Simulation aided co-design for robust robot optimization

Gabriele Fadini[1], Thomas Flayols[1], Andrea Del Prete[2], Philippe Souères[1]

Abstract— **This paper outlines a bi-level algorithm to concurrently optimize robot hardware and control parameters in order to minimize energy consumption during the execution of tasks and to ensure robust performance. The outer loop consists in a genetic algorithm that optimizes the co-design variables according to the system 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 finds the optimal reference trajectories. Our simulations feature a complete actuation model, including friction compensation and bandwidth limits. This strategy can potentially account for arbitrary perturbations, and discards solutions that cannot robustly meet the task requirements. The results show improved performance of the designed platform in realistic application scenarios, autonomously leading to the selection of lightweight and more transparent hardware.**

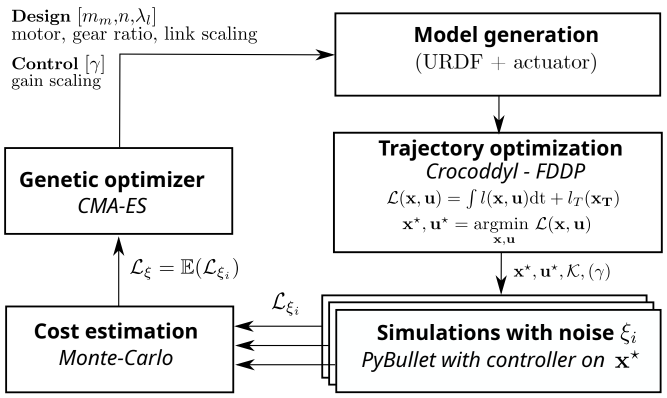

Design $[m_m, n, \lambda_l]$
motor, gear ratio, link scaling
Control $[\gamma]$
gain scaling
Model generation
(URDF + actuator)
Trajectory optimization
Crocoddyl - FDDP
$\mathcal{L}(\mathbf{x},\mathbf{u}) = \int l(\mathbf{x},\mathbf{u})\mathrm{dt} + l_T(\mathbf{x_T})$
$\mathbf{x}^\star, \mathbf{u}^\star = \underset{\mathbf{x},\mathbf{u}}{\mathrm{argmin}}\ \mathcal{L}(\mathbf{x},\mathbf{u})$
$\mathbf{x}^\star, \mathbf{u}^\star, \mathcal{K}, (\gamma)$
Genetic optimizer
CMA-ES
$\mathcal{L}_\xi = \mathbb{E}(\mathcal{L}_{\xi_i})$
$\mathcal{L}_{\xi_i}$
Cost estimation
Monte-Carlo
Simulations with noise ξ_i
PyBullet with controller on $\mathbf{x}^\star$

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

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 is another aspect to consider in order to deploy the 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 a metric that represents the sensitivity of the trajectory to perturbation. 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 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 already large state dimension and time horizon [8], [9]. Controller optimization is often 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 shares 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 (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 which optimizes robot hardware design and control in order to minimize energy consumption and ensure robust performance, is proposed. It relies on a bi-level optimization scheme, with a parallelized version of CMA-ES [16]. The main contribution is to extend this scheme in order to encompass robustness. To this end, we use as metric the average performance of the controller over multiple simulations [17], each one including noise and an actuator model. In simulation, the system is stabilized around an optimal trajectory with the Riccati gains [18], automatically computed by the Differ-

[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 and by ANITI 3IA Institute (ANR-19- P3IA-000)

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 co-design 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         $\mathbf{x}^\star, \mathbf{u}^\star, K \leftarrow$ solveDDP($params$)
7         $\mathcal{L}_\xi$.append(simu($params, \mathbf{x}^\star, \mathbf{u}^\star, \mathcal{K}, \gamma, N_{sim}$))
8     $pop \leftarrow$ evolveCMEAS($pop, \mathcal{L}_\xi$)
9     $N_p \leftarrow N_p + 1$
```

ential 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 just needs to introduce perturbations in the simulations instead than in the OCP, significantly increasing the number of tested scenarios. This comes at the price of the blindness of the inner loop (DDP) to perturbations, which theoretically limits the quality of the outcome. However, our tests show extremely promising results for the design of an energy-efficient robot manipulator and a jumping monoped, which proved to perform better under perturbations than their counterparts designed with our previous framework [4]. Finally we present also how an extension of the explicit optimization of controller parameters is still possible in this framework and can further improve robustness.

