# Comparing Metrics for Robustness Against External Perturbations in Dynamic Trajectory Optimization

Henrique Ferrolho, Wolfgang Merkt, Carlo Tiseo, and Sethu Vijayakumar

*Abstract*— Dynamic trajectory optimization is a popular approach for generating optimal and dynamically consistent trajectories. In order to deal with model errors and perturbations, the trajectories are usually tracked with feedback controllers. Their robustness thus largely depends on the *margins of stability* and control authority the system retains. Manipulability ellipsoids and force polytopes are well-known tools for evaluating force and motion capabilities of robot manipulators. Increased control authority can be achieved by incorporating task constraints within those tools. However, they come with an increased computational cost. Additionally, their impact on resulting trajectory quality and control authority has not yet been benchmarked and compared. In this letter, we introduce a novel robustness metric, the *residual force polytope*, which takes the nominal torque required to maintain a posture into account. We further detail a benchmarking protocol including evaluation criteria and visualization tools to compare robustness metrics in dynamic trajectory optimization. To foster benchmarking and allow for reproducibility, we open source a flexible framework for dynamic trajectory optimization via direct transcription along with our benchmark protocols as supplementary materials. Finally, we include — to the best of our knowledge — the first holistic comparison between traditional energy minimization metrics, kinematic manipulability maximization, and force polytope methods.

## I. INTRODUCTION

Force-controlled robots are widely studied for safe and compliant interaction with people and their environment. Some examples include collaborative robots working safely in proximity and in direct cooperation with humans and legged robots navigating complex terrain of unknown properties. In these scenarios, the robots have to adhere to a number of constraints, for instance: their ability to enact forces at the support contacts, the contact properties such as friction, and their ability to maintain balance — all while remaining within actuation limits and completing the task objectives.

Efficient motion-planning methods that satisfy these criteria have been proposed for high-dimensional systems. Nevertheless, execution of synthesized trajectories with tracking controllers fails frequently due to errors/uncertainties in the assumptions and model, or as a result of unforeseen changes in the environment and perturbations. Error sources such as sensor noise, signal delay, and model mismatches are often addressed with a feedback controller. Regardless, the ability
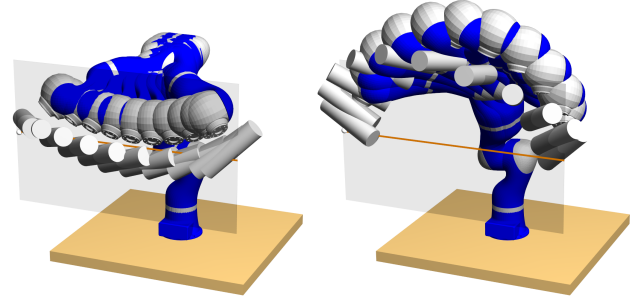
Fig. 1: Dynamic motions of a robot carrying a heavy load.

to handle larger perturbations mainly depends on the planned trajectory and the remaining control authority. While it is hard to predict and model some uncertainty during planning, information about the capabilities and limitations of a system can be used to adapt motions and increase their robustness.

### A. Related work

Improving robustness generally focuses on two different approaches: (a) increasing robustness in the control when executing a nominal motion plan, e.g., [1]–[3], and (b) taking uncertainty and robot capability into account at the planning stage in the form of objectives, and giving (robust) controllers larger margins of capability to achieve robustness [4], [5]. For the latter approach, several metrics have been proposed to favor selection of more robust postures within the null-space of task constraints on highly-redundant systems. Yoshikawa [4] proposed a manipulability metric in order to take into account *kinematic reachability* based on the geometry of the robot. Zacharias *et al.* [6] extended this idea to build a *capability map* of a fixed-base manipulator using offline computation for fast online lookup. Vahrenkamp *et al.* [7] used an inversion of this reachability map for optimal base placement of a floating-base robot, and Yang *et al.* [8] extended this to account for obstacles by introducing a mapping from workspace occupation to configuration space. However, for tasks interacting with the environment, metrics relating to dynamic properties, e.g., forces, need to be considered.

To take into account the ability to apply and resist forces, the *force manipulability ellipsoid* based on the robot geometry has been proposed. Nonetheless, it is an approximation and is not able to incorporate known information or capture multiple constraints. To this end, polytope-based methods (e.g., wrench polytopes) have recently gained interest as they allow for the intersection with additional constraints or information, like bounded Center of Mass (CoM) accelerations, contact properties, or force directions. For example, Audren and Kheddar [9] extended 2D stability regions to

3D by accounting for possible CoM accelerations in order to achieve robust multi-contact stability in whole-body posture generation. Orsolino *et al.* [5] proposed two new six-dimensional bounded polytopes: (i) the Actuation Wrench Polytope (AWP), the set of all wrenches a robot can generate while considering its actuation limits; and (ii) the Feasible Wrench Polytope (FWP), the convex polytope resultant from the intersection of the Actuation Wrench Polytope (AWP) with the Contact Wrench Cone (CWC). The authors demonstrated the advantage of using a (constant) feasibility polytope to optimize the CoM trajectory in the $xy$-plane for the base-transfer motion of a quadruped. However, computing such complex polytopes efficiently remains an issue. As a result, using feasibility polytopes for kinematic or dynamic whole-body trajectory optimization has yet to be shown.

Concerning robustness at the control stage, Del Prete *et al.* [1], [10], [11] have thoroughly studied this subject: a method to test robust equilibrium that allows the robot not to lose contact in case of bounded force-tracking errors has been proposed in [10]. However, being robust at the control stage does not necessarily result in a robust execution overall if the commanded motion itself is not robust. In other words, a robust controller can only perform as intended given that the commanded motion plan does not drive the robot too close to the boundaries of its feasibility region — at those points, the robot loses part of its control authority and is, consequently, unable to take advantage of the control-level robustness. Manchester and Kuindersma [12] presented a new algorithm for robust trajectory optimization which explicitly takes into account disturbances and modeling errors in closed form at the trajectory planning stage. A key advantage of their method is that the resulting control trajectories do not saturate actuation limits (in the sense of *bang-bang* control) and leave margins of stability for LQR feedback control around the nominal trajectory. Regardless, combining robustness at the planning level with a robust control scheme remains an open challenge.

