Addressing Reachability and Discrete Component Selection in Robotic Manipulator Design through Kineto-Static Bi-Level Optimization

Enrico Mingo Hoffman, Daniel Costanzi, Gabriele Fadini, Narcis Miguel, Andrea Del Prete, Luca Marchionni

Addressing Reachability and Discrete Component Selection in Robotic Manipulator Design through Kineto-Static Bi-Level Optimization

E. Mingo Hoffman[1], D. Costanzi[2], G. Fadini[3], N. Miguel[2], A. Del Prete[4], and L. Marchionni[2]

***Abstract*— Designing robotic manipulators for generic tasks while meeting specific requirements is a complex, iterative process involving mechanical design, simulation, control, and testing. New computational design tools are needed to simplify and speed up such processes. This work presents an original formulation of the computational design problem, tailored to help design generic manipulators with strong reachability requirements. The primary challenges addressed in this work are twofold. First, the necessity to consider the design of both continuous quantities and discrete components. Second, the ability to guide the design using high-level requirements, like the robot's workspace, without needing a specific manipulation task, unlike other co-design frameworks. These two challenges are addressed by employing a novel kineto-static formulation, resulting in a Mixed Integer Nonlinear Programming problem, which is solved using bi-level optimization. A compelling use case from a real industrial application is presented to highlight the practical effectiveness of the proposed method.**

I. Introduction

Designing robots is a complex and iterative process, involving mechanical design, control synthesis, and experimental testing. Without the aid of computational design (co-design) tools, this process can become more laborious and may result in a sub-optimal design. The development of an industrial manipulator exemplifies these complexities. Requirements such as nominal payload, workspace coverage, and maximum control effort must be met while ensuring structural integrity and appropriate selection of actuators.

Initial design choices are often based on heuristics, influenced by the designer's experience with previous models. If these heuristics are based on inaccurate assumptions, they can trigger a domino effect of necessary changes, prolonging the design process through multiple iterations. On the other hand, co-design frameworks can accelerate the achievement of a functional design while optimizing specific user-defined metrics [1] [2]. However, modern co-design tools oriented towards robotics are often complex and task-specific. The metrics to be optimized must be specified through mathematical expressions to be minimized or maximized, and the resulting design may depend on the specific task used for the optimization, reducing its generality. Furthermore, the mechanical design may include components that can vary continuously within a range, such as link lengths, or components that must be selected from a discrete catalog, such as motor sizes. This aspect is often overlooked, requiring further iterations to adapt the output of the co-design tool to real components.

To tackle these challenges, this work introduces a novel method to streamline generic robotic manipulators' design, formulating the co-design problem as a kineto-static Mixed Integer Nonlinear Programming (MI-NLP) problem solved using bi-level optimization. Bi-level optimization addresses two interconnected problems, with one nested within the other, enabling different optimization strategies for the outer and inner loops. This formulation allows the inclusion of continuous and discrete design variables and the optimization of control effort, reachability, and manipulability, regardless of the specific manipulation task to be performed, thus being as generic as possible. This paper presents the formulation and the mathematical tools used to solve the MI-NLP problem and a real use case of a manipulator design, mounted on a mobile platform developed by the company PAL Robotics.

II. State of the Art

This work takes inspiration from and extends the bi-level optimization scheme to concurrently optimize hardware and control of a leg's design for jumping in [3]. In such a framework, the outer level optimization made use of a genetic algorithm (GA), the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [4], to optimize gait timings and design variables, such as link lengths and motor choice. The inner level optimization was instead in charge of optimizing the state and control trajectories through Differential Dynamic Programming (DDP) [5]. However, the hardware parameters in [3] were treated as continuous variables. As a result, the optimized results from CMA-ES are refined by selecting the closest available discrete options. The use of DDP in the inner loop makes it difficult to consider constraints on state variables, so this bi-level approach was further extended in [6] introducing interior-point optimization in the trajectory optimization and the possibility of studying several tasks with design discrete variables

Another notable tool in this domain includes the open-source Matlab toolbox Vitruvio [7]. Vitruvio guides designers in choosing the design parameters of the legs of walking robots; as optimization criteria, they made available minimum energy consumption and minimum peak control effort. The proposed approach is divided into three steps: trajectory generation for a baseline robot, followed by motion analysis, and finally, optimization of the design parameters

[1] Université de Lorraine, CNRS, Inria, LORIA, 615 Rue du Jardin-Botanique, Villers-lès-Nancy, France, enrico.mingo-hoffman@inria.fr

[2] PAL Robotics, Carrer de Pujades 77-79, Barcelona, Spain, {daniel.costanzi, narcis.miguel, luca.marchionni}@pal-robotics.com

[3] ETH Zürich, Computational Robotics Lab (CRL), Wasserwerkstrasse 12, Zürich, Switzerland gfadini@ethz.ch

[4] University of Trento, Via Sommarive 9, Povo, Italy andrea.delprete@unitn.it

through a GA, again CMA-ES. An issue of this approach, as highlighted in the letter, is that design and motion are not optimized concurrently.

Co-design has also been an important paradigm in the development of humanoid robots, where it is important to choose the hardware parameters such that the robot can effectively and safely interact with the environment and with humans. In [8], ergonomic indexes are used as fitness functions in a bi-level optimization scheme that shares a structure similar to ours: a GA for the outer loop, which generates populations of robot designs, then evaluated in terms of ergonomic indices through NLP optimization. They leveraged the GAs also to generate initial guesses for the inner optimization, to reduce the sensitivity to the problem's initial conditions.

In [9], a Python toolbox named *Timor* for industrial modular robotics is introduced, which allows the optimization of a manipulator's design according to the tasks to be performed by selecting existing modular components. From the same authors, in [10] a GA with a lexicographic evaluation of solution candidates is used for identifying an optimal module composition in task-tailored modular robots.

