Overcoming Some Drawbacks of Dynamic Movement Primitives

Journal Title
XX(X):1–12

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Michele Ginesi[1], Nicola Sansonetto[1], and Paolo Fiorini[1]

Abstract

Dynamic Movement Primitives (DMPs) is a framework for learning a point-to-point trajectory from a demonstration. Despite being widely used, DMPs still present some shortcomings that may limit their usage in real robotic applications. Firstly, at the state of the art, mainly Gaussian basis functions have been used to perform function approximation. Secondly, adaptation of the trajectory generated by the DMP heavily depends on the choice of hyperparameters and on the new desired goal position. Lastly, DMPs are a framework for 'one-shot learning', meaning that they are constrained to learn from a unique demonstration. In this work, we present and motivate a new set of basis functions to be used in the learning process, showing their ability to accurately approximate functions while having both analytical and numerical advantages w.r.t. Gaussian basis functions. Then, we show how to use the invariance of DMPs w.r.t. affine transformations in order to make the generalization of the trajectory robust against both the choice of hyperparameters and new goal position, performing synthetic tests to show this increased robustness. Finally, we propose an algorithm to extract a common behavior from multiple observations, validating it both on a synthetic dataset and on a dataset obtained by performing a task on a real robot.

1 Introduction

The recent improvements in robot dexterity have given rise to an increasing attention to Learning from Demonstration (LfD) approaches to make robots faithfully mimic human motions.

Dynamic Movement Primitive (DMP) Ijspeert et al. (2002, 2003); Schaal (2006); Ijspeert et al. (2013) is one of the most used frameworks for trajectory learning from a single demonstration. They are based on a system of second order Ordinary Differential Equations (ODEs), in which a *forcing term* can be "learned" to encode the desired trajectory. This approach has already been proven effective in teaching a robot how to perform some human task as, for instance, (table) tennis swing Ijspeert et al. (2002); Matsubara et al. (2010), to play drums Ude et al. (2010), and to write Kulvicius et al. (2012); Wang et al. (2016). The framework of DMPs have been shown to be flexible and robust enough to allow learning sensory experience Pastor et al. (2011, 2012); Kappler et al. (2015), handling obstacle avoidance Park et al. (2008); Hoffmann et al. (2009); Ginesi et al. (2019a), describing bi-manual tasks Gams et al. (2014), learning orientations Ude et al. (2014); Saveriano et al. (2019), and working in scenarios with human-robot interaction Amor et al. (2014).

In spite of their wide use, DMPs approach has still some shortcoming that needs to be fixed to obtain a more robust framework.

In this work we discuss three aspects of DMPs, and propose modifications that guarantee a more robust implementation of the framework. In more details, we discuss different classes of basis functions to be used instead of the Gaussian radial basis functions. Then we show how to exploit invariance of DMPs with respect to particular transformations in order to make the generalization of the learned trajectory more robust. Finally we present an algorithm to learn a unique DMP from multiple observations without the need to rely on probabilistic approaches or additional parameters.

The work is organized as follows. In Section 2 we review the two classical formulations of DMPs, emphasizing their shortcomings. In Section 3 we review multiple classes of basis functions and introduce a new one, which has both the desirable properties of being smooth and compactly supported. We then test and compare various numerical aspects of all the sets of basis functions presented, showing that our proposed one gives rise to a more numerically stable optimization problem. In Section 4 we show how to generalize the DMPs to any new starting and goal positions by exploiting the invariance property of DMPs under affine transformations. In Section 5 we present and test a novel algorithm to learn a unique DMP from a set of observations. Lastly, in Section 6 we present the conclusions.

Our implementation of DMPs, written in Python 3.6, is available at the link `https://github.com/mginesi/dmp_pp`. Together with the implementation of DMPs, and of our extensions, the repository contains the scripts to run all the tests presented in Sections 3.2, 4.2, and 5.2, together with the tests that are mentioned but not shown.

[1]Department of Computer Science, University of Verona

Corresponding author:
Michele Ginesi, Department of Computer Science, university of Verona.
Email: michele.ginesi@univr.it

2 Dynamic Movement Primitives: an Overview

DMPs are used to model both periodic and discrete movements Ijspeert et al. (2002, 2003); Schaal (2006). However, in this work, we will focus on the latter.
They consist of a system of second order ODEs (one for each dimension of the ambient space) of mass-spring-damper type with a forcing term. The aim of DMPs is to model the forcing term in such a way to being able to generalize the trajectory to new start and goal positions while maintaining the shape of the learned trajectory.

The one-dimensional formulation of DMPs is:

$$\begin{cases} \tau \dot{v} = K(g - x) - Dv + (g - x_0) f(s) & \text{(1a)} \\ \tau \dot{x} = v & \text{(1b)} \end{cases}$$

where $x, v \in \mathbb{R}$ are, respectively, position and velocity of a prescribed point of the system. $x_0 \in \mathbb{R}$ is the initial position, while $g \in \mathbb{R}$ is the *goal*. Constants $K, D \in \mathbb{R}^+$ are, respectively, the spring and damping terms, chosen in such a way that the associated homogeneous system is critically damped: $D = 2\sqrt{K}$. Parameter $\tau \in \mathbb{R}^+$ is a temporal scaling factor, and f is a real-valued, non-linear *forcing* (also called *perturbation*) *term*. s is a re-parametrization of time $t \in [0, T]$, governed by the so-called *canonical system*:

$$\tau \dot{s} = -\alpha s. \tag{2}$$

$\alpha \in \mathbb{R}^+$ determines the exponential decay of canonical system (2).
The forcing term f in (1a) is written in terms of basis functions as

$$f(s) = \frac{\sum_{i=0}^{N} \omega_i \psi_i(s)}{\sum_{i=0}^{N} \psi_i(s)} s, \tag{3}$$

where

$$\psi_i(s) = \exp\left(-h_i (s - c_i)^2\right) \tag{4}$$

are Gaussian basis functions with centers c_i and widths h_i defined respectively as:

$$c_i = \exp\left(-\alpha \, i \, \frac{T}{N}\right), \quad i = 0, 1, \ldots, N, \tag{5}$$

and

$$\begin{aligned} h_i &= \frac{1}{(c_{i+1} - c_i)^2}, \quad i = 0, 1, \ldots, N-1, \\ h_N &= h_{N-1}. \end{aligned} \tag{6}$$

The learning process focuses on the computation of the weights ω_i that best approximate the desired forcing term, obtained by solving (1a) for f. In details, the vector $\boldsymbol{\omega} = [\omega_0, \omega_1, \ldots, \omega_N]^\mathsf{T}$ can be computed by solving the linear system

$$\mathbf{A}\boldsymbol{\omega} = \mathbf{b}, \tag{7}$$

where $\mathbf{A} = [a_{hk}]$ is a $(N+1) \times (N+1)$ matrix and $\mathbf{b} = [b_h]$ is a vector in $\mathbb{R}^{N+1}$ with components

$$\begin{aligned} a_{hk} &= \int_{s(0)}^{s(T)} \frac{\psi_h(s)\psi_k(s)}{\left(\sum_{i=0}^{N} \psi_i(s)\right)^2} s^2 \, ds, \\ b_h &= \int_{s(0)}^{s(T)} \frac{\psi_h(s)\psi_k(s)}{\sum_{i=0}^{N} \psi_i(s)} f(s) \, s \, ds. \end{aligned} \tag{8}$$

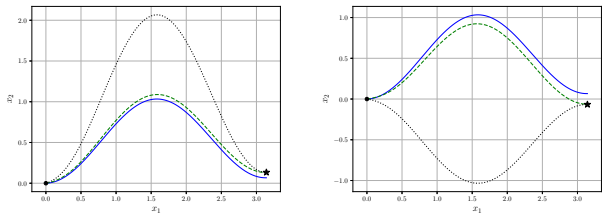


Figure 1. Comparison between DMPs' formulation (9) and (10). In both plot the blue solid line shows the learned trajectory. The dot and star mark, respectively, the new initial position $\mathbf{x}_0$ (which, for simplicity, in this example is kept the same as the learned one) and new goal position $\mathbf{g}$. The dotted line shows the generalization to the new goal using DMP formulation (9), while the dashed line shows the generalization using formulation (10). The grid is plotted to emphasize that in Figure 1b the vertical component of the new goal position is of opposite sign than the learned one.

When dealing with d-dimensional trajectories, we make d decoupled copies of system (1), obtaining the following vector formulation:

$$\begin{cases} \tau \dot{\mathbf{v}} = \mathbf{K}(\mathbf{g} - \mathbf{x}) - \mathbf{D}\mathbf{v} + (\mathbf{g} - \mathbf{x}_0) \odot \mathbf{f}(s) & \text{(9a)} \\ \tau \dot{\mathbf{x}} = \mathbf{v} & \text{(9b)} \end{cases}$$

where $\mathbf{x}, \mathbf{v}, \mathbf{g}, \mathbf{x}_0, \mathbf{f}(s) \in \mathbb{R}^d$, $\mathbf{K}, \mathbf{D} \in \mathbb{R}^{d \times d}$ are diagonal matrices, so to maintain each direction decoupled from the others. The operator $\odot$ denotes the component-wise multiplication: given $\mathbf{v} = [v_i], \mathbf{w} = [w_i] \in \mathbb{R}^d$, we define $\mathbf{v} \odot \mathbf{w} = [v_i w_i]_i$.
Thanks to the decoupling of the system, the forcing term can be learned direction by direction.