### B. Contribution

In this letter, we argue that by explicitly taking into account robot-specific capability at the planning stage, larger control authority margins can be ensured for responding to errors at the control stage. To this end, we:

1. Review and compare a number of previously proposed robustness objectives (Sections III-A, III-C, III-D, III-F);
2. Describe a simple — yet to the best of our knowledge unreported—metric, the *residual actuator torques* (III-B);
3. Propose a novel force polytope-based metric taking into account actuation requirements for gravity compensation and task achievement, the *residual force polytope* (III-E);
4. Detail a protocol for comparing robustness objectives along with evaluation criteria and metrics (Section IV);
5. Open source a flexible dynamic trajectory optimization framework for evaluating robustness objectives;
6. Report results on the described robustness objectives using the open sourced framework, along with parameters for reproducibility (cf. Section V).

With this letter, we aim to close a gap in literature as many robustness metrics have been put forward in isolation, and yet, often without being benchmarked against other existing approaches. We also hope that open sourcing our framework and benchmarking tools will aid the community in developing new robustness objectives and comparing them with other approaches with more ease in the future.

## II. PRELIMINARIES

Consider a fixed-base robot manipulator with $n$ degrees of freedom (DoF) and whose end effector operates in an $m$-dimensional task space. Let $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ be the Jacobian matrix of the mechanism's forward kinematics (FK). The forward dynamics of the system are then given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\Big(\boldsymbol{\tau} - \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})\Big) \qquad (1)$$

where $\boldsymbol{q} \in \mathbb{R}^n$, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, and $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ are the vectors of joint positions, velocities, and accelerations, respectively, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is a symmetric positive-definite mass matrix, and $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ are the combined forces of centripetal, Coriolis, gravity, and friction terms dependant on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$.

### A. Force Manipulability Ellipsoid

Yoshikawa [4] defined the quantitative measure of manipulability as a scalar value given by $w = \sqrt{\det(\boldsymbol{J}\boldsymbol{J}^T)}$. Later, Chiacchio *et al.* [13] proposed a more accurate definition of the force ellipsoid by scaling the joint forces with $\boldsymbol{W} = \text{diag}\left(1/\tau_1^{\text{lim}}, \cdots, 1/\tau_n^{\text{lim}}\right)$. This allowed to define a scaled Jacobian as $\boldsymbol{J}'^T = \boldsymbol{W}\boldsymbol{J}^T$ and in turn the definition of a more accurate measure of manipulability:

$$w' = \sqrt{\det(\boldsymbol{J}'\boldsymbol{J}'^T)}. \qquad (2)$$

### B. Force Polytope

The relationship between the force vector $\boldsymbol{f}_{\text{tip}} \in \mathbb{R}^m$ applied by the end effector and the joint torque vector $\boldsymbol{\tau} \in \mathbb{R}^n$ experienced at the joints can then be written as

$$\boldsymbol{\tau} = \boldsymbol{J}^T \boldsymbol{f}_{\text{tip}}. \qquad (3)$$

The joint force polytope [13] is the bounded polyhedron of the manipulator's upper and lower actuation limits. It is described by the $2n$ bounding inequalities

$$\tau_i^L \leq \tau_i \leq \tau_i^U \qquad i = 1, \cdots, n, \qquad (4)$$

where $[\tau_i^L, \tau_i^U]$ is the range of the feasible torque for the $i$-th joint. Moreover, the force polytope $\mathcal{F}$ is a bounded polyhedron whose vertex representation ($\mathcal{V}$-rep) results from the image of the vertices of the joint force polytope:

$$\boldsymbol{f}_{\text{tip}} = \boldsymbol{J}^{-T} \boldsymbol{\tau}. \qquad (5)$$

For the quasi-static scenario, the relation above is effectively mapping joint-space torques into task-space forces; and thus, the force polytope $\mathcal{F}$ can be thought of as the convex set of all the realizable forces by the end effector.

## C. Double Description Method

The force polytope $\mathcal{F}$ can also be described in terms of its halfspace representation ($\mathcal{H}$-rep). The problem of computing the $\mathcal{H}$-rep from the $\mathcal{V}$-rep, or vice versa, is called the representation conversion problem, and it can be solved by the double-description method [14], [15]. However, a representation conversion should be avoided if possible since this process is known to be very computational demanding.

## D. Chebyshev Center

The *Chebyshev center* of a polyhedron $\mathcal{P}$ is the center of the largest Euclidean ball that lies in $\mathcal{P}$ [16]. In other words, the Chebyshev center is the farthest point from any of the boundaries of the polyhedron $\mathcal{P}$. The Chebyshev center can be computed by solving the linear program (LP)

$$\operatorname*{argmin}_{\hat{\boldsymbol{x}},r} \left\{ r : \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 \le r, \forall \boldsymbol{x} \in \mathcal{P} \right\}, \tag{6}$$

where $\hat{\boldsymbol{x}}$ denotes the Chebyshev center of $\mathcal{P}$ and $r$ is the radius of the ball. Henceforth, we shall refer to the Chebyshev center of a polyhedron $\mathcal{P}$ and the radius of its associated Euclidean ball as $\boldsymbol{x}_{\text{cheb}}(\mathcal{P})$ and $r_{\text{cheb}}(\mathcal{P})$, respectively.

## E. Trajectory Optimization

In general, trajectory optimization is concerned with finding a control trajectory as a function of *time*, $u(t)$, that drives a system from an initial state $x_0$ towards a final state $x_T$:

$$\begin{aligned} \operatorname*{argmin} \quad & \int_0^T g(x_t, u_t)dt \\ \text{subject to} \quad & \dot{x} = f(x, u) \\ & x_0 = x_{\text{initial}}^* \\ & x_T = x_{\text{final}}^* \end{aligned} \tag{7}$$

where $g(\cdot)$ is an objective function, $x_t \in \mathcal{X}$ and $u_t \in \mathcal{U}$ are the state of the system and the control input at instant $t$ (and $0 \le t \le T$), $\dot{x} = f(x, u)$ describes the nonlinear dynamics of the system, $x_{\text{initial}}^*$ and $x_{\text{final}}^*$ are the desired initial and final states of the trajectory, and $\mathcal{X}$ and $\mathcal{U}$ are sets of feasible states and inputs, respectively. In robotics, this problem is often a second-order dynamical system (1) where $x \triangleq (\boldsymbol{q}, \dot{\boldsymbol{q}})$. Trajectory optimization can be used to obtain local solutions to an optimal control problem.

A popular approach within trajectory optimization is direct transcription, which first discretizes the trajectory controls and states and then applies numerical optimization to find an optimal solution satisfying the problem constraints. Alternative methods discretize the controls but not the states, e.g., shooting method. They are more accurate but more complicated to formulate and solve, less numerically stable, and harder for including general state constraints. In this letter, we focus on direct transcription [17], [18].

