A correct formulation for the Orientation Dynamic Movement Primitives for robot control in the Cartesian space

Leonidas Koutras
Automation & Robotics Lab
Dept. of Electrical & Computer Engineering
Aristotle University of Thessaloniki, Greece
kleonidas@ece.auth.gr

Zoe Doulgeri
Automation & Robotics Lab
Dept. of Electrical & Computer Engineering
Aristotle University of Thessaloniki, Greece
doulgeri@eng.auth.gr

Abstract: Dynamic movement primitives (DMP) are an efficient way for learning and reproducing complex robot behaviors. A singularity free DMP formulation for orientation in the Cartesian space is proposed by Ude et al. [1] and has been largely adopted by the research community. In this work, we demonstrate the undesired oscillatory behavior that may arise when controlling the robot's orientation with this formulation, producing a motion pattern highly deviant from the desired and highlight its source. A correct formulation is then proposed that alleviates such problems while guaranteeing generation of orientation parameters that lie in $SO(3)$. We further show that all aspects and advantages of DMP including ease of learning, temporal and spatial scaling and the ability to include coupling terms are maintained in the proposed formulation. Simulations and experiments with robot control in $SO(3)$ are performed to demonstrate the performance of the proposed formulation and compare it with the previously adopted one.

Keywords: Dynamic Movement Primitives, Control in Orientation Trajectories

1 Introduction

Inspired by biological systems, complex, goal directed kinematic behaviors are in recent years modeled by dynamical systems which are then utilized via on-line integration as motion generators for robot control. A popular form of dynamical system is the DMP which was introduced and later revisited by Ijspeert et al. [2], [3]. Their basic form is composed of a linear second order part which ensures convergence to a specified attractor point and a nonlinear forcing term which allows modeling of more complex movements. The latter usually consists of a linear combination of basis functions (most commonly Gaussians) with parameters (weights) that can be learned without affecting the system's convergence. DMP have a number of useful properties including robustness to perturbations; they allow spatial and temporal scaling via a non-explicit time representation and can be manipulated to adapt to different situations via coupling terms during the execution of the movement [4], [5]. When used in Cartesian space the issue of orientation representation arises since it is known that unlike the position there is no minimal representation of orientation that is singularity free. Minimal representations introduce artificial discontinuities that should be avoided. Orientations are defined in the $SO(3)$ group which is a three dimensional manifold that can be represented by rotation matrices and quaternions. In the field of robotics, quaternions are generally preferred over rotation matrices, as they require less parameters and have been used in many applications [6], [7], [8]. An approach for imitation learning in Riemannian manifolds can be found in Zeestraten et al. [9].

The generalization of the DMP framework to orientation trajectories with quaternion representations has been introduced by Pastor et al. [10] and expanded by Ude et al. [1]. The approach in [10] usually leads to slow convergence as pointed out in [1]. In contrast, faster convergence characterizes the formulation proposed in [1]. Since its publication, this formulation has been the dominant approach for orientation DMP robot learning and control.

In this work we show that when controlling the robot's orientation with this dominant formulation undesired oscillatory behaviors may arise. As we show in Section 2 the orientation DMP, unlike the positional formulation, does not yield a linear tracking system. Then in Section 3 we propose a correct formulation which alleviates this problem. We evaluate our theoretical results with simulations and experiments in Sections 4 and 5 respectively and draw conclusions in Section 6.

2 DMP Preliminaries and Motivation

A DMP for a single degree of freedom point to point movement y is defined by the following second order dynamical system:

$$\begin{aligned} \tau\dot{z} &= \alpha_z(\beta_z(g-y)-z)+(g-y_0)f(x) \\ \tau\dot{y} &= z \end{aligned} \tag{1}$$

where $y, \dot{y}$ represent position and velocity, y_0 and g are the initial and goal position, α_z, β_z are the positive gains of the linear part of the system, x is the phase variable introduced to avoid explicit time dependency, z is the scaled velocity, $\tau > 0$ is a temporal scaling term and $(g-y_0)f(x)$ is a nonlinear forcing term which allows the modeling of complex motion patterns. By setting $\alpha_z = 4\beta_z$ the linear part of (1) becomes critically damped and the state y, z converges monotonically to the unique equilibrium goal $y = g, z = 0$. The function $f(x)$ is defined as follows:

$$f(x) = \frac{\sum_i w_i \Psi_i(x)}{\sum_i \Psi_i(x)} x \tag{2}$$

where $\Psi_i(x), i = 1, ..., N$ are Gaussian kernel functions:

$$\Psi_i(x) = \exp[-h_i(x-c_i)^2] \tag{3}$$

with c_i being the centers of the Gaussians distributed along the phase of the motion and h_i their inverse widths. System (1) is known as the transformation system. The phase variable's x evolution is governed by the canonical system:

$$\tau\dot{x} = -\alpha_x x \tag{4}$$

with α_x a positive gain and $x_0 = 1$ the initial value of x. Notice that x converges exponentially to 0.

In the Cartesian space we represent each coordinate's position trajectory with a transformation system of the form (1) but with a common canonical system to synchronize them.

