Fitting Parameters of Linear Dynamical Systems to Regularize Forcing Terms in Dynamical Movement Primitives

Freek Stulp[†], Adrià Colomé[*], and Carme Torras[*]

***Abstract*— Due to their flexibility and ease of use, Dynamical Movement Primitives (DMPs) are widely used in robotics applications and research. DMPs combine linear dynamical systems to achieve robustness to perturbations and adaptation to moving targets with non-linear function approximators to fit a wide range of demonstrated trajectories.**

We propose a novel DMP formulation with a generalized logistic function as a delayed goal system. This formulation inherently has low initial jerk, and generates the bell-shaped velocity profiles that are typical of human movement. As the novel formulation is more expressive, it is able to fit a wide range of human demonstrations well, also without a non-linear forcing term. We exploit this increased expressiveness by automating the fitting of the dynamical system parameters through optimization. Our experimental evaluation demonstrates that this optimization regularizes the forcing term, and improves the interpolation accuracy of parametric DMPs.

I. Introduction

Introduced over twenty years ago [1], Dynamical Movement Primitives (DMPs) remain an active topic of research [2]–[8]. DMPs represent movements by combining linear differential equations with non-linear forcing terms. The forcing term is a gated function approximator, whose open parameters are determined through regression, based on a demonstrated trajectory. In contrast, the parameters of the dynamical systems are tuned by hand, which is straightforward, and therefore usually not given much consideration[1].

We investigate the fitting of the linear dynamical systems (LDS) to a demonstrated trajectory *before* fitting the non-linear function approximators. As the LDS in current DMP formulations cannot represent typical human demonstrations well, hitherto not much could be gained from fitting LDS parameters. Doing so has thus hardly been explored [9],

Our first contribution is therefore to propose a modification of the DMP formulation that enables the LDS to represent bell-shaped velocity profiles – which are typical of human movement and thus human demonstrations – even without a forcing term. The modification is to use a generalized logistic differential equation as a delayed goal system [10]. The new formulation also has a richer parameterization than previous formulations, enabling the LDS to express a much wider range of demonstrations, as illustrated in Fig. 1.

With the more expressive formulation now available, our second contribution is to automate the hand-tuning of the LDS parameters through optimization. Our evaluation on several datasets with human demonstrations shows that the optimization of LDS parameters improves the fitting without a forcing term for all DMP formulations, and that the effect is far more substantial with our novel formulation.

The third contribution is to demonstrate that if more 'fitting work' is done by the LDS, less work needs to be 'delegated' to the function approximators, leading to a regularization of their parameters. We consider such regularization to be a desirable property in itself, and also demonstrate its practical impact on the interpolation accuracy of a task-parameterized DMP [7], [11].

All implementations described in this paper are available as part of the open-source library `dmpbbo` [12].

The rest of this paper is structured as follows. In the next section, we describe related work and previous DMP formulations. In Section III, we propose the novel DMP formulation, and highlight its expressiveness. The procedure for fitting the parameters of the dynamical systems is described in Section IV, and it is evaluated on several datasets and tasks in Section V. We conclude with Section VI.

II. Related Work

Motion primitive representations can be classified according to whether they are time-dependent (e.g. DMPs), state-dependent (SEDS [14]), or can be both (TP-GMMs [15], ProMPs [8], [16], KMPs [17]). A comprehensive, recent overview of DMP research is provided in [2].

Whereas TP-GMMs, ProMPs and KMPs are typically used in position space, DMPs and SEDS inherently represent velocities and/or accelerations. The aim of this paper is not to argue or demonstrate the advantages of DMPs over SEDS, TP-GMMs, ProMPs or KMPs; the appropriate method depends on the properties of the task. But *if* DMPs are the appropriate approach for a particular problem, our DMP formulation has important advantages over other formulations.

We consider the work of Li et al. [9] to be most similar to our approach, in that it also optimizes the parameters of the dynamical systems. In [9], the aim is to tune these parameters for non-experts. Our approach rather aims at regularizing the forcing term, which hinges on providing more expressive dynamical systems (see Section III).

To focus on the main contributions of our paper, we do not take end-effector orientations into account. However, all movement primitive representations above, including our formulation, can be readily extended to represent orientations [3], [18]–[20].

[†]German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchner Str. 20, 82234 Weßling, Germany

[*]Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, Spain

[1]In [1] for instance, these parameters are briefly mentioned in a figure caption, along with the remark "The same parameters will be used throughout the article."

