AN EVOLUTIONARY MODEL-FREE CONTROLLER AND ITS APPLICATION TO THE SWING-UP OF A DOUBLE INVERTED PENDULUM

A THESIS

Presented to the Department of Electrical Engineering

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

Committee Members:

Fumio Hamano, Ph.D. (Chair)
Hen-Geul Yeh, Ph.D.
I-Hung Khoo, Ph.D.

College Designee:

Antonella Sciortino, Ph.D.

By Venkata Dhruva Pamulaparthy

B.E., Rashtrasant Tukadoji Maharaj Nagpur University, India

December 2017

ABSTRACT

AN EVOLUTIONARY MODEL-FREE CONTROLLER AND ITS APPLICATION TO THE SWING-UP OF A DOUBLE INVERTED PENDULUM

By

Venkata Dhruva Pamulaparthy

December 2017

Advancements in the field of machine learning has made a model-free approach for nonlinear control of dynamical systems more viable. Traditionally, the controller design is based on the analysis of the system model. In practice, however, it might not be possible to estimate a system model that truly reflects the complex behavior of the real system. A model-free controller self-learns the required control decisions by applying machine learning techniques, avoiding the need for estimation and analytical design. In this thesis, a parameterized dynamical system known as Dynamic Movement Primitive (DMP) is used as a feedforward model-free controller. An advanced, nature-inspired, evolutionary machine learning algorithm called Covariance Matrix Adaption Evolution Strategy (CMA-ES) was used to self-learn the control decisions. It was demonstrated through computer simulated experiments that such an evolutionary model-free controller could successfully learn to accomplish the difficult task of swinging up a double inverted pendulum, motivating further research.

ACKNOWLEDGEMENTS

Firstly, I must thank my advisor Dr. Fumio Hamano for his immense patience and enthusiasm during our lengthy discussions over the many phases of this thesis. His valuable guidance and coordination made this thesis possible.

I should also thank Dr. Henry Yeh and Dr. I-Hung Khoo who agreed to read my thesis and to be a part of my defense committee at a very short notice.

I would be remiss if I didn’t thank my friends and roommates who made my time at CSULB very pleasant.

Next, I should thank my sister, who helped me edit this document to its present form.

Above all, I must thank my parents who have been a source of infinite emotional strength, encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The application of machine learning techniques to the control of dynamical systems has drawn attention among researchers in recent years [1] - [3]. Such techniques are characterized as "model-free," due to their independence from the use of system models. The rationale behind the application of model-free techniques is to avoid model estimation errors that occur in analytical control.

The focus of this thesis is to develop an evolutionary model-free controller – a controller that learns to drive the system to the desired output using a class of machine learning algorithms known as evolutionary algorithms. Parameterized artificial dynamic systems known as Dynamic Movement Primitives (DMPs), invented by Ijspeert and Schaal [4], provide an adaptable structure that is suitable for the application of evolutionary algorithms to learn the desired control. This thesis studies the application of a DMP as an open-loop, feedforward, and model-free controller, which uses an evolutionary algorithm and eventually learns to take the required control decisions.

Evolutionary algorithms, which have gained popularity due to their ability to effectively solve black-box optimization problems, are machine learning techniques that are inspired from the process of natural evolution of biological species. Such techniques have been recently used to develop model-free controllers. In this thesis, the Covariance Matrix Adaption Evolution Strategy (CMA-ES), by Hansen and Ostermeier [5], is used to show evolutionary learning in a DMP-based model-free controller.

In this chapter, we introduce the goals of this thesis and the experiments performed to realize them.

Research Goals and Problem Statement

The primary goal of this thesis is to demonstrate that a DMP-based evolutionary model-free controller can eventually learn to control the dynamics of a highly nonlinear system using CMA-ES. The swing-up of a double inverted pendulum from its pendant position to its upright position is the specific problem chosen in this study.

The double inverted pendulum consists of two pendulums with a common pivot point (or cart). A single actuator controls the dynamics of the pivot point. The pendulums are controlled due to coupling of the different degrees of freedom. Since the number of actuators is less than the degrees of freedom, the system is underactuated. This makes controller design by analytical methods difficult, as inherent nonlinearities in the system are not easily removed through full feedback linearization [6]. So, model-free methods become necessary.

To accomplish evolutionary model-free control for the task described above, the evaluation of an appropriate objective function is required to measure the effectiveness of the controller. This thesis formulates and studies energy-based objective functions for this purpose.

Computer simulations of the experiments in this thesis were performed using MATLAB modeling software.

Organization of Chapters

The following section describes the organization of this thesis. Chapter 2 provides a survey of literature available on model-free methods for control. Chapter 3 describes the structure of a Dynamic Movement Primitive (DMP). Chapter 4 focuses on the use of evolutionary algorithms for control, covering the Covariance Matrix Adaption Evolution Strategy (CMA-ES) used in this thesis in detail. Chapter 5 presents the model of the double inverted pendulum used for experimentation. Chapter 6 describes the methodology used to develop swing-up control of a double inverted pendulum. Chapter 7 presents the simulation results obtained. Chapter 8 analyzes the results obtained. Chapter 9 concludes the thesis, suggests future applications, and speculates on possible future research.

Mathematical Notations

In this thesis, all non-bold characters within the equations represent scalar quantities that belong to the real space, unless specified otherwise. All vectors are represented by boldface characters and belong to an n-dimensional real space, where $n > 1$. The vector declaration is compactly represented as $\boldsymbol{x} \in \mathbb{R}^n$. All matrices are represented by uppercase and boldface characters. The matrices belong to an $n \times m$-dimensional real matrix space represented by $\mathbb{R}^{n \times m}$, where $n > 1$ and $m > 1$. In this definition n and m represent the number of rows and columns of the matrix respectively. The matrices are declared compactly as $\boldsymbol{C} \in \mathbb{R}^{n \times m}$. When the matrix has equal number of rows and columns, i.e., $n = m$, it is known as a square matrix.

The variables of the form $\dot{x}$ denote the differentiation of x with respect to time. Similarly, $\ddot{x}$ denotes that x has been differentiated twice with respect to time. The variable t has been reserved to denote time and must be read as such whenever it appears in this thesis.

CHAPTER 2

LITERATURE REVIEW

Modern control theory provides various analytical methods for control. These are discussed in texts such as Slotine and Li [7], Spong, Hutchinson, and Vidyasagar [8], and Lewis, Jagannathan, and Yesidirek [9]. As seen in these texts, analytical methods require the description of a system model. Such a description is not always available in practice, therefore necessitating research into model-free methods.

In model-free methods, the controller learns to make decisions by processing the data obtained from the system, implying the use of machine learning methods. In this thesis, we are interested in the self-learning of the controller without any prior information of the desired solution. This problem is traditionally solved by designing environmental reinforcements that encourage good controller action. The methods that use this process to develop a controller are called reinforcement learning algorithms. The research in this branch of machine learning in the context of model-free control is described in the subsequent paragraphs.

The value-based approaches presented by Sutton and Barto [10] and Doya [11] describe the early attempts to introduce model-free control with reinforcement learning. These involve the tedious construction of a map of data in the form of environmental reinforcements on the state space of the system dynamics through repeated exploration.

Newer methods based on direct policy search offer a solution to reinforcement learning-based model-free control in high dimensional and continuous spaces. These involve the description of an adaptable controller, called a policy generator in the existing machine learning literature. In this thesis, we focus on dynamical system-based policy generators, called Dynamic Movement Primitives, given by Ijspeert and Schaal [4]. In direct policy search, exploration and learning takes place in the parameter space of the controller (or policy generator). These methods are broadly classified as gradient-based and gradient-free.

Gradient-based methods involve estimation of a gradient for the descent to the optimum in parameter space. Peters and Schaal [12] give a good survey and description of gradient-based methods. The most important of them, commonly used in robotic trajectory planning, is the episodic Natural Actor Critic (eNAC) by Peters and Schaal [13], which uses natural gradient estimates described by Amari [14].

Alternatively, gradient-free methods are completely based on system statistics. Koeber and Peters [15] invented an algorithm called Policy Learning by Weighted Exploration Returns (PoWER) that uses state dependent exploration and Expectation Maximization (EM) updates to learn the required optimal controller. The Policy Improvement with Path Integrals algorithm by Theodorou, Buchli, and Schaal [16], based on the path integral formulation of stochastic optimal control, has achieved significant results in robotic task learning in high dimensions.

Gradient-free methods point to the application of another class of machine learning algorithms, called evolutionary algorithms, to the problem of model-free control. The distinction between evolutionary learning and direct policy search reinforcement is vague. In general, evolutionary learning involves the evolution of the structure of the controller such that it can make optimal decisions given the effect of all reinforcements available in a trial [17].

Next, the research on evolutionary algorithms along with their application to control problems similar to the one chosen in this thesis is tracked.

Evolutionary algorithms, based on the theory of evolution by Darwin [18], have shown promise in solving black-box optimization problems. Evolutionary algorithms begin with the work of Holland [19], who observed that concepts inspired from natural selection of biological species could be used for eventually finding optimal solutions. Most evolutionary algorithms that were subsequently developed are based on Holland's insight. However, there also exist other evolutionary algorithms, which are not entirely based on Holland's insight [20], [21].

The evolutionary algorithm used in this thesis, known as the Covariance Matrix Adaption Evolution Strategy by Hansen and Ostermeier [5], gives a structured procedure to solve black-box optimization problems by combining the principles of natural evolution as envisaged by Holland with the properties of normal distributions.