Given a demonstrated trajectory $y_d, \dot{y}_d, \ddot{y}_d$ from initial position $y_{0,d}$, to final position g_d the forcing term (2) is trained using Locally Weighted Regression (LWR) in order to generate the desired motion:

$$f(x) = \frac{1}{g_d - y_{0,d}}(\tau^2\ddot{y}_d - \alpha_z(\beta_z(g_d - y_d) - \tau\dot{y}_d)) \tag{5}$$

Parameter τ is usually set to the time duration of the demonstrated motion $\tau = T_d$ or to the normalized value $\tau = 1$.

Assuming a training error ϵ, $f(x)$ in (1) can be substituted by the sum of the right hand side of (5) plus ϵ to yield the following system:

$$\tau^2(\ddot{y} - s_g\ddot{y}_d) + \tau\alpha_z(\dot{y} - s_g\dot{y}_d) + \alpha_z\beta_z(y - s_g y_d) = \alpha_z\beta_z(g - s_g g_{0,d}) + (g - y_0)\epsilon \tag{6}$$

where $s_g = \frac{g-y_0}{g_d - y_{0,d}}$ is a spatial scaling parameter to accommodate new goals. By further substituting g in (6) from $g = s_g(g_d - y_{0,d}) + y_0$ yields the tracking error $y - s_g y_d$ dynamics, which is linear, stable and convergent with a rate dictated by the choice of the DMP parameters α_z, β_z and τ:

$$\tau^2(\ddot{y} - s_g\ddot{y}_d) + \tau\alpha_z(\dot{y} - s_g\dot{y}_d) + \alpha_z\beta_z(y - s_g y_d) = \alpha_z\beta_z(y_0 - s_g y_{0,d}) + s_g(g_d - y_{0,d})\epsilon \tag{7}$$

Notice that the first term in the right hand side of (7) is a bias that transfers the scaled trajectory to the actual initial position while the second term dictates the steady state value of the tracking error owing to the training error; the smaller is the training error the more accurate is the following of the reference trajectory.

Let us now turn our attention to the dominant formulation for orientation DMP suggested in [1]. In this formulation, the following quaternion based transformation system is suggested:

$$\begin{aligned} \tau\dot{\eta} &= \alpha_z(\beta_z 2\log(Q_g * \overline{Q}) - \eta) + \mathrm{diag}(2\log(Q_g * \overline{Q}_0))F(x) \\ \tau\dot{Q} &= \tfrac{1}{2}\begin{bmatrix} 0 \\ \eta \end{bmatrix} * Q \end{aligned} \tag{8}$$

where Q represents the orientation as a unit quaternion, Q_0, Q_g are the initial and goal orientations, $\eta = \tau\omega \in \mathbb{R}^3$ represents the scaled angular velocity, $'*'$ denotes the quaternion product, $\overline{Q}$ denotes the quaternion conjugate which is equal to the inverse quaternion for unit quaternions and $2\log(Q_2 * \overline{Q}_1) \in \mathbb{R}^3$ expresses the rotation of Q_1 around a fixed axis to reach Q_2. The forcing term $F(x) \in \mathbb{R}^3$ in (8) is defined as in (2) for each orientation coordinate and is trained by utilizing (4) and (8) to encode the desired orientation trajectory provided by the demonstrated data $Q_d, \omega_d, \dot{\omega}_d$ with $Q_{0,d}$ and $Q_{g,d}$ the initial and goal orientations. This formulation is utilizing the exponential map for integrating (8) which is a distance preserving map between the tangent space and the manifold and its inverse called the logarithmic map. They both stem from the Lie Algebra of $SO(3)$. A brief introduction to the basics of Lie group theory and its connections with rigid body kinematics can be found in Murray et al. [11]. Basic quaternion definitions and operations used in this work are given in Appendix A.

We denote by e_Q the quaternion error between the current and goal orientations values which is given by:

$$e_Q = 2\log(Q_g * \overline{Q}) \tag{9}$$

This error can be written as a rotation around a fixed axis n by an angle $2(\theta_g - \theta)$ (see Appendix A). Then:

$$2\log(Q_g * \overline{Q}) = 2n(\theta_g - \theta) \tag{10}$$

By ignoring the forcing term, we can then write the first equation of (8) using (10) and substituting $\eta = \tau\omega = \tau\dot{\theta}n$ as follows:

$$\tau^2 \frac{d(\dot{\theta}n)}{dt} = \alpha_z(\beta_z n(\theta_g - \theta) - \tau\dot{\theta}n) \tag{11}$$

As claimed in [1], system (11) is linear but this is only true if the rotation axis is fixed which is not generally true since then $\dot{n} \neq 0$. For a demonstrated orientation trajectory, axis n is not fixed and hence the addition of the forcing term affects the linearity of the first part of the transformation system. We can demonstrate this claim by trying to derive the tracking error dynamics following a similar procedure with the one utilized for the position transformation system.

