Residual Force Polytope: Admissible Task-Space Forces of Dynamic Trajectories[☆]

Henrique Ferrolho[a,*], Wolfgang Merkt[b], Carlo Tiseo[a], Sethu Vijayakumar[a]

[a]*Institute for Perception, Action, and Behaviour, School of Informatics, University of Edinburgh (10 Crichton St, Edinburgh, EH8 9AB, United Kingdom)*
[b]*Oxford Robotics Institute, University of Oxford (23 Banbury Rd, Oxford, OX2 6NN, United Kingdom)*

Abstract

We propose a representation for the set of forces a robot can counteract using full system dynamics: the *residual force polytope*. Given the nominal torques required by a dynamic motion, this representation models the forces which can be sustained without interfering with that motion. The residual force polytope can be used to analyze and compare the set of admissible forces of different trajectories, but it can also be used to define metrics for solving optimization problems, such as in trajectory optimization or system design. We demonstrate how such a metric can be applied to trajectory optimization and compare it against other objective functions typically used. Our results show that the trajectories computed by optimizing objectives defined as functions of the residual force polytope are more robust to unknown external disturbances. The computational cost of these metrics is relatively high and not compatible with the short planning times required by online methods, but they are acceptable for planning motions offline.

Keywords: Robustness, Polytopes, Trajectory optimization, Robotic arms

1. Introduction

Robots have well-defined actuation limits and, usually, a clear definition of the task to be completed, but the conditions of the environment in which they operate may be a source of uncertainty. Besides environmental uncertainty, robots can also be affected by sensor noise, signal delay, and model mismatches, and these sources of error are often addressed with a feedback controller. However, controllers have their own limitations, and their ability to execute a motion depends not only on the complexity of the trajectory but also on the control authority available to track the motion plan and counteract any external disturbances at the same time. In general, there are two ways to improve robustness:

- During *control* ([1, 2, 3]), by increasing robustness when executing a nominal motion plan.

- During *planning* ([4, 5]), by considering uncertainty and robot capabilities to find trajectories with larger feasibility regions that can be exploited by controllers.

Being robust at the control stage does not necessarily result in a robust execution overall if the commanded motion is not robust itself. In fact, a bad motion plan will inherently compromise the robustness strategy of a controller. Despite the importance of robust controllers, we believe that ensuring robustness at an earlier stage is paramount for reliable deployment of robotic systems and, for that reason, this paper tackles the problem of increasing robustness during planning. While predicting and modeling uncertainty at the planning stage is difficult, we can exploit well-known capabilities and limitations of a system to optimize highly-robust trajectories. We argue that, by explicitly taking into account robot-specific capabilities and computing the set of admissible forces in task-space, we can define a metric as a function of that set to find trajectories that are more capable of resisting unexpected forces. To that end, we first propose a representation of admissible task-space forces taking into account the dynamics of the system (i.e., not limited to quasi-static scenarios), and then we test our hypothesis by defining an objective function based on our proposed representation and by comparing the optimization results with the trajectories obtained from using other existing approaches.

The main contributions of this work are:

1. Proposal of a representation of all the realizable forces given a configuration, a vector of forces/torques, and the system dynamics: the *residual force polytope*.
2. Elucidation of two models for representing force uncertainty and their combination with the residual force polytope for optimizing robust trajectories.
3. Comparison of several objective functions from related work with an objective function based on the residual force polytope for dynamic trajectory optimization.

2. Related work

In previous work [6], we exploited the kinematic redundancy of robots with many degrees of freedom in order to select configurations more robust to torque-tracking errors. Our

method indexes a previously-sampled database efficiently, but it does not optimize the robot's ability to resist unknown external disturbances and is limited to choosing single configurations. In contrast, this paper focus on robustness against unexpected forces and demonstrates how the states of the system can be optimized for entire trajectories to achieve more robust motions. Other researchers have also exploited kinematic redundancy to improve robot capabilities. For example, Yoshikawa [7] proposed the *force manipulability ellipsoid* to take into account the ability to apply and resist forces based on the robot geometry. Nonetheless, the use of ellipsoids is only an approximation, and makes it difficult to capture and incorporate descriptions of other system constraints. The polytope of admissible forces that we propose in this paper is not an approximation, and allows for easy integration of extra constraints through polytope manipulation.

The ability to manipulate and intersect polytopes can be very useful. For example, it allows the aggregation of multiple constraints into a single description of necessary conditions for feasibility of a system, provided that each individual constraint can be modeled in the form of a polytope. For example, Audren and Kheddar [8] extended 2D stability regions to 3D by accounting for possible center-of-mass accelerations in order to achieve robust multi-contact stability in whole-body posture generation. Orsolino et al. [4] proposed the actuation wrench polytope and intersected it with the contact wrench cone [9] to create the feasible wrench polytope. The actuation wrench polytope is a representation of all the wrenches a robot can generate given its actuation limits. However, it is limited to quasi-static scenarios. In this paper, we propose a new representation that accounts for the dynamics of the system and the torques required by a nominal motion, hence, providing a description of the admissible forces for dynamic scenarios. In [4], the feasibility polytope was used to optimize the center-of-mass position of a quadruped's static crawl gait. However, due to the required computational cost, they calculated the polytope once at the beginning of the optimization and used that as a constant approximation thereafter. As such, computing the exact polytope at every point of the trajectory during optimization and the impact of this approach on performance are two important aspects that have not been studied before, and which we address in this work. It is also worth noting that the optimization problem in [4] optimizes four variables in time (the positions and velocities of the center of mass in the *xy*-plane), while our problem optimizes 21 variables in time (the state and control inputs of a 7-DoF robot arm) and is therefore significantly more complex.

Finally, the idea of improving the robustness of robot motions using trajectory optimization has been explored before: Manchester and Kuindersma [5] presented an algorithm that incorporates linear feedback, bounded disturbances, and a penalty for closed-loop deviations from a nominal trajectory. A key advantage of their method is that the resulting control trajectories avoid *bang-bang* control, and leave margins of stability for LQR feedback control around the nominal trajectory. Our approach also retains these advantages as a result of the polytope-based objective functions and, additionally, the new representa-tion we propose allows determining the exact margins remaining before torque saturation occurs.

3. Preliminaries

3.1. Polytopes and the Double Description Method

A convex polytope [10] can be defined in one of two ways:

- Vertex representation ($\mathcal{V}$-rep): a finite set of points;

- Half-space representation ($\mathcal{H}$-rep): a bounded intersection of a finite set of half-spaces.

For some mathematical operations, one representation has some inherent advantages over the other. For example, the intersection of two or more polytopes is easier to perform in $\mathcal{H}$-rep than in $\mathcal{V}$-rep, and a Minkowski sum is easier to carry out in $\mathcal{V}$-rep than in $\mathcal{H}$-rep.