Whitley et al. [22] used the genetic algorithms to evolve a controller, based on a neural-network, for the balance of an inverted pendulum. Stulp and Sigaud [23] compared different covariance matrix-based evolutionary strategies for the trajectory planning of robotic manipulators. Heidrich-Meisnerm and Igel [24] evolved a controller, based on a neural-network, with CMA-ES to balance an inverted pendulum. Encouraged by the above research, CMA-ES has been used to evolve a controller for the swing-up of a double inverted pendulum in this thesis.

CHAPTER 3

DYNAMIC MOVEMENT PRIMITIVES AS MODEL-FREE CONTROLLERS

Dynamic Movement Primitives (DMPs) are controllers (or policy generators) that can learn to produce desired optimal movements in the physical system to be controlled. DMPs are artificial nonlinear systems with a parameterized component that allows for learning. The trajectories generated by the DMP could either be used to provide the reference dynamics for an analytically designed controller, or be directly applied as open-loop, feedforward control inputs.

As reference trajectory generators, DMPs have been especially used in skill learning tasks that require mimicking human motor movement [25], [26].

This thesis uses the output of DMP system for open-loop, feedforward control. In this format, the DMP system itself is the controller to the system. In other words, the forces produced by the artificial DMP system are used to drive the physical system towards the goal state.

Formalization of DMPs

In this section, the structure of manipulatable systems that are categorized as DMPs is studied. DMP systems are usually classified into two types based on their behavior – Discrete or Rhythmic. Discrete DMPs exhibit transient trajectories that converge to a stable equilibrium point while Rhythmic DMPs generate an oscillating response. Based on the required application, the DMP system with the appropriate behavior is selected. In this section, we define a Discrete DMP that generates transient forces to accomplish the swing up control of a double inverted pendulum.

In the format suggested by Ijspeert and Schaal [4], the forced mass-damper-spring system shown in Figure 1 is used as the Dynamic Movement Primitive. The dynamics of the DMP is given by the Equations (1) – (6) below.

$$\ddot{x} = \left(\omega_0^2(x_g - x) - 2\zeta\omega_0\dot{x}\right) + w^T\varphi(z) \tag{1}$$

$$\dot{z} = -\gamma z \tag{2}$$

$$\varphi_i(z) = \frac{\psi_i(z)}{\sum_{i=1}^{N}\psi_i(z)} z\left(x_g - x_0\right) \tag{3}$$

$$\psi_i(z) = \exp\left(-\frac{1}{2\sigma_i^2}(z - c_i)^2\right);\ i = 1,2 \dots, N \tag{4}$$

In Equation (1), the state variables $x, \dot{x},$ respectively represent the position and velocity of the DMP system, while $\ddot{x}$ represents acceleration output. The parameters of the DMP system are the mass m, the spring constant k, and the damping coefficient c. In the Equations (1) and (3), the term x_g represents the natural length which corresponds to the relaxed position of the spring in the DMP system. In Equation (3), the term x_0 is the initial deviation from relaxed position of the spring in the DMP system. The terms of the form given in Equation (4) are N radial basis functions with centers c_i and widths given by $\frac{1}{2\sigma_i^2}$. The term,

$$\omega_0 = \sqrt{\frac{k}{m}} \tag{5}$$

is the natural frequency of the mass-spring system of the DMP, and the damping ratio is,

$$\zeta = \frac{c}{\sqrt{2mk}} \tag{6}$$

where, c is the damping coefficient. Equation (2) shows the dynamics of an internal state variable z and the term γ is a positive time constant that controls its decay. Each component of the system described by Equations (1) – (6) is analyzed in detail in the next section. The terminology used for the components was given by Ijspeert and Schaal [4].

Components of a DMP

A Dynamic Movement Primitive consists of three components as shown below in Figure 2.

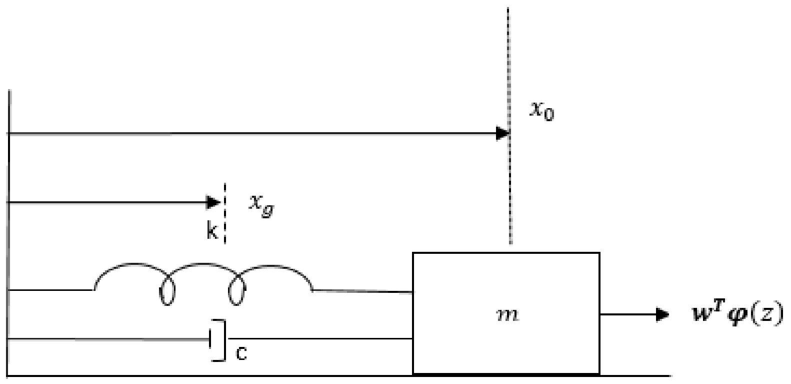

FIGURE 1. Mass-damper-spring system with external force used as DMP.

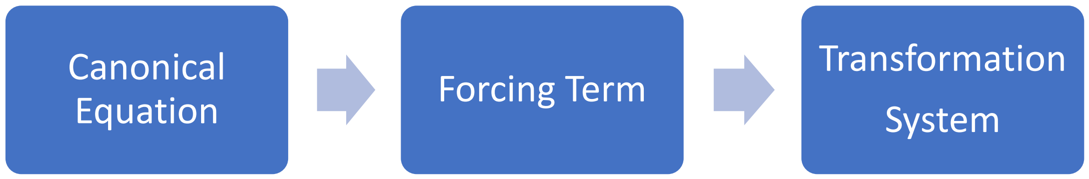

FIGURE 2. Components of DMP.

Transformation System

Equation (1) is called the transformation system of the DMP. It consists of a damped mass-spring system given by,

$$\ddot{x} = \left(\omega_0^2(x_g - x) - 2\zeta\omega_0\dot{x}\right) \tag{7}$$

with an adjustable nonlinear component called the forcing term given by,

$$f(z) = \boldsymbol{w}^T\boldsymbol{\varphi}(z) = \frac{\sum_{i=1}^{N}\psi_i(z)w_i}{\sum_{i=1}^{N}\psi_i(z)} z\left(x_g - x_0\right) \tag{8}$$

where, ψ_i represents the i^{th} basis function parameterized with weight w_i. The mass-damper-spring system is assumed to be critically damped, i.e., $\zeta = 1$.

When $f(z) = 0$, the Equation (1) becomes a critically damped mass-spring system, shown in Equation (7), with a unique stable point attractor at $x = x_g$ (as shown in proof 1). The behavior of this system is shown in Figure 3. The effect of including $f(z)$ is discussed next.

Forcing Term

The forcing term $f(z)$ is the nonlinear function approximator shown in Equation (8). It forms a dynamic movement forms a dynamic neural-network with a vector of normalized basis functions $\boldsymbol{\varphi}(z)$ of the internal state variable z and adjustable weight vector $\boldsymbol{w}$. Each element of the basis vector $\boldsymbol{\varphi}(z)$ is the normalized form of the radial basis function $\psi_i(z)$. Each radial basis function $\psi_i(z)$ is given in Equation (4) and its normalized form is given in Equation (3). The dynamics of internal state variable z, shown in Equation (2), controls the activation or firing of the basis. Figure 4 shows the sequential firing of the basis functions $\psi_i(z)$in Equation (4) with respect to time t. Nine basis functions were used.

To summarize, the forcing term is a linear combination of a fixed basis function with adjustable weights. In the context of Equations (1) – (6), the forcing term represents the time-varying external force on the mass-damper-spring system. It is responsible for shaping the dynamics of the DMP system by generating the driving fields. Structural evolution of the DMP occurs by adjusting the parameters of the forcing term.

The effect of the forcing term is shown in Figure 5. The forcing term influences the attractor landscape of the damping dynamics of Equation (7). The damping depends on weight parameter vector $\boldsymbol{w}$ in Equation (8). In Figure 5, the effect of 20 different parameter sets of weight vector $\boldsymbol{w}$ on the damping dynamics of Equation (7) are shown with respect to time t. This influence was interpreted as the shaping of the attracter landscape of Equation (7) by Ijspeert and Schaal [4].

Canonical System

The Equation (2) is known as the canonical system. As described in Equations (1) – (6), the canonical system implicitly controls the dynamics of the transformation system through an internal state variable z. The states of the canonical system are used to form the basis of the dynamic neural-network in the transformation system, whose weights are given by the parameter vector. The nature of the canonical system dictates the behavior of the output of the DMP. It could be a simple decaying system, as used in this thesis, which ends at a stable equilibrium and produces transient trajectories, or a limit cycle with a specific periodic orbit that produces rhythmic (or repetitive) movement.

The canonical equation can also be thought of as a timing signal. It allows for coupling of different degrees of freedom or even sequencing of multiple motions. In the transient DMP structure chosen in Equations (1) – (6), the decay of the canonical system controls the time-varying forces on the damped mass-spring system in Equation (7). Figure 6 shows the canonical system used in this thesis as described by Equation (2).

