



HAL
open science

Simulation aided co-design for robust robot optimization

Gabriele Fadini, Thomas Flayols, Andrea del Prete, Philippe Souères

► **To cite this version:**

Gabriele Fadini, Thomas Flayols, Andrea del Prete, Philippe Souères. Simulation aided co-design for robust robot optimization. 2022. hal-03592085v1

HAL Id: hal-03592085

<https://laas.hal.science/hal-03592085v1>

Preprint submitted on 1 Mar 2022 (v1), last revised 17 Nov 2022 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Simulation aided co-design for robust robot optimization

Gabriele Fadini¹, Thomas Flayols¹, Andrea Del Prete², Philippe Souères¹

Abstract—This paper outlines a bi-level optimization method to concurrently optimize robot hardware parameters and control trajectories that ensure robust performance. The outer loop consists in a genetic algorithm that optimizes the hardware according to its average performance when tracking a locally optimal trajectory in perturbed simulations. The tracking controller exploits the locally optimal feedback gains computed in the inner loop with a Differential Dynamic Programming algorithm, which also finds the optimal reference trajectories. Our simulations feature a complete actuation model, including friction compensation and bandwidth limits. Our method can potentially account for arbitrary perturbations, and it discards hardware designs that cannot robustly track the reference trajectories. Our results show improved performance of the designed platform in realistic application scenarios, autonomously leading to the selection of lightweight and more transparent hardware.

I. INTRODUCTION

The problem of designing complex robotic hardware using numerical optimization has received considerable attention in recent years [1]–[3]. However, robustness remains an open challenge for co-design, while being crucial to ensure the practical applicability of the designed solutions. This becomes even more challenging for inherently unstable systems, such as legged robots. Following a common assumption in co-design, our previous work [4] was mainly targeting the adequacy of the hardware for optimizing the motion. However, optimal trajectories may be unfeasible on a real system, e.g., because the system cannot reject external perturbations due to unmodeled dynamics, noise, delays, saturation or actuator dynamics. So, even if optimality remains a fundamental criterion, robustness seems even more so to bridge the gap between the optimization results and the real system. To do so, the synthesis and evaluation of a tracking controller to compensate for noise must be addressed.

The problem of robustness in co-design has been investigated in the literature. A common approach to deal with the stability of perturbed systems is to optimize the self-stability of the trajectory. This can be done in open-loop [1] or with closed-loop sensitivity analysis [2], [5]. Both options rely on custom-made and differentiable cost formulations that increase the complexity and nonlinearity of the problem itself. Another possibility is stochastic programming, in which the optimal trajectory is found for a set of perturbed scenarios [3], [6], [7]. However, such methods do not scale favourably as the dimension of the problem increases significantly for each additional scenario. Another

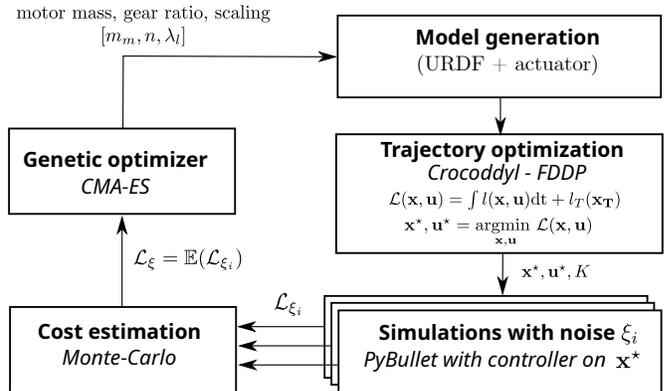


Fig. 1: Robust bi-level scheme with additional simulations

possibility is to include the controller in the optimization problem, with additional decision variables. Especially in the case of co-design, this becomes prohibitive in terms of computational complexity, because of the large state dimension [8], [9]. Controller optimization is performed with some assumptions such as the use of reduced models [10], sequential optimization [11] or constant gains along the trajectory. In the realm of co-design, some heuristic approaches to generate controllers, such as PD gains tuning, have also been used [12]. In addition, swarm exploration [13] and multi-objective co-design with gain tuning have been proposed in the past, but without specifically addressing robustness [14]. Our approach has some similarities to the technique of *domain randomization* used in reinforcement learning (RL) [15], which aims to learn a control policy that performs well with a variety of different (possibly perturbed) robot models. Instead, we try to find hardware parameters such that a locally optimal linear controller can perform well under different perturbations.

Overview of the paper: A co-design algorithm to ensure robustness is proposed by implementing a bi-level optimization, with a parallelized version of CMA-ES [16]. To ensure robustness we use as metric the average performance of the controller over multiple simulations [17], each one including noises and an actuator model. In simulation, the system is stabilized around an optimal trajectory with the Riccati gains [18], automatically computed by the Differential Dynamic Programming (DDP) algorithm, to avoid the extra computational complexity of explicitly optimizing for a control policy. Our algorithm scales much better than other robust co-design approaches that explicitly optimize for a robust controller, and it does not require an explicit model of the perturbations, which simply need to be introduced in

¹ LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France

² Industrial Engineering Department, University of Trento

This work has been supported by the MEMMO European Union project within the H2020 Program under Grant Agreement 780684.

