Redundancy Resolution at Position Level

Alin Albu-Schäffer, *Fellow, IEEE*, and Arne Sachtler, *Member, IEEE*

Abstract—Increasing the degrees of freedom of robotic systems makes them more versatile and flexible. This usually renders the system kinematically redundant: the main manipulation or interaction task does not fully determine its joint maneuvers. Additional constraints or objectives are required to solve the under-determined control and planning problems. The state-of-the-art approaches arrange tasks in a hierarchy and decouple lower from higher priority tasks on velocity or torque level using projectors. We develop an approach to redundancy resolution and decoupling on position level by determining subspaces of the configurations space independent of the primary task. We call them *orthogonal foliations* because they are, in a certain sense, orthogonal to the task self-motion manifolds. The approach provides a better insight into the topological properties of robot kinematics and control problems, allowing a global view. A condition for the existence of orthogonal foliations is derived. If the condition is not satisfied, we will still find approximate solutions by numerical optimization. Coordinates can be defined on these orthogonal foliations and can be used as additional task variables for control. We show in simulations that we can control the system without the need for projectors using these coordinates, and we validate the approach experimentally on a 7-DoF robot.

I. INTRODUCTION

ROBOTS are called *redundant* if the number of degrees of freedom of the mechanical structure exceeds the degrees of freedom of the task.

Kinematics and dynamics of redundant robots belong to the classical and most well-studied theoretical foundations of robotics [1]–[9]. The state of the art for redundancy resolution is based on projection operators and arranges multiple tasks in a hierarchy of priorities. It is summarized below.

A. Projection-based Redundancy Resolution in a Nutshell

Projection-based redundancy resolution is one of the basic approaches presented in the standard literature [10]–[14]. The main idea is to add additional tasks that use up the remaining degrees of freedom of the mechanical structure. Then, the projection operators ensure that lower-priority tasks are decoupled from higher-priority tasks. We denote the forward kinematics mapping for two task priorities between the configuration (joint) space $\mathcal{Q}$ and the task space $\mathcal{M}_i$ by $\boldsymbol{h}_i : \mathcal{Q} \to \mathcal{M}_i$. Local coordinates for $\mathcal{Q}$ are denoted by $\boldsymbol{q} \in \mathbb{R}^n$ while $\boldsymbol{x}_i \in \mathbb{R}^{m_i}$, $i = \{1, 2\}$, $m_1 + m_2 \leq n$ are local coordinates for two levels of task priorities, where task 1 is higher in priority than task 2. The Jacobians are $\boldsymbol{J}_i = \frac{\partial \boldsymbol{h}_i(\boldsymbol{q})}{\partial \boldsymbol{q}}$.
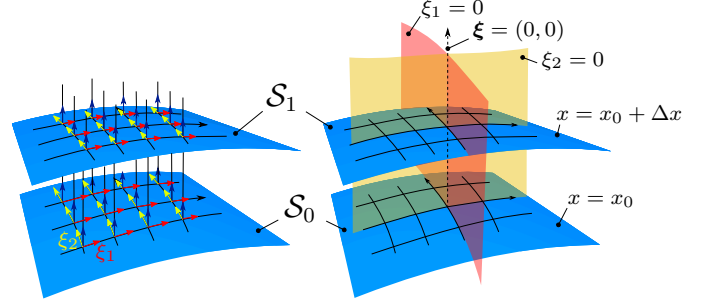
Fig. 1. The idea of orthogonal foliations (and coordinates): given the foliation induced by the forward kinematics of a redundant manipulator, we assign coordinates (ξ_1, ξ_2) on the self-motion motion manifolds (denoted in blue), such that the coordinates remain invariant when moving to another leaf of the foliation by following the task gradient (i.e the Jacobian in the kinematic case). This leads to foliations that are orthogonal to the task-induced foliation in the sense of a specified metric. Two leaves of the orthogonal foliations are shown in red and yellow.

While the higher priority task is usually a specification of the manipulation task performed with the end-effector, the lower priority often optimizes kinematic or dynamic properties of the robot. These objectives are formulated as functions of the configurations variables, usually being some kind of thresholded inverse distance functions for (self) collision avoidance [15]–[17], values required to optimize manipulability [18], [19] or inertial properties [20], [21].

The classical redundancy resolution on velocity level [3]–[7] at non-singular configurations $(\det(\boldsymbol{J}_1(\boldsymbol{q})\boldsymbol{J}_1^\mathsf{T}(\boldsymbol{q})) \neq 0)$ imposes the following relations between task and joint space velocities

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^{\#}(\boldsymbol{q})\dot{\boldsymbol{x}}_1 + \dot{\boldsymbol{q}}_N \tag{1a}$$

$$\dot{\boldsymbol{q}}_N = \boldsymbol{P}(\boldsymbol{q})\boldsymbol{J}_2^{\#}(\boldsymbol{q})\dot{\boldsymbol{x}}_2. \tag{1b}$$

Therein, $\dot{\boldsymbol{q}}_N$ are null space velocities, $\cdot^{\#}$ a pseudo-inverse and $\boldsymbol{P}(\boldsymbol{q})$ is a projector onto the null space of $\boldsymbol{J}_1$

$$\boldsymbol{P}(\boldsymbol{q}) = \boldsymbol{I} - \boldsymbol{J}_1^{\#A}(\boldsymbol{q})\boldsymbol{J}_1(\boldsymbol{q}), \tag{2}$$

with $\boldsymbol{J}_1^{\#A} = \boldsymbol{A}^{-1}\boldsymbol{J}_1^\mathsf{T}(\boldsymbol{J}_1\boldsymbol{A}^{-1}\boldsymbol{J}_1^\mathsf{T})^{-1}$ being the generalized pseudo-inverse [22], with $\boldsymbol{A}$ positive-definite and symmetric, satisfying $\boldsymbol{J}_1(\boldsymbol{q})\boldsymbol{P}(\boldsymbol{q}) = \boldsymbol{0}$ and thus ensuring $\boldsymbol{J}_1(\boldsymbol{q})\dot{\boldsymbol{q}}_N = \boldsymbol{0}$.

Similarly, in the framework of operational space control [1], redundancy resolution of the generalized joint force $\boldsymbol{\tau}$ considering primary and secondary task forces $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, respectively, is obtained according to

$$\boldsymbol{\tau} = \boldsymbol{J}_1^\mathsf{T}(\boldsymbol{q})\boldsymbol{f}_1 + \boldsymbol{\tau}_N \tag{3a}$$

$$\boldsymbol{\tau}_N = \boldsymbol{P}^\mathsf{T}(\boldsymbol{q})\boldsymbol{J}_2^\mathsf{T}(\boldsymbol{q})\boldsymbol{f}_2. \tag{3b}$$

Since the projector satisfies $\boldsymbol{P}^\mathsf{T}(\boldsymbol{q})\boldsymbol{J}_1^\mathsf{T}(\boldsymbol{q})\boldsymbol{f}_1 = \boldsymbol{0}$, it removes any generalized joint force component, which would interfere with the one generated by a primary task force $\boldsymbol{f}_1$ over the Jacobian $\boldsymbol{J}_1^\mathsf{T}(\boldsymbol{q})$. Using the inertia tensor $\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{q})$ in the pseudo-inverse, one obtains the dynamically decoupled projection, additionally satisfying that $\boldsymbol{\tau}_N$ generates no instantaneous primary task acceleration $\ddot{\boldsymbol{x}}_1$ [1].

B. Drawbacks of the Projection-based Approach

The projection-based approaches are the state-of-the-art approach for redundancy resolution for a good reason: they provide a flexible framework by allowing almost arbitrary additional tasks. However, there are also some problems that we would like to identify here. Later, in Section VIII-A and a supplementary video, we show examples of these problems.

Generally, tasks on different priorities are incompatible: we can not simultaneously achieve the tasks on all levels. The control errors of lower priority tasks will not vanish but instead converge to some non-zero minimum. The projection framework *allows* to have incompatible tasks. Incompatible tasks need projection[1]! The state-of-the-art relies on projection to make the tasks at least *locally* compatible on the level of velocities or forces. The global incompatibility of tasks causes a couple of follow-up problems.

Problem P1. Lower priority tasks may have no effect.

The lower priority tasks may totally conflict with the primary task in certain configurations. In this case, the lower priority tasks do not perform redundancy resolution because not all the robot's degrees of freedom are used.

Problem P2. Snapping between various minima.

This is probably among the largest problems in practice. Depending on the current configuration, the robot may converge to various local minima of the secondary tasks. Imagine a hand-guiding scenario: the robot has converged to some configuration. Now the user interacts and moves the robot to a new configuration that is contained in the region of attraction of another local minimum of the secondary tasks. In practice, users observe unexpected jumps in the robot's behavior when it snaps between different local minima of the secondary tasks: the robot behaves unpredictably.

The same effect occurs when controlling the robot in the primary task. Moving in the primary tasks causes a variation of the configuration — and thus possible movement to regions of attraction of other minima of the secondary tasks.

Problem P3. No integrability, no guaranteed return.

The joint velocity vector fields obtained by the pseudo-inverse of $\boldsymbol{J}_1$ and also the corresponding force fields are generally not integrable [24]: performing a closed loop $\boldsymbol{x}_1(t)$ for $\boldsymbol{x}_1(t_0) = \boldsymbol{x}_1(t_1)$ in the main task will in general not drive the system back to the original configuration $\boldsymbol{q}(t_0) \neq \boldsymbol{q}(t_1)$. Using the projection framework to add secondary tasks, helps, but does not solve this problem. Depending on the past configuration path, the robot may converge to different local minima of the secondary tasks (example in Sec. VIII-A. P3 also leads to:

Problem P4. Entirety of tasks are not usable for planning.

Specifying the task coordinates on all levels is insufficient to determine the robot's joint configuration: one also needs past joint configurations. Hence, planning is not possible in the task coordinates.

Problem P5. Projected forces are no gradients of potentials.

Passivity-based multi-layer control approaches [1], [16], [25] like the well-known impedance controller generate the task

[1]Especially for a small degree of redundancy. For highly redundant systems, the performance increase of projection methods stagnates, and one might consider simply superimposing impedance controllers [23].

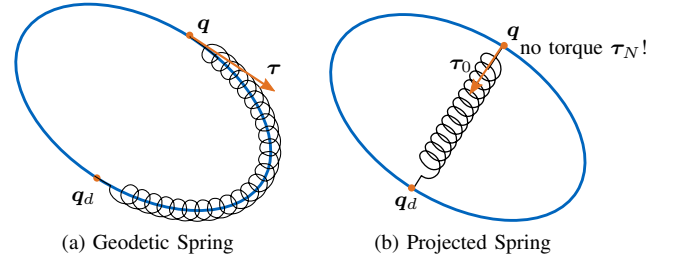(a) Geodetic Spring   (b) Projected Spring

Fig. 2. Comparison of a geodetic spring due to the position-level redundancy resolution approach in this paper and a projected spring by the state of the art approach (b). The spring in (b) has no effect as the joint torques vanish after the projection. The blue ellipses sketch a task self-motion manifold.

forces in (3a), (3b) as gradient force fields $\boldsymbol{f}_i^{\mathsf{T}} = \frac{\partial U_i(\boldsymbol{x}_i)}{\partial \boldsymbol{x}_i}$, $i = 1, 2$, derived from potential fields U_i formulated in task coordinates $\boldsymbol{x}_i$. The secondary task force $\boldsymbol{f}_2$ is projected (3b) before it acts on the robot. The projection modifies the energy flows in the system and destroys the safety-critical property of passivity[2] [27], [28]. Because of the projection, we cannot use the virtual potentials in secondary tasks as potential fields (Lyapunov functions) to prove convergence of the system. Providing effective dynamical decoupling and proving convergence under these conditions is a complex challenge, still a topic of active research today [16], [26], [29]–[32]. In most null space projection-based controllers, the assumption of structural feasibility of the task coordinates is made [32] and this problem is deferred to a planning task.

Problem P6. Hidden springs.

Suppose the controller in the secondary task is realized by some virtual spring $\boldsymbol{f}_2 = -\boldsymbol{K}_2(\boldsymbol{x}_2 - \boldsymbol{x}_2^d)$ in secondary task coordinates as in impedance control. The robot usually converges to local minima in the secondary tasks, i.e., one obtains non-zero torques $\boldsymbol{\tau} = \boldsymbol{J}_2^{\mathsf{T}} \boldsymbol{f}_2 \neq \boldsymbol{0}$ which vanish after the projection, such that $\boldsymbol{P}^{\mathsf{T}} \boldsymbol{J}_2^{\mathsf{T}} \boldsymbol{f}_2 = \boldsymbol{0}$. This corresponds to a tensioned virtual spring hidden behind the projector. Even for compatible tasks, the spring may not relax entirely in some cases: Fig. 2b sketches this situation for a joint space spring.

Slight variations of the configurations or the commanded motions may cause the previously hidden spring to discharge and create large forces and high velocity. Additionally, a strongly loaded virtual spring often causes shaky behavior and unpleasant sounds at the configuration where it hides behind the projector.

C. Redundancy Resolution on Position Level

In this publication, we would like to introduce a completely new approach to redundancy resolution:

Problem Statement. *Given the family of self-motion manifolds determined by the task space of a redundant robot, find subspaces of the configuration space that are compatible, not interfering with the task space.*

These compatible (or decoupled) subspaces will be locally described by constraint functions $\boldsymbol{\xi}(\boldsymbol{q}) = c$. Compatible means that the Jacobian of the main task coordinates $\boldsymbol{x} = \boldsymbol{h}(\boldsymbol{q})$ is orthogonal to the Jacobian of the additional task $\boldsymbol{\xi}(\boldsymbol{q})$, i.e., $\boldsymbol{J}_x \boldsymbol{J}_\xi^{\mathsf{T}} = \boldsymbol{0}$ or $\boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{J}_\xi^{\mathsf{T}} = \boldsymbol{0}$, depending on the desired properties of the coordinates.

[2]Unless also feeding back the external forces [26].

Fig. 1 schematically visualizes the idea: let the blue surfaces denote two self-motion manifolds [33] of the main task for two given task positions. These are embedded submanifolds of the configuration space. The desired new subspaces (or submanifolds) are chosen such that they intersect the task self-motion manifolds orthogonally[3], thereby generating coordinate systems on each self-motion manifold simultaneously. The red and yellow surfaces sketch surfaces on which the new coordinates $\boldsymbol{\xi}_i(\boldsymbol{q})$ are constant. It turns out that the entirety of self-motion manifolds can be interpreted as a foliation, which is a key concept for the following sections. Likewise, our new compatible subspaces can be interpreted as a foliation, and the yellow and red surfaces in Fig. 1 are leaves of this foliation. Because the new foliation is orthogonal to the task foliation, we call them *orthogonal foliations*. Such a redundancy resolution at configuration level, based on coordinates for the self-motion manifolds, would provide a more global perspective, which is not limited to the linearized vicinity of the current configuration.

Such subspaces would solve all the problems P1-P6 by using the coordinates $\boldsymbol{\xi}(\boldsymbol{q})$ instead of the projections $\boldsymbol{P}(\boldsymbol{q})$. This would also restore the passivity of the controlled system. Potential fields in the coordinates $\boldsymbol{\xi}(\boldsymbol{q})$ will correspond to *geodetic springs*, directly depending on the distance between the two end-points on the self-motion manifolds (Fig. 2a). The new Jacobian $\boldsymbol{J}_\xi(\boldsymbol{q})$ will span the same space as the projector matrix in the projection framework. $\boldsymbol{J}_x \boldsymbol{J}_\xi^\mathsf{T} = \boldsymbol{0}$ corresponds to $\boldsymbol{J}_\xi$ spanning the same space as the Moore-Penrose-based projector $\boldsymbol{P}(\boldsymbol{q})$ and $\boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{J}_\xi^\mathsf{T} = \boldsymbol{0}$ to $\boldsymbol{J}_\xi$ spanning the same space as the dynamically consistent projector.

In [34], we looked at a foliation of surfaces in task space. This time we look at a foliation in configuration space induced by the choice of the main task. In contrast to the previous work, however, these are not always two-dimensional, they are in configuration space, and we need another kind of measure of orthogonality.

Note that we do not aim to *repair* an existing secondary task that one would use in the projection framework. Instead, we aim to *construct* a new secondary task that is compatible with the main task - and does not need projection at all.

D. Contributions

Our contributions are:

(C1) Formulating redundancy resolution on position level for the first time;

(C2) Introducing the differential geometric tools required to systemically develop this new theory, including the concepts of involutivity and foliations;

(C3) A condition for the existence of orthogonal foliations corresponding to exact position-level redundancy resolution;

(C4) In-depth analysis of orthogonal foliations for two and three degrees of freedom robots;

(C5) Numerical schemes for approximate solutions to orthogonal coordinates for higher dimensional robots; and

(C6) Simulation and hardware experiments for position-level redundancy resolution using passive impedance control in task and orthogonal coordinates and comparison to the state-of-the-art projection-based control.

[3]In a certain sense of *orthogonal* to be derived later in Sec. IV

II. RELATED WORK

Some research groups have previously dealt with the non-integrability problem (P3). Already in 1991, Mussa-Ivaldi and Hogan [35] proposed using a specially weighted pseudo-inverse of the Jacobian to obtain an integrable system. Combining this with kinematic control, they achieved that one can start at some configuration, perform a closed path in task space, and return to the initial configuration. This is a subtask to what we want to achieve but does not provide coordinates $\boldsymbol{\xi}(\boldsymbol{q})$. One year later De Luca et al. [36] found special start configurations for which the system is cyclic.

More recently, Hauser and Emmons [37] have proposed an approach to *global redundancy resolution* constructing a probabilistic roadmap in configuration space that allows continuous pseudo-inversion of the forward kinematics; and Xie et al. [38] use a recurrent neural network for repetitive motions of redundant robots. These publications provide solutions to global redundancy resolution by ensuring that each main task configuration is achieved by exactly one joint configuration. This way, one achieves integrability, but loses flexibility as the remaining degrees of freedom are eliminated and cannot be used for secondary tasks. In this paper, we want to retain all the degrees of freedom.