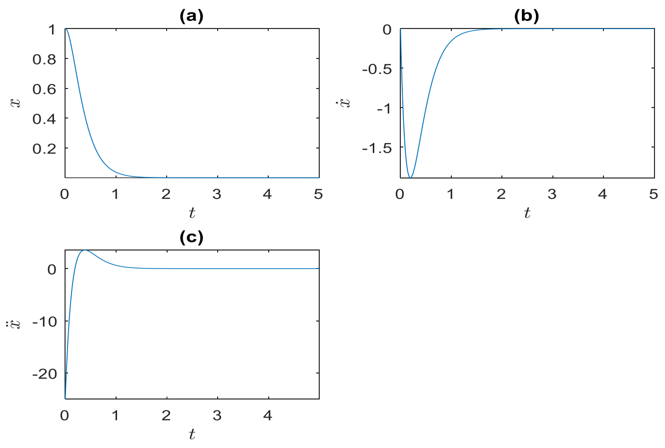

FIGURE 3. System behavior of DMP for $f(z) = 0$ and $\zeta = 1$ and $\omega_0 = 5$. The behavior is that of a critically damped mass-spring system. Plot (a) shows the dynamics of DMP position. Plot (b) shows the dynamics of DMP velocity. Plot (c) shows DMP acceleration. All the dynamics are plotted with respect to time t.

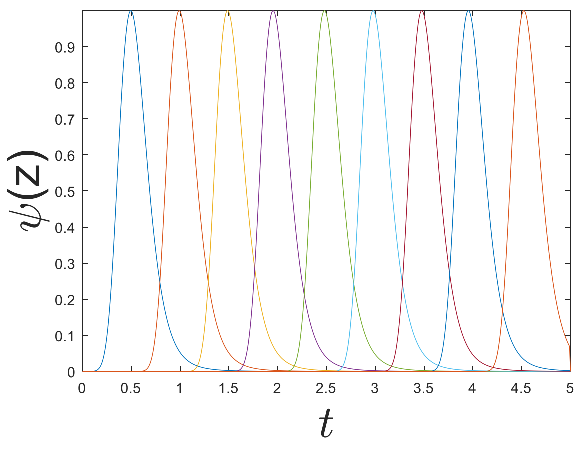

FIGURE 4. Basis functions. The basis functions are plotted with respect to time *t*.

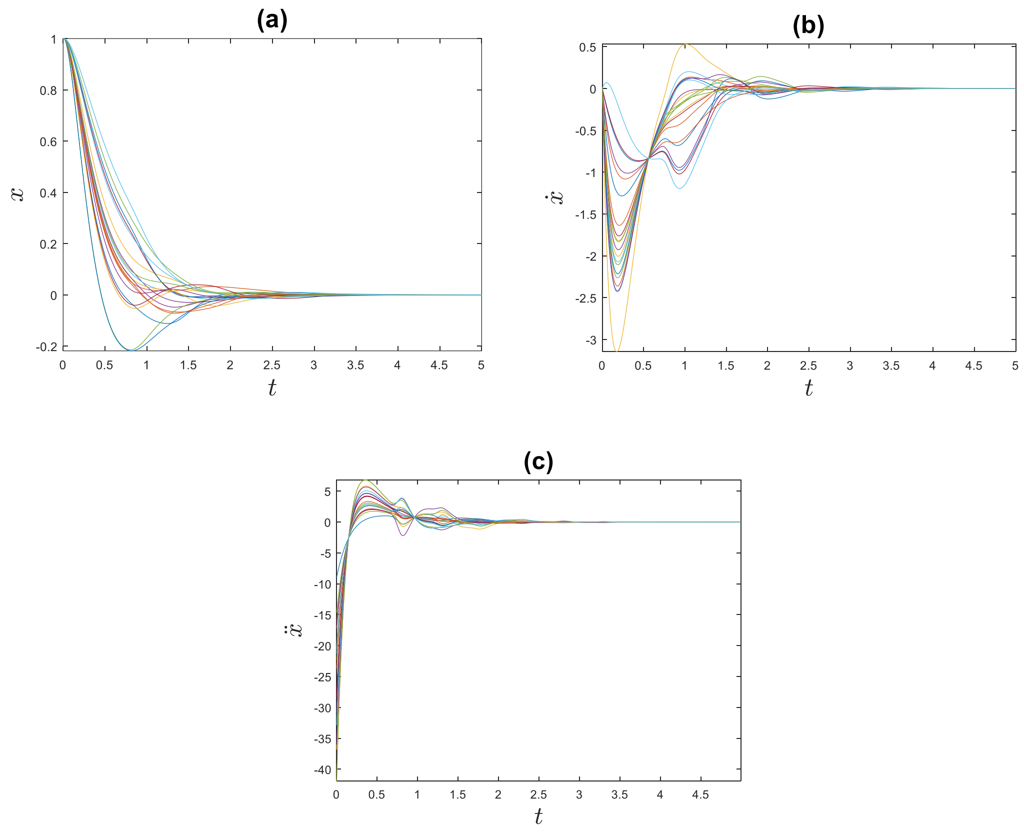

FIGURE 5. Effect of forcing term. Plot (a) shows twenty DMP position trajectories with different parameter vectors *w*, each containing nine elements chosen at random. Plot (b) shows twenty DMP velocity trajectories with different parameter vectors *w*, each containing nine elements chosen at random. Plot (c) shows twenty DMP acceleration trajectories with different parameter vectors *w*, each containing nine elements chosen at random. All the trajectories are plotted with respect to time *t*.

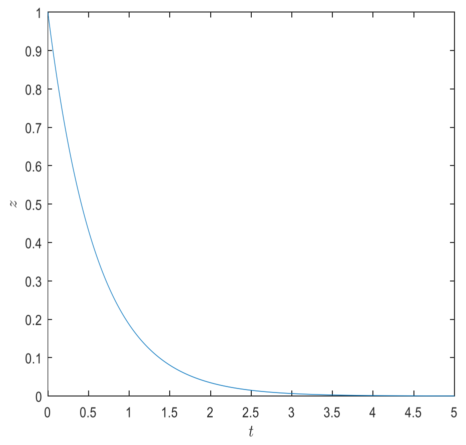

FIGURE 6. Canonical system for discrete movements.

DMP as a Trajectory Generator: Integrating DMP Reference Trajectories with Lower Level Architecture

Integrating the output of DMPs with lower level controller is quite straightforward and is suggested by Ijspeert and Schaal [4]. The output of the DMP is applied as reference input to the trajectory controllers as shown in Figure 7. The states of reference systems are to be tracked by the physical system (or plant). The states of the physical system are $\boldsymbol{q}, \dot{\boldsymbol{q}}$. The states of the desired trajectory generated by the DMP system are $\boldsymbol{q_d}, \dot{\boldsymbol{q}}_{\boldsymbol{d}}$ with feedforward term $\ddot{\boldsymbol{q}}_{\boldsymbol{d}}$**.** The term R is the reinforcement signal from the environment which causes change in parameters of the DMP. If evolutionary algorithms are used, the reinforcement signal is replaced by a fitness value obtained by the evaluation of an objective function. The scheme is usually used offline for evolution algorithms.

Standard controllers that use reference input and feedforward terms such as inverse dynamics controllers, adaptive controllers, and sliding mode controllers, could be used as the lower level controllers that track the reference trajectory provided by the DMP [7] - [9]. Some examples of applications of DMPs as reference trajectory generators in combination with reinforcement learning are given by Peters and Schaal [13], Koeber and Peters [15], and Theodorou, Buchli and Schaal [16].

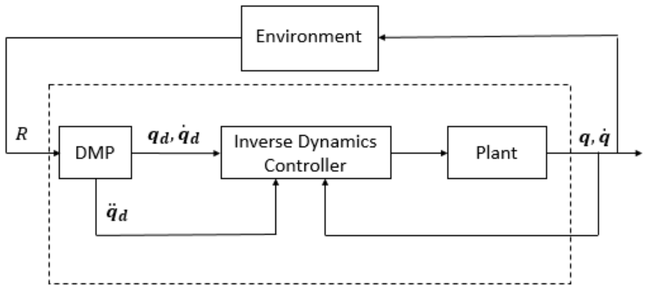

FIGURE 7. An architecture for the application of DMP to low-level controller.

DMP as an Open-Loop Controller

This thesis uses DMP as an open-loop controller. The field generated by an external artificial parameterized system described by the DMP is fed to the control input u in open loop. By choosing the appropriate DMP, we can program the control forces on the physical system to drive it to the desired state. For evolutionary algorithms, the reinforcement R from the environment is replaced with the offline measurement of the fitness from an objective function. The procedure is applied to evolve a controller for the swing-up of a double inverted pendulum in Chapter 6. The open-loop control is represented in Figure 8. The properties of the feedforward DMP controller are discussed next.

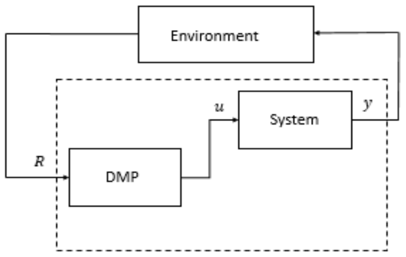

FIGURE 8. DMP as an open-loop controller. u is the control input from DMP and y is the output.

DMP Stability

The following shows the stability analysis of the DMP system given by Equations (1) - (6).

Claim 1: The DMP system given by Equations (1) - (6) is asymptotically stable for any bounded parameter vector $\boldsymbol{w}$.

Proof 1: Consider the system in Equation (1) without the term in f(z) given in Equation (6). The system is a damped mass-spring system without forcing given by Equation (7). This system is dissipative and convergent. It is quite simple to prove that this system is stable and is covered in detail in Slotine and Li [7]. The proof is presented here.