Algorithm 1: Bi-level optimization

Input : $N_{gen}, N_{pop}, N_{sim}, tol, setup$
1 $N_p \leftarrow 0$
2 $pop \leftarrow$ random N_{pop} comb. of HW params
Outer loop:
3 **while** $N_p < N_{gen}$ **or** *stop condition* $\leq tol$ **do**
 Inner loop:
4 $\mathcal{L}_\xi \leftarrow []$
5 **for** $params \in pop$ **do**
6 $\mathbf{x}^*, \mathbf{u}^*, K \leftarrow$ solveDDP($params$)
7 $\mathcal{L}_\xi.append(simu(params, \mathbf{x}^*, \mathbf{u}^*, K, N_{sim}))$
8 $pop \leftarrow$ evolveCMEAS(pop, \mathcal{L}_ξ)
9 $N_p \leftarrow N_p + 1$

simulation. This comes at the price of the blindness of the inner loop (DDP) to perturbations, which theoretically limits the quality of the design outcome. However, our tests show extremely promising results for the design of an energy-efficient robot manipulator and a jumping monopod, which proved to perform much better under perturbations than their counterparts designed with our previous framework [4].

II. METHODOLOGY

Robust bi-level co-design optimization scheme

Our approach is to introduce robustness information in the bi-level optimization structure introduced in [4], by adding a simulation step with perturbations and a controller as shown in Fig. 1 and in Algorithm 1 line 7. This algorithm is briefly outlined. In an outer loop the hardware parameters are optimized with CMA-ES. Initially a population pop of N_{pop} possible robot hardware configurations is randomly initialized. Then, for each individual set of parameters $params \in pop$, a model of the robot is generated and the corresponding OCP solved. This inner loop is just optimizing the reference trajectories. Each optimized trajectory $\mathbf{x}^*, \mathbf{u}^*$ is then tracked in N_{sim} simulations with a controller using the Riccati gains \mathcal{K} computed by DDP. Each simulation includes a set of perturbation sources ξ acting at the joint torques. Moreover, the robot model corresponding to $params$ is tested in simulations that include a model of friction and actuator dynamics. The problem cost function \mathcal{L} is then averaged on the ensemble of N_{sim} simulation trajectories $\mathbf{x}_\xi, \mathbf{u}_\xi$ with a Monte-Carlo approach to obtain the robust metric under perturbations \mathcal{L}_ξ as in (7). Then, the outer-loop gets the values of the robust cost evaluated on the reference trajectories. With the evaluated cost information a new population can be selected by the genetic optimizer as the next one to explore. The outer loop keeps iterating until a termination condition *stop condition* $< tol$ or the maximum number of generations N_{gen} is attained. In the implementation of the algorithm, for each generation, the evaluation of the cost \mathcal{L}_ξ is parallelized asynchronously in order to speed up the algorithm. Unlike stochastic optimization, the number of these perturbed scenarios can be rather high (in the order of 10^3), at a reduced computational cost of

multiple simulation runs (linear in the number of time-steps and simulations $O(N_{sim})$). In this way the genetic algorithm selects the hardware associated to the most robust trajectories and feedback controllers.

Local state feedback controller

Many approaches can be applied to perform feedback control of unstable and underactuated robotic systems. A promising one is to locally re-plan online the ideal trajectory on a receding horizon using model predictive control (MPC) [19], [20]. Another method is to learn offline a global control policy using techniques such as reinforcement learning. These methods often require problem specific tuning and thus cannot be easily automatized for co-design. Furthermore, as the goal is just to treat relatively small deviations from the reference trajectory, it may be unnecessarily too complex. Close to optimality, the dynamics can be linearized. The MPC controller is then equivalent to a local linear controller. This guarantees to follow the planned trajectory while counteracting small disturbances [18]. This consideration is used in our approach to make the problem tractable. Moreover, using Riccati gains proves to be a natural way to synthesize a local controller around the optimal trajectory. For the unconstrained LQR problem, such gains are known to be optimal, as detailed in [21], however an extension may be possible also with equality or inequality constraints. In order to follow the state trajectory \mathbf{x}^* from a state $\mathbf{x} \approx \mathbf{x}^*$, the control \mathbf{u} should include a feedback term linear in the state error:

$$\mathbf{u} = \mathbf{u}^* + \mathbf{K}(\mathbf{x} - \mathbf{x}^*) \quad (1)$$

The gain matrix \mathbf{K} maps state deviations to corrections of the control input. This controller requires no tuning once the OCP is solved with DDP, contrary to other techniques such as PD control with gain scheduling. A drawback is that this feedback is guaranteed to work only close to the optimal trajectory. If the perturbations on \mathbf{x} are too large, or if the dynamics is highly nonlinear, these considerations may not hold anymore and other strategies must be used instead.

a) Joint dynamics: In a realistic application, to implement a control law such as (1) on the robot, an actuator model has to be considered. In order to deal with joint dynamics, the same methodology as in [4] is used: the OCP solution provides the joint trajectories that minimize the electrical energy consumption, without modeling friction in the robot dynamics, but considering it only in the cost function. Then the reference control $\boldsymbol{\tau}$ is computed so to compensate for friction:

$$\boldsymbol{\tau} = \mathbf{u} + \boldsymbol{\tau}_{\mu,0} \text{sign}(\mathbf{v}_a) + \mathbf{b} \mathbf{v}_a \quad (2)$$

To overcome joint friction, the torque \mathbf{u} is changed with an additional feedforward compensation for joint static friction $\boldsymbol{\tau}_{\mu,0}$ [Nm] and joint damping \mathbf{b} [Nms]. The correction compensates friction from the real state $\mathbf{x} = [\mathbf{q}_u, \mathbf{q}_a, \mathbf{v}_u, \mathbf{v}_a]^\top$, where the subscripts (a) and (u) respectively denote the actuated and underactuated parts of positions \mathbf{q} and velocities \mathbf{v} .