Haug [39] recently proposed a cyclic differentiable manifold representation of redundant manipulator kinematics. He achieves cyclic behavior and still retains the freedom to use redundancy. The approach is based on the insight that in an enrivonment around some *base point* $\bar{\boldsymbol{q}}^j$ [40]:

$$\boldsymbol{q}(\boldsymbol{x}, \boldsymbol{v}) = \bar{\boldsymbol{q}}^j + \boldsymbol{V}^j \boldsymbol{v} - \boldsymbol{U}^j \psi^j(\boldsymbol{x}, \boldsymbol{v}) \qquad (4)$$

where $\boldsymbol{U}^j = \boldsymbol{J}_x^\mathsf{T}(\bar{\boldsymbol{q}}^j)$ and $\boldsymbol{V}^j \in \mathbb{R}^{n \times (n-m)}$ spans the null space of $\boldsymbol{J}_x(\bar{\boldsymbol{q}}^j)$ such that $\boldsymbol{J}_x(\bar{\boldsymbol{q}}^j) \boldsymbol{V}^j = \boldsymbol{0}$ and $\boldsymbol{V}^{j\mathsf{T}} \boldsymbol{V}^j = \boldsymbol{I}$. The function $\psi^j(\boldsymbol{x}, \boldsymbol{v})$ is smooth and is determined iteratively by a Newton-Raphson method during operation [39]. The values $\boldsymbol{v}$ can be interpreted as coupled local coordinates, describing the arm configuration within the subspace around $\bar{\boldsymbol{q}}^j$. Because they are coupled to the main task $\boldsymbol{x}$, the function $\psi^j(\boldsymbol{x}, \boldsymbol{v})$ depends on $\boldsymbol{x}$ and $\boldsymbol{v}$.

Constructing a null space basis $\boldsymbol{J}_{\xi'}(\boldsymbol{q})$ independently of an integral manifold $\boldsymbol{\xi}(\boldsymbol{q})$, i.e., purely based on linear algebra, is, of course, always possible and has been frequently addressed in the literature [8], [29], [41]–[43]. Performing a null space projection based on (1b) or (3b) can be equivalently formulated using the null space basis, see Sec. IV-B. However, it is well-known [24] that the velocity vector fields obtained by the projector $\boldsymbol{P}(\boldsymbol{q})$ in (1b), and equivalently, obtained by $\dot{\boldsymbol{\xi}}' = \boldsymbol{J}_\xi'(\boldsymbol{q})\dot{\boldsymbol{q}}$, are in general not integrable, not fulfilling the conditions of the Frobenius theorem [44].

III. A GEOMETRIC LIGHT ON REDUNDANT ROBOTS

We address the question of generalized configuration coordinates, which are orthogonal to given high-priority task coordinates. This requires moving from vector spaces and linear algebra to smooth manifolds and the tools of differential geometry and topology. The first differential geometry-based analysis of redundant robots goes back to [33], who introduced the notion of self-motion manifolds to robotics. Differential-geometric robot kinematics and dynamics was given in [40], [45], [46] and has since then been constantly used in robotics [47]–[49], although not yet belonging to the standard, widely used repertoire.
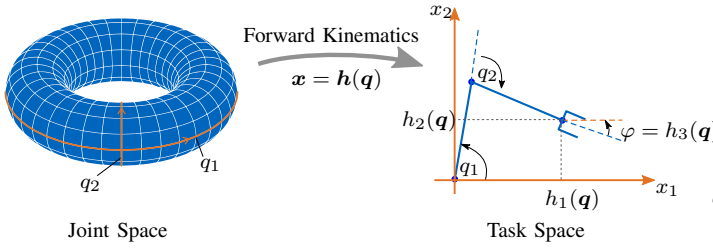
Fig. 3. The forward kinematics is a mapping between the joint manifold $\mathcal{Q}$ and the task manifold $\mathcal{M}$. The task space manifold in this example is $\mathcal{M} = SE(2)$, which we sketch in a coordinate representation on the right.

Let $\mathcal{Q}$ denote the configuration space manifold of a robot and $\mathcal{M}$ the task space manifold. We call $\mathcal{H} : \mathcal{Q} \to \mathcal{M}$ the forward kinematics map. Further, let $n = \dim \mathcal{Q}$ and $m = \dim \mathcal{M}$. A robot will be called redundant for a specific task if $m < n$ [2]. Being redundant depends not only on the robot's degrees of freedom but also on the choice of task space. For instance, when $\mathcal{M} = SE(2)$ or $\mathcal{M} = SE(3)$, this is precisely the case if $n > 3$ or $n > 6$.

Of course, various local coordinates can describe the task manifold, as highlighted in [11] for the representation of orientations. The same applies to the configuration space manifold $\mathcal{Q}$. Most statements will be coordinate-invariant when following a differential geometric perspective.

We denote local coordinates of a point in $\mathcal{Q}$ by $\boldsymbol{q} \in \mathbb{R}^n$ and point in $\mathcal{M}$ by $\boldsymbol{x} \in \mathbb{R}^m$. In coordinates the mapping $\mathcal{H}$ will be denoted $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ such that $\boldsymbol{x} = \boldsymbol{h}(\boldsymbol{q})$. Fig. 3 shows an example of such a mapping. We will use a fair amount of abuse of notation in this paper: by $\boldsymbol{q}/\boldsymbol{x}$ we refer to both a point $\boldsymbol{q} \in \mathcal{Q}/\boldsymbol{x} \in \mathcal{M}$ and to its coordinates $\boldsymbol{q} \in \mathbb{R}^n/\boldsymbol{x} \in \mathbb{R}^m$.

We assume there are no points in the configuration space $\mathcal{Q}$ where the map $\mathcal{H}$ becomes singular, i.e. its push-forward has constant rank. A singularity-free manifold $\bar{\mathcal{Q}}$ can be constructed by removing the singular points from $\mathcal{Q}$. Note that $\bar{\mathcal{Q}}$ is still a (now open) manifold when removing singular configurations. We will use the symbol $\mathcal{Q}$ to denote the singularity-free manifold throughout the paper.

A. Push-Forwards and Pullbacks

Velocities live in the tangent spaces $\mathcal{T}_q\mathcal{Q}$ and $\mathcal{T}_x\mathcal{M}$ of the manifolds. Generalized forces being covectors lie within the dual cotangent spaces $\mathcal{T}_q^*\mathcal{Q}$ and $\mathcal{T}_x^*\mathcal{M}$. The push-forward $\mathcal{H}_*$ of $\mathcal{H}$ maps vectors in $\mathcal{T}_q\mathcal{Q}$ to vectors in $\mathcal{T}_{\mathcal{H}(q)}\mathcal{M}$ providing a local linear approximation of $\mathcal{H}$ (Fig. 4). Similarly, the pullback $\mathcal{H}^*$ maps covectors in $\mathcal{T}_{\mathcal{H}(q)}^*\mathcal{M}$ to covectors in $\mathcal{T}_q^*\mathcal{Q}$. Written in coordinates, the push-forward $\mathcal{H}_*$ can be thought of the Jacobian $\boldsymbol{J} = \frac{\partial \boldsymbol{h}(\boldsymbol{q})}{\partial \boldsymbol{q}}$ and the pullback the transposed Jacobian. This vocabulary is beneficial to apply statements and theorems from the differential geometry world to robotics.

B. Forward Kinematics, Submersions & Foliations

From a differential geometric perspective, the forward kinematics map of a redundant robot is a *submersion*: a map whose push-forward is surjective everywhere [50]. Note that this requires the absence of singularities, as assumed in this paper. From the submersion level set theorem [50, Cor. 5.13] tells us that for each $\boldsymbol{x}_0 \in \mathcal{M}$ the level set $\mathcal{H}(\boldsymbol{q}) = \boldsymbol{x}_0$ of $\mathcal{H}$ is a closed embedded submanifold $\mathcal{S}_0 \subset \mathcal{Q}$, i.e., $\forall \boldsymbol{q} \in \mathcal{S}_0 : \mathcal{H}(\boldsymbol{q}) = \boldsymbol{x}_0$. Further, the dimension of $\mathcal{S}$
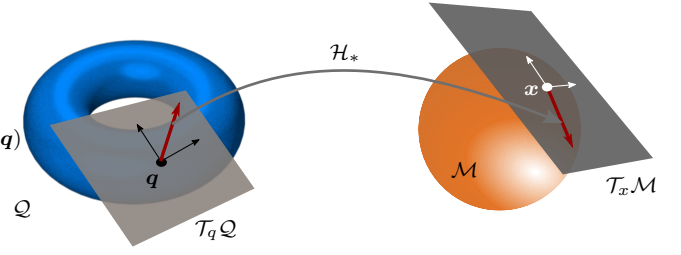

Fig. 4. The push-forward transforms vectors in the tangent space of the configuration space to vectors in the tangent space of the task space manifold.
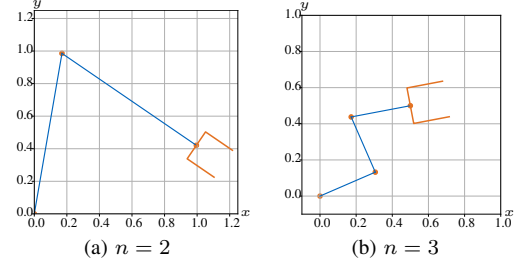

Fig. 5. Planar robots with $n = 2$ and $n = 3$ degrees of freedom used as example in this paper.

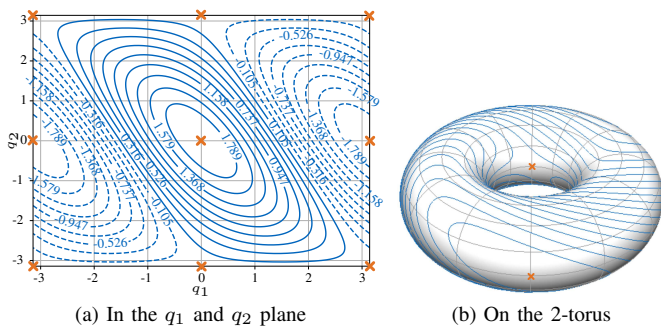is $n - m$. These level set submanifolds are nothing but the well-known self-motion manifolds [11], [33].

Taking the entirety of the level-set submanifolds of $\mathcal{Q}$ induced by $\mathcal{H}$, a foliation of codimension $m = \dim \mathcal{M}$ is obtained. The foliation has as many dimensions as the degree of redundancy $r = n - m$. Each leaf of the foliation is a self-motion manifold for one task configuration. This foliation view on forward kinematics provides insight into the global behavior of the self-motion manifolds: some structure is present. *The self-motion manifolds fit together locally like slices in a flat chart* [50, p. 501].

Consider the two degrees of freedom (DoF) example robot in Fig. 5a, whose configuration manifold is the 2-torus $\mathbb{T}^2$ (cf. Fig. 3). A one-dimensional task manifold and, thus, a scalar forward kinematics function is required to obtain a redundant system. Without loss of generality, let us choose $h(\boldsymbol{q}) = x$, i.e., we only care about the x-component of the end-effector position (horizontal). The blue lines in Fig. 6a show some level sets of the forward kinematics function h. Note that the manifolds are closed on the torus $\mathbb{T}^2$ (Fig. 6b). Except at the singularities $\boldsymbol{q}_1 = [0, 0]$, $\boldsymbol{q}_2 = [\pm\pi, 0]$, $\boldsymbol{q}_3 = [0, \pm\pi]$ and $\boldsymbol{q}_4 = [\pm\pi, \pm\pi]$ the self-motion manifolds look locally like slices of a flat chart. Also, note that those nine singular configurations in coordinate representation correspond to only four points on the torus.

Similarly, consider the second example robot with 3 DoF in Fig. 5b. We again consider only the x-component of the end-effector position and, consequently, obtain submanifolds of dimension two. Consider the submanifold of $\mathcal{Q}$ corresponding to $h(\boldsymbol{q}) = 0$ shown in Fig. 7a. Now, we obtain a curved surface as self-motion manifold. The surface is still closed on the 3-torus $\mathbb{T}^3$. We show only one leaf of the foliation. An additional rendering of multiple leaves for different level sets of $h(\boldsymbol{q})$ is shown in Fig. 7b. Imagine taking all of them. Then the manifolds look locally like a stack of paper.

Finally, let us consider another choice of task coordinates for the 3 DoF robot by taking the end-effector position's x

(a) In the q_1 and q_2 plane   (b) On the 2-torus

Fig. 6. Foliation on the 2-torus induced by a forward kinematics function. The leaves of the foliation correspond to the well-known self-motion manifolds. In the left image, the opposing sides of the rectangle are identified according to the torus topology. The orange crosses mark the singular configurations.

(a) One leaf   (b) Multiple leaves

Fig. 7. Self-motion manifold(s) for a planar manipulator with three degrees of freedom and the x-component of the end-effector position as single task space. The surfaces are generated using marching cubes [51] on the forward kinematics function. On the left: a single leaf of the foliation. You may want to 3D-print the model from the supplementary material for a more intuitive visualization. The white lines additionally show level sets of the y-coordinate. On the right: multiple leaves of the foliation.

and y components. We then obtain line-shaped self-motion manifolds as shown in Fig. 8. Note that one can classify the manifolds into two categories. One class of lines encircles the $q_2 = q_3 = 0$ line, while the others do not. Also, some positions in task space generate one self-motion manifold, while others generate two disconnected ones. The latter ones correspond to the redundant version of *elbow-up and elbow-down* configurations. The video in the supplementary material helps to understand Fig. 8. Consider [33] for a more rigorous analysis. The main observation required for this paper is that the lines form a foliation of $\mathcal{Q}$. Each point $\boldsymbol{q} \in \mathcal{Q}$ is part of exactly one of the curves.

IV. ORTHOGONAL FOLIATIONS

This paper aims to find new foliations orthogonal to the task-induced foliation $\boldsymbol{h}(\boldsymbol{q})$ to the best extent possible. Thinking of the leaves of the new foliations as submanifolds of $\mathcal{Q}$, we denote the constraints inducing these foliations by $\boldsymbol{\xi}(\boldsymbol{q})$. Orthogonality is defined here as follows: Given the Jacobians $\boldsymbol{J}_x$ and $\boldsymbol{J}_\xi$ defined in terms of

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_x(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{5a}$$

$$\dot{\boldsymbol{\xi}} = \boldsymbol{J}_\xi(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{5b}$$

(a) View 1   (b) View 2

Fig. 8. Foliation of the 3-torus. Opposing faces of the cube are identified. Considering the planar Cartesian position of a 3 DoF planar robot as task induces a one-dimensional foliation. A video in the supplementary material helps to understand this Figure.

we would like to find $\boldsymbol{\xi}(\boldsymbol{q})$ such that the two Jacobians are orthogonal with respect to the metric tensor $\boldsymbol{A}$:

$$\boldsymbol{J}_x(\boldsymbol{q})\boldsymbol{A}^{-1}\boldsymbol{J}_\xi^{\mathsf{T}}(\boldsymbol{q}) = \boldsymbol{0}, \tag{6}$$

with $\boldsymbol{A} = \boldsymbol{I}$ or $\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{q})$ to achieve kinematic or dynamic decoupling, respectively[4].

The property (6) corresponds to the condition on the Jacobians and projectors for the classical velocity- and torque-based redundancy resolution. However, instead of first defining a null space base matrix $\boldsymbol{J}_\xi$ and then checking for integrability, we follow another procedure: we start with a parametrizable foliation and then ask for orthogonality of the Jacobians. If orthogonality is achievable everywhere, we will call the resulting foliations *orthogonal foliations*.

On the other hand, if the task-induced foliation does not admit orthogonal foliations satisfying (6) everywhere, we will minimize the residuum over the entire considered configuration space. In the latter case, we will call the foliations *quasi-orthogonal foliations*.

A. Geometric Intuition of Orthogonal Foliations

Before introducing the proposed approach's rigorous differential geometric formulation, we explain the main ideas on simple 2 and 3 DoF examples.

Let us consider again the 3 DoF robot with 1-dimensional task space (cf. Fig. 7). Fig. 1 sketches two leaves of the foliation induced by the forward kinematics. On a base leaf, say $\mathcal{S}_0$ defined by $h(\boldsymbol{q}) = x_0$, we choose a chart $\boldsymbol{\xi}_{x_0}$ with coordinates $\{\xi_1, \xi_2\}$. The Jacobian[5] $\boldsymbol{J}_\xi^{x_0}$ associated to $\boldsymbol{\xi}_{x_0}$ forms a basis for the tangent space at each point, as denoted by the red and yellow arrows. The vectors corresponding to the Jacobian $\boldsymbol{J}_x$ of the task coordinate are dark blue. Regarding the manifolds as embedded in Euclidean space, $\boldsymbol{J}_x$ is at each point orthogonal to the manifold and thus normal to the tangent space, satisfying $\boldsymbol{J}_x \boldsymbol{J}_\xi^{\mathsf{T}} = \boldsymbol{0}$. Now, integrating along the Jacobian $\boldsymbol{J}_x$ starting from a point $\boldsymbol{\xi}_{x_0} = \{0, 0\}$ from $\mathcal{S}_0$, one obtains a curve which intersects $\mathcal{S}_1$. We assign the coordinates $\boldsymbol{\xi}_{x_1} = \{0, 0\}$ to this point. Using this procedure, we can define an equivalence relation between all points in the chart $\boldsymbol{\xi}_{x_0}$ and points locally from $\mathcal{S}_1$, thus inducing a chart $\boldsymbol{\xi}_{x_1}$ on $\mathcal{S}_1$. This way, we constructed two new foliations $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ orthogonal to $\boldsymbol{x}$ in the Euclidean sense. The leaves $\xi_1 = 0$ (red) and

---

[4]See Sec. IV-B for clarifying why the inverse of $\boldsymbol{A}$ appears in (6).

[5]The superscript in $\boldsymbol{J}_\xi^{x_0}$ shall denote that this Jacobian is only valid at configurations $\boldsymbol{q}$ where $h(\boldsymbol{q}) = x_0$.
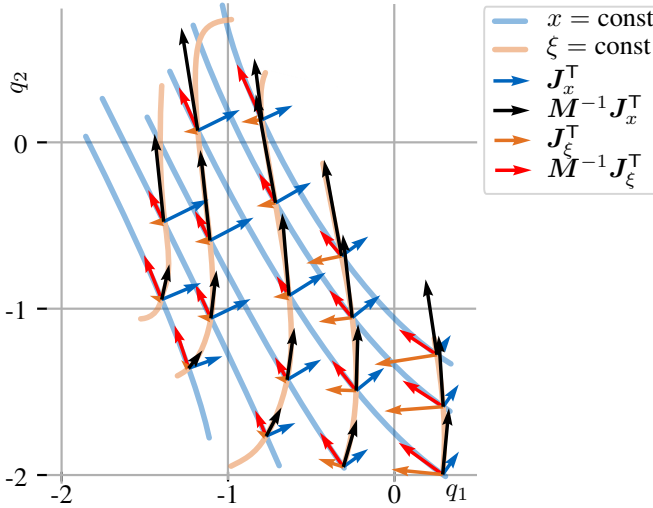
Fig. 9. Orthogonal foliations for the example of a bi-dimensional configuration space and a single task coordinate for dynamic consistency.