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. In an outer loop the co-design parameters (hardware and optional gain scaling γ) 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 $\mathbf{x}^\star, \mathbf{u}^\star$ is then tracked in N_{sim} simulations with a controller using the Riccati gains $\mathcal{K}$ computed by DDP and optionally scaled by γ. 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 $\mathcal{L}_\xi$ 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 $\mathcal{L}_\xi$ is parallelized asynchronously in order to speed up the algorithm. Unlike stochastic optimization, the number of these perturbed scenarios can be rather high (up to 10^3), at a reduced computational cost of multiple simulation runs (linear in the number of time-steps and simulations $O(N_{sim})$). This way the genetic algorithm selects the hardware leading 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 using model predictive control (MPC) [19], [20]. Another method is to learn offline a 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 treating relatively small deviations from the reference trajectory, they 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. 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 optimal, as detailed in [21], and an extension may be possible also with equality or inequality constraints. To follow the reference $\mathbf{x}^\star$ from $\mathbf{x} \approx \mathbf{x}^\star$, the control $\mathbf{u}$ includes a feedback term linear with respect to the state error:

$$\mathbf{u} = \mathbf{u}^\star + \gamma \mathcal{K}(\mathbf{x} - \mathbf{x}^\star) \tag{1}$$

The gain matrix $\mathcal{K}$ maps state deviations to corrections of the control input. The matrix $\mathcal{K}$ can optionally be multiplied by a gain scaling factor $\gamma \in \mathbb{R}$, ($\gamma = 1$ by default). This controller requires almost 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: To implement a control law such as (1) in a realistic application, 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 control $\boldsymbol{\tau}$ is computed so to compensate friction. To this end, in Eq. 2, the torque $\mathbf{u}$ includes compensation terms for

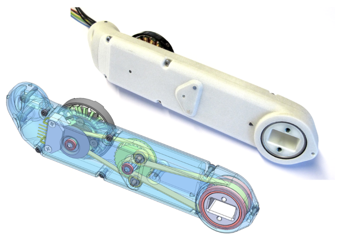


Fig. 2: Actuator module, showing the belt-drive transmission and the BLDC motor placement, courtesy of ODRI [22].

joint static friction $\boldsymbol{\tau}_{\mu,0}$ [Nm] and for joint damping $\mathbf{b}$ [Nms].

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

Such correction compensates friction from the 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 $\mathbf{v}$.

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

$$\mathbf{u}_k = \alpha\boldsymbol{\tau}_k + (1-\alpha)\boldsymbol{\tau}_{k-1}, \ \text{for}\ k \in 1,..N-1 \tag{3}$$

where the initial torque is imposed $\mathbf{u}_0 = \mathbf{u}_0^\star$ 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 joint torques that can reasonably be applied within the bandwidth limitations of each actuator. Finally, a saturation is introduced to enforce the torque limits: $\underline{\mathbf{u}} < \mathbf{u}_k < \overline{\mathbf{u}}$.

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 $\boldsymbol{\tau}_{\mu,0}$) 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 joint inertia adding the rotor inertia
2) adding the motor mass to the link

Power components

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

$$P_m = \mathbf{u}^{\star\top}\mathbf{v}_a\ [W] \tag{4}$$

$$P_f = \left(\boldsymbol{\tau}_{\mu,0}\,\mathrm{sign}(\mathbf{v}_a) + \mathbf{b}\,\mathbf{v}_a\right)^\top \mathbf{v}_a\ [W] \tag{5}$$

$$P_t = \mathbf{u}^\top \tilde{\mathbf{K}}\mathbf{u}\ [W] \tag{6}$$

Mechanical power P_m is shown in Eq. (4) where $\mathbf{u}^\star$ is the ideal torque at the joint and $\mathbf{v}_a$ is the actuated joint velocities. *Joint friction power loss P_f* as in Eq. (5) corresponding to the joint friction, reconstructed from $\mathbf{v}_a$ multiplied by the friction compensation torque introduced in Eq. (2).