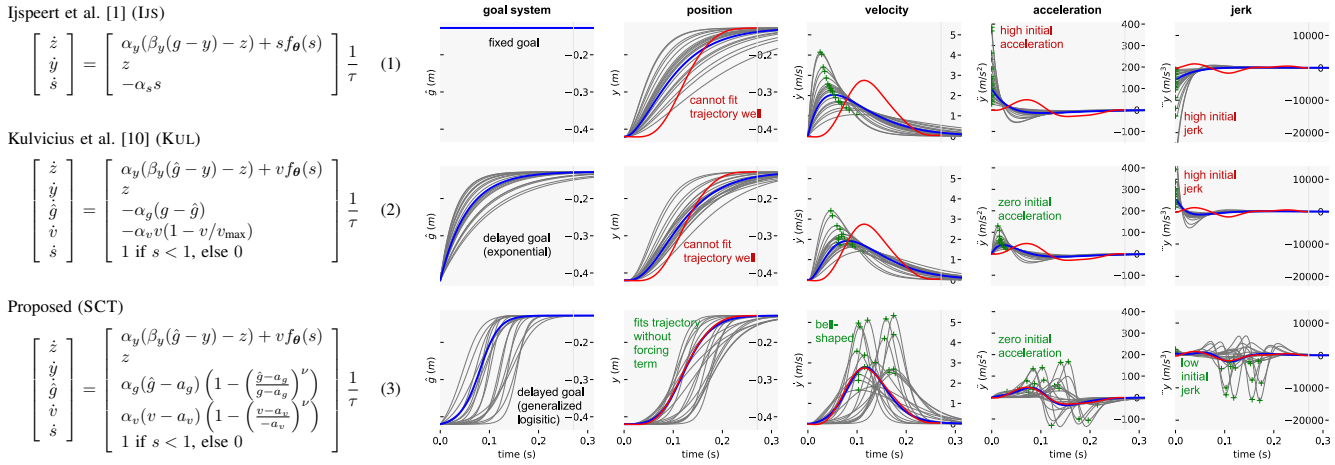

Fig. 1. Illustration of the contributions. By representing the delayed goal system with a generalized logistic differential equation (bottom row), a DMP can represent a much larger range of movements (gray trajectories generated by random sampling of the parameters of the dynamical systems). As the formulation is able to generate the bell-shaped velocity profiles typical of human movement (Section III), trajectories demonstrated by humans (an example of one of the y coordinates of a trajectory from [13] in red) can therefore often be fitted well with this novel formulation (blue), without requiring a non-linear forcing term. The parameters of the blue trajectory in the lower row have been acquired with gradient-free optimization (Section IV). This parameter fitting minimizes the targets for the function approximator that represents the forcing term, which leads to better numerical stability and generalization.

A. Initial DMP formulation (IJS formulation)

Dynamical Movement Primitives were introduced in [1]. DMPs consist of a spring-damper system, with a *forcing term* added to the accelerations[2].

$$\begin{bmatrix} \dot{z} \\ \dot{y} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} \alpha_y(\beta_y(g-y) - z + sf_{\boldsymbol{\theta}}(s) \\ z \\ -\alpha_v s \end{bmatrix} \frac{1}{\tau} \quad (4)$$

Movements are generated by integrating this system with the initial state $y = y(t_0)$ (e.g. the initial end-effector pose), whereby y will converge to the attractor state g (the goal).

The forcing term $sf_{\boldsymbol{\theta}}(s)$ modulates the acceleration profile. The parameters $\boldsymbol{\theta}$ of $f_{\boldsymbol{\theta}}$ are trained from a demonstrated trajectory through regression. The aim is for the DMP to fit the acceleration profile of the demonstrated trajectory.

The *canonical system* with state s decays exponentially from 1 towards 0 during the movement with $\dot{s} = -\alpha_s s$. It serves as an input phase variable for the forcing term $f_{\boldsymbol{\theta}}(s)$, which makes the DMP *autonomous*, i.e. independent of time. It also gates the function approximator with $sf_{\boldsymbol{\theta}}$. Because the gated forcing term converges towards 0, the overall system is guaranteed to converge towards the attractor state g of the spring-damper system [1].

To avoid overshooting, the spring-damper system is critically damped with $\beta_y = \alpha_y/4$. The time scaling factor τ allows the duration of the movement to be scaled. To improve invariance properties, various further scaling approaches of $f_{\boldsymbol{\theta}}$ have been proposed [21].

In multi-dimensional DMPs, y, z and g will be vectors, and one function approximator f_i is trained for each dimension $i = 1 \dots N$. The canonical system remains 1-D, and is shared between all dimensions to synchronize the movement of all dimensions. Also α_y and β_y are shared between dimensions. We use the 1-D DMP notation for simplicity, and consider multi-dimensional DMPs in Section IV-B.

