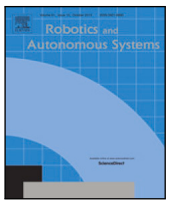
Robotics and Autonomous Systems 180 (2024) 104763

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Learning periodic skills for robotic manipulation: Insights on orientation and impedance

Fares Abu-Dakka [a,*], Matteo Saveriano [b], Luka Peternel [c]

[a] Electronic and Computer Science Department, Faculty of Engineering, Mondragon Unibertsitatea, Arrasate, 20500, Spain
[b] Department of Industrial Engineering, University of Trento, Trento, Italy
[c] Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, Netherlands

ARTICLE INFO

Keywords:
Learning from demonstration
Learning periodic skills
Riemannian manifold learning

ABSTRACT

Many daily tasks exhibit a periodic nature, necessitating that robots possess the ability to execute them either alone or in collaboration with humans. A widely used approach to encode and learn such periodic patterns from human demonstrations is through periodic Dynamic Movement Primitives (DMPs). Periodic DMPs encode cyclic data independently across multiple dimensions of multi-degree of freedom systems. This method is effective for simple data, like Cartesian or joint position trajectories. However, it cannot account for various geometric constraints imposed by more complex data, such as orientation and stiffness. To bridge this gap, we propose a novel periodic DMP formulation that enables the encoding of periodic orientation trajectories and varying stiffness matrices while considering their geometric constraints. Our geometry-aware approach exploits the properties of the Riemannian manifold and Lie group to directly encode such periodic data while respecting its inherent geometric constraints. We initially employed simulation to validate the technical aspects of the proposed method thoroughly. Subsequently, we conducted experiments with two different real-world robots performing daily tasks involving periodic changes in orientation and/or stiffness, *i.e.,* operating a drilling machine using a rotary handle and facilitating collaborative human–robot sawing.

1. Introduction

A fundamental aspect that enables robots to perform their intended tasks is their capability to control movements and physical interactions. The robotic control system demands a task-specific reference behavior, typically encoded and represented by desired trajectories of motion, force, and/or impedance. Consequently, having a robust and reliable method for trajectory generation is crucial for attaining practically applicable robots.

Among the most common trajectory encoding strategies is the Dynamic Movement Primitive (DMP) [1]. DMPs encode discrete and periodic trajectories in stable second-order dynamics. In addition, they may include different types of coupling terms for various functionalities such as obstacle avoidance [2,3]. Discrete (point-to-point) motions are effectively used in many daily activities, such as picking and placing, because they have different starting and ending points. However, there exist several daily activities demanding the execution of periodic trajectories. Periodic trajectories involve repetitive movements at regular intervals, suitable for tasks requiring continuous and well-timed actions. Modeling these trajectories accurately is challenging due to their periodic nature and the need for smooth cycle transitions while interacting with the environment (e.g., periodic drilling in Fig. 1-*left*). This challenge is further amplified in collaborative tasks where continuous periodic coordination between multiple agents needs to be achieved (e.g., collaborative sawing in Fig. 1-*right*). To cope with these characteristics, Ijspeert et al. [1] reformulated DMPs to handle rhythmic/periodic motions. Since then, periodic DMPs have been successfully applied to various periodic tasks, such as surface wiping [3], sawing [4], bolt screwing [5], and locomotion [6,7].

The original design intent of DMPs was to encode one Degree of Freedom (DoF) trajectories, with the approach being highly suitable for depicting independent signals such as Cartesian or joint positions. While synchronization between DoFs can be achieved by exploiting a common phase variable, several robotic skills belong to spaces where the DoFs are interrelated through geometric constraints. This is the case in common orientation representations and in SPD matrices. Achieving effective embedding of these kinds of skills is only possible if the underlying geometric structure of the space is properly considered and the constraints arising from this structure are fulfilled during both training and execution. For example, since stiffness matrices must satisfy positiveness and definiteness constraints or UQs have to fulfill a

* Corresponding author.
E-mail address: fares.abudakka@gmail.com (F. Abu-Dakka).

https://doi.org/10.1016/j.robot.2024.104763
Received 5 February 2024; Received in revised form 12 July 2024; Accepted 23 July 2024
Available online 26 July 2024

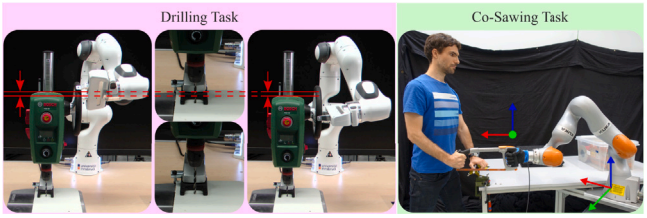


Fig. 1. Experimental setups involving Panda and Kuka IIWA equipped with qb SoftHand. *Left*: Panda operates a drilling machine using a rotary handle, which requires adjustments of periodic pose and stiffness. *Right*: Kuka performs collaborative sawing with a human that requires periodic stiffness modulation.

unit-norm constraint, the use of classical Euclidean operators, as used in previous periodic DMP methods, to operate such variables is inadequate as such data (SPD and UQs) do not lie on a vector space. While the existing methods [8–10] can encode discrete (point-to-point) robotic skills, an approach for effectively encoding manipulation skills with specific geometric constraints in a periodic manner is still missing.

To address this critical gap in periodic DMPs, we propose a novel formulation that allows us to encode general periodic manipulation skills with specific geometric constraints. Our *Riemannian periodic DMP (R-pDMP)* exploits tools from Riemannian geometry to properly handle periodic manipulation data with specific geometric types. The proposed approach is tested on synthetic data and then used, with two different robots, to operate a drilling machine using the rotary handle and do collaborative human–robot sawing (see Fig. 1).

We explored encoding periodic orientation DMPs in a short preliminary study [11]. However, that method lacked generality and was limited to encoding quaternion type of data. This work adds several significant novel contributions:

- We propose a novel periodic DMP formulation that extends the classical formulation by incorporating tools from Riemannian geometry for encoding and reproducing periodic robotic skills with geometric constraints.
- We introduce a novel state-to-action collaborative model that dynamically adjusts end-effector stiffness, enhancing safety and efficiency in human–robot interactions.
- We validate our method through simulation and experiments on Franka Emika Panda and Kuka IIWA platforms, demonstrating superior performance compared to state-of-the-art methods.

The empirical validation of our approach is demonstrated through extensive experiments on Franka Emika Panda and Kuka IIWA robotic platforms. These experiments involve tasks requiring precise orientation control and varying stiffness, such as operating a drilling machine and collaborative human–robot sawing. We provide a comprehensive comparison to state-of-the-art methods, showing that our Riemannian periodic DMPs outperforms existing techniques in terms of accuracy, stability, and computational efficiency. The experimental results clearly illustrate the advantages of our approach in real-world robotic applications.

2. Related work

Learning from Demonstration (LfD) enables an effective way of transferring human manipulation skills to robots. By using human demonstrations, relevant motion patterns, can be extracted and generalized to different situations. The community developed various LfD including: DMPs [10,12–14], Probabilistic Movement Primitives (ProMP) [15], Dynamical Regressive Models [16], stable modeling for dynamical systems [17], Gaussian Mixture Model (GMM) and Task-Parameterized GMM (TP-GMM) [18], and Kernelized Movement Primitive (KMP) [19]. Many of these approaches treat training data as Euclidean vectors arranged as time series. Alternative methods learn quaternion trajectories [20,21], but often do so without enforcing the unit norm constraint, leading to improper quaternions that require additional renormalization steps.

Learning of manipulation skills subject to specific geometric constraints has been investigated by several works in the literature to some degree. Prominent examples of such skills are orientations, impedance, and manipulability matrices. We examine the state-of-the-art for each category in the following paragraphs.

Orientation: UQs are powerful for representing orientations, overcoming the limitations of others like Euler angles [22]. Some approaches extend DMPs to encode UQs. For example, the DMP formulation in [8] uses the vector part of the quaternion product to estimate the error between the current and target orientations, which can lead to slow convergence, to the goal, due to incomplete consideration of geometric constraints. Other methods [9,13] use geometrically consistent rotation error by estimating the angular velocity needed to rotate the robot tip, from its current to the goal orientation, in one unit of time. While these approaches learn *point-to-point* orientation trajectories, our focus is on *periodic* profiles.

Although DMPs have several favorable properties and are widely applied [14], there are alternative LfD approaches for learning *point-to-point* UQ trajectories. For instance, Kim et al. [23] modeled UQ displacements using GMM. This probabilistic encoding is combined with Riemannian metrics in [24] for learning point-to-point orientation trajectories with TP-GMM. In [19], KMP is trained in the tangent space of UQs, where periodic orientation trajectories were tested. Nevertheless, the orientation-KMP does not solve this problem for periodic-DMP formulation, since it is an alternative approach. Many existing robotic systems are based on DMPs due to the popularity of their formulation, therefore there is still a significant functionality gap for the existing DMP-based systems and for future applications of the DMP approach. Furthermore, the computation time of KMPs is higher and they need a separate approach to encode the distribution. Finally, the approach in [19] works on periodic UQ trajectories, while our approach applies to other manifolds.

SPD profiles: SPD matrices are useful in storing data in various applications, *e.g.,* brain–computer interfaces [25], transfer learning [26], and robotics [27]. Recently, Abu-Dakka and Kyrki [10] reformulated discrete (point-to-point) DMPs to learn SPD profiles directly in the $\mathcal{S}^m_{++}$ manifold, allowing adaptation to a new SPD-goal-point. Alternatively to DMP, the method in [28] used a tensor-based formulation of GMM and Gaussian Mixture Regression (GMR) on the SPD manifold to learn and reproduce skills involving SPD matrices without the need of extra pre-/post-processing of data. To enable adaptation to new start-/goal-points, Zeestraten et al. [24] extended TP-GMM to the $\mathcal{S}^3$ manifold, embedding orientation data, which and can be straightforwardly extended to $\mathcal{S}^m_{++}$.

Physical interaction tasks often require adjusting the stiffness matrix of the robot EEF [5,29,30]. The stiffness matrix is an SPD matrix, and several works in the literature have investigated learning it. For example, Kronander et al. [30] used GMM in Euclidean space to learn the vectorization of the Cholesky factor from the decomposition of full stiffness matrices. This approach relies on physical wiggling of the robot EEF to demonstrate the stiffness skill, which is not always feasible when in contact with the environment. A possible solution is teleoperation with various impedance command interfaces to demonstrate the desired stiffness [5]. Nevertheless, to make teleoperation intuitive and immersive for the human teacher, the setup typically becomes complex and expensive, which is not always a viable option. Among the presented approaches, only [5] considers periodic stiffness behaviors, but it only encodes diagonal stiffness matrices with independent DMPs.

State-to-Action Collaborative Model: While some tasks in a shared human–robot workspace require robots to avoid physical collisions with humans, other collaborative tasks involve direct physical interactions and coordinated variable stiffness skills [4,31]. This allows

for more adaptive and responsive interactions. Existing methods often use Electromyography (EMG) [4,32] or Electrical Impedance Tomography (EIT) [33] to infer human intent, which can be complex and intrusive. Unlike these approaches, our state-to-action collaborative model eliminates the need for such measurements by directly using task state (e.g., position and velocity) to determine appropriate actions. Furthermore, while previous approaches [4,32] could implicitly learn some collaborative actions in a semi-black-box fashion, they offered no explicit formalization of periodic collaboration in terms of the collaborative states and actions.