It may happen that a $\mathcal{V}$-rep is required when only an $\mathcal{H}$-rep is available, or vice versa—this is known as the representation conversion problem. It is possible to convert from one representation to the other using the double-description method [11]. Nonetheless, switching between representations can be computationally very expensive and should be avoided.

3.2. Robot Model Formulation

Consider a fully-actuated robot manipulator with n degrees of freedom, a fixed base, and with an end-effector operating in an m-dimensional task-space. Such a system can be parameterized with a generalized coordinates vector $\boldsymbol{q} \in \mathbb{R}^n$ and a generalized velocities vector $\boldsymbol{v} \in \mathbb{R}^n$. The dynamics of the system are given by the equations of motion:

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{\tau} + \boldsymbol{J}_e^\top(\boldsymbol{q})\boldsymbol{f}_{\text{tip}}, \tag{1}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is a symmetric positive-definite mass matrix, $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^n$ is the vector of Coriolis, centrifugal, and gravity terms, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of joint forces and torques, $\boldsymbol{J}_e \in \mathbb{R}^{m \times n}$ is the Jacobian matrix that maps joint velocities to the linear velocity of the end-effector, and $\boldsymbol{f}_{\text{tip}} \in \mathbb{R}^m$ is a force applied to the end-effector. The transpose of $\boldsymbol{J}_e$ maps a linear force applied at the end-effector to a vector of torques experienced at the joints of the mechanism—in the following referred to as $\boldsymbol{\tau}_{f_{\text{tip}}}$. Conversely, we can determine an end-effector force generated from a vector of input torques with

$$\boldsymbol{f}_{\text{tip}} = \boldsymbol{J}_e^{-\top}\boldsymbol{\tau}_{f_{\text{tip}}}. \tag{2}$$

This mapping is instrumental for computing force polytopes.

3.3. Joint Force Polytope and Force Polytope

The *joint force polytope* [12] is an n-dimensional region bounded by the upper and lower actuation limits of the system. It is described by the $2n$ bounding inequalities

$$|\tau_i| \leq \tau_{i,\text{lim}} \qquad i = 1, \cdots, n, \tag{3}$$

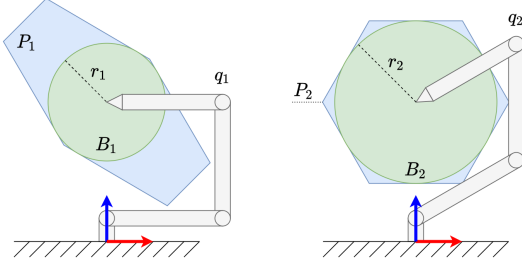where $\tau_{i,\text{lim}}$ is the bound on the i-th joint force.

2

Figure 1: Two valid configurations for reaching the same end-effector target. The blue polygons P_1 and P_2 are the force polytopes of configurations q_1 and q_2, respectively. The green circles B_1 and B_2 are the largest balls centered at the end-effector that can be inscribed inside those polytopes. The radius of B_1 and B_2 are denoted by r_1 and r_2, and here $r_2 > r_1$.
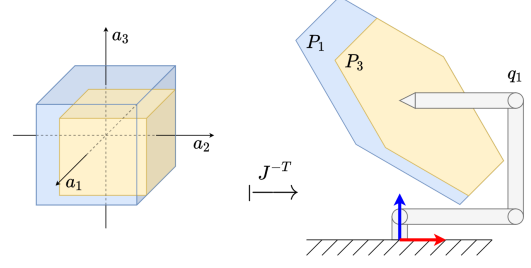

Figure 2: Equation (2) transforms actuation-space representations (on the left) into task-space representations (on the right). The blue polyhedron on the left is the *joint force polytope*, and by taking into account a given vector of torques it is reduced along some dimensions into the yellow polyhedron. The yellow polygon P_3 (on the right) is the *residual force polytope*.

The *force polytope* is the convex set of all the realizable forces by the end-effector for quasi-static scenarios, given the actuation limits of the system. A force polytope P_f results from transforming a joint force polytope P_τ with $P_f = J_e^{-\top} P_\tau$, analogous to how equation (2) converts a vector of joint-space forces and torques into a task-space force. Because of this nonlinear relationship, different robot configurations result in force polytopes with different shapes. Figure 1 illustrates this trait: two redundant configurations, q_1 and q_2, reach the same end-effector target, but their respective force polytopes, P_1 and P_2, have distinct shapes.

4. Residual Force Polytope

In subsection 3.3 we have reviewed what a force polytope is and how it results from the mapping of the actuation limits of a robot into the task-space. The force polytope is limited to quasi-static scenarios and, besides the kinematic configuration of the robot, it does not take into account any information about the task being performed.

We propose a new representation called the *residual force polytope*, which takes the dynamics of the robot into account, as well as the nominal forces and torques required by a task. We define the residual forces and torques of a robot state as the difference between the absolute actuation limits and a given vector of joint forces and torques. Residual forces and torques are important to deal with disturbances, as they represent the control authority left in a system after accounting for the task at hand. The residual force polytope is the result of transforming those residual forces and torques with $J_e^{-\top}$, similarly to equation (2).

Figure 2 shows the relationship between forces/torques in actuation-space and forces in task-space. For convenience of illustration, it displays a planar manipulator with three degrees of freedom in joint-space and two-dimensional task-space forces. The figure highlights how the residual force polytope P_3 is obtained for a given configuration q_1.

5. Modeling Force Uncertainty

As we have seen so far, polytopes are useful to represent and model regions of interest in space. But in addition to this, we may want to extract a single metric that quantifies one of those regions. For example, given a configuration q_1 and its corresponding force polytope P_1, we may want to know how robust that configuration is with respect to forces applied at the end-effector of the robot.

5.1. Largest Ball Inscribed in a Polytope

One way to tackle this problem is to consider the worst-case scenario, i.e., the situation with most uncertainty, where a force could originate from any given direction. In order to represent this uncertainty, we can use a ball to model a set of forces with any given direction and with a magnitude ranging from $0\,\mathrm{N}$ to the radius of the ball. Then, if we constrain the ball to be centered at the end-effector, and maximize the size of the ball without exceeding the boundaries of the force polytope, we obtain the set of all forces that the robot is able to deliver without saturating its torque limits. Consequently, the radius of this ball denotes the magnitude of the greatest force that the robot can counteract, and it can be used as a metric for isotropic robustness of a configuration.[1] For example, both q_1 and q_2 shown in Figure 1 solve the same reaching task, but q_2 is more robust than q_1 because $r_2 > r_1$.