[2]Readers for whom this introduction is too brief are encouraged to read the DMP tutorial in the accompanying open-source software:
https://github.com/stulp/dmpbbo/blob/master/tutorial/dmp.md

B. Delayed Goal System (KUL formulation)

Kulvicius et al. [10] proposed an adapted formulation to resolve several issues in the original formulation. The first issue was the high initial accelerations of the spring-damper system, see the fourth graph in the first row in Fig. 1. When fitting a DMP, the function approximator must counteract these high accelerations. Kulvicius et al. took the stance that the function approximator should not have to compensate for a shortcoming of the spring-damper system. They therefore introduced a *delayed goal system* $\hat{g}$ which starts at the initial state and (quickly) converges towards the goal state. It is also represented with a linear system $\dot{\hat{g}} = -\alpha_g(g - \hat{g})/\tau$ with initial state $\hat{g}(t_0) = y(t_0)$ and attractor state g, see the first graph in the second row in Fig. 1. As the initial states $y(t_0)$ and $\hat{g}(t_0)$ are the same, $\dot{z} = \alpha_y(\beta_y(\hat{g} - y) - z$ does not generate accelerations at the beginning, see the fourth graph in the second row in Fig. 1.

Another important contribution in [10] was to split the canonical system into a phase system (s) and a gating system (v), so that the forcing term becomes $vf_{\boldsymbol{\theta}}(s)$. It was demonstrated that replacing the exponentially decaying system with a logistic function (for gating) and a constant velocity system (for the phase) leads to more regularized training data for the function approximators, as our experiments will confirm.

See Eq. (2) for the full formulation. It is easy to implement, yet has a profound impact on the numerical stability and ease of applying DMPs to real-world tasks. We therefore consider [10] to be a key contribution to DMPs.

III. Increasing the Expressiveness of the Linear Dynamical Systems (LDS)

By introducing a delayed goal system, Kulvicius et al. [10] ensured that the LDS generate zero initial accelerations, alleviating the function approximator from having to compensate for the high accelerations in the initial formulation. We take this idea one step further, by proposing a generalized logistic equation for the delayed goal system, which ensures low initial jerk, and inherently generates bell-shaped velocity profiles. These are desirable properties, since they are typical for human point-to-point movements [22]. As is apparent from the fourth column of graphs in Fig. 1, IJS and KUL do not have these properties, and are not able to represent bell-shaped velocity profiles without a forcing term. Furthermore, our novel formulation is more expressive, with 4 open parameters instead of the 1 (α_y) and 2 (α_y and α_g) in the previous formulations.

A. The Generalized Logistic Differential Equation (GLDE)

GLDEs are commonly used in population dynamics [23] to describe the growth of a population Y towards its maximum carrying capacity K

$$\dot{Y} = \eta Y(1 - \left(\tfrac{Y}{K}\right)^{\nu}), \tag{5}$$

where right asymptote of this differential equation is K, and the left asymptote is 0. We can rewrite the equation by setting the left asymptote to A instead with

$$\dot{Y} = \eta(Y - A)\left(1 - \left(\tfrac{Y-A}{K-A}\right)^{\nu}\right). \tag{6}$$

B. GLDE as a delayed goal system in a DMP

We use the GLDE as a delayed goal system as follows. The parameter K corresponds to the goal g to which the movement should converge, and η is replaced with α_g/τ for compatibility with the DMP terminology:

$$\dot{\hat{g}} = \frac{\alpha_g}{\tau}(\hat{g} - a)\left(1 - \left(\frac{\hat{g} - a}{g - a}\right)^{\nu}\right), \tag{7}$$

The delayed goal $\hat{g}$ converges from its initial state $\hat{g}(t_0) = y(t_0)$ to its attractor state g. The state $\hat{g}$ is then used as the attractor state of the spring-damper system, so that it too converges to the target position g.

The novel formulation is shown in Eq. (3) in Fig. 1. The shape of using the GLDE as a delayed goal system is illustrated in the lower left graph of Fig. 1. We see that the resulting velocity profile of the overall DMP is bell-shaped for all variations of the parameters.

1) Inflection time as an explicit parameter: The left asymptote of the GLDE – A in Eq. (6) – corresponds to a in Eq. (7). It is a virtual parameter that has no physical interpretation in the context of DMPs. However, it can be computed from a desired inflection point with:

$$a = \frac{b\hat{g}_{t_0} - g}{b - 1}, \text{ with } b = \sqrt[\nu]{1 + \nu e^{\alpha_g \nu (t^* - t_0)/\tau}}. \tag{8}$$

Fig. 2 illustrates the shape of the delayed goal function for different inflection times t^* and the effect on the virtual asymptote a in (8). The clear interpretation of the inflection time makes it a convenient parameter to be able to set.