Given the demonstrated data, the forcing term is trained as before using LWR in order to generate the desired motion:

$$F(x) = \left(\mathrm{diag}(2\log(Q_{g,d} * \overline{Q}_{0,d}))\right)^{-1} \left(\tau^2\dot{\omega}_d - \alpha_z(\beta_z(2\log(Q_{g,d} * \overline{Q}_d) - \tau\omega_d))\right) \tag{12}$$

In order to derive the tracking error dynamics as before we first assume a training error ϵ_o so that $F(x)$ in (8) can be substituted by the sum of the right hand side of (12) plus ϵ_o to yield the following system:

$$\begin{gathered}\tau^2(\dot{\omega} - S_g\dot{\omega}_d) + \tau\alpha_z(\omega - S_g\omega_d) + \alpha_z\beta_z\left(2\log(Q_{g,d} * \overline{Q}_d) - S_g 2\log(Q_g * \overline{Q})\right) = \\ S_g\,\mathrm{diag}\left(2\log(Q_{g,d} * \overline{Q}_{0,d})\right)\epsilon_o\end{gathered} \tag{13}$$

with $S_g = \mathrm{diag}(2\log(Q_g * \overline{Q}_0))\left(\mathrm{diag}(2\log(Q_{g,d} * \overline{Q}_{0,d}))\right)^{-1}$ being a spatial scaling diagonal matrix term providing generalizations to new orientation goals. Notice that (13) is not a linear system of the tracking error. In fact, by setting for the sake of simplicity $Q_{g,d} = Q_g = [1\ 0\ 0\ 0]^T$ and $S_g = I_3$ which holds when $Q_{0,d} = Q_0$, the third term of the left hand side of (13) becomes $2(\log\overline{Q}_d - \log\overline{Q}) = 2(\log Q - \log Q_d)$ which is not a typical quaternion tracking error consistent with the rest of the formulation. It is indeed easy to establish that taking this term's time derivative does not yield $\omega - \omega_d$. Owing to this problem, this formulation may have serious consequences when used in robot control because it can generate undesired oscillatory behaviors highly deviant from the demonstrated trajectory as we clearly show in the rest of this paper by simulations and experiments.

To demonstrate the problems of the dominant DMP orientation formulation, we conducted two simulations of the system (4), (8). An orientation trajectory was recorded by physically guiding a KUKA robot and used to train an orientation DMP. The parameters used in the first simulation (case 1) were $\alpha_z = 60$, $\beta_z = 15$, $\tau = 1$, $N = 60$, $\alpha_x = 4.6052$, where N is the number of Gaussian kernels and α_x value ensures that x reaches 0.01 at the end of motion. The demonstrated trajectory was between initial orientation $Q_0 = [-0.0092\ -0.7126\ 0.7015\ 0.0090]^T$ and goal orientation $Q_g = [0.8104\ 0.3364\ 0.2141\ 0.4293]^T$. For the above initial and goal orientations

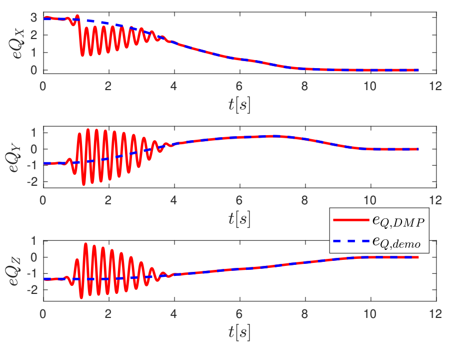

Figure 1: Case 1 Simulation Results: Evolution of quaternion error. The demonstration error trajectory is also depicted.

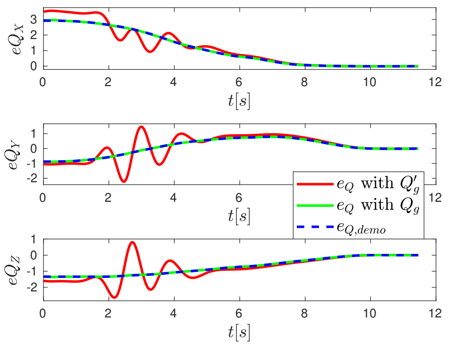

Figure 2: Case 2 Simulation Results: Evolution of quaternion error towards the original and new goal. The demonstration error trajectory is also depicted.

the initial orientation distance is expressed by the error $e_{Q,0} = [2.9199 \ -0.8740 \ -1.3364]^T$. In Figure 1, the evolution of the quaternion error is shown. It is evident that the non-linearities induce undesired oscillations, producing a motion pattern highly deviant from the desired. Those oscillations vary in frequency and amplitude according to the DMP parameter values. For larger values of $\frac{\alpha_z}{\tau}, \frac{\beta_z}{\tau}$ they become more intense and for lower they diminish. However, one does not know a priori the appropriate parameter values for the successful reproduction of every orientation pattern.

