2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Towards the best robot for the job: Optimising actuation design through multi-task co-design and component selection

Wesley Roozing[1], Jonathan Schaaij, and Alessandro Forino[2]

***Abstract*— We propose a multi-task co-design approach to design a robot's actuation (motor sizes and gear ratios) based on trajectory optimisation. Leveraging an actuation model fit on data of series of components, we find the optimal set of design parameters for all joints over a set of representative tasks for the given robot. Critically, we close the loop towards component selection, given a finite set of available components. This enables more practical use of co-design tools. Our results show that the method is effective, and critically, show that it is possible to find a robot design that is capable of performing an entire set of tasks at an efficiency that is comparable to a robot co-designed for each specific task. Finally, we perform an extensive analysis of hyperparameter effects, and select discrete actuation components from catalogues and compare to co-design results.**

I. Introduction

The mechatronic design of robotic systems is challenging task, relying on the engineer's expertise, simplifying assumptions, and iteration (often with simulation) to achieve satisfactory designs. This is time consuming and typically leads to suboptimal results. Moreover, design and behaviour are not independent; design changes alter the robot's dynamics, and thereby how it is best for the robot to move.

Co-design is a class of methodologies that approaches robot design as a concurrent optimisation problem of design and control. This rapidly growing field has spawned numerous methods, that can be broadly categorised in either bi-level optimisation [1]–[6] and single-level optimisation [7]–[13] methods. In the former, most methods rely on genetic algorithms as outer loop to evolve the robot design, with an inner loop based on trajectory optimisation (TO) to optimise motion [1]–[3]. Others use differentiable motion planners [4] or stochastic programming [5], [6]. Conversely, single-level optimisation methods generally extend a TO problem [14] with design parameters [7]–[9], or rely on sensitivity analysis to simultaneously alter design and motion [11]–[13] on a manifold.

Another way these works can be categorised is by what design elements are optimised; morphology [11]–[13], actuation [7], [9], or both [1]–[6], [8], [10], although most methods can be relatively easily extended to include other design parameters. The authors of [2], [6] provide overviews of the capabilities of some of these (and other) approaches.

One aspect that is not addressed by most works is the fact that a robot typically needs to perform a range of tasks, rather than one or two specific tasks with fixed parameters. Notable exceptions are [5], [6] that rely on stochastic programming with TO to address multi-task optimisation.

This work addresses two main shortcomings of existing works: 1) analysis of single-task compared to multi-task design, and 2) closing the loop towards component selection, given a finite set of available components (i.e., motors and gear reducers). We formulate a continuously parametrised actuation model for both the motor and gearing, similar to approaches in [1], [4], [7], [8]. However, compared to (most of) those approaches, ours 1) includes the (reflected) drivetrain inertia, 2) parametrises gearbox-dependent friction, and 3) optimises the design of each joint separately. Using this model we develop a co-design procedure based on TO that optimises the robot design over a large set of tasks.

The main contributions are as follows:

1) A co-design procedure that optimises a robot's actuation design parameters on a large set of tasks, and, importantly, closes the loop towards component selection from co-design results;
2) An actuation model fit on motor and gearbox catalogue and experimental data that includes (reflected) motor inertia and gearbox-dependent friction;
3) Extensive co-design and simulation results of a robot manipulator that validate the approach, including an evaluation of hyperparameter effects, comparison of single-task and multi-task co-design, and comparison of co-design results with selected components.

The remainder of this paper is structured as follows. Sec. II introduces the co-design approach, followed by the continuously parametrised actuation model in Sec. III. Sec. IV presents extensive results analysing our approach. Finally, we conclude the work in Sec. V.

II. Co-design approach

A. Single task co-design using trajectory optimisation

A single co-design problem is formulated as a trajectory optimisation (TO) problem, extended to include the robot design parameters. It has the following general form:

$$\min_{\boldsymbol{x},\boldsymbol{u},T,\boldsymbol{\gamma}} \quad J(\boldsymbol{x},\boldsymbol{u},T,\boldsymbol{\gamma},\boldsymbol{\lambda}), \tag{1}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\gamma}) \qquad \forall\, t \in [0,T], \tag{2}$$

$$\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t),T,\boldsymbol{\gamma},\boldsymbol{\lambda}) \leq \boldsymbol{0} \qquad \forall\, t \in [0,T], \tag{3}$$

$$\boldsymbol{g}(\boldsymbol{x}(0),\boldsymbol{x}(T),T,\boldsymbol{\gamma},\boldsymbol{\lambda}) = \boldsymbol{0}, \tag{4}$$

where $\boldsymbol{x}$ denotes the system state, $\boldsymbol{u}$ the control input, T the final time, $\boldsymbol{\gamma}$ the design parameters, and t the time variable. $\boldsymbol{\lambda}$ denotes the task parameters, e.g., start and end states.

[1]W. Roozing is with the Robotics & Mechatronics (RaM) group, University of Twente, The Netherlands. E-mail: w.roozing at utwente.nl.

[2]A. Forino is with maxon motor AG, Switzerland. E-mail: alessandro.forino at maxongroup.com.

Furthermore, (1) describes minimisation of the cost function J, (2) the dynamics of the system $\boldsymbol{f}$ constraining the state time derivative $\dot{\boldsymbol{x}}$, (3) the design and path constraints $\boldsymbol{h}$, and (4) the boundary conditions $\boldsymbol{g}$. The cost function is given in standard form, comprised of a running and terminal cost, as:

$$J(\boldsymbol{x}, \boldsymbol{u}, T, \boldsymbol{\gamma}, \boldsymbol{\lambda}) = \int_0^T J_{\mathrm{run}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\gamma})\,\mathrm{d}t + J_{\mathrm{term}}(\boldsymbol{x}(T), T, \boldsymbol{\gamma}, \boldsymbol{\lambda}) \tag{5}$$

where J_{run} and J_{term} denote the running and terminal cost, respectively. In the following, for notational simplicity, we will omit the time dependence of variables in most cases.