Three main drawbacks characterize DMP formulation (9) (see Hoffmann et al. (2009)). Firstly, if in any direction the goal position coincides with the starting position, $g = x_0$, the perturbation term f clearly does not contribute to the evolution of the dynamical system. Secondly, if $g - x_0$ is "small" the scaling of the perturbation term f by the quantity $g - x_0$ may produce unexpected (and undesired) behaviors. Finally, if the scaling factor $g - x_0$ changes sign from the learned trajectory to the new one, the trajectory will result mirrored.
In Figure 1 we present an example similar to that in Hoffmann et al. (2009), Figure 2 to show the aforementioned drawbacks. In both tests, the vertical component of the difference $\mathbf{g} - \mathbf{x}_0$ is quite small: $(\mathbf{g} - \mathbf{x}_0)|_2 = \frac{1}{15} \approx 0.0667$. Figure 1a shows that even a slight change in goal position results in a complete different trajectory when using formulation (1). Figure 1b, instead, shows the 'mirroring' problem. In this case, the vertical component of $\mathbf{g} - \mathbf{x}_0$ changes sign from the learned to the new execution. Thus, the trajectory results mirrored w.r.t. the horizontal axis $x_2 = 0$.

To overcome these disadvantages, the following formulation was proposed in Park et al. (2008); Pastor et al. (2009); Hoffmann et al. (2009):

$$\begin{cases} \tau \dot{\mathbf{v}} = \mathbf{K}(\mathbf{g} - \mathbf{x}) - \mathbf{D}\mathbf{v} - \mathbf{K}(\mathbf{g} - \mathbf{x}_0)s + \mathbf{K}\mathbf{f}(s) & \text{(10a)} \\ \tau \dot{\mathbf{x}} = \mathbf{v} & \text{(10b)} \end{cases}$$

where the evolution of s is still described by the canonical system (2), and the forcing term is still written as in (3).

By removing the scaling term $g - x_0$ from the forcing term f, this new formulation solves the aforementioned problems of (1), as it can be seen in Figure 1. However, (10) still presents an important drawback: the generalization to different spatial scales is not always feasible, since the forcing term does not have any dependence on the relative position between starting and ending points (we will discuss the details of this aspect in Section 4).

3 New Set of Basis Functions

A change of the set of basis functions is motivated by the fact that a desirable property of basis functions is to be compactly supported Wang et al. (2016), since compactly supported basis functions provide an easier update of the learned trajectory. Indeed, if only a portion of the trajectory has to be modified, only a subset of the weights need to be re-computed. Moreover, while with Gaussian basis functions the forcing term never vanishes, compactly supported basis functions allow the forcing term to be zero outside the support, thus providing a faster convergence to the goal position.

In Wang et al. (2016) the authors proposed to use truncated Gaussian basis functions:

$$\tilde{\psi}_i(s) = \begin{cases} \exp\left(-\frac{h_i}{2}(s-c_i)^2\right), & \text{if } s - c_i \leq \theta_i \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where c_i and h_i are defined as in (5) and (6) respectively, and $\theta_i = K/\sqrt{h_i}$. Unfortunately, this approach has two main drawbacks. Firstly, a discontinuity is introduced at the truncation points θ_i, that may produces unexpected behaviors. Secondly, in order to work properly, this approach requires the introduction of *biases terms* in (3), which now reads

$$f(s) = \frac{\sum_{i=0}^{N}(\omega_i s + \beta_i)\,\tilde{\psi}_i(s)}{\sum_{i=0}^{N}\tilde{\psi}_i(s)}, \tag{12}$$

thus doubling the number of parameters that need to be learned (a weight ω_i and a bias β_i for each basis function), and increasing the overall computational cost.
Finally, we remark that these basis functions are not compactly supported, even though they are zero after a finite time. Indeed their support is an interval $(-\infty, L]$ with $L \in \mathbb{R}$.

3.1 Mollifier-like and Wendland's Basis Functions

In this work, we propose a new set of basis fuctions which are both smooth and compactly supported.
Consider the *mollifier* $\varphi : \mathbb{R} \to \mathbb{R}$ defined as

$$\varphi(x) = \begin{cases} \frac{1}{I_n}\exp\left(-\frac{1}{1-|x|^2}\right) & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

where I_n is set so that $\int_{\mathbb{R}} \varphi(x)\,dx = 1$. Function φ is smooth and compactly supported (its support is the interval $[-1, 1]$). It is then natural to define the set of *mollifier-like basis functions* $\{\varphi_i(s)\}_{i=0,1,\dots,N}$ given by

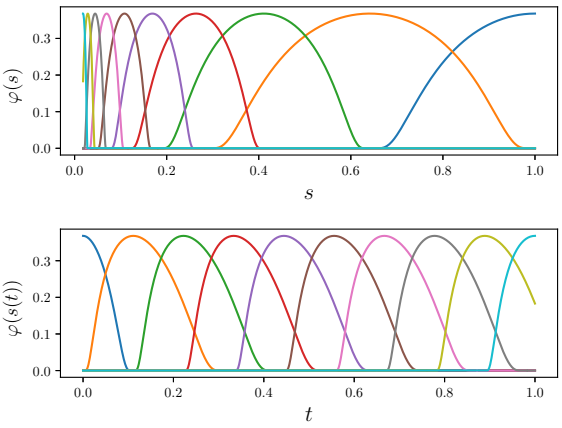


Figure 2. Plot of mollifier-like basis functions as defined in (14). In this example $N = 9$, $\alpha = 4$ and $T = 1$. The first plot shows the basis functions as function of s, while the second plot shows them as functions of time t.

$$\varphi_i(s) = \begin{cases} \exp\left(-\frac{1}{1-|a_i(s-c_i)|^2}\right) & \text{if } |a_i(s-c_i)| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where, given the centers c_i as in (5), we define the widths a_i as

$$\begin{aligned} a_i &= \frac{1}{c_i - c_{i-1}}, \quad i = 1, 2, \dots, N, \\ a_0 &= a_1. \end{aligned} \tag{15}$$

The usual normalization term I_n in (13) is omitted since it does not enter our approach.
In Figure 2 we plot an example of mollifier-like functions both as function of s and of t. We remark that the basis functions are equispaced in t. Moreover, due to the fact that s is a strictly decreasing function of t, their order changes from s to t (the first basis function in s is the last in t and vice versa), as it was for the Gaussian basis functions used so far in the literature.

Other well known regular (see Table 1 for details on the regularity) and compactly supported basis functions are *Wendland's functions* Wendland (1995); Schaback (2007). In Section 3.2 we test the following Wendland's basis function (where the operator $(\cdot)_+$ denotes the *positive part function*: $(x)_+ = \max\{0, x\}$):

$$\phi_i^{(2)}(r) = (1-r)_+^2, \tag{16a}$$
$$\phi_i^{(3)}(r) = (1-r)_+^3, \tag{16b}$$
$$\phi_i^{(4)}(r) = (1-r)_+^4(4r+1), \tag{16c}$$
$$\phi_i^{(5)}(r) = (1-r)_+^5(5r+1), \tag{16d}$$
$$\phi_i^{(6)}(r) = (1-r)_+^6(35r^2+18r+3), \tag{16e}$$
$$\phi_i^{(7)}(r) = (1-r)_+^7(16r^2+7r+1), \tag{16f}$$
$$\phi_i^{(8)}(r) = (1-r)_+^8(32r^3+25r^2+8r+1). \tag{16g}$$

where c_i are the centers and a_i the widths as in (5) and (15) respectively, and we defined r in (16) as $r = |a_i(s - c_i)|$.

In Table 1 we summarize the properties of the basis functions we presented in this section. As it can be seen,

mollifier-like basis functions are the only ones being both smooth and compactly supported.

3.2 Results

We now investigate the goodness of the approximations obtained using the various examples of basis functions we introduced. In particular, we test both the classical (4) and the truncated (11) Gaussian basis functions, the mollifier-like basis functions (14), and the Wendland's functions (16). We test two numerical aspects: in Section 3.2.1 the approximation error on a particular target function, and in Section 3.2.2 the condition number of matrix $\mathbf{A}$ in (7).