Even in the case of a set of parameters that lead to successful, oscillation-free execution, there is no assurance that those parameters will be able to maintain those performance characteristics towards a new goal. In the second simulation (case 2), the DMP is trained with the same trajectory but using parameters $\alpha_z = 15$, $\beta_z = 3.75$ and keeping the same values for τ, N, α_x. The trajectory is successfully executed but when we set a new goal at $Q'_g = [0.7442 \ 0.5414 \ -0.0343 \ 0.3897]^T$ the execution of (8) leads again to oscillations. In this case the same initial orientation is used corresponding to a scaled error $e'_{Q,0} = [3.5039 \ -1.0488 \ -1.6036]^T = 1.2e_{Q,0}$. Figure 2 shows this oscillatory behavior towards the new goal.

3 A correct DMP orientation formulation based on Quaternion Error

3.1 Proposed DMP formulation

To alleviate the problems identified in the previous Section with the currently adopted DMP orientation formulation, we propose the following formulation based on the quaternion error:

$$\begin{gathered} \tau \dot{z} = -\alpha_z(\beta_z e_Q + z) + \mathrm{diag}(2\log(Q_g * \overline{Q}_0))F(x) \\ \tau \dot{e}_Q = z \end{gathered} \tag{14}$$

where α_z, β_z are positive gains, e_Q, is the quaternion error given by (9), z is the scaled quaternion error velocity and $F(x)$ is trained as in (2) for each orientation coordinate. Notice that equation (14) has the typical form of a linear part and a non-linear forcing term and unlike (8) it follows the typical structure of the position DMP in the Cartesian space. The latter is easy to establish by setting $y = e_Q$, $g = 0_{3\times1}$, $g - y_0 = 2\log(Q_g * \overline{Q}_0)$. The proposed DMP is using the canonical system defined in (4).

For robot control we need to feed the robot with a reference quaternion and angular velocity. Notice that by integrating (14) we get the orientation error and its derivative from which we need to extract the quaternion and the angular velocity. To obtain the orientation, we solve (9) for Q:

$$Q = \overline{\exp\left(\frac{1}{2}e_Q\right)} * Q_g \tag{15}$$

The angular velocity can be provided by:

$$\omega = 2\,\mathrm{vec}(\dot{Q} * \overline{Q}) \tag{16}$$

where vec represents the vector part of the quaternion and $\dot{Q}$ can be evaluated by relating the derivative of the quaternion error with the derivative of the quaternion. Thus we firstly compute equations that connect the derivative of the quaternion logarithm with the derivative of the quaternion:

$$\dot{Q} = J_{\log Q}(Q)\frac{d\log(Q)}{dt} \tag{17}$$

$$\frac{d\log(Q)}{dt} = J_Q(Q)\dot{Q} \tag{18}$$

with $J_{\log Q}(Q)$, $J_Q(Q)$, being 4×3 and 3×4 Jacobians, whose elements depend on the current quaternion. Their derivation is detailed in Appendix B. They are given by :

$$J_{\log Q}(Q) = \begin{cases} \begin{bmatrix} -\sin\theta n^T \\ \frac{\sin\theta}{\theta}(I - nn^T) + \cos\theta nn^T \end{bmatrix}, \theta \neq 0 \\ \\ \begin{bmatrix} 0_{1\times 3} \\ I \end{bmatrix}, \theta = 0 \end{cases} \tag{19}$$

$$J_Q(Q) = \begin{cases} \begin{bmatrix} \frac{-\sin\theta + \theta\cos\theta}{\sin^2\theta} n & \frac{\theta}{\sin\theta} I \end{bmatrix}, & \theta \neq 0 \\ \\ \begin{bmatrix} 0_{3\times 1} & I \end{bmatrix}, & \theta = 0 \end{cases} \tag{20}$$

where n, θ are defined as in Appendix A for a unit quaternion Q.

Differentiating (9) utilizing (17) and (18) yields the following relations:

$$\dot{Q} = -\frac{1}{2} Q * \overline{Q}_g * J_{\log Q}(Q_g * \overline{Q})\dot{e}_Q * Q \tag{21}$$

$$\dot{e}_Q = -2J_Q(Q_g * \overline{Q})(Q_g * \overline{Q} * \dot{Q} * \overline{Q}) \tag{22}$$

During execution the quaternion derivative is computed from (21). We use (22) to get $\dot{e}_Q$ for the DMP training.

It is easy to show that the proposed formulation alleviates the problems identified in the dominant formulation by deriving the tracking dynamics. In fact, given a demonstrated trajectory $Q_d, \omega_d, \dot{\omega}_d$ with initial and final position $Q_{0,d}$ and $Q_{g,d}$ respectively, we calculate $e_{Q,d}$, $\dot{e}_{Q,d}$, $\ddot{e}_{Q,d}$ to train the forcing term:

$$F(x) = \mathrm{diag}\left(2\log(Q_{g,d} * \overline{Q}_{0,d})\right)\left(\tau^2 \ddot{e}_{Q,d} - \alpha_z(\beta_z e_{Q,d} - \tau\dot{e}_{Q,d})\right) \tag{23}$$

Assuming a training error ϵ_o, $F(x)$ in (14) can be substituted by the sum of the right hand side of (23) plus ϵ_o to yield the following system:

$$\tau^2(\ddot{e}_Q - S_g\ddot{e}_{Q,d}) + \tau\alpha_z(\dot{e}_Q - S_g\dot{e}_{Q,d}) + \alpha_z\beta_z(e_Q - S_g e_{Q,d}) = \mathrm{diag}\left(2\log(Q_g * \overline{Q})\right)\epsilon_o. \tag{24}$$

Equation (24) is a linear dynamical system of the tracking error $e_Q - S_g e_{Q,d}$ which converges to a steady state value dependent on the training error and in a critically damped way given the right choice of the DMP parameters. Consequently, when controlling the robot the generated orientation trajectory will follow the encoded pattern without exhibiting any undesired oscillations.