The specific advantages of our state-to-action approach include (*i*) improved safety and efficiency, (*ii*) simplicity and practicality, and (*iii*) versatility. By dynamically adjusting stiffness based on task-specific states, the robot can more effectively manage physical interactions with humans, improving adaptability and coordination. Our approach simplifies the setup by removing the need for additional sensors and measurements, making it easier to implement and more practical for real-world applications. Additionally, the state-to-action model is generalizable to various collaborative tasks beyond sawing, providing a framework for encoding and reproducing a wide range of periodic behaviors in human–robot collaboration. It is also important to note that none of the existing collaborative sawing approaches [4,32,33] enabled encoding of rotated stiffness matrices.

Presented approaches exploit different LfD techniques (DMPs and others) for learning point-to-point orientation, stiffness, and manipulability trajectories. To the best of our knowledge, this is the first work addressing the learning of periodic DMPs for encoding general rhythmic manipulation skills with specific geometric constraints. A clear and separate formulation for periodic Riemannian DMPs is valuable for communities familiar with periodic DMPs and those who use them exclusively, and may not be so familiar with point-to-point ones. Additionally, the different applications of periodic DMPs compared to point-to-point ones require different types of tasks and problems for experimental analysis. Thus, this paper contributes a periodic formulation of Riemannian DMPs and the accompanying experimental analysis in periodic tasks. Another contribution is the state-to-action collaborative model to help encode periodic human–robot interaction tasks, demonstrated in collaborative sawing.

3. Background

Here, we briefly go over the formulation of the classical periodic DMP. Moreover, we give basic notations and operations for quaternion, rotation matrix, and SPD matrix that are used throughout the paper.

3.1. Classical periodic DMP formulation

Periodic DMPs encode repetitive movements by a system of nonlinear differential equations that convergences to a specified rhythmic cycle [1]. For a single DoF, the DMP is

$$\dot{z} = \Omega\left(\alpha_z\left(\beta_z\left(g - y\right) - z\right) + f(\phi)\right), \tag{1}$$

$$\dot{y} = \Omega z, \tag{2}$$

$$\dot{\phi} = \Omega, \tag{3}$$

where y is a variable that is being encoded (often a periodic position), z is an auxiliary state variable, and the derivative of y, namely $\dot{y}$, is z scaled by the frequency of the movement Ω. The ϕ defines the state of the movement and its change over time is determined by the frequency. The phase-variable is obtained by integrating (3). This is a simple and effective choice for a canonical system to learn limit cycle attractors. The phase oscillator ensures that the phase variable ϕ progresses smoothly and consistently over time, guiding the system along the desired limit cycle trajectory.

In periodic DMPs there is no specific goal g due to its repetitive nature, thus in the periodic case g can be either set to zero, or to the average value of the demonstrated trajectory. α_z and β_z are control gains of the second-order basis. Since the basis itself does not encode the shape, a phase-dependent function $f(\phi)$ is added to learn an arbitrary shape. The forcing term function $f(\phi)$ contains N weights $\mathbf{w} = [w_1, \ldots, w_N]$ that determine the local shape of the trajectory and are uniformly distributed over the entire phase of the cycle. Each weight has one Gaussian kernel Ψ linked to it that determines the locality. Weights are adjusted during the learning stage to encode the desired shape of the trajectory. The shape function is defined as

$$f(\phi) = \frac{\sum_{i=1}^{N} \Psi_i(\phi) w_i}{\sum_{i=1}^{N} \Psi_i(\phi)}\, r, \tag{4}$$

$$\Psi_i(\phi) = \exp\left(h\left(\cos\left(\phi - c_i\right) - 1\right)\right), \tag{5}$$

where r is the amplitude modulator of the periodic signal [1,34] (as a default value $r = 1$).

Periodic and discrete DMPs differ in a way that the frequency of trajectory execution replaces the time constant related to trajectory duration [1,12]. The periodic DMPs also has an additional constraint where the initial and final phases ($\phi = 0$ and $\phi = 2\pi$) must coincide to guarantee smooth transitions during the repetitions. The frequency and phase of periodic DMPs can be controlled by an adaptive oscillator, which estimates them [34,35].

To derive the weights of the DMP in (4) that encode a particular demonstration, one can minimize the difference between the DMP shape and the demonstrated shape. Locally Weighted Regression (LWR) based on recursive least squares [36] is commonly applied for this purpose [1,5,34], which we also used in this paper.

3.2. Riemannian manifold

A manifold is a topological space that resembles an Euclidean space in the vicinity of a certain point, *i.e.,* properties of the Euclidean space apply *locally*. If this manifold is smooth, differentiable, and equipped with a Riemannian metric then it is called a Riemannian manifold $\mathcal{M}$. We can compute a tangent space $\mathcal{T}_{\boldsymbol{\Gamma}}\mathcal{M}$ for every point $\boldsymbol{\Gamma}$ on a manifold $\mathcal{M}$, *i.e.,* $\boldsymbol{\Gamma} \in \mathcal{M}$,. The key advantage is that we can employ classical arithmetic tools since the metric in the tangent space is flat. Thus, typical Euclidean operations can be performed in the tangent space, and then the result can be projected back to the manifold.

To project the data back and forth between $\mathcal{M}$ and $\mathcal{T}_{\boldsymbol{\Gamma}}\mathcal{M}$, we can employ the following two mapping systems.

Exponential map projects data from the $\mathcal{T}_{\boldsymbol{\Gamma}}\mathcal{M}$ to the $\mathcal{M}$:

$$\mathrm{Exp}_{\boldsymbol{\Gamma}}(\cdot) : \mathcal{T}_{\boldsymbol{\Gamma}}\mathcal{M} \mapsto \mathcal{M}, \tag{6}$$

Logarithmic map projects data from the $\mathcal{M}$ to the $\mathcal{T}_{\boldsymbol{\Gamma}}\mathcal{M}$:

$$\mathrm{Log}_{\boldsymbol{\Gamma}}(\cdot) : \mathcal{M} \mapsto \mathcal{T}_{\boldsymbol{\Gamma}}\mathcal{M}. \tag{7}$$

These two maps depend on a specific manifold, thus in the following sections, we provide the expressions for UQ, rotation matrix, and SPD matrix.

The Riemannian geometric mean: Given a set of points $\left\{\boldsymbol{\Gamma}_i\right\}_{i=1}^{n} \in \mathcal{M}$ and a geodesic distance $\mathrm{d}\left(\boldsymbol{\Gamma}_j, \boldsymbol{\Gamma}_i\right)$ between two points in $\mathcal{M}$, the Fréchet mean [37] is estimated by minimizing the sum of squared geodesic distances

$$\overline{\boldsymbol{\Gamma}} = \arg\min_{\boldsymbol{\Gamma}\in\mathcal{M}} \sum_{i=1}^{n} \mathrm{d}^2\left(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_i\right), \tag{8}$$

This estimation can be efficiently computed iteratively by following Algorithm 1 [37].

Algorithm 1 Intrinsic mean

Initialization: $\overline{\boldsymbol{\Gamma}} = \boldsymbol{\Gamma}_1$

1: **while** $\|\boldsymbol{\gamma}\| < \delta$ **do**
2: $\quad \boldsymbol{\gamma} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Log}_{\overline{\boldsymbol{\Gamma}}}\left(\boldsymbol{\Gamma}_i\right)$
3: $\quad \overline{\boldsymbol{\Gamma}} = \mathrm{Exp}_{\overline{\boldsymbol{\Gamma}}}(\epsilon\boldsymbol{\gamma})\,;\ \epsilon \leq 1$
4: **end while**

3.3. The unit m-sphere manifold $\mathcal{S}^m$

The manifold $\mathcal{S}^m$ is a topological space embedded in $\mathcal{R}^{m+1}$ Cartesian space, where $\mathcal{S}^m = \left\{\mathbf{Q} \in \mathcal{R}^{m+1} : \|\mathbf{Q}\| = 1\right\}$. For $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathcal{S}^m$ and $\mathbf{q} \in \mathcal{T}_{\mathbf{Q}_2}\mathcal{S}^m$ then, the logarithmic and exponential maps (6) and (7) are defined as in [38]

$$\mathbf{q} = \mathrm{Log}^{q}_{\mathbf{Q}_2}\left(\mathbf{Q}_1\right) = \frac{\mathbf{Q}_1 - \left(\mathbf{Q}_2{}^{\top}\mathbf{Q}_1\right)\mathbf{Q}_2}{\left\|\mathbf{Q}_1 - \left(\mathbf{Q}_2{}^{\top}\mathbf{Q}_1\right)\mathbf{Q}_2\right\|}\,\mathrm{d}\left(\mathbf{Q}_2, \mathbf{Q}_1\right), \tag{9}$$

$$\mathbf{Q}_1 = \mathrm{Exp}^{q}_{\mathbf{Q}_2}\left(\mathbf{q}\right) = \mathbf{Q}_2\cos\left(\|\mathbf{q}\|\right) + \frac{\mathbf{q}}{\|\mathbf{q}\|}\sin\left(\|\mathbf{q}\|\right), \tag{10}$$

where $\mathrm{d}\left(\mathbf{Q}_2, \mathbf{Q}_1\right) \equiv \arccos\left(\mathbf{Q}_1{}^{\top}\mathbf{Q}_2\right)$ defines the geodesic distance between $\mathbf{Q}_1$ and $\mathbf{Q}_2$.

In robotics, $\mathcal{S}^m$ are used to represent directions and orientations. For example, the robot's EEF orientation, in 3D-space, can be described using the space of unit quaternions $\mathcal{S}^3$. A quaternion is an element of the quaternion algebra $\mathbb{H}$, where $\mathbb{H}$ is isomorph to $\mathcal{R}^4$. A UQ is a quaternion with a unit norm, where $\mathbf{Q}$ and $-\mathbf{Q}$ represent the same rotation. Its norm is obtained by $\|\mathbf{Q}\| = \sqrt{v^2 + u_x^2 + u_y^2 + u_z^2}$ while its conjugate is formulated as $\bar{\mathbf{Q}} = v + (-\mathbf{u})$. The multiplications of $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathcal{S}^3$ is defined as

$$\mathbf{Q}_1 * \mathbf{Q}_2 = \left(v_1 v_2 - \mathbf{u}_1{}^{\top}\mathbf{u}_2\right) + \left(v_1\mathbf{u}_2 + v_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2\right).$$

It is worth mentioning that UQs are often computed numerically from rotation matrices, for instance when robot's EEF poses are collected via kinesthetic teaching. In these cases, it may happen that the algorithm returns a quaternion at step t and an antipodal at $t+1$. To ensure that the UQs demonstrated trajectory is discontinuity-free, we can check that the dot product between each adjacent UQ is greater than zero. Otherwise, we flip $\mathbf{Q}_{t+1}$ such as $\mathbf{Q}_{t+1} = -\mathbf{Q}_{t+1}$.