3.2.1 Numerical Accuracy We test the accuracy of the approximation of the following *hat function*:

$$\eta(t) = \begin{cases} -4\left|t-\frac{1}{2}\right|+1 & t \in \left(\frac{1}{4},\frac{3}{4}\right) \\ 0 & \text{otherwise} \end{cases}. \tag{17}$$

Figure 3 shows the approximation error, measured w.r.t. the L^2-norm, both w.r.t. the number of basis functions (Figure 3a), and w.r.t. the number of parameters that need to be learned (Figure 3b). We recall that for truncated Gaussian basis functions the number of parameters is double the number of basis functions, since every basis function require a weight and a bias. On the other hand, for all other classes of basis functions, the number of parameters coincide with the number of basis functions, since for each basis function only one parameter (the weight) has to be computed.
The plot shows that truncated Gaussian functions (11) work properly only using the biased formulation (denoted by $\tilde{\psi}_B$ in the legend) for the forcing term (12) , which means that given N basis functions we are solving a $2N$-length linear system when computing the weights and the biases. On the other hand, when using the unbiased formulation (3) (denoted by $\tilde{\psi}_U$ in the legend) the approximant does not converges to the desired forcing term.
We see that Gaussian basis functions are the best approximantors (however, we recall, they are not compactly supported). When comparing the L^2 error w.r.t. the number of basis functions, we observe that (biased) truncated Gaussian functions are a better approximator than mollifier-like functions. However, when comparing the error w.r.t. the number of parameters that need to be computed, mollifier-like basis functions are better than truncated Gaussians.
We tested classical Gaussians, Wendland, and mollifier-like basis functions also using the biased formulation (12), without noticing any difference in the goodness of the approximation.

3.2.2 Condition Number We now discuss the numerical accuracy of the minimization problem (7) used to compute the weights ω_i in (3). To do so, we investigate the condition number of matrix $\mathbf{A}$, $\text{cond}(\mathbf{A}) \stackrel{\text{def}}{=} \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$, in (7). The importance of this test is given from the fact that when one solves numerically a linear system, the approximation error is directly proportional to the condition number (for further details, see Appendix A).
In this test, we do not consider truncated Gaussian basis functions since, in this case, the components $\mathbf{A}, \mathbf{b}$ of the linear problem (7) have a different formulations, having to learn both the weights and the biases.

Figure 4 shows that the condition number $\text{cond}(\mathbf{A})$ of matrix $\mathbf{A}$ increases faster for Gaussian basis functions, while the compactly supported basis functions show a slower increase. Moreover, even with only ten basis functions, the condition number obtained with Gaussian basis functions is significantly bigger than any compactly supported basis function. This means that solving the minimization problem (7) using Gaussian basis functions results in a more severe numerical cancellation error than mollifier-like or Wendland's functions.
The lower condition number can be explained by the fact that mollifier-like and Wendland's basis functions are compactly supported, and then the resulting matrix $\mathbf{A}$ will have many off-diagonals components equal to zero, as it can be seen in the sparsity pattern shown in Figure 5. On the other hand, when one uses Gaussian basis functions, all the components of $\mathbf{A}$ are non-zero, since $\psi_i(s) > 0, \forall s \in \mathbb{R}$. Thus, $\mathbf{A}$ is a full matrix when using Gaussian basis functions, while it is a sparse $n-$diagonal matrix when using compactly supported basis functions; and, in general, full matrices have a bigger condition number than sparse $n-$diagonal ones.
Moreover, solving a sparse linear system is faster than solving a full one, since fewer operations have to be performed.

We remark that convergence shown in Figure 3 is done on a particular choice of the target function. The convergence error may differ depending on the forcing term that needs to be approximated, and various basis functions should be tested to choose the best one for the particular case. However, we stress that the condition number shown in Figure 4 does not depend on the target function, but only on the choice of basis functions.

The tests above let us argue that the use of mollifier-like basis functions is the most reliable in the applications. Indeed, the use of these basis functions results in a null forcing term after a finite time $f(s(t)) = 0, \forall t > T^\star$, that ensures faster convergence to the goal position. On the other hand, the use of Gaussian basis functions results in a non null perturbation term, $f(s(t)) \neq 0, \forall t \in \mathbb{R}$, that may slow down the convergence to the goal. Moreover, if a trajectory needs to be updated, only a subset of the weights has to be re-computed if one uses mollifier-like basis functions. In particular, if a trajectory has to be updated in a time interval $[t_0, t_1]$, only the weights corresponding to the basis functions φ_i satisfying $\text{supp}(\varphi_i) \cap [t_0, t_1] \neq \emptyset$ have to be re-computed.
Finally, the goodness of the approximation shown in Figure 3 is comparable to the one of Gaussian basis functions, while resulting in a linear problem with lower condition number (see Figure 4).

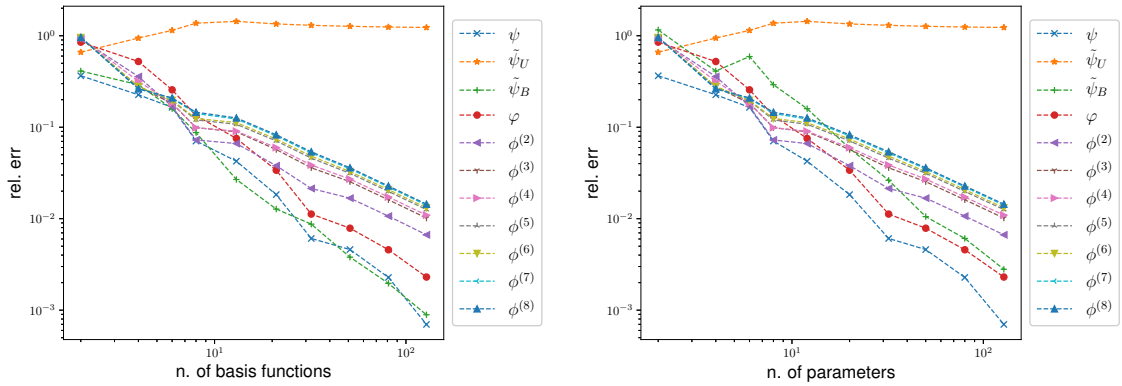
4 Invariance Under Invertible Linear Transformations

Invariance under translations of DMPs formulations (1) and (10) is straightforward. Indeed, by changing the initial position from $\mathbf{x}_0$ to $\mathbf{x}'_0$, and by considering, instead of $\mathbf{g}$, the new goal position $\mathbf{g}' = \mathbf{g} + (\mathbf{x}'_0 - \mathbf{x}_0)$, the whole trajectory is translated by a quantity $\mathbf{x}'_0 - \mathbf{x}_0$.

DMPs are able to adapt to small changes of the relative position between goal and start, $\mathbf{g} - \mathbf{x}_0$; however, the robustness of DMPs w.r.t big changes of the Euclidean norm

Table 1. Summary of the properties (regularity and support compactness) of the different sets of basis functions presented in Section 3. Bold font is used to mark the more desirable properties (smoothness and compact support).

Function		Regularity	Compactly Supported
Gaussian	$\psi_i(s)(\cdot)$	$\boldsymbol{\mathcal{C}^\omega(\mathbb{R})}$	No
Truncated Gaussian	$\tilde{\psi}_i(s)(\cdot)$	Discontinuous	No
Mollifier-like	$\phi_i(s)(\cdot)$	$\boldsymbol{\mathcal{C}^\infty(\mathbb{R})}$	**Yes**
Wendland	$\phi^{(2)}(\cdot)$	$\mathcal{C}^0(\mathbb{R})$	**Yes**
	$\phi^{(3)}(\cdot)$	$\mathcal{C}^0(\mathbb{R})$	**Yes**
	$\phi^{(4)}(\cdot)$	$\mathcal{C}^2(\mathbb{R})$	**Yes**
	$\phi^{(5)}(\cdot)$	$\mathcal{C}^2(\mathbb{R})$	**Yes**
	$\phi^{(6)}(\cdot)$	$\mathcal{C}^4(\mathbb{R})$	**Yes**
	$\phi^{(7)}(\cdot)$	$\mathcal{C}^4(\mathbb{R})$	**Yes**
	$\phi^{(8)}(\cdot)$	$\mathcal{C}^6(\mathbb{R})$	**Yes**



(a) Error as function of the number of basis functions

(b) Error as function of the number of parameters

Figure 3. Plot of the L^2 error done by approximating the hat function (17). First plot shows the error w.r.t. the number of basis functions, while second plot shows the error w.r.t. the number of parameters that has to be learned. Truncated Gaussian basis functions are tested both using the (classical) unbiased formulation (3) and the biased one (12). These two different approaches are denoted respectively by $\tilde{\psi}_U$ and $\tilde{\psi}_B$ in the legend.