## III. DIRECT TRANSCRIPTION

Direct transcription is a method for transcribing a continuous optimization 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 [19] which can then be solved using a sparse, large-scale nonlinear programming solver.

Here, we firstly discretize the system trajectory in $T$ equally spaced knot points at time intervals $\Delta t$. The robot state is described by the joint positions and velocities $x \triangleq (\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the the control inputs are the forces and torques $u \triangleq \boldsymbol{\tau}$ applied at the joints. Thus, given the nonlinear dynamics of the robot $\dot{x} = f(x, u)$, we discretize the state trajectory as $x_{1:T} = \{x_1, \cdots, x_T\}$ and the control trajectory as $u_{1:T-1} = \{u_1, \cdots, u_{T-1}\}$. The decision variables of the nonlinear program (NLP) can be extensively represented as $\boldsymbol{\Gamma} \triangleq \{\boldsymbol{q}_1, \dot{\boldsymbol{q}}_1, \boldsymbol{\tau}_1, \cdots, \boldsymbol{q}_{T-1}, \dot{\boldsymbol{q}}_{T-1}, \boldsymbol{\tau}_{T-1}, \boldsymbol{q}_T, \dot{\boldsymbol{q}}_T\}$.

Let us now consider the problem of drawing a curve in the workspace with the tool center point (TCP) of the robot while minimizing the actuator torques. This curve can be represented using the same discretization scheme used for the problem transcription: let a curve $c$ be split into $T$ segments and whose waypoints are given in Cartesian space, i.e., $c_{1:T} = \{c_1, \cdots, c_T\}$ and $c_t \in \mathbb{R}^3, \forall t = 1 : T$. This problem can be solved by the following NLP:

$$\begin{aligned} \operatorname*{argmax}_{\boldsymbol{\Gamma}} \quad & -\sum_{t=1}^{T} \|\boldsymbol{\tau}\|_t^2 & (8) \\ \text{subject to} \quad & \|c_t - f^{\text{FK}}(\boldsymbol{q}_t)\| \le \varepsilon & (9) \\ & \boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \Delta t \dot{\boldsymbol{q}}_t & (10) \\ & \dot{\boldsymbol{q}}_{t+1} = \dot{\boldsymbol{q}}_t + \Delta t \ddot{\boldsymbol{q}}_t & (11) \\ & \boldsymbol{q}^L \le \boldsymbol{q}_t \le \boldsymbol{q}^U & (12) \\ & \dot{\boldsymbol{q}}^L \le \dot{\boldsymbol{q}}_t \le \dot{\boldsymbol{q}}^U & (13) \\ & \boldsymbol{\tau}^L \le \boldsymbol{\tau}_t \le \boldsymbol{\tau}^U & (14) \\ & \dot{\boldsymbol{q}}_1 = \dot{\boldsymbol{q}}_T = \dot{\boldsymbol{q}}^* & (15) \end{aligned}$$

where (8) is the objective function to maximize, e.g., the negative summation of squared torques, and (9)–(15) are the problem constraints. Representing the FK of a configuration as $f^{\text{FK}}(\cdot)$, the TCP position over time can be constrained to match the desired waypoint in the curve $c$ with (9), where $\varepsilon$ is some user-defined tolerance, e.g., $1\,\text{mm}$. The dynamics of the system are enforced in (10)–(11) using forward Euler integration. Finally, the joint positions, velocities, and torques are constrained to be within their respective lower and upper bounds with (12)–(14), and the trajectory extremes are further constrained to have some desired velocity $\dot{\boldsymbol{q}}^*$ with (15) which we set to $\boldsymbol{0}$ (zero). It is especially important to note that, for this problem in specific, we are *not* constraining the joint positions at the trajectory extremes, $\boldsymbol{q}_1$ and $\boldsymbol{q}_T$.

In the remaining of this section, we enumerate some ($g_A$–$g_F$) of the numerous different objective functions which could be used in place of (8) for achieving different optimal results under the same problem constraints.

## A. Minimization of Actuator Torques

The above NLP formulation employed the negative sum of actuator torques as the objective function to be maximized:

$$g_A(\mathbf{\Gamma}) = -\sum_{t=1}^{T} \|\boldsymbol{\tau}\|_t^2. \tag{16}$$

This objective function is often used in optimal control [3] but it fails to explicitly capture or model any constraints imposed in the system, e.g., the actuator torque limits. It is true that the actuation bounds can be added to the problem in the form of linear inequality constraints — just like in (14) — but the objective function itself has no notion about these constraints whatsoever. As far as an off-the-shelf optimizer is concerned, this objective function is unable to distinguish between two identical control trajectories $u^A$ and $u^B$ if they result in the same value according to (16), even if the values of $u^A$ are relatively distant from actuator torque saturation but some values in $u^B$ are very close to them, i.e., $\exists u_t \in u^B : u_t \lesssim \tau^U$. For such a scenario, we can argue that $u^A$ is more robust to external disturbances in comparison to $u^B$ even though the problem formulation fails to capture it: because control trajectory $u^A$ is not comprised of torques near actuation limits, this remaining control margin can be exploited by other methods (e.g., [11]) for increased robustness during the control stage.

## B. Maximization of Residual Actuator Torques

Rather than minimizing actuator torques (16), we could instead maximize the summation of the distances between required torques and their closest boundary $\boldsymbol{\tau}^{\mathrm{lim}}$ with

$$g_B(\mathbf{\Gamma}) = \sum_{t=1}^{T} \|\boldsymbol{\tau}^{\mathrm{lim}} - \boldsymbol{\tau}\|_t^2. \tag{17}$$

## C. Maximization of Manipulability

Alternatively, we can aim to maximize the manipulability $w'$ [13] (*cf.* Equation 2 in Section II) over the trajectory:

$$g_C(\mathbf{\Gamma}) = \sum_{t=1}^{T} w'_t. \tag{18}$$

## D. Maximization of the Chebyshev Radius of $\mathcal{F}_{x_t}$

The fourth objective function we consider is the maximization of the Chebyshev radius [16] of the force polytope [13] (*cf.* II-B and II-D). Let us represent the force polytope of each discretized state $x_t \in \mathcal{X}$ as $\mathcal{F}_{x_t}$ and the radius of its Chebyshev center with $r_{\mathrm{cheb}}(\mathcal{F}_{x_t})$. Objective function $g_D$ is then given as the summation of the radii of the largest Euclidean balls that fit inside the force polytopes for each discretized state:

$$g_D(\mathbf{\Gamma}) = \sum_{t=1}^{T} r_{\mathrm{cheb}}(\mathcal{F}_{x_t}). \tag{19}$$

