Path Integral Learning of Multidimensional Movement Trajectories

João André[*,†], Cristina Santos[*,†] and Lino Costa[**,†]

[*]*Departamento de Electrónica Industrial*
[†]*Universidade do Minho*
[**]*Departamento de Produção e Sistemas*

Abstract. This paper explores the use of Path Integral Methods, particularly several variants of the recent Path Integral Policy Improvement (PI^2) algorithm in multidimensional movement parametrized policy learning. We rely on Dynamic Movement Primitives (DMPs) to codify discrete and rhythmic trajectories, and apply the PI^2-CMA and PI^{BB} methods in the learning of optimal policy parameters, according to different cost functions that inherently encode movement objectives. Additionally we merge both of these variants and propose the PI^{BB}-CMA algorithm, comparing all of them with the *vanilla* version of PI^2. From the obtained results we conclude that PI^{BB}-CMA surpasses all other methods in terms of convergence speed and iterative final cost, which leads to an increased interest in its application to more complex robotic problems.

Keywords: Path Integral, Dynamic Movement Primitives, Parametrized Policies, Reinforcement Learning, Robotics, Black Box Optimization
PACS: 87.85.St

INTRODUCTION

The application of traditional *Reinforcement Learning (RL)* methods to continuous high-dimensional state spaces (more than 5-10 dimensions, e. g. humanoid robotics) remains problematic [1, 2, 3]. Despite the variety of RL algorithms available for relatively small state-spaces, where state discretization is possible [4], only recently have alternative methods been proposed that escape this *curse of dimensionality*, e. g. algorithms based on policy learning through stochastic trajectory sampling - although still associated with a high number of tuning parameters [1, 3].

A recent algorithm with already impressive and promising results was suggested in [3], derived from the framework of *Stochastic Optimal Control (SOC)* and its application to parametrized policies, known as *Path Integral Policy Improvement (PI^2)*. This approach is further reviewed and analysed in [5, 6], where the authors conduct a step-by-step comparison with similar RL methods, and, following two separate lines of work, implement significant changes, arriving at two distinct variants of the same algorithm: PI^2-CMA, which introduces iterative *Covariance Matrix Adaptation*, and PI^{BB}, inspired by *Black-Box Optimization (BBO)* methods.

Generically, RL problems can be described by the expression $\mathbf{u} = \pi(\mathbf{x}, t, \mathbf{w})$ [2] , where π represents a certain *control policy* that assigns a *motor command* $\mathbf{u}$ to a *state* $\mathbf{x}$ at *instant* t, dependent on the *parameter vector* $\mathbf{w}$. The learning goals is thus to find the optimal policy π^* that optimizes the performance of the agent/robot in a certain task - which directly entails finding the optimal parameter vector $\mathbf{w}^*$ [2]. When there is a relatively small number of states, the discretization of the state space is possible and π becomes a simple mapping between states and actions [4]. However, with large state spaces this is not feasible, which leads to π being often represented by dynamical systems.

A framework that proved to fit very well with this approach are *Dynamical Movement Primitives (DMPs)*, non-linear dynamical systems that make use of second order attractor dynamics to design complex parametrized trajectories, which makes them robust against external perturbations and easily modulated, adaptable to both discrete and rhythmic movements by means of both point and limit cycle attractors [2, 7]. By including a modular non-linear term in the form of a weighted sum of D Gaussian kernels, the DMP system becomes modifiable through a set of D inputs that express the weights of each Gaussian kernel.

Applying the generic formulation of RL problems to this concrete DMP structure, we will use several variations of the PI^2 algorithm to find the optimal weight vector (for now on referred to as θ) for the D Gaussian kernels that minimizes several cost functions. Because it is our belief than both PI^2-CMA and PI^{BB} present relevant evolutions to the initial algorithm, we additionally propose a new variant of PI^2, PI^{BB}-CMA, that merges both of these trends, keeping the BBO structure PI^{BB} and appending the CMA step of PI^2-CMA. Our purpose is to understand up to what

11th International Conference of Numerical Analysis and Applied Mathematics 2013
AIP Conf. Proc. 1558, 1025-1028 (2013); doi: 10.1063/1.4825679

point are both of these trends compatible with each other, and what advantages could the combined use of their features bring. We explore the learning process of both discrete and rhythmic multidimensional trajectories, particularly in the case of two and three degrees of freedom (DOF) systems. The results are then characterized in terms of convergence speed and final cost obtained, and compared to the PI^2 algorithm initially proposed by Theodorou et al. in [3].