3.4. The special orthogonal group $\mathcal{SO}(m)$

$\mathcal{SO}(m)$ is a subgroup of the orthogonal group $\mathcal{O}(m)$ where its determinant is 1. $\mathcal{SO}(m)$ represents rotations around an origin in m-dimensional space. Let us define $\mathbf{R}_1, \mathbf{R}_2 \in \mathcal{SO}(m)$ and $\mathbf{v} \in \mathcal{T}_{\mathbf{R}_1}\mathcal{SO}(m)$, then the logarithmic and exponential maps (6) and (7) are defined as in [38]

$$[\boldsymbol{\omega}]_{\times} = \mathrm{Log}^{R}_{\mathbf{R}_1}\left(\mathbf{R}_2\right) = \mathrm{logm}\left(\mathbf{R}_1{}^{\top}\mathbf{R}_2\right), \tag{11}$$

$$\mathbf{R}_2 = \mathrm{Exp}^{R}_{\mathbf{R}_1}\left([\boldsymbol{\omega}]_{\times}\right) = \mathrm{expm}\left([\boldsymbol{\omega}]_{\times}\right)\mathbf{R}_1. \tag{12}$$

where $[\boldsymbol{\omega}]_{\times}$ is a skew symmetric matrix $\left([\boldsymbol{\omega}]_{\times}^{\top} = -[\boldsymbol{\omega}]_{\times}\right)$ built from the angular velocity $\boldsymbol{\omega}$. The function to pass from $[\boldsymbol{\omega}]_{\times}$ to $\boldsymbol{\omega}$ is $[\boldsymbol{\omega}]_{\times}^{\vee}$.

In robotics, $\mathcal{SO}(3)$ is used to describe the robot's EEF orientation, in 3D-space, by means of rotation matrices $\mathbf{R}$. An $\mathbf{R} \in \mathbb{R}^{3\times3}$ has 9 real parameters and belongs to the special orthogonal group $\mathcal{SO}(3)$. Thus, it has the following properties: (*i*) $\mathbf{R}^{-1} = \mathbf{R}^{\top}$, (*ii*) $\mathbf{R}^{\top}\mathbf{R} = \mathbf{I}$, where $\mathbf{I}$ is the identity, and (*iii*) $\det(\mathbf{R}) = 1$.

3.5. SPD matrix

In mathematics, a real $m \times m$ matrix $\mathbf{S}$ is Symmetric Positive Definite if it is symmetric (*i.e.,* $\mathbf{S} = \mathbf{S}^{\top}$) and $\mathbf{v}^{\top}\mathbf{S}\mathbf{v} > 0$, $\forall$ nonzero vectors $\mathbf{v}$. The space of $m \times m$ SPD matrices is known as SPD manifold $\mathcal{S}^m_{++}$. Its tangent space is the space of $m \times m$ symmetric matrices $\mathbf{Sym}^m$.

In order to map data from $\mathcal{S}^m_{++}$ to $\mathcal{T}_{\mathbf{S}}\mathcal{S}^m_{++}$ and vice versa, we need to redefine (6) and (7). For $\mathbf{S}, \mathbf{A} \in \mathcal{S}^m_{++}$ and $\mathbf{B} \in \mathcal{T}_{\mathbf{S}}\mathcal{S}^m_{++} \subset \mathbf{Sym}^m$, then [39]

$$\mathrm{Exp}^{+}_{\mathbf{S}}(\mathbf{B}) = \mathbf{S}^{\frac{1}{2}}\,\mathrm{expm}\left(\mathbf{S}^{-\frac{1}{2}}\mathbf{B}\mathbf{S}^{-\frac{1}{2}}\right)\mathbf{S}^{\frac{1}{2}}, \tag{13}$$

$$\mathrm{Log}^{+}_{\mathbf{S}}(\mathbf{A}) = \mathbf{S}^{\frac{1}{2}}\,\mathrm{logm}\left(\mathbf{S}^{-\frac{1}{2}}\mathbf{A}\mathbf{S}^{-\frac{1}{2}}\right)\mathbf{S}^{\frac{1}{2}}, \tag{14}$$

where $\mathrm{expm}(\cdot)$ and $\mathrm{logm}(\cdot)$ are the matrix exponential and logarithm, respectively. Note that previous expressions for logarithmic and exponential maps correspond to the affine-invariant distance on SPD manifold

$$\mathrm{d}\left(\mathbf{S}, \mathbf{A}\right) = \left\|\mathrm{logm}\left(\mathbf{S}^{-\frac{1}{2}}\mathbf{A}\mathbf{S}^{-\frac{1}{2}}\right)\right\|_{\mathrm{F}}. \tag{15}$$

To reduce data space dimensionality, we vectorize the resulting symmetric matrix $\mathrm{Log}^{+}_{\mathbf{S}}(\mathbf{A})$ such that $\boldsymbol{\zeta} = \mathrm{vec}(\mathrm{Log}^{+}_{\mathbf{S}}(\mathbf{A}))$, where $\mathrm{vec}(\cdot)$ is a function that transforms symmetric matrices into vectors using Mandel's notation.

3.6. Double diagonalization design

The double diagonalization design [40] is a method used to compute the damping matrix $\mathbf{D}$ from the stiffness matrix $\mathbf{K}$ to ensure dynamic stability. It involves two main steps:

1. Diagonalization of the Stiffness Matrix: The stiffness matrix $\mathbf{K}$ is diagonalized as $\mathbf{K} = \mathbf{V}\boldsymbol{\Lambda}_K\mathbf{V}^{\top}$, where $\mathbf{V}$ is the matrix of eigenvectors and $\boldsymbol{\Lambda}_K$ is the diagonal matrix of eigenvalues.
2. Construction of the Damping Matrix: The damping matrix $\mathbf{D}$ is then constructed as $\mathbf{D} = \mathbf{V}\boldsymbol{\Lambda}_D\mathbf{V}^{\top}$, where $\boldsymbol{\Lambda}_D$ is a diagonal matrix of eigenvalues with elements $\lambda_D = 2\sqrt{\lambda_K}$, and λ_K are the eigenvalues of $\mathbf{K}$. This ensures that the system has critical damping, leading to a stable dynamic response.

4. Proposed approach

Traditionally, one DMP for each DoF can be used to encode multidimensional periodic variables, which are then synchronized by a common phase. This is applicable for variables such as Cartesian or joint positions, forces, joint torques, *etc*, since every DoF of a certain variable can be encoded and integrated independently to still reproduce the desired combined behavior of the robot. However, the elements of orientations or SPD-based variables are interdependent and subject to additional constraints (*i.e.,* the orthogonality in the case of orientations, and definiteness and positiveness in the case of SPD data). Thus, the traditional approach cannot successfully encode these variables, without pre- and/or post-processing the data.

To be able to learn periodic orientation movements and SPD data, we propose the *Riemannian periodic DMP (R-pDMP)*. Let us consider a periodic demonstration of length T as the trajectory $\boldsymbol{D}^{demo} = \{\boldsymbol{\Gamma}_t\}_{t=1}^{T}$, where $\boldsymbol{\Gamma}_t$ contains the data of the trajectory (*e.g.,* UQs, rotation matrices, stiffness matrices, *etc*.). In this case, we assume that the $\boldsymbol{\Gamma}_t$ are of rhythmic nature and are collected from a single demonstration in order to be used to train a periodic DMP.

4.1. Riemannian periodic DMP (R-pDMP)

Approaches presented in this section exploit operations on Riemannian manifolds, like distance and interpolation, to modify the structure of the DMP and satisfy the geometric constraints during learning and generation. We first present a general DMP formulation to encode periodic data from an arbitrary Riemannian manifold and then particularize the general approach to quaternions as well as rotation and SPD matrices. In order to encode a Riemannian trajectory, we consider the expression of a general second-order system evolving on a manifold [41]

$$\nabla_{\mathbf{z}}\mathbf{z} = \boldsymbol{\Omega}\mathbf{n}\left(\mathbf{z}, \boldsymbol{\Gamma}, \phi\right), \tag{16}$$

$$\dot{\boldsymbol{\Gamma}} = \boldsymbol{\Omega}\mathbf{z}, \tag{17}$$

where $\dot{\boldsymbol{\Gamma}}$ represents the time derivative of $\boldsymbol{\Gamma}$, specifically the Riemannian logarithm of the change in $\boldsymbol{\Gamma}$ over a small time step δt, such that; $\dot{\boldsymbol{\Gamma}} = \mathrm{Log}_{\boldsymbol{\Gamma}(t)}(\boldsymbol{\Gamma}(t+\delta t))/\delta t$. $\boldsymbol{\Omega}$ is a frequency scaling matrix similar to the Ω used in (1) and (2). The dimension of $\boldsymbol{\Omega}$ depends on the specific manifold. $\nabla_{\mathbf{z}}\mathbf{z}$ is the *covariant derivative*, which can be defined from the total derivative $\dot{\mathbf{z}}$ using parallel transport [10,41]. However, computing the parallel transport is, in general, time-consuming. Assuming that consecutive points on the manifold are sufficiently close (*i.e.,* the geodesic between them is almost a straight line), the covariant derivative is well approximated by manifold-valued finite differences [42,43]. This approximation greatly simplifies the computation while introducing negligible errors, as shown in Section 5. Therefore, in this work, we assume that the covariant derivative in (16) can be approximated as:

$$\nabla_{\mathbf{z}}\mathbf{z} = \boldsymbol{\Omega}\mathbf{n}(\mathbf{z}, \boldsymbol{\Gamma}, \phi) \approx \dot{\mathbf{z}}, \tag{18}$$

where $\mathbf{n}(\cdot)$ may consists of several additive contributions. In this paper, we assume that

$$\mathbf{n}(\mathbf{z}, \boldsymbol{\Gamma}, \phi) = \alpha_z\left(\beta_z \mathrm{Log}_{\boldsymbol{\Gamma}}(\boldsymbol{\mathcal{G}}) - \mathbf{z}\right) + \mathbf{f}(\phi), \tag{19}$$

where $\boldsymbol{\mathcal{G}} \in \mathcal{M}$ is the goal point, $\mathrm{Log}_{\boldsymbol{\Gamma}}(\cdot)$ is defined in (7), and α_z and β_z are positive gains. The term $-\alpha_z\mathbf{z}$ is a *dissipative force* (playing the role of damping in a mechanical system). The term $\alpha_z(\beta_z \mathrm{Log}_{\boldsymbol{\Gamma}}(\boldsymbol{\mathcal{G}}))$ is a *conservative force*, *i.e.,* it is the negative gradient of a potential. This is easy to show recalling that $-\frac{1}{2}\nabla_{\boldsymbol{\Gamma}}\mathrm{d}^2(\boldsymbol{\Gamma}, \boldsymbol{\mathcal{G}}) = \mathrm{Log}_{\boldsymbol{\Gamma}}(\boldsymbol{\mathcal{G}})$ [41], where $\mathrm{d}(\cdot,\cdot)$ is the Riemannian distance. The term $\mathbf{f}(\phi)$ is a phase-dependent forcing term and it is learned from the demonstration as discussed later in this section.

As a result, we can reformulate the dynamic system in (1) and (2) as

$$\dot{\mathbf{z}} = \boldsymbol{\Omega}\left(\alpha_z\left(\beta_z \mathrm{Log}_{\boldsymbol{\Gamma}}(\boldsymbol{\mathcal{G}}) - \mathbf{z}\right) + \mathbf{f}(\phi)\right), \tag{20}$$

$$\dot{\boldsymbol{\Gamma}} = \boldsymbol{\Omega}\mathbf{z}. \tag{21}$$

The phase variable ϕ is used to track the progress of the movement, and it is defined as in (3), while the spatial aspects of the motion are captured by (20) and (21). In R-pDMP, the phase-dependent forcing term $\mathbf{f}(\phi)$ is defined as:

$$\mathbf{f}(\phi) = \mathbf{A}_r \frac{\sum_{i=1}^{N} \mathbf{w}_i \Psi_i(\phi)}{\sum_{i=1}^{N} \Psi_i(\phi)}, \tag{22}$$

where weights $\mathbf{w}_i$ determine the trajectory shape and are computed using LWR based on the recursive least-squares method. $\mathbf{A}_r$ is the 3×3 diagonal matrix containing amplitude modulators.