Setting this function as the NLP objective to be maximized allows the solver to take advantage of the kinematic redundancy of the system, resulting in more robust robot configurations. A pictorial example of how the redundancy

of the system can be used in order to confer *different shapes* to a force polytope is shown in Figure 2 for two identical inverse kinematics (IK) reaching tasks. The reaching target
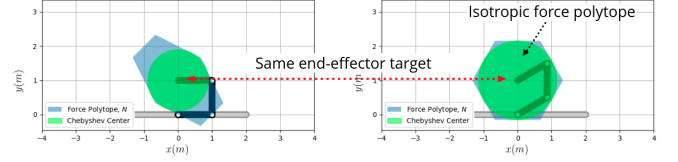


Fig. 2: Comparison between two robot states for an IK reaching task: $x^A$ (left) and $x^B$ (right).

is located at $(0, 1)$ for both cases. The scenario on the left shows the solution of a generic IK solver; for the scenario on the right, an additional term has been used for maximizing $r_{\mathrm{cheb}}(\mathcal{F}_{x^B})$. The augmented IK objective is able to drive the solution into a state for which the Chebyshev radius is larger. Fundamentally, the solver adapted the robot configuration to one whose force polytope is able to accommodate for a larger Euclidean ball inside its boundaries. As a result, and in general, $x^B$ is able to resist forces greater in magnitude when compared to $x^A$, assuming such external disturbances can originate from any given direction in the robot's workspace.

## E. Maximization of the Chebyshev Radius of $\mathcal{F}_{x_t}^*$

As previously discussed in Section II-B, the force polytope $\mathcal{F}_{x_t}$ results from the direct mapping of *absolute* lower and upper actuator torque bounds to Cartesian space; in contrast, for our fifth objective, we propose to map the actuator torque limits *reduced* by the discretized control $u_t \in \mathcal{U}$:

$$g_E(\mathbf{\Gamma}) = \sum_{t=1}^{T-1} r_{\mathrm{cheb}}(\mathcal{F}_{x_t}^*) \tag{20}$$

where $\mathcal{F}_{x_t}^*$ represents what we shall refer to as the *residual force polytope* henceforth. Figure 3a illustrates the differ-
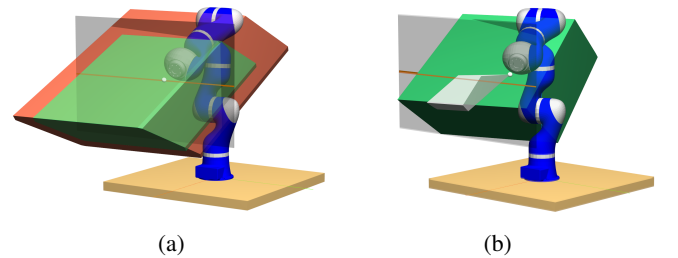


(a)         (b)

Fig. 3: *Left:* Shows the force polytope $\mathcal{F}_{x_t}$ (in red) and the *residual* force polytope $\mathcal{F}_{x_t}^*$ (in green). *Right:* Shows $\mathcal{F}_{x_t}^*$ and its intersection with the force cone $C_{\hat{\boldsymbol{f}}}$ (in white).

ences between $\mathcal{F}_{x_t}$ and $\mathcal{F}_{x_t}^*$ for an arbitrary instant of a control trajectory: notice how the residual force polytope (plotted in green) has been reduced in some directions; this reduction is caused by accounting for the nominal torques, i.e., gravity compensation and feed-forward torque, of the planned trajectory. Similarly to $g_B$, this function is only valid for methods akin to direct transcription as the values of the control inputs must be known ahead of the $\mathcal{F}_{x_t}^*$ mapping. To

best of our knowledge, it is the first time this mapping is being proposed and used as a term for NLP optimization.

### F. Maximization of the Volume of $\mathcal{F}^*_{x_t} \cap C_{\hat{f}}$

Finally, let us consider a scenario for which there exists an estimate concerning the directionality of an external force. Let $C_{\hat{f}}$ be a linearized cone directed by $\hat{f}$ and with apex at the end effector. This cone represents the disturbance estimation where $\hat{f}$ is the expected force direction and its aperture is proportional to the uncertainty on direction. Thus, we define the final objective studied in this letter as:

$$g_F(\Gamma) = \sum_{t=1}^{T-1} \text{vol}(\mathcal{F}^*_{x_t} \cap C_{\hat{f}}), \qquad (21)$$

where $\text{vol}(\cdot)$ gives the volume of a polyhedron and $\cap$ the intersection of two polyhedra. Effectively, the convex set given by $\mathcal{F}^*_{x_t} \cap C_{\hat{f}}$ defines the subset of all forces that the end effector may realize without saturating actuation limits. Figure 3b shows the residual force polytope (plotted in green) and the result of its intersection with $C_{\hat{f}}$ (plotted in white) for an arbitrary instant of an optimal trajectory. This is a similar approach to the one presented in [5] where the authors intersect the force polytope with a linearized friction cone to obtain a *bounded* friction cone.

## IV. ROBUSTNESS ASSESSMENT PROTOCOL

In this section we describe the experimental setup and the proposed task protocol. The purpose of this protocol is to evaluate the performance of trajectories obtained from solving the dynamic trajectory optimization problem detailed in Section III. Namely, we are interested in benchmarking the robustness of those trajectories against external disturbances. The specific details and numeric values in this section have been designed for the *KUKA LWR* robot manipulator. The protocol can be easily adapted to other robot models.

### A. Task Description

The following task assumes the robot's base is fixed at the origin of the workspace. The goal is to solve the NLP formulated in Section III with some changes concerning the constraints imposed on the end effector (*cf.* Constraint 9). The robot is to move the TCP from one point to another: the starting point ($t = 1$) is located at $(0.5, 0.4, 0.5)$ and the finishing point ($t = T$) at $(0.5, -0.4, 0.5)$. For all other instances, i.e., $1 < t < T$, the TCP needs only lie on a planar surface defined by $x = 0.5$, $-0.4 \le y \le 0.4$, and $-0.25 \le z \le 0.75$; this surface is visible in Figures 3a and 3b. Apart from (9), all the other constraints of the NLP from before are still enforced, i.e., constraints (10)–(15).

