left\|f_{C_{n}}\right\|$. The second-order cone can be approximated with the polyhedral cone in Fig. 4-2b, which is generated by a set of $n_{d}$ extreme rays

$$
\begin{equation*}
\mathbf{V}_{C}=\left[v_{C_{1}}, \ldots, v_{C_{n_{d}}}\right] \in \mathbb{R}^{3 \times n_{d}} \tag{4.2}
\end{equation*}
$$

such that

$$
\begin{equation*}
f_{C}=\sum_{i=1}^{n_{d}} v_{C_{i}} \beta_{i}=\mathbf{V}_{C} \beta \tag{4.3}
\end{equation*}
$$

where $\beta_{i} \geq 0 \forall i[106]$.


Figure 4-2: A contact force on Link 6 of the KUKA IIWA robot. The orange mesh is the surface of the link. The triad represents the body frame of the link. (a) second-order friction cone. (b) Polyhedral approximation of the friction cone with $n_{d}=4$.

Solving (4.1) with the friction cone constraint is equivalent to solving the following
optimization problem:

$$
\begin{equation*}
\min _{\beta \geq 0, p_{C} \in \mathcal{S}}\left\|\mathbf{J}\left(q, p_{C}\right)^{\top} \beta-\tau_{\mathrm{ext}}\right\|^{2}, \tag{4.4}
\end{equation*}
$$

where $\mathbf{J}\left(q, p_{C}\right)=\mathbf{V}_{C}^{\top} \mathbf{J}_{C}\left(q, p_{C}\right) \in \mathbb{R}^{n_{d} \times n_{q}}$, and $\mathcal{S}$ is the manifold of the robot's surface. For fixed $q$ and $\tau_{\text {ext }}$, a contact position estimate $p_{C}$ has an associated cost defined as

$$
\begin{align*}
l\left(p_{C} ; q, \tau_{\text {ext }}\right) & :=\min _{\beta \geq 0}\left\|\mathbf{J}\left(q, p_{C}\right)^{\top} \beta-\tau_{\text {ext }}\right\|^{2}  \tag{4.5a}\\
& =\min _{\beta \geq 0}(\beta^{\top} \underbrace{\mathbf{J} \mathbf{J}^{\top}}_{\mathbf{Q}} \beta-2(\underbrace{\mathbf{J} \tau_{\text {ext }}}_{-b})^{\top} \beta+\tau_{\text {ext }}^{\top} \tau_{\text {ext }}) . \tag{4.5b}
\end{align*}
$$

The function $l\left(\cdot ; q, \tau_{\text {ext }}\right): \mathbb{R}^{3} \rightarrow \mathbb{R}$ is also called the residual, and can be computed by solving (4.5b), which is a convex QP. Using $l(\cdot)$, the set of solutions of (4.4) can be described by the set

$$
\begin{equation*}
P_{0}\left(q, \tau_{\mathrm{ext}}\right):=\left\{p_{C} \in \mathcal{S}: l\left(p_{C} ; q, \tau_{\mathrm{ext}}\right)=0\right\} . \tag{4.6}
\end{equation*}
$$

The true contact location $p_{C^{\star}}$ is clearly in $P_{0}\left(q, \tau_{\text {ext }}\right)$. However, $P_{0}$ can have many or even a continuum of elements, i.e. different contact forces at different positions can generate the same joint torque $\tau_{\text {ext }}$, as shown in Fig. 4-3. In general, it is not possible to distinguish $p_{C^{\star}}$ from other elements of $P_{0}\left(q, \tau_{\text {ext }}\right)$ using only $q$ and $\tau_{\text {ext }}$.

### 4.4 Detectability of contacts

The joint torque sensor of a link is only able to measure the torque about the axis of the link's revolute joint. It is therefore possible for the torque sensor to register zero or small torque measurement, even if the link is in contact with significant contact force. In this section, we give a quantitative analysis on how likely a contact force creates little or no measurable torque by studying the simplest case of a single link.


Figure 4-3: The residual $l(\cdot)$ for 20000 sampled points on link 5 and 6 of the IIWA arm. The true contact position and direction is indicated by the red line. Multiple global minima of $l(\cdot)$ can be found on the robot's surface $\mathcal{S}$.

### 4.4.1 Detectability

We assume that a link's joint axis is aligned with the z-axis of the link's body frame. Let $p_{C} \in \mathcal{S}$ denote a generic point on the robot's surface, $\mathcal{K}_{C}$ the friction cone at $p_{C}$, and $\tau_{z}(\cdot): \mathbb{R}^{3} \rightarrow \mathbb{R}$ the function that returns the z-component of the torque generated by a force. For illustrative purposes, the friction coefficient $\mu$ is set to 1 . We call the contact at $p_{C}$ fully detectable ${ }^{1}$ if $\forall f_{C} \in \mathcal{K}_{C},\left\|f_{C}\right\| \neq 0 \Rightarrow \tau_{z}\left(f_{C}\right) \neq 0$. Similarly, the contact is partially detectable if $\exists f_{C} \in \mathcal{K}_{C},\left\|f_{C}\right\| \neq 0 \Rightarrow \tau_{z}\left(f_{C}\right) \neq 0$, and undetectable if $\forall f_{C} \in \mathcal{K}_{C}, \tau_{z}\left(f_{C}\right)=0$.

We first look at the condition under which a contact at $p_{C}$ is fully detectable. A non-zero force at $f_{C}$ satisfies $\tau_{z}\left(f_{C}\right)=0$ if and only if its line of action intersects with the z-axis. The set of such $f_{C}$ 's belong to the plane which passes through both $p_{C}$ and the z-axis. We denote this plane by $\mathcal{P}_{C}$. The contact at $p_{C}$ is fully detectable if and only if $\mathcal{K}_{C} \cap \mathcal{P}_{C}=\{0\}$, which is equivalent to the following linear program being

[^4]infeasible:

Find $\beta$, subject to

$$
\begin{array}{r}
\left(p_{C} \times e_{z}\right)^{\top}\left(\mathbf{V}_{C} \beta\right)=0,  \tag{4.7b}\\
\beta \geq 1,
\end{array}
$$

where $e_{z}$ is the unit vector along the z-axis; $\left(p_{C} \times e_{z}\right)$ is normal to $\mathcal{P}_{C}$. (4.7b) constrains the contact force $f_{C}=\mathbf{V}_{C} \beta$ to $\mathcal{P}_{C}$. (4.7c) makes sure that $f_{C}$ is non-zero.

To find out which part of a link is fully detectable, we can sample uniformly on the link's surface and solve (4.7) for all samples. The results for two links with distinct geometries are shown in Fig. 4-4. As the IIWA link's surface is "concentrated" around its joint axis, only a tiny fraction of the link's surface is fully detectable. In contrast, the "elongated" link of UR5 has significantly more fully detectable surface, which is located further away from the joint axis. On both links, the majority of the samples are not fully detectable.


Figure 4-4: Full detectability of (a) link 6 of the IIWA arm and (b) "forearm" link of the UR5 arm. Out of the 20000 samples generated on each link, $2.19 \%$ and $32.58 \%$ of the samples on (a) and (b) are fully detectable, respectively. Fully detectable samples are shown in cyan, others in black. The meshes of the links are shown in (c) and (d). Points $A, B$ and $D$ are examples of different levels of detectability, which will be defined in Sec. 4.4.2.

### 4.4.2 Quantifying "undetectability"

A contact at $C$ is fully detectable if any contact force in $\mathcal{K}_{C}$ generates any nonzero torque measurement. However, as real torque sensor measurements are usually corrupted by noise, it is useful to know when a contact force generates a torque measurement no less than a threshold $\epsilon$.

Moreover, for contacts that are not fully detectable, it is worth noting that the extent of their "undetectability" varies. For example, point $A$ in Fig. 4-4c is undetectable, as it is the intersection of the z-axis and the link's surface. Although both point $D$ (Fig. $4-4 \mathrm{~d}$ ) and $B$ (Fig. $4-4 \mathrm{c}$ ) are partially detectable, it generally takes a smaller force to generate $\tau_{z}>\epsilon$ at $D$ than at $B$, as $D$ is further away from its joint axis.

The degree of a contact's "undetectability" with respect to the threshold $\epsilon$ can be quantified by:

$$
\begin{align*}
\eta_{\epsilon}\left(p_{C}\right) & :=\frac{\operatorname{Area}\left(\mathcal{D}_{c}^{\epsilon}\right)}{\operatorname{Area}\left(\mathcal{D}_{c}\right)} \in[0,1],  \tag{4.8a}\\
\mathcal{D}_{C} & :=\left\{f_{C} \in \mathbb{R}^{2}:\left\|f_{C}\right\|=1, f_{C} \in \mathcal{K}_{C}\right\},  \tag{4.8b}\\
\mathcal{D}_{C}^{\epsilon} & :=\left\{f_{C} \in \mathcal{D}_{C}: \tau_{z}\left(f_{C}\right) \leq \epsilon\right\}, \tag{4.8c}
\end{align*}
$$

where $\mathcal{D}_{C}$ is a "dome" in the friction cone consisting of unit-norm forces, and $\mathcal{D}_{C}^{\epsilon}$ is the subset of $\mathcal{D}_{C}$ consisting of forces whose torque measurement $\tau_{z}\left(f_{C}\right)$ is less than $\epsilon$. Intuitively, $\eta_{\epsilon}\left(p_{C}\right)$ is the proportion of contact forces in $\mathcal{K}_{C}$ that create small $(\leq \epsilon)$ torque measurements. The closer $\eta\left(p_{C}\right)$ is to 1 , the less detectable $p_{C}$ becomes. Using Fig. 4-4c and 4-4d again as examples, we expect $\eta_{\epsilon}\left(p_{A}\right)=1, \eta_{\epsilon}\left(p_{B}\right)$ be close to 1 , and $\eta_{\epsilon}\left(p_{D}\right)<\eta_{\epsilon}\left(p_{B}\right)$.

Computing $\eta_{\epsilon}\left(p_{C}\right)$ can be done analytically, but a sampling based approach is much easier to implement. Uniform samples on $\mathcal{D}_{C}$ can be easily generated, and checking membership of $\mathcal{D}_{C}^{\epsilon}$ is trivial. We can then approximate $\eta_{\epsilon}\left(p_{C}\right)$ by the ratio of samples in $\mathcal{D}_{C}^{\epsilon}$ to the total number of samples.

To evaluate the "undetectability" of an entire link, we can once again compute $\eta_{\epsilon}(\cdot)$


Figure 4-5: $\eta_{\epsilon}(\cdot)$ of IIWA link 6 and UR5 forearm.
for samples generated uniformly from the link's surface. The results for two different links using different $\epsilon$ 's are summarized in Fig. 4-5. As $\epsilon$ increases, the surface close to the z-axis quickly becomes fully undetectable. If we treat $\epsilon$ as a detection threshold, the result becomes particularly alarming for links with a more "concentrated" shape: even for $\epsilon=0.02$ (Fig. $4-5 \mathrm{a}$ ): $\eta_{\epsilon} \geq 0.4$ for $80 \%$ of the samples. In other words, if a force of 1 N is applied at a random point $C$ along a random direction in $\mathcal{K}_{C}$, the probability of not detecting the contact force is at least 0.32 .