The work in [11] proposed a method for concurrently optimizing both the morphology parameters of a robot and its control policy for multi-task purposes. Optimizing the design for a broad range of tasks, rather than for a specific task, increases complexity but can yield robots capable of performing a wider array of tasks with superior performance compared to robots optimized for a single task. Their approach combines Stochastic Programming with Trajectory Optimization (TO) to tackle the scalability issues inherent in multi-task co-design problems.

In [12] the co-design problem is formulated for legged robots as a TO problem, augmented with the design variables of the model. To parameterize the robot model, they considered simple prismatic legs and derived analytical equations to compute their inertial properties. While our methodology may seem similar to that in [12], where hardware and control variables are optimized concurrently within a single NLP framework, our method implements on top of that a bi-level optimization scheme to address also the optimization of discrete actuator choices. This consideration is not supported in [12], as integer design variables were not included.

Recently, a task-driven computational framework that simultaneously optimizes the mounted pose and morphology of modular manipulators was presented in [13].

As seen in this section, multiple co-desing works relying on bi-level optimization, however, sufficient and necessary conditions for optimality are still an open question [14]. However, the literature highlights that GA-like optimization techniques are effective in locating the global optimum region due to their inherent parallelism. These methods do not require the objective function to be differentiable and are highly robust in addressing non-convex problems [15].

Most of the literature on computational design applied to robotics still frames the design problem around solving a specific task. While these approaches can achieve optimal results for specific applications, they often struggle to scale when the number of tasks to be performed increases. Only a few solutions have been proposed that can manage the complexity and variability introduced by multiple tasks. This gap highlights the need for more versatile methodologies that can address requirements to adapt to different use cases without compromising performance. In these regards, the main contribution of our work is a method that:

- enables the optimal design of manipulators without requiring the specification of particular manipulation tasks, relying on the definition of a *minimum fully reachable workspace*;
- accommodates both continuous and discrete design variables, allowing for the selection of components from a catalog.

This is achieved by employing a kineto-static MI-NLP formulation of the co-design problem, which is solved using bi-level optimization. The proposed approach integrates the benefits of concurrent optimization for continuous design and control variables while addressing the complexity of selecting discrete components and ensuring the generality of the obtained design regardless of the manipulation task to be performed.

III. Computational Design

This section presents the kineto-static formulation of the co-design problem and the method to solve the resulting MI-NLP problem. As previously stated, this work employs a similar bi-level optimization scheme as in [3], with the difference that the continuous design variables are optimized within the inner optimization loop using the interior point method, and the discrete design variables are optimized in the outer optimization loop, using genetic algorithms (see Fig. 1). [16] provides a comprehensive review of bi-level optimization methodologies, while [17] compares various outer-loop methods.

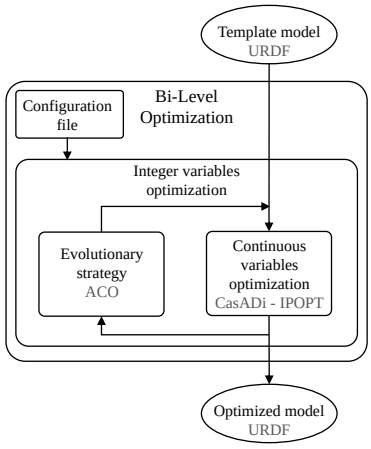


Fig. 1: Bi-level optimization scheme.

This work focuses on designing robotic arms with serial kinematic structures and a fixed number of Degrees of Freedom (DOFs) n and links $n+1$. The following assumptions are considered (see Fig. 2):

- the axis of rotation of each joint J_i is fixed;
- each actuator A_i is placed at the joint J_{i+1} and aligned with its rotation axis;
- each link L_i carries the mass and inertia of the consecutive actuator A_i, where the inertia is computed without considering any reduction.

The design parameters include the relative position of each joint, the joint limits, the link's mass, the center of mass (CoM), and the inertia tensor. These design parameters cannot be chosen freely; for example, the relative position between two joints affects the link lengths, CoM position, and mass. At the same time, the choice of actuators impacts these dynamic quantities, as each actuator contributes with its mass and inertia. Ultimately, the selection of actuators for each joint affects the velocity limits, the torque limits, and the efficiency.

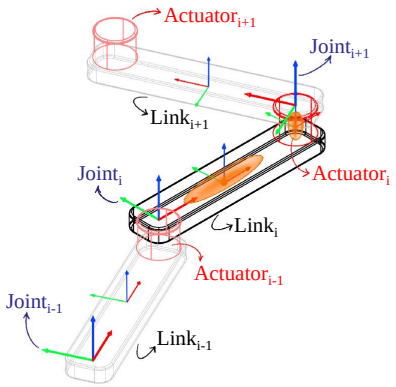


Fig. 2: Serial manipulator modeling with the frames of both the joints and links depicted. Additionally, for the $i-th$ link and its associated actuator body, an orange ellipsoid is illustrated to represent the body's inertia.

A. Model Parameterization and Design Variables

We denote the set of all design variables as $\mathcal{D} = \mathcal{D}_{\mathbb{R}} \cup \mathcal{D}_{\mathbb{Z}}$, where $\mathcal{D}_{\mathbb{R}}$ contains the continuous design variables and $\mathcal{D}_{\mathbb{Z}}$ contains the integer ones.

1) Continuous Design Variables $\mathcal{D}_{\mathbb{R}}$: The continuous design variables include:

- the pose ${}^b\mathbf{T}_{J_0} \in SE(3)$ of the first joint J_0;
- the scaling factor of links' lengths, represented by the vector $\boldsymbol{\lambda} \in \mathbb{R}^{n-1}$.

The link length scaling factor $\boldsymbol{\lambda}$ affects kinematic and dynamic properties and requires scaling inertial properties when modified. In this work, the factor scales the main link axis while maintaining the cross-section and density, with limits set to prevent weakened links.