$\xi_2 = 0$ (yellow) are exemplarily displayed in Fig. 1. Note that the procedure described above works for any joint space dimension n, as long as the task space is one-dimensional and thus the dimension of the task-space induced foliation is $n-1$.

The above example leads to foliations $\boldsymbol{\xi}$ satisfying the generalized orthogonality condition (6) for $\boldsymbol{A} = \boldsymbol{I}$. For the case $\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{q})$, the geometric interpretation and visualization become more complex, and we discuss it for the simpler 2 DoF case with 1-dimensional task space. The task-induced foliation is then one-dimensional. The blue curves in Fig. 9 show some of its leaves $\mathcal{S}_i$, and dark blue arrows show the Jacobian $\boldsymbol{J}_x$. As usual, it satisfies $\boldsymbol{J}_x(\boldsymbol{q})\boldsymbol{v} = 0$ for any tangent vector $\boldsymbol{v} \in \mathcal{T}_q\mathcal{S}_i$.[6] The condition $\boldsymbol{J}_x\boldsymbol{M}^{-1}\boldsymbol{J}_\xi^\mathsf{T} = 0$ thus implies that now $\boldsymbol{M}^{-1}\boldsymbol{J}_\xi^\mathsf{T}$ is tangent to the foliation and forms a basis for the null space of $\boldsymbol{J}_x$. Red arrows display this vector field in Fig. 9. In this case, $\boldsymbol{J}_\xi$ being orthogonal to the $\boldsymbol{\xi}$ foliation, does not belong to the tangent bundle of the task foliation but is rotated accordingly (orange arrows). Finally, integrating the $\boldsymbol{M}^{-1}\boldsymbol{J}_x^\mathsf{T}$ vector field (black arrows) yields leaves of the $\boldsymbol{\xi}$ foliation, displayed as orange lines.

Let's conclude: the orthogonal foliation for dynamic consistency has $\boldsymbol{M}^{-1}\boldsymbol{J}_x^\mathsf{T}$ as tangent vectors and $\boldsymbol{J}_\xi$ as normal vectors, while the task foliation has $\boldsymbol{M}^{-1}\boldsymbol{J}_\xi^\mathsf{T}$ as tangent vector and $\boldsymbol{J}_x$ as normal vector. Again, these results can be easily extended to configuration spaces of any dimension as long as the task space has a single dimension.

Finally, let us consider the 3 DoF case, with 2-dimensional task space. We again choose the Euclidean orthogonality condition for simplicity: $\boldsymbol{A} = \boldsymbol{I}$. The task foliation consists of curves, as displayed in Fig. 10. At each point, the rows of $\boldsymbol{J}_x$ span a plane orthogonal to the corresponding task foliation leaf (or self-motion manifold). They define two vector fields in the tangent bundle. For these vector fields to be integrable, they must satisfy the Frobenius theorem [44]. In general, this condition will, however, not be satisfied.

Take a configuration $\boldsymbol{q}_0$. This will be part of precisely one leaf $\mathcal{S}_0$ of the task foliation. Let this leaf of the foliation be defined by $\forall \boldsymbol{q} \in \mathcal{S}_0 : \boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{x}_0$. Now, we follow the vector

[6]By $\mathcal{T}_q\mathcal{S}$ we refer to the subspace $\mathcal{T}_q\mathcal{S} \subset \mathcal{T}_q\mathcal{Q}$ of the tangent space $\mathcal{T}_q\mathcal{Q}$ spanned by the tangent vectors of the embedded submanifold $\mathcal{S} \subset \mathcal{Q}$, i.e. the tangent vectors of $\mathcal{T}_{\xi(q)}\mathcal{S}$ given as embedded vectors in $\mathcal{T}_q\mathcal{Q}$.
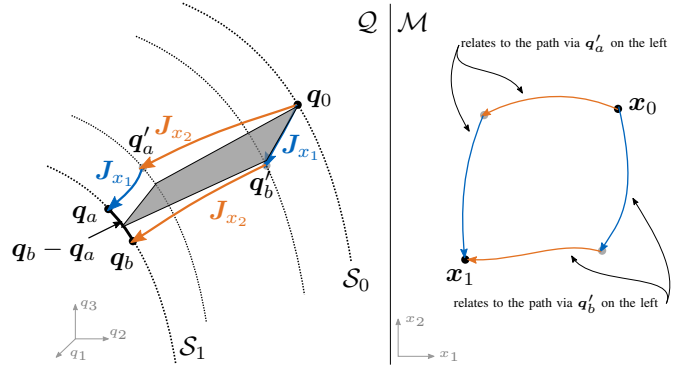
Fig. 10. The distribution of vector fields spanned by the task space Jacobian is generally non-involutive. Our solution provides the best integral manifold approximation (gray surface) in a specified optimalily sense.

fields given by the rows of the Jacobians as shown in Fig. 10. We take two paths leading to the same task configuration $\boldsymbol{h}(\boldsymbol{q}_a) = \boldsymbol{h}(\boldsymbol{q}_b) = \boldsymbol{x}_1$. Due to the non-involutivity we will generally have $\boldsymbol{q}_a \neq \boldsymbol{q}_b$, where $\boldsymbol{q}_a, \boldsymbol{q}_b \in \mathcal{S}_1$. Therefore, the two final configurations $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ cannot form, together with $\boldsymbol{q}_0$, a single leaf of an orthogonal foliation. When navigating from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$ based on the gradient vector fields, the final configuration at $\boldsymbol{x}_1$ will depend on the path between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ in task space. This is the essence of non-integrability. In the infinitesimal case, the difference $\boldsymbol{q}_b - \boldsymbol{q}_a$ corresponds to the Lie bracket $[\boldsymbol{J}_{x_1}^\mathsf{T}, \boldsymbol{J}_{x_2}^\mathsf{T}]$ of the two vector fields. In general, there is no integral surface that, even locally, is orthogonal to every leaf of the task foliation and would intersect $\mathcal{S}_1$ in a single point independent of the path.

However, in this case, one can ask to find the surface, which is *as orthogonal as possible* to the task foliation, in terms of a cost function. This means that $\boldsymbol{J}_x$ shall be *as tangent as possible* to the desired foliation $\boldsymbol{\xi}$. A leaf of this foliation, passing through $\boldsymbol{q}_0$ is visualized in gray in Fig. 10.

B. Null Space Jacobians and Projectors

Let us return to the relation between a null space basis $\boldsymbol{J}_\xi(\boldsymbol{q})$ and projection matrix $\boldsymbol{P}(\boldsymbol{q})$. Independent of the existence question about an integral manifold, a null-space basis on velocity level, as defined in (5a)-(6), can always be constructed. In compact form, (5a)-(5b) can be written as

$$\dot{\boldsymbol{\varphi}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{7}$$

with $\dot{\boldsymbol{\varphi}} = \begin{bmatrix} \dot{\boldsymbol{x}}^\mathsf{T} & \dot{\boldsymbol{\xi}}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and

$$\boldsymbol{J}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_x(\boldsymbol{q}) \\ \boldsymbol{J}_\xi(\boldsymbol{q}) \end{bmatrix}. \tag{8}$$

Since $\dot{\boldsymbol{\xi}}$ is constructed such that $\boldsymbol{J}(\boldsymbol{q})$ is full rank, the stacked Jacobian $\boldsymbol{J}(\boldsymbol{q})$ can be inverted. It can be verified using (6) that the inverse is

$$\boldsymbol{J}^{-1}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_x^{\#A}(\boldsymbol{q}) & \boldsymbol{J}_\xi^{\#A}(\boldsymbol{q}) \end{bmatrix}, \tag{9}$$

since $\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{q}) = \boldsymbol{I}$. This implies as well $\boldsymbol{J}^{-1}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}) = \boldsymbol{I}$ leading to

$$\boldsymbol{J}_x^{\#A}(\boldsymbol{q})\boldsymbol{J}_x(\boldsymbol{q}) + \boldsymbol{J}_\xi^{\#A}(\boldsymbol{q})\boldsymbol{J}_\xi(\boldsymbol{q}) = \boldsymbol{I}. \tag{10}$$

Therefore, there is a simple relation between the projector $\boldsymbol{P}(\boldsymbol{q})$ defined as in (2), the task Jacobian, and the null space basis $\boldsymbol{J}_\xi(\boldsymbol{q})$:

$$\boldsymbol{P}(\boldsymbol{q}) = \boldsymbol{I} - \boldsymbol{J}_x^{\#A}(\boldsymbol{q})\boldsymbol{J}_x(\boldsymbol{q}) = \boldsymbol{J}_\xi^{\#A}(\boldsymbol{q})\boldsymbol{J}_\xi(\boldsymbol{q}). \tag{11}$$
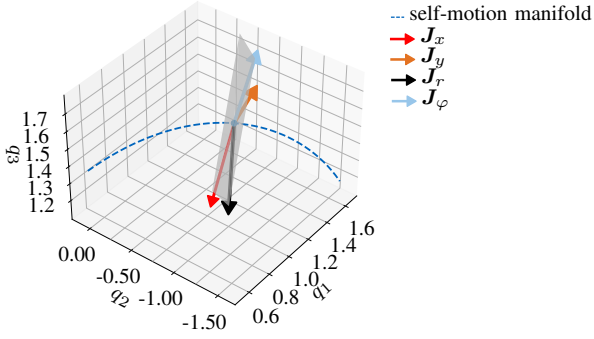
Fig. 11. The subspace spanned by the rows of the task Jacobian does not depend on the choice of task coordinates. The task is the position of the end-effector. We evaluate the Jacobian and plot the rows for Cartesian coordinates (red and orange) and for polar coordinates (black and light-blue). The gray plane visualizes the 2d subspace spanned by all four vectors.

This highlights that in cases in which exact orthogonality as demanded in (6) can be achieved for the foliation $\boldsymbol{\xi}(\boldsymbol{q})$, projection-based control approaches are equivalent on velocity and torque level to the controllers based on the orthogonal manifold. At the same time, the latter has the advantage of additionally providing a coordinate (position) level.

C. On Coordinate Independence

By looking at a concrete example, let us clarify why the integrability conditions on $\boldsymbol{J_x}$ are indeed coordinate-independent. Take the 3 DoF manipulator with two-dimensional task space and consider two different choices of task coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} = \boldsymbol{h}_c(\boldsymbol{q}) \text{ and } \begin{bmatrix} r \\ \varphi \end{bmatrix} = \boldsymbol{h}_p(\boldsymbol{q}), \qquad (12)$$

where both specify the position of the end-effector, once in Cartesian and once in polar coordinates. In the latter case, r represents the radius and φ the angle. The associated ($\mathbb{R}^{2\times 3}$) Jacobians are

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \boldsymbol{J}_c(\boldsymbol{q})\dot{\boldsymbol{q}} \text{ and } \begin{bmatrix} \dot{r} \\ \dot{\varphi} \end{bmatrix} = \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}}, \qquad (13)$$

where $\boldsymbol{J}_c = \partial \boldsymbol{h}_c/\partial \boldsymbol{q}$ and $\boldsymbol{J}_p = \partial \boldsymbol{h}_p/\partial \boldsymbol{q}$.

Next, consider the function $\mathrm{pol}(x, y) = (r, \varphi)$ mapping positions in the plane from Cartesian to polar coordinates. One can write the function $\boldsymbol{h}_p$ as a composition of $\boldsymbol{h}_c$ and pol, i.e.,

$$\boldsymbol{h}_p(\boldsymbol{q}) = \mathrm{pol}(\boldsymbol{h}_c(\boldsymbol{q})). \qquad (14)$$

Let $\boldsymbol{J}_{\mathrm{pol}} \in \mathbb{R}^{2\times 2}$ denote the Jacobian of pol and let us differentiate (14)

$$\begin{bmatrix} \dot{r} \\ \dot{\varphi} \end{bmatrix} = \boldsymbol{J}_{\mathrm{pol}}\boldsymbol{J}_c\dot{\boldsymbol{q}} = \boldsymbol{J}_p\dot{\boldsymbol{q}}. \qquad (15)$$

It follows that $\boldsymbol{J}_p = \boldsymbol{J}_{\mathrm{pol}}\boldsymbol{J}_c$. If $\boldsymbol{J}_{\mathrm{pol}}$ is full-rank, then $\boldsymbol{J}_p$ and $\boldsymbol{J}_c$ will have the same null space. Moreover, the rows of $\boldsymbol{J}_c$ and $\boldsymbol{J}_p$ all span the same subspace of $\mathbb{R}^3$, which is orthogonal to the self-motion manifold at each point. Fig. 11 shows this for one point on one of the task foliations.

This reasoning is valid for any dimension of configuration and task space. In differential geometric terms, the orthogonal linear subspaces at all points of the task foliation form a distribution[7] whose integrability we are interested in. This distribution only depends on the mapping to the task manifold

but not on its charts. Different choices of coordinates on the task manifold will create different vector fields. These, however, will all span the same distribution.

D. Differential Geometric Formulation

The dynamics of a multi-body system is naturally associated with a Riemannian manifold $(\mathcal{Q}, g)$ [52], i.e., we equip the configuration manifold with a Riemannian metric g. For mechanical systems, a natural choice for the metric is the inertia tensor of the system, which is given in a chart by $\boldsymbol{M}(\boldsymbol{q})$. We use the symbols g_I and g_M to denote the metrics, which are in coordinates given by $\boldsymbol{I}$ and $\boldsymbol{M}(\boldsymbol{q})$, respectively. Likewise, the metric g_A denotes a metric that can be any Riemannian metric, including g_I and g_M.[8]

In the previous sections, we considered Euclidean configuration space, identifying the covectors with their vector counterparts (w.r.t. the Euclidean metric) and visualizing them accordingly. While this is not very elegant from a differential geometric perspective, we think that the section provides valuable and comprehensive insights for roboticists.

From the differential geometric perspective, the rows of the Jacobian, being derivatives of scalar functions, are coordinate versions of differential one-forms (or covectors). Consider a task space parametrized by task coordinates $\boldsymbol{x} \in \mathcal{M}$ and a forward kinematics map $\boldsymbol{x} = \boldsymbol{h}(\boldsymbol{q})$. The differential one-form associated with each coordinate x_i can be written in a chart using the Jacobian (e.g. [47])

$$\mathrm{d}h_i = \sum_{j=1}^{n} \frac{\partial h_i(\boldsymbol{q})}{\partial q_j}\mathrm{d}q_j = \boldsymbol{J}_{x_i}\mathrm{d}\boldsymbol{q}. \qquad (16)$$

Taking a self-motion manifold $\mathcal{S}_0$ defined by $\boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{x}_0$ for an arbitrary constant $\boldsymbol{x}_0$, and considering a velocity $\boldsymbol{v} \in \mathcal{T}_q\mathcal{S}_0$, the reciprocal relation $\mathrm{d}h_i(\boldsymbol{v}) = 0$ holds. Application of a one-form to a vector is independent of any metric.

Moreover, having defined a metric g_A on $\mathcal{Q}$ naturally associates a vector to any covector. In particular, the gradient $\nabla_A h_i$ is the vector associated with the one-form $\mathrm{d}h_i$. The gradient $\nabla_A h_i$ does not only depend on the function itself but also on the metric. In coordinates, the gradient $\nabla_A h_i$ is obtained by multiplying the partial derivatives by the inverse metric tensor, an operation known as *raising an index* in tensor calculus[9]

$$\nabla_A h_i = \boldsymbol{A}^{-1} \begin{bmatrix} \frac{\partial h_i(\boldsymbol{q})}{\partial q_1} \\ \vdots \\ \frac{\partial h_i(\boldsymbol{q})}{\partial q_m} \end{bmatrix} = \boldsymbol{A}^{-1}\boldsymbol{J}_{x_i}^\mathsf{T}. \qquad (17)$$

An inverse operation associates a covector to a vector by *lowering an index*. In our case this is $\boldsymbol{J}_{x_i}^\mathsf{T} = \boldsymbol{A}\nabla_A h_i$ in a chart.

Therefore, (6) can be interpreted as applying one-forms to gradients. We take the one-forms $\mathrm{d}h_i$ for $i \in \{1..m\}$, and apply them to the gradients $\nabla_A \xi_j$ for $j \in \{1..r\}$ and require them to vanish

$$\mathrm{d}h_i(\nabla_A \xi_j) = 0. \qquad (18)$$

---

[7]Note that this is not a distribution in the probability sense, but a distribution in the differential geometric sense: an assignment of tangent subspaces to every point on the manifold. This may help for an introduction: [50, Ch. 19].

[8]These choices of using the Euclidean and the inertia tensor as metric are not complete. Arbitrary custom metrics can be used here, e.g., the stiffness matrix for stiffness consistency [29].

[9]In (16) and (17) we stick to the convention of representing contravariant vectors as columns and covectors as rows.

Due to symmetry of the metric, we can swap the roles in (18)

$$\mathrm{d}\xi_j(\nabla_A h_i) = 0. \tag{19}$$

Again, the metric does not appear explicitly but is, of course, contained in the definition of the gradient. However, the same condition can be expressed in the form of vector orthogonality as $\nabla_A^\mathsf{T} h_i \boldsymbol{A} \nabla_A \xi_i = 0$ or in the form of covector orthogonality as $\mathrm{d}h_i \boldsymbol{A}^{-1} \mathrm{d}\xi_i^\mathsf{T} = 0$. This should clarify why we talk about orthogonality w.r.t. the metric $\boldsymbol{A}$ in (6), although its inverse appears in the formula. From this perspective, it becomes clear that in Sec. IV-A and Fig. 1, Fig. 9, and Fig. 10 we actually visualize gradients w.r.t. either the metric $\boldsymbol{I}$ or $\boldsymbol{M}(\boldsymbol{q})$.

Gradients and one forms can be defined in a coordinate-independent way. Therefore, (19) can be thought of as a coordinate-free version of our main condition (6).

To conclude under which conditions functions ξ_j satisfying (19) exist, we need to introduce the notion of an exterior differential system (EDS) [53], [54]. An EDS is a tuple $(\mathcal{Q}, \mathcal{I})$ of a manifold and an ideal of k-forms. Integral elements of the EDS are linear subspaces of the tangent space of $\mathcal{Q}$ on which forms in $\mathcal{I}$ vanish [53].

Condition (19) can be reinterpreted using EDS-language: find an EDS $(\mathcal{Q}, \mathcal{I})$ such that the one-forms $\mathrm{d}\xi_j$ in $\mathcal{I}$ are the differentials of functions ξ_j and such that the integral elements are spanned by $\nabla_A h_i$. This is opposite to classical EDS problems, where the ideal of k-forms $\mathcal{I}$ is given and the integral elements are unknowns.

Let Δ_h denote a distribution[7] of vector fields, which is locally spanned by

$$\Delta_h = \mathrm{span}\{\nabla_{g_A} h_1, \ldots, \nabla_{g_A} h_m\}. \tag{20}$$

The distribution Δ_h will generally not be involutive [24].