PATH INTEGRAL POLICY IMPROVEMENT

As initially proposed, the PI^2 algorithm involves injecting stochastic Gaussian noise directly into the policy parameters on a instant-basis. This noise is sampled from a multivariate normal distribution $\mathcal{N}(\theta,\Sigma)$ where θ is the mean vector (the expected values) and Σ is the covariance matrix (dispersion between the samples). Considering a robotic task with fixed duration, K parameter vectors θ_k are generated which, through the use of DMPs, result in K different time-indexed trajectories $\tau_{k,i}$ (*rollouts*). While executing all the $\tau_{k,i}$, an instantaneous cost is assigned to each rollout, according to a cost function $C(\tau_{k,i})$, where the learning objective is indirectly expressed. Each rollout is then evaluated based on the *cost-to-go* $S_{k,i} = \sum_{j=i}^{N} C(\tau_{k,j})$ that yields the aggregated future cost at instant i, and a probability value $P_{k,i}$ value is calculated, that represents the *desirability* of the rollout (a form of *inverse cost probability-weighted averaging*, where high costs lead to lower probabilities and are thus less desirable). The mean vector θ is then iteratively updated according to the probability of each rollout - with lower cost rollouts having a larger contribution - until it converges to a solution, which can either be a global or local minimum.

By combining different exploration strategies and temporal weighting schemes, Stulp et al. [5, 6] arrived at two different forms of PI^2 algorithm: a BBO version of PI^2, PI$^{\mathrm{BB}}$, that involves constant exploration and an exponential decay of its magnitude [6], while relying only on the total cost of the rollouts to perform a parameter update, and PI^2-CMA [5], which also uses constant exploration but iteratively adapts the covariance matrix. Here we also implement CMA to PI$^{\mathrm{BB}}$, resulting in the PI$^{\mathrm{BB}}$-CMA algorithm, merging both variants of PI^2. The *vanilla* PI^2 version, on the other hand, adopts time-varying exploration and keeps the amount of exploration permanent during all iterations.

The use of DMP as the parametrized policy representation implies that the parameter vector $\theta \in \Re^{J*D\times 1}$ contains the D weights that characterize the trajectory shape, for each of the J DOFs of the movement, while the covariance matrix $\Sigma \in \Re^{J*D\times J*D}$ expresses the variance of the parameters, initially assumed to be independent ($\Sigma = \lambda_{init}\mathbf{I}$ on the first iteration, where λ_{init} is the initial exploration magnitude). The DMP internal parameters are chosen so that the dynamical system is critically damped, as suggested in [7], and are fixed for all the trials conducted. This leads to λ_{init} being the only open parameter of the algorithm, which presents one of the main advantages of PI^2 [3].

EVALUATION TASKS AND RESULTS

A crucial step in a proper implementation of the PI^2 algorithm is the correct design of the cost function $C(\tau_{k,i})$. In our case, we opted to create two distinct cost functions for discrete and rhythmic trajectories, that were ultimately dependent on the learning objective. In the case of 2D/3D discrete movements, the learning goal was to force the movement to pass through a viapoint at a specific time. For that purpose we assign a high cost at the relevant time step, proportional to the distance to the desired viapoint:

$$\begin{cases} C(\tau_{k,i}) = 0.5Q_e\ddot{q}_t^T\ddot{q}_t + 0.5R_e\theta^T\theta & when \quad t \neq t_v \\ C(\tau_{k,i}) = 0.5Q_e\ddot{q}_t^T\ddot{q}_t + 0.5R_e\theta^T\theta + P_{viapoint} & when \quad t = t_v \\ \quad P_{viapoint} = 10^{10}(q_t - q_{viapoint})^T(q_t - q_{viapoint}) \end{cases} \tag{1}$$

while trying to minimize both the norm of the parameter vector and movement acceleration in order to avoid extreme position changes (as used in [8]).