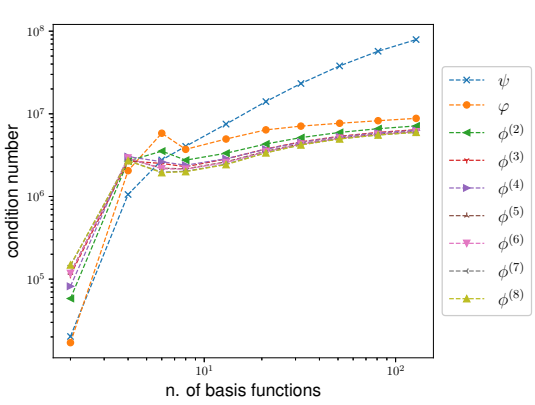


Figure 4. Condition number of the matrix $\mathbf{A}$ with elements a_{hk} defined in (26) w.r.t. the number of basis functions.

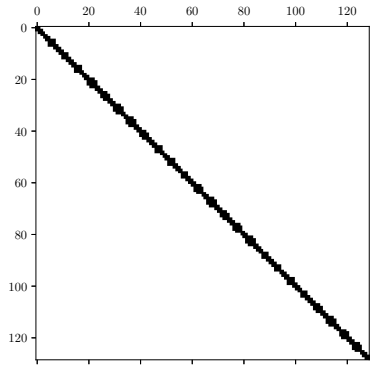


Figure 5. Sparsity pattern of matrix $\mathbf{A}$ with elements a_{hk} defined in (8) in the case of $N = 2^7 = 128$ using mollifier-like basis functions.

of $\mathbf{g} - \mathbf{x}_0$ heavily depends on the choice of the hyperparameters $\mathbf{K}, \mathbf{D}$ in (10) and α in (2) (see, for instance, the tests performed in Figures 6, 7). Generalization to arbitrary changes of quantity $\mathbf{g} - \mathbf{x}_0$ can be achieved by exploiting the invariance property of equations (10) w.r.t. (invertible) affine transformations, see Hoffmann et al. (2009). At the best of the authors' knowledge, this well known property has never been exploited in order to make DMPs globally robust w.r.t. arbitrary changes of $\mathbf{g} - \mathbf{x}_0$.

The invariance of (10) can be easily proven (see

Hoffmann et al. (2009)). Consider the invertible transformation matrix $\mathbf{S} \in \mathbb{R}^{d\times d}$. By substituting

$$\begin{aligned} &\mathbf{x}' = \mathbf{S}\mathbf{x}, \quad \mathbf{v}' = \mathbf{S}\mathbf{v}, \qquad \mathbf{x}'_0 = \mathbf{S}\mathbf{x}_0, \\ &\mathbf{g}' = \mathbf{S}\mathbf{g}, \quad \mathbf{K}' = \mathbf{S}\mathbf{K}\mathbf{S}^{-1}, \quad \mathbf{D}' = \mathbf{S}\mathbf{D}\mathbf{S}^{-1}, \end{aligned} \tag{18}$$

in (10) we obtain the transformation law of the forcing term

$$\boxed{\mathbf{f}' = \mathbf{S}\mathbf{f}.} \tag{19}$$

Thus, if we want to generate a new trajectory $\mathbf{S}\mathbf{x}$, it is sufficient to apply the transformations in (18) and (19) to (10). We remark that if the elastic and damping terms are the same for each direction, then $\mathbf{K}$ and $\mathbf{D}$ are multiple of the identity matrix and thus $\mathbf{K} = \mathbf{K}'$ and $\mathbf{D} = \mathbf{D}'$.

4.1 Invariance Under Roto-Dilatation

A case of particular interest is when $\mathbf{S}$ in (18), (19) represents a *roto-dilatation*.
Consider a trajectory which is learned together with the relative position between the goal and the starting point, i.e. the vector $\mathbf{g} - \mathbf{x}_0$; and a new trajectory, with starting and ending points $\mathbf{x}'_0$ and $\mathbf{g}'$ respectively has to be reproduced.
We first compute the rotation matrix $\mathbf{R}^{\widehat{\mathbf{g}'-\mathbf{x}'_0}}_{\widehat{\mathbf{g}-\mathbf{x}_0}}$, which maps the unit vector $\widehat{\mathbf{g} - \mathbf{x}_0}$ to $\widehat{\mathbf{g}' - \mathbf{x}'_0}$, where operator $\widehat{\cdot}$ denotes the normalization: $\widehat{\mathbf{v}} \stackrel{\text{def}}{=} \mathbf{v}/\|\mathbf{v}\|$. Rotation matrix $\mathbf{R}^{\widehat{\mathbf{g}'-\mathbf{x}'_0}}_{\widehat{\mathbf{g}-\mathbf{x}_0}}$ can be computed using the algorithm presented in Zhelezov (2017). After performing the rotation, we perform a dilatation of $\|\mathbf{g}' - \mathbf{x}'_0\|/\|\mathbf{g} - \mathbf{x}_0\|$ obtaining the transformation matrix

$$\boxed{\mathbf{S} = \frac{\|\mathbf{g}' - \mathbf{x}'_0\|}{\|\mathbf{g} - \mathbf{x}_0\|} \mathbf{R}^{\widehat{\mathbf{g}'-\mathbf{x}'_0}}_{\widehat{\mathbf{g}-\mathbf{x}_0}}.} \tag{20}$$

Therefore, when a learned DMP has to be used to generate a new trajectory, characterized by the parameters $\mathbf{x}'_0$ and $\mathbf{g}'$, we compute the matrix $\mathbf{S}$ as in (20), and then the quantities $\mathbf{K}'$, $\mathbf{D}'$, and $\mathbf{f}'$ as in (18) and (19).
This approach can be used even if the goal position changes with time during the execution of the trajectory simply by updating the matrix $\mathbf{S}$ at each time. This, in particular, means that $\mathbf{S}$ depends on time. In Section 4.2 we will test this approach both for static and moving goal positions.

We remark that this method cannot be performed when starting and ending points coincide: $\mathbf{x}_0 = \mathbf{g}$, since the matrix $\mathbf{S}$ in (20) would result in a null matrix, $\mathbf{S} = \mathbf{0}$. However, in such cases, one should use *rhythmic* (or *periodic*) DMPs instead of discrete ones, which are beyond the scope of this paper.

Remark 1. The choice to use a roto-dilatation as transformation matrix $\mathbf{S}$ is not the unique possibility. Indeed, any invertible matrix can be used. For example, if we write $\mathbf{S}$ as the diagonal matrix $\mathbf{S} = \text{diag}(s_1, s_2, \ldots, s_d)$ in which each component of the diagonal is defined as (assuming that the learned goal and starting positions differ in each component)

$$s_i = \frac{\mathbf{g}'_i - \mathbf{x}'_{0\,i}}{\mathbf{g}_i - \mathbf{x}_{0\,i}}, \quad i = 1, 2, \ldots, d,$$

we would obtain a formulation similar to (9), in which each direction is scaled by $g - x_0$.

Remark 2. Choosing roto-dilatation has the advantage that the evaluation of the inverse transformation does not require numerically inverting the matrix. Indeed, since the matrix $\mathbf{S}$ of the transformation can be written as a non-zero scalar value multiplied by an orthogonal matrix

$$\mathbf{S} = a\,\mathbf{R}, \quad a \in \mathbb{R} \setminus \{0\}, \mathbf{R} \in \mathbb{R}^{d\times d}, \mathbf{R}^{-1} = \mathbf{R}^{\mathsf{T}},$$

where $\mathbf{R}$ is the rotation matrix, the inverse is simply

$$\mathbf{S}^{-1} = \frac{1}{a}\mathbf{R}^{-1} = \frac{1}{a}\mathbf{R}^{\mathsf{T}}.$$

Thanks to this property, the computation of the inverse matrix $\mathbf{S}^{-1}$ in (18) can be performed efficiently (since transposing a matrix is faster than inverting one).

4.2 Results

We investigate the robustness gain ensured by the invariance property under rotation and dilatation of the reference system by considering two examples in which a trajectory is learned and then executed changing the relative position between goal and starting point $\mathbf{g} - \mathbf{x}_0$. In these tests, we do not change $\mathbf{x}_0$ since DMPs are natively translational invariant. We compare the different behaviors obtained with and without the scaling term $\mathbf{S}$ defined in (20).* In the tests mentioned above we use mollifier-like basis functions (14), specifying that tests done with other classes of basis functions give similar results.

In the following, we refers to *classical* DMPs when talking about the DMPs implementation without exploiting the invariance property. Similarly, we refers to *extended* DMPs when talking about the implementation we introduced in Section 4.1, in which the invariance property is exploited by using, as linear invertible transformation, the roto-dilatation defined in (20).