From the Frobenius theorem for EDSs [53, Thm. 1.3] we know that the integral manifolds of $\mathcal{I}$ form the leaves of a foliation of $\mathcal{Q}$ if and only if the vector fields on which all forms in $\mathcal{I}$ vanish are closed under the Lie bracket and thus involutive. This is generally not the case for our redundancy resolution problem, as the distribution Δ_h is generally not involutive. However, it will be trivially involutive, if one has a one-dimensional task space and a single task coordinate, as a one-dimensional distribution is always involutive. We now take the Frobenius theorem and replace the geometrical vocabulary with their respective robotics terms:

Theorem 1 (Existence of Orthogonal Foliations). *A task-induced foliation admits orthogonal foliations if and only if the distribution formed by the gradients of the task coordinates is involutive.*

Corollary 1. *A one-dimensional task space always admits orthogonal foliations.*

Note that the above statements are coordinate-invariant and depend only on the forward kinematics map. Indeed, taking different coordinate charts on the task manifolds leads to different gradients, but they all span the same space.

Take, for instance, a three-dimensional configuration space and a single task coordinate. Two orthogonal foliations ξ_1 and ξ_2 must be determined in this case. This kind of problem has been studied in some publications, where it is known as *triply orthogonal system* [55] or *triply orthogonal web* [53]. However, these publications ask that the foliations ξ_1 and ξ_2 are also

orthogonal. This is a condition we relax in this work, as one usually cannot expect it to be satisfied.

Theorem 1 provides a necessary and sufficient condition for the task-induced foliation to admit orthogonal foliations. Throughout the paper, we will deal with cases where this condition is satisfied and cases where it is not. In the latter case, we find an approximate solution by relaxing the conditions and optimizing a cost function.

V. LOCAL SOLUTION: PLANE STACKS

Before starting a more rigorous study on generating orthogonal foliations $\boldsymbol{\xi}(\boldsymbol{q})$, we start with a simple baseline version: we use linear, locally orthogonal foliations. In particular, we will approximate each orthogonal coordinate $\xi_j(\boldsymbol{q})$ by a function, which is linear in $\boldsymbol{q}$.

We construct a foliation where the leaves are planes. Consider a configuration $\boldsymbol{q}_0 \in \mathcal{Q}$, which we call the *base configuration for the locally orthogonal foliations* and also consider a hyperplane $E_0 = E(\boldsymbol{q}_0, \boldsymbol{n}_0)$ with support vector $\boldsymbol{q}_0$ and unit normal vector $\boldsymbol{n}_0$. The signed distance of a query point $\boldsymbol{q}$ to the hyperplane can be computed using the Hesse normal form of the plane

$$d(E_0, \boldsymbol{q}) = \boldsymbol{n}_0^\mathsf{T}(\boldsymbol{q} - \boldsymbol{q}_0) \tag{21}$$

This signed distance function can also be interpreted as a coordinate function with a trivial Jacobian being the covector version of $\boldsymbol{n}_0$.

Equation (21) also creates a foliation for which each leaf $E_n = E(\boldsymbol{q}_n, \boldsymbol{n}_0)$ is a hyperplane of constant distance and thus parallel to the base hyperplane E_0. Locally, around $\boldsymbol{q}_0$, the function $\xi_j(\boldsymbol{q}) = d(E_0, \boldsymbol{q})$ provides approximate orthogonal coordinates if the normal vector $\boldsymbol{n}_0$ is chosen appropriately. For each locally orthogonal foliation $\xi_j(\boldsymbol{q})$, we require a unit normal vector $\boldsymbol{n}_j$ such that they are mutually linearly independent. Additionally, we require the normal vectors $\boldsymbol{n}_j$ to span the orthogonal complement of the rows of $\boldsymbol{J}_x(\boldsymbol{q}_0)$ for Euclidean metric or of the columns of $\boldsymbol{M}^{-1}(\boldsymbol{q}_0)\boldsymbol{J}_x^\mathsf{T}(\boldsymbol{q}_0)$ for inertia metric. In other words, we can say that the unit vectors associated with the functions $\xi_j(\boldsymbol{q})$ span the null space of $\boldsymbol{A}^{-1}(\boldsymbol{q}_0)\boldsymbol{J}_x(\boldsymbol{q}_0)$.

The entirety of locally orthogonal coordinates can be written in compact form

$$\boldsymbol{\xi}(\boldsymbol{q}) = \underbrace{\begin{bmatrix} \boldsymbol{n}_1 & \ldots & \boldsymbol{n}_r \end{bmatrix}^\mathsf{T}}_{\boldsymbol{J}_\xi}(\boldsymbol{q} - \boldsymbol{q}_0). \tag{22}$$

The Jacobian $\boldsymbol{J}_\xi$ of $\boldsymbol{\xi}(\boldsymbol{q})$ is constant. Based on this construction, we ensure that (6) is satisfied at $\boldsymbol{q}_0$: at $\boldsymbol{q}_0$, the orthogonal coordinates $\boldsymbol{\xi}(\boldsymbol{q})$ are dynamically decoupled from the task. The performance on the regions around $\boldsymbol{q}_0$ will depend on the curvature of the self-motion manifolds and the metric. For configurations far from the base configuration, the simple local solution by plane stacks (22) cannot be expected to perform well — let's move to more elaborate and global solutions.

VI. EXACT SOLUTION: COORDINATE GROWING

In this section, we show how an orthogonal foliation can be numerically generated for the case of one-dimensional task space. To perform the numerical propagation of the orthogonal foliation coordinates, we start by selecting one value for the

(a) In $q_1 q_2$-plane

(b) On the 2-torus
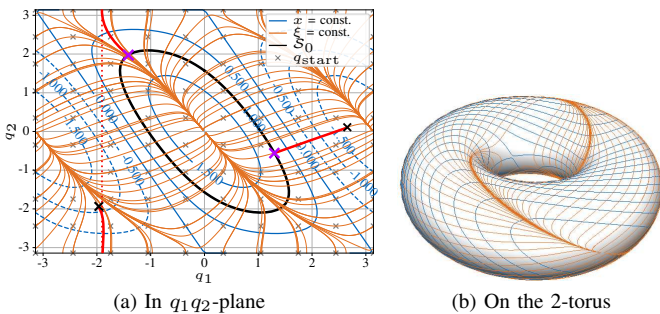
Fig. 12. Following the gradient flow of the x-coordinates for Euclidean metric.

task-coordinate function and call it x_0. Then, we compute the corresponding self-motion manifold $\mathcal{S}_0$ such that

$$\forall \boldsymbol{q} \in \mathcal{S}_0 : \boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{x}_0. \tag{23}$$

We call $\mathcal{S}_0$ the *base manifold*. Note that $\dim \mathcal{S}_0 = n-1$. Then, we define a chart $\boldsymbol{\xi}_0 : \mathcal{S}_0 \rightarrow \mathbb{R}^{n-1}$ on the base manifold.

We determine the coordinates $\boldsymbol{\xi}(\boldsymbol{q}_1)$ of a point $\boldsymbol{q}_1 \in \mathcal{S}_1$ on another leaf $\mathcal{S}_1$ by taking the differential equation

$$\dot{\boldsymbol{q}} = (\boldsymbol{x}_0 - \boldsymbol{h}(\boldsymbol{q})) \nabla_{\boldsymbol{A}} \boldsymbol{h}(\boldsymbol{q}) \tag{24}$$

with initial condition $\boldsymbol{q}(0) = \boldsymbol{q}_1$. The flow (24) will converge onto the base manifold at a point $\hat{\boldsymbol{q}} \in \mathcal{S}_0$. We then set $\boldsymbol{\xi}(\boldsymbol{q}_1) = \boldsymbol{\xi}_0(\hat{\boldsymbol{q}})$ to obtain coordinates of $\boldsymbol{q}_1 \in \mathcal{S}_1$ in the propagated chart.

In the following sections, we investigate the resulting coordinates for the two example robots. We show results for Euclidean (kinematically decoupled) and inertia tensor (dynamically decoupled) as metric.

The point of convergence of the gradient flow (24) is numerically evaluated using a Runge-Kutta integration scheme [56]. We stop the integration when the change of $\boldsymbol{q}$ between two integrator steps falls below a threshold of 10^{-6}.

A. In Two-Dimensional Configuration Space

Take the example manipulator with two degrees of freedom (Fig.5a) and select the x-component of the end-effector position as task coordinate.

1) Euclidean Metric ($\boldsymbol{A} = \boldsymbol{I}$): We begin with the choice $\boldsymbol{A} = \boldsymbol{I}$, i.e., we choose the standard Euclidean metric on $\mathcal{Q}$. Fig. 12a shows isolines of the forward kinematics in blue.

Take a grid of points in the configuration space denoted by the gray crosses. We take the gradient flow $\pm \nabla_{\boldsymbol{I}} \boldsymbol{h}(\boldsymbol{q})$ and integrate it into both directions for each point of the grid. The orange lines in Fig. 12a show the resulting integral curves. Dependent on the sign, the flow converges to one of the singularities. Note the toroidal topology of the configuration space. Especially, the points of convergence on both sides of the plot are the same point on the torus (Fig. 12b)! The entirety of integral curves *foliates* the configuration space $\mathcal{Q}$.

Pick a base manifold $\mathcal{S}_0$ such that $x_0 = 1$ (black curve in Fig. 12a). The base manifold is homeomorphic to the 1-sphere $\mathbb{S}^1$: we cannot find a global diffeomorphism onto the real numbers $\xi_0 : \mathcal{S}_0 \rightarrow \mathbb{R}$. Instead, we can find a smooth function $\hat{\xi}_0 : \mathcal{S}_0 \rightarrow \mathcal{SO}(2)$. The diffeomorphism $\hat{\xi}_0$ must be used if the entire base manifold $\mathcal{S}_0$ shall be covered. We apply a simplification here and map the entire manifold $\mathcal{S}_0$ but one point to $(-\pi, \pi]$ to stay within a single coordinate chart. The drawback is that ξ_0 fails at $\pm\pi$, and we cannot pass through that point.

Finally, we evaluate the resulting coordinate function $\xi : \mathcal{Q} \rightarrow (-\pi, \pi]$ by pulling a point $\boldsymbol{q} \in \mathcal{Q}$ onto the base manifold

(a) Euclidean ($\boldsymbol{A} = \boldsymbol{I}$)

(b) Inertia weighted ($\boldsymbol{A} = \boldsymbol{M}$)
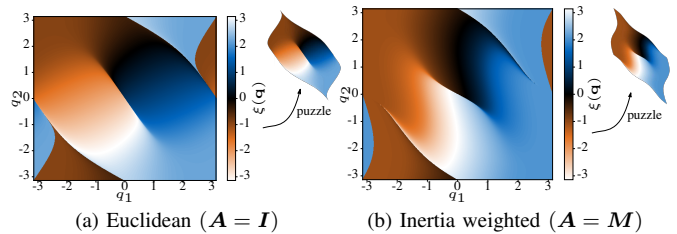
Fig. 13. Orthogonal coordinate function $\xi(\boldsymbol{q})$ for Euclidean metric (left) and non-Euclidean metric (right). The color encodes the value of the function $\xi(\boldsymbol{q})$. Sketches on the top-right: most discontinuities can be pushed to the boundary when considering the toroidal structure.

(a) In $q_1 q_2$-plane

(b) On the 2-torus
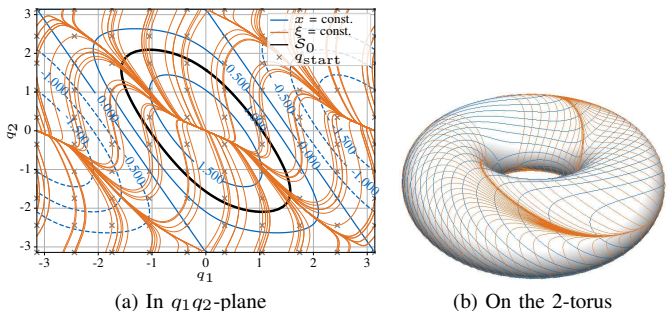
Fig. 14. Following the gradient flow of the x-coordinate for inertia tensor as metric ($\boldsymbol{A} = \boldsymbol{M}$).

(24) and evaluating ξ_0 at that point. This procedure is visualized for two example configurations in Fig. 12. When starting from arbitrary joint configurations (black crosses), the flow will trace out a path (red lines) and converge at a configuration on the base manifold (violet crosses). Fig. 13a shows the function color-coded.

The function is smooth in large regions of Fig. 13a. However, there are also edges of jumps in the color rendering of ξ. Consider the edge at the origin $\boldsymbol{q} = [0, 0]^\mathsf{T}$ of the plot. The function ξ becomes increasingly steep when moving on this edge towards the center and discontinuous directly at the origin. Considering the toroidal structure of the space and solving the puzzle (top-right corner of Fig. 13a) shows that the discontinuous spot coincides with a task-space singularity. As before, we assume we never reach singular configurations and forget about the singularity at the center for now. Within the white region of Fig. 13a is another discontinuous spot, an artifact of the mapping of $\mathbb{S}^1$ onto $\mathbb{R}$.

2) Mass Metric ($\boldsymbol{A} = \boldsymbol{M}$): We choose the same chart ξ_0 on $\mathcal{S}_0$: the value of ξ on $\mathcal{S}_0$ does not change at all! What does change are the values of points in $\mathcal{Q} \setminus \mathcal{S}_0$. Compared to the Euclidean case, the coordinates on $\mathcal{S}_0$ grow in different directions as the gradient in (24) depends on the metric.

Fig. 14 shows integral curves of $\pm \nabla_{\boldsymbol{M}} \boldsymbol{h}(\boldsymbol{q})$ (orange), as well as the contour lines of the task coordinate (blue) and grid of start points $\boldsymbol{q}_0$ (gray crosses) for the integration and Fig. 13b shows the coordinate function color-coded.

B. In Three-Dimensional Configuration Space

Next, we consider $n = 3$ degrees of freedom and choose the x-component of the end-effector pose as task coordinate. Hence, we obtain a $r = 2$-dimensional task-induced foliation of the configuration space.

We again select a base manifold $\mathcal{S}_0$ and define a coordinate chart on it (cmp. Fig. 1). Parametrization of a surface embedded

in three-dimensional space is a well-studied problem within the field of computer graphics [57], where this is required for, among others, texture mapping. We consider the simplest case of parametrization: a surface of disk-topology. This does not involve the need for cuts of the surface for parametrization.

We select $x_0 = 0$ and constrain $q_1 \geqslant 0$, i.e., we consider a subregion of the configuration space $\mathcal{Q}$, and the simplified model generated here will only be valid in this subregion. By this constraint, we achieve a disk topology of $\mathcal{S}_0$. Further, we apply marching cubes [51] and obtain a triangle mesh approximating the base manifold $\mathcal{S}_0$ (Fig. 15a). Because of the mesh structure, we can use discrete differential geometry to rewrite the continuous constraints for Φ in a discrete fashion, and the condition boils down to solving a linear system of equations as summarized in [57]. Boundary conditions are required for the linear system, which we generate by mapping the boundary of the base manifold to the unit circle. Afterward, we solve for the interior vertices and obtain a discrete version of a bijective mapping between the base manifold and the unit disk in the plane (Fig. 15b). Fig. 15c additionally shows isolines of the parameter space on the base manifold. The visualizations are for a coarse triangle mesh resolution to visualize the mesh structure in the parametrization domain.

Following the above procedure, we obtain a discrete version of $\boldsymbol{\xi}_0(\boldsymbol{q})$ by having a unique value $\boldsymbol{\xi}_0(\boldsymbol{q})$ for every mesh vertex. For every other point $\boldsymbol{q} \in \mathcal{S}_0$, not coinciding with a vertex of the mesh, we query a KDTree [58] generated from the mesh to find the face containing $\boldsymbol{q}$ (approximately) and interpolate between the values for $\boldsymbol{\xi}_0$ of the three vertices of the face. Then, we follow the gradient flow (24) to pull any configuration $\boldsymbol{q} \in \mathcal{Q}$ onto the base manifold $\hat{\boldsymbol{q}} \in \mathcal{S}_0$ and interpolate $\boldsymbol{\xi}_0(\hat{\boldsymbol{q}})$ as stated above. We have achieved a coordinate function $\boldsymbol{\xi} : \bar{\mathcal{Q}} \to \mathbb{R}^2$.

Fig. 15d shows the result for $\boldsymbol{A} = \boldsymbol{I}$. We show three self-motion manifolds and the isolines of the coordinate functions ξ_1 and ξ_2. The dark blue manifold in the middle of the stack is the base manifold $\mathcal{S}$ at $x_0 = 0$, while the other manifolds (gray and light blue) correspond to $x = \pm 0.3$.

VII. OPTIMIZATION SOLUTION: NEURAL NETWORK

The previous section showed how a coordinate function defined on one of the self-motion manifolds could be propagated to other points in the configuration space in the case of only one task coordinate. This approach has three drawbacks: (1) the algorithm is rather expensive in execution time, (2) there is no easy way to obtain a Jacobian $\boldsymbol{J}_\xi$ besides the obvious numerical Jacobian, and (3) it only works for one-dimensional tasks. For a higher dimensional task space, there are multiple gradient flows (24), one for each row of the Jacobian. Therefore, there are multiple directions in which the coordinates of the base manifold would need to be propagated. For the same approach to work for higher dimensional task spaces, all the flows (24) starting from a point $\boldsymbol{q}_1 \in \mathcal{S}_1$ would need to end at the same point in $\boldsymbol{q}_0 \in \mathcal{S}_0$. This is, in general, not possible due to non-involutivity.

We relax the orthogonality condition to *as orthogonal as possible*, use a smooth parametric function $\boldsymbol{\xi}_\theta(\boldsymbol{q})$, and optimize the parameters $\boldsymbol{\theta}$ such that the resulting foliation is *quasi-orthogonal*. Another benefit of this approach is that it allows for analytical Jacobians.