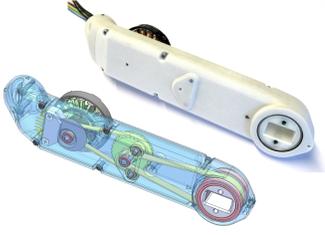


Fig. 2: Actuator module, showing the belt-drive transmission and the BLDC motor placement, courtesy of ODRI [22].

b) Actuator bandwidth and torque limits: In addition to friction compensation, the actuator dynamics is modeled as a first-order low-pass filter (3):

$$\mathbf{u}_k = \alpha \tau_k + (1 - \alpha) \tau_{k-1}, \text{ for } k \in 0, \dots, N-1 \quad (3)$$

where the initial torque is $\mathbf{u}(0) = \mathbf{u}^*(0)$ and α is a parameter depending on the cut-off frequency that was fixed to 20 Hz based on testing. This filtering is introduced to simulate controls that can be reasonably applied at the robot joints within the bandwidth limitations of the actuator. Finally, a saturation is introduced to enforce the torque limits: $\underline{\mathbf{u}} < \mathbf{u}_k < \bar{\mathbf{u}}$.

Parametric actuation model

The actuator technology is shown in Fig. 2: each joint is controlled by a BLDC motor with a low gear-ratio belt transmission. For this hardware, a parametric model is introduced to obtain the values of the friction parameters (damping \mathbf{b} and Coulomb friction $\tau_{\mu,0}$) and motor properties (rotor inertia, winding resistance and torque constant). Such parametrization is just dependent on the motor mass and the gear ratio of the transmission [4], following an approach already used in co-design [23], [24]. With the values obtained, the dynamics of the system will be modified by:

- 1) modifying the joint inertia of the robot to include also the additional motor inertia
- 2) adding the motor mass to the links.

Power components

The values of the electro-mechanical characteristics are used to compute the power components used as costs:

$$P_m = \mathbf{u}^{*\top} \mathbf{v}_a \text{ [W]} \quad (4)$$

$$P_f = (\tau_{\mu,0} \text{ sign}(\mathbf{v}_a) + \mathbf{b} \mathbf{v}_a)^\top \mathbf{v}_a \text{ [W]} \quad (5)$$

$$P_t = \mathbf{u}^\top \tilde{\mathbf{K}} \mathbf{u} \text{ [W]} \quad (6)$$

Mechanical power P_m is shown in (4) where \mathbf{u}^* is the ideal torque at the joint and \mathbf{v}_a is the actuated joint velocities.

Joint friction power loss P_f as in (5) corresponding to the joint friction, reconstructed from \mathbf{v}_a multiplied by the friction compensation torque (2).

Joule dissipation power loss P_t is obtained as in (6) from the value of the torque on the joint \mathbf{u} , including friction compensation as in (2). $\tilde{\mathbf{K}}$ is a diagonal matrix mapping joint torques to power. Each $\tilde{\mathbf{K}}_{i,i}$ entry depends on: the i^{th} motor torque constant, its winding resistance and on the

actuated joint reduction. The power components (4), (5) and (6) are summed to get the total electrical power, considering perfect regeneration and efficiency of the electronic inverter. In the OCP, from a friction-less dynamics the value of the friction compensation (2) is reconstructed and then the overall associated energy minimized.

Ensembled final cost

The main feature of the introduced method is to consider perturbation sources for each simulation $\xi \in \mathbb{R}^{N_{sim} \times n_u \times N}$, where n_u is the number of actuated joints and N the number of timesteps. The single joint torque noise realization ξ_i with $i \in \{0, \dots, N_{sim} - 1\}$ acts on the ideal joint torque of the robot \mathbf{u}^* , proportionally to its value: $\xi_i \sim \mathcal{N}(0, \sigma^2) \mathbf{u}^*$. Given ξ_i , the trajectories $\mathbf{x}_{\xi_i}, \mathbf{u}_{\xi_i}$ obtained in simulation with the controller are used to re-evaluate the optimal cost function of the problem $\mathcal{L}(\mathbf{x}_{\xi_i}, \mathbf{u}_{\xi_i})$, using a cost function that includes both task fulfillment and energy minimization terms. This will have similar information with respect to the minimum cost obtained by the DDP solver $\mathcal{L}(\mathbf{x}^*, \mathbf{u}^*)$, but will enrich it with that from simulations with the control correction. Since each \mathcal{L}_{ξ_i} is a random variable, depending on the realizations of the noise ξ_i , its expected value \mathcal{L}_{ξ} is obtained using a Monte-Carlo approach. CMA-ES finally minimizes this last metric \mathcal{L}_{ξ} .

$$\mathcal{L}_{\xi} = \mathbb{E}(\mathcal{L}_{\xi_i}) \approx \frac{1}{N_{sim}} \sum_{i=0}^{N_{sim}-1} \mathcal{L}(\mathbf{x}_{\xi_i}, \mathbf{u}_{\xi_i}) \quad (7)$$

III. RESULTS

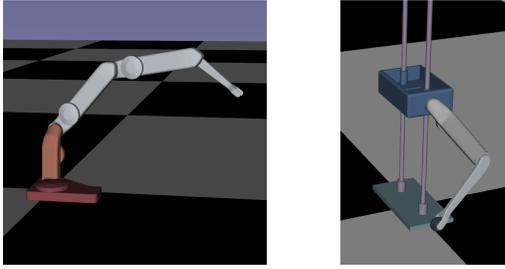
In this section, after having validated that the local controller based on the Riccati gains stabilizes the system despite a wide set of perturbations, we use it for co-design. To solve the OCP and obtain the Riccati gains, we use `crocodyl` [25]. A custom URDF is shared between the OCP and the simulator `PyBullet` [17]. It is generated parametrically with the ROS module `xacro` [26]. Two types of robot are optimized for robust task tracking, as shown in Fig. 3. Each one is made up with variants of the same actuator module (see Fig. 2) developed in the framework of ODRI [22]:

- *Serial manipulator* (Fig. 3a) With a fixed base (red), it includes 4 links actuated by 4 motors ($R_Z - 3 \times R_X$). Only the sizes of the last 3 links are optimized, while the shape of the base and the Z-axis link (orange) are fixed.
- *Monoped* (Fig. 3b) With a non-actuated base (blue) that can move freely along a vertical prismatic link (2 bars), it includes two optimized links and two actuated revolute joints [4], [22], [27].