The NLP decision variables are seeded with feasible joint positions. A valid kinematic motion plan is used to set the joint positions for the seed, and it can be computed as follows: $T$ equally spaced points are sampled from a line segment defined by the task start and end points; then, an IK solver is used to compute a valid robot configuration reaching for each of the sampled points. For fairness and valid comparison, all the optimizations to be simulated and benchmarked must use the same initial seed.

The task duration is $1\,\text{s}$ and the system trajectory is to be discretized at $100\,\text{Hz}$, i.e., $T = 101$ and $\Delta t = 10\,\text{ms}$.

### B. Simulation

*1) External Disturbance:* A force is to be applied at the end effector of the robot during execution of the planned optimal trajectory. Figure 4 shows a plot of force magnitude over time for an example of such an impulse. The impulse total duration is the same as the task duration. $F_{\text{peak}}$ defines the peak magnitude of the impulse at $t = 0.5\,\text{s}$. The magnitude of the force at each instant is given by $F_{\text{peak}} \cdot \exp(-(t - 0.5)^2/0.02)$. We only consider forces orthogonal to the $z$-axis, and their direction is defined by a single angle $\theta$ as $(\cos(\theta), \sin(\theta), 0)$.
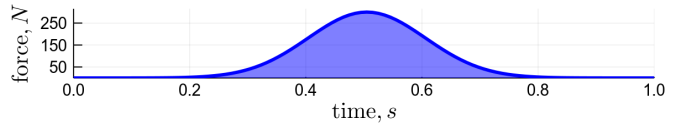


Fig. 4: Profile of an applied test force. At $t = 0.5\,\text{s}$ the force applied is at its peak. For this example the impulse (area under the curve) is $75.2\,\text{N s}$ and the peak force is $300\,\text{N}$.

*2) Payload:* The benchmark uses two robot models: one without a payload ($\mathcal{M}_1$), and another one with ($\mathcal{M}_2$). The payload for $\mathcal{M}_2$ is a $5\,\text{kg}$ solid cylinder with $22\,\text{cm}$ length and $3.25\,\text{cm}$ radius. The NLP must be solved for both robot models. Let the output trajectories of those two optimizations be $\mathcal{O}_1$ and $\mathcal{O}_2$. After the solving process, the benchmark should run the following three simulations:

$\mathcal{S}_1$. Execute trajectory $\mathcal{O}_1$ using robot model $\mathcal{M}_1$;
$\mathcal{S}_2$. Execute trajectory $\mathcal{O}_1$ using robot model $\mathcal{M}_2$;
$\mathcal{S}_3$. Execute trajectory $\mathcal{O}_2$ using robot model $\mathcal{M}_2$.

The above correspond to the three test scenarios: no payload ($\mathcal{S}_1$), unaccounted payload ($\mathcal{S}_2$), and accounted payload ($\mathcal{S}_3$).

*3) Robot Control:* The trajectories are simulated using a feed-forward torque controller with feedback on state error:

$$\boldsymbol{\tau} = \boldsymbol{\tau}^* + k_p \boldsymbol{I}(\boldsymbol{q}^* - \boldsymbol{q}) + k_d \boldsymbol{I}(\dot{\boldsymbol{q}}^* - \dot{\boldsymbol{q}}) \qquad (22)$$

where $\boldsymbol{\tau}$ are the torques commanded to the robot joints, $\boldsymbol{q}^*$, $\dot{\boldsymbol{q}}^*$, and $\boldsymbol{\tau}^*$ are the desired joint positions, velocities, and torques, respectively, and $k_p$ and $k_d$ are some user-defined gains for state feedback, i.e., position and velocity errors.[1]

### C. Performance Quantification

Robustness to external disturbances is evaluated in simulation. Following are the evaluation guidelines.

---

[1] These gains were tuned for stable trajectory tracking for each scenario: $k_p = 40$, $k_d = 0.4$ for $\mathcal{S}_1$ and $\mathcal{S}_3$; $k_p = 160$, $k_d = 1.07$ for $\mathcal{S}_2$.

TABLE I: Mechanical works under nominal and disturbed conditions, and normalized peak torques of $g_A-g_F$ from simulation. The disturbed work and peak torques have been collected for an impulse of $62.67\,\mathrm{N\,s}$ ($F_{\mathrm{peak}} = 250\,\mathrm{N}$ at $t = 0.5\,\mathrm{s}$).

| Goal | No payload | | | Unaccounted 5 kg payload | | | Accounted 5 kg payload | | |
|---|---|---|---|---|---|---|---|---|---|
| | nominal (J) | disturbed (J) | $\tau_{\max}$ (%) | nominal (J) | disturbed (J) | $\tau_{\max}$ (%) | nominal (J) | disturbed (J) | $\tau_{\max}$ (%) |
| $g_A$ | $5.6 \pm 0.0$ | $40.2 \pm 18.1$ | $72.9 \pm 23.8$ | $14.6 \pm 0.0$ | $45.8 \pm 17.0$ | $83.8 \pm 23.9$ | $8.1 \pm 0.0$ | $35.4 \pm 14.9$ | $79.4 \pm 19.9$ |
| $g_B$ | $4.4 \pm 0.0$ | $37.4 \pm 17.3$ | $72.7 \pm 22.8$ | $10.7 \pm 0.0$ | $40.9 \pm 16.3$ | $84.5 \pm 23.3$ | $47.3 \pm 0.0$ | $80.0 \pm 18.9$ | $79.9 \pm 6.9$ |
| $g_C$ | $6.9 \pm 0.0$ | $43.2 \pm 18.9$ | $73.5 \pm 23.7$ | $24.0 \pm 0.0$ | $52.3 \pm 16.9$ | $83.4 \pm 21.5$ | $8.1 \pm 0.0$ | $35.4 \pm 14.9$ | $79.4 \pm 19.9$ |
| $g_D$ | $4.9 \pm 0.0$ | $39.3 \pm 17.8$ | $75.8 \pm 17.4$ | $7.5 \pm 0.0$ | $38.9 \pm 17.0$ | $79.2 \pm 19.6$ | $5.9 \pm 0.0$ | $33.8 \pm 14.9$ | $78.8 \pm 20.2$ |
| $g_E$ | $4.9 \pm 0.0$ | $39.3 \pm 17.8$ | $75.8 \pm 17.4$ | $7.5 \pm 0.0$ | $38.9 \pm 17.0$ | $79.2 \pm 19.6$ | $5.9 \pm 0.0$ | $33.7 \pm 14.8$ | $78.8 \pm 20.2$ |
| $g_F$ | $6.1 \pm 1.2$ | $42.6 \pm 19.2$ | $73.4 \pm 23.2$ | $18.7 \pm 7.2$ | $48.4 \pm 19.1$ | $81.6 \pm 21.8$ | $5.8 \pm 0.6$ | $34.2 \pm 15.1$ | $79.2 \pm 20.6$ |

*1) Torque Expenditure and Mechanical Work:* For every trajectory benchmarked against a test impulse defined by $F_{\mathrm{peak}}$ (*cf.* Section IV-B.1), a standalone simulation is run for scenarios $\mathcal{S}_1-\mathcal{S}_3$ and for angles between $[0, 2\pi)$ for at least a recommended 64 equally spaced samples. The purpose of these simulations is to benchmark the trajectory performance when subjected to the test impulse.