### 4.4.3 Implications

The analysis in this section reveals a fundamental limitation of existing joint-torquebased contact estimation methods: when a contact force generates small torque about the axis of a link, the observed $\tau_{\text {ext }}$ could be equally well explained by a different contact point on a different link.

Haddadin's method [101] determines the link in contact as the most distal link with $\tau_{z}(\cdot) \geq \epsilon$. As there is a significant chance that $\tau_{z}\left(p_{C^{\star}}\right) \leq \epsilon$ at the true contact position $C^{\star}$, it is likely that their strategy believes the contact to be on a wrong link. MCMC methods $[102,104]$ evaluate the likelihood of points being the true contact point $C^{\star}$ using the residual (4.5). If $\eta_{\epsilon}\left(p_{C^{\star}}\right)$ is large, it is likely for other points to have a residual that is only larger by $\epsilon^{2}\left\|f_{C^{\star}}\right\|^{2}$, thereby trapping MCMC methods in a wrong local minimum of $l(\cdot)$.

### 4.5 Contact estimation from external torque measurements

We have demonstrated that the best contact estimate achievable from $q$ and $\tau_{\text {ext }}$ is $P_{0}\left(q, \tau_{\text {ext }}\right)$ (Section 4.3), which is likely to have more than one element (Section 4.4). Elements of $P_{0}\left(q, \tau_{\text {ext }}\right)$ are global optimizers of (4.4), which are also global minima of the residual $l\left(\cdot ; q, \tau_{\text {ext }}\right)$ on $\mathcal{S}$. However, due to the dependence of $J$ on $p_{C}$ and the manifold constraint $p_{C} \in \mathcal{S}$, (4.4) is nonlinear and difficult to solve directly.

In this section, we present a contact estimation strategy called RSGD, which is named after the combination of Rejection Sampling and Gradient Descent. RSGD is able to find every local minima of $l\left(\cdot ; q, \tau_{\text {ext }}\right)$. Although not as ideal as finding $P_{0}\left(q, \tau_{\text {ext }}\right)$, RSGD is stronger than MCMC methods: it locates every point on $\mathcal{S}$ to which MCMC methods may converge.

### 4.5.1 Rejection sampling

Starting with $P:=\left\{p_{C} \in \mathcal{S}\right\}$, a set of finitely many samples drawn uniformly from $\mathcal{S}$, we can calculate the residual $l(\cdot)$ for all samples in $P$, and keep the samples which satisfy $l \leq \epsilon$ :

$$
\begin{equation*}
P_{\epsilon}\left(q, \tau_{\mathrm{ext}}\right):=\left\{p_{C} \in P: l\left(p_{C} ; q, \tau_{\mathrm{ext}}\right) \leq \epsilon\right\} . \tag{4.9}
\end{equation*}
$$

When using a dense $P$ (Fig. 4-6a), $\left|P_{\epsilon}\right|$ is usually sufficiently large so that the local minima of $l(\cdot)$ can be estimated, for instance, by clustering the points in $P_{\epsilon}$, finding the cluster centers, and then projecting the centers back onto the robot surface $\mathcal{S}$. In contrast, when $P$ is sparse (Fig. 4-6b), the few samples in $P_{\epsilon}$ are typically too noisy to make a reasonable estimate.

The biggest drawback of rejection sampling is the high rejection rate, which, for example, can get to approximately $98 \%$ for the robot and contact configuration in Fig. 4-6. Calculating residuals for a very dense $P$ is therefore needed for an accurate contact estimate, which will incur prohibitively high computational cost.


Figure 4-6: Finding elements of $P_{\epsilon}$ on link 5 and link 6 with $\epsilon=0.005$ using rejection sampling. A force of 10 N is applied to the robot along the green line. Enclosed by magenta boxes, the colored squares represent accepted samples. (a) Dense P: 491 out of 20000 samples are accepted. (b) Sparse $P: 24$ out of 1000 samples are accepted.

### 4.5.2 Rejection sampling + gradient descent (RSGD)

The high computational cost of vanilla rejection sampling, due to the high rejection rate for small $\epsilon$, can be reduced significantly by

1. Generating a set of potential contact positions by rejection sampling using a sparse sample set $P$ and a large threshold $\delta$ :

$$
\begin{equation*}
P_{\delta}\left(q, \tau_{\mathrm{ext}}\right):=\left\{p_{C} \in P: l\left(p_{C} ; q, \tau_{\mathrm{ext}}\right) \leq \delta\right\} . \tag{4.10}
\end{equation*}
$$

2. Running gradient descent (Algorithm 4) for every $p_{C} \in P_{\delta}\left(q, \tau_{\text {ext }}\right)$, collecting the converged points $p_{\bar{C}^{\star}}$ into a set of locally optimal contact position estimates:

$$
\begin{equation*}
P_{\delta}^{\star}\left(q, \tau_{\mathrm{ext}}\right):=\left\{p_{\bar{C}^{\star}} \in \mathcal{S}\right\} . \tag{4.11}
\end{equation*}
$$

Note the bar over $\bar{C}^{\star}$. The bar indicates that the converged point $\bar{C}^{\star}$ is only an estimate of the true contact point $C^{\star}$.

As shown in Fig. 4-7, a large $\delta$ in Step 1 increases the acceptance rate, ensuring that $P_{\delta}$ has enough samples even if the initial sample set $P$ is sparse. Although samples in $P_{\delta}$ are spread out at the beginning, Step 2 runs them through Algorithm 1 , making most of them converge to local minima of $l(\cdot)$.

```
Algorithm 4: Gradient descent on manifold \(\mathcal{S}\)
    Input: \(q, \tau_{\text {ext }}, p_{C}\);
    Output: \(p_{\bar{C}^{\star}}\);
    while \(\left\|\nabla_{p_{C}} l\left(p_{C} ; q, \tau_{e x t}\right)\right\|>\varepsilon_{G}\) do
        \(t \leftarrow\left(I_{3}-n_{C} n_{C}^{\top}\right) \nabla_{p_{C}} l\left(p_{C} ; q, \tau_{\text {ext }}\right) ;\)
        \(a \leftarrow \operatorname{LineSearch}\left(p_{C}, t\right) ;\)
        \(p_{C} \leftarrow p_{C}+a t ;\)
        \(p_{C} \leftarrow \operatorname{Retract}\left(p_{C}, \mathcal{S}\right) ;\)
    \(p_{\bar{C}^{\star}} \leftarrow p_{C} ;\)
```

Algorithm 4 is the standard Gauss-Newton method for Riemannian optimization. In Line 4, the gradient $\nabla_{p_{C}} l$ is projected to the local tangent plane, which has normal $n_{C}$ and passes through $p_{C}$. In Line 5, a standard line search method is used to ensure that $l$ is decreasing after taking the gradient step at [49]. In Line 7, the new point in the local tangent plane is projected back onto $\mathcal{S}$.


Figure 4-7: Running RSGD on the same robot and contact configuration as Fig. 4-6. Step 1: 175 out of 1000 samples are accepted with $\delta=0.1$, which are shown as small squares color-coded by their residual values $l$. Step 2: 155 of the 175 samples in $P_{\delta}$ converge after running Algorithm 1, which are shown as large squares. Every $p_{C} \in P_{\delta}$ is connected by a line to the corresponding element in $P_{\delta}^{\star}$ to which it converges.

Using (4.5b), the gradient $\nabla_{p_{C}} l$ can be written as

$$
\begin{equation*}
\left(\nabla_{p_{C}} l\right)^{\top}=\frac{\partial l}{\partial p_{C}}=\frac{\partial l}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial p_{C}}+\frac{\partial l}{\partial b} \frac{\partial b}{\partial p_{C}} \tag{4.12}
\end{equation*}
$$

where $\frac{\partial \mathbf{Q}}{\partial p_{C}}$ and $\frac{\partial b}{\partial p_{C}}$ can be obtained using automatic differentiation [107]; $\frac{\partial l}{\partial \mathbf{Q}}$ and $\frac{\partial l}{\partial b}$ can be obtained from differentiating the implicit function defined by the optimality condition of QP (4.5b) [108].

As shown in Fig. 4-8a, Algorithm 1 is effective at reducing the residual $l(\cdot)$ of samples in $P_{\delta}$. Note that about 60 samples converge to positions on $\mathcal{S}$ with $l=0.003$, which is a local minimum on link 5 , but not the global minimum on link 6 . An example gradient descent run is shown in Fig. 4-8b.

The ability of an approach such as RSGD to find all local minima depends on two factors: the sampling strategy and the convergence properties of gradient descent.


Figure 4-8: (a) Distribution of residual $l(\cdot)$ of the $P_{\delta}$ and $P_{\delta}^{*}$ from Fig. 4-7. (b) A gradient descent run on link 6 of IIWA. Algorithm 4 starts at $p_{C} \in \mathcal{S}$ and converges to $p_{\bar{C}^{\star}}$. Red lines represent the path taken by gradient descent. White translucent disks represent local tangent planes.

By construction the entire surface of the robot is contained with the support of the distribution of the sampling procedure. As such, for every local minimum that has a non-measure-zero region of attraction, the probability that we draw a sample in that region and converge to the minimum in non-zero. Characterizing these regions of attraction, however, is more challenging, and one cannot rule out, e.g. saddle points and limit cycles. Nevertheless, empirically, we observe that the algorithm succeeds in finding all local minima for sufficiently dense sampling.

### 4.6 Active contact discrimination

In the event that RSGD returns multiple (possible) contacts, some form of active exploration may be desirable to discriminate the true contact from the spurious. Assuming contact with a static object, that will remain (approximately) in place irrespective of the robot's motion, a simple strategy to falsify a spurious contact is to move the robot so as to break contact (pull away) at that location; if residual torque remains, then this cannot have been the true contact (assuming no additional contacts were introduced during the robot's motion). Similarly, the robot motion may preserve (push into) a possible contact; if the residual torque vanishes, then this (spurious) contact is falsified. Given $N$ possible contacts $\left\{p^{i}\right\}_{i=1}^{N}$, rather than test
each (possible) contact individually, it is more efficient to "pull away from" $\lfloor N / 2\rfloor$ such contacts, and "push into" the other $\lceil N / 2\rceil$, thereby falsifying half of the contacts with each change of robot pose. The following program searches for such a change in pose, $\delta q \in \mathbb{R}^{n_{q}}$.

$$
\begin{align*}
& \min _{\delta q, b \in\{0,1\}^{N}}\left|\sum_{i=1}^{N} b_{i}-\lfloor N / 2\rfloor\right|  \tag{4.13a}\\
& n_{i}^{\top} \mathbf{J}_{i} \delta q \leq \epsilon_{\text {push }}^{\max }-\left(\epsilon_{\text {push }}^{\max }+\epsilon_{\text {pull }}^{\min }\right) b_{i}, i=1, \ldots, N  \tag{4.13b}\\
& n_{i}^{\top} \mathbf{J}_{i} \delta q \geq \epsilon_{\text {push }}^{\min }-\left(\epsilon_{\text {push }}^{\min }+\epsilon_{\text {pull }}^{\max }\right) b_{i}, i=1, \ldots, N  \tag{4.13c}\\
& \left|\left(I-n_{i} n_{i}^{\top}\right) \mathbf{J}_{i} \delta q\right| \leq r_{\text {orth }}, i=1, \ldots, N  \tag{4.13d}\\
& |\delta q| \leq \delta_{q}^{\max } \mathbf{1} . \tag{4.13e}
\end{align*}
$$

Here, $b \in\{0,1\}^{N}$ represents the decision(s) to pull away ( $b_{i}=1$ ) or push into ( $b_{i}=0$ ) the $i$ th contact. The objective (4.13a) attempts to push into as close to half $(\lfloor N / 2\rfloor)$ of the contacts as possible. The constraints (4.13b) and (4.13c) require that, e.g., a "pull away" moves the $i$ th possible contact point (on the robot) at least $\epsilon_{\text {pull }}^{\min }$, but at most $\epsilon_{\text {pull }}^{\max }$, in the opposite direction to the outward facing surface normal $n_{i}$, assuming a linearized relationship between the change in position and change in pose, $\delta p \approx \mathbf{J}_{i} \delta q$. Constraint (4.13d) restricts the motion (of each contact point) close to the corresponding surface normal, to minimize the chance of introducing new contacts after the change in pose. Constraint (4.13e) restricts the change in each joint angle. An example of this method in action is shown in Fig. 4-9.