The inertial properties of the i-th link can be fully described by 10 scalars [18]:

- the link's mass $m_i \in \mathbb{R}$;
- the position of the CoM ${}^i\mathbf{r}_i \in \mathbb{R}^3$ w.r.t. the joint frame associated to the link, with ${}^i\mathbf{T}_{A_i}$ the homogeneous transformation from the joint to the actuator frames and $\mathbf{r}_{A,i}$ the CoM of the actuator in actuator frame;
- the inertia matrix ${}^i\mathbf{I}_{\mathbf{r}_i} \in \mathbb{R}^{3\times3}$ at the CoM, expressed in the joint frame, defined by 6 scalars due to its symmetry, and its eigenvalues satisfying the triangle inequality.

It is important to note that, when scaling a link that carries an actuator, the latter should remain unchanged, and only the actual link should be scaled. For this reason, we express the previous quantities in terms of their link and actuator parts:

$$m_i = m_{L,i} + m_{A,i}, \tag{1}$$

$${}^i\mathbf{r}_i = \frac{m_{L,i}\mathbf{r}_{L,i} + m_{A,i}({}^i\mathbf{p}_{A_i} + {}^i\mathbf{R}_{A_i}\mathbf{r}_{A,i})}{m_i}, \tag{2}$$

$$\begin{aligned} {}^i\mathbf{I}_{\mathbf{r}_i} &= \mathbf{I}_{L,i} + {}^i\mathbf{R}_{A_i}\mathbf{I}_{A,i}{}^i\mathbf{R}_{A_i}^T + \\ &+ m_{A,i}\left[\mathbf{r}_{L,i} - ({}^i\mathbf{p}_{A_i} + {}^i\mathbf{R}_{A_i}\mathbf{r}_{A,i})\right]_\times^2. \end{aligned} \tag{3}$$

In (1), $m_{L,i} \in \mathbb{R}$ and $m_{A,i} \in \mathbb{R}$ are the mass of the i-th link and actuator, respectively. In (2), $\mathbf{r}_{L,i} \in \mathbb{R}^3$ is the CoM position of the i-th link, expressed in the i-th joint frame, $\mathbf{r}_{A,i} \in \mathbb{R}^3$ is the CoM position of the i-th actuator, expressed in the actuator frame, i.e. the $(i+1)$-th joint, ${}^i\mathbf{p}_{A_i} \in \mathbb{R}^3$ is the position of the i-th actuator w.r.t. the i-th joint, and ${}^i\mathbf{R}_{A_i} \in SO(3)$ is the rotation matrix from the i-th joint frame to the following one. Finally, in (3), $\mathbf{I}_{L,i} \in \mathbb{R}^{3\times3}$ and $\mathbf{I}_{A,i} \in \mathbb{R}^{3\times3}$ are the inertia tensor of the link and actuator, respectively, expressed at their CoM, and $[\cdot]_\times^2$ the square of the skew-symmetric matrix applied to the vector going from the CoM of the actuator to the CoM of the associated link.

Given m_i under the hypothesis of homogeneous density, its scaled version m_i' is:

$$m_i' = \lambda_i m_{L,i} + m_{A,i}. \tag{4}$$

Let us consider a scaling matrix acting in a specific direction:

$$\boldsymbol{\Lambda}_i = \begin{bmatrix} \lambda_i & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

We scale the body along a local joint axis defined by the versor $\hat{\mathbf{u}}_i$. The rotation matrix $\mathbf{R}_i$ is constructed with columns $\hat{\mathbf{u}}_i$ and a pair of unit vectors forming an orthogonal basis for $ker(\hat{\mathbf{u}}_i)$. The final scaling matrix is given by

$$\mathbf{S}_i = \mathbf{R}_i^T \boldsymbol{\Lambda}_i \mathbf{R}_i. \tag{6}$$

Scaling the link moves the position of the actuator w.r.t. the joint of a quantity:

$${}^i\mathbf{p}'_{A_i} = \mathbf{S}_i {}^i\mathbf{p}_{A_i}. \tag{7}$$

The scaled CoM can be computed as:

$${}^i\mathbf{r}'_i = \frac{m_{L,i}\mathbf{S}_i\mathbf{r}_{L,i} + m_{A,i}({}^i\mathbf{p}'_{A_i} + {}^i\mathbf{R}_{A_i}\mathbf{r}_{A,i})}{m_i'}. \tag{8}$$

Finally, the scaled inertia tensor can be computed as the scaled inertia of the link plus the inertia of the actuator adjusted for its new position:

$$\begin{aligned} {}^i\mathbf{I}'_{\mathbf{r}_i} &= \mathbf{I}'_{L,i} + {}^i\mathbf{R}_{A_i}\mathbf{I}_{A,i}{}^i\mathbf{R}_{A_i}^T + \\ &+ m_{A,i}\left[\mathbf{r}_{L,i} - ({}^i\mathbf{p}'_{A_i} + {}^i\mathbf{R}_{A_i}\mathbf{r}_{A,i})\right]_\times^2. \end{aligned} \tag{9}$$

As demonstrated in [19], an inertia matrix $\mathbf{I}$ (see Appendix I) can be derived from the *density-weighted covariance* matrix $\mathbf{J}$ using the following relationship:

$$\mathbf{I} = tr(\mathbf{J})\mathbb{I}_{3x3} - \mathbf{J}, \tag{10}$$

where $tr(\cdot)$ denotes the matrix trace operator, and $\mathbb{I}_{3\times3}$ is the 3×3 identity matrix. Being the density-weighted covariance scaling given by:

$$\mathbf{J}' = \lambda_i \mathbf{S}_i \mathbf{J} \mathbf{S}_i^T, \tag{11}$$

the scaled inertia $\mathbf{I}'_{L,i}$ can be written as:

$$\mathbf{I}'_{L,i} = tr(\mathbf{J}'_{L,i})\mathbb{I}_{3x3} - \mathbf{J}'_{L,i}. \tag{12}$$

2) Integer Design Variables $\mathcal{D}_\mathbb{Z}$: The integer design variables represent actuator choices and are optimized using the outer genetic loop. Specifically, we used the Ant Colony Optimization (ACO) algorithm [20], an effective strategy for mixed-integer search spaces. An explicit mapping between the actuator identifier $\mathcal{A}_{ID} \in \{0, 1, ..., a-1\}$, with a the number of possible actuator choices, and its specifics has been used to modify the robot model used in the inner loop.