3.2 DMP Properties of the proposed formulation

The proposed formulation preserves the original scaling properties and modulation abilities. In the following we examine temporal and spatial scaling, phase stopping and couplings terms with the proposed formulation.

By setting the temporal scaling parameter to a different value than the one in training, the trajectory is scaled in time, at a ratio equal to $s_\tau = \frac{\tau_{new}}{\tau_{train}}$. Thus the resulted velocity and acceleration become $\frac{\dot{e}_Q(t)}{s_\tau}$, $\frac{\ddot{e}_Q(t)}{s_\tau^2}$ expanding or contracting the orientation error trajectory and its phase in time.

To spatially scale the executed motion's trajectory, consider a new goal $Q_{g,new}$. This goal selection, leads to the respective scaling of the trajectory at each dimension. The new trajectory becomes $S_g e_Q(t), S_g \dot{e}_Q(t), S_g \ddot{e}_Q$, with goal scaling matrix S_g defined as in Section 2. When the transition to a new goal is performed during the execution, a smooth transition is the best option, using a first order goal filtering system:

$$\omega_g = \alpha_g 2\log(Q_{g,new} * \overline{Q}_g) \tag{25}$$

with α_g is a positive gain term and $Q_{g,0} = Q_{g,train}$. Equation (25) is integrated using the exponential map.

In the presence of an external perturbation during execution, the robot deviates from its reference trajectory. The objective of phase stopping is to stop the evolution of the DMP, in order to resume its execution when the disturbance disappears. This can be achieved by slowing down the evolution of the phase variable x, modifying its dynamics as follows:

$$\tau\dot{x} = -\frac{\alpha_x}{1 + \alpha_{px}\|2\log(Q_{robot} * \overline{Q}_{DMP})\|^2} x \tag{26}$$

with α_{px} a positive gain parameter and Q_{robot} the robot's orientation. An improved phase stopping method proposed for position DMPs (in the joint or Cartesian space) can be found in Karlsson et al. [12], modifies the temporal scaling parameter thus slowing the evolution of both the phase and the transformation system. For the proposed orientation DMP this can be accomplished by modifying the time scaling with the following expression :

$$\tau = \tau_{train}\left(1 + \alpha_{p\tau}\|2\log(Q_{robot} * \overline{Q}_{DMP})\|^2\right) \tag{27}$$

with $\alpha_{p\tau}$ being a positive gain. Notice that as the robot follows the DMP generated trajectory, the temporal scaling term is equal to the one used in training. However, when the orientation error becomes large, τ obtains large values, leading to a very slow evolution of the system.

In many DMP applications, the augmentation of the DMP framework with coupling terms is utilized. Those modifications can be inserted to (14) as follows:

$$\begin{gathered} \tau\dot{z} = -\alpha_z(\beta_z e_Q + z) + \mathrm{diag}(2\log(Q_g * \overline{Q}_0))f(x) + z_C \\ \tau\dot{e}_Q = z + e_{Q,C} \end{gathered} \tag{28}$$

The terms z_C and $e_{Q,C}$ can be chosen to encapsulate any modulation of the system. For example, a popular spatial coupling term is to set $z_C = 0_{3\times1}$ and $e_{Q,C} = \alpha_{pe} 2\log(Q_{robot} * \overline{Q}_{DMP})$, with α_{pe} a positive gain term. This selection is usually used in conjunction with traditional phase stopping (26) and forces the DMP to stay close to the robot's trajectory in presence of perturbations.

4 Simulations

4.1 Evaluation of proposed DMP

To evaluate the proposed system' s performance, we used the same setup and parameters with the first simulation in Section 2. We trained the proposed DMP using (4), (14). Simulation results are shown in Figures 3, 4 for the orientation error and the angular velocity respectively. For comparison purposes they also include results with the dominant DMP orientation formulation presented in Section 2. It is evident that the proposed method does not demonstrate the oscillatory behaviour exhibited with the dominant. The initial velocity spike at Figure 4 is due to the acceleration discontinuity, a known problem which can be remedied by any of the different methods proposed in the literature e.g. by multiplying the linear part of the system with a suitable gating function.