The center of the largest ball inscribed in a bounded set of non-empty interior is known as the *Chebyshev center* [13]. We can find the Chebyshev center by solving the following Linear Programming (LP) problem:

$$\underset{\hat{x}, r}{\mathrm{argmin}} \left\{ r : \|\hat{x} - x\|^2 \le r, \forall x \in P \right\}, \tag{4}$$

where $\hat{x}$ is the Chebyshev center of a polytope P, and r is the radius of the ball. In our work, we are interested in a problem similar to (4), but where the center of the ball is constrained to lie at the origin of the end-effector frame.

5.2. Largest Intersection with a Polytope

The previous subsection demonstrated how to calculate the robustness of a robot to completely unknown external disturbances. However, there are cases where the interaction between

---

[1]In light of directional uncertainty, an isotropic robustness metric is more useful than other general quantities like the overall volume of a polytope.
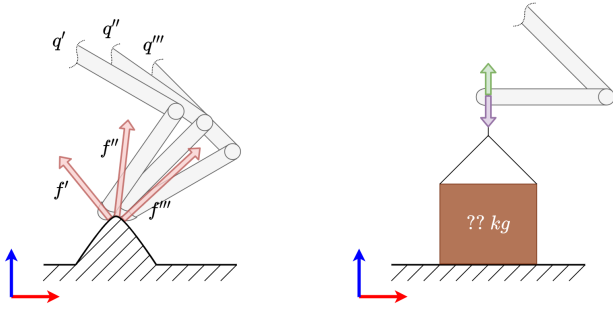
(a) For legged robots locomoting on complex terrains, the direction of the terrain normals can change greatly with small variations in the contact location, leading to very different contact forces applied to the feet.

(b) In the context of a manipulation task, the direction of action/reaction forces for lifting a box attached to a rope is well-known *a priori*, and regardless of the weight of the box.

Figure 3: Illustration of two different levels of uncertainty concerning the direction of interaction forces for two real-world scenarios. On the left, factors such as controller tracking errors or noisy state estimation can ultimately lead to inaccurate foot placement, which in turn, and depending on the terrain, can induce forces applied in unexpected directions (high direction uncertainty). In contrast, on the right, the forces are expected to be close-to-vertical due to the nature of the task (low direction uncertainty).

the robot and its environment is not fully uncertain. As an example, consider a task where the robot needs to open or close a door of unknown mass: the robot may not know *a priori* how much force is needed to solve the task, but the door can only open or close in a specific way—see Figure 3 for a further example. The direction of the interaction can therefore be exploited to our advantage. We can use a cone to model the set of forces originating from some expected direction and applied to the end-effector: the cone axis is aligned with the expected direction, the cone apex is fixed at the end-effector frame, and the aperture of the cone represents the prediction uncertainty of the force direction. Then, if we intersect the cone with a force polytope, we obtain a subset of the forces in the cone which the robot can cancel out within its actuation limits. Consequently, the volume of the resulting intersection is proportional to how much the robot is capable of resisting forces modeled by the cone, and it can be used as a surrogate metric of robustness to expected forces. An example of modeling expected forces using this approach is illustrated in Figure 4, where the intersection of a cone C_1 with a residual force polytope P_3 results in the purple polygon P_4, i.e., $P_4 = P_3 \cap C_1$.

In this section, we showed that redundant configurations result in different capabilities to counteract external forces applied to the end-effector. We proposed a representation for modeling those capabilities, and discussed two robustness metrics that can be extracted from it. In the next section, we will demonstrate how to formulate a trajectory optimization problem with objective functions that employ those metrics in order to plan dynamic motions more robust to unexpected forces through exploitation of kinematic redundancy.
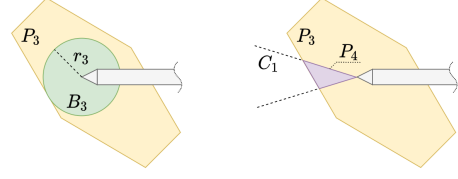

Figure 4: Two distinct models for representing force disturbances: the ball B_3 models unexpected forces, whereas the cone C_1 models the direction of an expected force. The aperture of the cone is proportional to the uncertainty of the force direction. The purple polygon P_4 results from the intersection of the *residual force polytope* with the cone, i.e., $P_4 = P_3 \cap C_1$.

6. Optimization of Robust Trajectories

Trajectory optimization is a process that allows to compute control trajectories as functions of time that drive a system from an initial state towards a final state while satisfying a given set of constraints [14]. In robotics, the problem is a second-order dynamical system governed by the equations of motion (1).

Direct transcription [15] is a popular approach within trajectory optimization and works by transcribing a continuous problem into a constrained nonlinear optimization problem by means of explicit discretization of the state and control trajectories. The result of this transcription is the formulation of a large and sparse nonlinear problem which can be solved using a large-scale nonlinear programming solver [14].

We have chosen direct transcription to demonstrate how the residual force polytope can be used to plan robust and dynamic trajectories for the following reasons:

- Discretization of both states and controls is particularly convenient for computing polytope representations;

- Defining general state and path constraints using direct transcription is more straightforward than alternatives;

- Its simplicity of formulation and implementation.

6.1. Problem Formulation

We divide the trajectory into N equally spaced segments

$$t_I = t_1 < t_2 < \cdots < t_M = t_F, \qquad (5)$$

where t_I and t_F are the start and final instants, respectively. Thus, the number of discretized *mesh points* is $M = N + 1$. Let $x_k \equiv x(t_k)$ and $u_k \equiv u(t_k)$ be the values of the state and control variables at the k-th mesh point, respectively. We treat $x_k \triangleq \{q_k, v_k\}$ and $u_k \triangleq \{\tau_k\}$ as a set of nonlinear programming variables, and formulate the trajectory optimization problem as:

$$
\begin{aligned}
\underset{\xi}{\text{argmin}} \quad & \sum_{k=1}^{M} g(x_k, u_k) \\
\text{subject to} \quad & \dot{x} = f(x, u) \\
& x_k \in \mathcal{X} \\
& u_k \in \mathcal{U}
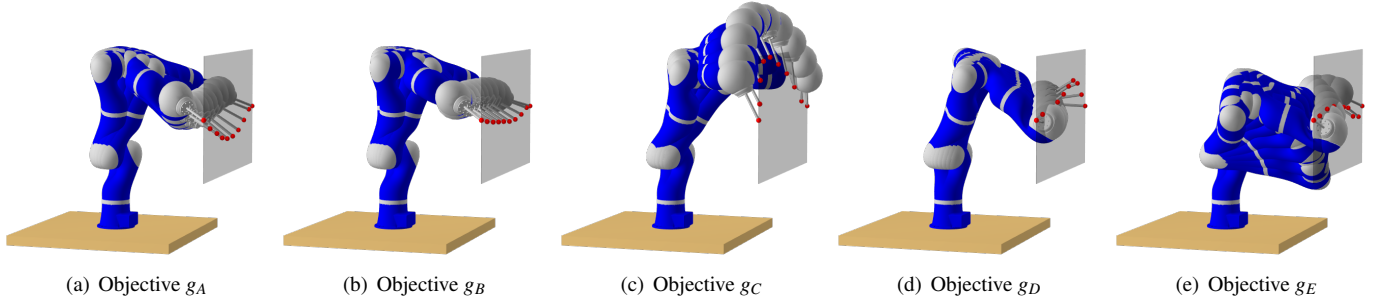\end{aligned}
\qquad (6)
$$