The integration of (20) is done using the *Euler–Riemann stepping method* [41] as

$$\boldsymbol{\Gamma}(t+\delta t) = \mathrm{Exp}_{\boldsymbol{\Gamma}(t)}(\delta t \boldsymbol{\Omega}\mathbf{z}), \tag{23}$$

where $\mathrm{Exp}_{\boldsymbol{\Gamma}}(\cdot)$ is defined in (6) and δt is the sampling time. As already mentioned, the derivatives of $\boldsymbol{\Gamma} \in \mathcal{M}$ are defined using manifold-valued finite differences [42], which read

$$\begin{aligned}
\dot{\boldsymbol{\Gamma}} &= \mathrm{Log}_{\boldsymbol{\Gamma}(t)}(\boldsymbol{\Gamma}(t+\delta t))/\delta t, \\
\dot{\mathbf{z}} &= \boldsymbol{\Omega}^{-1}\ddot{\boldsymbol{\Gamma}} \\
&= \boldsymbol{\Omega}^{-1}\left(\mathrm{Log}_{\boldsymbol{\Gamma}(t)}(\boldsymbol{\Gamma}(t+\delta t)) + \mathrm{Log}_{\boldsymbol{\Gamma}(t)}(\boldsymbol{\Gamma}(t-\delta t))\right)/\delta t^2.
\end{aligned}$$

Previous steps are summarized in Alg. 2.

In case the manifold is a Lie group, the expression of a general second-order system on a Lie group becomes [41]

$$\dot{\mathbf{z}} = \boldsymbol{\Omega}\mathbf{n}(\mathbf{z}, \boldsymbol{\Gamma}, \phi), \tag{24}$$

$$\dot{\boldsymbol{\Gamma}} = \boldsymbol{\Omega}\mathbf{m}(\mathbf{z}, \boldsymbol{\Gamma}), \tag{25}$$

from which is straightforward to derive that

$$\dot{\mathbf{z}} = \boldsymbol{\Omega}\left(\alpha_z\left(\beta_z \log\left(\boldsymbol{\Gamma}_g * (\boldsymbol{\Gamma})^{-1}\right) - \mathbf{z}\right) + \mathbf{f}(\phi)\right), \tag{26}$$

Algorithm 2 Riemannian periodic DMP

Require: Trajectory $\{\boldsymbol{\Gamma}(t)\}_{t=1}^{T}$, gains α_z, β_z, frequency scaling $\boldsymbol{\Omega}$, number of weights N, sampling time δt

```
Initialize goal 𝒢 ← compute_mean({Γ(t)}_{t=1}^T)
Initialize weights w ← 0_N
for t ∈ {1, ..., T} do
    Log_{Γ(t)}(𝒢) ← logarithmic_map(Γ(t), 𝒢)
    w ← w + update_weights(Log_{Γ(t)}(𝒢), N)
end for
Initialize generated trajectory {Γ̂(1)} ← []
for t ∈ {1, ..., T} do
    ż(t) ← pdmp_step(α_z, β_z, Log_{Γ(t)}(𝒢), Ω, w)
    z(t) ← integrate(ż(t), δt)
    {Γ̂(t)} ← exponential_map(𝒢, z(t))
end for
return {Γ̂(t)}_{t=1}^T
```

$$\dot{\boldsymbol{\Gamma}} = \boldsymbol{\Omega}\mathbf{m}(\mathbf{z}, \boldsymbol{\Gamma}). \tag{27}$$

Eq. (26) is formally the same as (20), provided we use the logarithmic map $\mathrm{Log}_{\boldsymbol{\Gamma}}(\cdot) = \log\left(\boldsymbol{\Gamma}_g * (\boldsymbol{\Gamma})^{-1}\right)$ defined using Lie group theory. The term $\mathbf{m}(\cdot)$ in (27) is the *inverse left translation*, which maps a tangent vector from the Lie algebra to the tangent space at $\boldsymbol{\Gamma}$ and depends on the specific Lie group. The expressions of $\mathbf{m}(\cdot)$ and $\log_{\boldsymbol{\Gamma}}(\cdot)$ for UQs and rotation matrices, two Lie groups commonly used in robotics, are given in [9].

As a final remark, we used the Riemannian formulation (20)–(21) in the rest of the paper. However, for the sake of completeness, we also have provided a formulation for Lie groups in (26)–(27).

4.1.1. Quaternion-based periodic DMP (Quat-pDMP)

Inspired by the work on discrete quaternion DMPs [9], we encode a UQ trajectory by reformulating the dynamic system in (20) and (21) as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{\Omega}\left(\alpha_z\left(\beta_z \mathrm{Log}^{q}_{\mathbf{Q}}(\mathbf{Q}_g) - \boldsymbol{\eta}\right) + \mathbf{f}(\phi)\right), \tag{28}$$

$$\dot{\mathbf{Q}} = \boldsymbol{\Omega}\boldsymbol{\eta}. \tag{29}$$

The UQ $\mathbf{Q}_g \in \mathcal{S}^3$ in (28) is the goal orientation, which can be the identity orientation $1 + [0\ 0\ 0]^{\mathsf{T}}$ or the mean of the demonstration quaternion profile. The estimation of the mean can be done by using (8). $\boldsymbol{\Omega}$ is the 3×3 diagonal matrix containing frequencies. We estimate the weights $\mathbf{w}_i$ in (22) as follows:

$$\begin{aligned}
&\frac{\sum_{i=1}^{N} \mathbf{w}_i \Psi_i(\phi)}{\sum_{i=1}^{N} \Psi_i(\phi)} = \\
&\mathbf{A}_r^{-1}\left(\boldsymbol{\Omega}^{-1}\dot{\boldsymbol{\omega}} - \left(\alpha_z\left(\beta_z \mathrm{Log}^{q}_{\mathbf{Q}}(\mathbf{Q}_g)\right) - \boldsymbol{\eta}\right) - \boldsymbol{\omega}\right),
\end{aligned} \tag{30}$$

We perform the integration of (29) by

$$\mathbf{Q}(t+\delta t) = \mathrm{Exp}^{q}_{\mathbf{Q}(t)}(\delta t \boldsymbol{\Omega}\boldsymbol{\eta}(t)) \tag{31}$$

where $\mathrm{Exp}^{q}_{\mathbf{q}(t)}(\cdot)$ is defined in (10).

4.1.2. Rotation matrix-based periodic DMP (Rot-pDMP)

Similar to quaternions, in order to encode and adapt rotation matrix $\mathbf{R}(t) \in \mathbb{R}^{3\times3}$ trajectories while ensuring orthonormality constraints, we propose to reformulate (20) and (21) as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{\Omega}\left(\alpha_z\left(\beta_z\ \left[\mathrm{Log}^{R}_{\mathbf{R}}(\mathbf{R}_g)\right]^{\vee}_{\times} - \boldsymbol{\eta}\right) + \mathbf{f}(\phi)\right), \tag{32}$$

$$\dot{\mathbf{R}} = \boldsymbol{\Omega}[\boldsymbol{\eta}]_{\times}. \tag{33}$$

The nonlinear forcing term $\mathbf{f}(\phi)$ is defined as in (22). The weights defining the generated rotation profile are estimated as in (30) by

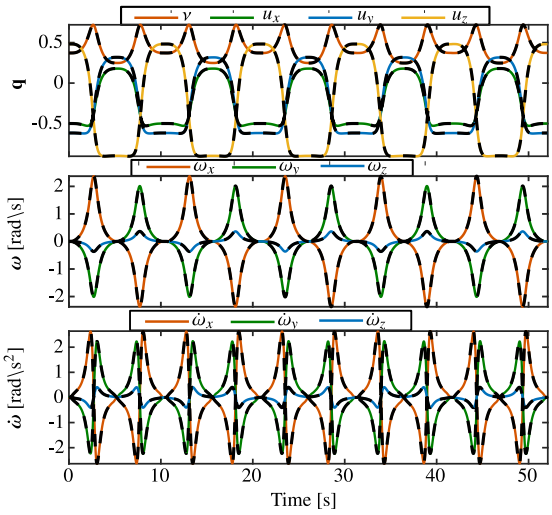


Fig. 2. Reproduction of a rhythmic motion using Quat-pDMP. *Top*: shows the quaternion components. *Middle*: shows the angular velocity components. *Bottom*: shows the angular acceleration components. Dashed black lines represent the synthetic periodic orientation trajectory, while solid colored lines represent the encoded trajectory using Quat-pDMP. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

replacing $\text{Log}^q_{\mathbf{Q}}\left(\mathbf{Q}_g\right)$ with $\left[\text{Log}^R_{\mathbf{R}}\left(\mathbf{R}_g\right)\right]^{\vee}_{\times}$. Finally, (33) is integrated as

$$\mathbf{R}(t+\delta t) = \text{Exp}^R_{\mathbf{R}(t)}\left(\delta t \boldsymbol{\Omega}\ [\boldsymbol{\eta}(t)]_{\times}\right), \tag{34}$$

where $\text{Exp}^R_{\mathbf{R}(t)}(\cdot)$ is defined in (12).

4.1.3. SPD-based periodic DMP (SPD-pDMP)

Inspired by the work on discrete SPD DMPs [10], we encode a periodic SPD matrix trajectory by reformulating the dynamic system in (20) and (21) as

$$\dot{\mathbf{z}} = \boldsymbol{\Omega}\left(\alpha_z\left(\beta_z \text{vec}\left(\text{Log}^+_{\mathbf{Y}}(\mathbf{G})\right) - \mathbf{z}\right) + \mathbf{f}(\phi)\right), \tag{35}$$

$$\dot{\mathbf{y}} = \text{vec}\left(\dot{\mathbf{Y}}\right) = \boldsymbol{\Omega}\,\mathbf{z}. \tag{36}$$

The operator vec$(\cdot)$ represents Mandel's notation for vectorizing the input symmetric matrix. Here, $\dot{\mathbf{Y}} = \text{Log}_{\mathbf{Y}(t)}\left(\mathbf{Y}(t+\delta t)\right)/\delta t \in \mathbf{Sym}^m$ is the 1st-time derivative of $\mathbf{Y}$ which belongs to the space of symmetric matrices. The goal $\mathbf{G} \in \mathcal{S}^m_{++}$, in (35), can be computed as the average of the demonstrated SPD matrix profile using Cholesky decomposition or the more accurate way using (8). $\boldsymbol{\Omega}$ is the $m \times m$ diagonal matrix containing frequencies, while nonlinear forcing term $\mathbf{f}(\boldsymbol{\phi})$ is defined as in (22). We estimate the weights by

$$\frac{\sum_{i=1}^N \mathbf{w}_i \Psi_i(\phi)}{\sum_{i=1}^N \Psi_i(\phi)} = \mathbf{A}_r^{-1}\left(\boldsymbol{\Omega}^{-1}\dot{\mathbf{z}} - \left(\alpha_z\left(\beta_z \text{vec}\left(\text{Log}^+_{\mathbf{Y}}(\mathbf{G})\right) - \mathbf{z}\right)\right)\right),$$

where $\mathbf{A}_r$ is the $m \times m$ diagonal matrix of amplitude modulators. The integration of (36) is done as

$$\mathbf{Y}(t+\delta t) = \text{Exp}^+_{\mathbf{Y}(t)}\left(\delta t \boldsymbol{\Omega}\, \text{vec}^{-1}\left(\mathbf{z}(t)\right)\right), \tag{37}$$

where $\text{vec}^{-1}(\cdot)$ is used to transform a vector (Mandel's notation) into a symmetric matrix.