B. Co-design over multiple tasks

Consider a set of tasks $\boldsymbol{\Lambda}$, each defined by task parameters $\boldsymbol{\lambda}_i \in \boldsymbol{\Lambda}$. Each task has associated state and input evolutions $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$, respectively, and final time T_i, while all tasks are coupled through the design parameters $\boldsymbol{\gamma}$. The design is then optimised over this set of tasks by minimising the total cost:

$$\min_{\{\boldsymbol{x}_i, \boldsymbol{u}_i, T_i\}, \boldsymbol{\gamma}} \sum_{i=1}^{N_{\mathrm{t}}} J(\boldsymbol{x}_i, \boldsymbol{u}_i, T_i, \boldsymbol{\gamma}, \boldsymbol{\lambda}_i), \tag{6}$$

subject to the same constraints as in (1)–(4) for each task, and where N_{t} denotes the number of tasks in the set $\boldsymbol{\Lambda}$. Shown graphically in Fig. 1, we refer to this as *multi-task co-design*. Such an optimisation problem is expected to find the 'average best' robot, for the given set of tasks. Hence, it is important to define the set $\boldsymbol{\Lambda}$ as a representative set of tasks the robot will see in use. In our case all tasks are weighted equally; optionally, one could weigh the individual tasks according to their relative occurrence or importance, as done by e.g. Bravo-Palacios et al. [5], [6].

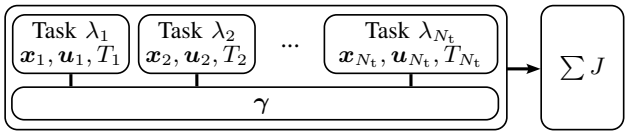


Fig. 1: Multi-task co-design: Multiple OCPs are solved simultaneously, with the design parameters $\boldsymbol{\gamma}$ shared across the OCPs.

C. Robot and drivetrain dynamics

In this work we will consider fully actuated n-DoF robots, with their state expressed in the usual Lagrangian canonical coordinates $\boldsymbol{q} \in \mathbb{R}^n$ and their time derivative $\dot{\boldsymbol{q}} \in \mathbb{R}^n$. The system's rigid-body dynamics are expressed by the standard Euler-Lagrange equation for a robotic system:

$$D(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{7}$$

where $D(\boldsymbol{q})$ denotes the inertia matrix, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the Coriolis matrix, $\boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q})$ the gravity vector, and $\boldsymbol{\tau}$ the applied joint torques. We have omitted the dependence on $\boldsymbol{\gamma}$ here. The mass of the actuation components is included in the parametrisation of the dynamics matrices through $\boldsymbol{\gamma}$; however, we explicitly formulate the reflected inertia and friction effects of the drivetrain. To this end we rewrite (7) as

$$(D(\boldsymbol{q}) + N^2 I)\,\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{f}}(\dot{\boldsymbol{q}}) + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) = N\,\boldsymbol{\tau}_{\mathrm{m}}, \tag{8}$$

where N and I denote diagonal matrices of transmission ratios and rotor inertiae, respectively, $\boldsymbol{\tau}_{\mathrm{f}}$ denotes the drivetrain friction model, and $\boldsymbol{\tau}_{\mathrm{m}}$ denotes induced electromotive motor torque as input. The parametrisation of all these will be further elaborated in Sec. III. Finally, Eq. (8) may be rewritten into the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ of (2) by choosing the augmented state $\boldsymbol{x} = [\boldsymbol{q}^{\mathsf{T}}, \dot{\boldsymbol{q}}^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{2n}$ and input $\boldsymbol{u} = \boldsymbol{\tau}_{\mathrm{m}}$.

D. Cost function and constraints

1) Cost function: Most established literature on trajectory optimisation of mechatronic and robotic systems relies on some form of minimisation of effort as running cost. Typically this takes the shape of $\|\boldsymbol{\tau}\|_2$, where $\boldsymbol{\tau}$ is the input of the robot's rigid-body dynamics (7), and where $\|\cdot\|_2$ denotes the 2-norm. In addition to neglecting drivetrain (reflected) inertia, these approaches miss critical aspects of electromechanical systems, including friction and the ability to dissipate negative mechanical power in electrical resistance, as well as possible regeneration of energy. Therefore, while effective at pure trajectory optimisation, this is a poor choice for co-design of efficient robotic systems. Therefore, co-design literature (e.g., [1], [4], [5]) typically leverages electrical power as running cost. In our case, we sum the electrical power over all joints:

$$J_{\mathrm{run}} = \sum_{i=1}^{n} P_i(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\gamma})^+, \tag{9}$$

where $(\cdot)^+$ denotes the positive part of the argument[1], indicating that the motor driver is assumed not to be four-quadrant capable (no regeneration). P_i denotes electrical power of the i-th joint and will be formulated in Sec. III. The terminal cost is defined as follows:

$$J_{\mathrm{term}} = w_{\mathrm{t}}\,T + w_{\mathrm{pos}}\left(\boldsymbol{p}^* - \boldsymbol{p}(\boldsymbol{x}(T))\right)^2, \tag{10}$$

where the first term comprises the final time with a weighing factor w_{t}, the use of which allows to balance the result (and thereby the robot design) between efficiency and speed, as will also be shown in the results in Sec. IV. The second term comprises a cost on the end-effector configuration $\boldsymbol{p}$ at the final time compared to some target configuration $\boldsymbol{p}^*$, again weighed by a (large) weighting factor w_{pos}. Note that this term may be replaced by a boundary condition $\boldsymbol{p}(\boldsymbol{x}(T)) = \boldsymbol{p}^*$ in (4), however we find that such a soft constraint improves convergence on many problems while not significantly altering the result.