Consider the following positive definite Lyapunov function,

$$V = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}k(x - x_g)^2 \tag{9}$$

Differentiating Equation (9) with respect to time gives,

$$\dot{V} = k(x - x_g)\dot{x} + m\dot{x}\ddot{x} \tag{10}$$

or

$$\dot{V} = k(x - x_g)\dot{x} + m\dot{x}\ (\left(\omega_0^2(x_g - x) - 2\zeta\omega_0\dot{x}\right))$$

$$= m\dot{x}(-\omega_0^2\left(x_g - x\right) + \omega_0^2(x_g - x) - 2\zeta\omega_0\dot{x})$$

$$\dot{V} = -2\zeta\omega_0 m\dot{x}^2 \tag{11}$$

Since, $\zeta > 0$ and $\omega_0 > 0$, $\dot{V}$is negative definite, thus the system is asymptotically stable. At $\zeta = 1$, the mass-spring system is critically damped.

Now, considering the entire system in Equations (1) – (6), it is clear that the system in Equation (2) is convergent due to the solution given by Equation (12) below.

$$z = \exp(-\gamma t)\ ,\gamma > 0 \tag{12}$$

Since z decays asymptotically according to time constant γ, the expression describing $f(z)$ in Equation (8) also decays asymptotically for bounded weights of the form w_i, due to modulation with z. After $f(z)$ decays sufficiently close to zero, the system behaves like a critically damped mass-spring system that was proven to be stable by Lyapunov stability analysis.

Invariance

From the procedure for invariant set analysis provided in Slotine and Li [7], we find that the point $(x, \dot{x}, z) = (x_g, 0{,}0)$ is the only stable equilibrium point of the system.

Claim 2: The point is $(x, \dot{x}, z) = (x_g, 0{,}0)$ is the only global invariant set of the DMP given by Equations (1) – (6), implying that it is an asymptotically stable equilibrium point.

Claim 3: Every trajectory of the dynamics of the DMP system in Equations (1) – (6) converges to the point $(x, \dot{x}, z) = (x_g, 0{,}0)$ for any choice of bounded values of weight vector $\boldsymbol{w}.$

Proof 2: To complete this proof, we again focus on the unforced mass-damper-spring system in Equation (7). Consider the time-differentiation of the Lyapunov function $\dot{V}$ in Equation (11), if $\dot{V} = 0$, it implies that,

$$\dot{x} = 0 \tag{13}$$

also implying from Equation (7) that,

$$\ddot{x} = \omega_0^2(x_g - x) \tag{14}$$

Equation (14) is zero only at $x = x_g$. However, Equation (11) shows that the system eventually reaches stability. Also, V is radially unbounded if x is radially unbounded. Thus, by global invariance theorem (see Slotine and Li [7]), since the point $(x, \dot{x}) = (x_g, 0)$ is the only element in the invariant set, it is a global asymptotically stable equilibrium point.

Also, trivially, from Equation (12) the term $f(z)$ in Equation (8) dies out, thus converging to $z = 0$.

Thus, the equilibrium point $(x, \dot{x}, z) = (x_g, 0,0)$ is the globally asymptotically stable point of the DMP.

Thus, every trajectory caused by the DMP system in Equations (1) – (6) due to a bounded choice of a parameter vector $\boldsymbol{w}$, eventually converges to the above invariant set. Thus, the proof is complete.

Another interpretation of the above proofs is that forcing term $f(z)$ given by Equation (1) controls the rate of dissipation of the mass-damper system in Equation (7). Since $f(z)$ itself is dissipating due to dissipative system in Equation (2), we can be sure that the entire DMP is stable and converges to invariance set.

Smoothness

The smoothness of the DMP is guaranteed due to the smooth convergence of the damped mass-spring system in Equation (7) and the smooth asymptotic decay of Equation (2). Also, from another viewpoint, since the entire system is continuously dissipative, we have a smooth convergence to invariance set. Hence, the open-loop controller described by a DMP is smooth.

Other Useful Properties of DMPs

DMPs have other useful properties that could be utilized especially in trajectory design. Although they are not focused in this thesis, they are briefly mentioned here.

- The choice of the structure of the DMP is flexible. DMPs could be Discrete (an individual maneuver) or Rhythmic (periodic pattern generators) or a combination of both [4].

- Synchronizing and joining of separate DMPs is possible using appropriate coupling terms [4], [27]. Addition of coupling terms can also make the system reactive to environmental changes. Coupling terms have been successfully used to integrate online obstacle avoidance with movement planning [28].
- It has been shown that DMPs allow temporal scaling of trajectories [4]. Scaling implies preservation of the shape of the original DMP even when the size of the overall trajectory is changed. DMPs can easily be scaled by multiplying the dynamics with a scaling constant.

CHAPTER 4

EVOLUTIONARY ALGORITHM-BASED CONTROL USING CMA-ES

This chapter describes the general framework of an evolutionary algorithm and its application to model-free control. The chapter also focuses on the Covariance Matrix Adaption Evolution Strategy (CMA-ES) by Hansen and Ostermeier [5], and shows its application to the evolutionary learning of a controller.

Evolutionary Algorithms: An Overview

In evolutionary algorithms, the solution to the given problem is computed by a process inspired by natural evolution of biological species. First, an initial population of solution candidates is generated at random. Then, the population is gradually improved using evolutionary operations to eventually find the best solution to the given problem. An objective function is defined and evaluated to describe the performance of an individual. The optimum of the objective function is obtained when it is evaluated with the best solution found by using evolutionary algorithm. Thus, the best solution is called the optimal solution. The proximity of an individual of the population to the best solution is described by a value known as “fitness,” which is a performance index based on the evaluation of the objective function with that individual.

The fundamental operations in evolutionary algorithms are selection and reproduction (see Hansen and Ostermeier [5], Holland [19], and Kennedy and Eberhart [21]). Selection is an operation that depends on the fitness value. The individuals of the population closer to the best solution, i.e., with better fitness are likely to be selected. The other important operation in evolution algorithms is reproduction. It generates exploration in the objective function space by the creation of new offspring population. A combination of the two operations will gradually drive the populations to the required optimal solution. The evolution algorithms vary in the way the two operations occur.

No knowledge of the physical system or its environment is required for evolution; only the fitness term needs to be computed. Thus, evolutionary algorithms are entirely model-free. Evolutionary algorithms are most useful when the relationship between the solution of the objective function and the input parameters is too complex to be modeled.

Covariance Matrix Adaption Evolutionary Strategy (CMA-ES)

CMA-ES can be effectively used for the optimization of ill-poised black-box functions which are non-linear and non-convex. Thus, CMA-ES is a powerful algorithm that can be almost universally applied and is very extensively used for difficult optimization problems. Here, we describe every step of the algorithm closely following the analysis given by Hansen and Ostermier [5]. For more details on convergence properties of the algorithm refer to Hansen and Ostermier [29].

Generating Population Samples

Population samples are generated by the mutation of the parent parameter vector of n elements, given by the term, $\boldsymbol{w}^g \in \mathbb{R}^n$.The mutation process is equivalent to drawing samples from a multivariate normal distribution with mean equal to the parent and a covariance $\boldsymbol{C}^{\boldsymbol{g}} \in \mathbb{R}^{n\times n}$. The samples that form the initial population are the offspring of the initially chosen parent. The k^{th} offspring is given by,

$$\boldsymbol{x}_k^{g+1} \sim \boldsymbol{\mathcal{N}}\left(\boldsymbol{w}_g, \mathbf{C}^g\right) \sim \boldsymbol{w}_{\boldsymbol{g}} + \sigma_g^2 \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}^g) \tag{15}$$

where, the symbol $\sim$ indicates distribution equivalence, $\boldsymbol{x}_k^{g+1} \in \mathbb{R}^n$ is the k^{th}offspring at generation $g+1$, and σ_g is the step size.

Finding Fitness Value

The fitness value of each of the offspring is obtained by evaluating the objective function. It represents the quality of the solution.

Reproduction: Shifting of the Mean

Evolution in CMA-ES happens due to the shifting of the mean to form a new multivariate distribution. To find this new mean, the samples drawn from the current distribution are recombined through weighted averaging after selection. The weights describe the influence of an offspring in generating the new mean. From the new distribution with the updated mean, new offspring are sampled, hence progressing the algorithm. The recombination process to form the mean from the offspring samples is shown below.

$$\boldsymbol{w}_{g+1=} \sum_{k=1}^{\mu} \eta_k \boldsymbol{x}_{k:\lambda}^{g+1} \tag{16}$$

Here, $\boldsymbol{x}_{k:\lambda}^{g+1} \in \mathbb{R}^{\boldsymbol{n}}$ is a vector of n decision variables representing the k^{th} best offspring of the λ samples generated by the mutation process, $\boldsymbol{w}_{g+1} \in \mathbb{R}^{\boldsymbol{n}}$ is the shifted mean, i.e., the updated parameter vector, $\eta_k$is the k^{th} positive weight corresponding to the k^{th} offspring, and μ is the number of offspring selected for recombination. If $\mu > 1$, the algorithm is known as (μ, λ) CMA-ES.

Covariance Matrix Adaption

The covariance of the distribution from which the offspring of the new generation are sampled needs to be estimated. Self-adaption of covariance matrix allows for control over exploration. Improvement in the estimation of covariance matrix occurs by increasing the influence of successful mutations by the same fitness-dependent weighting scheme that was used for updating the mean in Equation (16).

The covariance matrix can be estimated from the offspring samples at each generation as follows,

$$\boldsymbol{C}_{est}^{g+1} = \frac{1}{\lambda}((\boldsymbol{x}_k^{g+1} - \boldsymbol{w}_g)(\boldsymbol{x}_k^{g+1} - \boldsymbol{w}_g)^T) \quad (17)$$

where, $\boldsymbol{C}_{est}^{g+1} \in \mathbb{R}^{n \times n}$ is the estimate of covariance matrix at generation $g + 1$ of λ samples. All other terms are as defined before.