In the rhythmic case, our objective is to control the movement amplitude. For that we use a similar approach to the one used on discrete movement, but instead of creating time-indexed viapoints, we use phase-indexed viapoints spread in space according to the movement amplitude. Additionally, we also impose upper and lower limits to the movement position, to ensure that the viapoints represent the maximum and minimum of the oscillatory motion:

$$\begin{cases} C(\tau_{k,i}) = 0.5Q_e\ddot{q}_t^T\ddot{q}_t + 0.5R_e\theta^T\theta & when \quad q_{min} < q < q_{max} \; or \; \phi \neq 0 \; or \; \phi \neq \pi \\ C(\tau_{k,i}) = 0.5Q_e\ddot{q}_t^T\ddot{q}_t + 0.5R_e\theta^T\theta + P_{max} & when \quad q > q_{max} \; or \; \phi = 0 \\ C(\tau_{k,i}) = 0.5Q_e\ddot{q}_t^T\ddot{q}_t + 0.5R_e\theta^T\theta + P_{min} & when \quad q < q_{min} \; or \; \phi = \pi \\ \quad P_{max} = 10^{10}(q_t - q_{max})^T(q_t - q_{max}) & and \quad P_{min} = 10^{10}(q_t - q_{min})^T(q_t - q_{min}) \end{cases} \tag{2}$$

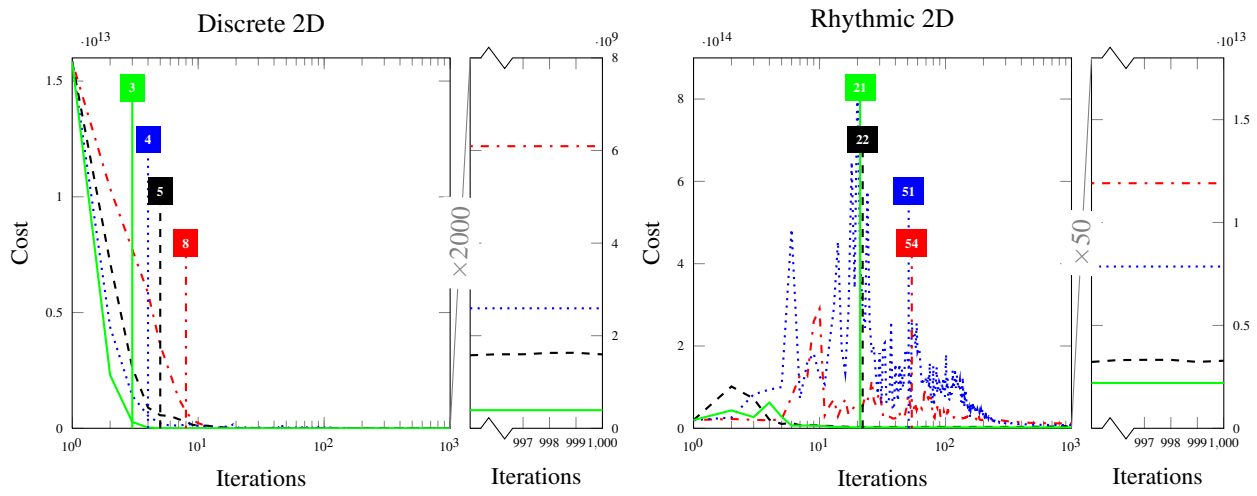

FIGURE 1. Cost evolution when learning 2D discrete and rhythmic trajectories (PI^2 (– · –);PI^2-CMA (—); PI^{BB} (· · ·); PI^{BB}-CMA (—)); Convergence speed is illustrated by the number of iterations necessary to verify convergence criteria;