There is no guarantee that this simple discrimination strategy will falsify all spurious contacts; success depends on problem specifics, e.g. robot pose, robot geometry, and the contact locations. In particular, (4.13) may return infeasible, or $b \equiv \mathbf{1}$ $(b \equiv \mathbf{0})$ (i.e. pull/push on all contacts, which gathers no useful information). However, problem (4.13) is a (convex) mixed-integer linear program (MILP) that can be efficiently solved to global optimality by commercial solvers. This means that, in the event of failure, we have a certificate that no such (sequence of) discriminating actions $\delta q$ exist, at least not without relaxing the constraints (4.13b)-(4.13e), or abandoning


Figure 4-9: (a) Two possible contact positions to disambiguate. The centers of the small red spheres are coincident with the candidate contact positions. (b) Solving (4.13) finds a motion that pulls away from the contact on link 6 and pushes into the contact on link 5 .
the linearized model and resorting to nonlinear motion planning.

### 4.7 Implementation

RSGD requires a lot more computation than existing methods. Nevertheless, by leveraging efficient open-source libraries, our implementation can run at real-time rates on a single CPU thread.

Fig. 4-10 shows the run-time breakdown of a typical iteration of RSGD, collected on a Mac mini with Intel i7-8700B CPU and 64GB of RAM. In Step 1, the residual $l(\cdot)$ is computed for 1000 points drawn uniformly from $S$, of which 184 points satisfy $l<\delta$. In Step 2, Algorithm 4 is run on each of the 184 points until convergence, or until the limit on gradient steps is reached.


Figure 4-10: Run-time breakdown of a typical iteration of RSGD.

As most of the time is spent on running Algorithm 4 for accepted samples in $P_{\delta}$, how long one iteration of RSGD takes is almost linearly proportional to $\left|P_{\delta}\right|$. In this case, $\left|P_{\delta}\right|=184$ leaves RSGD running at roughly 10 Hz . This can be improved, for instance, by using a sparser $P$, putting an upper bound on $\left|P_{\delta}\right|$, using more CPU threads, or a combination of these strategies.

An average run of Algorithm 4 takes $472 \mu s$. The most frequently-used atomic operation is computing the residual $l(\cdot)$, which involves solving QP (4.5b). In every gradient step, $l(\cdot)$ needs to be computed once to evaluate the gradient $\nabla_{p_{C}} l(\cdot)$, and a couple more times by line search. Moreover, $l(\cdot)$ is also computed for every sample in $P$ in the earlier rejection sampling step. The lightweight QP solver OSQP [109] allows us to compute $l(\cdot)$ quickly: it takes $6 \mu s$ on average to solve QP (4.5b).

Retracting points back onto the robot surface $\mathcal{S}$ is the second most time-consuming operation in Algorithm 4. With the robot surface $\mathcal{S}$ represented by triangle meshes, retraction can be done efficiently by a mature proximity query routine implemented in the Flexible Collision Library (FCL) [110].

The chain rule for computing $\nabla_{p_{C}} l(\cdot)$ in (4.12) is implemented with Eigen [73], which takes only $2 \%$ of the total time needed for Algorithm 4 to converge.

Algorithm 4 may fail to converge if gradient descent passes through a region of $\mathcal{S}$ with almost discontinuous surface normal, e.g. a groove or an engraved letter. Proximity queries also occasionally return a point off the mesh, throwing gradient descent off its track. Nonetheless, such failures are relatively rare and easy to detect and reject when they do occur.

Concerning the active contact discrimination, although complexity of the MILP is exponential in the number of contact locations to be falsified, moderate-size problems can be solved efficiently with SOTA solvers, such as GUROBI [80]; e.g., a problem with $N=10$ contacts can be solved in approximately 5 ms .

### 4.8 Conclusion

With a detailed analysis on two notions of contact detectability, we have demonstrated that a contact estimate from joint torque measurements typically consists of more than one possible contact positions, which are the global minima of the residual function $l(\cdot)$. Finding all global minima of $l(\cdot)$ is generally hard, but the proposed RSGD estimator empirically locates all local minima of $l(\cdot)$. Considering that joint torque measurements are inherently noisy, being able to find contact points with a small but positive residual could actually be beneficial. We have also provided a strategy to search for small robot motions which falsify as many spurious contact positions found by RSGD as possible. Moreover, when this strategy fails, it provides a certificate that no other small motion can do better.

On a robot that streams joint angle and residual torque ( $\tau_{\text {ext }}$ ) signals, such as the KUKA IIWA, deploying RSGD is expected to be straightforward. Nevertheless, pre-processing of the raw $\tau_{\text {ext }}$ signal provided by the robot's driver, which filters out noise and ensures that the signal is unbiased, will probably be necessary.

## Chapter 5

## Contact-aware Control with Quasi-static Models

### 5.1 Introduction

Most robots today are programmed to move through the world as if they are afraid of making contact. Perhaps they should be: unexpected collisions while a robot is tracking a trajectory can create a large force at the point of contact, putting at risk both the robot and the environment with which it interacts (Fig. 5-1). Consequently, a significant amount of effort and care in robot motion planning is spent on avoiding collisions. For example, sampling-based planners need to perform numerous collision checks [33]; optimization-based planners need to constantly evaluate the signed distance functions and their gradients [111]. Moreover, high resolution collision geometries are usually needed to increase the chances of finding a collision-free path, which further increases the computational cost [110].

However, the complete and total avoidance of contacts is a severe limitation, even if we are willing to tolerate the computational cost. First, the effectiveness of collisionfree planning is limited by the quality of the geometric models used for collision checks. Unless in structured environments where everything has been perfectly measured, models of the environment need to be reconstructed from range sensor (e.g. depth camera) measurements, which usually have a fair amount of uncertainty and can suffer


Figure 5-1: As transparent obstacles are almost invisible to depth sensors, a collision-free motion planner, with the goal to pick up the mug and unaware of the transparent tray, plans a trajectory that crushes the egg along the way. The crushed egg is highlighted in the red box. Our controller is able to keep the egg intact even when the reference trajectory would crush it.
from occlusion. Moreover, collision-free trajectories can be unnecessarily conservative [112]: a task achievable by making some contacts can be deemed infeasible by a collision-free planner.

In this chapter, we propose a QP controller which, given estimated contact positions and forces in unexpected contacts, tracks the reference trajectory as closely as possible while keeping contact forces below a user-defined upper bound. Compared with similar controllers based on null-space projection [113-117], the QP formulation shares the same underlying dynamics, but allows for more gentle separation when the robot breaks contact with the environment. The gentle separation happens naturally as a result of bounding the contact forces with inequality constraints, which are not supported by null-space projection.

Instead of the usual second-order dynamics constraints used in robot locomotion $[118,119]$, the proposed QP controller utilizes a simplified version of the CQDC dynamics model proposed in Chapter 2. Similar to CQDC, the simplified quasi-static model predicts future equilibrium configurations and contact forces for a stiffnesscontrolled robot in response to position commands. The simplified model consists
of only the robot, and assumes that the world is rigid and stationary. In addition, it assumes bi-lateral, frictionless contacts which can be modeled with only equality constraints. In addition to the numerical simplicity, this modeling choice also simplifies contact sensing in hardware experiments: it only needs contact locations on the robot, and does not need friction coefficients. As real-world contacts are uni-lateral and frictional, we also propose measures which both capitalize the simplicity of bilateral, frictionless contact models and mitigate the side effects of modeling real-world contacts as such.

### 5.2 Related Work

### 5.2.1 Interaction Control

As the primary objective of the proposed QP controller is to bound unexpected contact forces while tracking a joint-space or end-effector trajectory, we review existing methods for combined motion and force control, which are also referred to as interaction control in the literature [4, Chapter 9]. Interaction control techniques can be broadly classified by whether the interaction force is controlled directly or indirectly.

Direct force control typically splits the task space into two orthogonal subspaces based on the robot's kinematic constraints: one motion-controlled subspace along the tangents of the kinematic constraints, and one force-controlled subspace along the normals. Desired motion and force trajectories are specified in the motion and force controlled subspaces, and tracked independently using motor torque commands computed from the robot's second-order model [120-122].

However, the success of direct force control relies on accurate robot and environment models, which are not easily available in unstructured environments. Moreover, with few exceptions [100], most industrial robot arms, including the KUKA iiwa, do not have an interface for end-users to directly control motor torques [123]. Last but not least, by directly controlling motor torque, the interaction controller bypasses the robot's factory motion controller, and thus needs to run at high frequency in order
to maintain stability.
On the other hand, a classical example of indirect force control is impedance control [78], which regulates the robot's response to external forces to that of a mass-spring-damper system (a mechanical impedance), thereby guaranteeing interaction stability by passivity. When the robot moves slowly, which is often the case in manipulation tasks, impedance control can be simplified to stiffness control [124], which can be interpreted as connecting the robot's end effector to a user-specified set-point by virtual springs. In the presence of contact, contact force can be controlled by commanding how much the set point penetrates the obstacle.

Compared with direct force control, indirect force control schemes are usually implemented as an outer-loop around the robot's factory motion controller, and therefore does not bear the responsibility of maintaining stability and can run at a much lower rate.

### 5.2.2 Null-space Projection

Null-space projection is a classical and popular technique for executing a hierarchy of tasks defined by equality constraints [113-117]. The constraints imposed by higherpriority tasks are enforced by projecting the torque needed by low-priority tasks into the null space of the higher-priority tasks. The projections are defined over the ranges and null spaces of the task Jacobians and their weighted pseudo-inverses. It is noteworthy that the projections are generally not orthogonal, unless the weight matrix is identity [125].

More recently, controllers based on constrained optimizations such as quadratic programs (QP) have gained popularity in both locomotion $[118,119]$ and manipulation [126]. Compared with null-space projections, QP-based controllers can handle both equality and inequality constraints. In this work, we formulate the problem of trajectory tracking with bounded contact forces as a QP with a novel quasi-static dynamics constraint. We also show that a controller based on null-space projection implicitly enforces the same quasi-static dynamics constraint when the projection is stiffness-consistent [125].