2) (In)equality constraints: The design and path constraints (3) are given by the following inequalities:

$$T > 0, \tag{11}$$

$$\boldsymbol{q}_{\min} \le \boldsymbol{q}(t) \le \boldsymbol{q}_{\max} \qquad \forall\, t \in [0, T], \tag{12}$$

$$|\dot{\boldsymbol{q}}(t)| \le \dot{\boldsymbol{q}}_{\max} \qquad \forall\, t \in [0, T], \tag{13}$$

$$|\boldsymbol{u}(t)| \le \boldsymbol{u}_{\max} \qquad \forall\, t \in [0, T], \tag{14}$$

$$P_i(\boldsymbol{x}(t), \boldsymbol{u}(t)) \le P_{\max,i} \qquad \forall\, i \wedge t \in [0, T], \tag{15}$$

$$\boldsymbol{\gamma}_{\min} \le \boldsymbol{\gamma} \le \boldsymbol{\gamma}_{\max}, \tag{16}$$

[1]To ensure continuous differentiability, this function is approximated by a tanh-based function.

constraining time to be positive, joint limits, maximum joint velocities, motor torque limits, maximum electrical power per joint, and design parameters, respectively. Constraints (13)–(16) are parametrised on the system state, inputs, and design parameters $\boldsymbol{\gamma}$ through the actuation model, which will be elaborated in Sec. III. Finally, the equality constraints (4) are comprised of the initial state and final velocity:

$$\boldsymbol{x}(0) = \left[\boldsymbol{q}_0^\mathsf{T}, \dot{\boldsymbol{q}}_0^\mathsf{T}\right]^\mathsf{T}, \quad \dot{\boldsymbol{q}}(T) = \mathbf{0}. \tag{17}$$

Together with the target end-effector pose $\boldsymbol{p}^*$, the initial state defines each task: $\boldsymbol{\lambda}_i = \{\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \boldsymbol{p}^*\}_i$.

E. Transcription using direct collocation

In order to solve the optimal control problem described by (1)–(4), it is typically transcribed to a finite-dimensional nonlinear program (NLP) that may be numerically solved by a computer. In this work, we use direct collocation; we refer to [15] for a gentle introduction. Within collocation methods different quadratures may be used; we highlight a few important considerations for robotic systems.

First, several authors [16], [17] have shown that the assumption of dynamics of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, together with state augmentation $\boldsymbol{x} = [\boldsymbol{q}^\mathsf{T}, \dot{\boldsymbol{q}}^\mathsf{T}]^\mathsf{T}$ for second-order mechanical systems, leads to inconsistencies between the resulting position, velocity, and acceleration. The reason is that some states are integrals of others, while they are approximated by the same order polynomial. To this end, [16] propose a set of second-order collocation methods that use different collocation constraints for the position and velocity, which are consistent.

Secondly, other authors [18]–[20] have noticed that imposing dynamics constraints based on inverse dynamics has computational benefits, in terms of computation time, error, and convergence properties. We use the inverse dynamics method based on semi-implicit Euler proposed by [18].

F. Software Implementation

Fig. 2 shows an overview of the software implementation of the proposed co-design approach. We leverage standard URDF-based robot descriptions together with the Pinocchio rigid-body dynamics library [21] and the CasADI [22] symbolic package for automatic differentiation[2]. The NLP is solved by IPOPT [24] with MUMPS as solver. All code is implemented in Python 3.9 and we are working on refactoring towards open-sourcing the tools.

To enable efficient implementation of (8) using Pinocchio, the URDF file includes reference points for the actuators and end-effector (load), such that their mass can be added (Fig. 6a). Combining the robot model with the parametric actuator model symbolically, we formulate the co-design problem which is then transcribed to an NLP and solved.

Tasks are defined as start and end configurations of the robot. To initialise the optimisation these task definitions

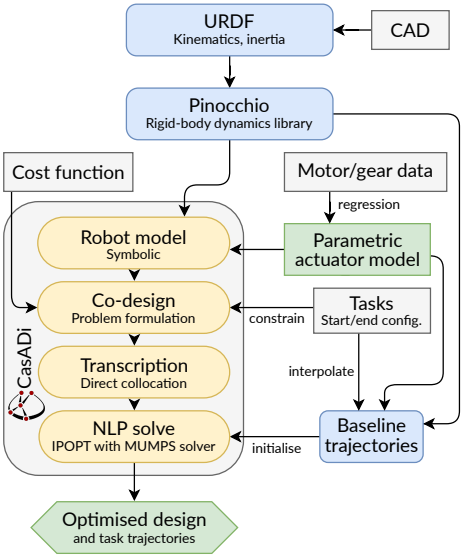


Fig. 2: Overview of the co-design implementation.

are used to provide linearly interpolated (in either joint or Cartesian space, and using trapezoidal velocity profiles) baseline trajectories. The Recursive Newton-Euler Algorithm (RNEA) is then used to initialise the joint torques.

III. Parametrised Actuation Model

The design of an actuation system typically involves selecting discrete options from a finite set of available components. That is, a finite set of motor and gear sizes and gear ratios that are available. This leads to a mixed-integer optimisation problem which rapidly grows in complexity as the number of joints and available components increase. Therefore, we find a continuously parametrised model for both the motor and gearing, similar to approaches in [1], [4], [7], [8]. However, compared to (most of) those approaches, ours 1) includes the (reflected) motor rotor inertia, 2) parametrises gearbox-dependent friction values, and 3) optimises the motor and gear ratio of each joint separately.

A. Parameter regression

The aforementioned works have found that the motor mass serves well as the base variable for regression of important motor parameters such as rotor inertia, motor constant, and stall torque. We proceed similarly.

In this work we regress on data of the maxon EC-flat motor series and compatible GP series of gearboxes. The regression results for the motor parameters is shown in Fig. 3, showing the rotor inertia I, nominal power P_{nom}, motor constant K_{m}, nominal torque τ_{nom}, and stall torque τ_{stall}, with linear regression[3] on the motor mass m_{motor}. It is important to note that limiting characteristics such as nominal power and torque do not fit well as they depend heavily on thermal (boundary) constraints. This is clearly reflected in this data as well. We apply a similar approach to the fitting of gearbox friction parameters.