For μ selections per generation covariance matrix update is given by,

$$\boldsymbol{C}_{\mu est}^{g+1} = \sum_{k=1}^{\mu} \eta_k \, (\boldsymbol{x}_{k:\lambda}^{g+1} - \boldsymbol{w}_g)(\boldsymbol{x}_{k:\lambda}^{g+1} - \boldsymbol{w}_g)^T \quad (18)$$

where, $\boldsymbol{C}_{\mu est}^{g+1} \in \mathbb{R}^{n \times n}$ is the weighted covariance estimate at generation $g + 1$ obtained when μ is the best offspring from the population of λ samples are selected. All other terms are defined as before.

Rank-one update: The rank-one update offers insight into the process involved in the covariance matrix adaption. Consider the vectors $\boldsymbol{y}_i \in \mathbb{R}^n$, where, $i = 1 \ldots g$, and $g \geq n$. Also, consider the independent, normally distributed random numbers between 0 and 1, given by $\mathcal{N}(0,1)$. Then, the relationship expressed in Equation (20) can be formulated.

$$\mathcal{N}(0,1)\boldsymbol{y}_1 + \mathcal{N}(0,1)\boldsymbol{y}_2 + \mathcal{N}(0,1)\boldsymbol{y}_3 + \cdots + \mathcal{N}(0,1)\boldsymbol{y}_g \sim \mathcal{N}(0, \sum_{i=0}^{g} \boldsymbol{y}_i \boldsymbol{y}_i^T) \quad (19)$$

Here, $\mathcal{N}(0, \sum_{i=0}^{g} \boldsymbol{y}_i \boldsymbol{y}_i^T)$ is a multivariate normally distributed random vector with zero mean and covariance $\sum_{i=0}^{g} \boldsymbol{y}_i \boldsymbol{y}_i^T$ that is formed by the summation of line distributions $\mathcal{N}(0,1)\boldsymbol{y}_i$. The symbol ~ indicates equivalence in distribution. From the description of each line distribution in the left- hand side of Equation (19), we find that $\mathcal{N}(0,1)\boldsymbol{y}_i \sim N(0, \boldsymbol{y}_i \boldsymbol{y}_i^T)$ yields the vector $\boldsymbol{y}_i$ with maximum likelihood among all normal distributions with zero mean. The term,

$$\boldsymbol{y}_{g+1} = \frac{\boldsymbol{y}_{1:\lambda}{}^{g+1} - \boldsymbol{w}^g}{\sigma_g} \quad (20)$$

describes the step to update the mean.

Thus, the rank-one covariance evolution is given by,

$$\boldsymbol{C}_1^{g+1} = (1 - c_1)\boldsymbol{C}_1^{g} + c_1 \boldsymbol{y}_{g+1}\boldsymbol{y}_{g+1}^{T} \tag{21}$$

where, $\boldsymbol{C}_1^{g+1} \in \mathbb{R}^{n\times n}$is the new rank-one covariance estimate at step $g+1$, $\boldsymbol{C}_1^{g} \in \mathbb{R}^{n\times n}$is the rank-one covariance estimate at generation g, and $\boldsymbol{y}_{g+1}$ is as defined before. The constant c_1 lies between 0 and 1, and describes the change rate of the covariance matrix. It allows for more influence from the present generation and fading of influence from previous generations.

Equation (21) increases the likelihood that a mutation $\boldsymbol{y}_{g+1}$ occurs in the evolution process by increasing the probability of generating this vector when sampled from the distribution with new covariance. Convergence occurs when there is no further change in the distribution. The rank of the covariance update at each step is one, hence, this is known as rank-one update. Figure 9 shows a representation of the rank-one update.

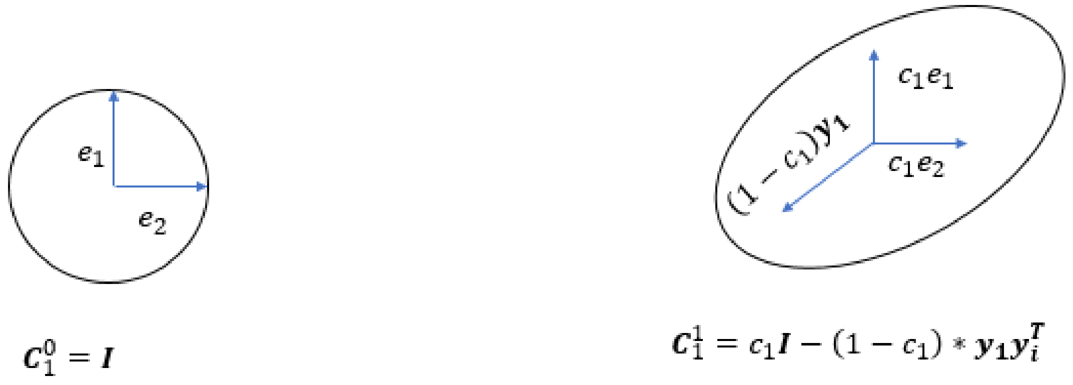
e_1
e_2
c_1e_1
$(1-c_1)y_1$
c_1e_2
$\boldsymbol{C}_1^0 = \boldsymbol{I}$
$\boldsymbol{C}_1^1 = c_1\boldsymbol{I} - (1-c_1) * \boldsymbol{y_1}\boldsymbol{y}_i^T$

FIGURE 9. A representation of a single step of the rank-one covariance matrix adaption.

Rank-μ update: Another type of update is known as the rank-μ update. It involves selecting μ offspring (or mutation steps) at each generation and using all the information provided by each generation for the covariance matrix update. This update is given by,

$$\boldsymbol{C}_{\mu}^{g+1} = (1 - c_{\mu})\boldsymbol{C}_{\mu}^{g} + c_{\mu} \textstyle\sum_{k=1}^{\mu} \eta_k \boldsymbol{y}_{k:\lambda}^{(g+1)} (\boldsymbol{y}_{k:\lambda}^{(g+1)})^{T} \tag{22}$$

where, $\boldsymbol{C}_{\mu}^{g+1} \in \mathbb{R}^{n\times n}$ is the rank-μ covariance matrix at generation $g + 1$, $\boldsymbol{C}_{\mu}^{g} \in \mathbb{R}^{n\times n}$, the rank-$\mu$ covariance matrix at generation g. The constant c_{μ} lies between 0 and 1. Its role in Equation (22) is similar to that of the constant $c_1$in Equation (21). All other terms are as described before.

The term $\sum_{k=1}^{\mu} \eta_k \boldsymbol{y}_{k:\lambda}^{(g+1)} (\boldsymbol{y}_{k:\lambda}^{(g+1)})^{T}$ in Equation (22) influences the mutation steps that can occur in the succeeding generations by updating the covariance matrix of the distribution.

Cumulation

From the rank-one update procedure we now define cumulation. Cumulation refers to the summation of all the successive steps that generate an evolution path of the distribution. The evolution path indicates the path traversed by the mean of the evolving distribution. The covariance adaption is now described in terms of path length rather than mutation steps. This description retains the sign information of the displacement of the mean which is lost due to the outer product of the mutation step in the update of covariance matrix in Equation (22).

Thus, evolution path is given by,

$$\boldsymbol{p}_{c}^{g+1} = (1 - c_c)\boldsymbol{p}_{c}{}^{g} + \sqrt{c_c(2 - c_c)\mu eff}\,\frac{\boldsymbol{w}^{g+1} - \boldsymbol{w}^{g}}{\sigma_g} \tag{23}$$

Here, $\boldsymbol{p}_{c}^{g}$ and $\boldsymbol{p}_{c}^{g+1} \in \mathbb{R}^{n}$ are vectors representing the evolution of path at generation g and $g + 1$ respectively, the term $\frac{\boldsymbol{w}^{g+1} - \boldsymbol{w}^{g}}{\sigma_g}$ is the shift in the mean from generation g to $g + 1$, σ_g is the step-size at generation g, and c_c is a constant between 0 and 1 and again plays a similar role to constants c_1 and c_{μ} in Equations (21) and (22) respectively, the term$\sqrt{c_c(2 - c_c)\mu eff}$ is a normalizing factor, where, μeff is called the variance effective selection mass given by,

$$\mu eff = \frac{1}{\sum_{k=1}^{\mu} \eta_k^2}\,;\ 0 \leq \mu eff \leq\ 1 \tag{24}$$

When the terms $c_c = 1$ and $\mu eff = 1$, the path evolution simply becomes the displacement of the mean. The rest of the terms are as defined before.

From Equations (22) and (24), the rank-one update of covariance matrix in terms of evolution path is defined as,

$$\boldsymbol{C}_1^{g+1} = (1 - c_1)\boldsymbol{C}_1^g + c_1 \boldsymbol{p}_c^{g+1} \boldsymbol{p}_c^{g+1^T} \tag{25}$$

Extending the above definition to rank-μ update,

$$\boldsymbol{C}_\mu^{g+1} = \left(1 - c_1 - c_\mu\right)\boldsymbol{C}_\mu^g + c_1 \boldsymbol{p}_c^{g+1} \boldsymbol{p}_c^{g+1^T} + c_\mu \sum_{k=1}^{\lambda} \boldsymbol{w}_k \boldsymbol{y}_{k:\lambda}^{g+1} \boldsymbol{y}_{k:\lambda}^{(g+1)^T} \tag{26}$$

where all terms are as defined before.