4.2 Evaluation of proposed DMP properties

To demonstrate the preservation of the DMP properties, the training setup was the same as the previous simulations. However, during the execution alterations were made to test the system robustness. The results are shown in Figures 5 - 8. Figure 5 presents results of the system simulation with double time scaling i.e $\tau = 2$. Results with $\tau = 1$ are also included to demonstrate the temporal scaling properties. In Figure 6, a new goal is set at $Q'_g = [0.3635\ 0.6599\ -0.5918\ 0.2867]^T$, which yields the following initial orientation error $e'_{Q,0} = [5.2725\ -0.4505\ -0.5420]^T$. Notice

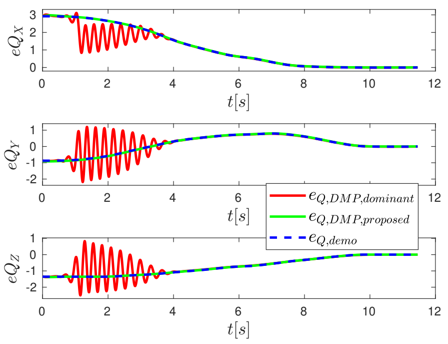

Figure 3: Evolution of quaternion error with the dominant and the proposed method. The demonstration error trajectory is also depicted.

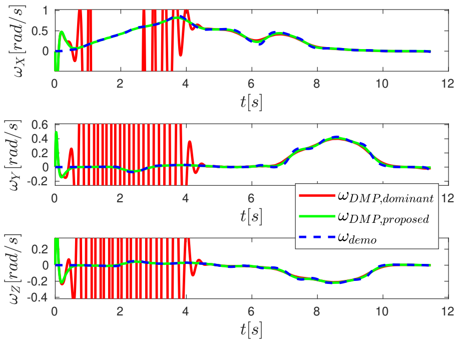

Figure 4: Evolution of angular velocity with the dominant and the proposed method. The demonstration angular velocity trajectory is also depicted.

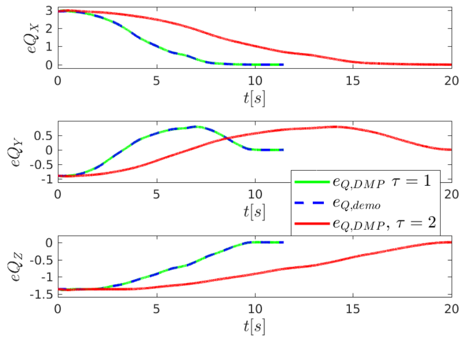

Figure 5: Evolution of quaternion error with $\tau = \tau_{train} = 1$ and $\tau = 2$. The demonstration error trajectory is also depicted.

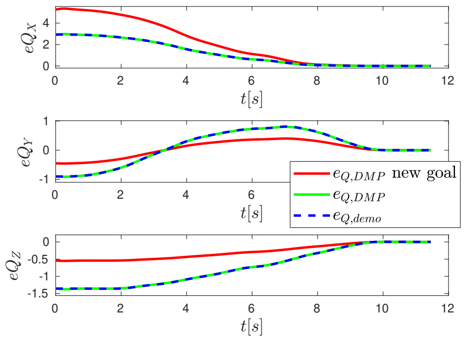

Figure 6: Evolution of quaternion error with the original and a new goal. The demonstration error trajectory is also depicted.

that $e'_{Q,0} = \text{diag}([1.8\ 0.5\ 0.4]^T)e_{Q,0}$. Results include the system evolution with the initial goal and demonstrate that the trajectory was scaled as expected in each dimension. To demonstrate the phase stopping results, the proposed DMP was simulated in conjunction with equation (27). To simulate the physical system, the following first order dynamics was used:

$$D(\omega_{robot} - \omega_{DMP}) + K(2\log(Q_{robot} * \overline{Q}_{DMP})) = u_{Dist} \tag{29}$$

with gains $K = 50, D = 2\sqrt{50}$ and $Q_{robot}, \omega_{robot}$ the simulated robot's orientation and angular velocity. The disturbance u_{Dist} was simulated by a trapezoidal function shown in Figure 7 which was applied in each direction. Figure 8 shows the evolution of the quaternion error for both the DMP and the simulated physical system. The τ evolution is included in the lower subplot of Figure 7. Results demonstrate the successful phase stopping in case of the disturbance. Notice the large values (max 48.58) reached by τ during the disturbance, effectively stop the DMP evolution.

5 Experimental Results

For the experimental evaluation of the proposed method, a 7 degree of freedom KUKA LWR4+ robotic manipulator was used. Both formulations were implemented in C++ and executed on a real-time Linux PC. For the communication with the robot the FRI library was used, accompanied with the ROS framework, with control frequency at $1kHz$. The desired orientation trajectory was provided kinesthetically by the user. To facilitate the demonstration, high stiffness values in the position coordinates were imposed to keep the wrist's position fixed. During execution a zero

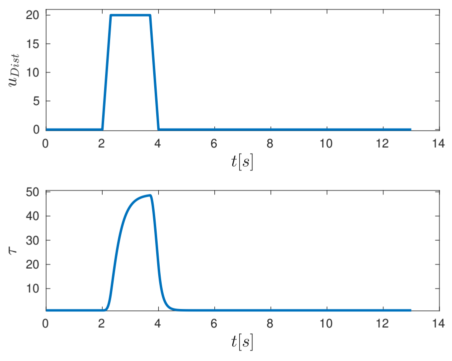

Figure 7: Upper subplot: Trapezoidal disturbance. Lower subplot: Temporal Scaling adaptation.