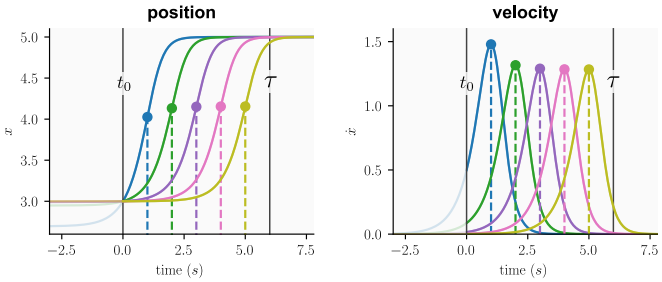

Fig. 2. Varying inflection times $t^* = 1, 2, 3, 4, 5$ for a generalized logistic function with $\tau = 6$, $y(t_0) = 3$, $g = 5.0$, $\alpha_g = 10$, $\nu = 2$. The virtual asymptote a is computed from these parameters, and visualized by extending the analytical solution before t_0.

2) Numerical stability: The virtual left asymptote a goes towards $-\infty$ and 0 for very small and large t^* respectively. In practice, this is not an issue, as $t^* < 0$ and $t^* > \tau$ are not sensible values.

If the end-goal g changes during the movement, numerical instabilities may arise when g drops below a. This is a real issue, which can be resolved by implementing Eq. (7) as follows:

$$\dot{d} = -\frac{\alpha_g}{\tau}(d - a)\left(1 - \left(\frac{d - a}{1 - a}\right)^{\nu}\right), \text{ with } d_{t_0} = 0 \tag{9}$$

$$\text{with } a = -\left(\sqrt[\nu]{1 + \nu e^{\alpha_g \nu (t^* - t_0)/\tau}} - 1\right)^{-1} \tag{10}$$

$$\hat{g} = y(t_0) + d(g - y(t_0)) \tag{11}$$

Here, d develops from 0 to 1, and thus $\hat{g}$ from $y(t_0)$ to g. Overall, this leads to the same behavior as in Eq. (7), but it is robust towards arbitrary changes of the goal during the movement because a does not depend on g. This approach is very similar to goal scaling [1], [21].

3) Illustrating the increased expressiveness: Fig. 1 shows the effect of varying the parameters of the different DMP formulations. With IJS there is only one parameter (α_y), and for KUL there are two (α_y, α_g). Their limited expressiveness inherently precludes them from fitting bell-shaped velocity profiles. In both cases, the maximum velocity cannot be set independently of the time at which this velocity is reached.

With the GLDE as a delayed goal system, there are 4 parameters: α_y, α_g, ν, t^*. The sensitivity analysis in Fig. 1 shows that decoupling the first three shape parameters from the timing of the maximum velocity (with t^*) enables a much wider range of velocity profiles. Furthermore, independent of the variations of these 4 parameters, the initial acceleration and jerk remain low, often orders of magnitude lower than for IJS or KUL.

A summary of the properties of the different formulations (without adding forcing term) is provided in Table I.

4) GLDE for the gating function: We also use a GLDE for the gating function which develops from 1 to 0. We have found that for high ν, even sharper declines towards the end of the movement can be achieved than with a standard logistic system as used in KUL (a special case with $\nu = 1$). This leads to further regularization of the forcing term. However, we have not found it advantageous to use it as a phase system [24].

	IJS	KUL	(novel) SCT
converges towards goal	**yes**	**yes**	**yes**
initial velocity	**zero**	**zero**	**zero**
initial acceleration	high	**zero**	**zero**
initial jerk	high	high	**zero / low**
bell-shaped velocity profiles	no	no	**yes**
# free parameters	1	2	**4**
free parameters	α_y	α_y, α_g	$\alpha_y, \alpha_g, t^*, \nu$

TABLE I

DMP FORMULATION PROPERTIES; GREEN INDICATES AN ADVANTAGE.

IV. AUTOMATING THE FITTING OF DYNAMICAL SYSTEM PARAMETERS

As Fig. 1 and Table I illustrate, IJS and KUL provide little room for tuning the dynamical system parameters to fit the bell-shaped velocity profiles typical of human movement. For this reason, this parameter tuning has not been emphasized in previous work. In contrast, the expressiveness of the novel formulation allows many different trajectories to be fitted well even without a forcing term. We exploit this by training a DMP in two phases: 1) optimize the parameters of the dynamical systems so that they best fit a demonstrated trajectory without a forcing term; 2) with these LDS parameters, train the forcing term, as is standard in DMP training.

A. Cost Function for the Optimization

To minimize the 'fitting work' that is delegated to the forcing term – which is in acceleration space – we regularize the absolute differences in acceleration between the demonstrated and reproduced trajectory over all times steps N and dimensions D:

$$c_{\ddot{y}} = \frac{1}{N \cdot D} \sum_{i=0}^{N_\tau} \sum_{d=0}^{D} |\ddot{y}_{d,t_i}^{\text{demo}} - \ddot{y}_{d,t_i}^{\text{repro}}| \tag{12}$$

This constitutes the summed L_1-norm in acceleration space. An advantage of penalizing accelerations – rather than regularizing the function approximator parameters – is that the approach becomes independent of the specific function approximator implementation that is used.