[2]The specific implementation uses a pre-release of Pinocchio version 3 with CasADi bindings in Python [23]. As of the time of writing, a release version has become available.

[3]Chosen for simplicity; some works instead use exponential regression. The drawback of linear regression is that the fit on our particular data set produces negative inertia for very small motor sizes (not used here).

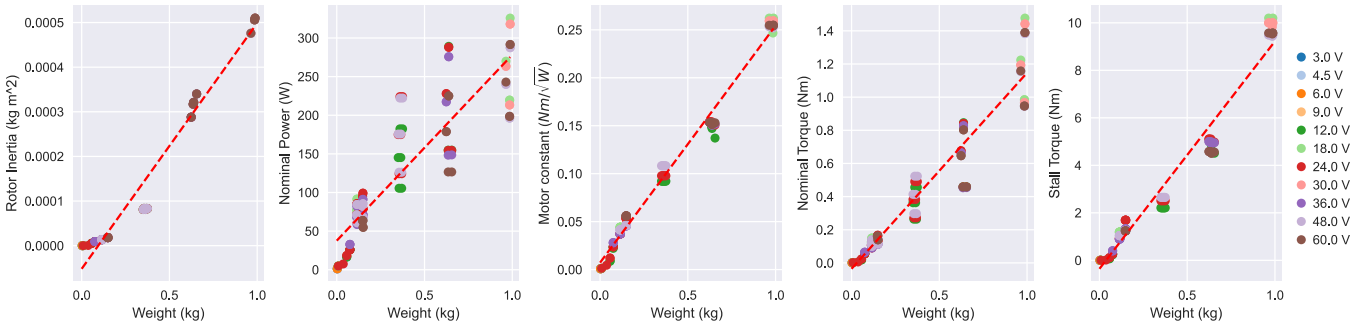


Fig. 3: Linear regression for parameters of the Maxon EC-flat series motors. Colours denote the nominal voltage.

B. Actuation model

To develop the actuation model we consider one individual joint. We begin by formulating the electrical power P_i in (9). Let τ_m be the electromotive torque of the motor, $\omega_m = n\,\dot{q}$ the rotor speed corresponding to some robot joint with speed $\dot{q}$, and n the gear ratio. The electrical power is then

$$P = \omega_{\mathrm{m}}\tau_{\mathrm{m}} + \frac{\tau_{\mathrm{m}}^2}{K_\tau^2}R\,, \tag{18}$$

where K_τ denotes the torque constant and R the winding resistance. This equation may be rewritten using the motor constant $K_{\mathrm{m}} = K_\tau/\sqrt{R}$ as:

$$P = \omega_{\mathrm{m}}\tau_{\mathrm{m}} + \frac{\tau_{\mathrm{m}}^2}{K_{\mathrm{m}}^2}. \tag{19}$$

The motor constant is independent of the motor's voltage rating and correlates well with motor mass. We set the maximum power (in (15)) to $P_{\mathrm{max},i} = P_{\mathrm{nom}}$ for a given motor. Similarly, $\tau_{\mathrm{m,max}}$ ($\boldsymbol{u}_{\mathrm{max}}$ in (14)) is set to $3\tau_{\mathrm{nom}}$ of a given motor, which is a torque that most motors in the considered size range can produce for a couple of seconds. While these two constraints only depend on design parameters, the motor speed limit also depends on the currently generated motor torque, and is given by:

$$\omega_{\mathrm{m,max}} = \frac{1}{K_\tau}(V_{\mathrm{max}} - R\,\frac{\tau_{\mathrm{m}}}{K_\tau}). \tag{20}$$

Recognising that $\frac{R}{K_\tau^2} = K_{\mathrm{m}}^{-2}$ and using the stall torque $\tau_{\mathrm{stall}} = V_{\mathrm{max}}\,K_{\mathrm{m}}^2/K_\tau$ yields

$$\omega_{\mathrm{m,max}} = (\tau_{\mathrm{stall}} - \tau_{\mathrm{m}})\frac{1}{K_{\mathrm{m}}^2}, \tag{21}$$

which only depends on K_{m} and τ_{stall} as parameters, which correlate well with motor mass. Recalling that $\omega_{\mathrm{m}} = n\,\dot{q}$, (21) defines the joint speed constraint (13) as a function of instantaneous joint torque and joint design parameters.

Proceeding with the friction model defined in (7), it is defined as

$$\tau_{\mathrm{f}}(\omega_{\mathrm{m}}) = -d_{\mathrm{v}}\,\omega_{\mathrm{m}} - d_{\mathrm{c}}\,\mathrm{sign}(\omega_{\mathrm{m}}), \tag{22}$$

where $d_{\mathrm{v}}, d_{\mathrm{c}}$ denote the viscous and Coulomb friction parameters as a function of the gearbox ratio n, using regression on measurement data of the different gearboxes.

Summarising, the vector of design parameters $\boldsymbol{\gamma}$ is given by sets of $\{m_{\mathrm{motor}}, n\}$ for each joint. Through regression, the former defines the motor constant K_{m}, rotor inertia I, maximum power P_{max}, and stall torque τ_{stall}, and the latter defines the joint friction parameters d_{v} and d_{c}. In addition, m_{motor} and n are directly used in (8) to define $D(\boldsymbol{q})$, I, and N. Note that currently we use a nominal fixed weight for the gearbox mass. This is accurate only within the same number of reduction stages.

C. Component selection

This Section concerns the final step in practical use of co-design tools; using their results to guide robot design. This requires selecting (typically from catalogue data) discrete actuation components based on the optimised continuously variable design parameters. Fig. 4 shows this conceptually: for each joint, the found optimum may not intersect with available components. We consider the next (up and down) available components, leading to 4 motor-gear combinations per joint. Not all may be feasible, limited by e.g. achievable output torque, motor power, and allowable gearbox torque, with respect to task requirements. Feasible combinations (up to 4^N) are re-optimised with their specific parameters, and the best components chosen. We believe that this is a more efficient approach than large-scale two-layer approaches using genetic algorithms such as [2], as long as parametrisation is sufficiently accurate[4].