For each combination of scenario, $F_{\mathrm{peak}}$, and angle $\theta$, the performance will be quantified in terms of *normalized peak torque*: the maximum normalized value recorded for a commanded joint torque; and *mechanical work*: the mechanical energy spent to execute the trajectory, given by $W = \sum_{t=1}^{T-1} \boldsymbol{\tau}_t \cdot (\boldsymbol{q}_{t+1} - \boldsymbol{q}_t)$, under nominal conditions and under influence of the external disturbance.

For collecting these results a few extra assumptions are made: we assume no delay and no error in sensing the external force and we also ignore the actuation limits of the robot. Essentially, we add to the commanded torques the direct mapping of force to torques with (3). Although unrealistic, these are acceptable for the theoretical comparison and benchmark we are interested in.

*2) Task Constraint Violation:* Another aspect evaluated is the end-effector task constraint defined in Section IV-A. The trajectories are simulated again with the assumptions explained above, with the exception of the unlimited actuation: this time the resulting commanded torques are clamped to the torque capabilities of the robot. The task constraint violation should then be computed as the summation of each $xyz$-component distance to the closest surface boundary for all the states of the trajectory.

*3) Runtime:* Finally, all the stages involved in computing the optimal trajectories are benchmarked concerning their execution time. This should include: the total time the solver took to converge, objective function evaluation per solver iteration, function and Jacobian evaluation per iteration for the end-effector task constraint and rigid body dynamics constraints. In case the objective functions employ some significant computational effort, those operations are timed as well; in this work, that includes the computation of the force polytope, residual force polytope, Chebyshev center, intersection of two polyhedra, and polyhedron volume.

## V. ROBUSTNESS ASSESSMENT BENCHMARK

We employed our optimization framework to benchmark the different objective functions $g_A-g_F$ enumerated in Sec-

tion III according to the protocol described in the previous section. In this section we report the benchmarking results.

A Jupyter notebook for reproducing the results presented in this section and all the required materials to run future benchmarks using the protocol are open-source and available at https://ferrolho.github.io/research/.

All evaluations in this section were carried out in a single-threaded process on an Intel i7-6700K CPU with $4.0\,\mathrm{GHz}$ and 32 GB 2133 MHz memory. For this work, we have used Julia [20] as the main programming language and Artelys Knitro [21] for solving the nonlinear optimization problems. For rigid body dynamics algorithms and simulation we used RigidBodyDynamics.jl [22] and RigidBodySim.jl, along with ForwardDiff.jl for forward-mode automatic differentiation (AD). Finally, we used Polyhedra.jl and QHull [23] for computational geometry operations.

TABLE II: End-effector task constraint violation.

| Goal | Scenario $\mathcal{S}_1$ | | Scenario $\mathcal{S}_2$ | | Scenario $\mathcal{S}_3$ | |
|---|---|---|---|---|---|---|
| | 250 N | 300 N | 250 N | 300 N | 250 N | 300 N |
| $g_A$ | 11.57 | 242.38 | 294.61 | 377.77 | 0.02 | 35.57 |
| $g_B$ | 9.03 | 212.96 | 344.76 | 405.05 | 43.28 | 44.51 |
| $g_C$ | 9.50 | 125.95 | 824.86 | 796.56 | 0.02 | 35.53 |
| $g_D$ | 0.72 | 99.70 | 405.73 | 410.71 | 2.60 | 38.55 |
| $g_E$ | 0.72 | 99.70 | 405.73 | 410.71 | 0.12 | 36.56 |
| $g_F$ | 3.03 | 65.46 | 197.14 | 205.08 | 0.25 | 37.90 |