A further cost component is added to ensure timely convergence to the goal. It is not often discussed in the DMP literature, but if the parameter α_y is set too low, the movement converges towards the goal only very slowly, and y may still be far from the target g at the end of the movement at $t = \tau$. Some examples can be seen in second column of graphs in Fig. 1. When tuning parameters by hand, one simply sets α_y high enough so that this does not happen. To formalize this requirement in the cost function, we integrate the DMP 25% beyond τ (corresponding to M extra time steps), and sum the absolute difference between the goal and the trajectory for $t \geq \tau$:

$$c_g = \frac{1}{M \cdot D} \sum_{i=N_\tau}^{N_\tau+M} \sum_{d=0}^{D} |y_{d,t_i}^{\text{demo}} - y_{d,t_i}^{\text{repro}}| \tag{13}$$

The overall cost is $c = c_{\ddot{y}} + u \cdot c_g$, with weighting factor u. Furthermore, as not all LDS parameters are valid (e.g. $\nu > 0$, $0 < t^* < \tau$), these constraints are enforced during the optimization.

We perform the optimization of the LDS parameters in the last row of Table I with the evolutionary strategy PI^{BB} [25]. Alternatively, Bayesian optimization could be used; for such low-dimensional spaces the algorithm used is not critical. In future work, we will investigate if the optimal parameters can be derived analytically rather than through iterative optimization.

B. Decoupling LDS Parameters for each Dimension

Different dimensions of a demonstrated trajectory may require different dynamical system parameters for optimal fitting. For this reason, it is advantageous to run the optimization separately for each dimension. Rather than all dimensions sharing the same α_y, α_y etc. (as in IJS and KUL) each dimension then has its own parameter set, as is customary for the forcing term parameters $\boldsymbol{\theta}$ in $f_{\boldsymbol{\theta}}$. As shown in the empirical evaluation, this separation (or 'decoupling') has no impact on fitting quality for IJS and KUL, but it is substantial for the novel SCT formulation.

V. EMPIRICAL EVALUATION

All implementations in this evaluation are based on the `dmpbbo` open-source library [12]. A separate repository allows the results of this paper to be reproduced[3].

A. Benchmarking on three human demonstration datasets

The aim of this experiment is to demonstrate that 1) optimization of the dynamical system parameters leads to a regularization of the function approximator parameters; 2) our formulation leads to the best regularization due to its increased expressiveness.

1) Data: We use the first 12 trajectories from the two datasets described in [13] and [26]. The first contains human reaching movements towards a cup, acquired with a magnetic tracking device. The second contains robot end-effector trajectories for reaching towards a box, which were acquired through kinesthetic teaching. The datasets are described in more detail in [13], [26]. A third dataset was acquired in the context of this paper, and is described in more detail in Section V-B. The raw recorded data was filtered with a third-order Butterworth filter.

2) Procedure: There are three steps, which are performed separately for each trajectory in the dataset and for each of the three DMP formulations: 1) Evaluate how well a DMP with default parameters and without a function approximator fits each trajectories. The default parameters were taken from previous work [12], and were not tuned further for any of the experiments in this paper. 2) Optimize the LDS parameters, with and without coupling of the dimensions (see Section IV-B), and evaluate again. Optimization is performed with an evolutionary strategy based on reward-weighted averaging of the mean (PI^{BB}, see Section IV-A), with 10 samples per update, and 25 optimization updates. The initial parameters correspond to the default LDS parameters of the DMP, and

[3] `https://github.com/DLR-RM/dmpbbo-glde-evaluation`

the covariance of the sampling distribution is $(p/4)^2$ for each parameter p. The cost scaling parameter u is 10^4 in all experiments in this paper. 3) Fit the function approximators of the forcing term both for the default *and* the optimized LDS parameters to each trajectory. We use a radial basis function network with 10 basis functions for all datasets.

3) Result 1: The main results of step 1 and 2 are summarized in Fig. 3. The cost before optimization ($\mu \pm \sigma$ over the 12 trajectories) is shown in gray for each DMP formulation and dataset. The cost after optimization is depicted in blue (with coupled LDS parameters) and green (with separate LDS parameters for each dimension); for clarity, they are connected with the cost before optimization.