4

Figure 5: Visualization of the trajectories obtained using the interior-point method without a payload. The configuration samples are equally spaced in time.

where $\boldsymbol{\xi}$ is the vector of decision variables, $g(\cdot, \cdot)$ is a cost function, $\dot{x} = f(x, u)$ gives the nonlinear dynamics of the system, and $\mathcal{X}$ and $\mathcal{U}$ are sets of feasible states and control inputs enforced by a set of equality and inequality constraints. The vector of decision variables $\boldsymbol{\xi}$ results from aggregating the generalized coordinates, generalized velocities, and control inputs of every mesh point:

$$\boldsymbol{\xi} \triangleq \{\boldsymbol{q}_1, \boldsymbol{v}_1, \boldsymbol{\tau}_1, \cdots, \boldsymbol{q}_N, \boldsymbol{v}_N, \boldsymbol{\tau}_N, \boldsymbol{q}_M, \boldsymbol{v}_M\}.^2 \qquad (7)$$

6.2. Constraints

We want to optimize trajectories that are consistent with the full dynamics of the robot, do not exceed the kinematic and actuation limits of the robot, and use the end-effector for a given task. We formulate all these requirements as equality and inequality constraints which the solver must respect.

6.2.1. End-effector Task

The exemplar task we use for this evaluation is to move the end-effector of a multi-DoF robot arm from an initial point $\boldsymbol{p}_I$ to a final point $\boldsymbol{p}_F$:

$$f_{\text{FK}}(\boldsymbol{q}_1) = \boldsymbol{p}_I \quad \text{and} \quad f_{\text{FK}}(\boldsymbol{q}_M) = \boldsymbol{p}_F \qquad (8)$$

where $f_{\text{FK}}(\cdot)$ is the forward kinematics function. In addition, the end-effector must always lie on a rectangular surface R positioned in its workspace:

$$f_{\text{dist}}(R, f_{\text{FK}}(\boldsymbol{q}_k)) = 0 \qquad \forall k = 1 : M \qquad (9)$$

where $f_{\text{dist}}(\cdot)$ is the distance between a surface and a point. This task is analogous to drawing a line on a whiteboard using a marker attached to the end-effector, where the initial and final points are given and the path taken by the end-effector does not matter as long as it does not lift the tip of the marker off from the surface of the whiteboard.

---

[2]The control inputs at the final state $\boldsymbol{\tau}_M$ need not be discretized.

6.2.2. System Dynamics

We enforce the nonlinear dynamics of the system with a finite set of *defect* constraints:

$$x_{k+1} - (x_k + h \cdot f(x_k, u_k)) = 0 \qquad (10)$$

For simplicity of exposition, we integrate the differential equations of the dynamics using the explicit Euler method, where $h = (t_F - t_I)/N$ is the integration time step.

6.2.3. Initial and Final Joint Velocities

We enforce the initial and final velocities of every joint to be zero with

$$\boldsymbol{v}_1 = \boldsymbol{v}_M = \boldsymbol{0}. \qquad (11)$$

6.2.4. Bounds of the Decision Variables

We constrain the joint positions, velocities, and torques to be within their corresponding lower and upper bounds:

$$\boldsymbol{q}_{\text{lb}} \leq \boldsymbol{q}_k \leq \boldsymbol{q}_{\text{ub}} \qquad \forall k = 1 : M \qquad (12)$$
$$\boldsymbol{v}_{\text{lb}} \leq \boldsymbol{v}_k \leq \boldsymbol{v}_{\text{ub}} \qquad \forall k = 1 : M \qquad (13)$$
$$\boldsymbol{\tau}_{\text{lb}} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_{\text{ub}} \qquad \forall k = 1 : M - 1 \qquad (14)$$

6.3. Objectives

There are many objective functions which could be used to achieve different optimal results under the same problem constraints. We will now list some well-known objectives as well as our own. Later, in our experiments, we will compare the obtained trajectories against each other in terms of their robustness, torque expenditure, and computation time.

It is typical in optimal control to use energy as a cost, and this is usually formulated as a minimization of torques:

$$g_A : \quad \min_{\boldsymbol{\xi}} \quad \sum_{k=1}^{M} \boldsymbol{\tau}_k^{\top} \boldsymbol{\tau}_k \qquad (15)$$

In order to avoid torque saturation, we can define a simple objective function to maximize residual actuator torques:

$$g_B : \quad \max_{\boldsymbol{\xi}} \quad \sum_{k=1}^{M} (\boldsymbol{\tau}_{\text{lim}} - \boldsymbol{\tau})^{\top} (\boldsymbol{\tau}_{\text{lim}} - \boldsymbol{\tau}) \qquad (16)$$
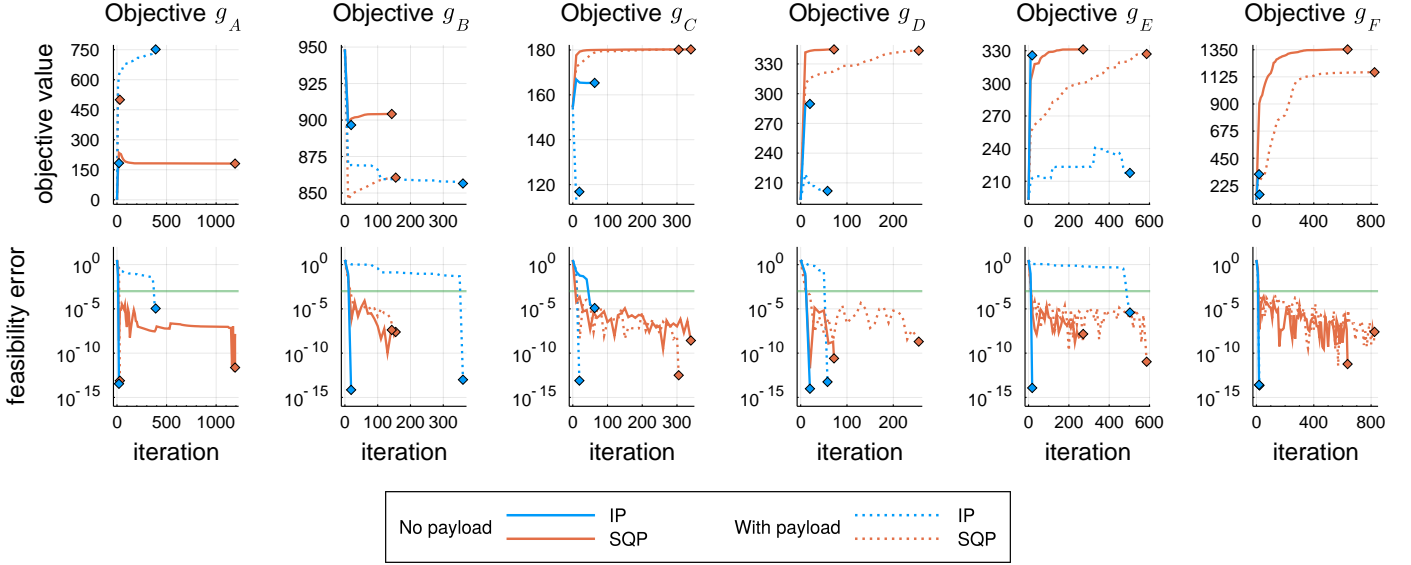
Figure 6: These plots show, for each function g_A–g_F, the evolution of the objective value and the feasibility error along the solver iterations. In the feasibility plots, the faint-green line at $y = 10^{-3}$ denotes the absolute tolerance under which a problem is considered feasible. We can see that all metrics were able to handle the payload. We can also see the back-and-forth progression of feasibility error for the SQP method due to excessive pivoting.