Joule dissipation power loss P_t is obtained as in Eq. (6) from the value of the torque on the joint $\mathbf{u}$, including friction compensation as in Eq. (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 in Eq. (4), Eq. (5) and Eq. (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 Eq. (2) is reconstructed and then the overall associated energy minimized.

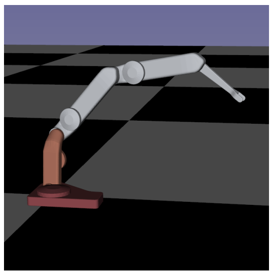
Ensembled final cost

The main feature of the introduced method is to consider perturbation sources for each simulation $\boldsymbol{\xi} \in \mathbb{R}^{N_{sim}\times n_u \times N}$, where n_u is the number of actuated joints and N the number of timesteps. The single joint torque noise realization $\boldsymbol{\xi}_i$ with $i \in \{0, ..., N_{sim}-1\}$ acts on the ideal joint torque of the robot $\mathbf{u}^\star$, proportionally to its value: $\boldsymbol{\xi}_i \sim \mathcal{N}(0, \sigma^2)\,\mathbf{u}^\star$. This is motivated as the source of perturbation has been shown not to greatly impact the robust solution in [25]. Given $\boldsymbol{\xi}_i$, 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 $\mathcal{L}(\mathbf{x}^\star, \mathbf{u}^\star)$, but will enrich it with that from simulations with the control correction. Since each $\mathcal{L}_{\xi_i}$ is a random variable, depending on the realizations of the noise ξ_i, its expected value $\mathcal{L}_\xi$ is obtained using a Monte-Carlo approach. Finally CMA-ES minimizes $\mathcal{L}_\xi$.

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

III. Results

In this section some results, including robustness in the co-design framework, are presented. To solve the OCP and obtain the Riccati gains, we use `crocoddyl` [26] an open-source DDP solver based on the robot dynamics library `pinocchio` [27], [28]. A custom URDF is shared between the OCP and the simulator `PyBullet` [17]. It is generated parametrically using the ROS module `xacro` [29]. In an initial phase we optimize just hardware parameters (fixing the gain scaling $\gamma = 1$). Two types of robot, shown in Fig. 3, are optimized for robust task tracking. 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], [30].


(a) Manipulator

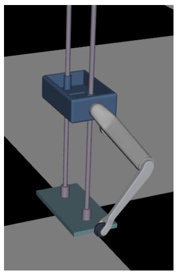

(b) Monoped

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

Co-design parameters: The hardware optimization involves 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 link scaling λ_l represents the ratio between the link length and the current module design length. A single scaling parameters is used because, once a given length is chosen, to keep the relative deflection of the link constant, the section needs to change too according to a predefined law as outlined in [4]. This change will also modify the inertia of the links. In the final section of the results, hardware and controller parameters are optimized together in the case of the manipulator. We show that this way it is possible to increase robustness even further by adding the gain scaling parameter γ (introduced in Eq. 1) to the co-design variables.

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 $p = p_0 + [-0.1, -0.1, 0.1]^\top$ m, and then bring it back to p_0. At the intermediate and final positions the joint velocities must be zero.

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 Tab. I. For the realization ξ the value of $\sigma_\xi = 0.2$ was selected and $N_{sim} = 100$.

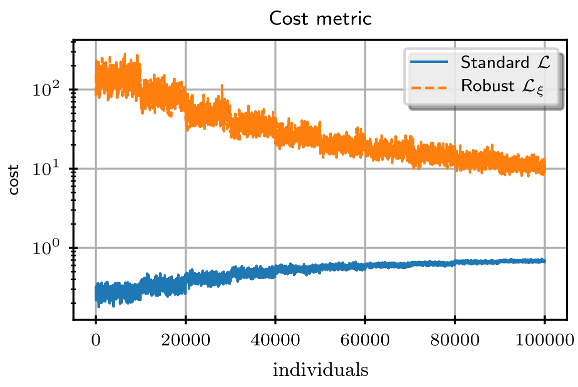


Fig. 4: Robust cost metric $\mathcal{L}_\xi$ evolution trend during CMA-ES (orange curve). $\mathcal{L}_\xi$ diminishes increasing the number of runs of Alg. 1 (the number of the overall evaluated individuals), while the standard cost $\mathcal{L}$ increases (blue curve).