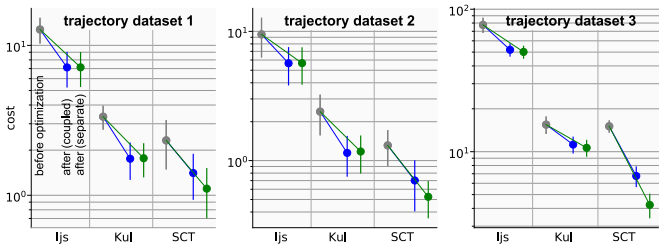

Fig. 3. Results of fitting the dynamical system parameters with different DMP formulations (IJS, KUL, SCT) and datasets (left: [13], center: [26], right: Section V-B). y-axis is logarithmic.

Analysis: From the results in Fig. 3, we draw the following conclusions. • The datasets are very different (from magnetic tracking to kinesthetic teaching on different robots), but the results are qualitatively similar. • All formulations benefit from LDS parameter optimization, which verifies the second contribution. • IJS does not fit the acceleration profiles well, neither before nor after optimization. There is no difference between optimizing LDS parameters coupled (blue) or separate (green). • KUL is able to fit much better, as it has 2 parameters for optimization, rather than 1. Again no difference between coupled/separate. • SCT fits the acceleration profiles best on all datasets, and doing so with separate LDS parameters leads to further improvement. This verifies the first contribution, that the novel formulation is more expressive, and better able to fit human motions.

4) Result 2: To illustrate the impact of the optimization on the distribution of the function approximator parameters (which is fitted in step 3 of this experiment), Fig. 4 depicts histograms of these parameters accumulated over all 12 trajectories.

Analysis: From Fig. 4, we draw the following conclusions: • Before and after optimization, the function approximator parameters with IJS are very large, i.e. very spread out around 0. • With KUL, values are much smaller. The main reason for this is the replacement of the exponentially decaying gating function in IJS with a sigmoid gating function. • SCT has the smallest means and the lowest standard deviations, i.e. -0.3 ± 0.91, -0.02 ± 0.42 and -0.12 ± 3.65 for the three datasets. In comparison to previous formulations, this leads to a reduction of function approximator parameter spread of 3.2 (2.97/0.91), 5.33 (2.24/0.42) and 3.2 (11.91/3.65) in

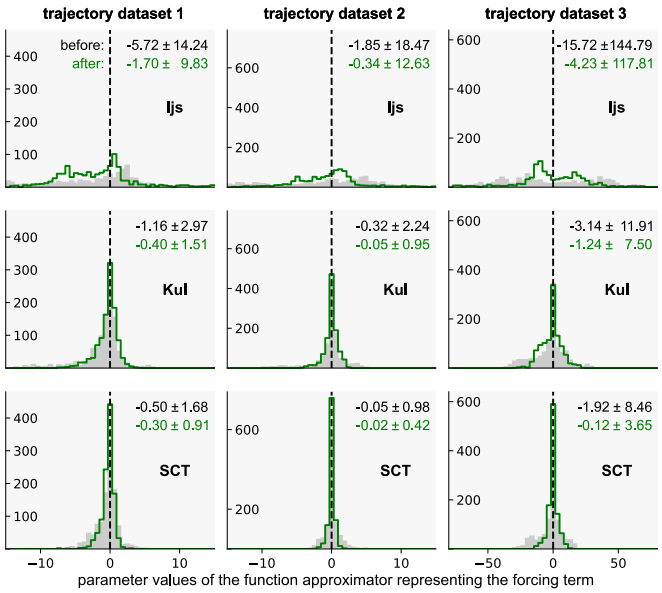

Fig. 4. Histogram of function approximator parameter values for each DMP type (top to bottom) and dataset (left to right), before (gray histogram) and after (green) optimization.

comparison to KUL, and at least a factor of 15 to IJS. This verifies contribution 3 of the paper, namely that both the novel formulation (contribution 1) and the optimization of the LDS parameters (contribution 2) contribute to regularization.

B. Impact of Regularization on Parametric DMPs

Regularization is generally a nice property, but not an aim in itself. The objective of the second experiment is to demonstrate the practical impact of regularization on task-parameterized motion primitives [7], [11], [15], [26], [27]. In parametric DMPs (pDMPs), parameters related to the task – for instance the location of a box [26] – modulate the DMP parameters. Since our aim is not to propose a novel pDMP formulation, but rather show the impact of regularization on its quality, we use the pDMP formulation originally proposed in [11]: 1) train a DMP on each trajectory in a dataset; 2) train a policy parameter function, which maps task parameters to DMP function approximator parameters. As we shall see, regularization in the first step is beneficial for training in the second step.

1) Data collection: The task is to hang a coat hanger on a rail at different positions, the position along the rail being the task parameter. On a UR5 robot, we gathered 4 trajectories for different positions, approximately 7cm apart. Trajectories were filtered with a Butterworth filter, and aligned with dynamic time warping. As the end-effector orientation was almost constant during the demonstration, it was not included in the pDMP training.