Neural networks provide a flexible and universal function approximator. By formulating partial or ordinary differential

(a) Base Manifold   (b) Parametrization

(c) Parametrization on Surface   (d) Propagation to other surfaces
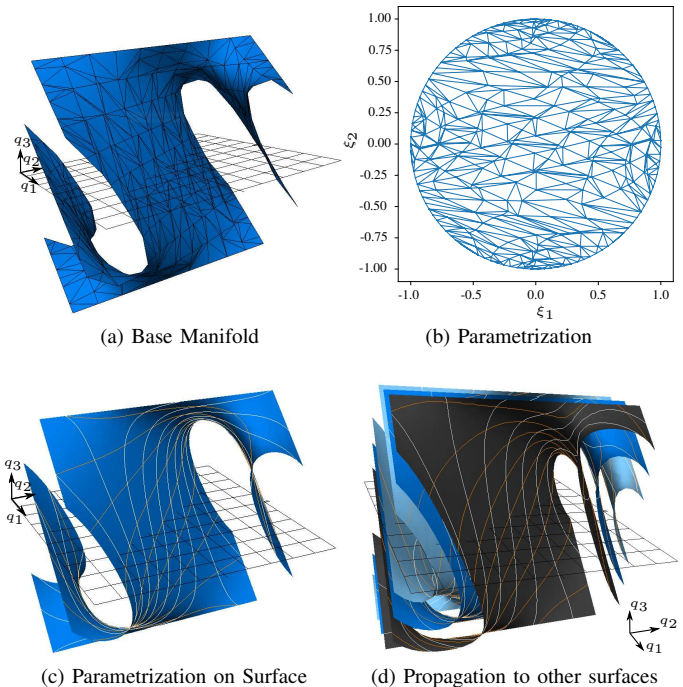
Fig. 15. Coordinate growing from a base manifold $\mathcal{S}_0$. **(a)** Triangulation of the base manifold shown for a coarse resolution. **(b)** Parametrization of the base manifold onto the unit circle. The edges do not intersect, and we obtain a bijection between the unit circle and the base manifold. **(c)** Parametrization shown on the base manifold using isolines of the parametrization. **(d)** Propagation to other leaves of the foliation.

equations as loss functions, machine learning techniques can be used to find solutions [59], [60]. We write the parametric function $\boldsymbol{\xi}_\theta(\boldsymbol{q})$ as neural network with n input and r output neurons. In particular, we use a network with two hidden layers and $\tanh$ activation functions

$$\boldsymbol{\xi}_\theta(\boldsymbol{q}) = \boldsymbol{W}_{\text{out}} \tanh\left[\boldsymbol{W}_2 \tanh\left(\boldsymbol{W}_1 {\textstyle\sin\atop\cos}(\boldsymbol{q}) + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right] + \boldsymbol{b}_{\text{out}},$$

where $\tanh$ acts componentwise and $\boldsymbol{\theta}$ is the collection of all elements of $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$. The function ${\sin\atop\cos} : \mathbb{R}^n \to \mathbb{R}^{2n}$ stacks the sines and cosines of the elements of the input vector. Using this transformation, we implement the toroidal structure of the input space.

Within this paper, we show results for robots with revolute joints only. However, the framework and the concepts are not restricted to purely revolute robots. The methods can also be used for robots having prismatic joints. In that case, the $\sin$ and $\cos$ terms will be omitted for prismatic joints.

Differentiating the network output with respect to its input yields the Jacobian $\boldsymbol{J}_{\xi,\theta}(\boldsymbol{q}) = \frac{\partial \boldsymbol{\xi}_\theta}{\partial \boldsymbol{q}}$. The expression $\frac{\partial \boldsymbol{J}_{\xi,\theta}(\boldsymbol{q})}{\partial \boldsymbol{\theta}}$ describes the effect of the parameters $\boldsymbol{\theta}$ on the input-output Jacobian of the network. All the network parameters $\boldsymbol{\theta}$ except the output layer's biases affect the Jacobian $\boldsymbol{J}_{\xi,\theta}(\boldsymbol{q})$.

Condition (6) is used to derive a scalar cost function that penalizes the cosine of the pairs $\nabla_{\boldsymbol{A}} h_i$ and $\mathrm{d}\xi_{\theta,j}$. We write this in matrix-vector notation and introduce $\boldsymbol{v}_{x,i}^\mathsf{T}(\boldsymbol{q})$ being the

i-th row in $\boldsymbol{J}_x(\boldsymbol{q})$ and $\boldsymbol{v}_{\xi,j}^{\mathsf{T}}(\boldsymbol{q})$ being the *j*-th row of $\boldsymbol{J}_{\xi,\theta}(\boldsymbol{q})$:

$$L(\boldsymbol{q}, \boldsymbol{\theta}) = \frac{\lambda_1}{2mr} \sum_{i=1}^{m} \sum_{j=1}^{r} \left(\frac{\boldsymbol{v}_{x,i}^{\mathsf{T}} \boldsymbol{A}^{-1} \boldsymbol{v}_{\xi,j}}{|\boldsymbol{A}^{-1}\boldsymbol{v}_{x,i}| \cdot |\boldsymbol{v}_{\xi,j}|} \right)^2 \quad (25)$$

$$+ \frac{\lambda_2}{2\binom{r}{2}} \sum_{i=1}^{r} \sum_{j=1}^{r} \left[\left(\frac{\boldsymbol{v}_{\xi,i}^{\mathsf{T}} \boldsymbol{A}^{-1} \boldsymbol{v}_{\xi,j}}{|\boldsymbol{A}^{-1}\boldsymbol{v}_{\xi,i}| \cdot |\boldsymbol{v}_{\xi,j}|} \right)^2 [i > j] \right],$$

where $[b]$ is the Iverson bracket, which is 1 when the b is true and 0 otherwise, and $\binom{n}{k}$ the binomial coefficient.

The first part of the cost function penalizes the square of the cosine between mutual rows of $\boldsymbol{J}_x\boldsymbol{A}^{-1}$ and $\boldsymbol{J}_{\xi,\theta}$, while the second part penalizes the cosine between two mutual rows of $\boldsymbol{J}_{\xi,\theta}\boldsymbol{A}^{-1}$ and $\boldsymbol{J}_{\xi,\theta}$. Those two terms can be weighted using the constants λ_1 and λ_2. The second part of the cost function resembles the full-rank condition. By using the cosines of the angles between the gradients, we ensure that the magnitude of the gradients is not penalized. This is required because the globally optimal solution would be $\boldsymbol{\xi}(\boldsymbol{q}) = \text{const}$ otherwise.

We chose $\lambda_1 = 1000$ and $\lambda_2 = 1$, i.e., we assign much higher priority to the decoupling between task- and orthogonal coordinates than mutual decoupling between the orthogonal coordinates. Generally, when not allowing constant functions for $\xi_i(\boldsymbol{q})$, (25) cannot be optimized to zero. Consider again the 3 DoF manipulator with 1 DoF task space: optimizing (25) to zero would imply that we have found a triply orthogonal system. This, is only possible in exceptional cases for specific task foliations [53], [55]. Therefore, optimizing (25) will find a trade-off between the two terms.

Suppose we have K samples $\boldsymbol{q}_k$ of the selected subset of $\mathcal{Q}$. The total cost will be

$$L(\boldsymbol{\theta}) = \frac{1}{K} \sum_{k=1}^{K} L(\boldsymbol{q}_k, \boldsymbol{\theta}). \quad (26)$$

We sample the training data on the selected regions of the joint space $\tilde{\mathcal{Q}}$.

Finally, we perform gradient descent on $L(\boldsymbol{\theta})$. By construction the function $\boldsymbol{\xi}_\theta$ is always an integral of $\boldsymbol{J}_{\xi,\theta}$, so we get the coordinate function $\boldsymbol{\xi}_\theta(\boldsymbol{q})$ for free when fitting the Jacobian $\boldsymbol{J}_{\xi,\theta}(\boldsymbol{q})$ to the condition (6). The optimization simultaneously finds a set of vector fields as well as an integral of them.

The solution $\boldsymbol{\xi}_\theta(\boldsymbol{q})$ to (6) is not unique and will depend on the initialization of the parameters. After initialization, we optimize on the cost function (26). This can be interpreted as bending and warping the randomly initialized foliation.

The training procedure is split into multiple epochs. We sample new training data in each epoch using a uniform distribution on $\mathcal{Q}$ to prevent overfitting. Some further details on the training procedure can be found in [61]. For the implementation, we use TensorFlow [62], which allows for evaluating analytical gradients and Jacobians and generating fast code running on the GPU. We denote the optimized parameters $\hat{\boldsymbol{\theta}}$.

A. Two Degrees of Freedom Example Revisited

Let us revisit the two degrees of freedom example from Sec. VI-A. The planar configuration space allows visualizing the results straightforwardly. Globally, no smooth function $\boldsymbol{\xi}(\boldsymbol{q})$ creates a non-singular parametrization of the task-induced foliation everywhere because it involves parametrizing a 1-sphere with a single coordinate. Still, we train the model
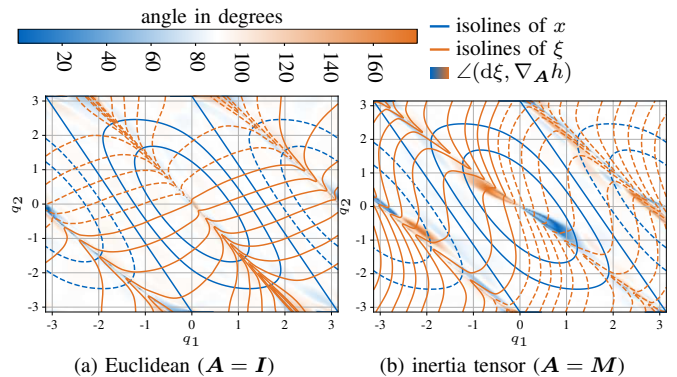
Fig. 16. Resulting contour lines of the trained neural network for different metrics. The lines show isolines of the task coordinate x (blue) and orthogonal coordinate (orange). Solid lines refer to positive values and dashed lines show negative values. The background shows the local training accuracy.
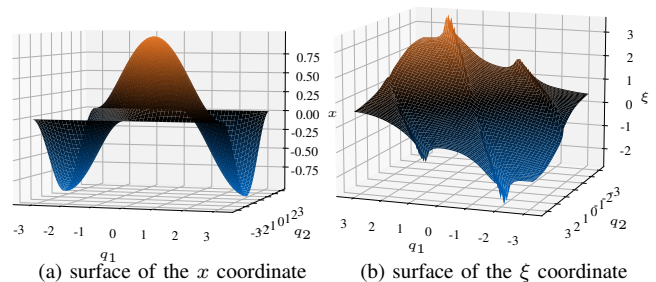
Fig. 17. Surface plots of the the task coordinate function x (left) and the neural network approximation of a orthogonal coordinate ξ (right) for a two degrees of freedom manipulator for the Euclidean case ($\boldsymbol{A} = \boldsymbol{I}$).

globally, i.e. we sample training data from the entire $\mathcal{Q}$ and check how the network handles this issue.

We first fit the neural network using $\boldsymbol{A} = \boldsymbol{I}$ and evaluate isolines (orange) of the resulting coordinate function $\xi_{\hat{\theta}}(\boldsymbol{q})$ in Fig. 16a. Isolines of the forward kinematics function $h(\boldsymbol{q})$ are shown in blue for reference. The background color of the plot encodes the angle between $\nabla_{\boldsymbol{A}}h$ and $d\xi$. If the background was white everywhere, the training would be optimal. However, as expected, there are certain regions where the model does not perform well visualized by the colored regions in Fig. 16. We observe that each value exists twice on each self-motion manifold, as each isoline of $\xi(\boldsymbol{q})$ intersects each self-motion manifold twice. As expected we did not obtain a global coordinate function on the self-motion manifolds. Fig. 17b additionally shows the functions $h(\boldsymbol{q})$ and $\xi(\boldsymbol{q})$ as surface plot.

Fig. 16b shows the results for $\boldsymbol{A} = \boldsymbol{M}$. Similarly to the Euclidean case, the model performs well in large regions (white background color), and the function only uniquely parametrizes half of the self-motion manifolds.

Compare the results obtained from the trained coordinate function and the numerical results from Sec. VI-A. Most importantly, we observe that the isolines of the trained function match the lines generated by the flow $\nabla_{\boldsymbol{A}}h$ to a large extent. However, the similarity worsens when reaching the attracting lines growing out of the singular configurations. The trained coordinate function $\xi_{\hat{\theta}}$ is, in contrast to the numerical results, smooth. Therefore, $\xi_{\hat{\theta}}$ can never be a global chart of the self-motion manifolds, which we also observe in the results.
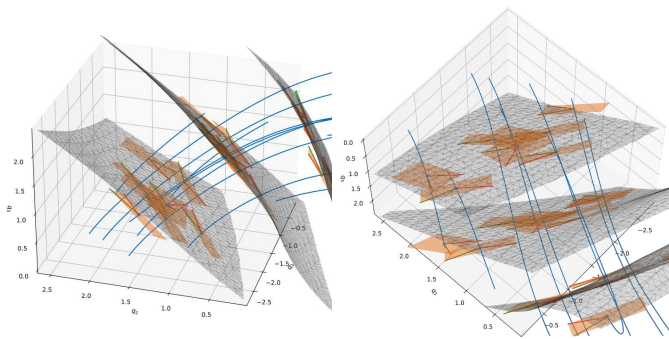
Fig. 18. Results for a subset of the joint space of a three DoF robot with two task coordinates. The blue lines show leaves of the task-induced foliation (self-motion manifolds) and the gray surfaces correspond to isosurfaces of the coordinate function the neural net found.
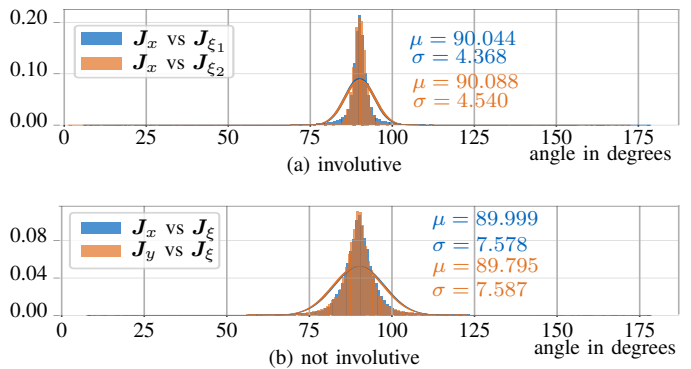

Fig. 19. Histogram of angles between gradients of the task- and orthogonal coordinates for randomly sampled configurations for a trained neural network. Top: involutive example, bottom: non-involutive example. The symbol μ denotes the mean and σ the standard deviation.

B. Non-Involutive Case

In the last section, we considered an example having an exact solution; now, a model with no exact solution, i.e., the distribution of gradients of the task coordinates, is non-involutive. The numerical propagation of coordinates according to Sec. VI fails, as the gradients of the task coordinates cannot be integrated to submanifolds of $\mathcal{Q}$.

Still, we can fit coordinate functions using the cost function (26) and aim at finding a ξ_θ which is *as orthogonal as possible*.

We consider again the manipulator with three degrees of freedom (Fig. 5b) and select both, the x- and y-component of the end effector position as task coordinates. Therefore, we obtain a non-involutive system and train a neural network model using the cost function (26).

The results are shown in Fig. 18 for a section of the configuration space. The blue lines show some leaves of the task-induced foliations, i.e., self-motion manifolds. Further, the red lines show $\nabla_I h_1$, and the green lines show $\nabla_I h_2$ scaled to unit length at that point. Each pair of $\nabla_I h_1$ and $\nabla_I h_2$ spans a plane visualized by the red rectangles. The planes belong to the distribution which we aim to integrate - an integral foliation for the distribution would have those planes as tangent planes. Finally, the gray surfaces correspond to isosurfaces of the foliation obtained by the learning process. These are surfaces where $\boldsymbol{\xi}(\boldsymbol{q})$ is constant. We observe that, despite the distribution not being involutive, the red planes are to a good approximation tangent to the isosurfaces.

To quantify the effect of the non-involutivity, we also train a neural model for an involutive case. This is achieved by removing the y-component of the end-effector position from the task coordinates. We uniformly sample $N = 10^5$ joint configurations and compute the angle between the gradients. Fig. 19a shows the histogram of the residual angles between $d\xi_j$ and $\nabla_M h_i$. The results can be compared to the histogram in Fig. 19b, which shows the results for the non-involutive case. The histogram is, as expected, much more needle-shaped in Fig. 19a than in Fig. 19b, and the standard deviations of the angles are lower for the involutive case.

VIII. SIMULATION

We use the 3 DoF example manipulator (Fig. 5b) for experimental validation. For the dynamics, we assume a uniformly distributed mass of $m = 3$ for each link.

To employ the controller (37) a desired trajectory in terms of $(\boldsymbol{\varphi}_d, \dot{\boldsymbol{\varphi}}_d, \ddot{\boldsymbol{\varphi}}_d)$ is required. We use a signal of pre-defined jumps

$\hat{\boldsymbol{\varphi}}(t)$ on which we apply a second-order low pass with natural frequency $\omega_0 = 2\pi$ and damping ratio $\zeta = 0.7$. This way, we obtain a twice-differentiable signal that can be used as the desired trajectory. In the following, we show simulation results for two different settings. First, we consider an involutive case by taking, as before, only the x-component of the end-effector as task coordinate. Afterward, we take a non-involutive system using the x- and y-component as task coordinates.

In both cases, we show results using the locally orthogonal foliations (Sec. V) and the neural network approximation (Sec. VII). We simulate the manipulator for every setting using the trajectory tracking controller (37). For the neural network models, we additionally apply pure impedance control (30) in dedicated experiments.

A. State-of-the-Art: Projection-based Reference Controller

As a state-of-the-art reference controller for redundancy resolution in combination with impedance control, we use the operational space framework [1]. In this framework, control torques stemming from secondary tasks are projected into the null space of the main task.

We choose the primary task coordinates x, y describing the position of the end-effector. Therefore, the forward kinematics function is $\boldsymbol{h}(\boldsymbol{q}) = [x, y]^\mathsf{T}$ and the Jacobian is $\boldsymbol{J}_x \in \mathbb{R}^{2 \times n}$. As secondary task we choose joint targets $\boldsymbol{q}_d$ and use a joint impedance controller.

$$
\begin{aligned}
\boldsymbol{\tau}_P = {} & \boldsymbol{J}_x^\mathsf{T} \left(-\boldsymbol{K}_x \left[\boldsymbol{h}(\boldsymbol{q}) - \boldsymbol{x}_d \right] - \boldsymbol{D}_x \boldsymbol{J}_x \dot{\boldsymbol{q}} \right) \\
& + \boldsymbol{P}_x^\mathsf{T} \left(-\boldsymbol{K}_q \left[\boldsymbol{q} - \boldsymbol{q}_d \right] - \boldsymbol{D}_q \dot{\boldsymbol{q}} \right) \\
& + \boldsymbol{g}(\boldsymbol{q}),
\end{aligned} \tag{27}
$$

where $\boldsymbol{P}_x = [\boldsymbol{I} - \boldsymbol{J}_x^{\#M} \boldsymbol{J}_x^\mathsf{T}]$ is a dynamically consistent [29] torque projection matrix.