In the first simulation, we generate the desired curve in the plane:

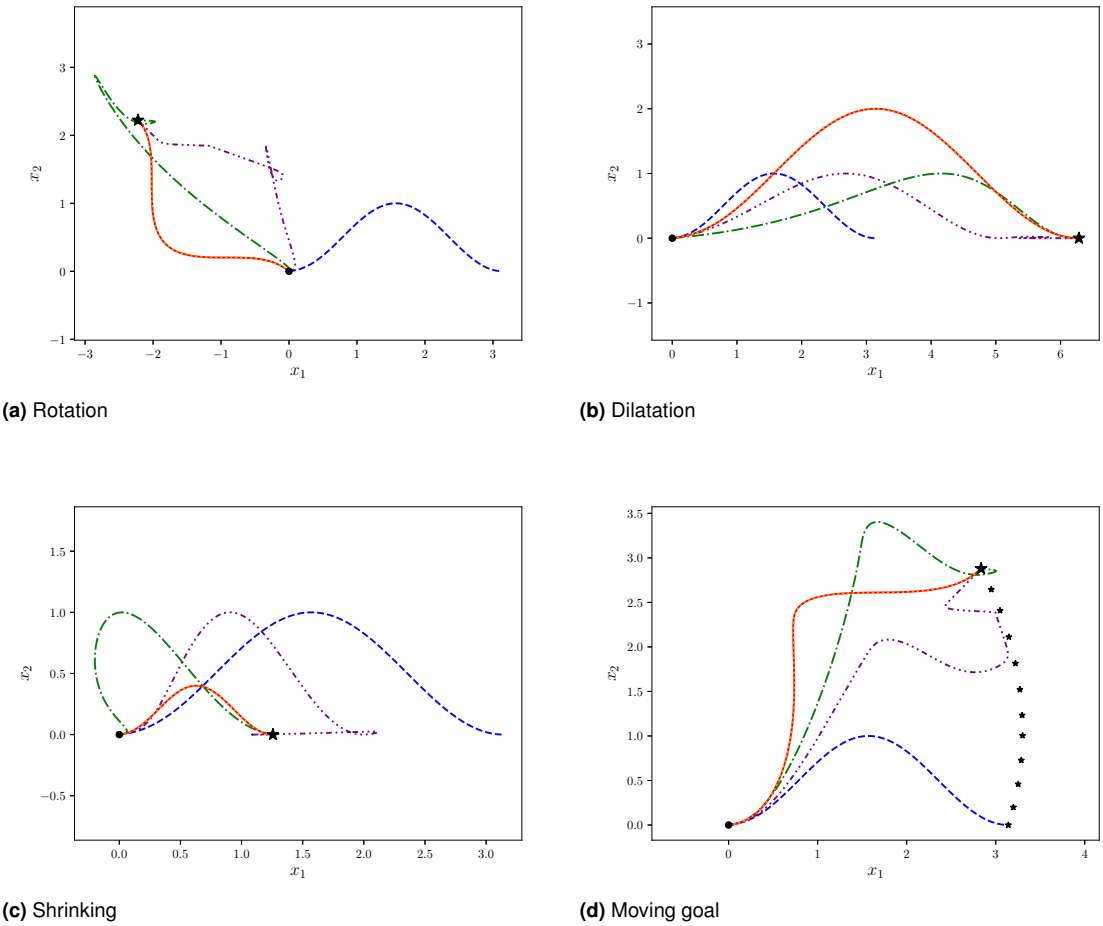
$$\mathbf{x}(t) = \left(t, \sin^2(t)\right), \quad t \in [0, \pi]. \tag{21}$$

Then we perform the learning step to compute the weights ω_i, and we test the generalization properties of DMPs by changing in different manners the goal position (both in a static and a dynamic way). All tests are executed both using classical and extended DMPs.
We performed these tests with elastic term $K = 1000$, and damping term $D = 2\sqrt{1000} \approx 63.25$ for both directions x_1 and x_2, that is $\mathbf{K} = K\mathbf{I}_2$ and $\mathbf{D} = D\mathbf{I}_2$, being $\mathbf{I}_2$ the 2×2 identity matrix. We test the generalization by using two different values of α in the canonical system (2), $\alpha = 4$ and $\alpha = 1$. Figure 6 shows the results of our tests.
In the second simulation, the desired curve is:

$$\mathbf{x}(t) = \left(t^2\,\cos(t), t\,\sin(t)\right), \quad t \in [0, 2\pi]. \tag{22}$$

Also in this case we test the generalization properties of both classical DMPs and extended DMPs when the relative position $\mathbf{g} - \mathbf{x}_0$ is changed. The main different is that in this second test (see Figure 7) we set $\alpha = 4$, and we use two

* We will not compare the results with the original DMPs (9) since the drawbacks of such formulation have already been discussed in the literature Park et al. (2008); Hoffmann et al. (2009); Pastor et al. (2009).



(a) Rotation

(b) Dilatation

(c) Shrinking

(d) Moving goal

Figure 6. Different behaviors obtained when changing the goal position. The desired curve is given by (21). In all four plots, the desired curved is plotted using the dashed line. The execution of extended DMP is shown by the full line for $\alpha = 4$, and by the dotted line for $\alpha = 1$ (in all these experiments, they overlap). The execution of classical DMPs are marked with the dash-dotted line when $\alpha = 4$, and with the one dash three dots line when α is set to 1. The dot and star mark, respectively, the desired starting and goal positions $\mathbf{x}_0$ and $\mathbf{g}$. The stars trial in Figure 6d shows the movement of the goal.

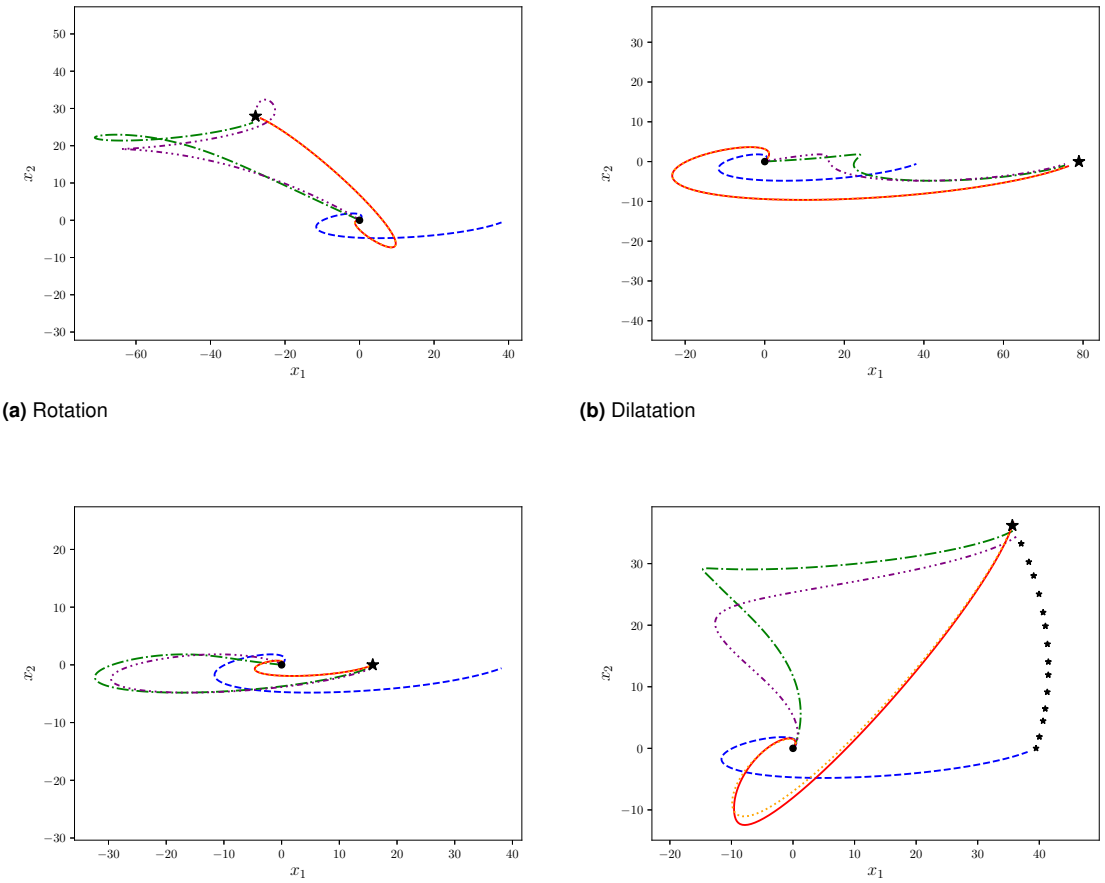
different values for $\mathbf{K}$ (and, thus, for $\mathbf{D}$). Indeed, we set $\mathbf{K} = K\mathbf{I}$ and set K to assume value 1000 and 100. In both cases, the damping term is set to $\mathbf{D} = \sqrt{K}\,\mathbf{I}_2$.