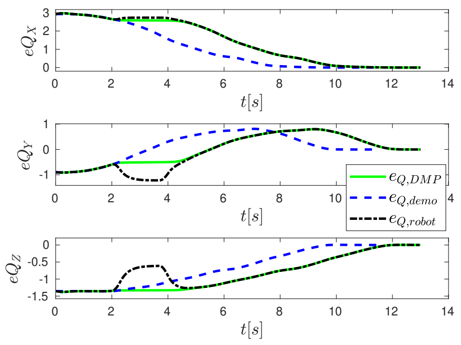

Figure 8: Evolution of quaternion error of DMP and simulated robot orientation in presence of disturbance. The demonstration error trajectory is also depicted.

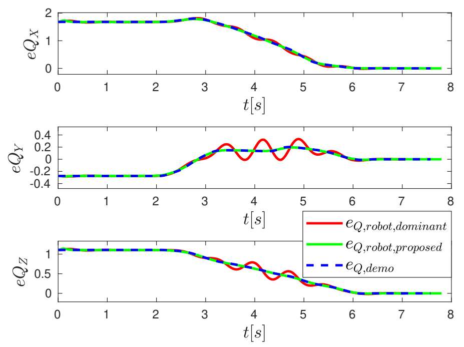

Figure 9: Experimental result: Evolution of quaternion error with the dominant and the proposed method. The demonstration error trajectory is also depicted.

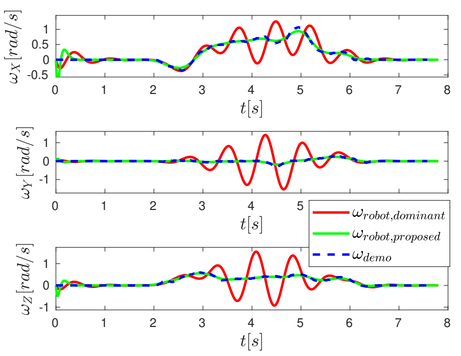

Figure 10: Experimental result:Evolution of angular velocity with the dominant and the proposed method. The demonstration angular velocity trajectory is also depicted.

desired linear velocity and the angular velocity generated by the DMP were mapped to the joint space with the Jacobian pseudo inverse. A number of orientation trajectory experiments were conducted to demonstrate the performance of the proposed approach as compared to the dominant. The results of one trajectory experiment are shown in this Section while the rest are given in the supplementary material, together with a related short video. The complete video can be found in `https://www.youtube.com/watch?v=AFWj58x8veQ`. The trajectory shown here starts from $Q_0 = [0.6659\ 0.5281\ 0.4669\ -0.2443]^T$ to goal $Q_g = [0.146\ 0.5587\ 0.5857\ 0.5688]^T$ yielding $e_{Q,0} = [1.682\ -0.2751\ 1.1065]^T$. Both formulations are trained with the parameters given in Section 4. Results are presented in Figures 9 and 10 depicting the quaternion error and the angular velocity with both formulations. It is evident that the proposed formulation accurately follows the demonstrated trajectory as opposed to the dominant which exhibits oscillatory behaviour.

6 Conclusions

In this paper we show that the existing formulation for orientation DMP is possible to lead to oscillatory behaviours during execution due to its non-linear nature. We then propose a correct formulation that is shown to remain linear while preserving DMP properties and modulation abilities. We demonstrate by simulations and experiments that the proposed formulation is predictable with respect to all DMP aspects.

Acknowledgments

The research leading to these results has received funding by the EU Horizon 2020 Research and Innovation Programme under grant agreement No 820767, project CoLLaboratE.

References

[1] A. Ude, B. Nemec, T. Petrić, and J. Morimoto. Orientation in cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2997–3004, May 2014. doi:10.1109/ICRA.2014.6907291.

[2] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1398–1403 vol.2, May 2002. doi:10.1109/ROBOT.2002.1014739.

[3] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation (NC)*, 25(2):328–373, Feb 2013. ISSN 0899-7667. doi:10.1162/NECO_a_00393.

[4] H. Hoffmann, P. Pastor, D. Park, and S. Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2587–2592, May 2009. doi:10.1109/ROBOT.2009.5152423.

[5] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics (T-RO)*, 30(4):816–830, Aug 2014. ISSN 1552-3098. doi:10.1109/TRO.2014.2304775.

[6] S. Chiaverini and B. Siciliano. The unit quaternion: A useful tool for inverse kinematics of robot manipulators. *Systems Analysis Modelling Simulation (Syst. Anal. Model. Simul.)*, 35 (1):45–60, Jan. 1999. ISSN 0232-9298. URL `http://dl.acm.org/citation.cfm?id=314861.314865`.

[7] J. S. Yuan. Closed-loop manipulator control using quaternion feedback. *IEEE Journal on Robotics and Automation (IEEE J Robot Autom)*, 4(4):434–440, Aug 1988. doi:10.1109/56.809.