Discussion: The values obtained with the standard method (without robustness optimization) and its robust counterpart are compared. Fig. 6b shows that the standard 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. Position and velocity tracking is better in the robust case as illustrated qualitatively in Fig. 6a and quantitatively by the lower RMSE = $\sqrt{\sum_{i=0}^{N_{sim}} ||\mathbf{x}_{\xi_i} - \mathbf{x}^\star||_2^2 / N_{sim}}$ in Tab. II. The hardware parameters in Tab. 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 as reported in Tab. II. 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 Tab. II. Reasonably, both methods select the link size to reduce the moving masses and system inertia. In Fig. 4 the cost metric profiles in case of the standard and the robust approach are shown versus the evolution of CMA-ES (all the evaluated individuals of each generation). Here we notice that the improvement of the robust metric $\mathcal{L}_\xi$ is accompanied with a degradation of the standard metric $\mathcal{L}$. 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 $\mathcal{L}_\xi$ produces designs that are able to follow more closely the task under perturbations. In Fig. 5 the convergence of $\mathcal{L}_\xi$ 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 $\mathcal{L}_\xi$ completely converged. Fig. 5 also shows that the computational time is roughly linear with the number of simulations (blue curve).

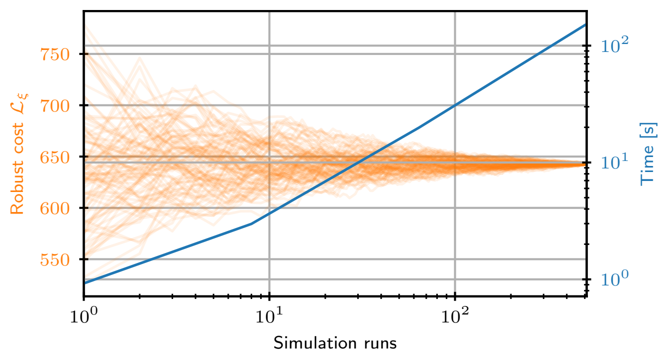