The tests show how extended DMPs are more robust than the classical DMP formulation, since the trajectories generated by taking advantage of the invariance property have a shape that closely resembles the learned trajectory, while classical DMPs may generate trajectories that do not resemble the learned behavior. Moreover, we notice that the goodness of the generalization of classical DMPs heavily depends on the choice of the parameters. For instance, in the cases of dilatation and moving goal for (21), they generalize well when $\alpha = 4$, but fails when $\alpha = 1$ (see Figure 6b and 6d). Similarly, we observe that the generalization of classical DMPs heavily depends also on the choice of parameter $\mathbf{K}$ (and, consequently, $\mathbf{D}$). For instance, it is possible to observe that the trajectories are different, even if the change in goal position is the same, see, for instance, Figure 7d. On the other hand, we see that the generalization of the trajectory is robust against the particular choice of hyperparameters when using extended DMPs. Indeed, in almost all our tests, the generalizations obtained by extended DMPs completely overlap when changing goal position. The only case in which there is an actual difference, even if hardly noticeable, is for the test in Figure 7d. This is due to two factors: the scaling matrix $\mathbf{S}$ depends explicitly on time, and different values for $\mathbf{K}$ influence the un-perturbed evolution of dynamical system (10). On the other hand, we observe that, in the case presented in 6d, there is no difference between the two trajectories, even if $\mathbf{S}$ depends on time, because the un-perturbed evolution of the system is not influenced by the change of α

5 Learning from Multiple Trajectories

DMPs have been used to learn trajectories from a single demonstration, because a set of demonstrated trajectories in general does not have the same starting and ending points, making the learning phase highly non-trivial.

So-called *Stylistic DMPs* Matsubara et al. (2010, 2011) were developed to learn from multiple demonstrations by introducing an additional variable, called *style parameter*, and by making the execution dependent on this new variable. This approach is useful when the "style" is needed to describe a trajectory. For instance, the trajectory may depend from the height of an obstacle. However, when the style is not



(a) Rotation

(b) Dilatation

(c) Shrinking

(d) Moving goal

Figure 7. Different behaviors obtained when changing the goal position. The desired curve is given by (22). In all four plots, the desired curved is plotted using the dashed line. The execution of extended DMP is shown by the full line for $K = 1000$, and by the dotted line for $K = 100$ (in Figures 7a–7c, they overlap). The execution of classical DMPs are marked with the dash-dotted line when $K = 1000$, and with the one dash three dots line when K is set to 100. The dot and star mark, respectively, the desired starting and goal positions $\mathbf{x}_0$ and $\mathbf{g}$. The stars trial in Figure 7d shows the movement of the goal.

an issue, this approach introduces additional and undesired variables that increase the complexity of the problem.

5.1 Linear Regression Over Aligned DMPs

We propose a technique that allows to extract a unique set of weights $\mathbf{w}_i \in \mathbb{R}^d$, $i = 0, 1, \ldots, N$, from a set of multiple observations as a single linear regression problem.

We consider a set of M *demonstrated trajectories* $\{\mathbf{x}^{(j)}(t)\}_{j=1}^{M}$. Using the technique introduced in Section 4.1 we transform each trajectory in such a way that all the starting and ending points are the same. We choose $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{g} = \mathbf{1}$, but we emphasize that any choice satisfying $\mathbf{x}_0 \neq \mathbf{g}$ would give the same results. We compute the roto-dilatation matrices, for $j = 1, 2, \ldots, M$,

$$\mathbf{S}^{(j)} = \frac{\|\mathbf{1} - \mathbf{0}\|}{\left\|\mathbf{x}^{(j)}(t_f) - \mathbf{x}^{(j)}(t_0)\right\|} \mathbf{R}^{\widehat{\mathbf{1}-\mathbf{0}}}_{\widehat{\mathbf{x}^{(j)}(t_f)-\mathbf{x}^{(j)}(t_0)}}, \tag{23}$$

and use these matrices to create the new set of transformed trajectories $\{\check{\mathbf{x}}^{(j)}(t) = \mathbf{S}^{(j)}\mathbf{x}^{(j)}(t)\}_{j=1,2,\ldots,M}$.
Next, we perform a *time scaling*, so that each curve has $[0, T]$ as time domain, for a given $T > 0$. This step is done as follows: let t_0 and t_1 be the initial and final time of a given curve $\check{\mathbf{x}}^{(j)}(t)$. The new parametrization of the curve is then

$$\tilde{\mathbf{x}}^{(j)}(t) = \check{\mathbf{x}}^{(j)}\left(\frac{T(t - t_0)}{t_1 - t_0}\right). \tag{24}$$

From these two step, we obtain a new set of 'transformed' curves $\{\tilde{\mathbf{x}}^{(j)}(t)\}_{j=1}^{M}$, each with time domain $[0, T]$, and $\mathbf{0}$ and $\mathbf{1}$ as starting and ending points respectively.
Equation (10a) allows to compute the set of forcing terms $\{\mathbf{f}^{(j)}(s)\}_{j=1,2,\ldots,M}$, and then we are able to compute the set of weights $\{\mathbf{w}_i\}_{i=0,1,\ldots,N}$ that minimizes the sum of the squared errors (w.r.t. the L_2-norm) between the function generated using (3) and the forcing terms $\mathbf{f}^{(j)}(s)$, $j = 1, 2, \ldots, M$.
For this purpose, we decompose the problem in the independent directions. For each direction $p = 1, 2, \ldots, d$, we look for the *weight vector* $\boldsymbol{\omega}^\star = [\omega_0^\star, \omega_1^\star, \ldots, \omega_N^\star]^\mathsf{T} \in \mathbb{R}^{N+1}$ satisfying

$$\boldsymbol{\omega}^\star = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^{N+1}} \underbrace{\sum_{j=1}^{M} \left\| \frac{\sum_{i=0}^{N} \omega_i \varphi_i(s)}{\sum_{i=0}^{N} \varphi_i(s)} s - f^{(j)}(s) \right\|_2^2}_{F(\boldsymbol{\omega})},$$

where $f^{(j)}(s)$ denotes the p-th component of $\mathbf{f}^{(j)}(s)$.
The existence and uniqueness of a minimum for F is guaranteed by its continuity and strict convexity. Moreover, since F is smooth, $\boldsymbol{\omega}^\star$ is the vector that nullify its gradient: $\nabla_{\boldsymbol{\omega}} F(\cdot)|_{\boldsymbol{\omega}^\star} = \mathbf{0}$. Let us denote with $\mathbf{G}$ the gradient of F: $\mathbf{G}(\cdot) \doteq \nabla_{\boldsymbol{\omega}} F(\cdot) : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$. Its h-th component, denoted by G_h is (recalling that $\|\zeta(s)\|_2^2 = \int_{s_0}^{s_1} (\zeta(s))^2 \, ds$)

$$
\begin{aligned}
G_h(\boldsymbol{\omega}) &= \frac{\partial F}{\partial \omega_h}(\boldsymbol{\omega}) \\
&= \sum_{j=1}^{M} \int_{s_0}^{s_1} 2 \left(\frac{\sum_{i=0}^{N} \omega_i \varphi_i(s)}{\sum_{i=0}^{N} \varphi_i(s)} \, s - f^{(j)}(s) \right) \\
&\qquad \left(\frac{\varphi_h(s)}{\sum_{i=0}^{N} \varphi_i(s)} \, s \right) ds
\end{aligned}
$$

Function G_h is linear in $\boldsymbol{\omega}$, thus the minimization problem can be written as a linear system:

$$
\mathbf{A}\,\boldsymbol{\omega} = \mathbf{b}. \tag{25}
$$

The component in row h and column k of $\mathbf{A}$, a_{hk}, is, for $h, k \in \{0, 1, \ldots, N\}$,

$$
\begin{aligned}
a_{hk} &= \frac{\partial G_h}{\partial \omega_k}(\omega) \\
&= \sum_{j=1}^{M} \int_{s_0}^{s_1} 2 \frac{\varphi_h(s)\varphi_k(s)}{\left(\sum_{i=0}^{N} \varphi_i(s)\right)^2} s^2 \, ds \\
&= 2M \int_{s_0}^{s_1} \frac{\varphi_h(s)\varphi_k(s)}{\left(\sum_{i=0}^{N} \varphi_i(s)\right)^2} s^2 \, ds,
\end{aligned} \tag{26}
$$

while the h-th component of the vector $\mathbf{b}$, b_h, is, for $h = 0, 1, \ldots, N$,

$$
b_h = \sum_{j=1}^{M} \int_{s_0}^{s_1} 2 \frac{\varphi_h(s)}{\sum_{i=0}^{N} \varphi_i(s)} f^{(j)}(s) \, s \, ds. \tag{27}
$$

Using any quadrature formula (e.g. Simpson's rule), the integrals in equations (26) and (27) can be computed for each $h, k = 0, 1, \ldots, N$ giving matrix $\mathbf{A}$ and vector $\mathbf{b}$ of (25). Thus, we can solve the linear problem and obtain $\boldsymbol{\omega}^\star$. We recall that this procedure has to be reaped for each direction $p = 1, 2, \ldots, d$.

The algorithm to perform regression is summarized in Algorithm 1.

Remark 3. Although we presented the algorithm using mollifier-like basis functions $\{\varphi_i\}_{i=0,1,\ldots,N}$, this algorithm works for any choice of the set of basis functions.