Co-design parameters: The hardware optimization concerns the following parameters:

- Motor mass $m_m \in [0.05, 1]$ kg
- Gear ratio $n \in [3, 20]$
- Link scaling $\lambda_l \in [0.8, 1.2]$ ($\lambda = 1$ is the nominal case)

The joint friction parameters and the actuator electro-mechanical properties can be estimated from the first two parameters. The scaling represents the ratio between the link length and its nominal value. A single scaling parameters is used because, to keep the relative deflection of the link



(a) Manipulator (b) Monoped

Fig. 3: Robot models used in our tests.

TABLE I: Manipulator cost function weights

Weight	Type	Value
Mechanical power	Running	1e-2
Power losses	"	1e-2
Final frame position	Terminal	1e4
Final frame velocity	"	1e6
Penalty on the max torque	Penalty	1e4
Intermediate frame position	"	1e6
Intermediate frame velocity	"	1e6

constant once a given length is chosen, the section needs to scale with a predefined law as detailed in [4].

MANIPULATOR BACK AND FORTH TASK

Task: The task is to displace a fixed mass payload of 0.1 kg from an initial position p_0 (from a initial configuration q_0) up to a given point $p = p_0 + [-0.1, -0.1, 0.1]^T$ m, and then bring it back to p_0 . At the intermediate and final positions the joint velocities must be null.

Hyper-parameters: The parameters related to CMA-ES are the number of generations $N_{gen} = 10$, and the number of problems per generation $N_{pop} = 10000$. Parallelization is used to speed up computation. On a standard desktop computer, ≈ 102 hours were necessary for solving 10^5 problems, with a mean time per problem of 3.7 s, including the simulation phase. The OCP has 1000 nodes and $dt = 1$ ms; the cost weights are reported in Table I. For the realization ξ the value of $\sigma_\xi = 0.2$ was selected and $N_{sim} = 100$.

Discussion: The values obtained with the classic method and its robust counterpart are compared. Fig. 6b shows that the classical method applies very high torques when the joint velocities are null (initial, intermediate and final configurations) to minimize the mechanical power (4). However, this solution is hardly applicable to the real system due to bandwidth limitations. Tracking is better in the robust case as illustrated qualitatively in Fig. 6a and quantitatively by the lower $RMSE = \sqrt{\sum_{i=0}^{N_{sim}} \|\mathbf{x}_{\xi_i} - \mathbf{x}^*\|_2^2} / N_{sim}$ in Table II. The hardware parameters in Table II show that, to provide a high impulsive torque, the standard method selects bulkier motors, dissipating less energy by Joule effect and more energy by Coulomb friction Fig. 7b. On the other hand, the robust method induces higher Joule losses, and selects smaller motors, but overall requiring less torque and adding less mass to the system Fig. 7a. Reasonably, both methods

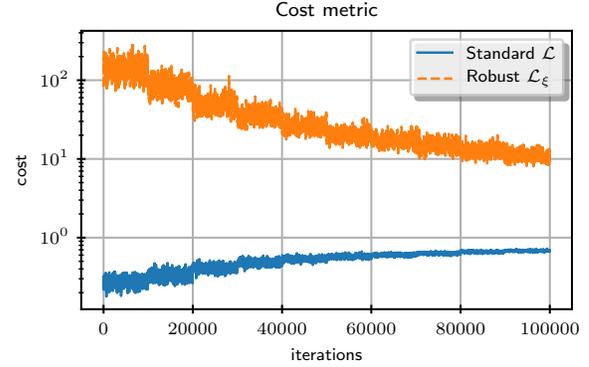


Fig. 4: Robust cost metric \mathcal{L}_ξ evolution trend during CMA-ES (orange curve). \mathcal{L}_ξ diminishes for each iteration of Alg. 1, while the standard cost \mathcal{L} increases (blue curve).