5. Simulation results

In this section, we validate the performance of the proposed approaches by performing several simulations:

- periodic quaternion-based DMP (Quat-pDMP),
- benchmark Quat-pDMP against two baselines: (emphi) classical pDMP + normalization and (emphii) with a trajectory from [13],
- periodic rotation matrix-based DMP (Rot-pDMP),
- periodic SPD-based DMP (SPD-pDMP),
- comparison between SPD-pDMP and a baseline.

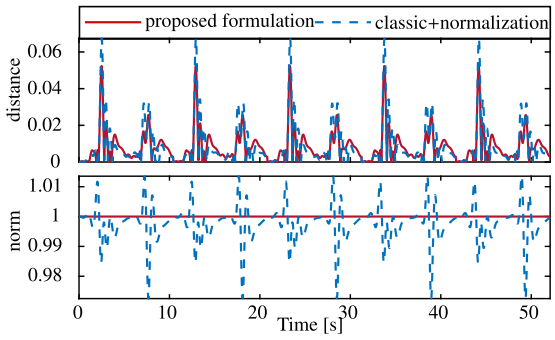


Fig. 3. Comparison between Quat-pDMP and baseline (classical PDMP + normalization). *Top*: Compares the distance between the demonstration and a reproduction using Quat-pDMP and the baseline. *Bottom*: Illustrates both norms resulting from Quat-pDMP and the baseline without normalization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1. Unit quaternion

Fig. 2 shows the simulation results obtained using Quat-pDMP to learn and reproduce a UQ trajectory. We generated a synthetic periodic orientation trajectory (dashed lines) that Quat-pDMP was able to successfully encode with UQs (solid lines). Fig. 2 (*top*) shows the individual elements of the quaternion trajectory, while the last two depict angular velocity (estimated from $\dot{\mathbf{Q}}$) and acceleration, respectively. The error between the demonstrated orientation motion and the encoded motion is negligible. In this test, we set $\alpha_z = 48$, $\beta_z = \alpha_z/4$, $N = 20$, $r = 1$ and $\mathbf{A}_r = \mathbf{I}$.

In order to correctly represent the orientation, the norm of the quaternions must be 1. We performed a simulation to evaluate the how well the proposed approaches maintain this condition. Fig. 3 shows the simulation results obtained using Quat-pDMP and a baseline. Let us first present results for Quat-pDMP. Using Quat-pDMP the norm has a negligible deviation from the expected value 1. In LfD, accurately reproducing the demonstration(s) is often needed. When dealing with Riemannian data, one has to further ensure that the learning algorithm does not introduce distortions to preserve the geometry. The mean error introduced by Quat-pDMP, computed over a single period, is 0.008 (max error 0.051). For comparison, we also computed the covariant derivative using the parallel transport, which results in a mean error of 0.007 (max error 0.047). In view of these results, we claim that Quat-pDMP can correctly encode periodic orientation trajectories and that approximating the covariant derivative using finite differences introduces negligible errors. It is worth mentioning that, when dealing with periodic motions, the phase variable can be used to reset the state of the integrator after 2π, helping to keep the error bounded.

5.1.1. Comparison to classical pDMP + normalization

We compare the results from the previous Section 5.1 to a baseline (Fig. 3). As baseline approach, we encode the quaternion trajectory using 4 classical periodic DMP described in Section 3.1. In other words, we consider each component of the UQ as an independent DoF synchronized through a common phase.

The baseline fails to generate a UQ during the execution of the periodic trajectory (Fig. 3-*bottom*). In this case, the resulting orientation is incorrect and can generate an unexpected behavior if it is used to control a robot. To alleviate this issue, an extra normalization step is

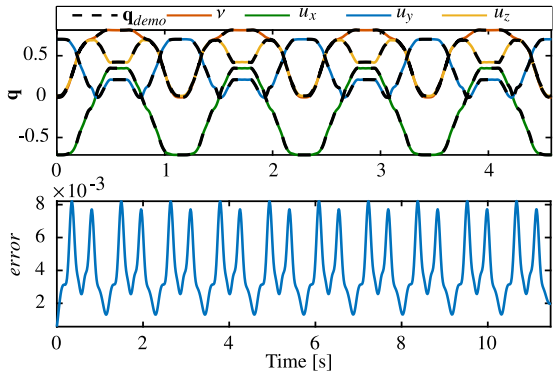


Fig. 4. Results obtained from the concatenation of the UQ trajectory tested in [13] and its mirror. *Top*: Four cycles of UQ elements are generated using our Quat-pDMP (colored solid lines). Dashed black lines are the demonstrated cycle repeated 4 times. *Bottom*: The distance (error) between 10 cycles of the trajectory and the demonstration one. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

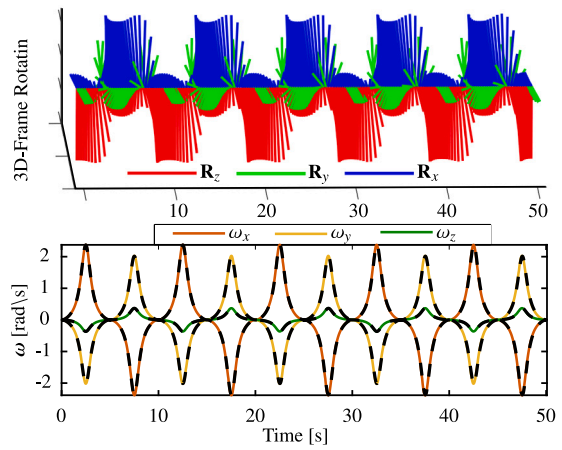


Fig. 5. *Top*: Reproduction of a rhythmic motion using Rot-pDMP (3D-frame representation of the rotation matrices). *Bottom*: Angular velocity (colored solid lines). Dashed black lines correspond to the original motion. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

needed before sending the desired orientation to the robot. Differently, the resulting quaternions from the proposed approach are already with a unit norm and no further post-processing is needed before sending it to the robot. Regarding the reproduction accuracy, the mean error introduced by the baseline, computed in the second period of the trajectory to ensure the convergence of the motion to the limit cycle, was 0.0091. Therefore, the baseline introduces a distortion that is 8.8% larger than Quat-pDMP.

5.1.2. Comparison with a trajectory from [13]

Here, we used the same quaternion trajectory $\mathbf{Q}$ tested by Koutras et al. [13][1] for discrete DMP. The trajectory $\mathbf{Q}$ is not periodic since it has different initial and final UQs. We make a periodic trajectory $\mathbf{Q}'$ by mirroring $\mathbf{Q}$ and concatenating the mirrored trajectory with the original one as $\mathbf{Q}' = \mathbf{Q} \parallel \text{mirror}(\mathbf{Q})$, where $\parallel$ is a concatenation operator. We then fed the periodic trajectory $\mathbf{Q}'$ to our Quat-pDMP. Fig. 4-*top* shows the reproduction of 4 cycles of the trajectory, while the *bottom* shows the error between 10 cycles and the demonstration cycle. The figure shows that the error is quite small and does not increase over cycles. Moreover, this simulation provides evidence that our formulation does not suffer the oscillation mentioned in [13].

5.2. Rotation matrix

Since rotation matrix is still used in various robot applications to represent an orientation, we show that our method can encode periodically changing rotation matrices. Fig. 5 illustrates the performance of the proposed Rot-pDMP in encoding and generating periodic rotation matrices trajectory. The bottom graph shows the angular velocity of the profile. We can see that the demonstrated periodic trajectory (9 elements of the encoded rotation matrix) is accurately reproduced. In addition, the angular velocity is accurately encoded too.

5.3. SPD profile

In this simulation, we use our SPD-pDMP formulations to learn a variable SPD profile in 2D. We use the B-Letter motion from the 2D-Letters handwriting dataset [27] that contains 10 Cartesian trajectories in 2D. We use these trajectories to fit a GMM model and then use GMR to retrieve a covariance matrix $\boldsymbol{\Sigma}_t \in \mathcal{S}^2_{++}$ for each time step t. The covariance matrix profile generated with GMR is not periodic since it has different initial and final points. To make it periodic, we simply mirror the covariance profile obtaining that $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_T$, where T is the last point. Finally, to cover a larger part of the SPD manifold, we scale the obtained covariance by a factor 200. The results of this procedure are shown in Fig. 6. Intuitively, we start at the top (blue ellipse) by drawing the "B-Letter" (green ellipses), and after drawing a complete "B", we reverse direction to end up at the top again, thus making one full period. The generated periodic SPD profile becomes the training data for our DMP-based approaches.

The left plots in Fig. 6 show demonstrated and learned with SPD-pDMP profiles. The right plots in Fig. 6 compare the learning performance of SPD-pDMP and a baseline approach that uses Cholesky decomposition to vectorize the data. More in detail, the baseline approach uses the classical periodic DMP formulation described in Section 3.1. In order to enforce symmetry, positiveness, and definiteness of the generated profiles we exploit Cholesky decomposition. In particular, we pre-process the demonstration and compute the Cholesky decomposition at each time step. Recall that, given an SPD matrix $\mathbf{S} \in \mathcal{S}^m_{++}$, the Cholesky decomposition returns a lower-triangular matrix $\mathscr{L}$, such that $\mathscr{L}^\top \mathscr{L} = \mathbf{S}$, which can be vectorized to reproduce the Cholesky vector $\mathbf{l}$ with dimension $m(m+1)/2$. In our case, we decompose each of the $\boldsymbol{\Sigma}_t \in \mathcal{S}^2_{++}$ matrices and then vectorize the resulting lower-triangular matrices to obtain a 3D trajectory $\mathbf{l}_t \in \mathbb{R}^3$. The obtained trajectory is learned using 3 classical periodic DMPs. To reproduce the profile, we first predict the next 3D vector $\mathbf{l}_{pred}$, reshape it into a lower-triangular matrix $\mathscr{L}_{pred}$, and then compute the SPD matrix as $\boldsymbol{\Sigma}_{chol} = \mathscr{L}_{pred}{}^\top \mathscr{L}_{pred}$.

Fig. 6 (*right*) shows the accuracy of SPD-pDMP and the baseline. All the approaches fulfill the geometric constraints. Moreover, we can see the ability of the 2 algorithms to learn smoothly the stiffness data from the first cycle. The distance in Fig. 6 (*bottom-right*), computed by means of the affine-invariant distance (15) between generated and demonstrated profiles, shows that the 2 methods have similar accuracy (the mean distance is 0.008 and the max distance is 0.03). For comparison, we also computed the covariant derivative using the parallel transport, which results in a mean distance of 0.006 (max distance 0.018). In view of these results, we claim that SPD-pDMP can correctly encode SPD trajectories and that approximating the covariant derivative using finite differences introduces negligible errors.

5.4. Broader implications for robotic manipulation

The results from our simulations demonstrate several important implications for the field of robot manipulation skills learning:

[1] Authors would like to thank Leonidas Koutras for providing us the unit quaternion trajectory used in [13].