Consider a trajectory in Cartesian positions $\boldsymbol{x}_d(t)$ and joint targets $\boldsymbol{q}_d(t)$. The desired values for $\boldsymbol{q}_d$ are shown as dashed lines in the top pane of Fig. 20 and the desired Cartesian positions as dashed lines in the second pane. We apply the control law (27) to the simulated robot and show the results by solid lines. In the bottom pane, we show the potential energies stored in the virtual springs by the task impedance controller $U_x = \frac{1}{2} \left(\boldsymbol{h}(\boldsymbol{q}) - \boldsymbol{x}_d \right)^\mathsf{T} \boldsymbol{K}_x \left(\boldsymbol{h}(\boldsymbol{q}) - \boldsymbol{x}_d \right)$ and the joint impedance controller $U_q = \frac{1}{2} \left(\boldsymbol{q} - \boldsymbol{q}_d \right)^\mathsf{T} \boldsymbol{K}_q \left(\boldsymbol{q} - \boldsymbol{q}_d \right)$.

At every jump in $\boldsymbol{q}_d$ at $t \in \{4, 8, 12, 14, 18\}$ we ensure that the new target configuration has the same forward kinematics as the current configuration, i.e., they are on the same self-motion
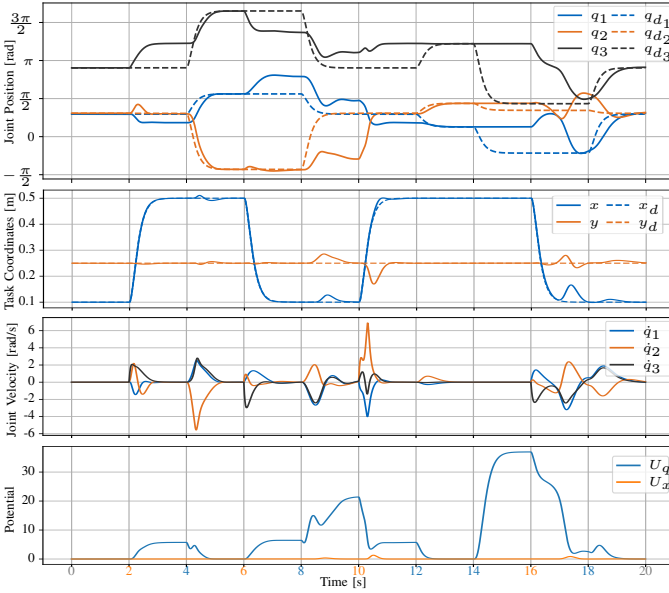
Fig. 20. Projection-based operational space control for comparison. The secondary tasks are desired joint configurations. Dashed lines: desired values and solid lines: true values. Third pane: joint velocities. Bottom pane: energy stored in the virtual springs by the primary and secondary task impedance controller. *Highly recommended for understanding: the video in the supplementary material.*

manifold. Therefore, as we never switch Cartesian target x_d and joint target q_d simultaneously, we design the joint target q_d to be always reachable when switching to it. This is not true when switching the Cartesian target x_d at $t \in \{2, 6, 10, 16\}$: the joint target q_d is generally not reachable when jumping the Cartesian target x_d.

At $t = 2$, we command a jump on the x-coordinate of the primary task. The targe is incompatible with the secondary task, and we observe that the secondary task q_d is unreachable. This results in a non-zero potential U_q. However, as the torques from the secondary task are projected out, no torques are acting on the joints. There is a tensioned virtual spring, which has no effect! At $t = 4$, we choose a reachable joint configuration q_d for the new x_d and observe that the potential energy U_q converges to zero. Once we command the x-coordinate back to the initial value at $t = 6$, the potential energy U_q rises again. At the end of the time interval $t \in [8, 10]$, all commanded values, both for the primary and secondary task, reach the same values as in the interval $[0, 2]$. They have been chosen to be compatible, as can be seen from the initial configuration. However, the potential rises further: the robot does not return to the initial configuration, even though the desired joint configuration is reachable! The intermediate robot motions have driven it to another configuration, which results in convergence to another local minimum of the potential energy U_q.

We repeat the experiment from the new initial configuration at $t = 10$. At $t = 12$ and $t = 14$, we again command jumps in q_d to feasible values. At $t = 12$, the robot tracks the desired joint configuration q_d well. However, at $t = 14$, something interesting happens: the jump in q_d is entirely in the null space of the primary task. Nothing happens; and the robot does not move at all since no torques survive the projector. Still, a highly tensioned virtual spring hides behind the projector. When interacting with the robot, this may lead to unexpected behavior. For different configurations, the stability of the local

minima of the potential energy U_q through the projector may change, and the robot may snap to the global minimum. At $t = 16$, we command a jump in x_d again. This time the robot moves a lot compared to the comparable jump $t = 6$. At some point, the local minimum of the projected potential is no longer stable, and the robot approaches the global minimum.

After showcasing the problems P1-P6 of a classical approach, let us move to global redundancy resolution coordinates on position level — as proposed in this paper.

B. Projection-Free Controllers

To evaluate the applicability of the new orthogonal foliations in simulation, we state well-known control approaches, now using orthogonal coordinates, and show what the condition (6) implies for the closed-loop dynamics.

The standard rigid multi-body dynamics equation is used to model the manipulators

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau. \tag{28}$$

1) Impedance Controller: As in Sec. IV-B we stack the task coordinates $x = h(q)$ and the (quasi-)orthogonal coordinates $\xi(q)$ and obtain

$$\varphi(q) = \begin{bmatrix} h(q) \\ \xi(q) \end{bmatrix} \tag{29}$$

with the associated Jacobian (8) $\dot{\varphi} = J(q)\dot{q}$. We use an impedance control law [63]

$$f_\varphi^{\text{imp}} = -K_\varphi[\varphi(q) - \varphi_d] - D_\varphi J \dot{q}, \tag{30}$$

where φ_d denotes the desired position, K_φ the desired stiffness, and D_φ the damping matrix. Given a diagonal matrix $Z = \text{diag}(\zeta_1 \ldots \zeta_m)$ of damping ratios, the damping matrix is computed using the damping design equation [64]

$$D_\varphi = M_\varphi^{1/2} Z K_\varphi^{1/2} + K_\varphi^{1/2} Z M_\varphi^{1/2}, \tag{31}$$

where $A^{1/2}$ denotes the matrix-square root of A such that $A = A^{1/2} A^{1/2}$. The matrix M_φ denotes the transformed mass matrix

$$M_\varphi = \left(J M^{-1} J^\mathsf{T} \right)^{-1}. \tag{32}$$

Suppose J_x and J_ξ are orthogonal with respect to the mass matrix, i.e., we have condition (6) for $A = M$ satisfied. One can verify that the term in the parenthesis of (32) will have block-diagonal structure (cf. [2])

$$M_\varphi^{-1} = J M^{-1} J^\mathsf{T} \overset{(6)}{=} \begin{bmatrix} M_x^{-1} & 0 \\ 0 & M_\xi^{-1} \end{bmatrix}. \tag{33}$$

Note that we notationally dropped the explicit dependency of J, M, D_φ and M_φ on q to remove some clutter.

Using the transformation law for covectors we transform the resulting force from the impedance controller f_φ^{imp} to joint torques

$$\tau_\varphi^{\text{imp}} = J^\mathsf{T} f_\varphi^{\text{imp}} \tag{34}$$

Suppose we apply the feedback control law

$$\tau_1 = J^\mathsf{T} f_\varphi^{\text{imp}} + g(q) \tag{35}$$

on the multi-body dynamics (28) and let us solve for the acceleration $\ddot{\varphi}$

$$\ddot{\varphi} = \underbrace{J M^{-1} J^\mathsf{T}}_{M_\varphi^{-1}} f_\varphi^{\text{imp}} + \underbrace{\dot{J}\dot{q}}_{\ddot{\varphi}_{\text{curv}}} - \underbrace{J M^{-1} C \dot{q}}_{\ddot{\varphi}_{\text{CC}}}. \tag{36}$$

The force due to the impedance controller f_φ^{imp} results in an acceleration $\ddot{\varphi}$ via the block-diagonal mobility matrix

M_φ^{-1} (33). Therefore, the impedance controller will not cause immediate accelerations in the task coordinates upon forces in the orthogonal coordinates and vice versa.

2) Trajectory Tracking: The impedance controller will not cause immediate disturbing accelerations between the mutual sets of coordinates. It may, however, cause delayed couplings via the accumulated velocity $\dot{q}$. The joint velocity couples into the acceleration $\ddot{\varphi}$ by the last to terms in (36): $\ddot{\varphi}_{CC}$ expressing the task acceleration due to Coriolis and centrifugal forces; and $\ddot{\varphi}_{curv}$ depending on the curvature of the self-motion manifolds.

Consequently, to analyze the inertial decoupling of the coordinates we need additional feed-forward terms. Assume a desired trajectory $(\varphi_d(t), \dot{\varphi}_d(t), \ddot{\varphi}_d(t))$ for the task and the orthogonal coordinates. We then use the feed-forward controller

$$f_\varphi^{FF} = M_\varphi \left[\ddot{\varphi}_d + \left(JM^{-1}C - \dot{J} \right) J^{-1}\dot{\varphi}_d \right] + D_\varphi \dot{\varphi}_d.$$

The resulting force is then transformed to joint torques using the covector transformation (34). The final control law is composed of the impedance controller (30) and the feed-forward terms

$$\tau_2 = J^\mathsf{T} \left[f_\varphi^{imp} + f_\varphi^{FF} \right] + g(q). \tag{37}$$

In (37) all the matrices are evaluated at the actual joint configuration. The control law is known as *operational space PD+ controller*.

Closed-loop stability of the system can be shown similarly to [65, Sec. 3.2], where passivity and global asymptotic stability is shown for a joint space PD+ controller by deriving a storage function in joint coordinates. When writing the storage function in operational coordinates φ, the same reasoning as in [65] applies for the closed-loop system using (37). However, due to singularities in the coordinates, globality is lost, and only local results are feasible.

C. Orthogonal Coordinates: Involutive Case

First, we show results for the involutive case: one task coordinate and two orthogonal coordinates. We choose, as before, the x-component of the end-effector position as task and will have two orthogonal coordinates ξ_1 and ξ_2.

We begin showing simulation results using the plane-stack-based locally orthogonal coordinates (Sec. V). We proceed to show simulation results using the neural network-based orthogonal coordinates and present how they overcome the simple approach's limitations and improve decoupling significantly.

1) Plane Stack Approximation: Let $q_0 \in \mathcal{Q}$ be the chosen base configuration for the locally orthogonal coordinates and consider the vector

$$v = \nabla_M h(q_0). \tag{38}$$

The vector v must be contained in both planes for the two orthogonal coordinates. We set

$$n_1 = \frac{\begin{bmatrix} v_2 & -v_1 & 0 \end{bmatrix}^\mathsf{T}}{\sqrt{v_1^2 + v_2^2}} \qquad n_2 = \frac{n_1 \times v}{|n_1 \times v|} \tag{39}$$

and use the base planes $E(q_0, n_1)$ and $E(q_0, n_2)$ for the linear, locally orthogonal coordinates (22). These locally orthogonal coordinates and the task coordinates are used in the trajectory tracking controller τ_2 (37). Two interleaved jumps in the x- and ξ_1-coordinate are commanded. Fig. 21 shows the results of this experiment.

The initial configuration q_0 is chosen such that the task self-motion manifold shows comparatively low curvature, and
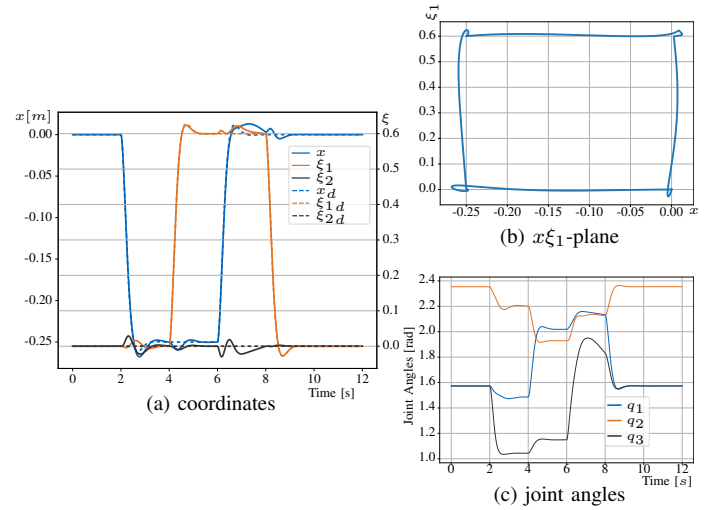

Fig. 21. Results of dynamic simulation using the plane approximation $\xi_P(q)$ for interleaved steps in the x- and ξ_1-coordinate.

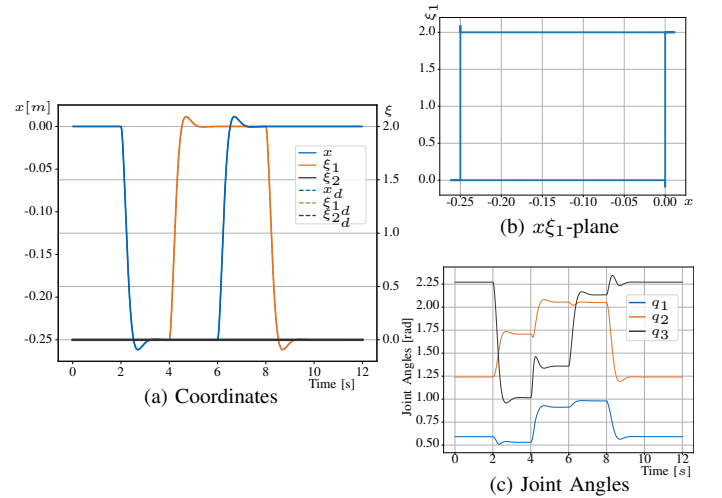Fig. 22. Results of dynamic simulation using the neural network model $\xi_{nn}(q)$ for interleaved steps in the x- and ξ_1-coordinate.

the base configuration for the orthogonal coordinates is set to the initial configuration. Even for relatively large motions in joint space (Fig. 21c), the dynamics of the task- and orthogonal coordinates look moderately decoupled, which can be observed in the time-evolution of the coordinates and their desired values (Fig. 21a) as well as in Fig. 21b where the trajectory is projected onto the $x\xi_1$-plane. The decoupling performs better when the configuration is closer to the initial and base configuration q_0.

The first column of Fig. 23 (adgj) shows results for a more extended simulation with multiple commanded jumps in all three coordinates for the same base configuration q_0. Fig. 23j shows the trajectory of the manipulator in joint space, where additionally, the two relevant task self-motion manifolds are rendered. The self-motion manifolds show comparatively low curvature in this area.

The linear orthogonal coordinates do not perform well for regions of larger curvature. The second column in Fig. 23 (behk) shows results for another initial configuration. Two major problems arise due to the curvature: (1) the decoupling is very poor and (2) the coordinates may contradict each other. We observe large cross-couplings between the coordinates on which the jump occurs and the other coordinates. Moreover, we observe that certain targets cannot be reached, which is
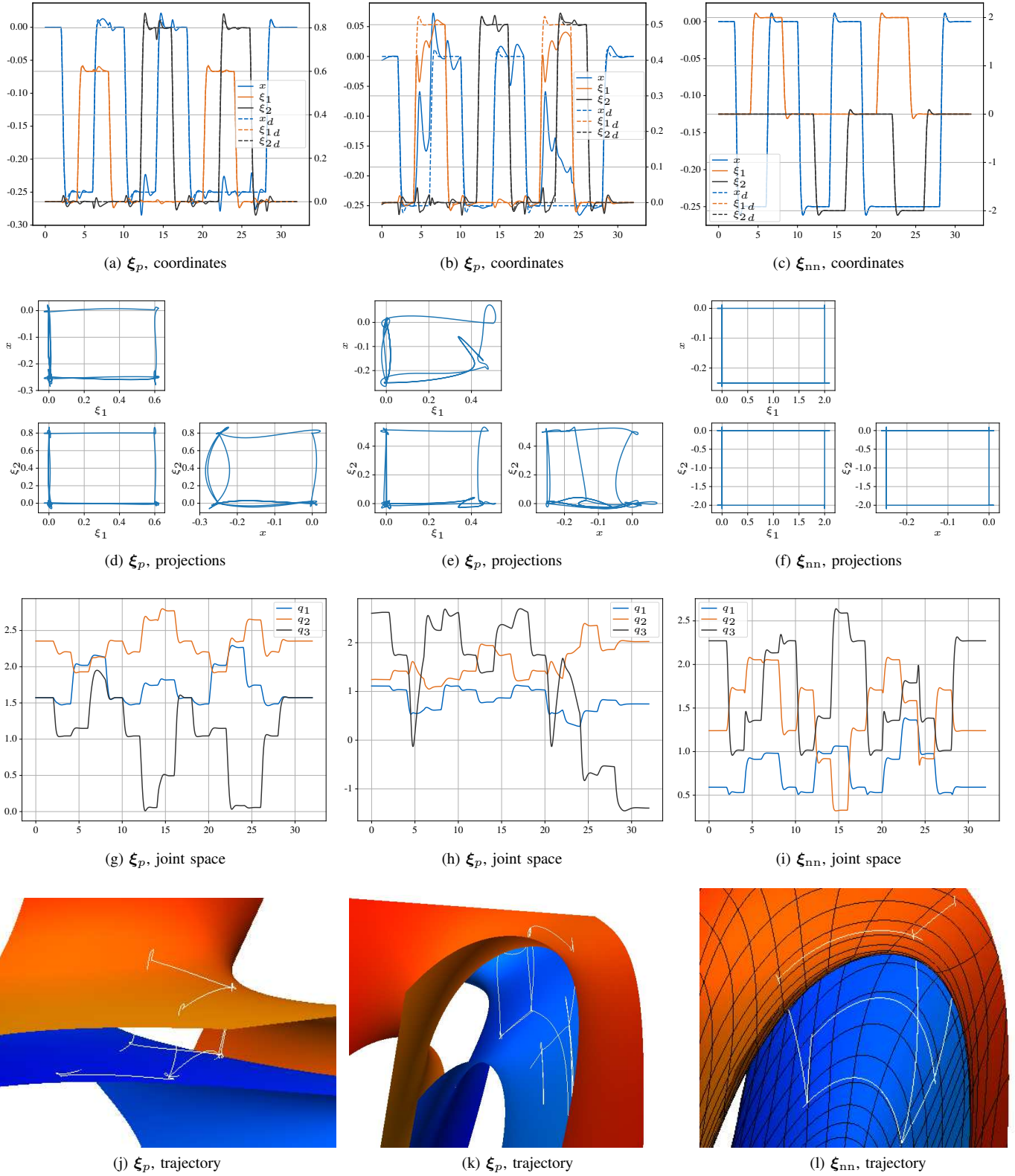
(a) $\boldsymbol{\xi}_p$, coordinates

(b) $\boldsymbol{\xi}_p$, coordinates

(c) $\boldsymbol{\xi}_{\mathrm{nn}}$, coordinates

(d) $\boldsymbol{\xi}_p$, projections

(e) $\boldsymbol{\xi}_p$, projections

(f) $\boldsymbol{\xi}_{\mathrm{nn}}$, projections

(g) $\boldsymbol{\xi}_p$, joint space

(h) $\boldsymbol{\xi}_p$, joint space

(i) $\boldsymbol{\xi}_{\mathrm{nn}}$, joint space

(j) $\boldsymbol{\xi}_p$, trajectory

(k) $\boldsymbol{\xi}_p$, trajectory

(l) $\boldsymbol{\xi}_{\mathrm{nn}}$, trajectory

Fig. 23. Simulation results for involutive case using the trajectory tracking controller for commanded jumps in the respective coordinates. While the first and middle columns show results for the simple linear, locally orthogonal coordinates $\boldsymbol{\xi}_p$, the last columns shows the results using $\boldsymbol{\xi}_{\mathrm{nn}}$. In the first row we show the time evolution of the desired- and actual values of the coordinates and in the second row we show the trajectory expressed in coordinates projected onto the coordinate planes. Comparison on the magnitude of the motion in joint space to ensure comparability can be read in the third row. Finally, the bottom row shows the trajectory of the manipulator in joint space rendered as the 3D-space curve. Additionally, the self-motion manifolds of the x-coordinates for the target values $x = 0$ and $x = 0.25$ are shown. For the neural network setting we also show the coordinate lines for ξ_1 and ξ_2 on the self-motion manifolds. All joint angles are in radians and Cartesian positions in meters; the orthogonal coordinates ξ_i have no unit.