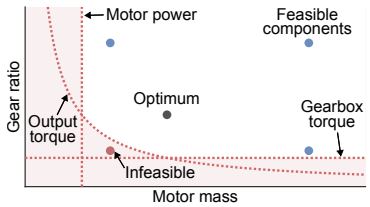


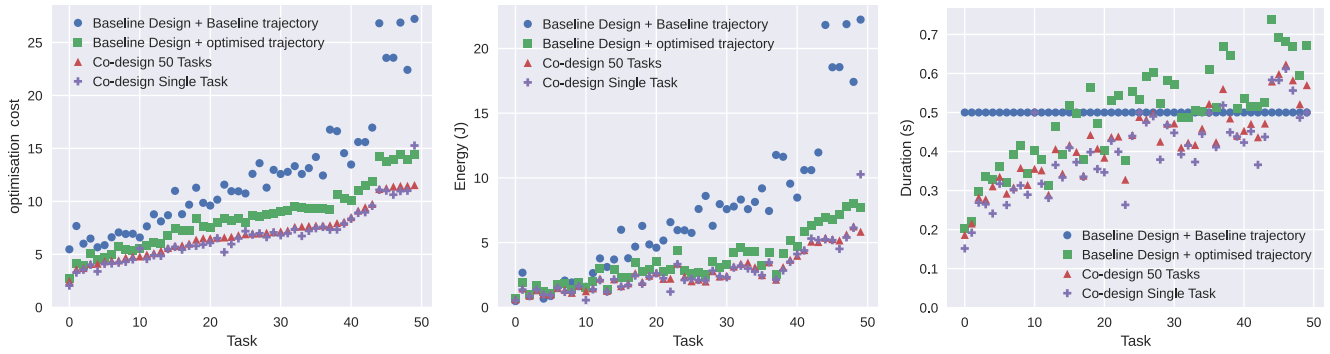
Fig. 4: Selection of discrete actuation components.

IV. Results

This Section presents extensive results and analyses of our co-design approach, specifically the following:

- **Baseline vs (multi-task) co-design:** We compare the co-design procedure to baseline and trajectory optimisation (TO) cases, over a set of 50 tasks. Notably, we

[4]Note that this typically limits the approach to e.g. a single motor series.



(a) Optimisation cost. **(b)** Energy consumed. **(c)** Task duration.

Fig. 5: Baseline vs. TO vs. (multi-task) co-design: Optimisation cost, energy, and task duration. Tasks are sorted by optimisation cost for co-design 50 tasks.

demonstrate that co-design over all tasks produces a robot design that on average is as fast and efficient as a robot optimised for each individual task;

- **Hyperparameters:** We investigate the influence of hyperparameters including duration cost, number of collocation points, and number of tasks;
- **Simulation:** We select catalogue motor and gearing components, and perform simulations of the robot with closed-loop control performing the optimised trajectories. We then compare with the co-design results.

We show these results for a 3-DoF robot arm, shown in Fig. 6a, that weighs 0.62 kg without actuation components. In all cases, we choose a 0.25 kg payload, a soft constraint weight of $w_{\text{pos}} = 10^6$, and a duration-cost of $w_{\text{t}} = 10$, unless otherwise specified. The video attachment also shows results produced on a 6-DoF robot arm (Fig. 6b).

A. Baseline vs. (multi-task) co-design

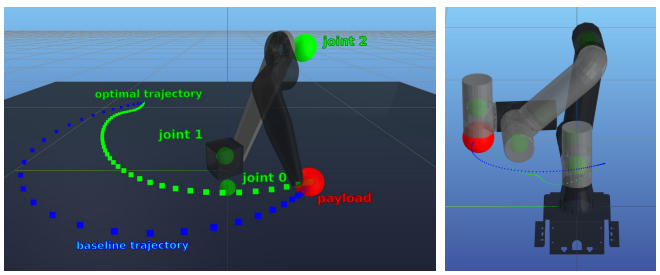
We begin our results by applying the proposed co-design procedure over a set of 50 tasks and comparing to baseline and trajectory optimisation cases. The tasks are generated by randomly sampling a start and end configuration from the robot's configuration space, with initial duration of $T = 0.5$ s (refer to Sec. II-F regarding initialisation). We re-stress that for optimal results the set of tasks should be representative of the tasks the robot will encounter in its application. As this robot has none, we sample randomly. The design parameters are initialised with $m_{\text{motor}} = 0.8$ kg and $n = 15$ for each actuator. We choose $N_{\text{p}} = 35$ collocation points. Now, to validate the co-design procedure, we compare 4 cases:

1) **Baseline design+trajectory:** Initial design parameters and initial (trapezoidal) trajectories as baseline.
2) **Baseline design + TO:** The trajectory of the baseline robot is optimised for each task.
3) **Co-design 50 tasks:** Design parameters and all 50 trajectories are simultaneously optimised.
4) **Co-design single task:** The robot is newly co-designed for each individual task.

Fig. 5 shows the optimisation cost, energy consumed, and task duration for each task in all four cases. Tasks are sorted by the optimisation cost of co-design 50 tasks. Table I summarises the results shown in Fig. 5 as averages.

The results show that in this case the most significant gains are made by optimising the trajectory with TO, which reduces the average energy consumption by more than 50% compared to the baseline, with similar average task duration. However, also optimising the robot's design by applying multi-task co-design over 50 tasks reduces average energy consumption by 23% and average task duration by 14% compared to optimising the trajectory alone.

Considering the optimised design parameters, listed in Table II, we observe that for the base (joint 0) and shoulder



(a) 3-DoF robot arm (yaw-pitch-pitch). **(b)** 6-DoF robot arm.