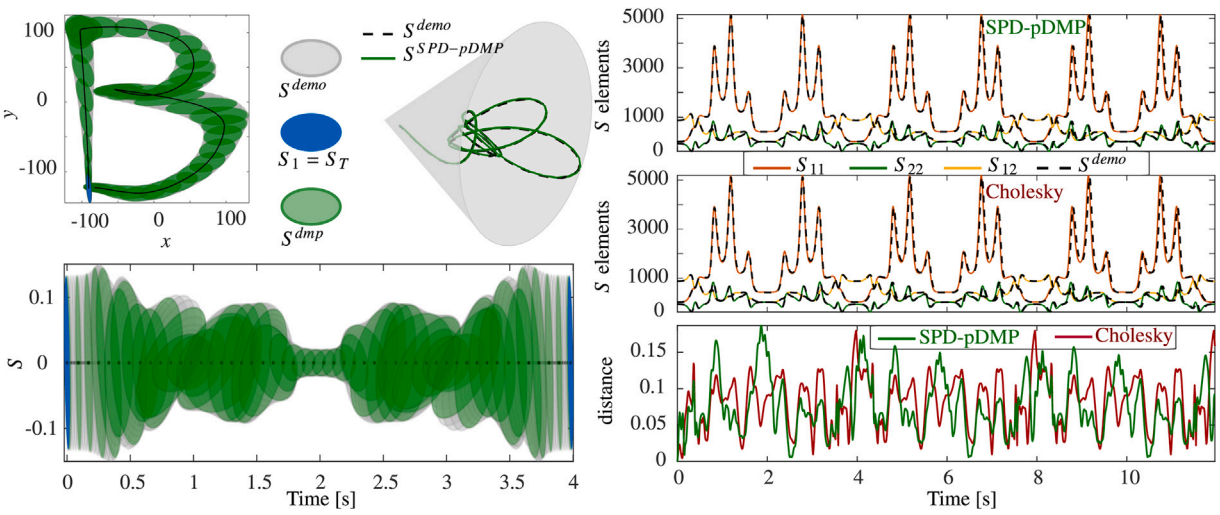


Fig. 6. Results obtained for the B-Letter covariance data. *Top-Left*: Demonstrated (gray ellipses) and learned with SPD-pDMP (green ellipses) covariance matrices are plotted along the Cartesian trajectory (black solid line). The blue ellipses represent the initial/final covariance. *Top-Middle*: Representation of the demonstrated (dashed black line) and learned with SPD-pDMP (green solid line) covariance matrices in the cone of SPD manifold. *Bottom-Left*: Demonstrated and learned covariance matrices shown over one period of time to illustrate the periodic nature of the considered profile. Covariance matrices are rescaled for better visualization. *Right-panels*: Learning results obtained over three periods using SPD-pDMP and Cholesky decomposition. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- Improved Encoding Accuracy: Our proposed Riemannian periodic DMPs (Quat-pDMP, Rot-pDMP, SSPD-pDMP) have shown a high level of accuracy in encoding and reproducing periodic trajectories with specialized geometric constraints. This improvement in accuracy is critical for tasks that require precise manipulation, such as surgical robots or industrial automation, where errors can lead to significant issues.
- Geometric Consistency: By incorporating tools from Riemannian geometry, our approach ensures that the encoded trajectories respect the underlying geometric constraints of the data (e.g., the unit norm for quaternions, positive definiteness for SPD matrices). This consistency is essential for maintaining the stability and reliability of robot actions, especially in complex environments.
- Reduction of Pre- and Post-Processing Steps: Traditional methods often require additional steps to normalize or convert data to ensure geometric constraints are met. Our approach eliminates the need for such steps, streamlining the learning process and reducing computational overhead. This efficiency can accelerate the deployment of learning-based robotic systems in real-world applications.
- Adaptability to Different Types of Data: The ability to handle different types of data (quaternions, rotation matrices, SPD matrices) within a unified framework makes our approach versatile and applicable to a wide range of robotic tasks. This adaptability can facilitate the development of more general-purpose robots capable of performing diverse manipulation tasks.

6. Robot experiments

This section illustrates two experiments[2] conducted on two different robots, namely a Franka Emika Panda and a Kuka IIWA equipped with qb SoftHand.

6.1. Operating a drilling machine

Operating a drilling machine by a rotary handle requires continuous adjustments of both orientation and stiffness. As shown in Fig. 1 (*left*), the handle must be rotated to control the vertical movement of the drill.

[2] A video of the experiments has been submitted as supplementary material.

For this experiment, we govern the robot interaction behavior with a Cartesian impedance controller defined as:

$$\begin{aligned}\mathbf{f}_p &= \mathbf{K}_p\left(\mathbf{x}_d - \mathbf{x}_a\right) + \mathbf{D}_p\left(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}_a\right),\\ \mathbf{f}_o &= \mathbf{K}_o \operatorname{Log}^{q}_{\mathbf{Q}_d}\left(\mathbf{Q}_a\right) + \mathbf{D}_o\left(\boldsymbol{\omega}_d - \boldsymbol{\omega}_a\right),\end{aligned} \tag{38}$$

where the subscript p stands for position and o for orientation. $\mathbf{x}_d$ and $\mathbf{x}_a$ are desired and actual positions of the robot's EEF, and $\mathbf{q}_d$ and $\mathbf{q}_a$ are desired and actual UQ orientations. Desired and actual linear velocities are indicated as $\dot{\mathbf{x}}_d$ and $\dot{\mathbf{x}}_a$, while the angular ones are $\dot{\boldsymbol{\omega}}_d$ and $\dot{\boldsymbol{\omega}}_a$. $\mathbf{K}_{p,o}$ and $\mathbf{D}_{p,o}$ are the robot stiffness and damping matrices expressed in the robot base frame. We used $\mathbf{K}_o = 150\,\mathbf{I}$ Nm/rad, while $\mathbf{K}_p$ was generated online by the proposed method, as detailed later in this section. The matrices $\mathbf{D}_p$ and $\mathbf{D}_o$ were obtained from $\mathbf{K}_p$ and $\mathbf{K}_o$ by the double diagonalization design [40], briefly described in Section 3.6.

Motion variables $\mathbf{x}_d$ and $\dot{\mathbf{x}}_d$ were learned from a kinesthetic demonstration using a classical periodic DMP. The same demonstration was used to learn $\mathbf{Q}_d$ and $\dot{\boldsymbol{\omega}}_d$ by Quat-pDMP. The desired and executed pose trajectories are shown in the first two rows of Fig. 7. To obtain the desired stiffness profile, we considered that most of the EEF linear motion occurs in the x-y plane (in the EEF frame). Moreover, the robot was constrained to move along a circular path defined by the rotary wheel (see Fig. 1). Given these observations and the set of demonstrated positions $\{\mathbf{x}_{d,t}\}_{t=1}^{T}$, we defined the EEF stiffness matrices for $t = 1, \ldots, T$ as

$$\mathbf{K}^{ee}_{p,t} = \begin{bmatrix} \mathbf{R}(\theta_t)\mathbf{K}_0\mathbf{R}(\theta_t)^{\top} & \mathbf{0} \\ \mathbf{0}^{\top} & k_{max} \end{bmatrix}, \tag{39}$$

where $\mathbf{R}(\theta_t)$ is the 2D rotation matrix defined by the angle θ_t between $\mathbf{x}_{d,t}$ and the initial position $\mathbf{x}_{d,0}$. The initial 2D stiffness was $\mathbf{K}_0 = \operatorname{diag}([k_{max}, k_{min}])$, where $k_{max} = 3000\,\text{N/m}$ and $k_{min} = 300\,\text{N/m}$. The stiffness profile (39) was designed to vary in x and y directions, while z was kept constant at the maximum value to stay in contact with the rotary wheel.

At the beginning of the motion, the robot exerted maximum force (highest stiffness) along the x direction. During the motion, the rotation $\mathbf{R}(\theta_t)$ continuously modified the initial stiffness matrix $\mathbf{K}_0$ to have the highest (lowest) stiffness along the y (x) direction as the drill approached full depth. After this point, the stiffness along y (x) started to decrease (increase) and returned to the initial value at the end of the motion, i.e. $\mathbf{K}^{ee}_{p,T} = \mathbf{K}^{ee}_{p,1} = \mathbf{K}_0$. The desired full 3D stiffness matrices were then computed as $\mathbf{K}_{p,t} = \mathbf{R}_t\mathbf{K}^{ee}_{p,t}\mathbf{R}_t{}^{\top}$, where $\mathbf{R}_t$ was the rotation

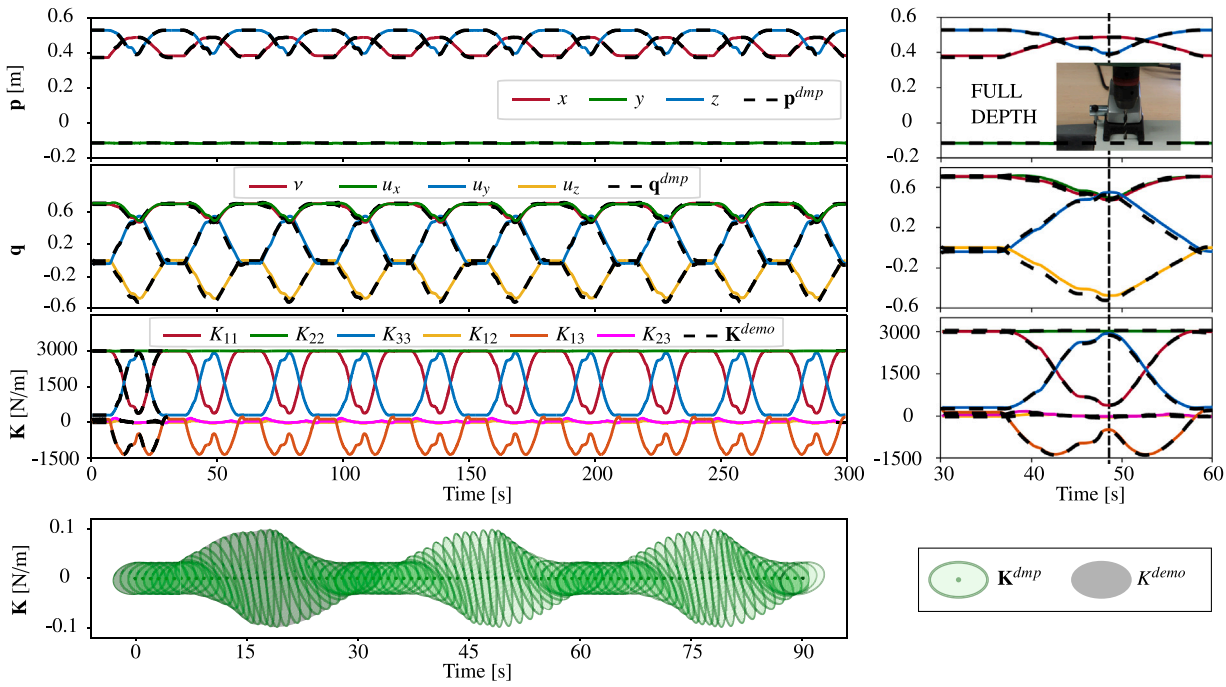


Fig. 7. Results of operating a drilling machine experiment on a Franka Emika Panda. Top two rows show learned (dashed lines) and measured (solid lines) position $\mathbf{p}$ and orientation $\mathbf{q}$ of the robot's EEF. The third row shows the six relevant elements of the stiffness matrix $\mathbf{K}$ (desired and learned). The left column depicts the entire task execution (10 periods), while the right column exhibits a particular zoomed-in section (1 period). The moment when the drill reached full depth is highlighted by the text "FULL DEPTH" and is further illustrated by the experiment photo. The bottom graph shows the desired (gray) and learned (green) stiffness profiles (scaled between $[-0.1, 0.1]$) in the x-z plane, where the robot linear motion takes place. Note that we learn a full 3D stiffness profile, as shown in the third row, but we show also the profile in the x-z plane for better visualization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
State-to-action collaborative model for sawing task.