### 5.3 Background and Notations

### 5.3.1 Constrained Inverse Dynamics Control

A popular controller in locomotion and manipulation is based on the following optimizationbased formulation [118, 119, 127]:

$$
\begin{gather*}
\min _{x^{l+1}, u^{l}} c_{x}\left(x^{l+1}\right)+c_{u}\left(u^{l}\right) \text {, s.t. }  \tag{5.1a}\\
\qquad x^{l+1}=f\left(x^{l}, u^{l}\right), \tag{5.1b}
\end{gather*}
$$

where $x^{l}$ and $u^{l}$ are the state and input at the current time step $l$, and $x^{l+1}$ is the state at the next time step, $l+1$. The controller (5.1) picks an action $u^{l}$ that minimizes the (usually LQR-style) state and action cost, $c_{x}(\cdot)$ and $c_{u}(\cdot)$, subject to the dynamics constraint (5.1b).

The most common choice for the dynamics constraint (5.1b) is the Newton's Second Law (N2L). For example, the DLR (German Aerospace Center) family of robots, including the KUKA iiwa and FRANKA panda, has the following closed-loop secondorder dynamics after gravity compensation [76]:

$$
\begin{equation*}
\mathbf{M}(q) \ddot{q}+\left(\mathbf{C}(q, \dot{q})+\mathbf{D}_{q}\right) \dot{q}+\mathbf{K}_{q}\left(q-q_{\mathrm{cmd}}\right)=\tau_{\mathrm{ext}} \tag{5.2}
\end{equation*}
$$

where $q \in \mathbb{R}^{n_{q}}$ is the joint angles of the robot, $\mathbf{C}(q, \dot{q}) \dot{q}$ is the Coriolis force, $\mathbf{D}_{q}$ is a diagonal damping matrix, $\mathbf{K}_{q}$ is a diagonal stiffness matrix, $q_{\mathrm{cmd}}$ is the commanded joint angles, and $\tau_{\text {ext }}$ is the torque by external contacts.

### 5.3.2 Contact and Multibody Notations

We consider rigid, point contacts in this work. The number of contacts the robot makes with the environment is denoted by $n_{c}$. Each contact point is denoted by $C_{i}$. The coordinates of the contact point relative to world frame, expressed in world frame is written as ${ }^{W} p^{C_{i}} \in \mathbb{R}^{3}$; contact force at $C_{i}$ expressed in world frame is represented by $f_{W}^{C_{i}} \in \mathbb{R}^{3}$.

Position Jacobian of the contact point $C_{i}$ relative to frame $W$, expressed in frame $W$, is denoted by $\mathbf{J}_{q}^{W_{p}{ }^{C_{i}}}(q): \mathbb{R}^{n_{q}} \rightarrow \mathbb{R}^{3 \times n_{q}}$. It maps the robot's joint velocity $\dot{q}$ to the velocity of point $C_{i}$ in world frame $W$ :

$$
\begin{equation*}
{ }^{W} v^{C_{i}}=\mathbf{J}_{q}^{W_{p}{ }^{C_{i}}}(q) \dot{q} \tag{5.3}
\end{equation*}
$$

We further define

$$
\begin{align*}
f_{i} & :=\left\|f_{W}^{C_{i}}\right\| \in \mathbb{R} ; u_{i}:=f_{W}^{C_{i}} / f_{i} \in \mathbb{R}^{3}  \tag{5.4a}\\
\mathbf{J}_{u_{i}} & :=u_{i}^{\top} \mathbf{J}_{q}^{W_{p} p_{i}} \in \mathbb{R}^{1 \times n_{q}}  \tag{5.4b}\\
\mathbf{J}_{u} & :=\left[\mathbf{J}_{u_{1}}^{\top}, \mathbf{J}_{u_{2}}^{\top}, \cdots \mathbf{J}_{u_{n_{c}}}^{\top}\right]^{\top} \in \mathbb{R}^{n_{c} \times n_{q}} . \tag{5.4c}
\end{align*}
$$

The $i$-th row of $\mathbf{J}_{u}$ maps $\dot{q}$ to the Cartesian velocity of $C_{i}$ along $u_{i}$ in world frame. We assume that $\mathbf{J}_{u}$ is full-rank.

### 5.4 Quasi-static Dynamics

### 5.4.1 Dynamics as Transitions between Equilibria

In this subsection, we introduce the simplified quasi-static contact model to be used by the inverse dynamics controller (5.1). When a CQDC dynamics model has only one robot and no objects, the system's state consists only of the joint angles $q$, and input the commanded joint angles $q_{\mathrm{cmd}}$. Similar to CQDC, the steady-state equilibrium condition can be obtained by setting the derivative terms in the secondorder dynamics (5.2) to 0 :

$$
\begin{equation*}
\mathbf{K}_{q}\left(q_{\mathrm{cmd}}-q\right)+\tau_{\mathrm{ext}}=0 \tag{5.5}
\end{equation*}
$$

As shown in Fig. 5-2, the simplifed quasi-static dynamics predicts $x^{l+1}:=q^{l+1}$, the equilibrium configuration at the next time step, from the current equilibrium configuration $q^{l}$ and the next commanded configuration $u^{l}:=q_{\mathrm{cmd}}^{l+1}$.


Figure 5-2: Quasi-static dynamics of a 2D, 2-link robot arm. The arm starts at $q^{l}$ (black) and is commanded to go to $q_{\mathrm{cmd}}^{l+1}(\mathrm{red})$. The virtual spring connecting $q_{\mathrm{cmd}}^{l+1}$ to $q^{l}$ pulls the robot towards $q_{\mathrm{cmd}}^{l+1}$, but the robot eventually stabilizes to $q^{l+1}$ (green) due to contact constraints. At $l+1$, the arm makes two contact with two obstacles at $C_{1}$ and $C_{2}$ with contact forces $\lambda_{1}$ and $\lambda_{2}$.

Assuming that the contacts are bi-lateral constraints instead of the usual unilateral constraints which need to be modeled with complementarity, the new equilibrium $q^{l+1}$ can be solved for by minimizing the potential energy of the robot subject to equality constraints:

$$
\begin{align*}
\min _{q^{l+1}} & \frac{1}{2}\left(q_{\mathrm{cmd}}^{l+1}-q^{l+1}\right)^{\top} \mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l+1}\right) \text { s.t. }  \tag{5.6a}\\
& \mathbf{J}_{u}\left(q^{l}\right)\left(q^{l+1}-q^{l}\right)=0 \tag{5.6b}
\end{align*}
$$

To derive the contact forces, we start with the Lagrangian of QP (5.6):

$$
\begin{equation*}
L\left(q^{l+1}, \lambda\right)=\frac{1}{2}\left(q_{\mathrm{cmd}}^{l+1}-q^{l+1}\right)^{\top} \mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l+1}\right)-\left(\lambda^{l+1}\right)^{\top} \mathbf{J}_{u}\left(q^{l+1}-q^{l}\right) \tag{5.7}
\end{equation*}
$$

where $\lambda^{l+1} \in \mathbb{R}^{n_{c}}$ is the Lagrange multipliers of the contact constraint (5.6b), which can also be interpreted as the contact forces generated by (5.6b); the dependency of $\mathbf{J}_{u}$ on $q^{l}$ is dropped for simplicity.

The KKT optimality condition of QP (5.6) is given by

$$
\begin{align*}
\nabla_{q^{l+1}} L=\mathbf{K}_{q}\left(q^{l+1}-q_{\mathrm{cmd}}^{l+1}\right)-\mathbf{J}_{u}^{\top} \lambda^{l+1} & =0  \tag{5.8a}\\
\mathbf{J}_{u}\left(q^{l+1}-q^{l}\right) & =0 \tag{5.8b}
\end{align*}
$$

where (5.8a) is equivalent to the steady-state force balance condition (5.5), assuming that $\tau_{\text {ext }}$ is generated by the $n_{c}$ point contacts, i.e. $\tau_{\text {ext }}=\mathbf{J}_{u}^{\top} \lambda^{l+1}$.

Explicit expressions for $\lambda^{l+1}$ and $q^{l+1}$ can also be derived from the KKT conditions (5.8):

$$
\begin{align*}
\lambda^{l+1} & =-\left(\mathbf{J}_{u} \mathbf{K}_{q}^{-1} J_{u}^{\top}\right)^{-1} \mathbf{J}_{u}\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right),  \tag{5.9a}\\
q^{l+1} & =q^{l}+\left(\mathbf{I}-\mathbf{K}_{q}^{-1} \mathbf{J}_{u}^{\top}\left(\mathbf{J}_{u} \mathbf{K}_{q}^{-1} \mathbf{J}_{u}^{\top}\right)^{-1} \mathbf{J}_{u}\right)\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right) . \tag{5.9b}
\end{align*}
$$

### 5.4.2 Relationship to Null-space Projection

In this sub-section, we show that when controlling stiffness-controlled robots using null-space projection, lower-priority tasks can be guaranteed to not interfere with higher-priority tasks during transients if the stiffness-consistent projection [125] is used. We also show that the underlying dynamics model of a controller based on stiffness-consistent projection is the same as the model proposed in Sec. 5.4.1.

Null-space projection technique revolves around two projections:

$$
\begin{align*}
P_{R}(W) & :=\mathbf{J}_{u}^{\top}\left(\mathbf{J}_{u}^{W+}\right)^{\top}  \tag{5.10a}\\
P_{N}(W) & : \mathbf{I}-\mathbf{J}_{u}^{\top}\left(\mathbf{J}_{u}^{W+}\right)^{\top} \tag{5.10b}
\end{align*}
$$

where $\mathbf{J}_{u}^{\mathbf{W}+}:=\mathbf{W}^{-1} \mathbf{J}_{u}^{\top}\left(\mathbf{J}_{u} \mathbf{W}^{-1} \mathbf{J}_{u}^{\top}\right)^{-1}$ is the pseudo-inverse weighted by a positivedefinite $\mathbf{W}$. The range and null space of $P_{R}(\mathbf{W})$ are respectively $R\left(\mathbf{J}_{u}^{\top}\right)$ and $N\left(\left(\mathbf{J}_{u}^{\mathbf{W}+}\right)^{\top}\right)$, whereas the range and null space of $P_{N}(\mathbf{W})$ are reversed. Note that such projections can be defined for arbitrary task Jacobians, but we specialize to the contact Jacobian
defined in (5.4c) without loss of generality.
For any choice of $\mathbf{W}$ and any joint torque $\tau_{\mathrm{m}}, P_{R}(\mathbf{W}) \tau_{\mathrm{m}} \in R\left(\mathbf{J}_{u}^{\top}\right)$ generates contact forces, whereas $P_{N}(\mathbf{W}) \tau_{\mathrm{m}}$ generates no joint torque in $R\left(\mathbf{J}_{u}^{\top}\right)$ after static equilibrium is reached. Choosing an appropriate $\mathbf{W}$, however, can provide additional guarantees during the transient into this steady state [125]. For instance, the dynamicallyconsistent pseudo-inverse $\mathbf{J}_{u}^{\mathbf{M}+}[128]$, which uses the robot's mass matrix $\mathbf{M}$ for $\mathbf{W}$, ensures that

$$
\begin{equation*}
0 \equiv \mathbf{J}_{u} \mathbf{M}^{-1} P_{N}(\mathbf{M}) \tau_{\mathrm{m}}, \forall \tau_{\mathrm{m}} \tag{5.11}
\end{equation*}
$$

Assuming that the dominant effect of $P_{N}(\mathbf{M}) \tau_{\mathrm{m}}$ during the transient is to generate acceleration, property (5.11) guarantees that and the generated acceleration lies inside $N\left(\mathbf{J}_{u}\right)$.