### A. Results

*1) Torque Expenditure and Mechanical Work:* The resultant trajectories from each objective function $g_A-g_E$ were simulated for 64 sampled $\theta$ impulse directions, and for each of the scenarios $\mathcal{S}_1-\mathcal{S}_3$. In contrast to these objective functions, 64 optimizations have been run for $g_F$ since this objective takes directionality into account; all those 64 resulting trajectories have been simulated with an impulse according to the expected direction they have been optimized for. Table I shows the normalized peak torques of each objective for each tested scenario; additionally, it also shows the nominal mechanical work (obtained from the optimal planned torques) and the actual mechanical work realized during simulation, which includes the nominal torques, the required torques for counteracting the applied impulse, and the feedback torques from state error (*cf.* Section IV-B.3). Figure 5 depicts the results concerning peak torques. To allow for a richer analysis, the results of six different impulses with $F_{\mathrm{peak}} = \{0, 50, 100, 150, 200, 250\}$N are plotted.
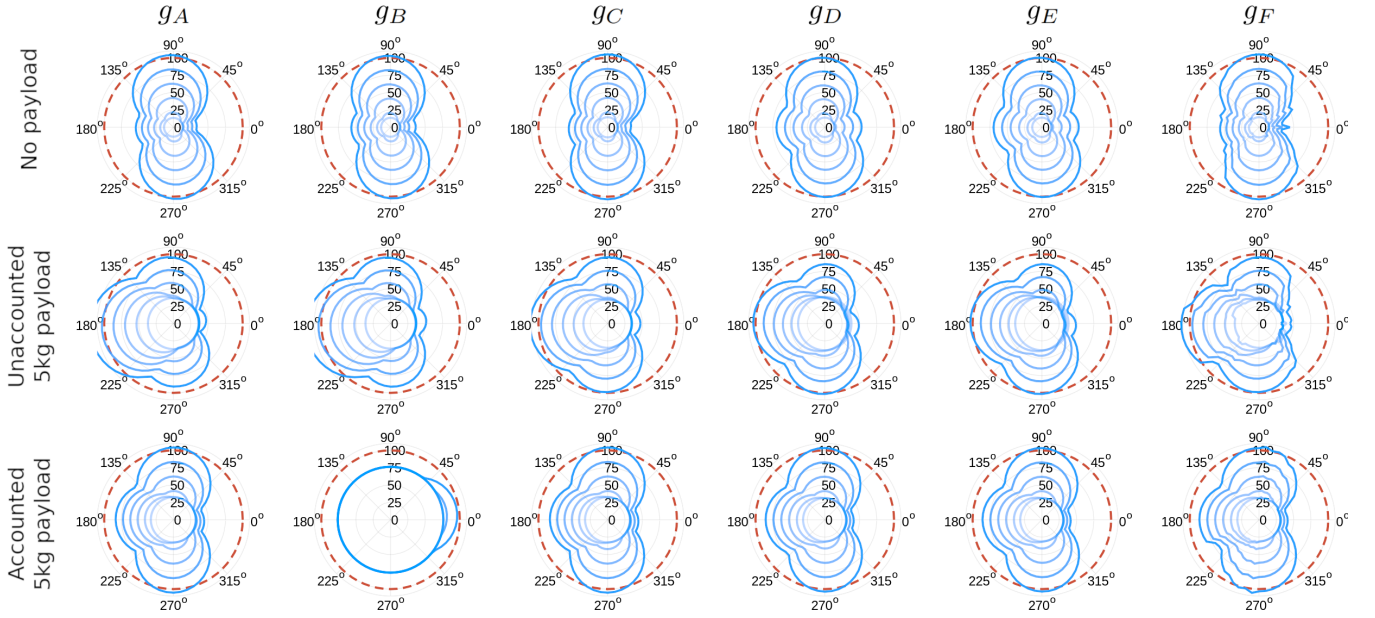
Fig. 5: Simulation results showing the normalized peak torques. The *angular coordinate* represents the acting direction $\theta$ of the force disturbance applied at the end effector. The *radial coordinate* represents the normalized peak torques in percentages. The dashed red circle represents the $100\%$ actuation limit. The solid blue lines show the normalized peak torques for six different impulses: $\{0, 12.53, 25.07, 37.6, 50.13, 62.67\}$ N s.

*2) Task Constraint Violation:* Figure 6 plots the total end-effector constraint error for different force directions $\theta$, for each function $g_A - g_F$. Table II shows the total error obtained from summing all errors for each of the $\theta$ shown in Figure 6.
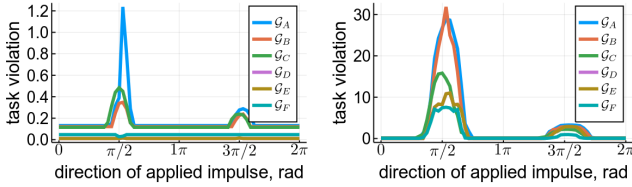


Fig. 6: Constraint violation analysis for the scenario without payload ($\mathcal{S}_1$). *Left:* $62.67$ N s impulse with peak force of $250$ N. *Right:* $75.2$ N s impulse with peak force of $300$ N.

TABLE III: Optimization convergence times without and with a payload, and objective function evaluation time. Values shown in the rightmost column are *per evaluation*.

| Goal | Total time until convergence | | Objective function |
| | No payload (s) | 5 kg payload (s) | evaluation (ms) |
|---|---|---|---|
| $g_A$ | $1.71 \pm 0.03$ | $16.85 \pm 0.24$ | $0.241 \pm 0.515$ |
| $g_B$ | $1.80 \pm 0.03$ | $7.10 \pm 0.15$ | $0.311 \pm 0.597$ |
| $g_C$ | $1.67 \pm 0.07$ | $17.35 \pm 0.35$ | $0.336 \pm 0.659$ |
| $g_D$ | $55.43 \pm 0.56$ | $130.03 \pm 1.76$ | $1183.712 \pm 49.427$ |
| $g_E$ | $54.99 \pm 0.55$ | $129.56 \pm 1.18$ | $1172.952 \pm 23.647$ |
| $g_F$ | $66.44 \pm 8.83$ | $183.48 \pm 33.05$ | $1560.032 \pm 18.755$ |

*3) Runtime:* Table III presents the times elapsed until solver convergence for $\mathcal{M}_1$ and $\mathcal{M}_2$, for each objective function $g_A - g_F$. It also lists the time required to evaluate each of the objective functions (per evaluation). Table IV lists the times taken to evaluate the function and the Jacobian of the end-effector task constraint and the dynamics constraints.

Finally, Table V lists the evaluation times for computationally expensive operations used by some of the objective functions.

TABLE IV: Times taken to evaluate the NLP constraints.

| End-effector constraints | | Dynamics constraints | |
| Function (ms) | Jacobian (ms) | Function (ms) | Jacobian (ms) |
|---|---|---|---|
| $0.116 \pm 0.37$ | $0.628 \pm 0.56$ | $0.517 \pm 0.54$ | $15.348 \pm 3.07$ |

### B. Discussion

The data and results collected from the benchmark show that using polytope-based optimization objectives renders the output optimal trajectories more robust to model errors and external perturbations in terms of realized mechanical work and actuation limit saturation. Concerning end-effector task constraints, these methods also show better performance under external disturbances; the same is not true for model errors (studied in this letter as unaccounted payload dynamics). However, if prior knowledge concerning perturbation directionality exists, polytope-based methods exploiting this insight outperform the other methods. In the remainder of this discussion, we dive into a more detailed analysis from which these claims have been inferred.

*For scenario $\mathcal{S}_1$:* concerning both realized work and observed peak torques, the results from Table I show little differences between the performance of objectives $g_A - g_F$. However, in terms of end-effector task constraint violation (see Table II), and for a $F_{\text{peak}} = 250$ N, $g_A$ performed the worst, followed by $g_C$ and $g_B$; the functions maximizing the Chebyshev radius, i.e., $g_D$ and $g_E$, performed equally well and violated the constraints only marginally.