When the ACO algorithm generates a new population, it creates a new set of chromosomes, representing the actuators chosen for each individual. Following this, the inner optimization loop calculates each individual's fitness using the robot model modified with the selected set of actuators. In the following, we describe the outer and inner optimization loops, respectively, the ACO algorithm and the NLP formulation.

B. ACO Algorithm

The selection of actuators in robotic manipulator design is as critical as their placement in the kinematic chain. It typically involves simulating tasks to evaluate required torques, often using nominal values for actuator mass and inertia. However, real actuators from vendor catalogs can differ significantly, requiring designers to balance project requirements and specifications. This often necessitates additional simulations to validate the selection, making optimal choices complex. To address this, we propose formulating an MI-NLP problem for actuator selection, structured as follows:

$$\begin{aligned} &\min_{\mathcal{D}_\mathbb{Z}} \mathcal{F}(\mathcal{D}_\mathbb{Z}) = \left\{ f(\mathcal{D}_{\mathbb{Z}_0}), f(\mathcal{D}_{\mathbb{Z}_1}), \ldots, f(\mathcal{D}_{\mathbb{Z}_{I-1}}) \right\} \\ &\text{s.t. } \mathcal{D}_{\mathbb{Z}_\text{k}} \in \mathcal{A}_\text{ID}, \end{aligned} \tag{13}$$

where I represents the number *individuals* of the *population* for each *generation*, and $\mathcal{D}_{\mathbb{Z}_k}$ is a set of discrete variables associated with the individual k. The objective is to minimize the fitness function $f(\mathcal{D}_\mathbb{Z})$, by selecting the optimal set of actuators from the catalog $\mathcal{A}_{ID}$. The ACO algorithm is an evolution strategy classified as a stochastic metaheuristic, that takes inspiration from ants and the way they find the shortest path when transporting resources to their colony. Real ants, while exploring their space, lay down pheromone trails that tend to dissolve as time passes, thus guiding ants to the path that takes less time to travel. ACO implements agents, simulated ants, keeping track of their positions, i.e. the decision variables, and the quality of their solution, an information for the agents of the later generations. Extended ACO (GACO) [21] generates future generations of ants by using a multi-kernel Gaussian distribution based on pheromone values, which are computed depending on the quality of each previous solution, ranked through an oracle penalty method. GACO can work on mixed integer search domains, in contrast to the original version which is thought to work only on combinatorial or continuous search domains. For this reason, the MI-NLP problem in (13) was solved using the GACO implementation in *PyGMO*, a Python scientific library derived from the PaGMO framework [22]. In nested methods, as the one proposed in this work to solve the bi-level optimization problem, the optimization in (13) selects the minimum residual of (14), see the following section, evaluated at all the *ants* of the current generation, and computes the successive generation.

C. NLP Formulation

When designing a robotic manipulator, a crucial issue is to define the relative placement of the joints so that the robot can reach all the points belonging to the operational space with its end-effector, a.k.a. reachable workspace. Moreover, the robot needs its actuators to be able to withstand the torques generated by its weight and the payload. Another important aspect that should be taken into account is the manipulability of the robot within the desired workspace. Having low isotropy of the manipulability, in a specific configuration, reflects a poor capability of producing speed in the operational space. These indices are usually considered with trade-offs based on specific tasks when designing manipulators [23]–[25]. Given a desired minimum fully reachable workspace, the kineto-static approach aims at optimizing the design of the robot such that:

- the robot can reach all the points with its end-effector;
- the robot can compensate for gravity by satisfying the torque limits of the actuators;
- the static control effort is minimized;
- the manipulability is maximized.

The minimum fully reachable workspace is defined as a Cartesian region, which is sampled with N elements that depend on the spacing of a uniform sampling grid. Once a set of integer variables (*individual*) $\mathcal{D}_{\mathbb{Z}_k}$ is selected by the outer loop, the remaining decision variables for the inner loop are $\mathbf{q}$ and $\mathcal{D}_\mathbb{R}$, resulting in a continuous NLP problem, formulated as:

$$\begin{aligned} f(\mathcal{D}_{\mathbb{Z}_k}) = \min_{\mathbf{q}, \mathcal{D}_\mathbb{R}} & \sum_{i=0}^{N-1} \ell_i(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_\mathbb{R}) \\ \text{s.t. } & \boldsymbol{\tau}_\text{min} \leq \boldsymbol{\tau}(\mathbf{q}_\text{i}, \mathcal{D}_{\mathbb{Z}_\text{k}}, \mathcal{D}_\mathbb{R}) \leq \boldsymbol{\tau}_\text{max} \\ & G(\mathbf{q}_i, \mathcal{D}_\mathbb{R}) \leq \mathbf{0} \quad \forall i \in 0, \ldots, N-1, \end{aligned} \tag{14}$$

with $\mathbf{q}_i \in \mathbb{R}^n$ the generalized position coordinates of the manipulator at the i-th sample, and $\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max} \in \mathbb{R}^n$ the minimum and maximum joint torques, respectively.