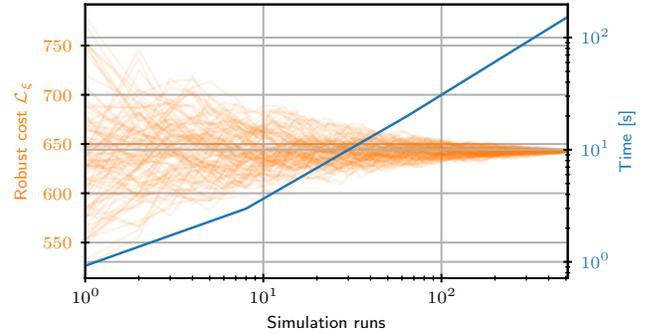
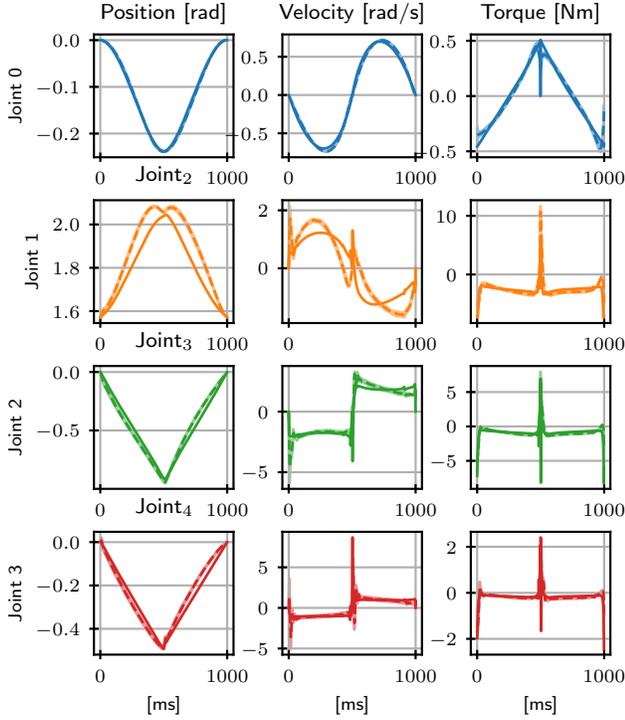
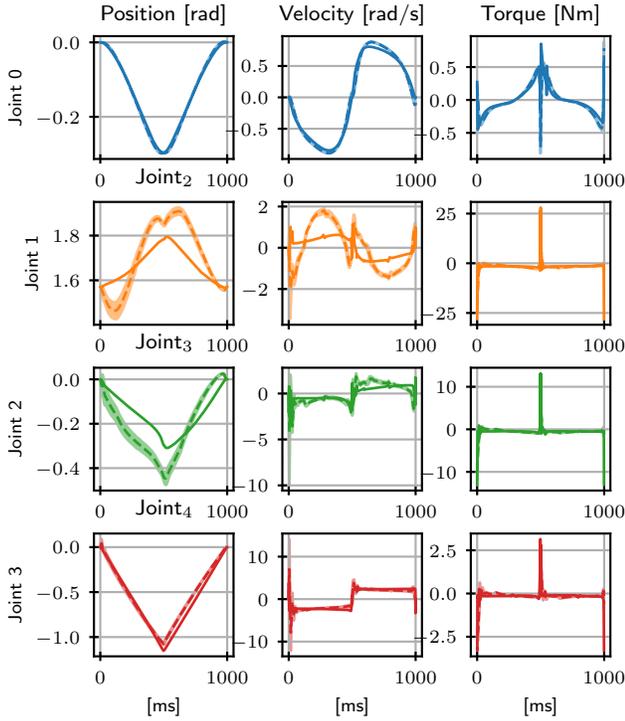


Fig. 5: Monte Carlo evaluation of \mathcal{L}_ξ for different number of simulations. Each curve represents the value of \mathcal{L}_ξ obtained by progressively increasing the number of perturbed simulations with a different random seed.

select the link size to reduce the moving masses and system inertia. Fig. 4 shows the cost metric profiles along the successive iterations of CMA-ES for the standard and the robust approach. In Fig. 4 we notice that the improvement of the robust metric is accompanied with a degradation of the standard metric. The robust version penalizes a lot the designs and controls that are not able to fulfill closely the task (given the high weight on the final position, this results in a cost orders of magnitude greater than the standard one). Despite this (expected) trade-off, the optimization of \mathcal{L}_ξ produces designs that are able to follow more closely the task under perturbations. In Fig. 5 the convergence of \mathcal{L}_ξ to the empirical expected value is shown, for different histories of ξ and different random seeds. From the trend it is noticeable that after around 100 simulations the deviation is two standard deviations from the empirical expected value, while for 1000 simulations we can consider \mathcal{L}_ξ completely converged. The plot also shows that the computational cost is roughly linear with the number of simulations.

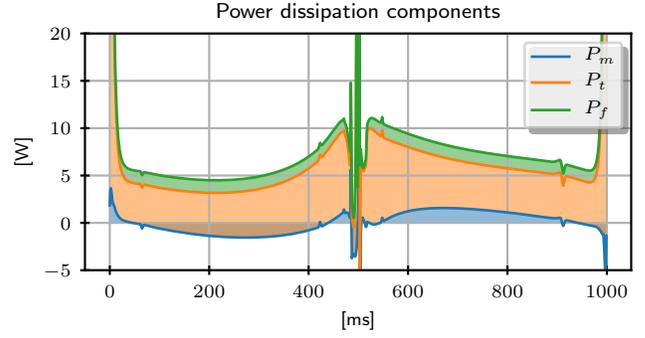


(a) Robust method.

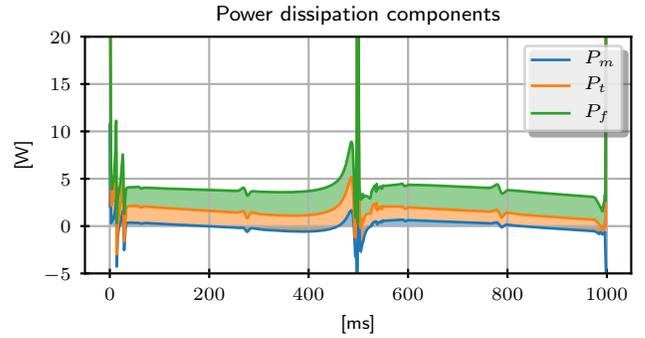


(b) Standard method.

Fig. 6: Manipulator back-and-forth tracking. Solid lines represent reference trajectories, dashed lines represent the mean of simulated trajectories; shaded regions show the deviation $\pm 3\sigma$ around the mean.



(a) Robust method



(b) Standard method

Fig. 7: Power components required by the back and forth task. Colored areas are proportional to the energetic expense. The upper bound of the curve is the sum of all the power components.

TABLE II: Results for the manipulator back and forth task.