Task State		Action
$\mathbf{x}_a$ (saw position)	$\dot{\mathbf{x}}_a$ (saw velocity)	$\mathbf{K}$ (robot stiffness)
close to the robot	moving toward the robot	decreasing
close to the robot	moving toward the human	compliant (zero)
close to the human	moving toward the human	compliant (zero)
close to the human	moving toward the robot	increasing

matrix between the robot's EEF and base at time t. $\mathbf{K}_{p,t}$ were used as reference stiffness profile and encoded by SPD-pDMP. The 6 independent components of the learned stiffness profile are shown in the third row of Fig. 7, while the last row shows the stiffness profile in x-z plane (*i.e.,* where the robot motion takes place). Nevertheless, we learn a full 3D stiffness matrix as shown in the third row of the figure. These results show that the proposed approach accurately encoded the desired profiles while fulfilling the underlying geometric constraints, *i.e.,* unit norm in variable orientation data, and the symmetry, positiveness, and definiteness of impedance matrices.

6.2. Collaborative human–robot sawing

To demonstrate the ability of the proposed method to encode and reproduce rotated (non-diagonal) stiffness matrices in a realistic periodic collaborative task, we performed an experiment on collaborative human–robot sawing. As shown in Fig. 1 (right), the task involved a human and a robot cutting a metal bar with a saw. The specific state-to-action collaborative model designed for the sawing task is given in Table 1. This strategy conforms with established studies on collaborative human–robot sawing [4]. While the designed model in Table 1 is specific to sawing, the proposed state-to-action approach is general and can be applied to other tasks.

The resulting robot stiffness strategy from the developed state-to-action collaborative model was used in combination with the proposed R-pDMP method to encode the changing robot stiffness matrix and facilitate the desired collaborative behavior. To demonstrate the ability to encode and reproduce non-diagonal stiffness matrices, we rotated the sawing axis and the task frame by 45 deg around the z-axis with respect to the robot base frame. In this task, the z-axis did not exhibit variable stiffness behavior, as it was involved in keeping contact with the object, so only x-axis and y-axis stiffness components were encoded by the proposed R-pDMPs method. The robot reference position was set near the point where the robot should pull back the saw when it was its turn to pull. The encoded DMPs were then reproduced online during the human–robot collaborative sawing to exhibit the appropriate stiffness behavior and facilitate the task execution.

Here, the robot interaction behavior was governed by a hybrid force/impedance controller defined as:

$$\mathbf{F} = \mathbf{F}_{for} + \mathbf{F}_{imp}, \tag{40}$$

where $\mathbf{F}_{for}$ maintains contact in the z-axis through a PI controller,[3] $\mathbf{F}_{imp}$ controls the sawing motion in the x-axis by the impedance controller defined as $\mathbf{F}_{imp} = \mathbf{K}\left(\mathbf{x}_d - \mathbf{x}_a\right) + \mathbf{D}\left(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}_a\right)$, where $\mathbf{x}_d$ and $\mathbf{x}_a$ are the desired and actual positions of the robot's EEF, respectively, and $\mathbf{K}$ and $\mathbf{D}$ are the robot stiffness and damping matrices expressed in the robot base frame. The matrix $\mathbf{K}$ was generated online by the proposed method, while $\mathbf{D}$ was obtained by the double diagonalization design [40], briefly described in Section 3.6.

The results of the collaborative human–robot sawing experiment are shown in Fig. 8. We observed that a stable collaborative behavior was achieved very quickly and maintained until the object was cut off at 88 seconds. After the cut-off moment, the robot remained stable at its reference position, waiting for any potential further human input. The position and force graphs show that the generated stiffness matrix

[3] Note that the derivative term was not used since sawing produces noisy force measurements and numerical derivation of such signal is not very useful.

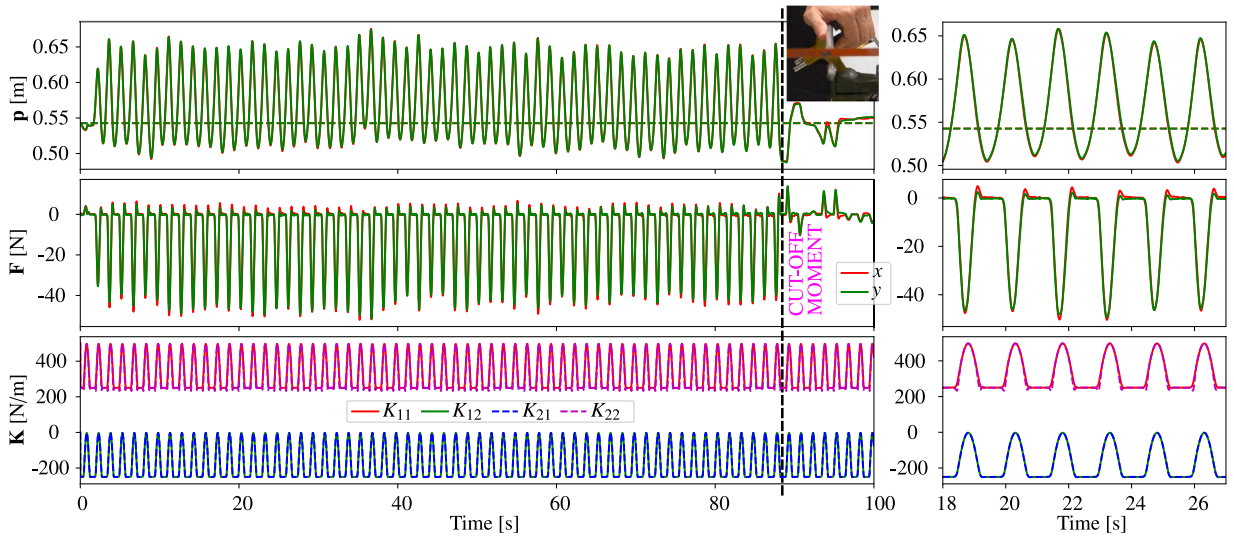


Fig. 8. Results of the collaborative sawing experiment. Top graphs show the measured position **p** of the robot's EEF. Middle graphs show the interaction force **F**. Bottom graphs show the relevant elements of the stiffness **K**. The left column depicts the entire task execution, while the right column exhibits a particular zoomed-in section. The moment when the object was cut off and the task was completed is highlighted by the magenta text "cut-off moment" and is further illustrated by the experiment photo showing the cut-off piece falling down. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

produced nearly identical position displacements and forces in both x-axis (red lines) and y-axis (green lines), corresponding to the task frame being rotated by 45 deg. This alignment indicates that the robot accurately followed the desired trajectory while maintaining the appropriate stiffness profile, ensuring effective collaboration with the human partner.

In more detail, the position graph shows a smooth and consistent motion throughout the sawing task, with minimal deviation from the desired trajectory. The smooth periodic sawing indicated that the robot properly performed the transitions between different stages of the collaborative tasks and continuously maintained coordination with the human at all times. This consistency is crucial for tasks requiring precise and repetitive movements. The force graph demonstrates that the robot applied a consistent force along the task direction, adapting its stiffness in response to the varying task state, which enhanced both the safety and efficiency of the interaction.

The experiment also highlighted the robustness of our state-to-action collaborative model (Fig. 8-*bottom*) approach in dynamically adjusting the stiffness matrix. When the saw was moving toward the human, the robot decreased the stiffness and then remained compliant in order not to oppose the human's action. When the saw reached the human end, the roles were reversed, and the human became complaint while the robot increased its stiffness to pull the saw back to its own end. We can also see that the elements of the stiffness matrix during the experiments adhered to the symmetric positive-definite matrix condition.

6.3. Broader implications for human–robot interaction

- Enhanced Safety and Efficiency: The ability to dynamically adjust the robot's stiffness based on the task state significantly enhances the efficiency of human–robot interactions for periodic tasks. By adapting to the varying demands of the task in real-time, the robot can respond more appropriately to human actions, ensuring adaptability and coordination.
- Reduction of Complexity and Intrusiveness: Traditional methods often rely on complex and intrusive measurements such as EMG or EIT to infer human intent. Our state-to-action approach eliminates the need for such measurements by directly using the task state, simplifying the setup and making it more practical for real-world applications. This reduction in complexity and intrusiveness can facilitate wider adoption of collaborative robots in various industries.
- Versatility and Generalizability: The state-to-action collaborative model demonstrated in the sawing task is generalizable to a wide range of other collaborative tasks. This versatility makes our approach applicable to various domains, including manufacturing, healthcare, and service industries, where robots need to perform diverse tasks in close collaboration with humans.
- Facilitation of Advanced Research: The insights gained from our experiments provide a foundation for further research into advanced human–robot interaction strategies. Researchers can build on our state-to-action model to develop more sophisticated algorithms that further enhance the adaptability and responsiveness of collaborative robots.

7. Discussion

The following discussion highlights the primary findings from both our simulation and real robot experiments, elucidating their implications for the field of human–robot interaction, and addresses the limitations of our proposed approach. The simulations demonstrated that the proposed Riemannian periodic R-pDMPs (Quat-pDMP, Rot-pDMP, SPD-pDMP) accurately encoded and reproduced periodic trajectories with specialized geometric constraints. The results showed that our approach maintained geometric consistency, such as unit norm for quaternions and positive definiteness for SPD matrices, without the need for additional normalization or conversion steps. The comparison with baseline methods highlighted the superiority of our approach in terms of accuracy and stability. The proposed methods effectively reduced the error introduced during the reproduction of periodic trajectories, thereby ensuring precise and reliable performance.

The collaborative human–robot sawing experiment demonstrated the practical applicability of our state-to-action collaborative model. The robot successfully adapted its stiffness based on the task state, achieving smooth and consistent motion throughout the sawing task. This adaptability improved both the safety and efficiency of the interaction. The experiment validated that our approach could handle complex, non-diagonal stiffness profiles in real-time, ensuring that the robot could respond appropriately to dynamic changes in the interaction dynamics. This capability is particularly beneficial for tasks requiring close collaboration between humans and robots.

Our findings have several important implications for advancing human–robot interaction: Dynamically adjusting the robot's stiffness based on the task state enhances safety and efficiency, allowing the robot to better respond to human actions and reduce injury risks. Our method eliminates the need for complex measurements like EMG or EIT, simplifying the setup and making it more practical for real-world applications, thus facilitating wider adoption of collaborative robots. The state-to-action model is generalizable to various tasks beyond sawing, making it applicable to manufacturing, healthcare, and service industries, where robots need to collaborate closely with humans. The precise and consistent execution of tasks enhances the performance and reliability of robotic systems. Finally, the insights from our experiments provide a foundation for advanced human–robot interaction strategies, enabling the development of more sophisticated algorithms that enhance robot adaptability and responsiveness.

Our current work focuses on specific types of Riemannian manifolds (e.g., $\mathcal{S}^3$, $\mathcal{SO}(3)$, and $\mathcal{S}^m_{++}$). Extending our approach to other types of manifolds and ensuring the same level of performance and reliability is an important area for future exploration. For instance, investigating the application of our method to Grassmann manifolds (used for subspace learning) and hyperbolic manifolds (beneficial for representing hierarchical data) could open new avenues for research and application. Moreover, one of the main challenges with our approach is the computational complexity associated with the calculation of exponential and logarithmic maps on Riemannian manifolds. These operations are essential for ensuring geometric consistency but can be computationally intensive, particularly for high-dimensional data. Future work will focus on optimizing these computations through more efficient algorithms and leveraging parallel processing techniques. Additionally, we will explore approximation methods that can reduce computational load while maintaining accuracy.

8. Conclusions

In this paper, we proposed a novel periodic DMP formulation that extends the classical formulation by incorporating tools from Riemannian geometry to encode and reproduce periodic robotic skills with geometric constraints. Moreover, we introduced a novel state-to-action collaborative model that dynamically adjusts end-effector stiffness based on task-specific states, enhancing safety and efficiency in human–robot interactions.