The cost function $\ell_i(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})$ is defined as:

$$\begin{aligned}\ell_i(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) = w_p \left\|\mathbf{p}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) - \mathbf{p}_i\right\|_2^2 + \\ + w_\tau \left\|\boldsymbol{\tau}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})\right\|_2^2 + \\ + w_{\mu,lin}\mu_{lin}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) + \\ + w_{\mu,ang}\mu_{ang}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}),\end{aligned} \tag{15}$$

with $\mathbf{p}_i \in \mathbb{R}^3$ the i-th position of the sampled element of the workspace, $\boldsymbol{\tau}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})$ the static torques:

$$\boldsymbol{\tau}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) = \mathbf{g}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) + \mathbf{J}^T(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})\mathbf{h}_e, \tag{16}$$

with $\mathbf{h}_e \in \mathbb{R}^3$ the estimated force due to a load attached to the end-effector, and $\mu_{lin}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})$ and $\mu_{ang}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})$ the manipulability cost, for the linear and angular part, respectively. To avoid the computation of symbolic eigenvalues of the manipulability:

$$\mathbf{A}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) = \left(\mathbf{J}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})\mathbf{J}^T(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})\right)^{-1}, \tag{17}$$

we introduce an alternative approach to evaluate the manipulability isotropy. First of all, given the following decomposition of the manipulability $\mathbf{A}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})$[1]:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{lin} & \mathbf{A}_{mix} \\ \mathbf{A}_{mix} & \mathbf{A}_{ang} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{18}$$

we consider the linear and angular terms separately, neglecting the mixed ones[2]. We want to evaluate the manipulability along a certain number of d directions. As shown in Fig. 3, the manipulability along a generic direction $\mathbf{u}_j$, can be evaluated as the inverse of the square root of α_j:

$$\alpha_j(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) = \mathbf{u}_j^T \mathbf{A}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}})\mathbf{u}_j. \tag{19}$$

If all the α_j have similar magnitude, the manipulability is approximately isotropic. Therefore, by minimizing the sample variance of the set containing all the α_j, we can encourage isotropy of the manipulability:

$$\begin{aligned}\bar{\alpha}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) = \frac{1}{d}\sum_{j=0}^{d-1} \alpha_j(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}), \\ \mu(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) = \sum_{j=0}^{d-1} (\alpha_j(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}) - \bar{\alpha}(\mathbf{q}_i, \mathcal{D}_{\mathbb{Z}_k}, \mathcal{D}_{\mathbb{R}}))^2,\end{aligned} \tag{20}$$

with $\bar{\alpha}$ the mean. The smaller it is μ, the more isotropic the robot performance in a given configuration. Finally, $G(\mathbf{q}_i, \mathcal{D}_{\mathbb{R}}) \leq \mathbf{0}$ contains other bounds and constraints such as joint limits, design variables limits, and arm length limit.

IV. Case Study

PAL Robotics has recently developed a new 7-DOF arm based on Series Elastic Actuators (SEAs) mounted on the torso of the TIAGo robot [26]. For this project, three distinct sizes of actuators were available: S+, S-, and XS. Each

[1]We here drop the dependency on the variables $\mathbf{q}_i$, $\mathcal{D}_{\mathbb{Z}_k}$, and $\mathcal{D}_{\mathbb{R}}$

[2]It is worth noting that the inverse of a 3×3 matrix can be expressed in closed form

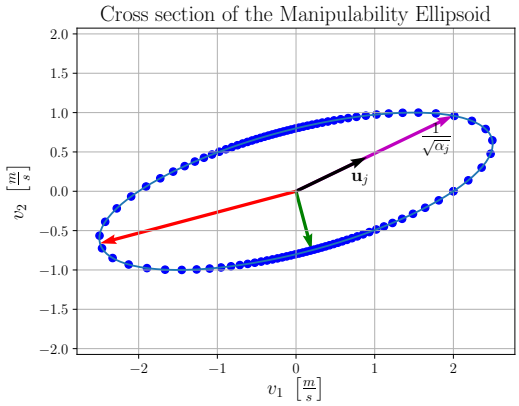


Fig. 3: Plot of the cross-section of a 3D manipulability ellipsoid.

actuator has unique specifications, particularly in terms of their maximum nominal torque, peak torque, and inertial properties, reported in Table I. In Fig. 4 and Table III are reported the initial design of the manipulator with its original mounting on the Tiago base, together with the design variables $\mathcal{D}_{\mathbb{R}}$ and the configuration variables $\mathbf{q}$. In this

TABLE I: Actuators specifications

Size	Nominal Torque [Nm]	Peak Torque [Nm]	Mass [kg]
S+	45	75	0.71
S-	35	60	0.60
XS	25	40	0.45

section, the proposed method has been used to improve the initial design of the arm and its mounting on the TIAGo base. In particular, the NLP problem in (14) has been formulated using CasADi [27] and solved using the IPOPT solver [28], and the model was computed using the Pinocchio [29] library. The main requirement for this robot was the ability

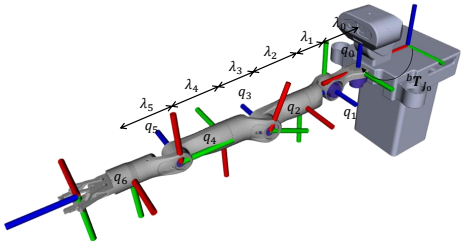


Fig. 4: Kinematic arrangement of the manipulator on the Tiago base together with the design variables $\boldsymbol{\lambda}$ and ${}^b\mathbf{T}_{J_0}$, and the configuration variables $\mathbf{q}$.