Fig. 5: Monte Carlo evaluation of $\mathcal{L}_\xi$ progressively increasing the number of simulations N_{sim}. For a number of simulation runs each orange curve represents the value of $\mathcal{L}_\xi$ obtained with a different random seed, in blue the associated computation time is plotted.

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

Quantity	**Robust**	**Standard**
Cost $\mathcal{L}$	$9.58\mathrm{e}-3$	$3.5\mathrm{e}-3$
Cost $\mathcal{L}_\xi$	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
$\sum_i P_{m,i}\mathrm{d}t$ [J]	-0.9	-1.7
$\sum_i P_{t,i}\mathrm{d}t$ [J]	6.5	1.8
$\sum_i P_{f,i}\mathrm{d}t$ [J]	1.3	3.2

Monoped jump

We selected a jumping task, a more complex scenario involving contact phases, to optimize a hopping leg.

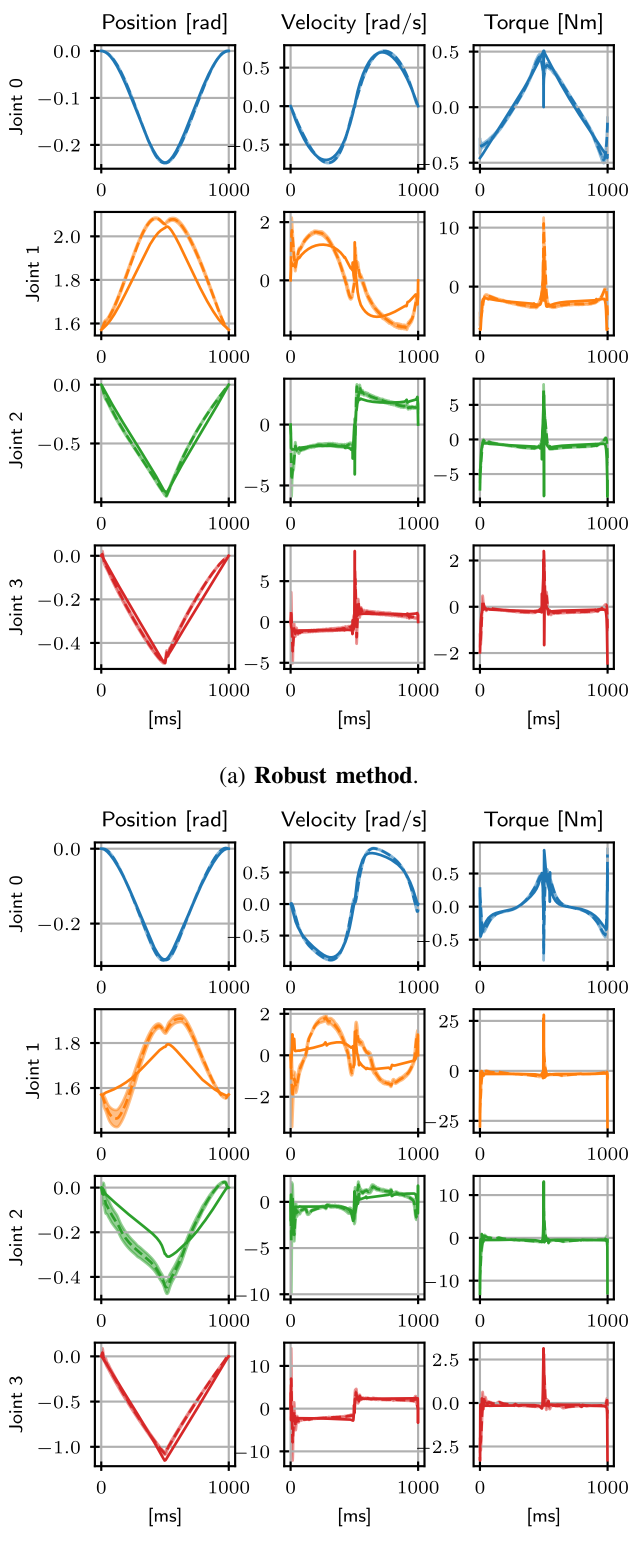
Task: The robot has to perform a jump with the base in a given time, and stabilize the system after touch-down. In the OCP this task is enforced weakly as a penalty on the prismatic joint z position.

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

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

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

	Robust		**Standard**	
Quantity	$\mu(\sum r)$	$\sigma(\sum r)$	$\mu(\sum r)$	$\sigma(\sum r)$
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



(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.

TABLE IV: Results for the monoped co-design

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

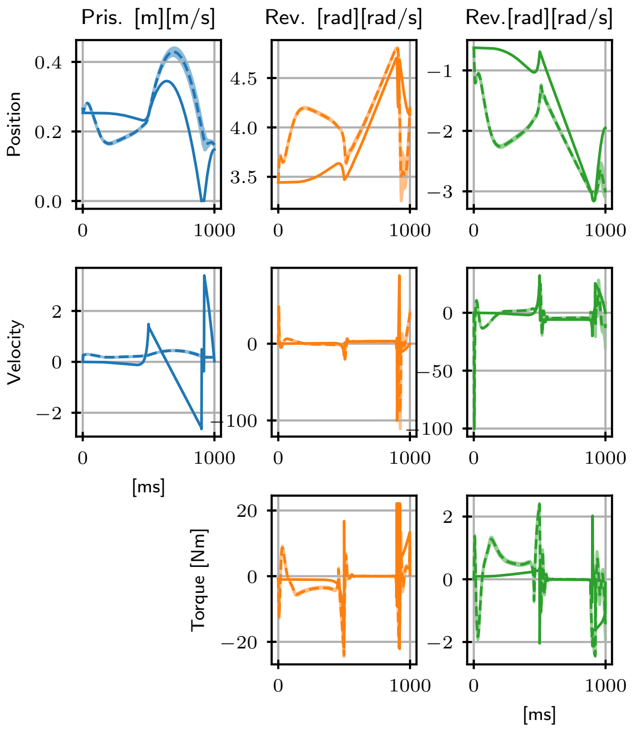
find solutions that satisfy the task timing sequence and the dynamics. In the OCP there are four predefined phases for the jumping motion:

- *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 monoped 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.