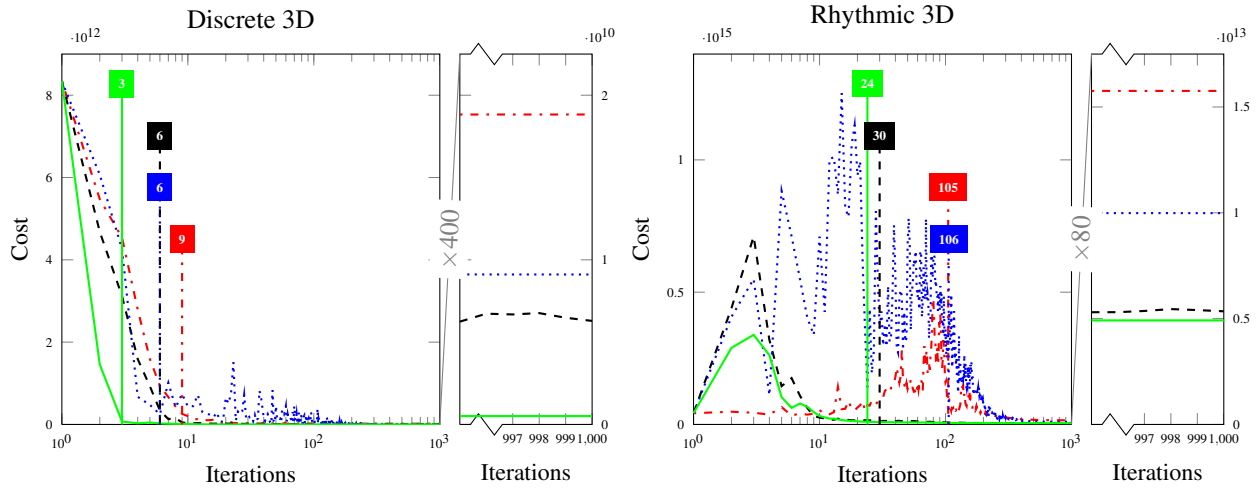

FIGURE 2. Cost evolution when learning 3D discrete and rhythmic trajectories (PI^2 (– · –);PI^2-CMA (—); PI^{BB} (· · ·); PI^{BB}-CMA (—)); Convergence speed is illustrated by the number of iterations necessary to verify convergence criteria;

In eq. 1, $q_{viapoint} \in \Re^{J\times 1}$ is the viapoint (at instant t_v), and in eq. 2 $q_{max} = g + 0.5A$ with $q_{max}, g, A \in \Re^{J\times 1}$ being, respectively, the upper limit for the movement position, the goal position (baseline of the oscillation) and the desired movement amplitude. Similarly, $q_{min} = g - 0.5A$, with $q_{min}, g, A \in \Re^{J\times 1}$. In both these expressions, $\ddot{q}_t \in \Re^{J\times 1}$ and $q_t \in \Re^{J\times 1}$ are respectively the acceleration and position at instant t. The dimension of all these terms is therefore dependent on the number of DOFs J in the problem (e.g. a 3D trajectory implies that $\ddot{q}_t^T = [\ddot{x}\ \ddot{y}\ \ddot{z}]$ and $q_t^T = [x\ y\ z]$).

The first and second terms of eqs. 1 and 2 illustrate the squared acceleration of $\tau_{k,i}$ and squared norm of θ, with factors Q_e and R_e in $C(\tau_{k,i})$ (as used in [8], with $Q_e = 1000$ and $R_e = 1$). The third and last term in both cost functions ($P_{viapoint}$, P_{max} and P_{min}) represent the forcing terms "punishing" undesirable trajectories. In all learning trials executed we used 10 DMP kernels ($D = 10$) and 100 rollouts per iteration ($K = 100$), with unitary temporal scaling ($\tau = 1$) and 100 time steps ($N = 100$), and movement duration of 1 second. The internal DMP parameters were kept constant during all experiments: in the discrete case we chose $\alpha_x = 25/3$, $\alpha_y = 25$ and $\beta = 25/4$; and in the rhythmic case $\omega = 6\pi, \alpha_y = 25/3$ and $\beta = 25/4$. On each iteration, one out of the K rollouts is *noiseless* (with mean θ as policy input parameters), and illustrates the iteration cost (value of $S_{i=1}$). We performed 1000 updates in all situations, and derive our conclusions from the cost evolution during these 1000 updates, as well as the cost of the final update.