Quantity	Robust	Standard
Cost \mathcal{L}	$9.58e-3$	$3.5e-3$
Cost \mathcal{L}_ξ	33.42	2e3
λ_l	[0.83, 1.02, 0.86]	[0.80, 0.80, 1.08]
m_m	[0.05, 0.05, 0.05, 0.06]	[0.2, 0.76, 0.49, 0.4]
n	[16.7, 11.6, 11.8, 15.2]	[17.1, 11.8, 11.4, 15.3]
RMSE	0.287	1.836
$\sum_i P_{m,i} dt$ [J]	-0.9	-1.7
$\sum_i P_{t,i} dt$ [J]	6.5	1.8
$\sum_i P_{f,i} dt$ [J]	1.3	3.2

TABLE III: Non-weighted cost residuals after perturbation for the manipulator case.

Quantity	Robust		Standard	
	$\mu(\Sigma r)$	$\sigma(\Sigma r)$	$\mu(\Sigma r)$	$\sigma(\Sigma r)$
Actuation penalty	0.00	0.00	0.00	0.00
Mechanical power	-0.44	1.54	64.54	10.2
Joule losses	8445.60	32.3	2408.0	143.6
Joint friction	1983.59	49.6	8343.0	223.5
Placing position	$4.7e-5$	$2.19e-6$	$1e-6$	$2.8e-10$
Placing zero velocity	0.531	0.173	6.44	4.74
Final position	0.193	0.136	21.41	13.2

TABLE IV: Costs for the monopod with associated weights

Quantity	Robust	Standard	Weights
Cost			
Mechanical power	-16.67	-111.24	10
Joule power	49.02	76.03	10
Joint friction	16.00	48.21	10
Penalty			
Base penalty	0	0	10^3
Foot penalty	0	0	10^4
Knee penalty	0	0	10^4
Actuation penalty	0.0035	0.012	10^4
Regularization			
Friction cone	1.52	1.054	10^{-1}
Jump threshold	2.53	0.00	10^5
Contact at zero	0.046	0.021	10^3
Terminal state	1.27	0.028	10^3

MONOPED JUMP

To test our method in a more complex scenario involving contact switches, we used it to design a jumping monopod.

Task: The robot has to perform a jump with the base in a given time, and stabilize the system after touch-down. In the OCP this task is enforced weakly as a penalty on the prismatic joint z position.

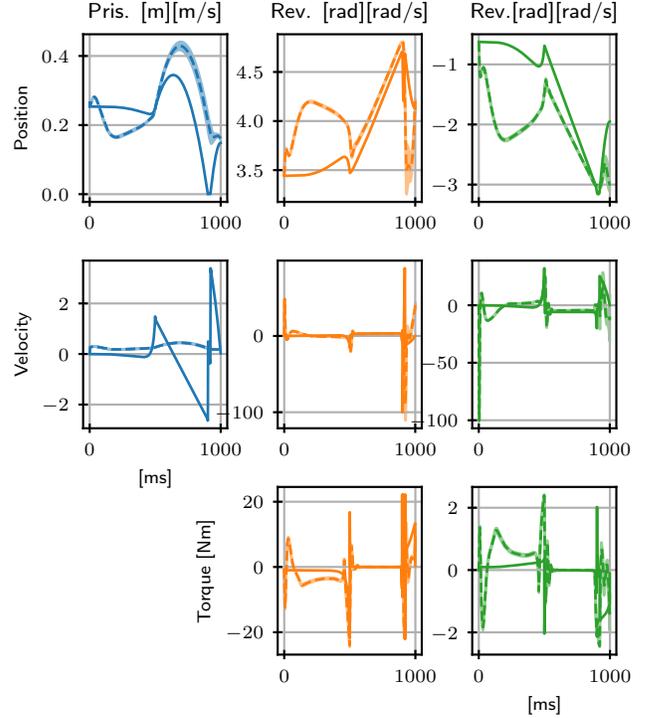
$$l_{jump}(z) = \begin{cases} 0 & \text{if } z \geq z_{ref} \\ ||z - z_{ref}||_2^2 & \text{if } z < z_{ref} \end{cases} \quad (8)$$

This task encourages motions that are jumping above the reference height threshold ($z_{ref} = 0.4$ m), but still allows smaller structures that under-perform the task. At the same time, letting the maximum height unspecified is beneficial to find solutions that satisfy the task timing sequence and the dynamics. In the OCP there are four predefined phases for the jumping motion:

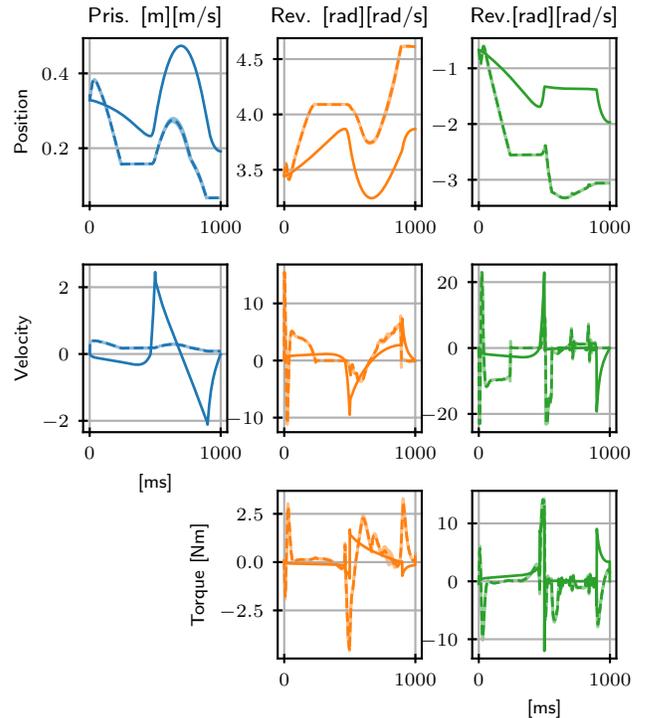
- *Contact phase:* the foot contact with the ground is enforced in the dynamics for a fixed number of nodes
- *Flying phase:* the contact with the ground is broken and the monopod is jumping. At the intermediate node the cost (8) is applied.
- *Impact phase:* the foot velocity at the new contact point is set to zero and a small regularization is added to penalize landing of the foot far from the origin.
- *(Post-impact) contact phase:* the leg, while in contact with the ground, can decelerate the base motion and stop the system.