Yoshikawa [7] defined a quantitative measure of manipulability as $w = \sqrt{\det(\boldsymbol{J}_e \boldsymbol{J}_e^\top)}$. Later, Chiacchio *et al.* [12] proposed a more accurate definition by scaling the joint forces with $\boldsymbol{W} = \mathrm{diag}(1/\tau_{1,\mathrm{lim}}, \cdots, 1/\tau_{n,\mathrm{lim}})$, which allowed to define a scaled Jacobian $\boldsymbol{J}_e'^\top = \boldsymbol{W} \boldsymbol{J}_e^\top$ and a more accurate measure of manipulability $w' = \sqrt{\det(\boldsymbol{J}_e' \boldsymbol{J}_e'^\top)}$. For our formulation, we can maximize the manipulability of every configuration in a discretized trajectory with the following objective:

$$g_C : \quad \max_{\boldsymbol{\xi}} \quad \sum_{k=1}^{M} w_t' \tag{17}$$

We can also define objectives with metrics extracted from polytopes. Let us denote the force polytope of a configuration as $P_k \equiv P(\boldsymbol{q}_k)$. Similarly to [4], and assuming static equilibrium, we can maximize the robustness to external forces from any given direction with:

$$g_D : \quad \max_{\boldsymbol{\xi}} \quad \sum_{k=1}^{M} B_r(P_k) \tag{18}$$

where $B_r(\cdot)$ denotes the radius of the largest ball centered at the end-effector and inscribed in the given polytope.

For the dynamic scenario, let us consider the residual force polytope as $P_k' \equiv P'(\boldsymbol{q}_k, \boldsymbol{\tau}_k)$, which is the novel representation we propose in this paper. Analogous to (18), we can maximize the largest ball centered at the end-effector and inscribed in P_k' for every mesh point with:

$$g_E : \quad \max_{\boldsymbol{\xi}} \quad \sum_{k=1}^{M} B_r(P_k') \tag{19}$$

The last objective function we consider in this work is the intersection of the residual force polytope with a cone that models an expected force but with some level of uncertainty—we

proposed this in subsection 5.2. An objective function that maximizes the robustness in this scenario is:

$$g_F : \quad \max_{\boldsymbol{\xi}} \quad \sum_{k=1}^{M} P_{\mathrm{vol}}(P_k' \cap C_k) \tag{20}$$

where $P_{\mathrm{vol}}(\cdot)$ denotes the volume of a given polytope, and $C_k \equiv C(t_k)$ is a cone modeling a disturbance at instant t_k.

7. Experimental Results

Using a *KUKA LWR* robot arm with 7-DoF, we solved the optimization problem formulated in the previous section for each of the objective functions g_A–g_F without changing the problem constraints. We considered 1 s trajectories divided into 10 equally spaced segments (11 mesh points). Figure 5 shows the motion trace of the resulting trajectories. We used Julia [16] to implement our trajectory optimization framework, and the library Knitro [17] to solve the nonlinear optimization problems.

In this section, we first compare the performance of two state-of-the-art optimization methods for solving the problem we formulated. Afterwards, we compare the obtained trajectories against each other in terms of their robustness, simulated torque expenditure, and computation time. All evaluations were carried out in a single-threaded process on an Intel i7-6700K CPU at 4.0 GHz and with 32 GB 2133 MHz memory.

7.1. Interior-Point vs. Active-Set Methods

Methods to solve constrained nonlinear optimization problems can be broadly categorized in two classes by how they handle constraints: *interior-point* (IP) methods incorporate the constraints into the objective (e.g., via a barrier function or an augmented Lagrangian), while *active-set* methods formulate a tractable model (e.g., by linearizing part of the constraints

Table 1: Convergence times in seconds. Mean and standard deviation were calculated from 4 samples.

		g_A (s)	g_B (s)	g_C (s)	g_D (s)	g_E (s)	g_F (s)
No payload	Interior Point	0.07 ± 0.01	0.08 ± 0.01	0.11 ± 0.01	150.46 ± 0.28	261.66 ± 3.42	384.28 ± 0.61
	Active Set	64.50 ± 0.85	22.51 ± 0.31	18.21 ± 0.21	590.44 ± 5.27	3698.82 ± 9.99	$14\,354.95 \pm 88.38$
With payload	Interior Point	0.82 ± 0.02	0.24 ± 0.01	0.11 ± 0.01	1529.20 ± 2.97	1120.90 ± 1.71	512.91 ± 1.10
	Active Set	14.44 ± 0.32	25.87 ± 0.47	20.99 ± 0.54	2177.93 ± 40.91	8195.97 ± 184.67	$19\,902.78 \pm 369.43$

Table 2: Number of function evaluations and gradient evaluations of the problem constraints.