Step-Size Control

The path of evolution is controlled by adapting the step-size σ. It is observed that the path of evolution depends on the direction of successive steps. If successive steps are correlated, the path length is long and step-size is increased. If correlation of successive steps is uncorrelated, then path length is small and step-size is decreased to accommodate for the changing direction of the paths. Thus, step-size is dependent on the comparison of actual (or conjugate) path with the expected path constructed by random selection. Step-Size control is summarized in the following equation.

$$\boldsymbol{p}_\sigma^{g+1} = (1 - c_\sigma)\boldsymbol{p}_\sigma^g + \sqrt{c_\sigma(2 - c_\sigma)\mu eff}\,\boldsymbol{C}_\sigma^{(g)-\frac{1}{2}} \frac{\boldsymbol{w}^{g+1} - \boldsymbol{w}^g}{\sigma} \tag{27}$$

The terms $\boldsymbol{p}_\sigma^{g+1}, \boldsymbol{p}_\sigma^g \in \mathbb{R}^n$ are conjugate path lengths at generation $g + 1$ and g respectively. The term $\boldsymbol{C}_\sigma^{(g)-\frac{1}{2}}$ is defined by $\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)-1}\boldsymbol{B}^{(g)\,T}$ that comes from the eigen value decomposition of $\boldsymbol{C}_\sigma^g$**,** given by $\boldsymbol{C}_\sigma^g = \boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)\,2}\boldsymbol{B}^g$**,** where $\boldsymbol{B}^g$ is the orthonormal basis of eigen vectors, and $\boldsymbol{D}^{(g)}$ is the diagonal matrix of the square root of the corresponding eigen values.

The step-size update is given by,

$$\sigma^{g+1} = \sigma^{g} \exp(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\boldsymbol{p}_\sigma^{g+1}\right\|}{E\|\mathcal{N}(0,\boldsymbol{I})\|} - 1\right)) \tag{28}$$

where, σ^{g+1} is the step-size at generation $g + 1$ and $E\|\mathcal{N}(0, \boldsymbol{I})\|$ is the expected path length. The constant d_σ is a positive damping coefficient.

Application of Evolutionary Algorithms to Model-Free Control

An evolutionary controller must consist of a parameterized component such as a neural-network that allows for adaption. Evolution occurs on the weights of the neural-network. This class of methods that apply evolutionary strategies to neural-networks is called as neuro-evolution. An example of neuro-evolution algorithm is SANE (Symbiotic Adaptive Neuro-evolution), that evolves the weights of neural-network-based controllers using genetic algorithm, as described in the work of Moriarty and Miikkulainen [30]. Igel [31] shows neuro-evolution for reinforcement learning in controllers using evolutionary algorithms.

Borrowing terminology from biological evolution, the neural-network is called the phenotype of the optimization problem and the corresponding parameter vector formed by its weights is called the genotype.

The DMP framework covered in Chapter 3 uses a dynamic neural-network, whose parameters manipulate the attracter landscape to change the shape of the trajectories. This parameterized network is given by the Equation (8). The CMA-ES technique covered in this chapter is applied to evolve the parameters of the neural-network in the DMP, thus falling under the purview of neuro-evolution. Figure 10 shows how a DMP could be set up for neuro-evolution. The dynamic neural-network given by Equation (8) gives the phenotype and its corresponding weight vector $\boldsymbol{w}$ gives the genotype of the neuro-evolution problem.

A Model-Free Control Architecture

A model-free architecture that allows the controller to structurally evolve is shown in Figure 11. The term f represents the fitness evaluation from an objective function. The vector $\boldsymbol{q_f}$ consists of the states of the system measured for evaluating fitness. The vector $\boldsymbol{w}$ consists of the adaptable parameters of the DMP. The fitness value f determines the parameter $\boldsymbol{w}$ drawn from the CMA-ES block.

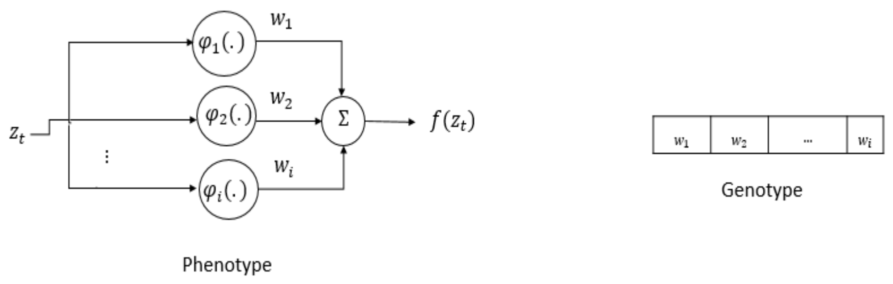

FIGURE 10. Parameter vector for neuro-evolution of DMP. The input z_t represents the internal state from Equation (2) at time t. The output is the nonlinear function $f(z_t)$ at time t from Equation (8). It is formed from the weighted sum of the N normalized radial basis, each given by $\varphi_i(z_t)$

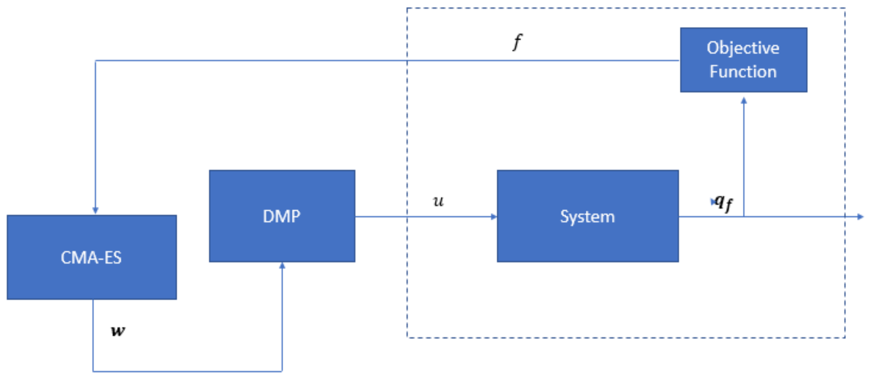

FIGURE 11. Evolving an open-loop controller with CMA-ES. The algorithm is applied offline.

CHAPTER 5

THE DOUBLE INVERTED PENDULUM MODEL

The double inverted pendulum considered for computer simulated experimentation is shown in Figure 12. The Equation (29) below, as derived by Hamano [32], describes the corresponding dynamics of the system.

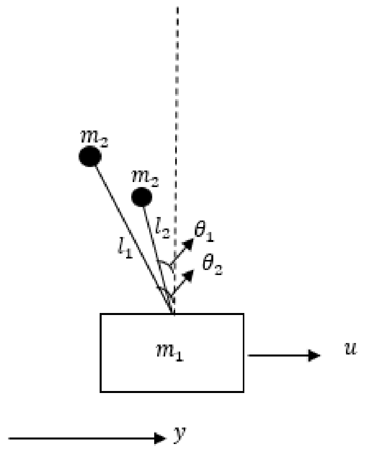

FIGURE 12. Double inverted pendulum.

$$\begin{bmatrix} m_1+2m_2 & -m_2 l_1\cos(\theta_1) & -m_2 l_2\cos(\theta_2) \\ -\cos(\theta_1) & l1 & 0 \\ -\cos(\theta_2) & 0 & l2 \end{bmatrix} \begin{bmatrix} \ddot{y} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} \begin{bmatrix} m_2 l_1 \sin(\theta_1)\dot{\theta}_1^2 + m_2 l_2 \sin(\theta_2)\,\dot{\theta}_2^2 \\ -g\sin(\theta_1) \\ -g\sin(\theta_2) \end{bmatrix} = \begin{bmatrix} u \\ 0 \\ 0 \end{bmatrix} \quad (29)$$

where, m_1 is the mass of the pivot point, the mass of each pendulum is m_2, l_1 and l_2 are the lengths of the two pendulums, y is the position of the pivot point, and θ_1 and θ_2 are the orientation of each pendulum with respect to the vertical. Let,

$$x_1 = \dot{y}, x_2 = \dot{\theta}_1,\ x_3 = \dot{\theta}_2 \quad (30)$$

$$M = \begin{bmatrix} m_1+2m_2 & -m_2 l_1\cos(\theta_1) & -m_2 l_2\cos(\theta_2) \\ -\cos(\theta_1) & l1 & 0 \\ -\cos(\theta_2) & 0 & l2 \end{bmatrix} = \begin{bmatrix} M11 & M12 & M13 \\ M21 & M22 & M23 \\ M31 & M32 & M33 \end{bmatrix} \quad (31)$$

$$M^{-1} = \begin{bmatrix} MI11 & MI12 & MI13 \\ MI21 & MI22 & MI23 \\ MI31 & MI32 & MI33 \end{bmatrix} \tag{32}$$

$$N = \begin{bmatrix} m_2 l_1 \sin(\theta_1) x_2^2 + m_2 l_2 \sin(\theta_2) x_3^2 \\ -g\sin(\theta_1) \\ -g\sin(\theta_2) \end{bmatrix} = \begin{bmatrix} N1 \\ N2 \\ N2 \end{bmatrix} \tag{33}$$

Thus, the complete state equations are given by,

$$\dot{x}_1 = x_4 \tag{34}$$

$$\dot{x}_2 = x_5 \tag{35}$$

$$\dot{x}_3 = x_6 \tag{36}$$

$$\begin{bmatrix} \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} MI11 & MI12 & MI13 \\ MI21 & MI22 & MI23 \\ MI31 & MI32 & MI33 \end{bmatrix} \begin{bmatrix} u - N1 \\ -N2 \\ -N3 \end{bmatrix} \tag{37}$$

From Equations (34) – (37) of the six-dimensional state dynamics of the double inverted pendulum model, we find that only the pivot point can be directly actuated due to control input u. The pendulums are controlled only through coupling. Since the number of directly controlled degrees of freedom is less than the total degrees of freedom, the system is underactuated. Thus, it can be seen from Equations (34) – (37) that the system cannot be easily linearized by feedback through control term u. Also, the dynamics due to term $N1$ are difficult to estimate due to higher degree terms as found by Equation (33). Thus, to control the full dynamics of the double inverted pendulum, a model-free approach is more useful.