To determine the appropriate choice of $\mathbf{W}$ when the dominant effect of the $P_{N}(\mathbf{W}) \tau_{\mathrm{m}}$ during transient is to stretch/contract the virtual spring of a stiffness-controlled robot, we start at the instant $l^{+}$, immediately after sending the joint angle command $u^{l}=q_{\mathrm{cmd}}^{l+1}$ at time step $l$. At $l^{+}$, the joint torque can be expressed as

$$
\begin{equation*}
\tau_{\mathrm{m}}^{l^{+}}=\mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right), \tag{5.12}
\end{equation*}
$$

which can be decomposed as

$$
\begin{equation*}
\tau_{\mathrm{m}}^{l^{+}}=\underbrace{P_{R}(\mathbf{W}) \tau_{\mathrm{m}}^{l^{+}}}_{\tau_{R}}+\underbrace{P_{N}(\mathbf{W}) \tau_{\mathrm{m}}^{l^{+}}}_{\tau_{N}}, \tag{5.13}
\end{equation*}
$$

where $\tau_{R} \in R\left(\mathbf{J}_{u}^{\top}\right)$ generates contact forces, and $\tau_{N} \in N\left(\left(\mathbf{J}_{u}^{W+}\right)^{\top}\right)$ generates a motion that needs to be in $N\left(\mathbf{J}_{u}\right)$.

Combining (5.12) and (5.13) yields

$$
\begin{equation*}
\mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right)=\tau_{R}+\tau_{N} . \tag{5.14}
\end{equation*}
$$

At time instant $(l+1)^{-}$, when the equilibrium at time step $l+1$ is reached but $u^{l+1}$ has not been commanded, $\tau_{N}$ has generated a displacement and been dissipated
by damping, but $\tau_{R}$, the generalized force due to contact, remains:

$$
\begin{equation*}
\tau_{\mathrm{m}}^{(l+1)^{-}}=\tau_{R} . \tag{5.15}
\end{equation*}
$$

Moreover, static equilibrium (5.5) at $(l+1)^{-}$dictates that

$$
\begin{equation*}
\tau_{\mathrm{m}}^{(l+1)^{-}}=\mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l+1}\right) . \tag{5.16}
\end{equation*}
$$

Combining (5.15) and (5.16) yields

$$
\begin{equation*}
\mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l+1}\right)=\tau_{R} \tag{5.17}
\end{equation*}
$$

Finally, subtracting (5.17) from (5.14) gives

$$
\begin{equation*}
\mathbf{K}_{q}\left(q^{l+1}-q^{l}\right)=\tau_{N}=P_{N}(\mathbf{W}) \tau_{\mathrm{m}}^{l^{+}} \tag{5.18}
\end{equation*}
$$

As the motion from $l$ to $l+1$ needs to stay in $N\left(\mathbf{J}_{u}\right)$, we need $\mathbf{J}_{u}\left(q^{l+1}-q^{l}\right) \equiv 0$, which implies through (5.18) that

$$
\begin{equation*}
0 \equiv \mathbf{J}_{u} \mathbf{K}_{q}^{-1} P_{N}(\mathbf{W}) \tau_{\mathrm{m}}^{l^{+}}, \forall \tau_{\mathrm{m}}^{l^{+}} \tag{5.19}
\end{equation*}
$$

which has the same form as dynamic-consistency defined in (5.11). Not surprisingly, choosing $\mathbf{W}=\mathbf{K}_{q}$ ensures the motion generated by $\tau_{N}$ stays in $N\left(\mathbf{J}_{u}\right)$, and the resulting pseudo-inverse $\mathbf{J}^{\mathbf{K}_{q}+}$ is called stiffness-consistent [125].

We can solve for $q^{l+1}$ by plugging (5.12) into (5.18) and setting $\mathbf{W}$ to $\mathbf{K}_{q}$, yielding

$$
\begin{equation*}
q^{l+1}=q^{l}+\left(\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q}+} \mathbf{J}_{u}\right)\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right) . \tag{5.20}
\end{equation*}
$$

Furthermore, as the contact force is the reaction to $\tau_{R}$, we have $\tau_{R}=-\mathbf{J}_{u}^{\top} \lambda^{l+1}$, combining this with the definition of $\tau_{R}$ in (5.13) gives

$$
\begin{equation*}
\lambda^{l+1}=-\left(\mathbf{J}^{\mathbf{K}_{q}+}\right)^{\top} \mathbf{K}_{q}\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right) . \tag{5.21}
\end{equation*}
$$

It can be shown that (5.20) and (5.21) are equivalent to (5.9b) and (5.9a), respectively. This equivalence implies that a controller based on stiffness-consistent projections and a controller (5.1) using QP (5.6) as its dynamics constraint share the same underlying dynamics model. Therefore, the QP formulation in Sec. 5.5 is preferred as it can handle inequality constraints.

### 5.5 QP Controller with Quasi-static Dynamics

### 5.5.1 Frictionless Contacts

To track a reference trajectory $q_{\mathrm{ref}}(t)$ as closely as possible while respecting dynamics constraints and upper bounds on contact forces, we can specialize the generic optimization-based controller (5.1) to the following QP:

$$
\begin{align*}
\min _{q^{l+1}, q_{\mathrm{cmd}}^{l+1}, \lambda^{l+1}} & \left\|q^{l+1}-q_{\mathrm{ref}}^{l+1}\right\|^{2}+\epsilon\left\|q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{ref}}^{l+1}\right\|^{2}, \text { s.t. }  \tag{5.22a}\\
& \mathbf{K}_{q}\left(q^{l+1}-q_{\mathrm{cmd}}^{l+1}\right)-\mathbf{J}_{u}^{\top} \lambda^{l+1}=0  \tag{5.22b}\\
& \mathbf{J}_{u}\left(q^{l+1}-q^{l}\right)=0  \tag{5.22c}\\
& \lambda^{l+1} \leq \lambda_{\max }  \tag{5.22d}\\
& \left|q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{cmd}}^{l}\right| \leq \Delta q_{\max } . \tag{5.22e}
\end{align*}
$$

Here, the dynamics constraint (5.1b) consists of (5.22b) and (5.22c), which are the KKT conditions (5.8) of the quasi-static dynamics (5.6). Constraint (5.22d) places an upper bound on contact forces. The last constraint (5.22e) bounds how quickly $q_{\text {cmd }}$ changes.

In the objective (5.22a), the first term penalizes deviation at $l+1$ from the reference trajectory. The second term, weighted by a small positive scalar $\epsilon$, adds regularization without which the objective would become semi-definite. To see why, we re-write $\left(q^{l+1}-q_{\mathrm{ref}}^{l+1}\right)$ by expressing $q^{l+1}$ explicitly as a function of $q_{\mathrm{cmd}}^{l+1}$ using (5.20):

$$
\begin{equation*}
q^{l+1}-q_{\mathrm{ref}}^{l+1}=\left(\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q}+} \mathbf{J}_{u}\right)\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right)+\left(q^{l}-q_{\mathrm{ref}}^{l+1}\right), \tag{5.23}
\end{equation*}
$$

where the second term is a constant, and the first term multiplies $\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right)$ by a projection which has a non-zero null space.

In the quasi-static dynamics (5.6), expressing contact constraints as equality constraints (5.6b) and contact forces as the constraints' Lagrange multipliers implies that the contacts are bi-lateral and frictionless. In reality, however, contacts are uni-lateral and frictional.

The bi-lateralness of (5.6b) is less concerning. As contact sensors are inevitably noisy, only contact forces above a threshold are added to QP (5.22). In addition, by (5.9a), the change in contact force is bounded as long as $\left|q_{\text {cmd }}^{l+1}-q^{l}\right|$ is bounded, and the boundedness of $\left|q_{\mathrm{cmd}}^{l+1}-q^{l}\right|$ is enforced by (5.22e). Therefore, as long as the bound $\Delta q_{\max }$ is sufficiently small, we do not need to worry about contact forces flipping sign in the middle of a control step.

On the other hand, naively ignoring friction will severely impact the performance of the controller, which motivates the mitigating measures detailed in the next subsection.

### 5.5.2 Frictional Contacts

It is possible to model friction contact in quasi-static dynamics [44], but control through a frictional contact requires estimating the contact normal and the friction coefficient, in addition to estimating contact forces. This requires either more sophisticated whole-arm contact sensors, or making additional assumptions about the environments that make the control-estimation pipeline more brittle.

Therefore, we will retain the simpler frictionless contact model for controlling through a frictional contact, and mitigate the side effects of the wrong contact model by modifying the frictionless QP (5.22) to

$$
\begin{align*}
\min _{q^{l+1}, q_{\mathrm{cmd}}^{l+1}, \lambda^{l+1}} & \underbrace{\left\|q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{ref}}^{l+1}\right\|^{2}}_{\text {tracking }}+\underbrace{w^{l}\left\|q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{cmd}}^{l}\right\|^{2}}_{\text {damping }}, \text { s.t. }  \tag{5.24a}\\
& (5.22 \mathrm{~b}),(5.22 \mathrm{c}),(5.22 \mathrm{~d}) \text { and }(5.22 \mathrm{e})
\end{align*}
$$

which has the same constraints as (5.22) but a different objective. In the rest of this section, we will elaborate on the reason for both terms in the objective (5.24a).

## Tracking

When the reference trajectory $q_{\mathrm{ref}}^{l+1}$ leads the robot to make contact with a frictional surface at point $C$ (Fig. 5-3a), the surface normal $n \in \mathbb{R}^{3}$ and the contact force direction $u \in \mathbb{R}^{3}$ can be different. However, the frictionless contact model (5.22b)(5.22c) assumes that $u$ is always the same as $n$. Therefore, it is possible for ${ }^{W} v_{\text {cmd }}^{C}$, the commanded velocity of $C$, to have a negative component along $u$ but a positive component along $n$, as shown in Fig. 5-3a. Such a ${ }^{W} v_{\text {cmd }}^{C}$ would lead to the robot separating from the obstacle at $l+1$. When the frictionless QP (5.22) is constructed again at $l+1$, no contact force constraints are added but $q_{\mathrm{ref}}^{l+2}$ can still lead the robot to contact with a large amount of penetration. Therefore, the robot could re-establish contact with a large contact force at $l+2$.

Although it is difficult to guarantee that ${ }^{W} v_{\text {cmd }}^{C}$ has a negative component along $n$ without knowing $n$, undesired contact jitters can be effectively reduced by making ${ }^{W} v_{\text {cmd }}^{C}$ as close as possible to ${ }^{W} v_{\text {ref }}^{C}$. In joint space, this translates to minimizing the distance between $q_{\mathrm{cmd}}^{l+1}$ and $q_{\mathrm{ref}}^{l+1}$, which can be achieved by replacing the term $\left\|q^{l+1}-q_{\mathrm{ref}}^{l+1}\right\|^{2}$ in (5.22a) by $\left\|q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{ref}}^{l+1}\right\|^{2}$ in (5.24a).