		Number of function evaluations						Number of gradient evaluations					
		g_A	g_B	g_C	g_D	g_E	g_F	g_A	g_B	g_C	g_D	g_E	g_F
No payload	Interior Point	509	484	1734	571	952	953	21	20	64	21	19	19
	Active Set	34629	4084	11066	2285	15041	34909	1192	144	341	73	272	637
With payload	Interior Point	9399	8686	542	1594	25217	1104	391	361	20	59	504	22
	Active Set	787	4829	9992	8470	32655	45864	30	156	306	256	587	825

and penalizing them as well in the objective, as done with Sequential Quadratic Programming (SQP) algorithms). For highly nonlinear problems, SQP methods are known to suffer from excessive pivoting requiring expensive gradient evaluations of the constraints to update the active-set. As such, they are said to scale poorly to systems with a large number of constraints. As a result, in robotics literature, IP methods are commonly used for direct transcription and collocation [18, 19], while some rely on SQP-based solvers [20]. However, few related work compare IP and SQP methods for solving equivalent problems, with the notable exception of [21]. In this subsection, we compare the performance of state-of-the-art, commercial large-scale sparse IP and SQP methods on the equivalent direct transcription problem (6) for all objective functions g_A–g_F. This emphasizes the differences between classical IP and SQP for direct transcription applications.

We used the SQP and IP method provided by [17]. For all comparisons and either method, we used automatic differentiation to obtain the Jacobian of the constraints, finite-differencing for the gradients of the objectives, and L-BFGS[3] for Hessian approximation (with 10 limited memory pairs). The results are presented in Figure 6, Table 1, and Table 2. In Figure 6, we can see that the interior-point method required very few iterations to converge when compared with the active-set method. As shown in Table 1, the total amount of time taken to find a locally optimal solution by the interior-point method was significantly less than the active-set method. In Table 2, we can see that the active-set method required significant more function and gradient evaluations than the interior-point method for the majority of the objective functions, which is expected and related to SQP's excessive pivoting (clearly observable in the feasibility error plots of SQP in Figure 6).

7.2. Robustness to External Disturbances

We want to evaluate each trajectory's ability to counteract external forces while executing its planned motion. As such,

we first consider the torques required by the planned motion, and then calculate the set of all admissible forces from the remaining torques available. We define our evaluation metric as the magnitude of the maximum admissible force considering all possible force directions. Therefore, each trajectory is evaluated as follows: for each point, (i) compute the residual force polytope, then (ii) find the largest ball centered at the end-effector inscribed in that polytope, and (iii) take the radius of the ball as the robustness metric.

7.2.1. Overview of all objective functions

Figure 7 shows the evaluation results considering all objectives g_A–g_E. In the plot, greater values correspond to greater robustness against unpredicted forces. The trajectory computed with the *residual force polytope* resulted in greater robustness than any other objective function considered.[4]

7.2.2. Force Polytope vs. Residual Force Polytope

Figure 8 shows the evaluation results for a scenario without a payload and for a scenario with a 2 kg cylindrical payload. The results in the plot correspond to trajectories obtained using g_D and g_E. We can see that the objective using the *residual force polytope* provided a significant improvement over the traditional force polytope; more specifically, for the 1-second long trajectories we computed, an improvement of 53.2 ± 6.52 N without payload, and 40.55 ± 21.37 N with the 2 kg payload.

7.3. Unexpected Forces vs. Expected Forces

In this experiment, we want to compare the torque required by the trajectories optimized using objectives g_E and g_F, which optimize a motion for resisting forces from any given direction

---

[3]Limited-memory quasi-Newton BFGS.

---

[4]The trajectory computed with g_C for the scenario with the payload and using the interior-point method resulted in an initial configuration with greater robustness than the other objective functions. However, we are interested in the robustness overall during the trajectory (area under the curve) and, for that, the objective function g_E defined as a function of the *residual force polytope* performed best.

(a) Interior-Point method.
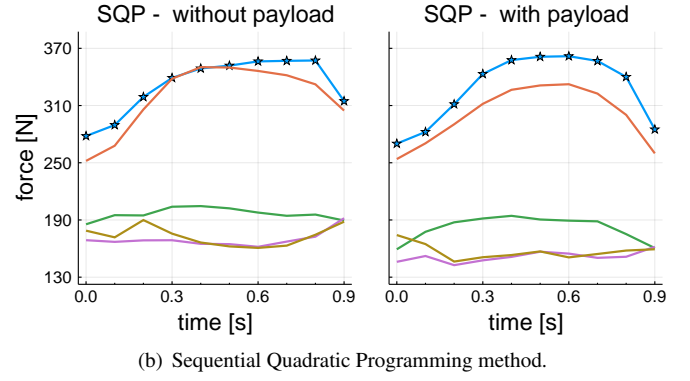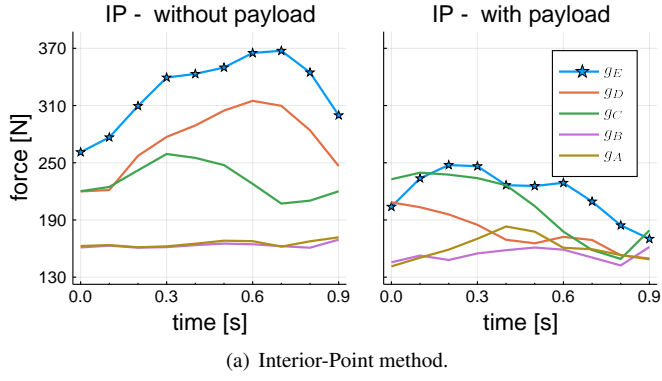
(b) Sequential Quadratic Programming method.

Figure 7: These plots show the maximum admissible force magnitudes over time of trajectories computed using objective functions g_A–g_E. We can see that the objective function g_E, which uses the *residual force polytope*, resulted in greater admissible magnitudes than any other objective function.
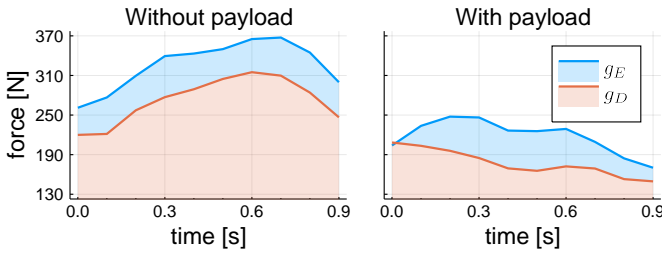

Figure 8: These plots show the magnitude of forces applied to the end-effector from any given direction which the robot is able to cancel out given its actuation limits. The solid lines represent the maximum admissible magnitude over time, and the shaded areas represent the magnitudes in between zero (no disturbance at all) and the maximum admissible magnitude. We can see that using the *residual force polytope* (shown in blue) provided a significant improvement over the classical force polytope (shown in red).

and from a specific direction, respectively. More specifically, we want to determine how much torque the robot would need to complete a planned motion while, at the same time, resisting an external force applied to its end-effector. In order to do that, we apply an impulse to the robot and, for each point of the trajectory, we compute the extra torques required to oppose the external force with equation (2). The magnitude of the force applied to the robot at each instant is given by $f(t) = f_{\text{peak}} \cdot \exp(-(t - 0.5)^2/0.02)$, where f_{peak} defines the magnitude at the peak of the impulse. The profile of this test force is shown in Figure 9.
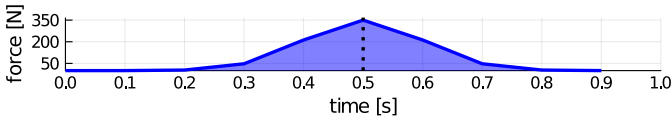

Figure 9: Profile of the test force applied to the end-effector. The impulse of this force is 87.73 N s and the peak magnitude is 350 N (at $t = 0.5$ s).