Our method was validated through extensive simulations and real robot experiments, demonstrating superior performance compared to state-of-the-art methods. The ability to maintain geometric consistency, reduce the need for pre- and post-processing steps, and dynamically adapt to changing task requirements highlights the potential of our approach for advancing human–robot collaboration.

Future work will address the identified limitations, focusing on optimizing computational performance and generalizing to other manifolds. By refining and extending our approach, we aim to further enhance collaborative robots' capabilities and integration into various applications.

CRediT authorship contribution statement

Fares Abu-Dakka: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization, Formal analysis, Funding acquisition, Project administration. **Matteo Saveriano:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Conceptualization. **Luka Peternel:** Writing – review & editing, Writing – original draft, Validation, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by Basque Government (ELKARTEK) projects Proflow KK-2022/00024 and HELDU KK-2023/00055, and by the European Union project INVERSE (GA No. 101136067).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2024.104763.

References

[1] A.J. Ijspeert, J. Nakanishi, S. Schaal, Learning rhythmic movements by demonstration using nonlinear oscillators, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, Lausanne, Switzerland, 2002, pp. 958–963.

[2] A. Rai, F. Meier, A. Ijspeert, S. Schaal, Learning coupling terms for obstacle avoidance, in: IEEE-RAS International Conference on Humanoid Robots, IEEE, Madrid, Spain, 2014, pp. 512–518.

[3] A. Gams, T. Petrič, M. Do, B. Nemec, J. Morimoto, T. Asfour, A. Ude, Adaptation and coaching of periodic motion primitives through physical and visual interaction, Robot. Auton. Syst. 75 (2016) 340–351.

[4] L. Peternel, N. Tsagarakis, D. Caldwell, A. Ajoudani, Robot adaptation to human physical fatigue in human–robot co-manipulation, Auton. Robots 42 (5) (2018) 1011–1021.

[5] L. Peternel, T. Petrič, J. Babič, Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation, Auton. Robots 42 (1) (2018) 1–17.

[6] E. Rückert, A. d'Avella, Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems, Front. Comput. Neurosci. 7 (2013).

[7] P. M. Wensing, J.-J. Slotine, Sparse control for dynamic movement primitives, IFAC-PapersOnLine 50 (1) (2017) 10114–10121.

[8] P. Pastor, L. Righetti, M. Kalakrishnan, S. Schaal, Online movement adaptation based on previous sensor experiences, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 2011, pp. 365–371.

[9] A. Ude, B. Nemec, T. Petric, J. Morimoto, Orientation in cartesian space dynamic movement primitives, in: IEEE International Conference on Robotics and Automation, Hong Kong, China, 2014, pp. 2997–3004.

[10] F.J. Abu-Dakka, V. Kyrki, Geometry-aware dynamic movement primitives, in: IEEE International Conference on Robotics and Automation, Paris, France (Online), 2020, pp. 4421–4426.

[11] F.J. Abu-Dakka, M. Saveriano, L. Peternel, Periodic DMP formulation for quaternion trajectories, in: IEEE International Conference of Advanced Robotics, Ljubljana, Slovenia, 2021, pp. 658–663.

[12] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors, Neural Comput. 25 (2) (2013) 328–373.

[13] L. Koutras, Z. Doulgeri, A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space, in: Conference on Robot Learning, Osaka, Japan, 2020, pp. 293–302.

[14] M. Saveriano, F.J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey, Int. J. Robot. Res. 42 (13) (2023) 1133–1184.

[15] A. Paraschos, C. Daniel, J. Peters, G. Neumann, Probabilistic movement primitives, in: Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, United States, 2013, pp. 2616–2624.

[16] S. Dutta, L. Behera, S. Nahavandi, Skill learning from human demonstrations using dynamical regressive models for multitask applications, IEEE Trans. Syst. Man Cybern.: Syst. 51 (1) (2018) 659–672.

[17] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, C. Xu, Fast and stable learning of dynamical systems based on extreme learning machine, IEEE Trans. Syst. Man Cybern.: Syst. 49 (6) (2017) 1175–1185.

[18] S. Calinon, D. Bruno, D.G. Caldwell, A task-parameterized probabilistic model with minimal intervention control, in: IEEE International Conference on Robotics and Automation, Hong Kong, China, 2014, pp. 3339–3344.

[19] Y. Huang, F.J. Abu-Dakka, J. Silvério, D.G. Caldwell, Toward orientation learning and adaptation in cartesian space, IEEE Trans. Robot. 37 (1) (2021) 82–98.

[20] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, Learning and generalization of motor skills by learning from demonstration, in: IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009, pp. 763–768.

[21] J. Silvério, L. Rozo, S. Calinon, D.G. Caldwell, Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg, Germany, 2015, pp. 464–470.
[22] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer Publishing Company, Incorporated, 2010.
[23] S. Kim, R. Haschke, H. Ritter, Gaussian mixture model for 3-dof orientations, Robot. Auton. Syst. 87 (2017) 28–37.
[24] M.J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, D.G. Caldwell, An approach for imitation learning on Riemannian manifolds, IEEE Robot. Autom. Lett. 2 (3) (2017) 1240–1247.
[25] L. Dodero, H.Q. Minh, M. San Biagio, V. Murino, D. Sona, Kernel-based classification for brain connectivity graphs on the Riemannian manifold of positive definite matrices, in: International Symposium on Biomedical Imaging, Brooklyn, NY, USA, 2015, pp. 42–45.
[26] S. Herath, M. Harandi, F. Porikli, Learning an invariant Hilbert space for domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017, pp. 3845–3854.
[27] S. Calinon, Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control, IEEE Robot. Autom. Mag. 27 (2) (2020) 33–45.
[28] N. Jaquier, L. Rozo, D.G. Caldwell, S. Calinon, Geometry-aware manipulability learning, tracking, and transfer, Int. J. Robot. Res. 40 (2–3) (2021) 624–650.
[29] A. Ajoudani, N. Tsagarakis, A. Bicchi, Tele-impedance: Teleoperation with impedance regulation using a body–machine interface, Int. J. Robot. Res. 31 (13) (2012) 1642–1656.
[30] K. Kronander, A. Billard, Learning compliant manipulation through kinesthetic and tactile human-robot interaction, IEEE Trans. Hapt. 7 (3) (2014) 367–380.
[31] L. Roveda, J. Maskani, P. Franceschi, A. Abdi, F. Braghin, L. Molinari Tosatti, N. Pedrocchi, Model-based reinforcement learning variable impedance control for human-robot collaboration, J. Intell. Robot. Syst. 100 (2) (2020) 417–433.
[32] X. Chen, N. Wang, H. Cheng, C. Yang, Neural learning enhanced variable admittance control for human–robot collaboration, IEEE Access 8 (2020) 25727–25737.
[33] E. Zheng, Y. Li, Z. Zhao, Q. Wang, H. Qiao, An electrical impedance tomography based interface for human–robot collaboration, IEEE/ASME Trans. Mechatronics 26 (5) (2021) 2373–2384.
[34] A. Gams, A. Ijspeert, S. Schaal, J. Lenarčič, On-line learning and modulation of periodic movements with nonlinear dynamical systems, Auton. Robots 27 (1) (2009) 3–23.
[35] T. Petrič, A. Gams, A.J. Ijspeert, L. Žlajpah, On-line frequency adaptation and movement imitation for rhythmic robotic tasks, Int. J. Robot. Res. 30 (14) (2011) 1775–1788.
[36] S. Schaal, C.G. Atkeson, Constructive incremental learning from only local information, Neural Comput. 10 (8) (1998) 2047–2084.
[37] M. Fréchet, Les éléments aléatoires de nature quelconque dans un espace distancié, in: Annales de l'institut Henri Poincaré, vol. 10, (no. 4) 1948, pp. 215–310.
[38] Q. Rentmeesters, A gradient method for geodesic data fitting on some symmetric Riemannian manifolds, in: IEEE Conference on Decision and Control and European Control Conference, IEEE, Orlando, FL, USA, 2011, pp. 7141–7146.
[39] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, Int. J. Comput. Vis. 66 (1) (2006) 41–66.
[40] A. Albu-Schaffer, C. Ott, U. Frese, G. Hirzinger, Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms, in: International Conference on Robotics and Automation, Taipei, Taiwan, 2003, pp. 3704–3709.
[41] S. Fiori, I. Cervigni, M. Ippoliti, C. Menotta, Synthetic nonlinear second-order oscillators on Riemannian manifolds and their numerical simulation, Discrete Contin. Dyn. Syst. Ser. B 27 (3) (2022) 1227–1262.
[42] N. Boumal, P.-A. Absil, A discrete regression method on manifolds and its application to data on SO (n), IFAC Proc. Vol. 44 (1) (2011) 2284–2289.
[43] P.-Y. Gousenbourger, E. Massart, P.-A. Absil, Data fitting on manifolds with composite Bézier-like curves and blended cubic splines, J. Math. Imaging Vision 61 (5) (2019) 645–671.

Fares J. Abu-Dakka received a B.Sc. in Mechanical Engineering from Birzeit University, Palestine (2003), and advanced degrees (DEA and Ph.D.) in robotics motion planning from the Polytechnic University of Valencia (UPV), Spain (2006, 2011), in addition to M.Sc. in Biomedical Engineering from UPV (2015). His postdoctoral journey began at the Jozef Stefan Institute, Slovenia, in 2012. From 2013 to 2016, he was a Visiting Professor at Carlos III University of Madrid, Spain, followed by a postdoctoral role at the Istituto Italiano di Tecnologia (IIT) from 2016 to 2019. He was a Research Fellow at Aalto University (2019–2022) before joining the Technical University of Munich as a Senior Scientist in 2022, where he led the Robot Learning group at MIRMI. Currently, he is a Lecturer and Researcher at Mondragon Unibertsitatea, Spain. His research spans control theory, differential geometry, and machine learning, with a focus on improving robot manipulation performance and safety. He also serves as an Associate Editor for IEEE Robotics and Automation Letters (RA-L) and IEEE Transactions on Robotics (T-RO). Webpage: https://sites.google.com/view/abudakka/

Matteo Saveriano received his B.Sc. and M.Sc. degree in automatic control engineering from University of Naples, Italy, in 2008 and 2011, respectively. He received his Ph.D. from the Technical University of Munich in 2017. Currently, he is an assistant professor at the Department of Industrial Engineering (DII), University of Trento, Italy. Previously, he was an assistant professor at the University of Innsbruck and a post-doctoral researcher at the German Aerospace Center (DLR). He is an Associate Editor for RA-L and IJRR. He is the coordinator of the EU project INVERSE. His research activities include robot learning, human–robot interaction, understanding and interpreting human activities. Webpage: https://matteosaveriano.weebly.com/

Luka Peternel received a PhD in robotics from the Faculty of Electrical Engineering, University of Ljubljana, Slovenia in 2015. He conducted his PhD studies at the Department for Automation, Biocybernetics and Robotics, Jožef Stefan Institute in Ljubljana from 2011 to 2015, and at the Department of Brain–Robot Interface, ATR Computational Neuroscience Laboratories in Kyoto, Japan in 2013 and 2014. He was with the Human–Robot Interfaces and Physical Interaction Lab, Advanced Robotics, Italian Institute of Technology in Genoa, Italy from 2015 to 2018. Since 2019, He is an Assistant Professor at the Department of Cognitive Robotics, Delft University of Technology in the Netherlands.