to reach all the points belonging to a given workspace, denoted as WS_0, with its end-effector. Moreover, it had to be capable of carrying a payload of up to 2 kg, while respecting the torque limits imposed by the size of the actuators. The manipulability needed to be maximized within WS_0, with the origin of the first joint positioned on the sagittal plane. Additionally, the maximum length of the arm was constrained to be within ± 0.2 m of the original design. Being all the joint limits of the robot symmetric, we can expect that

two points symmetrically placed w.r.t. the sagittal plane will have identical, but mirrored, optimal configurations to reach them. This symmetry is achievable by constraining the Roll and Yaw mounting angles, as well as the X mounting positions to zero of the first DOF. We enforce this through bounds on the corresponding optimization variables, see Table III. By leveraging this symmetry, we can reduce the size of the desired reachable workspace considered during optimization without affecting the actual reachable workspace. This effectively halves the problem size and enables a higher sampling density within the symmetrical workspace. The main aspects considered to assess the improvement were the reachability, the static control effort, and the manipulability, which were evaluated in the preliminary design of the robot and the optimized one. Those metrics were weighted and included in the NLP cost function. Adjusting the weights impacts the final design, and tuning is left to the designer based on the strictness of specific requirements. It is crucial then to carefully balance these parameters to achieve performance in accordance with the prioritized needs. In this case, since the reachability and versatility of the design are the highest priority, the following set of weights are used: $w_p = 10^4$, $w_\tau = 10^{-5}$, $w_{\mu_{lin}} = 0.0$, and $w_{\mu_{ang}} = 10.0$. Considering the system utilized for running the optimization[3], a suitable compromise regarding the number of optimization poses was determined to be 150. Given the volume of WS_0^{rhs} being approximately 0.9 m^3, the sampling density was accordingly set to 167 $\frac{\text{samples}}{m^3}$.

The GACO algorithm was configured to evolve a population of 40 individuals across 10 generations, running for a total of 3 hours. As illustrated in Fig. 5, the observed cost trend demonstrates that the cost plateaued after the initial 5 generations. Concerning the actuators, the initial and

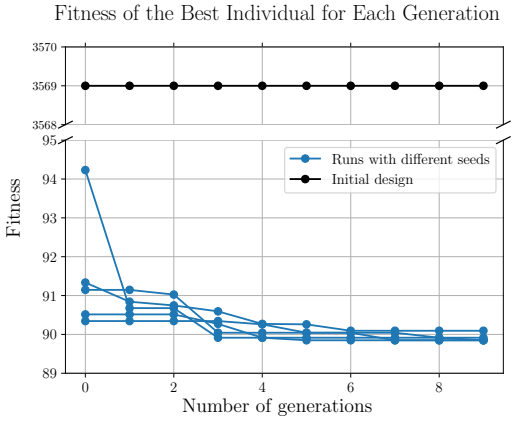


Fig. 5: Fitness trend of the best element per generation across runs with different seeds, compared to the initial design.

optimized choices are reported in Table II, with the numbers reflecting their physical arrangement relative to the robot's torso, starting from the shoulder joint and moving towards the end-effector. The actuators selected by the algorithm show that bigger actuators have been assigned to the joints

[3]Intel® Core™ i7-1065G7 CPU @ 1.30GHz × 8, RAM 8GiB

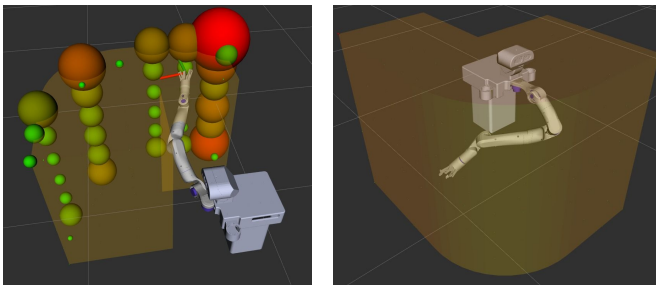
TABLE II: Actuator choice (discrete design params)

Joint	0	1	2	3	4	5	6
Initial Design	S+	S+	S+	S-	S-	S-	XS
Optimized Design	**XS**	S+	S+	S-	**XS**	**XS**	XS

TABLE III: Optimized continuous design params

Design Param	Opt. (Orig.) Value	Lower Lim.	Upper Lim.
Roll [rad]	0.0 (0.0)	0.0	0.0
Pitch [rad]	-0.13 (0.0)	-0.5	0.5
Yaw [rad]	0.0 (0.0)	0.0	0.0
X [m]	-0.02 (0.0)	-0.2	0.2
Y [m]	0.0 (0.0)	0.0	0.0
Z [m]	0.0 (0.0)	0.0	0.0
λ_0	1.5 (1.0)	0.7	1.5
λ_1	0.7 (1.0)	0.7	1.5
λ_2	0.85 (1.0)	0.7	1.5
λ_3	1.3 (1.0)	0.7	1.5
λ_4	1.5 (1.0)	0.7	1.5
λ_5	1.5 (1.0)	0.7	1.5

closer to the robot base, having to counteract more torque to overcome gravity and payload. On the other hand, lighter and less powerful actuators have been selected for the joints close to the robot's wrist. The solution chosen by the optimization offers a dual benefit: it balances the increased weight from the arm extension by using lighter actuators, and also helps reduce the overall manufacturing cost of the arm. Notice that for the first joint, which has its axis almost perpendicular to gravity, the smallest actuator (XS) has been selected. Regarding the continuous design variables, i.e., link length scaling and first joint mounting pose, the optimized results are reported in Table III. In Fig. 6 are visualized the position errors before and after optimization as spheres, directly on WS_0^{rhs}. The initial design could not reach every point belonging to WS_0^{rhs}, as shown in Fig. 6.a. The average reachability error, computed as the



a) b)

Fig. 6: Distribution of position errors for the initial design, on the left, and the optimized design, on the right within the desired reachable workspace. The yellow spheres represent the error in reaching points within the workspace. The radius and color of each sphere correspond to the Euclidean norm of the error (red means larger, and green means smaller).