Fig. 6: 3-DoF and 6-DoF robot manipulators with a baseline trajectory (blue) and optimised trajectory (green). Green and red spheres show actuator and payload locations, respectively.

TABLE I: Average values for results in Fig. 5.

Trajectories	Energy (J)	Duration (s)	Cost
Baseline design+trajectory	7.41	0.50	12.41
Baseline design + TO	3.56	0.49	8.46
Co-design 50 tasks	2.75	0.42	6.94
Co-design single task	2.73	0.40	6.70

TABLE II: Co-design 50 tasks: Optimised design parameters.

Joint	Motor mass m_{motor} (kg)	Gear ratio n
Base	0.985	5.91
Shoulder	0.985	14.83
Elbow	0.139	27.55

(joint 1) the optimisation converges to large motors with relatively low gear ratios (close to or at what is typically considered quasi-direct-drive). Conversely, the elbow (joint 2) converges to a light weight motor with higher gear ratio, as its mass has a large inertial and gravitational contribution.

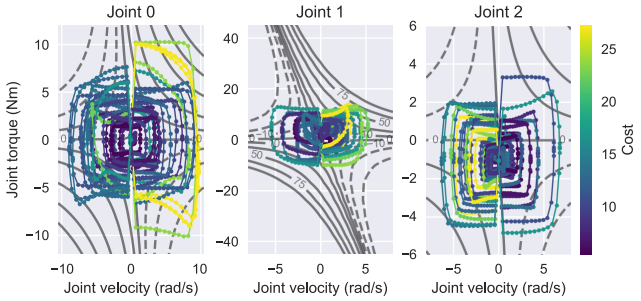
Finally, and critically, applying the co-design procedure to each individual task (in essence finding the optimal design for that task alone), only results in marginal further improvement. This demonstrates that the robot co-designed to be time- and energy-efficient at many different tasks is nearly as efficient as one co-designed for one specific task. This indicates some level of 'globally best' robot design using multi-task co-design, at least for this robot.

1) Torque-velocity profiles: To gain further insight, we analyse the torque-velocity profiles for the baseline design+trajectory and multi-task co-design cases. These are shown in Figs. 7a and 7b, respectively, for all three joints.

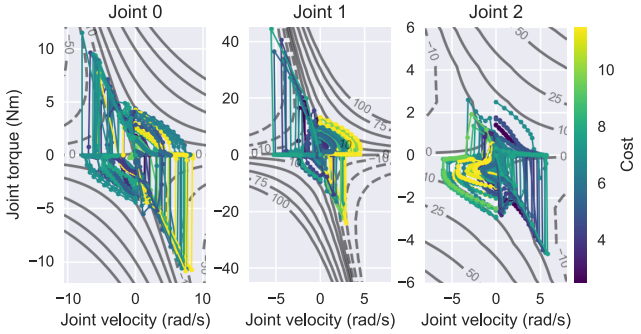
For the baseline case (Fig. 7a) the trapezoidal velocity profile is clearly visible as three segments, for acceleration, constant velocity, and deceleration, in a largely symmetrical profile. Notably, positive and negative electrical power are comparable in most cases.

Now considering the multi-task co-design case (Fig. 7b), the resulting profiles are significantly different: all gradually reduce acceleration as velocity increases. Combined with the comparable maximum velocity, this results in significantly reduced (maximum) positive power (which contributes to optimisation cost). Notice that the power isolines are different between Figs. 7a and 7b due to the changed design parameters. Similarly, when braking, typically a level of torque is applied that keeps electrical power zero or negative, effectively leveraging friction and the motor's electrical resistance to not incur electrical energy cost.

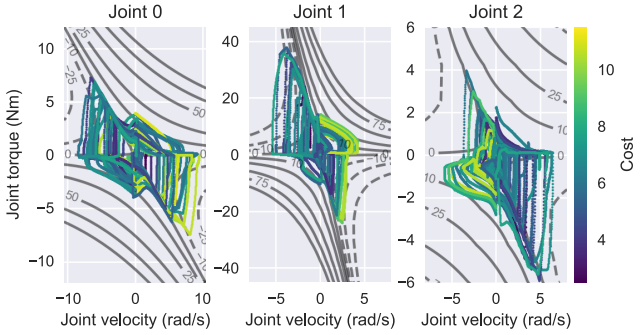
Finally, Fig. 7c shows the torque-velocity profiles for the same set of trajectories simulated with high temporal resolution and closed-loop control. These will be further discussed in Sec. IV-C.



(a) Baseline design + baseline trajectories.



(b) Multi-task co-design.



(c) Simulation of co-design results.

Fig. 7: Results for 50 tasks: Torque-velocity profiles for the baseline design+trajectories, multi-task co-design, and simulated trajectories. Colours indicate the optimisation cost as per (5). The isolines denote electrical power (19) at -50, -25, -10, 0, 10, 25, 50, 75, and 100 W, with negative values shown as dashed lines.

B. Hyperparameters

We investigate the effect of the hyperparameters involved in our co-design procedure. Specifically, we consider the influence of the duration cost w_t, which allows to bias the result towards a more time- or energy-efficient robot, and the number of tasks N_t. In Sec. IV-C we will also briefly analyse the influence of the number of collocation points N_p.

1) Task duration cost w_t: We perform the co-design procedure for logarithmically spaced values in the range $w_\mathrm{t} \in [10^{-1}, 10^{4}]$. We choose $N_\mathrm{t} = 5$ tasks due to the large number of co-design problems to be solved.

Fig. 8 shows the resulting average task duration and energy consumed[5]. As expected, as w_t increases, task duration decreases and energy consumed increases. Both reach a plateau, showing that (for large values) speed and/or torque limits are reached and that (for small values) a minimum amount of energy is required to overcome gravity and friction (and potentially increases in potential energy due to the task).