The plot of the iteration cost over all the 1000 iterations performed are presented in Figures 1 and 2 for the 4 types of movement (discrete 2D/3D, rhythmic 2D/3D). The convergence criteria adopted was either a 95% or greater decrease from the initial value, or a less than 1% difference between consecutive iterations. We opted to extend the criteria used in [5, 6] due to the fact that rhythmic movements showed a cost increase in the early stages of learning.

Overall, learning was successful in all learning problems, with trajectories being able to properly adjust their path to travel through the viapoint (discrete movements), and keeping the amplitude constraints (rhythmic movements). Looking only at PI^{BB} and PI^2-CMA, we see that the latter achieves a lower cost in all cases, despite showing slower convergence speed in discrete problems. Moreover, the PI^{BB} variant showed particularly poor results in rhythmic movement learning, which can be explained by the lack of specificity of the cost function (eq. 2 only imposes maximum and minimum values, regardless of movement shape), suggested by the large cost oscillations visible in the results.

PI^{BB}-CMA and PI^2-CMA, on the other hand, exhibit not only a very fast convergence, but also a lower cost in all cases, with the gap to both PI^{BB} and *vanilla* PI^2 increasing significantly in rhythmic movements. The CMA step, that drastically reduces exploration magnitude when an optimal solution is found, is thus directly related to a higher convergence speed. In addition, the proposed PI^{BB}-CMA algorithm outperforms all of the other variants of the PI^2 method in all the learning trials conducted, in terms of both convergence speed and final cost, which offers proof of the potential advantages of merging the two earlier variants of PI^2.

CONCLUSION AND FUTURE WORK

In this work, we asserted the usefulness of two recent variants of the PI^2 algorithm in the design of multidimensional movements, and proposed a novel hybrid form that combines features from both algorithms: the PI^{BB}-CMA algorithm, which, as the results demonstrate, exhibits both faster convergence and higher quality solutions. This shows that PI^2-CMA and PI^{BB} are not in any way mutually exclusive, but rather convergent algorithms that improve path integral learning through parallel lines of work. Furthermore, the tasks used to benchmark PI^2 methods are, to our knowledge, usually based only on simple discrete movements. Here we try to extend this approach to their rhythmic counterparts, providing a structure to control motion amplitude, and using both movement types as basis for learning evaluation. The promising results obtained encourage the application of Path Integral approaches to more complex robotic problems. We are particularly interested in their application towards biped locomotion - the possibility of using *Central Pattern Generators (CPGs)* as a starting point of the learning process, and recurring to PI^2 variants for additional tuning and self-improvement of a locomotion system looks a very attractive solution, and should be further pursued.

ACKNOWLEDGMENTS

This work is funded by FEDER Funding supported by the Operational Program Competitive Factors - COMPETE and National Funding supported by the FCT - Portuguese Science Foundation through scholarship UMINHO/BI/40/2012 inserted in project PTDC/EEACRO/100655/2008 and also project FCOMP-01-0124-FEDER-022674.

REFERENCES

1. E. Theodorou, J. Buchli, and S. Schaal, "Learning Policy Improvements with Path Integrals," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 828–835.
2. S. Schaal, P. Mohajerian, and A. Ijspeert, *Progress in Brain Research* **165**, 425–45 (2007), ISSN 0079-6123.
3. E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach," in *2010 IEEE International Conference on Robotics and Automation*, 3, IEEE, 2010, pp. 2397–2403, ISBN 978-1-4244-5038-1.
4. R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.
5. F. Stulp, and O. Sigaud, "Path Integral Policy Improvement with Covariance Matrix Adaptation," in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, 2012.
6. F. Stulp, and O. Sigaud, "Policy Improvement Methods: Between Black-Box Optimization and Episodic Reinforcement Learning," in *Journée Francophones de Planification, Decision et Apprentissage*, 2013.
7. A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, *Neural Computation* **25**, 328–73 (2013), ISSN 1530-888X.
8. S. Schaal, E. Theodorou, J. Buchli, and F. Stulp, "An Example Application of Policy Improvement with Path Integrals (PI²)," 2010, vol. 1.