2) Procedure: The two-step procedure above was used to train one pDMP from the 4 trajectories. In the first step, the function approximator parameters were determined through regression. In the second step, the policy parameter function was trained. We used a Gaussian Process with a length parameter of 7cm. These steps were performed for all

three DMP formulations, with the default parameters for IJS and KUL, and the optimized parameters for SCT. During the optimization, the cost function was the sum over the individual costs for each of the 4 trajectories.

We evaluate two performance measures: 1) the mean distance at each time step between the demonstrated path and the path generated by the pDMP for the same task parameter as the demonstration. This evaluates the accuracy of fitting on the training set. 2) the same distance between the path between two demonstrations, and the path generated by the pDMP for the task parameter between these demonstration. This evaluates the accuracy of interpolation.

To validate the repeatability of our experiment, the entire procedure of data gathering and training was repeated 5 times. One of these 5 batches needed to be discarded due to a recording error.

3) Results: Fig. 5 shows the placement of the coat hanger for the different DMP formulations for one of the 4 batches. The video supplement shows the performed trajectories.

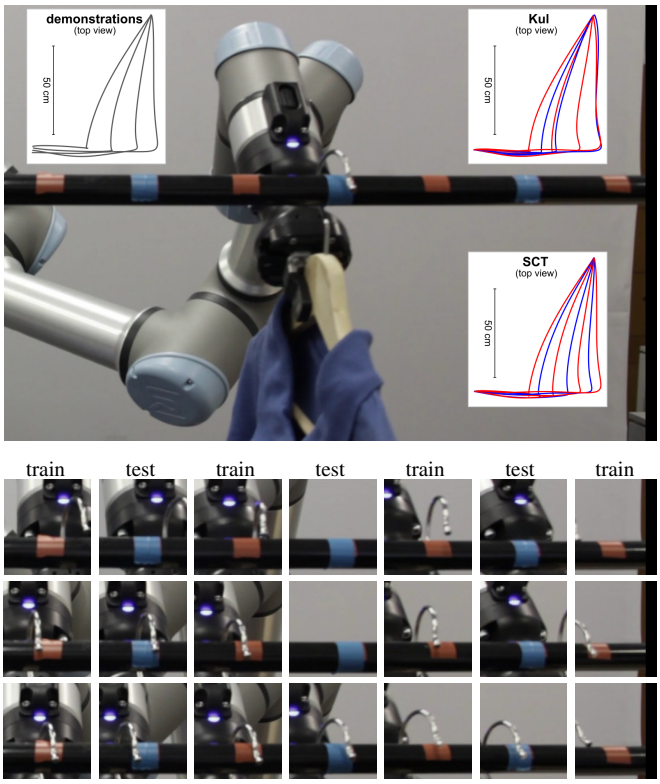

Fig. 5. Overview of the experiment. Demonstrations were given only for the red markers. The lower rows (IJS, KUL and SCT respectively) show snapshots of the video when the coat hanger is placed at each position.

Table II summarizes the quantitative results, by listing the distance between demonstrations and trajectories generated by the parametric DMP, averaged over all time steps. We report the mean and standard deviation over all 4 batches of demonstrations.

Analysis: Fig. 5 qualitatively shows that SCT (the last row) has more accurate placements both for the red (training) and blue (to test interpolation) markers. For IJS, the coat hanger can hardly be seen in the cropped images for any of the blue markers. From Table II we also conclude that the performance of IJS and KUL decreases substantially from the training to the test set, i.e. ($1.9 \rightarrow 3.0$ and $1.1 \rightarrow 2.3$ respectively). Due to the lower function approximator parameters resulting from LDS parameter optimization, SCT performs almost as well on training and test ($1.2 \rightarrow 1.3$). This demonstrates the practical relevance of the regularization (contribution 3), based on contributions 1 and 2.

	on demonstrations (training set)	with interpolation (test set)
IJS	1.9 ± 0.2	3.0 ± 1.0
KUL	1.1 ± 0.2	2.3 ± 0.8
SCT	1.2 ± 0.2	1.3 ± 0.9

TABLE II

AVERAGE DISTANCE (IN CM) BETWEEN DEMONSTRATIONS AND GENERATED TRAJECTORIES.

VI. CONCLUSION AND OUTLOOK

Most work on DMPs has focussed on extending the original formulation, or integrating DMPs into overarching frameworks for DMP selection and execution [2]. In this paper, we instead reconsidered the core formulation, and asked the question: What if the linear dynamical systems themselves could represent the movement, rather than leaving it to the function approximator? By replacing the delayed goal system from [10] with a generalized logistic system, we indeed found that bell-shaped velocity profiles can be generated, thereby ensuring low initial jerk.

Fitting the LDS parameters prior to fitting the function approximator leads to smaller function approximator parameter values for all formulations. Due to its expressiveness, the effect is more substantial with the novel formulation, We demonstrated the practical impact of regularization on a parametric DMP.