Euclidean norm of the vector connecting a specific point in the workspace and the end-effector, was 23.6 mm. Fig. 6.b. shows that the robot's capability to reach all the points of WS_0^{rhs} has increased considerably. Finally, Fig. 7 illustrates quantitatively the increase in the number of reachable points, which has significantly improved compared to the non-optimized case. Notably, for the optimized design, no points are associated with a reachability error in the $5-500$ mm range, and most points fall within the $0-1$ mm range. For the

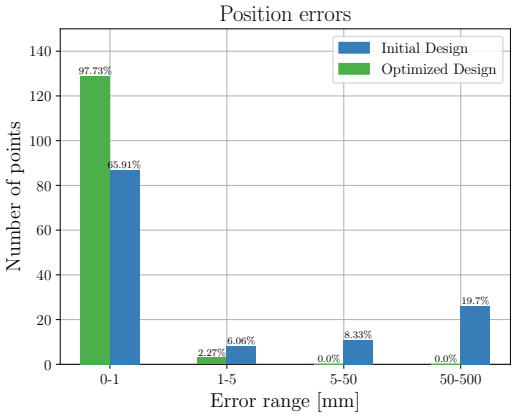


Fig. 7: Comparison of the number of points within a given error range for original and optimized design.

average angular manipulability isotropy, the initial design has an isotropy index of 1.66, while the optimized design reduces it to 1.4. It is worth noting that, with the optimal isotropy value being 1, the optimization achieves a 16% improvement.

Finally, in terms of static control effort, Fig. 8 reports the maximum and mean torque of each joint computed in all configurations explored to reach the points in the workspace WS_0^{rhs} while carrying the 2 kg payload, for both the initial and optimized design. It is possible to observe that the average maximum static torque is near or below the nominal actuator torque and way below the peak torque for each actuator. Although the kineto-static formulation does not allow for the optimization of design parameters according to dynamic motions, i.e. the inertia matrix scaling is only computed as a result and is not considered during optimization, Fig. 8 shows that the actuator torque available for dynamic motions is largely sufficient. To facilitate a comparison between the data in Fig. 8, the norm of the average torque across all joints was computed for both the original and optimized designs. The results were 9.86 Nm for the original design and 11. Nm for the optimized design, resulting in 11.6% increase in average static control effort. Although, at first glance, it may seem that the optimization process did not succeed in reducing the control effort, the result is logical considering its trade-off with manipulability. Moreover, as shown in Fig. 9, the increase in torque is explainable by an overall arm length increase of 20%, from 1.07 m to 1.26 m, and by a total weight increase of 10%, from ≈ 10 kg to ≈ 11 kg.

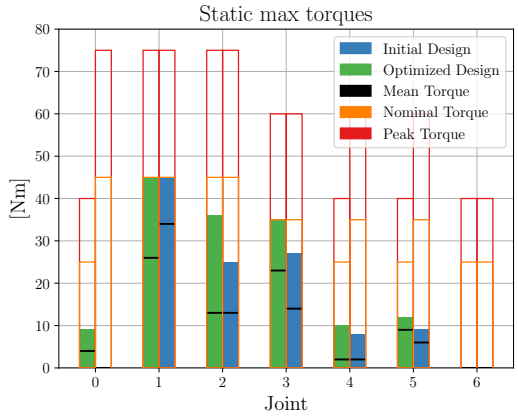


Fig. 8: Maximum static torques for the initial (blue) and optimized (green) designs computed considering all the configurations, in black the mean torques. The nominal and peak torques for the selected actuator are represented in orange and red, respectively. Notice that the last joint, the roll, is not loaded.

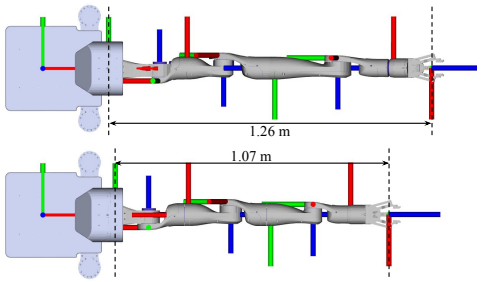


Fig. 9: Comparison between original, on the bottom, and optimized, on the top, Tiago manipulator link lengths.

V. Conclusions

This work presented a novel kineto-static formulation for designing manipulators, defining the co-design problem in terms of reachability region, meaning the area that the robot's end-effector is expected to cover, abstracting the specific task to be performed. The proposed method involves solving a constrained inverse statics problem for multiple positions simultaneously, while maximizing manipulability and minimizing effort, using instances of the robot model that share the same design variables. These can include both continuous and discrete design variables leading to a MI-NLP solved using bi-level optimization. The method's effectiveness has been evaluated by co-designing a novel manipulator mounted on the TIAGo mobile platform developed by PAL Robotics. The proposed method can be extended to parallel robots, closed linkages, and floating-base systems by incorporating design variables, kinematic constraints, and constraint forces into the formulation, ensuring closure and handling specific cases such as non-unique forces for certain contact types. Future work will explore the use of different outer-loop optimization algorithms, such as Particle Swarm Optimization or Genetic Algorithms, the use of parallel computing strategies to enable concurrent fitness evaluations, and the inclusion self-collision in the optimization problem. In particular, leveraging parallel computing strategies could

address a major limitation of the proposed approach: its current inability to optimize the design accounting for dynamic motions.

REFERENCES

[1] S. B. Liu and M. Althoff, "Optimizing performance in automation through modular robots," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 4044–4050.

[2] E. M. Hoffman, A. Laurenzi, F. Ruscelli, L. Rossini, L. Baccelliere, D. Antonucci, A. Margan, P. Guria, M. Migliorini, S. Cordasco, G. Raiola, L. Muratore, J. E. Rodrigo, A. Rusconi, G. Sangiovanni, and N. G. Tsagarakis, "Design and validation of a multi-arm relocatable manipulator for space applications," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 11 887–11 893.