*For Scenario $\mathcal{S}_2$:* $g_D$ and $g_E$ performed significantly well in terms of torque preservation (*cf.* Figure 5 middle row). For
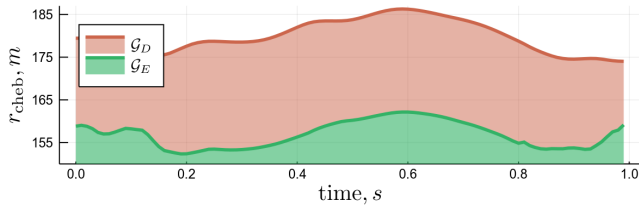
Fig. 7: Comparison between $r_{\text{cheb}}$ of $g_D$ and $g_E$ over time for scenario $\mathcal{S}_3$ under nominal circumstances.

a tested impulse of $62.67\,\text{N}\,\text{s}$ and $135° \leq \theta \leq 225°$ objectives $g_A$–$g_C$ overused torque beyond actuation capabilities (as shown by the blue lines crossing the dashed red limit); in contrast, $g_D$ and $g_E$ remained very close to the saturation limit. Nonetheless, these observations did not translate to task violation, where $g_D$ and $g_E$ did not perform the best; here $g_F$ was the clear winner and $g_C$ showed terrible results.

*For scenario $\mathcal{S}_3$:* the greater torques required for carrying the $5\,\text{kg}$ payload significantly reduce the residual force polytope. Because $g_E$ explicitly takes this into account during optimization it outperformed $g_D$ in this scenario. When $F_{\text{peak}}$ was increased to $300\,\text{N}$ this difference could still be observed. Figure 7 compares the radius over time for $g_D$ and $g_E$ and shows that they are not simply offset but rather have different profiles. The solver is able to find a different trajectory that maximizes these radii, thus increasing robustness.

For $\mathcal{S}_1$ and $\mathcal{S}_2$, all the objective functions converged to visually-similar trajectories. However, $g_B$ directed the optimizer in a significantly different direction for $\mathcal{S}_3$, resulting in the motion shown in Figure 1 (right). This difference can also be seen in the peak torque plots (see second plot on last row of Figure 5). The trajectory used similar peak torques for all tested impulses. Regardless, the intermediate states were challenging for the controller to track; this can be verified by the realized work both in nominal and disturbed conditions. Ultimately, this affected the end-effector task, which showed the poorest performance even for $F_{\text{peak}} = 250\,\text{N}$.

Concerning runtime, $g_A$–$g_C$ converge the fastest (one order of magnitude faster than $g_D$–$g_F$). Polytope-based methods involve extremely demanding computations (*cf.* Table V).

TABLE V: Time benchmark of some computational geometry operations performed by Polyhedra.jl.

| Operation | Time (µs) |
|---|---|
| Force polytope | $74 \pm 389$ |
| Force polytope (reduced) | $66 \pm 389$ |
| Chebyshev center | $10\,390 \pm 2470$ |
| Polyhedra intersection | $9623 \pm 1950$ |
| Polyhedron volume | $8968 \pm 1944$ |

## VI. CONCLUSIONS

In this letter, we set out to lay a comprehensive protocol and benchmark for assessing robustness against external perturbations in dynamic trajectory optimization. A framework for solving the optimization problem was implemented and made open-source, along with the benchmarking tools following the protocol guidelines. Extending this framework for legged robots is an interesting avenue for future work.

Different objective functions (including a proposed novel metric) were benchmarked for demonstration and validation of the protocol. Polytope-based methods showed little benefits compared with simpler metrics while being computationally expensive. However, we posit their advantages may arise for robots which need to maintain balance and use force interactions for locomotion. Regardless of technological advancements, opportunities of their deployment for time-critical applications currently seem challenging. Nonetheless, they carry significant value for feasibility analysis and system design, making them a compelling topic for further research.

## REFERENCES

[1] A. Del Prete and N. Mansard, "Addressing Constraint Robustness to Torque Errors in Task-Space Inverse Dynamics," in *RSS*, 2015.
[2] M. Plooij, W. Wolfslag, and M. Wisse, "Robust feedforward control of robotic arms with friction model uncertainty," *Robotics and Autonomous systems*, vol. 70, pp. 83–91, 2015.
[3] G. Xin, H.-C. Lin *et al.*, "A Model-Based Hierarchical Controller for Legged Systems Subject to External Disturbances," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
[4] T. Yoshikawa, "Manipulability of Robotic Mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.
[5] R. Orsolino, M. Focchi *et al.*, "Application of Wrench-Based Feasibility Analysis to the Online Trajectory Optimization of Legged Robots," *IEEE Robotics and Automation Letters*, 2018.
[6] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing Robot Workspace Structure: Representing Robot Capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.
[7] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot Placement based on Reachability Inversion," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1970–1975.
[8] Y. Yang, W. Merkt *et al.*, "Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps," *IEEE Robotics and Automation Letters*, 2017.
[9] H. Audren and A. Kheddar, "3D Robust Stability Polyhedron in Multicontact," *IEEE Transactions on Robotics*, 2018.
[10] A. Del Prete, S. Tonneau, and N. Mansard, "Fast Algorithms to Test Robust Static Equilibrium for Legged Robots," in *IEEE International Conference on Robotics and Automation*, 2016.
[11] A. Del Prete and N. Mansard, "Robustness to Joint-Torque-Tracking Errors in Task-Space Inverse Dynamics," *IEEE T-RO*, 2016.
[12] Z. Manchester and S. Kuindersma, "DIRTREL: Robust Nonlinear Direct Transcription with Ellipsoidal Disturbances and LQR Feedback," in *Robotics, Sciences and Systems (RSS)*, 2017.
[13] P. Chiacchio, Y. Bouffard-Vercelli, and F. Pierrot, "Force polytope and force ellipsoid for redundant manipulators," *Journal of Robotic Systems*, vol. 14, no. 8, pp. 613–620, 1997.
[14] T. S. Motzkin, H. Raiffa *et al.*, "The double description method," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 51–73, 1953.
[15] K. Fukuda and A. Prodon, "Double description method revisited," in *Combinatorics and computer science*. Springer, 1996, pp. 91–111.
[16] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[17] O. von Stryk and R. Bulirsch, "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, 1992.
[18] O. von Stryk, *Numerical Solution of Optimal Control Problems by Direct Collocation*. Basel: Birkhäuser Basel, 1993, pp. 129–143.
[19] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Siam, 2010, vol. 19.
[20] J. Bezanson, A. Edelman *et al.*, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, 2017.
[21] R. H. Byrd, J. Nocedal, and R. A. Waltz, *Knitro: An Integrated Package for Nonlinear Optimization*. Boston: Springer US, 2006.
[22] T. Koolen and R. Deits, "Julia for robotics: simulation and real-time control in a high-level programming language," in *IEEE International Conference on Robotics and Automation*, 2019.
[23] C. B. Barber, D. P. Dobkin *et al.*, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, 1996.