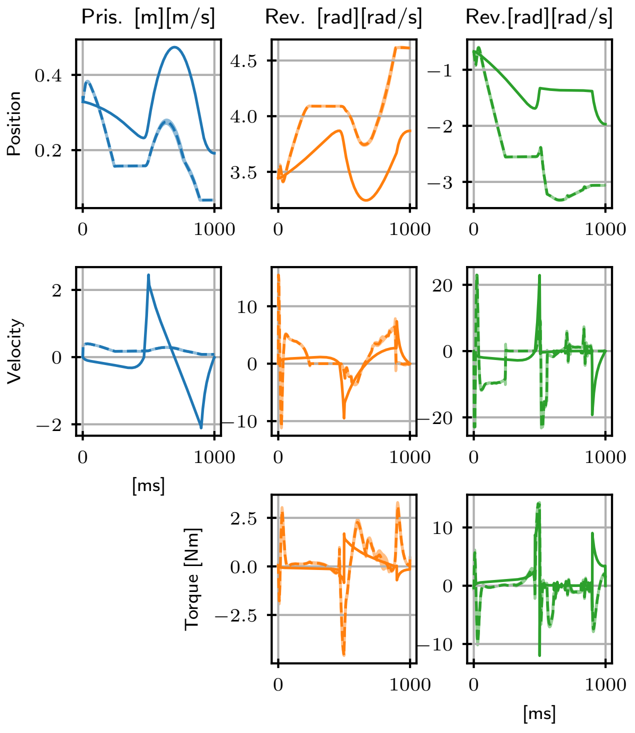
Hyper-parameters: For CMA-ES the following parameters were chosen: $N_{gen} = 5, N_{pop} = 1000$. The OCP has 1000 nodes and $dt = 1$ ms; the cost weights are reported in Tab. IV. For the realization ξ the value of $\sigma_\xi = 0.2$ was selected and $N_{sim} = 100$.

Cost comparison: Fig. 7 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 monoped perform worse, anticipating the contact phase with the ground. This means that the optimized motion needs high accelerations 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.

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



(a) **Robust method**.



(b) **Standard method**.

Fig. 7: 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. Note that the prismatic joint is underactuated.

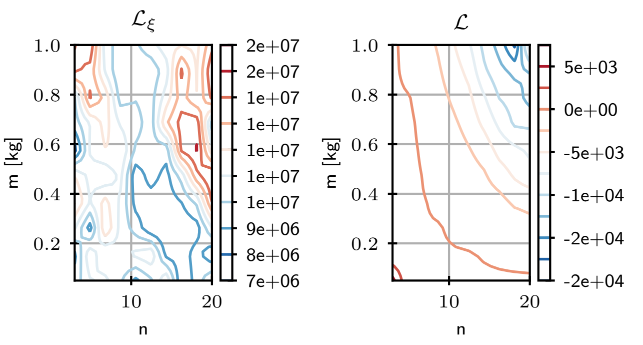


Fig. 8: Monoped jump: contour plot of the robust and standard cost as functions of the motor mass m_m and gear ratio n. Two very different optimal regions can be seen

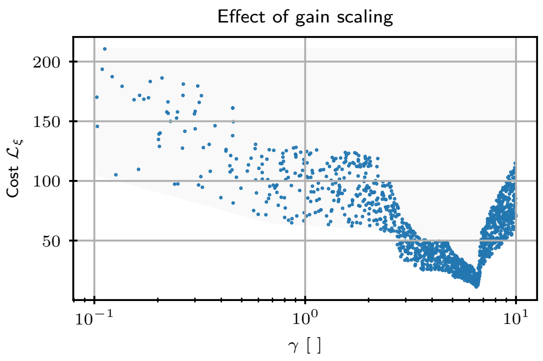


Fig. 9: Effect of the scaling γ with respect to the robust cost $\mathcal{L}_\xi$. Each dot represents an individual of the optimization. The presence of a minimum can be seen

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 $\mathcal{L}_\xi$. 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. 8. The robust cost adds insights that the standard cost is not able to capture, hence resulting in different cost landscapes. 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 non-linear 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, more transparent[1] hardware improves performance, which is aligned with recent studies on the subject [31].