Remark 4. This approach does not encode information about 'styles', differently from Matsubara et al. (2011). Thus, the generalization of the DMP depends only on the starting and goal positions $\mathbf{x}_0$ and $\mathbf{g}$.

Remark 5. The presented method performs classical linear regression over a set of observed trajectories, and it gives a non-probabilistic result, differently from other approaches, such as Probabilistic Movement Primitives Paraschos et al. (2013) and Kernelized Movement Primitives Huang et al. (2019), which are purely probabilistic approaches.

Algorithm 1 Regression Over Multiple Demonstrations

Input: Set of desired trajectories $\{\mathbf{x}^{(j)}(t)\}_{j=1,2,\ldots,M}$, set of basis functions $\{\varphi_i(s)\}_{i=0,1,\ldots,N}$, DMPs hyperparameter $\mathbf{K}, \mathbf{D}$ and α, final time $T > 0$;
Output: Set of weights $\{\boldsymbol{\omega}^{(p)}\}_p$ for each direction $p = 1, 2, \ldots, d$.

▷ Align the trajectories (both temporally and spatially)
for $j = 1, 2 \ldots, M$ **do**
  Compute $\mathbf{S}^{(j)}$ using (23)
  Compute $\breve{\mathbf{x}}^{(j)}(t) = \mathbf{S}^{(j)}\mathbf{x}(t)$
  Compute $\tilde{\mathbf{x}}^{(j)}(t)$ using (24)
end for
Evaluate the canonical system extrema $s_0 = 1, s_1 = e^{-\alpha T}$
▷ Compute the matrix $\mathbf{A} = [a_{hk}]_{h,k=0,1,\ldots,N}$
for $h = 0, 1, \ldots, N$ **do**
  $a_{hh} = 2M \int_{s_0}^{s_1} \frac{\varphi_h(s)^2}{\left(\sum_{i=0}^{N} \varphi_i(s)\right)^2} s^2 \, ds$
  for $k = h+1, h+2, \ldots, N$ **do**
    $a_{hk} = 2M \int_{s_0}^{s_1} \frac{\varphi_h(s)\varphi_k(s)}{\left(\sum_{i=0}^{N} \varphi_i(s)\right)^2} s^2 \, ds$
    $a_{kh} = a_{hk}$
  end for
end for
▷ For loop along the directions
for $p = 1, 2, \ldots, d$ **do**
  For each trajectory $\tilde{\mathbf{x}}^{(j)}(t)$ compute the forcing term $f^{(j)}(s)$ for direction p using (10a).
▷ Compute the vector $\mathbf{b} = [b_h]_{h=0,1,\ldots,N}$
  for $h = 0, 1, \ldots, N$ **do**
    $b_h = \sum_{j=1}^{M} \int_{s_0}^{s_1} 2 \frac{\varphi_h(s)}{\sum_{i=0}^{N} \varphi_i(s)} f^{(j)}(s) \, s \, ds$
  end for
  Solve the minimization problem: $\mathbf{A}\boldsymbol{\omega}^{(p)} = \mathbf{b}$
end for

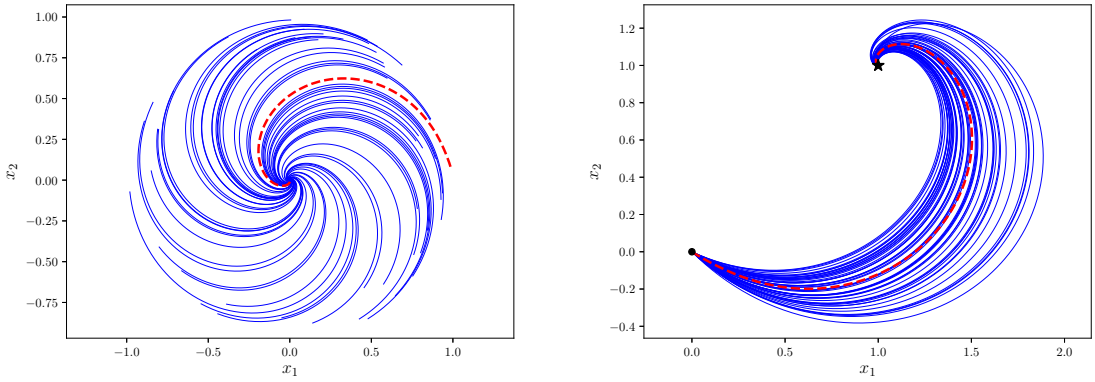
5.2 Results

In this Section, we teste the ability of learning from multiple demonstrations in two ways: at first, we create a set of trajectories $\{\mathbf{x}^{(j)}(t) = (x_1^{(j)}(t), x_2^{(j)}(t))\}_j$ by integrating a known dynamical system; and secondly we use the trajectories obtained by the execution of a task by a 7-DOF Panda industrial manipulator by Franka Emika.

For the test with synthetic data, we generate the trajectories computing the solution of the dynamical system

$$
\begin{cases} \dot{x}_1 = x_1^3 + x_2^2\, x_1 - x_1 - x_2 \\ \dot{x}_2 = x_2^3 + x_1^2\, x_2 + x_1 - x_2 \end{cases}, \tag{28}
$$

which is known to have a *alpha-limit* on the circumference of radius 1 and an attractive equilibrium at the origin. The set of (fifty) trajectories is generated by choosing a random angle $\theta_0 \in [0, 2\pi)$ and a random radius $\rho_0 \in (0.8, 1)$, and then setting as initial position $x_1(0) = \rho_0 \cos(\theta_0)$, $x_2(0) = \rho_0 \sin(\theta_0)$. Then the dynamical system is integrated using the *classic fourth-order Runge-Kutta method* up to a random final time $T \in (5, 7)$. After generating the set of observations, we use Algorithm 1 to generate a DMP. The DMP's hyper-parameters are $\mathbf{K} = K\,\mathbf{I}_2$, $\mathbf{D} = D\,\mathbf{I}_2$, with $K = 1000$ $D = 2\sqrt{K} \approx 63.25$, and $\alpha = 4$. Finally, we randomly select an initial point $\mathbf{x}_0 : \|\mathbf{x}_0\| \in (0.8, 1)$ and



(a) Synthetic test before rescaling.

(b) Synthetic test after rescaling.

Figure 8. Tests for our proposed approach to perform DMPs learning from multiple trajectories. The trajectories are obtained by integrating (28) with different initial conditions. In both plots the solid lines represent the demonstrated trajectories, while the dashed one is an execution of the learned DMP. The tests are plotted both before (Figure 8a) and after (Figure 8b) the spatial scaling step. In Figure 8b, the dot represents the initial position $\mathbf{x}_0 = (0, 0)$, while the star marks the goal $\mathbf{g} = (1, 1)$.

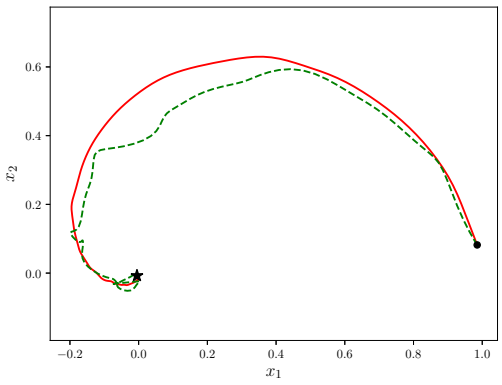


Figure 9. Comparison of two DMPs obtained by a noisy data-set. The solid line shows the trajectory obtained using the proposed method for regression over multiple observations. The dashed line shows the execution of a DMP learned from one sample of the dataset. The dot represent the initial position $\mathbf{x}_0$, while the star represent the goal $\mathbf{g}$.

execute the obtained DMP by setting as goal the attractive equilibrium of the system: $\mathbf{g} = (0, 0)$.

Figure 8 shows the results of these tests. In particular Figure 8a shows the set of trajectories obtained by integrating the ODE (28), together with an execution of the obtained DMP. Figure 8b shows the same trajectories when spatially rescaled to have $\mathbf{0}$ and $\mathbf{1}$ as starting and ending points respectively.

To highlight the usefulness of learning from a set of observations, we added a Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ of null mean and variance $\sigma^2 = 5 \cdot 10^{-5}$ to the set of trajectories obtained by solving the dynamical system (28). We then use Algorithm 1 to extract a DMP (the hyper-parameters are the same of the previous experiment). Moreover, we learned a second DMP, with the same hyper-parameters, using as desired trajectory a single trajectory of the dataset.

In Figure 9 it can be seen that learning from a single sample of the dataset results in a DMP with undesired oscillation. On the other hand, performing regression heavily reduce these oscillations.

For the experiments on a real setup, we perform a *Peg & Ring* task, in which some colored rings have to be moved to the corresponding peg, using an industrial manipulator. The task consists of two gestures, namely `move` (in which the robotic arm has to reach the ring) and `carry` (in which the robot has to carry the ring to the corresponding peg), that have to be repeated for each ring (in our case we have four rings). We executed the task for a total of fifty times, changing each time the grasping and leaving point (i.e. the initial and final position of both gestures), obtaining a total of 200 samples for each gesture.