[3] 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 International Conference on Robotics and Automation*, 2021, pp. 9898–9904.

[4] N. Hansen, "The CMA evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[5] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[6] G. Fadini, S. Kumar, R. Kumar, T. Flayols, A. Del Prete, J. Carpentier, and P. Souères, "Co-designing versatile quadruped robots for dynamic and energy-efficient motions," *Robotica*, vol. 42, no. 6, p. 2004–2025, 2024.

[7] M. Chadwick, H. Kolvenbach, F. Dubois, H. F. Lau, and M. Hutter, "Vitruvio: An open-source leg design optimization toolbox for walking robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6318–6325, 2020.

[8] C. Sartore, L. Rapetti, F. Bergonti, S. Dafarra, S. Traversaro, and D. Pucci, "Codesign of humanoid robots for ergonomic collaboration with multiple humans via genetic algorithms and nonlinear optimization," in *IEEE-RAS International Conference on Humanoid Robots*, 2023, pp. 1–8.

[9] J. Külz, M. Mayer, and M. Althoff, "Timor python: A toolbox for industrial modular robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023, pp. 424–431.

[10] J. Külz and M. Althoff, "Optimizing modular robot composition: A lexicographic genetic algorithm approach," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 16 752–16 758.

[11] G. B.-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. 1680–1687, 2020.

[12] A. E. Spielberg, B. Araki, C. R. Sung, R. Tedrake, and D. Rus, "Functional co-optimization of articulated robots," *IEEE International Conference on Robotics and Automation*, pp. 5035–5042, 2017.

[13] M. Lei, E. Romiti, A. Laurenzi, and N. G. Tsagarakis, "Task-driven computational framework for simultaneously optimizing design and mounted pose of modular reconfigurable manipulators," *arXiv preprint arXiv:2405.01923*, 2024.

[14] P. Mehlitz and A. B. Zemkoho, "Sufficient optimality conditions in bilevel programming," *Mathematics of operations research*, vol. 46, no. 4, pp. 1573–1598, 2021.

[15] V. Oduguwa and R. Roy, "Bi-level optimisation using genetic algorithm," in *IEEE International Conference on Artificial Intelligence Systems*, 2002, pp. 322–327.

[16] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2018.

[17] C. Sartore, M. Rando, G. Romuladi, C. Molinari, L. Rosasco, and D. Pucci, "Automatic gain tuning for humanoid robots walking architectures using gradient-free optimization techniques," in *IEEE International Conference on Humanoid Robotics*, 2024.

[18] L. Sciavicco, B. Siciliano, and L. Villani, "Lagrange and newton-euler dynamic modeling of a gear-driven robot manipulator with inclusion of motor inertia effects," *Advanced Robotics*, vol. 10, no. 3, pp. 317–334, 1995.

[19] P. M. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 60–67, 2018.

[20] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[21] M. Schlüter, J. A. Egea, and J. R. Banga, "Extended ant colony optimization for non-convex mixed integer nonlinear programming," *Computers & Operations Research*, vol. 36, no. 7, pp. 2217–2229, 2009.

[22] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," *Journal of Open Source Software*, vol. 5, no. 53, p. 2338, 2020.

[23] S. Kucuk and Z. Bingul, "Robot workspace optimization based on a novel local and global performance indices," in *IEEE International Symposium on Industrial Electronics*, vol. 4, 2005, pp. 1593–1598.

[24] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3229–3236.

[25] K. Leibrandt, L. Da Cruz, and C. Bergeles, "Designing robots for reachability and dexterity: Continuum surgical robots as a pretext application," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2989–3007, 2023.

[26] J. Pagès, L. Marchionni, and F. Ferro, "Tiago: the modular robot that adapts to different research needs," in *Int. Workshop on Robot Modularity, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[27] J. A E Andersson, J. Gillis, G. Horn, J. B Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[28] 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, pp. 25–57, 2006.

[29] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *International Symposium on System Integrations*, 2019.

APPENDIX I
DENSITY-WEIGHTED COVARIANCE MATRIX

The density-weighted covariance matrix $\mathbf{J}$ is defined as:

$$\mathbf{J} = \rho \iiint_{\mathcal{V}} \mathbf{r}\mathbf{r}^T d\mathbf{r}, \tag{21}$$

over the volume $\mathcal{V}$. The diagonal elements of the inertia matrix $\mathbf{I}$, are related to the diagonal elements of the density-weighted covariance matrix $\mathbf{J}$ through the following equations:

$$\begin{bmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} J_{xx} \\ J_{yy} \\ J_{zz} \end{bmatrix} \tag{22}$$

By inverting this system of equations, the diagonal elements of $\mathbf{J}$ can be computed from $\mathbf{I}$. Notice that according to the inverse of the system in (22) and equation (10), the relation between $\mathbf{J}$ and $\mathbf{I}$ can be written also as:

$$\mathbf{J} = \frac{1}{2} tr(\mathbf{I}) \mathbb{I}_{3\times 3} - \mathbf{I}. \tag{23}$$

Given the definition (21), we can compute the scaled density-weighted covariance matrix:

$$\mathbf{J}' = \rho \iiint_{\mathcal{V}} \mathbf{r}'\mathbf{r}'^T d\mathbf{r}' = \rho \iiint_{\mathcal{V}} (\mathbf{S}_x\mathbf{r})(\mathbf{S}_x\mathbf{r})^T s\, dx\, dy\, dz =$$
$$= \rho \iiint_{\mathcal{V}} \mathbf{S}_x(\mathbf{r}\mathbf{r}^T)\mathbf{S}_x^T s\, dx\, dy\, dz = s\mathbf{S}_x\mathbf{J}\mathbf{S}_x^T. \tag{24}$$