CHAPTER 6

EXPERIMENT METHODOLOGY

This chapter explains the methodology used for the computer simulated experiment in Chapter 7.

Problem Introduction and Experiment Goal

The design of a controller for a double inverted pendulum is a non-trivial problem due to highly coupled nonlinearities within the system. Also, closed form control with full feedback linearization is difficult as the system is underactuated. Hence, the double inverted pendulum is a benchmark problem in control and machine learning.

The research in double inverted pendulum control is divided into two camps, the stabilization problem and the swing-up problem. The stabilization problem finds a controller that balances an inverted pendulum (or pendulums) on a cart [22], [33], [34]. This requires synchronization of the cart movement with the oscillation of the inverted pendulum.

The swing-up problem [35], [36] has recently drawn interest and is considered here. This problem involves the design of a controller that brings the double inverted pendulum close to the unstable equilibrium at the upright position given by $\left[y, \theta_1, \theta_2, \dot{y}, \dot{\theta}_1, \dot{\theta}_2\right] =$ [0,0,0,0,0,0], where θ_1, θ_2 belong to modulo 2π space. The double inverted pendulum is initially at the stable equilibrium at the pendant position given by $\left[y, \theta_1, \theta_2, \dot{y}, \dot{\theta}_1, \dot{\theta}_2\right] = [0, \pi, \pi, 0,0,0]$; where θ_1, θ_2 belong to modulo 2π space. It is assumed that a stabilization controller takes over once the double pendulum is close to upright position. An open-loop controller described by the DMP is evolved by using the CMA-ES evolution algorithm.

Feedforward Control

In the open-loop control scheme used, the feedforward control is provided by the Dynamic Movement Primitive (DMP) described in Equations (1) – (6). This thesis asserts that the field generated by an external DMP system can drive the double inverted pendulum to the desired state.

$\ddot{x}$ from the DMP in Equations (1) – (6) generates the desired force plan applied to the control-input to the double inverted pendulum Equations (34) – (37).

We then end up with the following control law:

$$u = \ddot{x} \tag{38}$$

where $\ddot{x}$ comes from the DMP system in Equations (1) – (6).

Evolving the Controller

The CMA-ES evolution strategy is used to evolve the appropriate forces described by $\ddot{x}$ by evolving the parameter vector $\boldsymbol{w}$ in the Equations (1) – (6), that allows for swing-up of the two pendulums. The evolution occurs in offline mode. It is assumed that measurements of the final states of the experiment are available to calculate fitness from objective functions after each trial simulation.

Case Studies with Different Objective Functions

Three energy-based objective functions are used with the CMA-ES algorithm in Chapter 7. It is assumed that zero potential is fixed at upright position. An additional spring potential is added to the energy of the pivot point, with zero potential at the desired position of the pivot point.

Preliminaries

In this section, we consider the energies of the different subsystems of the double inverted pendulum of Figure 12 from the derivation provided by Hamano [32].

The kinetic energy and potential energy of the first pendulum are given below respectively,

$$K.E._{fp1} = \frac{1}{2}m_2\left(\dot{y}^2 - 2l_1\cos(\theta_1)\,\dot{\theta}_1\dot{y} + l_1^2\dot{\theta}_1^2\right) \tag{39}$$

$$P.E._{fp1} = m_2 g l_1(\cos(\theta_1) - 1) \tag{40}$$

The kinetic energy and potential energy of the second pendulum are given by Equations (40) and (41) given below respectively,

$$K.E._{sp2} = \frac{1}{2}m_2\left(\dot{y}^2 - 2l_2\cos(\theta_2)\,\dot{\theta}_2\dot{y} + l_2^2\dot{\theta}_2^2\right) \tag{41}$$

$$P.E._{sp2} = m_2 g l_2(\cos(\theta_2) - 1) \tag{42}$$

The kinetic energy of pivot point and artificial potential are given by,

$$K.E._{c1} = \frac{1}{2}m_1\dot{y}^2 \tag{43}$$

$$A.E._{c1} = \frac{1}{2}ky^2\ ;\ k > 0 \tag{44}$$

Case Study 1

In the first case, the sum of the squared energies of each of the subsystem of the double inverted pendulum is used as the objective function. From Equations (39) – (44), the sum of squared energies of each subsystem is given by,

$$E_S^2 = \left(k.E._{sp1} - P.E.sp_2\right)^2 + \left(k.E._{sp2} - P.E._{sp2}\right)^2 + (K.E._c + A.E._c)^2 \tag{45}$$

$$f_1 = (E_S^2)_{final} \tag{46}$$

where, $(E_S^2)_{final}$ is the evaluation of (E_S^2) at the final configuration of the double inverted pendulum. The value f_1 is to be minimized, preferably to zero, to get the best controller.

Case Study 2

In the second case, the square of the final value of the Lagrangian of the double inverted pendulum is the objective to be minimized. From Equations (39) – (44), the Lagrangian is given by,

$$L = \left(K.E._{sp1} + K.E._{sp2} + K.E.c\right) - \left(P.E._{sp1} + P.E._{sp2}\right) + A.E._{c} \tag{47}$$

The objective function is evaluated at the end of experiment to obtain a value that represents the fitness of the chosen controller.

$$f_2 = (L^2)_{final} \tag{48}$$

where, $(L^2)_{final}$ is the evaluation of (L^2) at the final configuration of the double inverted pendulum. The value f_2 is to be minimized, preferably to zero, to get the best controller.

Case Study 3

The objective is the final value of the Lagrangian L from Equation (48) without the square, i.e.,

$$f_3 = L_{final} \tag{49}$$

Motivations for the Choice of Energy-Based Objective Functions

In nonlinear control theory, the Lyapunov direct method is extensively used to design controllers [7]. This method involves the choice of an appropriate positive definite Lyapunov function. If the control is chosen such that the time-derivative of the Lyapunov function is negative definite, then eventually the system will be driven to stability. Thus, when minimum of Lyapunov function is reached, the system is at the desired state.

In case 1, we use the format of the Lyapunov function defined to derive the energy-based control for swing-up of inverted pendulum by Åström and Furuta [35]. For the positive definite Lyapunov function, square of the energy terms was chosen such that when it became zero, essentially the required state configuration was reached. Such a function could be used to derive an energy regulator to accomplish energy-based control of the system. Case 2 uses the square of the Lagrangian in the same spirit as case 1. Case 3 uses a simple Lagrangian to test the performance on a conventional energy-based Lyapunov function as described in standard texts [3], [4].

Using an evolution algorithm, the controller structurally evolves such that the dynamics of the system reach the minimum of the Lyapunov-like objective function at the final state of the experiment. It can be observed that if chosen objective functions $f_1, f_2,$ or f_3=0, then, the double pendulum has reached upright configuration with states $\left[y, \theta_1\ , \theta_2, \dot{y}, \dot{\theta}_1, \dot{\theta}_2\right] =$ [0,0,0,0,0,0]; where θ_1, θ_2 belong to modulo 2π space.

CHAPTER 7

SIMULATION AND RESULTS

The simulation results for evolution-based swing-up control for the double inverted pendulum system as described in Chapter 6 are presented here. The model parameters are shown in Table 1. The computer simulation was done using the MATLAB analytical software. Each experimental trial began at $[y, \theta_1, \theta_2, \dot{y}, \dot{\theta}_1, \dot{\theta}_2] = [0, \pi, \pi, 0,0,0]$; θ_1, θ_2 belong to modulo 2π space. The final configuration desired for the upright position was $[y, \theta_1, \theta_2, \dot{y}, \dot{\theta}_1, \dot{\theta}_2] = [0,0, 0,0,0,0]$; again, θ_1, θ_2 belong to modulo 2π space.

The methodology explained in Chapter 6 was followed to perform the experiment. The controller architecture was described by the DMP in Chapter 3. The DMP parameters were chosen arbitrarily and are given in Table 2. The Covariance Matrix Adaption Evolution Strategy (CMA-ES) covered in Chapter 4 was used to evolve the controller.

The dynamics of the states of the system were plotted before and after the evolution of the controller for each of the energy based objective functions f_1, f_2 and f_3 given in Chapter 6. The evolution rate with respect to the evaluation of each of the objective function was plotted. The evolution algorithm was terminated when value of objective function became less than or equal to 0.01 or showed no significant change for fifty generations. The analysis of experiments is provided in Chapter 8. The results are shown in Figures 13 – 42. Table 3 gives the data for the evolutionary process in terms of the initial and final generation of the evolution.

Note that units of measurement for all the quantities described are expressed in the metric system. Mass, length, angle, and time are measured in kilograms represented by kg, meters represented by m, radians represented by rad, and seconds represented by s respectively. The force exerted by the controller to drive the dynamics of the system is measured in Newtons. The unit of energy is Joule and represented by J.