Fig. 9 shows the corresponding optimal design parameters for each value of w_t. As the value of w_t increases, initially the shoulder motor (joint 1) is made larger with lower gear ratio, which continues until the maximum motor mass is reached. Simultaneously the elbow (joint 2) sees a reduction in mass and a reduced gear ratio. The latter continues until at $w_\mathrm{t} = 10^{2.8}$ the torque limit is reached for one of the tasks. At that point the design moves to a different regime comprising of a larger motor with significantly lower gear ratio. This occurs without a significant jump in time and energy cost, indicating that the two regimes are indeed close in terms of optimality at that point. These results clearly demonstrate how w_t may be used to bias the robot design towards higher energy efficiency or time efficiency.

[5]The optimisation did not converge for $w_\mathrm{t} = 10^{0.8}$. For fair comparison we did not attempt to remedy this by changing e.g. the initialisation.

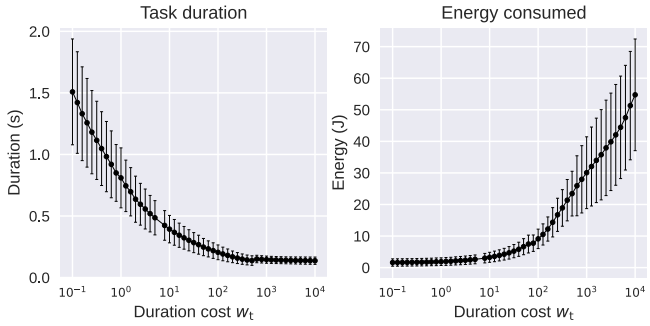


Fig. 8: Task duration and energy for different values of w_t.

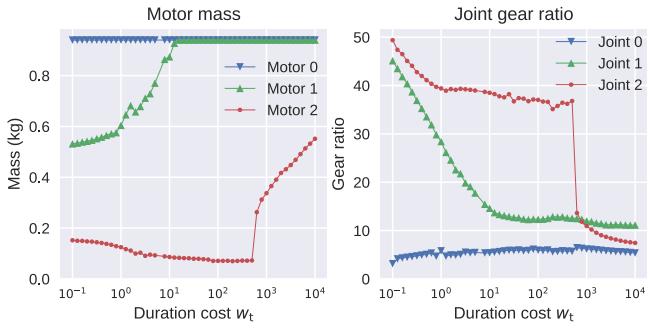


Fig. 9: Optimal design parameters as function of duration cost w_t.

2) Number of tasks N_t: Fig. 10 shows the optimal design parameters as the number of tasks is varied in $[1, 110]$. For a small number of tasks, each subsequent additional task has a large influence on the optimal design. Although most parameters are relatively stable $N_t \geq 10$, the gear ratio for the elbow sees relatively large changes until $N_t \geq 30$ tasks.

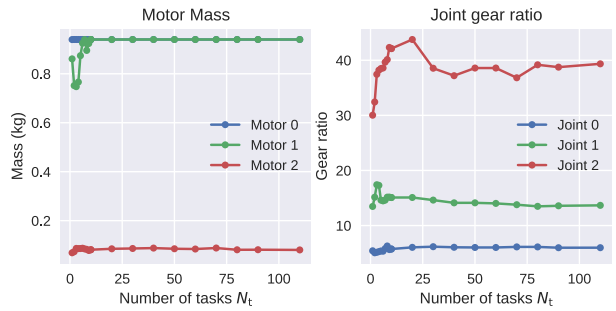


Fig. 10: Optimal design parameters over number of tasks N_t.

C. Simulation

The objectives of this Section are twofold. First, we consider the influence of component selection (Sec. III-C). Second, we show that the energy consumption results from co-design and TO are predictive of those obtained from simulation with closed-loop control and high temporal resolution. Based on the optimised design parameters (Table II), we select components listed in Table III. Using the catalogue parameters of these components, we then perform TO to find the optimal trajectory for each of the 50 tasks, both for $N_p = 50$ and $N_p = 200$ collocation points, and compare those to the multi-task co-design results obtained in Sec. IV-A. To then validate the TO and co-design results, we utilise simulation with high temporal resolution (10 kHz) and closed-loop control comprised of cascaded proportional position/velocity loops with references and torque feed-forward from the optimised results.

Fig. 11 shows the optimisation cost computed for all these cases, with circles denoting co-design and TO, and stars denoting the simulations of the corresponding robot and trajectories. First, comparing the co-design results (using parametric actuation models) to the TO results (using the selected catalogue components), we observe that the lower cost tasks tend to increase in cost, while the higher cost tasks tend to decrease, and the average difference is small. Second, comparing the TO results with $N_p = 50$ and $N_p = 200$, we observe a systematic difference, with $N_p = 50$ incurring lower cost. Indeed, the found optimal trajectories are slightly different. Third, we observe that in all three cases the co-design/TO result closely aligns with its simulation, indicating that their relative temporal coarseness does not adversely affect predictions of energy consumption for the found trajectory, even if the number of collocation points results in a different optimal trajectory.

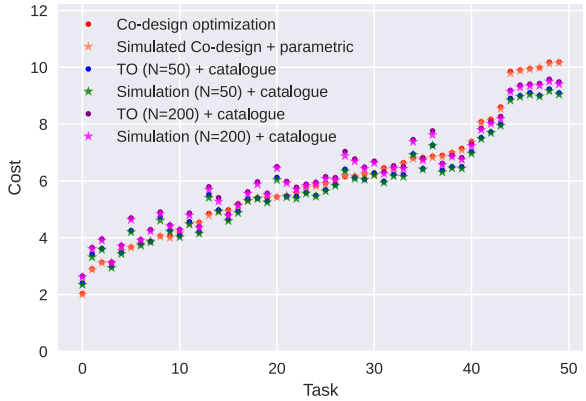


Fig. 11: Optimisation cost comparison between co-design results and TO+simulation results using catalogue components.