Controller parameters optimization

To further improve the controller robustness, the optimization of the gain scaling factor γ (as in Eq. 1) was introduced in the outer loop. In this case, for the same task in III, a single actuator choice was optimized to be used for all the manipulator joints and with $\gamma \in [0.1 - 10]$. The optimization was done with a population of 1000 individuals evolved for 5 generations. The optimum was found for hardware values in a similar range of the previous manipulator results, at $m_m = 0.05$, $n = 15.7$, $\lambda = 1.13$ and for $\gamma = 6.57$. The value of the robust cost though is significantly lower than what observed in the case of unscaled gains, with the value $\mathcal{L}_\xi = 10.17$. Plotting the values of $\mathcal{L}_\xi$ against the γ results in Fig. 9. Here we a value of γ greater than the unit is overall leading to a decrease of $\mathcal{L}_\xi$. The explanation is that the Riccati gains obtained by DDP are optimal but only in a close proximity with respect the ideal trajectory. This is not the case with the addition of perturbations in the simulator: an amplification of the correction seems to be a good option to reach the final goal. It can be observed however that this trend stops at higher values of γ, where a sharp increase of the robust cost is induced instead. With this last example shows that, with this framework, optimizing even simple controller parameters is possible at a reduced computational cost.

[1]As detailed in [31] 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

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 in simulation. This addition, with respect to our previous contribution [4], is an important step to evaluate the hardware properties *in silico* before starting the system integration of real prototypes. Our approach shows relevant improvement to robust tracking and requires little tuning: the hardware and trajectories that are less impacted by noise are selected without explicitly introducing the notion of robustness in the OCP 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 lies in the local nature of the controller. To overcome this limitation, an extension was proposed and successfully tested. It consist in optimizing a scaling of the feedback gains at the same level as the design parameters, in the external loop. As future work we plan to investigate the substitution of the controller with even more general approaches, such as MPC or RL, which allow a online replanning of the trajectory.

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 ICRA*, Brisbane, Australia, 2018.

[3] G. Bravo-Palacios, A. Del 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 ICRA*, Xi’an, China, 2021.

[5] P. Brault, Q. Delamare, and P. Robuffo Giordano, “Robust trajectory planning with parametric uncertainties,” in *IEEE ICRA*, 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. Del 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. 021 007, 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, 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,

[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,

[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. Pratihar, “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*,

[16] D. Wolz, “Fcmaes - a Python-3 derivative-free optimization library, pypi.org/project/fcmaes,”

[17] E. Coumans *et al.*, “Bullet physics library,” *Open source: bulletphysics. org*, vol. 15, no. 49, p. 5,

[18] E. L. Dantec, M. Taix, and N. Mansard, “First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback,” in *IEEE ICRA*, Philadelphia, US, 2022.

[19] N. Rathod, A. Bratta, M. Focchi, *et al.*, “Mobility-enhanced MPC for legged locomotion on rough terrain,” *arXiv: 2105.05998*,

[20] M. Neunert, M. Stäuble, M. Giftthaler, *et al.*, “Whole-body nonlinear model predictive control through contacts for quadrupeds,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465,

[21] J. Skaf and S. Boyd, “Solving the LQR problem by block elimination,” stanford.edu/class/ee363/notes/riccati-derivation.pdf.

[22] F. Grimminger, 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 monoped,” Stockholm, Sweden, 2016.

[24] Y. Yesilevskiy, Z. Gan, and C. David Remy, “Energy-optimal hopping in parallel and series elastic one-dimensional monopeds,” *Journal of Mechanisms and Robotics*, vol. 10, no. 3, 2018.

[25] A. D. Prete and N. Mansard, “Robustness to joint-torque-tracking errors in task-space inverse dynamics,” *IEEE T-RO*, vol. 32, no. 5, pp. 1091–1105, 2016.

[26] C. Mastalli, R. Budhiraja, W. Merkt, *et al.*, “Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control,” in *IEEE ICRA*, 2020.

[27] J. Carpentier, G. Saurel, G. Buondonno, *et al.*, “The pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives,” in *IEEE SII*, 2019.

[28] J. Carpentier and N. Mansard, “Analytical derivatives of rigid body dynamics algorithms,” in *RSS*, 2018.

[29] M. Quigley, “ROS: An open-source robot operating system,” in *IEEE ICRA*, Kobe, Japan, 2009.

[30] 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.

[31] 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 T-RO*, vol. 33, no. 3, pp. 509–522, 2017.