Minimum-jerk and bell-shaped velocity profiles are defining features of human movement. However, not all tasks require such a profile, i.e. some may even require high jerk. Due to its higher expressiveness and the resulting ability to represent all movements that IJS and KUL can (as long as they have zero initial acceleration and jerk), we expect our novel formulation to still be advantageous in these cases.

In our future work, we will study further human movement types, and see if other dynamical system equations may improve fitting of the resulting demonstrations. Furthermore, we intend to study the impact of regularization on reinforcement learning with DMPs.

ACKNOWLEDGMENTS

This work was partially funded by the European ERC/EIC/Horizon projects with grant numbers 741930 (CLOTHILDE), 951992 (VeriDream), and 101070596 (euROBIN), as well the CSIC project 202350E080 (ClothIRI), and the MCIN/AEI project PID2020-118649RB-I00 (CHLOE-GRAPH).

REFERENCES

[1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[2] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[3] L. Rozo and V. Dave, "Orientation probabilistic movement primitives on riemannian manifolds," in *Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 2021, pp. 373–383.

[4] Z. Cui, W. Ma, J. Lai, H. K. Chu, and Y. Guo, "Coupled multiple dynamic movement primitives generalization for deformable object manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5381–5388, 2022.

[5] M. Wu, B. Taetz, Y. He, G. Bleser, and S. Liu, "An adaptive learning and control framework based on dynamic movement primitives with application to human–robot handovers," *Robotics and Autonomous Systems*, vol. 148, p. 103935, 2022.

[6] S. Shaw, D. K. Jha, A. Raghunathan, R. Corcodel, D. Romeres, G. Konidaris, and D. Nikovski, "Constrained dynamic movement primitives for safe learning of motor skills," 2022.

[7] H. Kim, C. Oh, I. Jang, S. Park, H. Seo, and H. J. Kim, "Learning and generalizing cooperative manipulation skills using parametric dynamic movement primitives," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3968–3979, 2022.

[8] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, "ProDMP: A unified perspective on dynamic and probabilistic movement primitives," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2325–2332, 2023.

[9] M. Li, Z. Yang, F. Zha, X. Wang, P. Wang, W. Guo, D. Caldwell, and F. Chen, "Pattern analysis and parameters optimization of dynamic movement primitives for learning unknown trajectories," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8316–8322.

[10] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Wörgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[11] B. da Silva, G. Konidaris, and A. G. Barto, "Learning parameterized skills," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ser. ICML '12, J. Langford and J. Pineau, Eds. New York, NY, USA: Omnipress, July 2012, pp. 1679–1686.

[12] F. Stulp and G. Raiola, "dmpbbo: A versatile Python/C++ library for function approximation, dynamical movement primitives, and black-box optimization," *Journal of Open Source Software*, vol. 4, no. 37, p. 1225, 2019. https://github.com/stulp/dmpbbo.

[13] F. Stulp, I. Kresse, A. Maldonado, F. Ruiz, A. Fedrizzi, and M. Beetz, "Compact models of human reaching motions for robotic control in everyday manipulation tasks," in *Proceedings of the 8th International Conference on Development and Learning (ICDL)*, 2009.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, 2011.

[15] S. Calinon, *Springer Proceedings in Advanced Robotics*. Cham: Springer International Publishing, 2018, ch. Robot Learning with Task-Parameterized Generative Models, pp. 111–126.

[16] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proceedings of the 26th International Conference on Neural Information Processing Systems – Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, pp. 2616–2624.

[17] Y. Huang, L. Rozo, J. Silverio, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[18] 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*, 2011, pp. 365–371.

[19] M. J. A. Zeestraten, I. Havoutis, J. Silverio, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on riemannian manifolds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1240–1247, 2017.

[20] M. Saveriano, F. Franzel, and D. Lee, "Merging position and orientation motion primitives," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7041–7047.

[21] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical Movement Primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[22] M. Suzuki, Y. Yamazaki, N. Mizuno, and K. Matsunami, "Trajectory formation of the center-of-mass of the arm during reaching movements." *Neuroscience*, vol. 76, no. 2, pp. 597–610, 1997.

[23] F. J. Richards, "A flexible growth function for empirical use," *Journal of Experimental Botany*, vol. 10, no. 2, pp. 290–301, 06 1959.

[24] I. Iturrate, C. Sloth, A. Kramberger, H. G. Petersen, E. H. Østergaard, and T. R. Savarimuthu, "Towards reversible dynamic movement primitives," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5063–5070.

[25] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn. Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, September 2013.

[26] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[27] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient generalization of robot skills with contextual policy search," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. AAAI Press, 2013, pp. 1401–1407.