To further illustrate the advantage of the new objective, we first re-write $q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{ref}}^{l+1}$ using the relative quantities defined in Fig. 5-3b:

$$
\begin{equation*}
q_{\mathrm{cmd}}^{l+1}-q_{\mathrm{ref}}^{l+1}=\left(q_{\mathrm{cmd}}^{l+1}-q^{l}\right)-\left(q_{\mathrm{ref}}^{l+1}-q^{l}\right)=\Delta q_{\mathrm{cmd}}^{l+1}-\Delta q_{\mathrm{ref}}^{l+1} . \tag{5.25}
\end{equation*}
$$

It is also easy to see from (5.23) that

$$
\begin{equation*}
\Delta q^{l+1}=\left(\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q}+} \mathbf{J}_{u}\right) \Delta q_{\mathrm{cmd}}^{l+1} . \tag{5.26}
\end{equation*}
$$

The first term in the original objective (5.22a) thus becomes

$$
\begin{align*}
q^{l+1}-q_{\mathrm{ref}}^{l+1} & =\Delta q^{l+1}-\Delta q_{\mathrm{ref}}^{l+1}  \tag{5.27}\\
& =\left(\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q}+} \mathbf{J}_{u}\right) \Delta q_{\mathrm{cmd}}^{l+1}-\Delta q_{\mathrm{ref}}^{l+1}
\end{align*}
$$

where $\left(\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q}+} \mathbf{J}_{u}\right)$ is the projection into $N\left(\mathbf{J}_{u}\right)$ along $R\left(\mathbf{J}_{u}^{\mathbf{K}_{q}+}\right)$, as shown in Fig. 5-3b.

As a result, minimizing $\left\|\Delta q_{\mathrm{cmd}}^{l+1}-\Delta q_{\mathrm{ref}}^{l+1}\right\|^{2}$ encourages $\Delta q_{\mathrm{cmd}}^{l+1}$ to be close to $\Delta q_{\mathrm{ref}}^{l+1}$ in the entire vector space. In contrast, when $\left\|\Delta q^{l+1}-\Delta q_{\mathrm{ref}}^{l+1}\right\|^{2}$ is used as the cost, only the distance between $\Delta q_{\text {ref }}^{l+1}$ and the component of $\Delta q_{\mathrm{cmd}}^{l+1}$ along $N\left(\mathbf{J}_{u}\right)$ is minimized.


Figure 5-3: (a): The reference trajectory brings the robot into contact at $C$. Due to friction, the contact normal $n$ and the contact force direction $u$ are different. Therefore, the commanded velocity at $C$ can separate from the obstacle even when the angle between $u$ and ${ }^{W} v_{\mathrm{cmd}}^{C}$ is greater than $\pi / 2$. (b): Definitions of $\Delta q_{\mathrm{ref}}, \Delta q_{\mathrm{cmd}}$ and $\Delta q$. The range and null space of the projection $\mathbf{I}-\mathbf{J}_{u}^{\mathbf{K}_{q+}} \mathbf{J}_{u}$ are $N\left(\mathbf{J}_{u}\right)$ and $R\left(\mathbf{J}_{u}^{\mathbf{K}_{q}+}\right)$, respectively.

## Damping

The goal of this term is to command more conservative robot motions when we are less confident in the correctness of the frictionless contact model. As real-world contacts are almost always frictional, the contact force predicted by the frictionless model and the actual contact force measured by contact sensors are bound to be different. At every time step, this discrepancy can be quantified by

$$
\begin{equation*}
e_{\lambda}^{l}:=1-\exp \left(\left\|\lambda_{\text {pred }}^{l}-\lambda_{\text {est }}^{l}\right\|_{\infty} / a\right) \in[0,1], \tag{5.28}
\end{equation*}
$$



Figure 5-4: (a): mug placement task. (b): mug moving task. Contacts are highlighted in red boxes. Blue arrows denote the direction of end effector velocity. For both tasks, the photographs on the left show where the real robot makes contact while executing the task. With collision geometry disabled, the simulation frames on the right show how much penetration would happen if the original trajectory were strictly followed. Videos of the real robot executing the tasks are included in the attachment.
where $\lambda_{\text {pred }}^{l}$ is the contact forces predicted by the frictional QP (5.24) at time step $l-1 ; \lambda_{\text {est }}^{l}$ is the measured contact forces at time step $l ; a$ is a positive constant that weights the force prediction error. The discrepancy $e_{\lambda}$ is close to 1 when the force prediction error is large, and close to 0 when the error is small.

The weight of the second term of (5.24a), $w^{l}$, is the low-pass-filtered version of the discrepancy $e_{\lambda}^{l}$ :

$$
\begin{equation*}
w^{l}=w_{\max }\left[\alpha e_{\lambda}^{l}+(1-\alpha) e_{\lambda}^{l-1}\right], \tag{5.29}
\end{equation*}
$$

where $w_{\max }$ is the upper bound on $w^{l}$ and $\alpha$ is the forget rate of the low-pass filter. A larger $w^{l}$ encourages more conservative robot motions by more heavily penalizing the change in $q_{\text {cmd }}$ from $l$ to $l+1$.

### 5.6 Experiments

In this section, we demonstrate the advantages of the proposed contact-aware controller through two tasks that involve unexpected contact with the environment, which
are shown in Fig. 5-4. In both experiments, $q_{\text {cmd }}$ is tracked using the iiwa's factory impedance controller. The factory controller's stiffness is set to [800, 600, 600, 600, 400, 200, 200] N • m/rad, from base joint to wrist joint. The proposed contact-aware controller runs at 200 Hz . The external contacts are estimated from iiwa's external torque measurements using the Contact Particle Filter [102], which runs at around 100 Hz . QPs are constructed using Drake's MathematicalProgram interface [66] and solved by GUROBI [80].

In order to reduce sensitivity to measurement noise, only contact forces with norm $f_{i} \geq f_{\text {threshold }}$ are considered when constructing the contact Jacobian $\mathbf{J}_{u}$ (5.4c). We have chosen $f_{\text {threshold }}=5 \mathrm{~N}$, and set the upper bound on contact forces in (5.22d) to be $\lambda_{\max }=15 \mathrm{~N}$. Ignoring contacts with small contact forces can be justified by the passivity of the robot's internal controller [129], which ensures stability in the presence of external contacts.

### 5.6.1 Mug Placement Task (Fig. 5-4a)

This task is defined by an end-effector pose trajectory ( ${ }^{W} R^{T_{r}}(t),{ }^{W} p^{T_{r}}(t)$ ), where $T_{r}$ is the reference for the tool frame $T,{ }^{W} R^{T_{r}}$ is the orientation of frame $T_{r}$ w.r.t. the world frame $W$, and ${ }^{W} p^{T_{r}}$ is the position of the origin of $T_{r}$ in world frame. It is straightforward to modify the tracking term in the frictional QP objective (5.24a) to minimize the pose difference between frame $T$ and its reference $T_{r}$, as described in [119] [130, Chapter 3].

The robot starts with a mug held in the gripper. It then (i) reaches down $(-z$ of world frame) by $0.22 m$ in 4 s , (ii) opens the gripper and drops the mug on the cart below the table in 2 s , and (iii) moves back up to where it started in 4 s . The orientation of the gripper is kept constant throughout the trajectory. As shown in Fig. 5-4a, the "wrist" (link 6) of the robot collides with the edge of the table as the gripper moves down and up.


Figure 5-5: mug placement task. $x, y, z$ components and the norm of the contact force $f_{W}^{C}$. Top: the baseline controller without contact force upper bounds is shown in the top plot. Bottom: the contact-aware controller (5.24) with a modified end effector tracking objective. In both plots, the horizontal red dashed line represents $f_{\text {threshold }}$ and the green dashed line $\lambda_{\text {max }}$.



Figure 5-6: mug placement task. Left: position tracking error. Right: orientation tracking error.

## Contact force

As shown in Fig. 5-5, except during the initial impact, our controller is able to keep the contact force norm $\left\|f_{W}^{C}\right\|$ close to $\lambda_{\max }$. In contrast, the baseline controller we compare against, which computes $q_{\mathrm{cmd}}$ by greedily minimizing tracking error without the dynamics and contact force constraints (5.22b)-(5.22e), incurs larger contact forces.

## Tracking error

As shown in Fig. 5-6, compared with the baseline, the contact-aware controller produces significantly less tracking error in the presence of external contact.

### 5.6.2 Mug Moving Task (Fig. 5-4b)

This task is defined by a joint-space trajectory $q_{\text {ref }}(t)$, with the goal of moving the mug along a straight line while keeping the mug orientation constant. The 16s trajectory $q_{\text {ref }}(t)$ is an interpolation between joint-space knot points obtained by inverse kinematics. As shown in Fig. 5-4b, to move the mug to the desired destination, the robot needs to first establish a contact with the top face of the table, and then breaks contact with the side of the table.

The baseline we are comparing against is the class of controllers trying to achieve similar goals as ours but uses null-space projection, such as [117]. As null-space projection cannot handle inequality constraints, the contact force upper bound is usually enforced by an equality constraint which sets the contact force to $\lambda_{\max }$. When commanded to break contact by the reference trajectory, i.e.

$$
\begin{equation*}
\mathbf{J}_{u}\left(q_{\mathrm{ref}}^{l+1}-q^{l}\right) \leq 0, \tag{5.30}
\end{equation*}
$$

the contact force constraint needs to be abruptly removed in order to continue to track the reference trajectory [117]. This can lead to large joint velocity during separation, as shown in Fig. 5-7.

In contrast, the proposed QP controller does not explicitly make the decision to break contact based on (5.30), instead the break of contact comes naturally as a consequence of solving QP (5.24) with the inequality constraints on contact forces (5.22d). As shown in Fig. 5-7, when the robot separates from the table, the drop in $\left\|f_{W}^{C}\right\|$ occurs gradually with our QP controller, but abruptly with a null-projectionbased controller.

### 5.7 Conclusion

We have presented a contact-aware controller that reconciles trajectory tracking with safety in unexpected contacts. The proposed controller is formulated as a QP with a quadratic cost on tracking error, a quasi-static model of the robot dynamics as


Figure 5-7: Comparison between our QP controller and null-space projection-based control in the mug moving task. Top: contact force norm $\left\|f_{W}^{C}\right\|$. The red dashed line represents $f_{\text {threshold }}$ and the green dashed line $\lambda_{\max }$. Both controllers can keep $\left\|f_{W}^{C}\right\|$ near $\lambda_{\max }$ when the robot is in contact. Bottom: robot joint velocity norm $\left\|v_{q}\right\|$ during contact separation (from $t=10 s$ to $t=14 s$ ). Note the velocity spike in the null-space projection controller. In both plots, the solid lines are the mean of 10 runs; the shaded regions around the lines represent the maximum and minimum values of all runs.
constraints, and upper bounds on contact forces.
The tasks for hardware experiments are designed based on our vision of future motion planners: they are comprised of smooth, simple trajectories defined by only a few knot points. We have shown that the proposed controller is able to keep both the tracking error and contact forces small if the robot makes an accidental contact. In addition, our controller outperforms controllers based on null-space projection when an established contact needs to be broken as the robot follows a reference trajectory.

It is difficult to reliably sense more than one contacts on the arm from only joint torque [131]. Nevertheless, with a more capable contact sensor such as [132], we believe the proposed contact-aware controller will greatly reduce robots' reliance on environment sensing/monitoring and collision-free motion planning.

## Chapter 6

## Conclusion

We motivated our work by noting the stark contrast between the promise of robots that are capable of contact-rich interactions, and the reality that most robotic manipulators are simply executing grasps connected by collision-free trajectories. Furthermore, the recent empirical success of RL in contact-rich settings has made it even more desirable and urgent to seek deeper understanding of contact-rich manipulation from a model-based perspective.