(a) Coordinates

(b) Disturbing task accelerations

(c) Trajectory in $x\xi_1$-plane

(d) Joint Angles
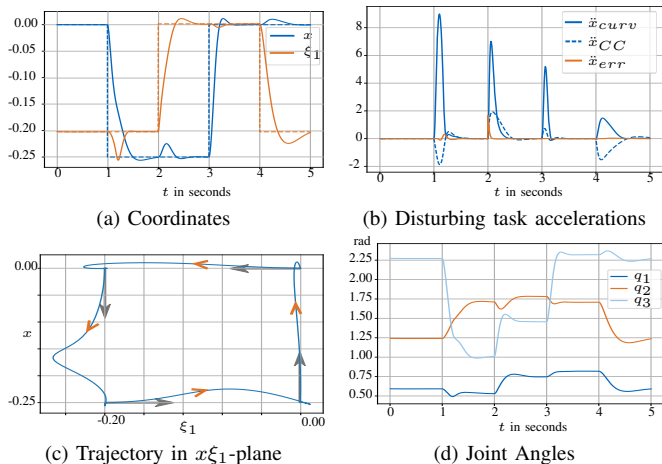
Fig. 24. Simulation results with disabled feed-forward controller for interleaved jumps in the x- and ξ_1-coordinate. **(a)**: time evolution of the desired and actual value of the coordinates, **(b)**: breakdown of the contribution to the disturbing unwanted task accelerations, **(c)** trajectory of the manipulator in coordinates projected onto the $x\xi_1$-plane and **(d)**: time evolution of the manipulator in joint space.

due to the second problem of mutual contradiction of target values for the coordinates, i.e., task inconsistency.

2) Neural Network: Next, we take the neural network $\boldsymbol{\xi}_{\mathrm{nn}}(\boldsymbol{q})$ approximation trained for $\boldsymbol{A} = \boldsymbol{M}$. We use the same simulation and controller as before but replace the coordinate function $\boldsymbol{\xi}_p(\boldsymbol{q})$ with the neural network $\boldsymbol{\xi}_{\mathrm{nn}}(\boldsymbol{q})$. We first repeat the same interleaved jumps as in Fig. 21. The results are shown in Fig. 22. Neither in the time evolution of the coordinates (Fig. 22a) nor in the trajectory plot (Fig. 22b) do we observe any couplings. The overshoots are due to the damping ratio $\zeta = 0.7$ of the pre-filter for the trajectory generation.

The third column in Fig. 23 (cfil) shows the experiment commanding jumps in all the coordinates. Also, in this experiment, we do not observe mutual coupling between the coordinates.

3) Feedback Only: We also show results for pure feedback control, i.e., we disable the feed-forward controller and only apply the feedback controller $\boldsymbol{\tau}_1$ (35). This way, we analyze the disturbing accelerations due to the curvature of the self-motion manifold and the Coriolis and centrifugal forces and compare them to the accelerations due to errors in the trained models. We consider the case of regulation: $\dot{x}_d = \dot{\xi}_{i_d} = 0$.

For this experiment, we use $\boldsymbol{\xi}_{\mathrm{nn}}(\boldsymbol{q})$ and command the same jumps as in Fig. 22 using the same initial configuration. Fig. 24 shows the results. Using the first row of (36) we compute the individual unwanted contributions to the task acceleration $\ddot{x}$. The term

$$\ddot{x}_{\mathrm{err}} = \boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{J}_\xi^{\mathsf{T}} \boldsymbol{f}_\xi^{\mathrm{imp}} \tag{40}$$

denotes the contribution due to the training inaccuracies, i.e., if the desired condition (6) is exactly satisfied, this term will vanish. In contrast to the setting including the feed-forward controller, disturbances and couplings are now clearly observable. However, when considering the breakdown of the individual contributions to the unwanted task acceleration in Fig. 24b, the inaccuracies of the orthogonal coordinates are very low compared to the other terms. In particular, we have negligible *immediate* task acceleration due to a force $\boldsymbol{f}_\xi$. Still, we observe delayed responses due to the accumulated speed

generated by $\boldsymbol{f}_\xi$ that couples in via the Coriolis- and centrifugal terms and the curvature of the self-motion manifolds.

4) Comparison to the Projection-based Approach: Fig. 22 and Fig. 23, right column, clearly show the massive benefit of decoupled and consistent coordinates compared to the classical approach in Fig. 20.

D. Quasi-Orthogonal Coordinates: Non-Involutive Case

For the non-involutive example, we use the x- and y-coordinate as task coordinates and have only a single quasi-orthogonal coordinate ξ. The distribution Δ with the local basis $\{\nabla_M x, \nabla_M y\}$ is generally not involutive.

1) Plane Stack Approximation: For the locally orthogonal coordinates we choose

$$\boldsymbol{n} = \frac{\nabla_M x(\boldsymbol{q}_0) \times \nabla_M y(\boldsymbol{q}_0)}{|\nabla_M x(\boldsymbol{q}_0) \times \nabla_M y(\boldsymbol{q}_0)|} \tag{41a}$$

$$\xi_p(\boldsymbol{q}) = \boldsymbol{n}^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{q}_0). \tag{41b}$$

The first two rows of Fig. 25 show simulation results for jumps in the x- and ξ-coordinate. We show the desired and actual values in the first, the evolutions of joint angles in the second, and the projection onto the $x\xi$-plane in the third column. In the last column, we show robot configurations upon convergence at the respective times. For large jumps we have the issue of contradicting targets. As expected, the overall decoupling is worse than for the involutive example because the problem cannot have a globally optimal solution.

2) Neural Network: Next, we use $\boldsymbol{\xi}_{\mathrm{nn}}(\boldsymbol{q})$, where the network has been trained for the non-involutive example. The last row of Fig 25 shows the results of the simulation with ξ_{nn}. Compared to the linear approximation with ξ_p the error is more distributed over the space. While we have the best decoupling at $\boldsymbol{q}_0$ for ξ_p, there is no priority to any points for the ξ_{nn}. The third column shows results with ξ_{nn} for large steps, i.e., we command jumps comparable to those where the linear model failed. Here, the neural network model still performs reasonably well.

Finally, Fig. 28 shows simulation results using ξ_{nn} commanding motion in all the coordinates with (first column) and without (second column) the feed-forward controller. Especially in the case without the feed-forward term we observe larger couplings between x and y than between ξ and x/y.

3) Comparison to the Projection-based Approach: In the non-involutive settings, we cannot have perfect decoupling, which becomes obvious when comparing the results to the involutive case. The effect of the non-involutivity shows up especially when comparing Figs. 23f and 28c. However, a decent performance can be achieved using the neural net. Compared to the results from the baseline approach in Fig. 20, the residual errors are almost negligible.

IX. EXPERIMENT ON 7 DOF ROBOT

This section validates the approach experimentally on our 7 DoF KUKA iiwa R800 robot. We select the position of the end-effector as task coordinates, i.e., we select $\boldsymbol{h} : \mathcal{Q} \to \mathbb{R}^3$; $\boldsymbol{h} : \boldsymbol{q} \mapsto [x, y, z]^{\mathsf{T}}$. Hence, we will have a degree of redundancy of $r = 4$ and four quasi-orthogonal coordinates $\xi_i(\boldsymbol{q})$ for $i \in \{1, 2, 3, 4\}$. We aim at a kinematic redundancy resolution here and set $\boldsymbol{A} = \boldsymbol{I}$.

We train the model in a subspace of the configuration space by randomly sampling joint configurations uniformly out of a

(a) ξ_p, small steps, $\boldsymbol{\tau}_2$

(b) ξ_p, large steps, $\boldsymbol{\tau}_2$

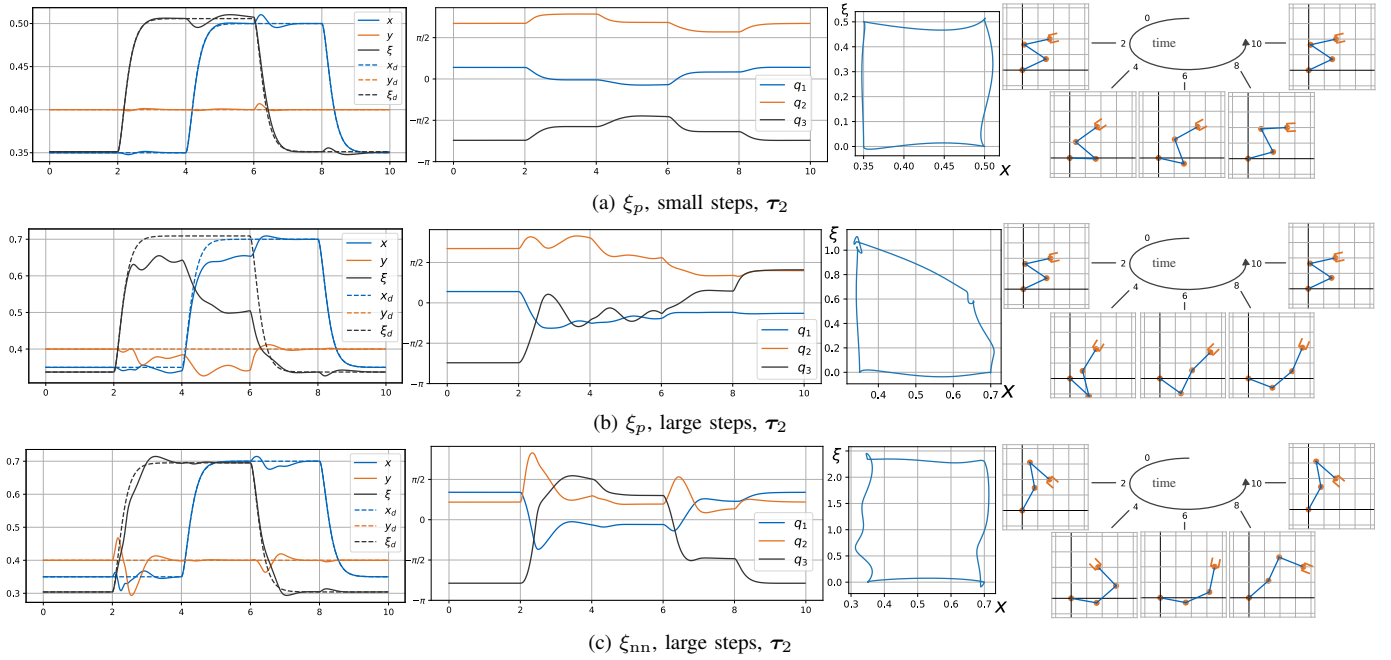

(c) ξ_{nn}, large steps, $\boldsymbol{\tau}_2$

Fig. 25. Simulation results for the non-involutive example for jumps in two of the coordinates. All joint angles are in radians and Cartesian positions in meters; the orthogonal coordinates ξ have no unit. *A video of the simulation with additional information is available in the supplementary material.*
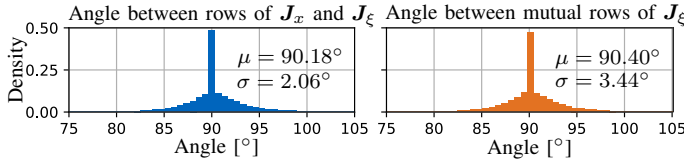

Fig. 26. Evaluation of the trained network for kinematic orthogonal coordinates for the 7 DoF robot.

hyper-cube. The hyper-cube has an edge length of $\pi/2$ and is centered at $\boldsymbol{q}_0 = [0, 0.52, 0, -1.05, 0, 0.79, 0]^\top$.

We train a neural network with $n_1 = 1024$ and $n_2 = 512$ neurons and the first and second layer. The training was in 200 epochs with 50 steps per epoch. 10^4 fresh training samples were generated at the beginning of each epoch. We sample test configurations from the hypercube and evaluate angles in the histogram in Fig. 26. The standard deviation of the residual angle between rows of $\boldsymbol{J}_x$ and $\boldsymbol{J}_\xi$ amounts to $\sigma = 2.06°$; and between mutual rows of $\boldsymbol{J}_\xi$ we obtain $\sigma = 3.44°$.

Finally, we generate a smooth desired trajectory $\boldsymbol{\varphi}_d(t) = (\boldsymbol{x}_d, \boldsymbol{\xi}_d)$ by low-pass filtering jumps in the coordinates as before. The neural network is trained for kinematic control, i.e., we apply velocity-based control using $\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{\varphi}}_d$.

We show results for a hardware experiment on our 7 DoF robot in Fig. 27. On the top, we show the time history of the primary task $\boldsymbol{x}$, and in the middle the $\boldsymbol{\xi}$ coordinates over time. Bold lines show desired values, and thin lines in lighter color show measured values. The bottom pane shows the measured joint positions over time. Comparing the jump heights in the bottom pane, we observe that the jumps in ξ_i also perform significant motions of the robot.

We interleave jumps in the task and orthogonal coordinates and perform a long and complex journey through the task- and orthogonal space. Ultimately, we set the final values for $\boldsymbol{x}$ and $\boldsymbol{\xi}$ back to the initial ones. Our robot has also returned to the same joint configuration! This can be observed with the help of the dotted lines in the bottom pane.
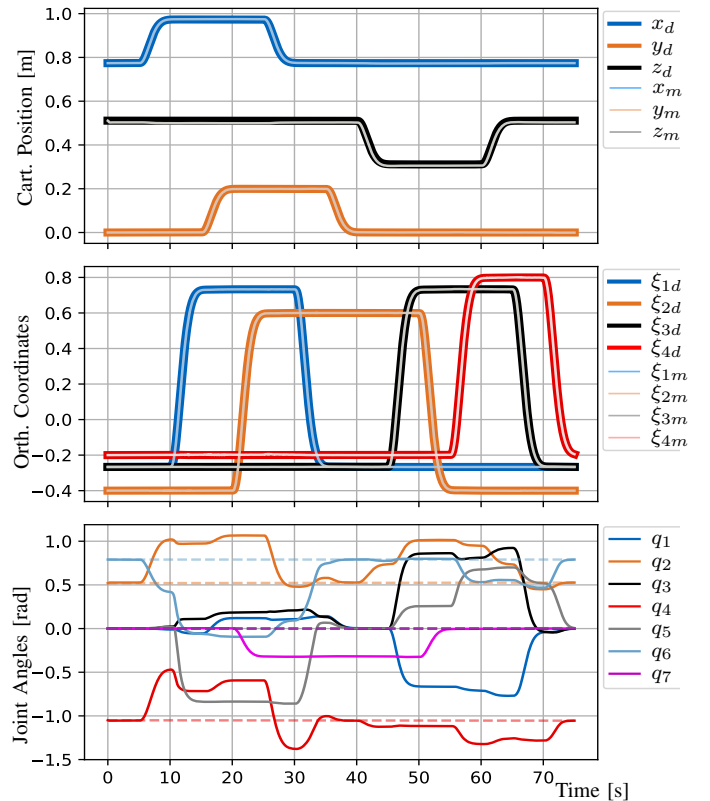

Fig. 27. Results of kinematic control using orthogonal coordinates on the KUKA LBR iiwa R800 for a desired trajectory in $\boldsymbol{x}$ and $\boldsymbol{\xi}$. Bold lines show desired values, and thin lines in lighter color show measured values. (a) Time history of the task coordinates; (b) Time history of the orthogonal coordinates $\boldsymbol{\xi}$; and (c) Joint angles over time. This last plot has no desired values.

On the level of the commanded values, there is no observable coupling between the primary task and the secondary task, which is consistent with the very low residuum reported above.

This demonstrates that the algorithm achieves an almost perfect kinematic decoupling. Moreover, note that after returning to the initial Cartesian point, the robot returns to the initial joint configuration. In contrast to state of the art, we are free to move at wish in the orthogonal coordinates in between, which can be used for fulfilling secondary tasks. On the measured values, one can observe some slight dynamic coupling between coordinates, which is unsurprising, as we are not performing decoupling on dynamic, but only on kinematic level in this experiment. Still, remarkably, the dynamic coupling among primary tasks is less than the couplings to the orthogonal coordinates. The seventh axis of the robot does not influence the main task. Interestingly, the second orthogonal coordinate ξ_2 learned to exploit this by only controlling this joint.

X. DISCUSSION AND CONCLUSIONS

We gave a systematic analysis of the geometrical aspects of robot redundancy resolution on configuration (position) level, i.e., in terms of quasi-orthogonal configuration space coordinates. Strictly decoupled coordinates can be achieved if the task space is one-dimensional. For higher dimensions, this is only possible in particular cases of forward kinematics, as shown in [53, pp. 151–154] for three dimensions. For all other cases, we introduced the concept of quasi-orthogonal coordinates, providing the best possible decoupling in a certain least squares sense. Beyond the quite exciting geometric insights into the problem, one might ask, what are the practical advantages of this new approach to robotics? While it is too early to draw general conclusions, some apparent aspects can already be discussed.