We return finally to Fig. 7c, which shows the torque-velocity profiles of simulation of the co-design results. Comparing to Fig. 7b shows that the general behaviour is indeed conserved, albeit with smoother variations of control torques due to the high temporal resolution.

V. Conclusions and Future Work

This work has presented a method for the co-design of robot actuation parameters. The procedure uses a parametric actuation model based on catalogue data and optimises the design over a large set of representative tasks. Closing the

TABLE III: Maxon components selected for simulation.

	Motor	Gear
Base	EC-flat 500267 (0.980 kg)	GP-52 223081 (4.3:1)
Shoulder	EC-flat 500267 (0.980 kg)	GP-52 223084 (15:1)
Elbow	EC-flat 651616 (0.150 kg)	GP-32 166937 (28:1)

loop towards practical co-design tools, we present a simple method for component selection using the co-design results.

The results demonstrate that such an approach is effective and, critically, leads to a robot that is capable of performing the entire set of tasks at an efficiency that is comparable to a robot co-designed for each specific task. Analyses of hyperparameters showed that the optimal design may be biased by the user towards either energy efficiency or speed.

Simulations using discrete actuation components from catalogue data showed that the co-design results are predictive of the energy consumption observed in simulation with closed-loop control and high temporal resolution.

We suggest directions for future work, some of which are ongoing. We are extending the modelling to account for stage-dependent effects of gearboxes, and more comprehensive modelling of torque and power limits (e.g., thermals). Furthermore, the critical step from co-design results to component selection/design is currently performed by re-optimising the closest available component combinations. While efficient, this may lead to suboptimal results in certain situations, and we suggest that further research is done for this step. Finally, we suggest more experimental evaluation of co-design approaches.

REFERENCES

[1] G. Fadini *et al.*, "Computational design of energy-efficient legged robots: Optimizing for size and actuators," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 30, 2021, pp. 9898–9904. DOI: 10.1109/ICRA48506.2021.9560988.

[2] G. Fadini *et al.*, "Co-designing versatile quadruped robots for dynamic and energy-efficient motions," *Robotica*, vol. 42, no. 6, pp. 2004–2025, Jun. 2024. DOI: 10.1017/S0263574724000730.

[3] M. Chadwick *et al.*, "Vitruvio: A leg design optimization toolbox for walking robots," p. 8, 2020.

[4] T. Dinev *et al.*, "A versatile co-design approach for dynamic legged robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 23, 2022, pp. 10 343–10 349. DOI: 10.1109/IROS47612.2022.9981378.

[5] G. Bravo-Palacios *et al.*, "One robot for many tasks: Versatile co-design through stochastic programming," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1680–1687, Apr. 2020. DOI: 10.1109/LRA.2020.2969948.

[6] G. Bravo-Palacios *et al.*, "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. DOI: 10.1115/1.4052463.

[7] Y. Yesilevskiy *et al.*, "Energy-optimal hopping in parallel and series elastic one-dimensional monopeds," *Journal of Mechanisms and Robotics*, vol. 10, no. 3, p. 031 008, Jun. 1, 2018. DOI: 10.1115/1.4039496.

[8] A. Spielberg *et al.*, "Functional co-optimization of articulated robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2017, pp. 5035–5042. DOI: 10.1109/ICRA.2017.7989587.

[9] G. Grandesso *et al.*, "Exploring the limits of a redundant actuation system through co-design," *IEEE Access*, vol. 9, pp. 56 802–56 811, 2021. DOI: 10.1109/ACCESS.2021.3072783.

[10] L. J. Maywald *et al.*, "Co-optimization of acrobot design and controller for increased certifiable stability," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 23, 2022, pp. 2636–2641. DOI: 10.1109/IROS47612.2022.9981825.

[11] R. Desai *et al.*, "Interactive co-design of form and function for legged robots using the adjoint method," *CLAWAR 2018*, Apr. 15, 2018. arXiv: 1801.00385.

[12] S. Ha *et al.*, "Computational co-optimization of design parameters and motion trajectories for robotic systems," *The International Journal of Robotics Research*, vol. 37, no. 13, pp. 1521–1536, Dec. 2018. DOI: 10.1177/0278364918771172.

[13] M. Geilinger *et al.*, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–12, Aug. 10, 2018. DOI: 10.1145/3197517.3201368.

[14] P. M. Wensing *et al.*, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2024. DOI: 10.1109/TRO.2023.3324580.

[15] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017. DOI: 10.1137/16M1062569. eprint: https://doi.org/10.1137/16M1062569.

[16] S. Moreno-Martín *et al.*, *Collocation methods for second and higher order systems*, 2023. DOI: 10.48550/ARXIV.2302.09056.

[17] L. Simpson *et al.*, "Direct collocation for numerical optimal control of second-order ODE," in *2023 European Control Conference (ECC)*, IEEE, Jun. 13, 2023, pp. 1–7. DOI: 10.23919/ECC57647.2023.10178181.

[18] H. Ferrolho *et al.*, "Inverse dynamics vs. forward dynamics in direct transcription formulations for trajectory optimization," 2020. DOI: 10.48550/ARXIV.2010.05359.

[19] S. Katayama and T. Ohtsuka, "Efficient solution method based on inverse dynamics for optimal control problems of rigid body systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 30, 2021, pp. 2070–2076. DOI: 10.1109/ICRA48506.2021.9561109.

[20] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2012, pp. 4914–4919. DOI: 10.1109/IROS.2012.6386181.

[21] J. Carpentier *et al.*, "The pinocchio c++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, Jan. 2019, pp. 614–619. DOI: 10.1109/SII.2019.8700380.

[22] J. A. E. Andersson *et al.*, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019. DOI: 10.1007/s12532-018-0139-4.

[23] N. Mansard, *Jnrh2023 - pinocchio 2.99*, https://github.com/nmansard/jnrh2023, Accessed: 2024-08-01.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006. DOI: 10.1007/s10107-004-0559-y.