Hyper-parameters: For CMA-ES the following parameters were chosen: $N_{gen} = 5$, $N_{pop} = 1000$. The OCP has 1000 nodes and $dt = 1$ ms; the cost weights are reported in Table I. For the realization ξ the value of $\sigma_\xi = 0.2$ was selected and $N_{sim} = 100$.

Cost comparison: Fig. 8 shows the reference joint velocities for robust and standard methods. Qualitatively the simulated trajectories are similar to the ideal trajectories. A jump of the base is performed even if perturbations and actuator bandwidth limitations make the monopod perform worse, anticipating the contact phase with the ground. This means that the optimized motion needs high accelerations



(a) Robust method.



(b) Standard method.

Fig. 8: Monoped jump: solid lines represent reference trajectories, dashed lines the mean of the simulated trajectories, while shaded regions show the deviation $\pm 3\sigma$ around the mean.

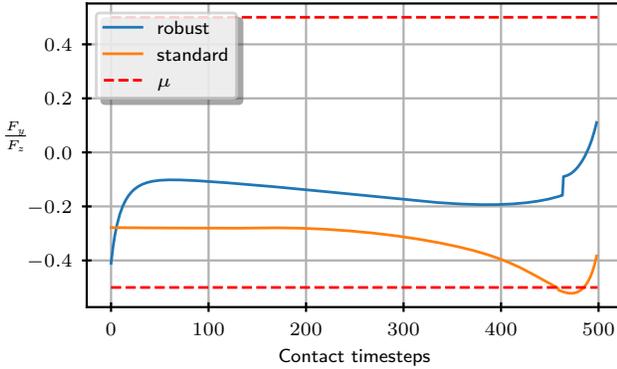


Fig. 9: Monoped jump: ratio between vertical and lateral contact forces, and friction cone bounds $\pm\mu$.

TABLE V: CMA-ES results for the monoped.

Quantity	Robust	Standard
Cost \mathcal{L}	50.91	13.67
Cost \mathcal{L}_ξ	631.08	1.09e4
Scaling	[1.2, 0.973]	[0.87, 0.8]
Motor mass	[0.125, 0.282]	[0.812, 0.05]
Gear ratio	[5.43, 7.83]	[5.19, 7.00]
RMSE on x^*	1.893	4.675

that are not feasible with the more accurate hardware modeling introduced in the simulation. The reference trajectories of the standard case are minimizing the cost, but such optimality does not translate to the real system, as these trajectories are also more brittle and not easy to follow in perturbed scenarios. A property of the robustly optimized trajectory can be seen in Fig. 9, which shows the ratio between lateral and vertical contact forces. The forces obtained with the robust method are further away from the friction cone bounds, whereas the standard method stays closer to them, and even slightly violates the limits (friction cones are modeled as penalties in DDP). Having a higher margin, proves to be beneficial in simulation. In this case the controller, counteracting the perturbations, may produce additional lateral contact forces, which then cause sliding of the contact point. Such sliding is unmodelled in the OCP dynamics, so it is a major source of tracking failure. The proposed method is able to account for this unwanted behaviour without explicitly adding custom terms in the cost function.

Cost landscape: To better understand the impact of the modified framework, in the case of the monoped, an additional investigation is proposed. It explores the value of the standard and robust cost against combinations of the actuator parameters. The costs have been evaluated on a grid of motor mass and gear ratio parameters, reconstructing the landscape of the standard cost \mathcal{L} and the robust one \mathcal{L}_ξ . The task and problem formulation does not change but, in this exploration, the actuator is chosen to be the same for both joints and link lengths are fixed to the nominal values so to visualize the landscape of the cost functions against variations of two parameters in Fig. 10. The robust cost

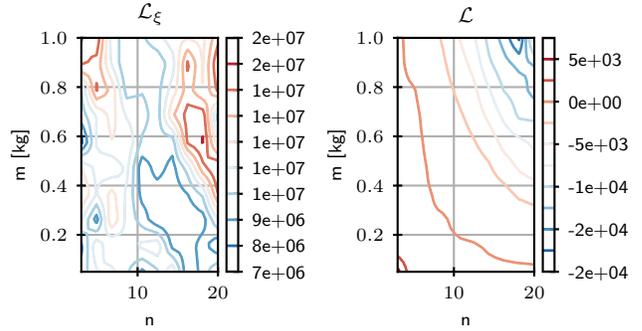


Fig. 10: Monoped jump: contour plot of the reconstructed cost landscapes, as functions of the motor mass m_m and gear ratio n .

adds insights that the standard cost is not able to capture and the two landscapes are different. In the standard case, minimizing \mathcal{L} , the best hardware combination involves large motors and large reductions. Conversely, such choice is highly penalized in the robust case, when joint friction and actuator bandwidth are accounted for. One explanation is that the increased motor size increases the inertia and nonlinear dynamic effects, so the controller has to apply higher feedback torques to compensate for perturbations. Moreover, in our parametric model, larger motors are accompanied by increased rotor inertias, and thus greater reflected inertias. In the robust case instead, smaller motors are selected with a reductions that do not reach the maximum values. This seems to hint that, when robustness comes into play, a more transparent¹ hardware improves performance, which is aligned with recent studies on the subject [28].