A. Comparison to Projection-based Methods

Classical projection-based methods destroy passivity and can lead to *hidden springs* that suddenly discharge, causing unexpected behavior and causing shaky and vibrating behavior in certain configurations. These problems vanish in the case of orthogonal coordinates, where no projection is needed, and convergence analysis works simply based on the sum of all potentials. Note that for topological reasons, the potentials may have several (possibly unstable) equilibria, for instance, when controlling on $SO(3)$ [66]. In the quasi-orthogonal coordinates (non-involutive) case, we can circumvent the need for projectors and avoid the problems of local redundancy resolution. This comes at the cost of accepting slight couplings. In this case, an additional projection could be applied to remove the remaining couplings, using the projectors $\boldsymbol{P}$ or $\boldsymbol{P}^\mathsf{T}$ on the quasi-orthogonal foliations. The projection will only remove a small portion of the torque $\boldsymbol{\tau} = \boldsymbol{J}_\xi^\mathsf{T} \boldsymbol{f}_\xi$.

B. Comparison to Global Redundancy Resolution

Several publications have dealt with a restrictive version of global redundancy resolution (e.g., [37], [38]). These publications solve the redundancy by fixing one joint configuration for every end-effector pose; for closed end-effector paths, the joint configuration will never drift away. This comes at the cost of virtually removing degrees of freedom, preventing secondary tasks. Other publications on global redundancy resolution do not fix a one-to-one correspondence but do not provide coordinates (e.g. [35], [36]).

Haug's cyclic differentiable manifold representation [39] provides global redundancy resolution and retains the freedom to use redundant degrees of freedom. It is based on first identifying linear subspaces in a neighborhood of a base configuration and then re-adding the nonlinearity of the main task by a nonlinear function that depends on both the main task and the additional coordinates (see (4)). In this approach, the two sets of coordinates no longer directly act on the robot via an analytical Jacobian with interpretable directions, which limits the use of the approach for operational space control.

Our approach extends these results: for any fixed desired value $\boldsymbol{\xi}_d$, we can do the same cyclic tasks as in the global redundancy resolution publications. However, we can still perform self-motions at any time by commanding time-dependent trajectories $\boldsymbol{\xi}_d(t)$ and even superimpose these motions to the main task. The dynamical decoupling will ensure that the main task is not disturbed. We can perform secondary tasks *and* have global redundancy resolution on position-level.

C. Representation of Orthogonal Foliations

The objective of finding orthogonal foliations can be written as a system of partial differential equations. We have presented a proof of concept for the orthogonal foliations by formulating the objective as an optimization problem and using a neural network. A variety of other solutions to find a $\boldsymbol{\xi}(\boldsymbol{q})$ optimally satisfying (6) exist. For instance, discrete methods and finite elements are often used in computer graphics, structure analysis, and materials science to solve PDEs. We recently discovered that in the field of molecular dynamics, researchers find learned potential fields by Gaussian processes [67]. We will investigate if we can apply their methods to our problem in the future.

The approximation by linear local quasi-orthogonal coordinates, which we employed for comparison in the experiments, might provide a practical value. If one considered the learning of quasi-orthogonal coordinates too expensive, one could still use the linear approximation in combination with classical projection-based methods instead of defining potentials classically, directly in joint space. One can expect in practice that the components which need to be projected out will be substantially reduced.

The comparison and detailed investigation of different approaches to find a parametrization $\boldsymbol{\xi}(\boldsymbol{q})$ for quasi-orthogonal foliations based on (6) was not within the scope of this paper and will be addressed in future work.

D. Computational Complexity and Training Times

In contrast to classical projection-based methods, our approach requires a training phase. However, we would like to emphasize that the training phase is required once and can happen offline. During operation, only forward passes of the trained network are needed. The computation times for a forward pass are deterministic and, therefore, real-time capable.

We measured the time needed for a forward pass, i.e., computing the learned coordinates $\boldsymbol{\xi}(\boldsymbol{q})$ and the corresponding Jacobian $\boldsymbol{J}_\xi(\boldsymbol{q})$, for the large model for the LBR iiwa robot. Without any optimization and using a single core on a regular laptop, we achieve a mean forward pass time of $133\mu s$ with a standard deviation of $28.2\mu s$ for our naïve Eigen-based C++ implementation. This could even be further improved using

TABLE I
TRAINING TIMES

Sec.	n	m	n_1	n_2	metric	training minutes
VII-A	2	1	256	64	I	1.96
					M	2.05
VII-B	3	2	256	64	I	4.33
VIII-C2	3	1	512	256	M	6.69
VIII-D2	3	2	512	256	M	4.44
IX	7	3	1024	512	I	49.36

hardware acceleration or multiple cores, and the variance can be brought to zero by a real-time implementation.

The network must be trained for every robot and choice of task space. The manufacturer of the robot could even do this. There is no need to retrain the network for different positions in joint- or task space! Yet, we provide the training times for the experiments in Table I.

E. Interpretability of the Learned Coordinates

The learned coordinates span all the remaining degrees of freedom. However, they do so arbitrarily, and the robot motions upon motions in the learned coordinates are not necessarily intuitive and will depend on what the network converges to. For now, we are happy with any solution that the network finds. One approach to make them more interpretable is to additionally use interpretable secondary tasks as in the projection framework and incorporate them into the optimization problem. The cost function will penalize how much the learned coordinates deviate from the secondary tasks. This was beyond the scope of this paper and will be addressed in future work.

ACKNOWLEDGMENTS

REFERENCES

[1] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robot. Autom.*, vol. 3, no. 1, pp. 43–53, 1987.
[2] ——, "Inertial properties in robotic manipulation: An object-level framework," *Int. J. Robot. Res.*, vol. 14, no. 1, pp. 19–36, 1995.
[3] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *Int. J. Robot. Res.*, vol. 6, no. 2, pp. 3–15, 1987.
[4] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. IEEE Int. Conf. Adv. Robot.*, 1991, pp. 1211–1216 vol.2.
[5] P. Baerlocher and R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst..*, vol. 1, 1998, pp. 323–329 vol.1.
[6] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst. Man*, vol. 7, no. 12, pp. 868–871, 1977.
[7] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 737–757, 2008.
[8] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *J. Int. Robot. Syst.*, vol. 3, pp. 201–212, Sep. 1990.

[9] A. Dietrich, C. Ott, and A. Albu-Schäffer, "Multi-objective compliance control of redundant manipulators: Hierarchy, control, and stability," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst.*, Nov. 2013, pp. 3043–3050.
[10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer, 2009.
[11] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. CRC Press, Inc., 1994.
[12] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st ed. Cambridge University Press, 2017.
[13] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2005.
[14] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.
[15] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 109–117, 1985.
[16] A. Dietrich, T. Wimböck, A. Albu-Schäffer, and G. Hirzinger, "Integration of reactive, torque-based self-collision avoidance into a task hierarchy," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1278–1293, 2012.
[17] U. Hagn *et al.*, "DLR MiroSurge: a versatile system for research in endoscopic telesurgery," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 5, no. 2, pp. 183–193, 2010.
[18] J. Hutzl, D. Oertel, and H. Wörn, "Knowledge-based direction prediction to optimize the null-space parameter of a redundant robot in a telemanipulation scenario," in *Proc. IEEE Int. Symp. Robot. Sens. Environ.*, Oct. 2014, pp. 25–30.
[19] H. Su, S. Li, J. Manivannan, L. Bascetta, G. Ferrigno, and E. De Momi, "Manipulability optimization control of a serial redundant robot for robot-assisted minimally invasive surgery," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 1323–1328.
[20] I. D. Walker, "The use of kinematic redundancy in reducing impact and contact effects in manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, May 1990, pp. 434–439.
[21] N. Mansfeld, B. Djellab, J. R. Veuthey, F. Beck, and S. Haddadin, "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst.*, 2017, pp. 5390–5397.
[22] K. L. Doty, C. Melchiorri, and C. Bonivento, "A theory of generalized inverses applied to robotics," *Int. J. Robot. Res.*, vol. 12, no. 1, pp. 1–19, 1993.
[23] J. Hermus, J. Lachner, D. Verdi, and N. Hogan, "Exploiting redundancy to facilitate physical interaction," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 599–615, 2022.
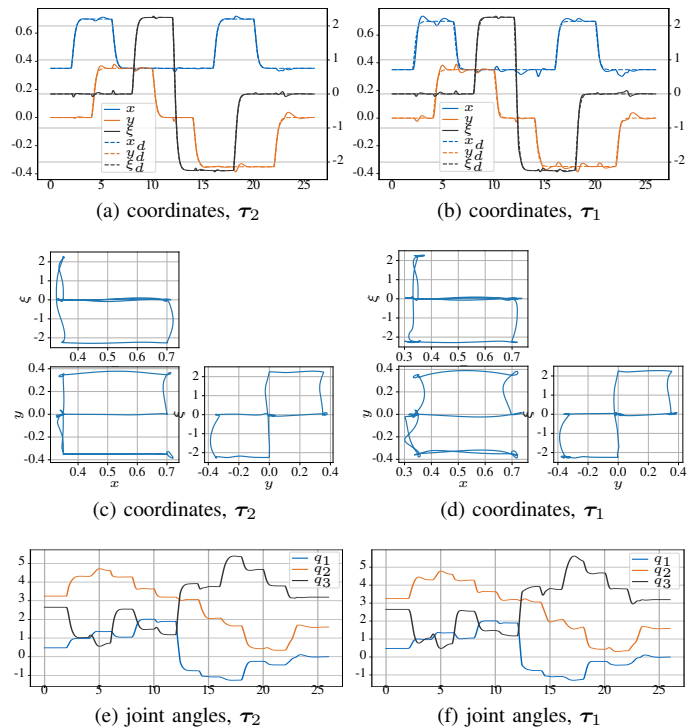

Fig. 28. Additional simulation results for the non-involutive example. Left: trajectory tracking, right: impedance control. All joint angles are in radians and Cartesian positions in meters; the orthogonal coordinates ξ_i have no unit.

[24] C. A. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. Syst. Man Cybern.*, vol. 13, no. 2, pp. 245–250, Mar. 1983.

[25] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *Int. J. Humanoid Robot.*, vol. 02, no. 04, pp. 505–518, 2005.

[26] X. Wu, C. Ott, A. Albu-Schäffer, and A. Dietrich, "Passive decoupled multitask controller for redundant robots," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 1, pp. 1–16, 2023.

[27] A. Dietrich, X. Wu, K. Bussmann, C. Ott, A. Albu-Schäffer, and S. Stramigioli, "Passive hierarchical impedance control via energy tanks," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 522–529, 2017.

[28] G. A. Folkertsma and S. Stramigioli, "Energy in robotics," *Foundations and Trends® in Robotics*, vol. 6, no. 3, pp. 140–210, Oct. 2017.

[29] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *Int. J. Robot. Res.*, vol. 34, no. 11, pp. 1385–1400, 2015.

[30] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 985–994, 2009.

[31] C. Ott, A. Dietrich, and A. Albu-Schäffer, "Prioritized multi-task compliance control of redundant manipulators," *Automatica*, vol. 53, pp. 416–423, 2015.

[32] A. Dietrich and C. Ott, "Hierarchical impedance-based tracking control of kinematically redundant robots," *IEEE Trans. Robot.*, vol. 36, no. 1, pp. 204–221, 2020.

[33] J. W. Burdick, "On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE Computer Society, 1989, pp. 264–270.

[34] M. Dyck, A. Sachtler, J. Klodmann, and A. Albu-Schäffer, "Impedance control on arbitrary surfaces for ultrasound scanning using discrete differential geometry," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7738–7746, 2022.

[35] F. A. Mussa-Ivaldi and N. Hogan, "Integrable solutions of kinematic redundancy via impedance control," *Int. J. Robot. Res.*, vol. 10, no. 5, pp. 481–491, 1991.

[36] A. De Luca, L. Lanari, and G. Oriolo, "Control of redundant robots on cyclic trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1992, pp. 500–506 vol.1.

[37] K. Hauser and S. Emmons, "Global redundancy resolution via continuous pseudoinversion of the forward kinematic map," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 932–944, 2018.

[38] Z. Xie, L. Jin, X. Luo, Z. Sun, and M. Liu, "RNN for repetitive motion generation of redundant robot manipulators: An orthogonal projection-based scheme," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 615–628, 2022.

[39] E. J. Haug, "A Cyclic Differentiable Manifold Representation of Redundant Manipulator Kinematics," *J. Mech. Robot.*, vol. 16, no. 6, p. 061005, 2023.

[40] E. J. Haug and A. Peidro, "Redundant Manipulator Kinematics and Dynamics on Differentiable Manifolds," *J. Comput. Nonlinear Dyn.*, vol. 17, no. 11, p. 111008, 2022.

[41] J. Park, W. Chung, and Y. Youm, "On dynamical decoupling of kinematically redundant manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst.*, 1999, pp. 1495–1500.

[42] C. Ott, A. Kugi, and Y. Nakamura, "Resolving the problem of non-integrability of nullspace velocities for compliance control of redundant manipulators by using semi-definite Lyapunov functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1999–2004.

[43] E. Monari, Y. Chen, and R. Vertechy, "On locally optimal redundancy resolution using the basis of the null space," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 9665–9671.

[44] G. Frobenius, "Über das Pfaffsche Problem." *Journal für die reine und angewandte Mathematik*, vol. 82, pp. 230–315, 1877.

[45] J. M. Selig, *Geometric Fundamentals of Robotics*, D. Gries and F. B. Schneider, Eds. Springer Science+Business Media New York, 1996.

[46] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems*. Springer Science+Business Media New York, 2004, vol. 49.

[47] F. C. Park and J. W. Kim, "Manipulability and singularity analysis of multiple robot systems: a geometric approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, 1998, pp. 1032–1037.

[48] F. Bullo and R. M. Murray, "Tracking for fully actuated mechanical systems: a geometric framework," *Automatica*, vol. 35, no. 1, pp. 17–34, 1999.

[49] J. Lachner, V. Schettino, F. Allmendinger, M. D. Fiore, F. Ficuciello, B. Siciliano, and S. Stramigioli, "The influence of coordinates in robotic manipulability analysis," *Mech. Machine Theory*, vol. 146, p. 103722, 2020.

[50] J. M. Lee, *Introduction to Smooth Manifolds*, 2nd ed., S. Axler and K. Ribet, Eds. Springer New York, 2012.

[51] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares, "Efficient implementation of marching cubes cases with topological guarantees," *J. Graph. Tools*, vol. 8, no. 2, pp. 1–15, Dec. 2003.

[52] L. P. Eisenhart, "Dynamical trajectories and geodesics," *Ann. Math.*, vol. 30, no. 1/4, pp. 591–606, 1928.

[53] B. McKay, "Introduction to exterior differential systems," arXiv:1706.09697 [math.DG], Sep. 2020.

[54] R. L. Bryant, P. A. Griffiths, and D. A. Grossman, *Exterior Differential Systems and Euler-Lagrange Partial Differential Equations*. University of Chicago Press, 2003.

[55] A. I. Bobenko, D. Matthes, and Y. B. Suris, "Discrete and smooth orthogonal systems: C^∞-approximation," *Int. Math. Res. Not.*, vol. 2003, no. 45, pp. 2415–2459, Jan. 2003.

[56] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Berlin Heidelberg, 2008.

[57] M. S. Floater and K. Hormann, "Surface parameterization: a tutorial and survey," in *Advances in Multiresolution for Geometric Modelling*. Springer Berlin Heidelberg, 2005, pp. 157–186.

[58] S. Maneewongvatana and D. Mount, "It's okay to be skinny, if your friends are fat," *4th Annual Workshop on Comp. Geom.*, 1999.

[59] T. Graepel, "Solving noisy linear operator equations by gaussian processes: Application to ordinary and partial differential equations," in *Proc. Int. Conf. Mach. Learn.* Washington, DC, USA: AAAI Press, 2003, p. 234–241.

[60] A. Sachtler and A. Albu-Schäffer, "Strict modes everywhere - bringing order into dynamics of mechanical systems by a potential compatible with the geodesic flow," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2337–2344, 2022.

[61] A. Sachtler, "Orthogonal manifold foliations for impedance control of redundant kinematic structures," Master's thesis, Technical University of Munich, 2020. [Online]. Available: https://mediatum.ub.tum.de/1578939

[62] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," in *Proc. USENIX Symp. on Op. Syst. Design Impl.*, vol. 12, Nov. 2016, pp. 265–283.

[63] N. Hogan, "Impedance control: An approach to manipulation: Part I-Theory; Part II-Implementation; Part III-Applications," *J. Dyn. Syst. Meas. Control*, vol. 107, no. 1, pp. 1–24, 1985.

[64] A. Albu-Schäffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: recent results with the DLR-light-weight-arms," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, Sep. 2003, pp. 3704–3709.

[65] A. Dietrich, X. Wu, K. Bussmann, M. Harder, M. Iskandar, J. Englsberger, C. Ott, and A. Albu-Schäffer, "Practical consequences of inertia shaping for interaction and tracking in robot control," *Control Eng. Pract.*, vol. 114, p. 104875, 2021.

[66] D. Koditschek, "The application of total energy as a Lyapunov function for mechanical control systems," *Contemp. Math.*, vol. 97, 1989.

[67] N. F. Schmitz, K.-R. Müller, and S. Chmiela, "Algorithmic differentiation for automated modeling of machine learned force fields," *J. Phys. Chem. Lett.*, vol. 13, no. 43, p. 10183 – 10189, 2022.

Alin Albu-Schäffer received his M.S. in electrical engineering from the Technical University of Timișoara, in 1993 and his Ph.D. in automatic control from the Technical University of Munich in 2002. Since 2012 he is the head of the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR), which he joined in 1995. Moreover, he is a professor at the Technical University of Munich, holding the Chair on "Sensor Based Robotic Systems and Intelligent Assistance Systems" at the School of Computation, Information and Technology. His personal research interests include robot design, modeling and control, nonlinear control, flexible joint and variable compliance robots, impedance control, physical human-robot interaction, bio-inspired robot design and control.
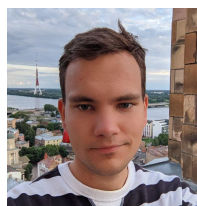
Arne Sachtler is a PhD student at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) and at the TUM School of Computation, Information and Technology of the Technical University of Munich (TUM). He received a M.Sc. degree in *Robotics, Cognition, Intelligence* from the Technical University of Munich in 2020 after spending an exchange year at the Ecole Polytechnique Fédérale de Lausanne (EPFL) in 2018/19. His research focuses on the application of differential geometry to robotics.