In order to compare optimal resistance to forces from any given direction (g_E) against optimal resistance to forces from a specific direction (g_F), we compute the torque required by the optimized trajectories for a test impulse that matches the direction estimation used during optimization of the specialized tra-

jectory with g_F. Afterwards, we invert the direction of the impulse and repeat the test to compute the required torques again. The results are shown in Figure 10.
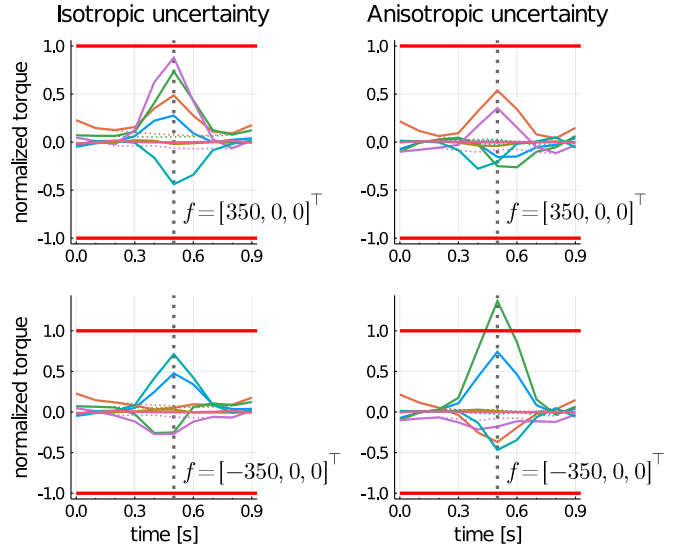

Figure 10: Joint torques required to complete the planned task and resist the disturbance. The torques have been normalized to $[-1, 1]$ according to actuation limits (solid red horizontal lines). The nominal torques are shown as dotted lines. The left and right columns correspond to the trajectories computed with g_E and g_F, respectively. On the left, we can see that the limits were not exceeded. On the top right, the trajectory resisted the impulse with less torque than g_E (this impulse was applied in the same direction as the estimation during optimization). On the bottom right, when we applied the impulse in the opposite direction to what the specialized trajectory expected, the torques required exceeded the actuation limits of the robot.

7.4. Summary of Computational Runtime

Table 3 shows the average time required to evaluate each of the objective functions per solver iteration. It is clear that the objectives defined as functions of polytopes take significantly longer to evaluate than the other objective functions tested.

Table 4 shows the average time required to compute: a force polytope, a residual force polytope, the largest ball inscribed in a polytope, the intersection of two polytopes, and the volume of

8

Table 3: Time required to evaluate each objective function once. Calculated from 10 samples.

g_A(ms)	g_B(ms)	g_C(ms)	g_D(ms)	g_E(ms)	g_F(ms)
0.007 ± 0.029	0.012 ± 0.002	0.023 ± 0.096	85.004 ± 10.342	72.565 ± 7.209	102.036 ± 9.069

a polytope. These methods are considerably expensive and are the reason why objectives g_D–g_F take so much time to evaluate.

Table 4: Time benchmark of computational geometry methods.

Operation	Time (µs)
Force polytope	22 ± 124
Residual force polytope	24 ± 146
Largest inscribed ball	7354 ± 2405
Polytope intersection	7081 ± 2049
Polytope volume	6312 ± 1923

8. Discussion

Our initial hypothesis was that optimizing trajectories with an objective defined as a function of admissible forces in task-space—after accounting for the torques required by the motion itself—would result in motion plans more robust to external disturbances. We defined an objective function based on the *residual force polytope* to optimize a trajectory robust to forces from any given direction, and compared it against other objective functions commonly used in trajectory optimization, such as torque minimization, and manipulability maximization. The results we obtained support our initial hypothesis: as shown in Figure 7, for both the interior-point and active-set methods tested, the objective function g_E we propose leads to optimal trajectories that are able to counteract forces from any direction with greater magnitude than any other objective function we explored. Moreover, the objective function g_F, which optimizes trajectories specialized in specific directions, leads to even more robust motion plans than g_E if the disturbance is applied approximately in the same direction as the one considered for the specialization. However, specialized trajectories are less robust if the direction taken into account during optimization does not match the actual disturbance accurately (case shown in the bottom right plot of Figure 10). Therefore, in terms of robustness, if a disturbance originating from a completely unexpected direction is not out of question, the objective considering any given direction (g_E) should be preferred over the optimization of a specific direction (g_F). On the other hand, any accurate bias about disturbance directions that may arise out of known environmental constraints (e.g., axis of fixation of articulated objects being manipulated) should be incorporated into g_E to allow more dynamic range of motion.

Despite the promising results in terms of robustness, the objective functions we proposed are very demanding computationally: even though we used a coarse problem discretization, all the objectives defined as functions of polytopes took at least 3 orders-of-magnitude longer to converge than the simpler objective functions g_A–g_C. This significant difference is due to the double description method required to convert across polytope representations as discussed in subsection 3.1, and due to the other mathematical operations involving polytopes (benchmarked in Table 4). Nonetheless, the objectives g_E and g_F utilizing the residual force polytope representation did not incur significant convergence time differences compared to objective g_D using the traditional force polytope.

8.1. On the Scalability of Our Metric

We did not carry out experiments using different robot arms. While the absolute values shown in our results will vary across different manipulators, we speculate that the relative differences observed should generalize to manipulators of different sizes and more or less joints.

Regarding the scalability of our approach to floating-base robots—such as quadrupeds or bipeds—there is a distinction to be made: whether the metric is to be used as an evaluation metric for existing trajectories, or if it is to be used as an objective function in a trajectory optimization setting.

Robustness as an evaluation metric. Given an existing dynamic trajectory, computing the residual force polytope for each point in time is straightforward. A possible application for this is to evaluate the robustness of different trajectories, and to compare them against each other. In fact, this is exactly what we did in subsection 7.2 in order to evaluate the robustness of the motions obtained from the optimization of different objective functions. For this use-case, our metric should be scalable to different platforms, but it will become more computationally demanding—and therefore slower—as the degrees of freedom of the system increase: the number of vertices of the polytope grows with the number of degrees of freedom of the system, and the complexity of converting representations (from $\mathcal{V}$-rep to $\mathcal{H}$-rep, or vice-versa) grows with the number of vertices.