IV. CONCLUSIONS AND FUTURE WORK

The main contribution of this work is a co-design framework that includes the information of a feedback controller performance. This approach shows relevant improvement to robust tracking and requires little tuning: hardware and trajectories that are less impacted by noise are selected without explicitly introducing the notion of robustness in the optimal control problem itself. This has been detailed in depth for two different robotic platforms. Interestingly in both cases the selected hardware is chosen to be more transparent, hinting at a trade-off between energetic optimality and the capability to counteract perturbations. The major drawback of the method is the locality of the controller. As future work we plan to investigate its substitution with more general approaches, such as MPC or RL.

¹As detailed in [28] transparent actuators can be obtained by minimizing friction and reflected inertias at the joint level, so with quasi-direct-drive actuation and low rotor inertia. This way the actuator bandwidth and back-drivability are both increased. These properties are necessary for proprioception and rapid control corrections on highly dynamic robotic platforms.

REFERENCES

- [1] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, May 2009.
- [2] P. R. Giordano, Q. Delamare, and A. Franchi, "Trajectory generation for minimum closed-loop state sensitivity," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018.
- [3] G. Bravo-Palacios, A. D. Prete, and P. M. Wensing, "One robot for many tasks: Versatile co-design through stochastic programming," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1680–87, 2020.
- [4] G. Fadini, T. Flayols, A. Del Prete, N. Mansard, and P. Souères, "Computational design of energy-efficient legged robots: Optimizing for size and actuators," in *IEEE Int. Conf. on Robotics and Automation*, Xi'an, China, 2021.
- [5] P. Brault, Q. Delamare, and P. Robuffo Giordano, "Robust trajectory planning with parametric uncertainties," in *IEEE Int. Conf. on Robotics and Automation*, Xi'an, China, 2021.
- [6] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.
- [7] G. Bravo-Palacios, G. Grandesso, A. D. Prete, and P. M. Wensing, "Robust co-design: Coupling morphology and feedback design through stochastic programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 2, p. 021007, Feb. 1, 2022.
- [8] J. Deese and C. Vermillion, "Nested Plant/Controller Codesign Using G-Optimal Design and Continuous Time Adaptation Laws: Theoretical Framework and Application to an Airborne Wind Energy System," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 12, Aug. 2018, 121013.
- [9] S. Boyd, C. Baratt, and S. Norman, "Linear controller design: Limits of performance via convex optimization," *Proceedings of the IEEE*, vol. 78, no. 3, pp. 529–574, 1990.
- [10] A. Sari, C. Espanet, and D. Hissel, "Particle swarm optimization applied to the co-design of a fuel cell air circuit," *Journal of Power Sources*, vol. 179, no. 1, pp. 121–131, 2009.
- [11] D. L. Peters, P. Y. Papalambros, and A. G. Ulsoy, "Sequential co-design of an artifact and its controller via control proxy functions," *Mechatronics*, vol. 23, no. 4, pp. 409–418, 2009.
- [12] T. Dinev, C. Mastalli, V. Ivan, S. Tonneau, and S. Vijayakumar, "Co-designing robots by differentiating motion solvers," *ArXiv: 2103.04660*, 2021.
- [13] R. Rajendra and D. K. Pratihari, "Particle swarm optimization algorithm vs genetic algorithm to develop integrated scheme for obtaining optimal mechanical structure and adaptive controller of a robot," in *Int. Conf. on Information Systems Design and Intelligent Applications*, Visakhapatnam, India, 2012.
- [14] H. V. Hultmann Ayala and L. dos Santos Coelho, "Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator," *Expert Systems with Applications*, vol. 39, no. 10, pp. 8968–8974, 2012.
- [15] J. Tan, T. Zhang, E. Coumans, *et al.*, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv: 1804.10332*, 2018.
- [16] D. Wolz, "Fcmaes - a Python-3 derivative-free optimization library, pypi.org/project/fcmaes/," 2020.
- [17] E. Coumans *et al.*, "Bullet physics library," *Open source: bulletphysics.org*, vol. 15, no. 49, p. 5, 2012.
- [18] E. L. Dantec, M. Taix, and N. Mansard, "First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback," in *Int. Conf. on Robotics and Automation*, Philadelphia, US, 2022.
- [19] N. Rathod, A. Bratta, M. Focchi, *et al.*, "Mobility-enhanced MPC for legged locomotion on rough terrain," *arXiv: 2105.05998*, 2021.
- [20] M. Neunert, M. Stäubli, M. Gifftthaler, *et al.*, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.
- [21] S. Joelle and B. Stephen, "Solving the LQR problem by block elimination," [Online]. Available: stanford.edu/class/ee363/notes/riccati-derivation.pdf.
- [22] F. Grimmering, A. Meduri, M. Khadiv, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, pp. 3650–3657, 2020.
- [23] Y. Yesilevskiy, Z. Gan, and C. Remy, "Optimal configuration of series and parallel elasticity in a 2d monopod," Stockholm, Sweden, 2016.
- [24] Y. Yesilevskiy, Z. Gan, and C. David Remy, "Energy-optimal hopping in parallel and series elastic one-dimensional monopods," *Journal of Mechanisms and Robotics*, vol. 10, no. 3, 2018.
- [25] C. Mastalli, R. Budhiraja, W. Merkt, *et al.*, "Crocodyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *IEEE Int. Conf. on Robotics and Automation*, 2020.
- [26] M. Quigley, "ROS: An open-source robot operating system," in *IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, 2009.
- [27] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6129–6136, 2020.
- [28] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the MIT cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, 2017.