By analyzing the successes and pitfalls of existing model-based methods for contactrich planning, and understanding how RL was able to alleviate such pitfalls, we have identified two key ingredients for successful contact-rich planning: (i) smoothing of the non-smooth contact dynamics, and (ii) global exploration through contact dynamics constraints. By proposing a sampling-based motion planner guided by a smoothed contact dynamics model, we have shown that traditional model-based approaches can be effective in contact-rich manipulation planning. Compared to existing tools in RL which use heavy offline computation on the order of hours or days, our contribution offers a powerful alternative in the spectrum of solutions by enabling efficient online planning in the order of a minute while being generalizable with respect to environments and tasks.

At the heart of our model-based planner is CQDC, a novel contact dynamics model that is convex, quasi-static, and differentiable. In contrast with the standard secondorder dynamics, CQDC removes transients that can lead to myopic linearizations
which are uninformative about long-term planning. Through a number of theoretical arguments and empirical studies, we have shown the efficacy of our contact model. We have further shown that by inspecting the structure in our proposed model, we can analytically smooth out the contact dynamics with a log-barrier relaxation. With experiments, we have shown that our method of analytic smoothing has computational benefits over randomized smoothing.

In our analysis of existing methods for model-based contact-rich planning, we observed that smoothing has been tied to local trajectory optimization. Due to its weakness to local minima, local trajectory optimization has been less effective in difficult problems compared to RL-based approaches that attempt to perform global search. On the other hand, the SBMP methods for contact-rich systems have explicitly considered contact modes which fall into the pitfall of mode enumeration. Our contribution fills in a gap in existing methods by combining mode smoothing with RRT, where local approximation to the smooth surrogate was used to guide the exploration process of RRT via the local Mahalanobis metric.

By enabling SBMP to effectively search through contact dynamics constraints guided by smoothed CQDC, we have enabled efficient global motion planning for highly contact-rich and high-dimensional systems that were previously not achievable by existing model-based or RL-based methods. We believe that in the future, a highly optimized version of our planner can be used to perform real-time motion planning, or be used to guide policy search. With this capability, we hope to enable robots to find contact-rich plans online in previously unseen environments within seconds of planning time.

Planning is only part of the "sense-plan-act" pipeline that powers almost every modern robot. Although Chapter 4 and 5 only dabble in sensing and control, they shed light on what a complete contact-rich robotic system might look like. For instance, we believe that tactile sensing is needed to accurately determine contact forces, points and normals; and that a controller, which is responsible for short-horizon feedback, may function well with only local contact geometry and simplified contact dynamics.

We hope that our work has taken a small yet meaningful step towards building robots capable of human-level contact-rich interactions, which we hope will help make our world a better place.