Robustness as an objective function. In the context of trajectory optimization, using our metric as an objective function for floating-base systems with many degrees of freedom is not straightforward and presents significant scalability issues. The reason for this is related (but not limited) to the point mentioned above: computing the residual force polytope becomes more demanding and slower as the number of degrees of freedom of the robot increases. For purposes of evaluating a trajectory, the polytope only needs to be computed once for each mesh point. On the other hand, in trajectory optimization, the solver takes several iterations (in our case, hundreds of iterations) while converging to a locally optimal solution, and for each of those iterations it may need to perform more than one function or gradient evaluation, which requires computing the residual force polytope again and again. As a consequence, optimizing trajectories for high-DoF robots in a reasonable amount of time is not possible, and could take multiple days to complete. We would like to emphasize that this is not a limitation of the residual

force polytope we propose, but a limitation of using any poly-tope. Since this is a well-known issue, other authors have tried to use approximations to work around it. For example, the optimization of the center-of-mass position of a quadruped's static crawl gait shown in [4] calculated the feasibility polytope once at the beginning of the optimization and used that as a constant approximation thereafter. Another approach was used in [22], where the authors employed polytope morphing for each leg of a quadruped: they computed an exact polytope representation for two key configurations, and then approximated the polytope for intermediate configurations by interpolating its shape. To conclude, in trajectory optimization, objective functions which require computing exact polytope representations are not scalable to high-DoF floating-base robots, but this issue can be tackled by exploring approximate computations of polytopes.

9. Conclusions

In this paper we proposed the *residual force polytope*, a representation for admissible task-space forces taking into account the nominal torques of dynamic trajectories, and defined two functions based on that representation for two different levels of disturbance uncertainty. We employed those functions as objectives in trajectory optimization to plan motions highly robust to external disturbances. Despite the qualitative benefits of the resulting trajectories, their computational cost makes them unsuitable for deployment as online planning methods. Nonetheless, they are appropriate for offline planning and can also be applied in areas besides trajectory optimization, such as system analysis and co-design.

Addressing the long computation times required by using polytope-based methods is an interesting direction for future work. In this paper, we used explicit polytope descriptions, but with recent developments in [23], it may be possible to use approximate descriptions to considerably decrease the computational cost of planning methods based on polytopes. For that, choosing the right level of approximation becomes an important decision and the trade-off between speed and accuracy will need to be investigated carefully.

References

[1] A. Del Prete, N. Mansard, Addressing Constraint Robustness to Torque Errors in Task-Space Inverse Dynamics, in: Robotics, Sciences and Systems (RSS), Rome, Italy, 2015.

[2] M. Plooij, W. Wolfslag, M. Wisse, Robust feedforward control of robotic arms with friction model uncertainty, Robotics and Autonomous Systems 70 (2015) 83 – 91.

[3] G. Xin, H.-C. Lin, J. Smith, O. Cebe, M. Mistry, A Model-Based Hierarchical Controller for Legged Systems Subject to External Disturbances, in: IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 4375–4382.

[4] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell, C. Semini, Application of Wrench-Based Feasibility Analysis to the Online Trajectory Optimization of Legged Robots, IEEE Robotics and Automation Letters 3 (4) (2018) 3363–3370.

[5] Z. Manchester, S. Kuindersma, DIRTREL: Robust Nonlinear Direct Transcription with Ellipsoidal Disturbances and LQR Feedback, in: Robotics, Sciences and Systems (RSS), 2017.

[6] H. Ferrolho, W. Merkt, Y. Yang, V. Ivan, S. Vijayakumar, Whole-Body End-Pose Planning for Legged Robots on Inclined Support Surfaces in Complex Environments, in: IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), 2018, pp. 944–951.

[7] T. Yoshikawa, Manipulability of Robotic Mechanisms, The International Journal of Robotics Research (IJRR) 4 (2).

[8] H. Audren, A. Kheddar, 3D Robust Stability Polyhedron in Multicontact, IEEE Transactions on Robotics (T-RO).

[9] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, M. Morisawa, A Universal Stability Criterion of the Foot Contact of Legged Robots - Adios ZMP, in: IEEE International Conference on Robotics and Automation (ICRA), 2006.

[10] G. M. Ziegler, Lectures on polytopes, Vol. 152, Springer Science & Business Media, 2012.

[11] K. Fukuda, A. Prodon, Double Description Method Revisited, in: Combinatorics and Computer Science (CCS), Springer, 1995.

[12] P. Chiacchio, Y. Bouffard-Vercelli, F. Pierrot, Force Polytope and Force Ellipsoid for Redundant Manipulators, Journal of Robotic Systems 14 (8) (1997) 613–620.

[13] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[14] J. T. Betts, Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, 2nd Edition, Vol. 19, Society for Industrial and Applied Mathematics (SIAM), 2010.

[15] O. von Stryk, R. Bulirsch, Direct and Indirect Methods for Trajectory Optimization, Annals of Operations Research.

[16] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, Julia: A Fresh Approach to Numerical Computing, SIAM Review 59 (1) (2017) 65–98.

[17] R. H. Byrd, J. Nocedal, R. A. Waltz, Knitro: An Integrated Package for Nonlinear Optimization, Springer US, 2006.

[18] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, S. Vijayakumar, Dyadic collaborative Manipulation through Hybrid Trajectory Optimization, in: Proceedings of The 2nd Conference on Robot Learning (CoRL), Vol. 87, PMLR, 2018, pp. 869–878.

[19] A. W. Winkler, C. D. Bellicoso, M. Hutter, J. Buchli, Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization, IEEE Robotics and Automation Letters (RA-L) 3 (3) (2018) 1560–1567.

[20] M. Posa, C. Cantu, R. Tedrake, A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact, The International Journal of Robotics Research (IJRR) 33 (1) (2014) 69–81.

[21] A. W. Winkler, Optimization-based motion planning for legged robots, Ph.D. thesis, ETH Zurich, Zurich (2018).

[22] A. Bratta, R. Orsolino, M. Focchi, V. Barasuol, G. G. Muscolo, C. Semini, On the Hardware Feasibility of Nonlinear Trajectory Optimization for Legged Locomotion based on a Simplified Dynamics, in: IEEE International Conference on Robotics and Automation (ICRA), 2020.

[23] J. Zhen, D. den Hertog, Computing the Maximum Volume Inscribed Ellipsoid of a Polytopic Projection, INFORMS Journal on Computing 30 (1) (2018) 31–42.