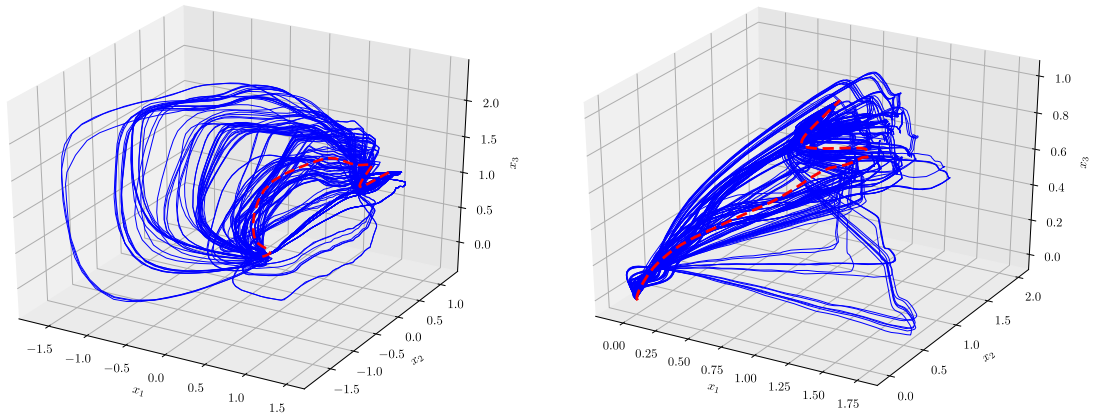
Figure 10a and 10b show the trajectories of, respectively, `move` and `carry` obtained with the industrial manipulator together with an execution of the obtained DMP. These are plotted after the rescaling so that $\mathbf{x}_0 = \mathbf{0}$, and $\mathbf{g} = \mathbf{1}$ for both gestures.

These tests show the capability of Algorithm 1 to extract a common behavior from multiple demonstrations.

6 Conclusions

In this work, we have highlighted some of the weak aspects of the DMP framework, proposing three major modifications to overcome them.

At first, we proposed a new formulation for the set of basis functions. In particular, we introduced a new set of basis functions, showing how it is able to well approximate the desired forcing term while having both the desirable properties of being smooth and, most importantly, compactly supported. Secondly, we proposed a strategy to exploit the invariance under affine transformation of the DMP formulation (10). In particular, we showed how to generalize the learned trajectory to any length scale and any rotation of the reference frame maintaining the qualitative shape of the learned trajectory, specifying that the same approach can be extended to any invertible transformation. Finally, we solved one of the major issues in DMPs, that is the inability of



(a) `move` **(b)** `carry`

Figure 10. Experiments on the real robot. For both gestures (`move` and `carry`), the trajectories of the dataset are plotted using solid lines, while the execution of the obtained DMP is plotted using a dashed line. All the trajectories are plotted after the rescaling, so that $\mathbf{x}_0 = \mathbf{0}$, and $\mathbf{g} = \mathbf{1}$ for both gestures.

the original framework to learn a common behavior from multiple observations.

As future work, we aim to extend these improvements to the unit quaternion formulation of DMPs, in order to make more robust also the orientation formulation of DMPs.

Moreover, we aim to use the proposed formulation to improve the 'Surgical Task Automation Framework' presented in Ginesi et al. (2019b), in which DMPs are used to generate surgical gestures.

A Condition Number Theory

Here we present a quick recall on the theory of condition number of matrices.

Consider the linear problem $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a non singular matrix, $\mathbf{b} \in \mathbb{R}^N$ is a vector, and $\mathbf{x} \in \mathbb{R}^N$ is the unknown vector. The *condition number* of a non-singular matrix is defined as

$$\text{cond}(\mathbf{A}) \stackrel{\text{def}}{=} \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\|,$$

where with $\|\mathbf{A}\|$ we denote the 2-norm of the matrix $\mathbf{A}$.

When the linear system is numerically solved, we obtain a solution $\tilde{\mathbf{x}}$. Let us define the *residual* $\mathbf{r}$ as

$$\mathbf{r} \stackrel{\text{def}}{=} \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}.$$

From $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b} - \mathbf{r}$ and $\mathbf{A}\mathbf{x} = \mathbf{b}$, it follows $\mathbf{A}(\mathbf{x} - \tilde{\mathbf{x}}) = \mathbf{r}$, from which

$$\|\tilde{\mathbf{x}} - \mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \, \|\mathbf{r}\| \, .$$

Using the inequality $\|\mathbf{b}\| \leq \|\mathbf{A}\| \, \|\mathbf{x}\|$, we obtain the following result on the relative error of the solution $\mathbf{x}$:

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\| \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = \text{cond}(\mathbf{A}) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

The above inequality shows that the relative error made when solving a linear system is amplified by the condition number of the matrix $\mathbf{A}$.

Declaration of conflicting interests

The Authors declare that there is no conflict of interest.

Funding

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, ARS (Autonomous Robotic Surgery) project, grant agreement No. 742671.

References

Amor HB, Neumann G, Kamthe S, Kroemer O and Peters J (2014) Interaction primitives for human-robot cooperation tasks. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 2831–2837.

Gams A, Nemec B, Ijspeert AJ and Ude A (2014) Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Trans. Robotics* 30(4): 816–830.

Ginesi M, Meli D, Calanca A, Dall'Alba D, Sansonetto N and Fiorini P (2019a) Dynamic movement primitives: Volumetric obstacle avoidance. In: *2019 19th International Conference on Advanced Robotics (ICAR)*. pp. 234–239. DOI:10.1109/ICAR46387.2019.8981552.

Ginesi M, Meli D, Nakawala HC, Roberti A and Fiorini P (2019b) A knowledge-based framework for task automation in surgery. In: *2019 19th International Conference on Advanced Robotics (ICAR)*. pp. 37–42. DOI:10.1109/ICAR46387.2019.8981619.

Hoffmann H, Pastor P, Park DH and Schaal S (2009) Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, pp. 2587–2592.

Huang Y, Rozo L, Silvério J and Caldwell DG (2019) Kernelized movement primitives. *The International Journal of Robotics Research* 38(7): 833–852.

Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P and Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation* 25(2): 328–373.

Ijspeert AJ, Nakanishi J and Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2. IEEE, pp. 1398–1403.

Ijspeert AJ, Nakanishi J and Schaal S (2003) Learning attractor landscapes for learning motor primitives. In: *Advances in neural information processing systems*. pp. 1547–1554.

Kappler D, Pastor P, Kalakrishnan M, Wüthrich M and Schaal S (2015) Data-driven online decision making for autonomous manipulation. In: *Robotics: Science and Systems*.

Kulvicius T, Ning K, Tamosiunaite M and Worgötter F (2012) Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics* 28(1): 145–157.

Matsubara T, Hyon SH and Morimoto J (2010) Learning stylistic dynamic movement primitives from multiple demonstrations. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. pp. 1277–1283.

Matsubara T, Hyon SH and Morimoto J (2011) Learning parametric dynamic movement primitives from multiple demonstrations. *Neural networks* 24(5): 493–500.

Paraschos A, Daniel C, Peters JR and Neumann G (2013) Probabilistic movement primitives. In: *Advances in neural information processing systems*. pp. 2616–2624.

Park DH, Hoffmann H, Pastor P and Schaal S (2008) Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*. IEEE, pp. 91–98.

Pastor P, Hoffmann H, Asfour T and Schaal S (2009) Learning and generalization of motor skills by learning from demonstration. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, pp. 763–768.

Pastor P, Kalakrishnan M, Righetti L and Schaal S (2012) Towards associative skill memories. In: *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, pp. 309–315.

Pastor P, Righetti L, Kalakrishnan M and Schaal S (2011) Online movement adaptation based on previous sensor experiences. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 365–371.

Saveriano M, Franzel F and Lee D (2019) Merging position and orientation motion primitives. In: *International Conference on Robotics and Automation (ICRA), 2019*.

Schaal S (2006) Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In: *Adaptive motion of animals and machines*. Springer, pp. 261–280.

Schaback R (2007) A practical guide to radial basis functions. *Electronic Resource* 11.

Ude A, Gams A, Asfour T and Morimoto J (2010) Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics* 26(5): 800–815.

Ude A, Nemec B, Petrić T and Morimoto J (2014) Orientation in cartesian space dynamic movement primitives. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 2997–3004.

Wang R, Wu Y, Chan WL and Tee KP (2016) Dynamic movement primitives plus: For enhanced reproduction quality and efficient trajectory modification using truncated kernels and local biases. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, pp. 3765–3771.

Wendland H (1995) Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics* 4(1): 389–396.

Zhelezov OI (2017) N-dimensional rotation matrix generation algorithm. *American Journal of Computational and Applied Mathematics* 7(2): 51–57.