## Bibliography

[1] Wonik Robotics. Allegro hand v4, 2022.
[2] Kenneth Salisbury, William Townsend, B Ebrman, and David DiPietro. Preliminary design of a whole-arm manipulation system (wams). In Proceedings. 1988 IEEE International Conference on Robotics and Automation, pages 254-260. IEEE, 1988.
[3] Sean O'Neill. How amazon robotics researchers are solving a "beautiful problem", 2022.
[4] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. Springer handbook of robotics, volume 200. Springer, 2008.
[5] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.
[6] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. arXiv preprint arXiv:2205.04422, 2022.
[7] Aykut Özgun Önol, Radu Corcodel, Philip Long, and Taşkın Padır. Tuning-free contact-implicit trajectory optimization. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 1183-1189. IEEE, 2020.
[8] Eric Huang, Xianyi Cheng, and Matthew T Mason. Efficient contact mode enumeration in 3d. In International Workshop on the Algorithmic Foundations of Robotics, pages 485-501. Springer, 2021.
[9] Russ Tedrake. Underactuated Robotics. 2022.
[10] David E Stewart. Rigid-body dynamics with friction and impact. SIAM review, 42(1):3-39, 2000.
[11] Mihai Anitescu and Florian A Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. Nonlinear Dynamics, 14(3):231-247, 1997.
[12] Lorenz T Biegler. Nonlinear programming: concepts, algorithms, and applications to chemical processes. SIAM, 2010.
[13] Sheldon Andrews, Kenny Erleben, and Zachary Ferguson. Contact and friction simulation for computer graphics. In ACM SIGGRAPH 2022 Courses, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.
[14] Dimitri P Bertsekas. Nonlinear programming. Journal of the Operational Research Society, 48(3):334-334, 1997.
[15] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. SIAM review, 47(1):99-131, 2005.
[16] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locallyoptimal feedback control of constrained nonlinear stochastic systems. In Proceedings of the 2005, American Control Conference, 2005., pages 300-306. IEEE, 2005.
[17] 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.
[18] Zachary Manchester and Scott Kuindersma. Variational contact-implicit trajectory optimization. In Robotics Research, pages 985-1000. Springer, 2020.
[19] Jorge Nocedal and Stephen J Wright. Numerical optimization. Springer, 1999.
[20] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4906-4913. IEEE, 2012.
[21] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ international conference on intelligent robots and systems, pages 5026-5033. IEEE, 2012.
[22] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pages 137-144, 2012.
[23] Roman Kolbert, Nikhil Chavan-Dafle, and Alberto Rodriguez. Experimental validation of contact dynamics for in-hand manipulation. In International Symposium on Experimental Robotics, pages 633-645. Springer, 2016.
[24] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In 2014 IEEE-RAS International Conference on Humanoid Robots, pages 295-302. IEEE, 2014.
[25] Dimitris Bertsimas and John N Tsitsiklis. Introduction to linear optimization, volume 6. Athena scientific Belmont, MA, 1997.
[26] Tobia Marcucci and Russ Tedrake. Mixed-integer formulations for optimal control of piecewise-affine systems. In Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pages 230-239, 2019.
[27] Tobia Marcucci, Robin Deits, Marco Gabiccini, Antonio Bicchi, and Russ Tedrake. Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In 2017 IEEE-RAS 17 th International Conference on Humanoid Robotics (Humanoids), pages 31-38. IEEE, 2017.
[28] Tobia Marcucci and Russ Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. IEEE Transactions on Automatic Control, 66(6):2433-2448, 2020.
[29] Robin Deits, Twan Koolen, and Russ Tedrake. Lvis: Learning from value function intervals for contact-aware robot controllers. In 2019 International Conference on Robotics and Automation (ICRA), pages 7762-7768. IEEE, 2019.
[30] Alp Aydinoglu and Michael Posa. Real-time multi-contact model predictive control via admm. In 2022 International Conference on Robotics and Automation (ICRA), pages 3414-3421. IEEE, 2022.
[31] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusherslider system: A story of hybrid and underactuated contact dynamics. In Algorithmic Foundations of Robotics XII, pages 800-815. Springer, 2020.
[32] Bernardo Aceituno-Cabezas and Alberto Rodriguez. A global quasi-dynamic model for contact-trajectory optimization. In Robotics: Science and Systems (RSS), 2020.
[33] Steven M LaValle. Planning algorithms. Cambridge university press, 2006.
[34] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. arXiv preprint arXiv:2105.14431, 2021.
[35] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 4245-4251. IEEE, 2020.
[36] Tobia Marcucci, Jack Umenberger, Pablo A Parrilo, and Russ Tedrake. Shortest paths in graphs of convex sets. arXiv preprint arXiv:2101.11565, 2021.
[37] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 39(1):3-20, 2020.
[38] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. arXiv preprint arXiv:2210.13702, 2022.
[39] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 23-30. IEEE, 2017.
[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
[41] Hyung Ju Terry Suh, Tao Pang, and Russ Tedrake. Bundled gradients through contact via randomized smoothing. IEEE Robotics and Automation Letters, pages 1-1, 2022.
[42] Tao Pang, HJ Suh, Lujie Yang, and Russ Tedrake. Global planning for contactrich manipulation via local smoothing of quasi-dynamic contact models. arXiv preprint arXiv:2206.10787, 2022.
[43] Mihai Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. Mathematical Programming, 105(1):113-143, 2006.
[44] Tao Pang and Russ Tedrake. A convex quasistatic time-stepping scheme for rigid multibody systems with contact and friction. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 6614-6620. IEEE, 2021.
[45] Alejandro M Castro, Frank N Permenter, and Xuchen Han. An unconstrained convex formulation of compliant contact. IEEE Transactions on Robotics, 2022.
[46] Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walaa M Moursi. Differentiating through a cone program. arXiv preprint arXiv:1904.09043, 2019.
[47] Matthew T Mason. Mechanics of robotic manipulation. MIT press, 2001.
[48] Nikhil Chavan-Dafle, Rachel Holladay, and Alberto Rodriguez. In-hand manipulation via motion cones, 2019.
[49] Stephen P Boyd and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.
[50] Taylor A Howell, Simon Le Cleac’h, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable simulator for robotics. arXiv preprint arXiv:2203.00806, 2022.
[51] Tao Pang and Russ Tedrake. Easing reliance on collision-free planning with contact-aware control. arXiv preprint arXiv:2109.09846, 2021.
[52] Matthew T Mason. Mechanics and planning of manipulator pushing operations. The International Journal of Robotics Research, 5(3):53-71, 1986.
[53] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. The international journal of robotics research, 15(6):533556, 1996.
[54] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. Wear, 143(2):307-330, 1991.
[55] Robert D Howe and Mark R Cutkosky. Practical force-motion models for sliding manipulation. The International Journal of Robotics Research, 15(6):557-572, 1996.
[56] Jiaji Zhou, J Andrew Bagnell, and Matthew T Mason. A fast stochastic contact model for planar pushing and grasping: Theory and experimental validation. In Robotics: Science and systems XIII, 2017.
[57] Jeffrey C Trinkle, RC Ram, AO Farahat, and Peter F Stiller. Dexterous manipulation planning and execution of an enveloped slippery workpiece. In [1993] Proceedings IEEE International Conference on Robotics and Automation, pages 442-448. IEEE, 1993.
[58] Jong-Shi Pang, Jeffrey C Trinkle, and Grace Lo. A complementarity approach to a quasistatic multi-rigid-body contact problem. Computational Optimization and Applications, 5(2):139-154, 1996.
[59] Benoit Landry, Zachary Manchester, and Marco Pavone. A differentiable augmented lagrangian method for bilevel nonlinear optimization. arXiv preprint arXiv:1902.03319, 2019.
[60] Vince Kurtz and Hai Lin. Contact-implicit trajectory optimization with hydroelastic contact and ilqr. arXiv preprint arXiv:2202.13986, 2022.
[61] Alp Aydinoglu, Victor M Preciado, and Michael Posa. Contact-aware controller design for complementarity systems. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 1525-1531. IEEE, 2020.
[62] Mathew Halm and Michael Posa. A quasi-static model and simulation approach for pushing, grasping, and jamming. In International Workshop on the Algorithmic Foundations of Robotics, pages 491-507. Springer, 2018.
[63] Tao Pang and Russ Tedrake. A robust time-stepping scheme for quasistatic rigid multibody systems. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5640-5647. IEEE, 2018.
[64] MOSEK ApS. Mosek Modeling Cookbook, 2021.
[65] Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 6054-6061. IEEE, 2014.
[66] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.
[67] Neville Hogan. Impedance control: An approach to manipulation: Part i-theory. 1985.
[68] Mihai Anitescu, James F Cremer, and Florian A Potra. Formulating threedimensional contact dynamics problems. Journal of Structural Mechanics, 24(4):405-437, 1996.
[69] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. SCS: Splitting conic solver, version 3.2.1. https://github. com/cvxgrp/scs, November 2021.
[70] Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E Brown, and Laurence Boissieux. An implicit frictional contact solver for adaptive cloth simulation. ACM Transactions on Graphics (TOG), 37(4):1-15, 2018.
[71] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. arXiv preprint arXiv:2103.16021, 2021.
[72] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In Robotics: Science and systems (RSS 2018), 2018.
[73] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
[74] Simon Le Cleac'h, Mac Schwager, Zachary Manchester, Vikas Sindhwani, Pete Florence, and Sumeet Singh. Single-level differentiable contact simulation, 2022.
[75] Louis Montaut, Quentin Le Lidec, Antoine Bambade, Vladimir Petrik, Josef Sivic, and Justin Carpentier. Differentiable collision detection: a randomized smoothing approach, 2022.
[76] Christian Ott, Alin Albu-Schaffer, Andreas Kugi, and Gerd Hirzinger. On the passivity-based impedance control of flexible joint robots. IEEE Transactions on Robotics, 24(2):416-429, 2008.
[77] Arun Samantaray. About bond graphs, 2001.
[78] Neville Hogan. Impedance control: An approach to manipulation. In 1984 American control conference, pages 304-313. IEEE, 1984.
[79] David E Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. International Journal for Numerical Methods in Engineering, 39(15):2673-2691, 1996.
[80] Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2019.
[81] Patrick Varin and Scott Kuindersma. A Constrained Kalman Filter for Rigid Body Systems with Frictional Contact, pages 474-490. 052020.
[82] Sertac Karaman and Emilio Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In 49th IEEE Conference on Decision and Control (CDC), pages 7681-7687, 2010.
[83] Claire Chen, Preston Culbertson, Marion Lepert, Mac Schwager, and Jeannette Bohg. Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 8262-8268. IEEE, 2021.
[84] H. J. T. Suh, Xiaobin Xiong, Andrew Singletary, Aaron D. Ames, and Joel W. Burdick. Energy-efficient motion planning for multi-modal hybrid locomotion. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7027-7033, 2020.
[85] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In 2009 IEEE International Conference on Robotics and Automation, pages 2859-2865. IEEE, 2009.
[86] Shadi Haddad and Abhishek Halder. Anytime ellipsoidal over-approximation of forward reach sets of uncertain linear systems, 2021.
[87] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018.
[88] Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S. Srinivasa, Michael Erdmann, Matthew T. Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 1578-1585, 2014.
[89] Taylor Howell, Simon Le Cleac'h, Sumeet Singh, Peter Florence, Zachary Manchester, and Vikas Sindhwani. Trajectory optimization with optimizationbased dynamics. IEEE Robotics and Automation Letters, 2022.
[90] Eric Huang, Xianyi Cheng, and Matthew T Mason. Efficient contact mode enumeration in 3d. In International Workshop on the Algorithmic Foundations of Robotics, pages 485-501. Springer, 2020.
[91] Richard M Murray, Zexiang Li, and S Shankar Sastry. A mathematical introduction to robotic manipulation. CRC press, 2017.
[92] Matei T. Ciocarlie, Corey Goldfeder, and Peter K. Allen. Dexterous grasping via eigengrasps : A low-dimensional approach to a high-complexity problem. 2007.
[93] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In IJCAI, pages 1930-1936, 2015.
[94] Roland Geraerts and Mark H. Overmars. Creating high-quality paths for motion planning. The International Journal of Robotics Research, 26(8):845-863, 2007.
[95] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In Conference on Robot Learning, pages 297-307. PMLR, 2022.
[96] Alejandro M. Castro, Ante Qu, Naveen Kuppuswamy, Alex Alspach, and Michael Sherman. A transition-aware method for the simulation of compliant contact with regularized friction. IEEE Robotics and Automation Letters, 5(2):1859-1866, apr 2020.
[97] Giorgio Cannata, Marco Maggiali, Giorgio Metta, and Giulio Sandini. An embedded artificial skin for humanoid robots. In 2008 IEEE International conference on multisensor fusion and integration for intelligent systems, pages 434438. IEEE, 2008.
[98] Advait Jain, Marc D Killpack, Aaron Edsinger, and Charles C Kemp. Reaching in clutter with whole-arm tactile sensing. The International Journal of Robotics Research, 32(4):458-482, 2013.
[99] Clive Loughlin, A Albu-Schäffer, S Haddadin, Ch Ott, A Stemmer, T Wimböck, and G Hirzinger. The dlr lightweight robot: design and control concepts for robots in human environments. Industrial Robot: an international journal, 2007.
[100] Franka Emika GmbH. Franka control interface documentation, 2019.
[101] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. IEEE Transactions on Robotics, 33(6):1292-1312, 2017.
[102] Lucas Manuelli and Russ Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5062-5069. IEEE, 2016.
[103] Adrian Zwiener, Christian Geckeler, and Andreas Zell. Contact point localization for articulated manipulators with proprioceptive sensors and machine learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 323-329. IEEE, 2018.
[104] Adrian Zwiener, Richard Hanten, Cornelia Schulz, and Andreas Zell. Armcl: Arm contact point localization via monte carlo localization. In IROS, pages 7105-7111, 2019.
[105] Lucas Manuelli. Localizing external contact using proprioceptive sensors: the contact particle filter. PhD thesis, Massachusetts Institute of Technology, 2018.
[106] David Stewart and Jeffrey 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. IEEE, 2000.
[107] Andreas Griewank et al. On automatic differentiation. Mathematical Programming: recent developments and applications, 6(6):83-107, 1989.
[108] John CG Boot. On sensitivity analysis in convex quadratic programming problems. Operations Research, 11(5):771-786, 1963.
[109] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. Mathematical Programming Computation, pages 1-36, 2020.
[110] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In 2012 IEEE International Conference on Robotics and Automation, pages 3859-3866. IEEE, 2012.
[111] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In 2009 IEEE International Conference on Robotics and Automation, pages 489494. IEEE, 2009.
[112] Matthew T Mason. Toward robotic manipulation. Annual Review of Control, Robotics, and Autonomous Systems, 1:1-28, 2018.
[113] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. Task-priority based redundancy control of robot manipulators. The International Journal of Robotics Research, 6(2):3-15, 1987.
[114] Bruno Siciliano and J-JE Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments, pages 1211-1216. IEEE, 1991.
[115] Farhad Aghili. A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. IEEE Transactions on Robotics, 21(5):834-849, 2005.
[116] Niels Dehio, Joshua Smith, Dennis Leroy Wigand, Guiyang Xin, Hsiu-Chin Lin, Jochen J Steil, and Michael Mistry. Modeling and control of multi-arm and multi-leg robots: Compensating for object dynamics during grasping. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 294-301. IEEE, 2018.
[117] Mikael Jorda, Elena Galbally Herrero, and Oussama Khatib. Contact-driven posture behavior for safe and interactive robot operation. In 2019 International Conference on Robotics and Automation (ICRA), pages 9243-9249. IEEE, 2019.
[118] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 2589-2594. IEEE, 2014.
[119] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas De Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. Design of a momentumbased control framework and application to the humanoid robot atlas. International Journal of Humanoid Robotics, 13(01):1650007, 2016.
[120] Matthew T Mason. Compliance and force control for computer controlled manipulators. IEEE Transactions on Systems, Man, and Cybernetics, 11(6):418432, 1981.
[121] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal on Robotics and Automation, 3(1):43-53, 1987.
[122] Marc H Raibert, John J Craig, et al. Hybrid position/force control of manipulators. Journal of Dynamic Systems, Measurement, and Control, 103(2):126-133, 1981.
[123] KUKA Roboter GmbH. Code based documentation of kuka fast robot interface c++ sdk, 2019.
[124] J Kenneth Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In 1980 19th IEEE conference on decision and control including the symposium on adaptive processes, pages 95-100. IEEE, 1980.
[125] Alexander Dietrich, Christian Ott, and Alin Albu-Schäffer. An overview of null space projections for redundant, torque-controlled robots. The International Journal of Robotics Research, 34(11):1385-1400, 2015.
[126] Advait Jain, Marc D Killpack, Aaron Edsinger, and Charles C Kemp. Manipulation in clutter with whole-arm tactile sensing. arXiv preprint arXiv:1304.6146, 2013.
[127] Yuquan Wang and Abderrahmane Kheddar. Impact-friendly robust control design with task-space quadratic optimization. In Robotics: Science and Systems (RSS), 2019.
[128] Roy Featherstone and Oussama Khatib. Load independence of the dynamically consistent inverse of the jacobian matrix. The International Journal of Robotics Research, 16(2):168-170, 1997.
[129] Alin Albu-Schäffer, Christian Ott, and Gerd Hirzinger. A unified passivitybased control framework for position, torque and impedance control of flexible joint robots. The international journal of robotics research, 26(1):23-39, 2007.
[130] Russ Tedrake. Robot manipulation: Perception, planning, and control (course notes for mit 6.881), 2021.
[131] Tao Pang, Jack Umenberger, and Russ Tedrake. Identifying external contacts from joint torque measurements on serial robotic arms and its limitations. In 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.
[132] Yiyue Luo, Yunzhu Li, Pratyusha Sharma, Wan Shou, Kui Wu, Michael Foshey, Beichen Li, Tomás Palacios, Antonio Torralba, and Wojciech Matusik. Learning human-environment interactions using conformal tactile textiles. Nature Electronics, 4(3):193-201, 2021.


[^0]:    ${ }^{1}$ Each contact can be in separation, sticking or four directions of sliding $(6=1+1+4)$. There are 20 bodies in the system.

[^1]:    ${ }^{1}$ At the time of writing this thesis, implicit PD controller is still an experimental feature of Drake, pending the merge of https://github.com/RobotLocomotion/drake/pull/17674. Implicit integration provides stability at much larger step sizes than explicit integration. The latter has been the default integration scheme in Drake due to the separation of plant and controller into individual systems. In our earlier work [44], we compared CQDC which uses implicit integration against Drake's MBP, which is an unfair comparison due to the difference in integration schemes.

[^2]:    ${ }^{2}$ In trajectory optimization, the complementarity constraint is usually relaxed to $\beta\left(q^{\mathrm{a}}+\delta q^{\mathrm{a}}\right) \leq$ $\kappa^{-1}$, instead of the equality in (2.33). The inequality gives the constraint non-empty interior, making it numerically better.

[^3]:    Algorithm 3: Extend
    Input: $q_{\text {nearest }}, q_{\text {subgoal }}$;
    Output: $q_{\text {new }}$;
    $\delta u^{\star}=\operatorname{argmin}_{\delta u}\left\|\mathbf{B}_{\rho}^{\mathrm{u}} \delta u+c_{\rho}^{\mathrm{u}}-q_{\text {subgoal }}^{\mathrm{u}}\right\| ;$
    return $f\left(q_{\text {nearest }}, q_{\text {nearest }}^{\mathrm{a}}+\varepsilon \cdot \delta u^{\star} /\left\|\delta u^{\star}\right\|\right)$;

[^4]:    ${ }^{1}$ The term "detectable", which is rigorously-defined in control theory, is abused in this section in order to facilitate exposition.