[8] A. Tayebi. Unit quaternion-based output feedback for the attitude tracking problem. *IEEE Transactions on Automatic Control (IEEE T AUTOMAT CONTR)*, 53(6):1516–1520, July 2008. doi:10.1109/TAC.2008.927789.

[9] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell. An approach for imitation learning on riemannian manifolds. *IEEE Robotics and Automation Letters (RA-L)*, 2 (3):1240–1247, July 2017. doi:10.1109/LRA.2017.2657001.

[10] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 365–371, Sep. 2011. doi:10.1109/IROS.2011.6095059.

[11] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994. ISBN 0849379814. doi:10.1201/9781315136370.

[12] M. Karlsson, A. Robertsson, and R. Johansson. Convergence of dynamical movement primitives with temporal coupling. In *2018 European Control Conference (ECC)*, pages 32–39, June 2018. doi:10.23919/ECC.2018.8550135.

Appendix A Basic Quaternion Operations

Quaternions are comprised of a real scalar part $w \in \mathbb{R}$ and a vector part $v \in \mathbb{R}^3$ and are usually written as:

$$Q = \begin{bmatrix} w & v^T \end{bmatrix}^T \tag{30}$$

Unit quaternions are widely used in robotics, to represent orientations in a compact, singularity free way. A unit quaternion can also be expressed by utilizing an angle axis parameterization as follows:

$$Q = \begin{bmatrix} \cos\theta & \sin\theta n^T \end{bmatrix}^T \tag{31}$$

where n represents the rotation's unit axis and θ the half of the rotation's angle. To maintain the one to one representation, θ is confined to the interval $[0, \pi)$.

The product of two quaternions is defined as follows: $Q_1 * Q_2 = \begin{bmatrix} w_1 w_2 - v_1^T v_2 \\ w_1 v_2 + w_2 v_1 + v_1 \times v_2 \end{bmatrix}$

The inverse of a unit quaternion is equal to its conjugate: $Q^{-1} = \overline{Q} = \begin{bmatrix} w & -v^T \end{bmatrix}^T$

The logarithm of a unit quaternion is defined as:

$$\log(Q) = \begin{cases} n\theta & \theta \neq 0 \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T & \theta = 0 \end{cases} \tag{32}$$

For the integration of a set of quaternion differential equations, one uses:

$$Q(t + \Delta t) = \exp\left(\frac{1}{2}\omega(t)\Delta t\right) * Q(t) \tag{33}$$

where the exponential mapping used above is defined by:

$$\exp(v) = \begin{cases} \begin{bmatrix} \cos(\|v\|) & \sin(\|v\|)\frac{v^T}{\|v\|} \end{bmatrix}^T & \|v\| \neq 0 \\ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T & \|v\| = 0 \end{cases} \tag{34}$$

Appendix B Differentiation of Quaternion Logarithm

We will initially compute the derivatives of n, θ, for $\theta \neq 0$. Notice that as n is a unit axis, then (32) implies $\theta = \|\log Q\|$. Its derivative yields $\dot{\theta} = \frac{(\log Q)^T}{\|\log Q\|}\frac{d\log(Q)}{dt} = n^T \frac{d\log(Q)}{dt}$.

The axis n can be written as $n = \frac{\log Q}{\theta} = \frac{\log Q}{\|\log Q\|}$. Its derivative yields $\dot{n} = \frac{1}{\theta}(I - nn^T)\frac{d\log(Q)}{dt}$.

Then we differentiate equation (31) using the previous results to find:

$$\dot{Q} = \begin{bmatrix} -\sin\theta n^T \\ \frac{\sin\theta}{\theta}(I - nn^T) + \cos\theta nn^T \end{bmatrix} \frac{d\log Q}{dt}$$

thus, obtaining matrix $J_{\log Q}(Q)$ of equation (19). The case of $\theta = 0$ can be easily computed by evaluating the limit with $\theta \to 0$.

To compute the inverse equation we use (31) and (30). Then $\theta = \cos^{-1} w$ and $n = \frac{1}{\sin\theta}v$. Differentiating yields $\dot{\theta} = -\frac{\dot{w}}{\sqrt{1-w^2}} = -\frac{\dot{w}}{\sin\theta}$ and $\dot{n} = \frac{1}{\sin\theta}\dot{v} + \frac{\dot{w}\cos\theta}{\sin^2\theta}n$. Then we differentiate equation (32) for $\theta \neq 0$ using the previous results to find:

$$\frac{d\log(Q)}{dt} = \dot{\theta}n + \theta\dot{n} = \begin{bmatrix} \frac{-\sin\theta + \theta\cos\theta}{\sin^2\theta}n & \frac{\theta}{\sin\theta}I \end{bmatrix}\dot{Q}$$

thus obtaining matrix $J_Q(Q)$ of equation (20). For $\theta = 0$ we also compute the limit as before.

Omitting the arguments for notation simplicity , it is easy to show that $J_Q J_{\log Q} = I_3$. It is further easy to show that $J_{\log Q} J_Q$ is a projection matrix since it is idempotent, and one can show that $J_{\log Q} J_Q \dot{Q} = \dot{Q}$. However this projection is not orthogonal since the matrix is not symmetric.