TABLE 1. Parameters of the Inverted Pendulum Chosen for the Experiment

Parameters of Inverted Pendulum	**Symbols**	**Values Chosen**
Mass of Cart (pivot point)	m_1	$1\ kg$
Mass of each pendulum	m_2	$0.3\ kg$
Acceleration due to Gravity	g	$9.8\ m/s^2$
Length of First pendulum	l_1	$1\ m$
Length of Second pendulum	l_2	$2\ m$

TABLE 2. Parameters of the DMP Mass-Spring-Damper Chosen for the Experiment

DMP Parameter	**Symbol**	**Value chosen**
Natural Frequency	ω_0	$5\ rad/s$
Damping Ratio	ζ	1
Initial Deviation from relaxed length of the spring	x_0	$1m$
Relaxed length of the spring	x_g	$0\ m$
Initial velocity	$\dot{x}_0$	$0\ m/s$
Number of basis functions	N	20

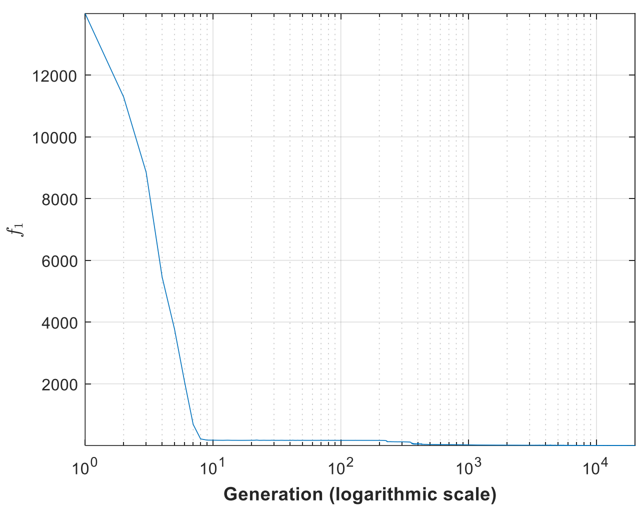

FIGURE 13. Results with objective f_1: evolution with square of sum of physical energies (in $Joule^2$) of pendulums and cart. x-axis is in logarithmic scale.

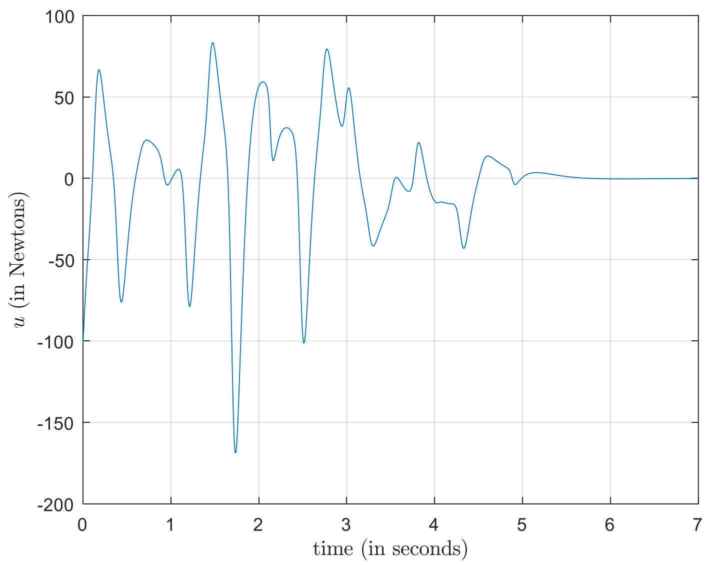

FIGURE 14. Results with objective f_1: evolved feedforward control input u.

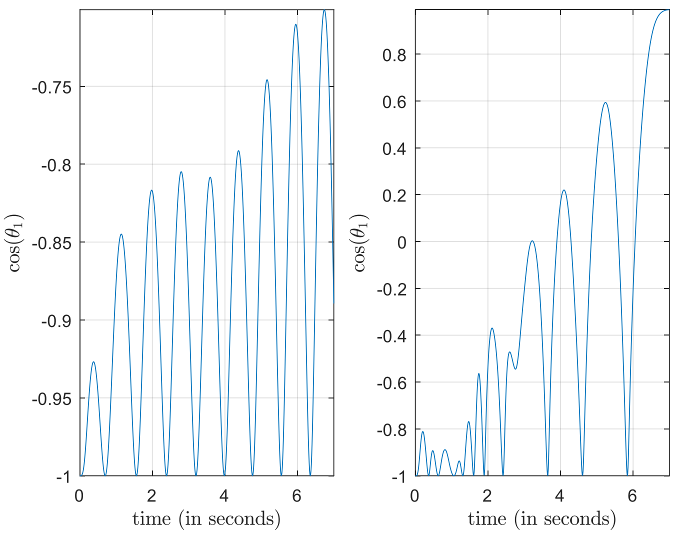

FIGURE 15. Results with objective f_1: $cos(\theta_1)$ trajectory. Left: Before controller evolution. Right: After controller evolution.

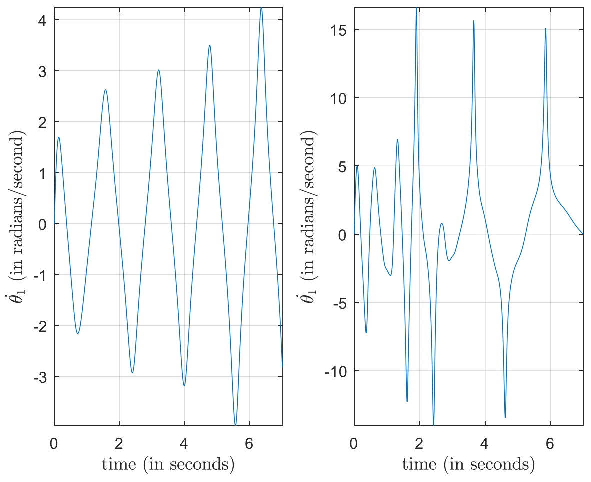

FIGURE 16. Results with objective f_1: $\dot{\theta}_1$ trajectory (in $radians/second$). Left: Before controller evolution. Right: After controller evolution.

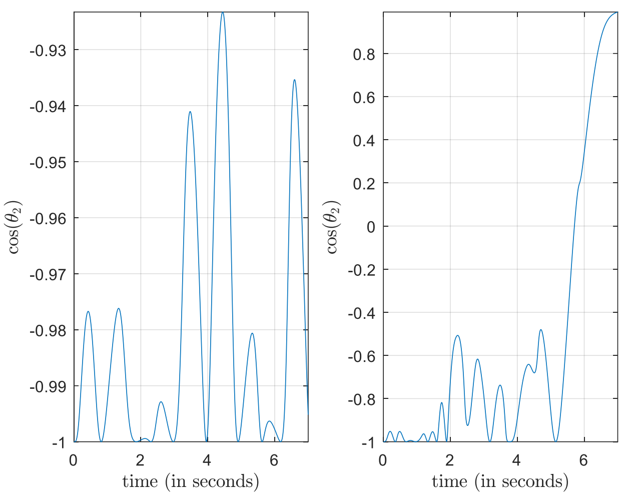

FIGURE 17. Results with objective f_1: $cos(\theta_2)$ trajectory. Left: Before controller evolution. Right: After controller evolution.

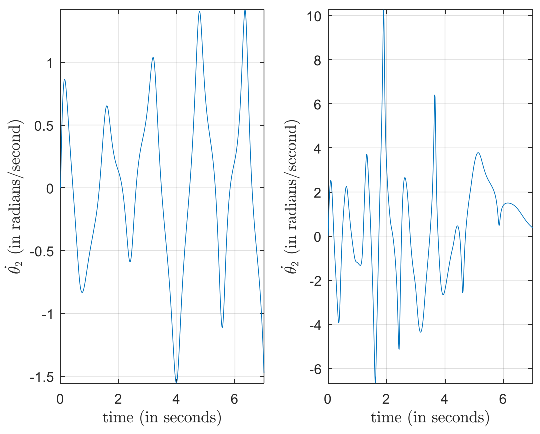

FIGURE 18. Results with objective f_1: $\dot{\theta}_2$ trajectory (in $radians/second$). Left: Before controller evolution. Right: After controller evolution.

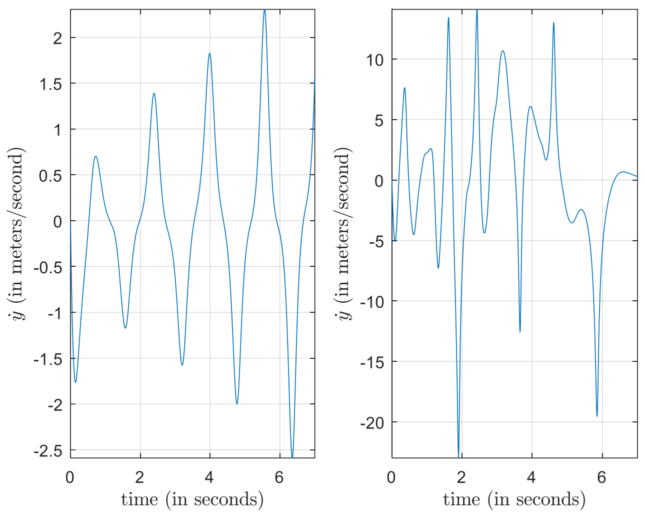

FIGURE 19. Results with objective f_1: $\dot{y}$ trajectory (in $meters/second$). Left: Before controller evolution. Right: After controller evolution.

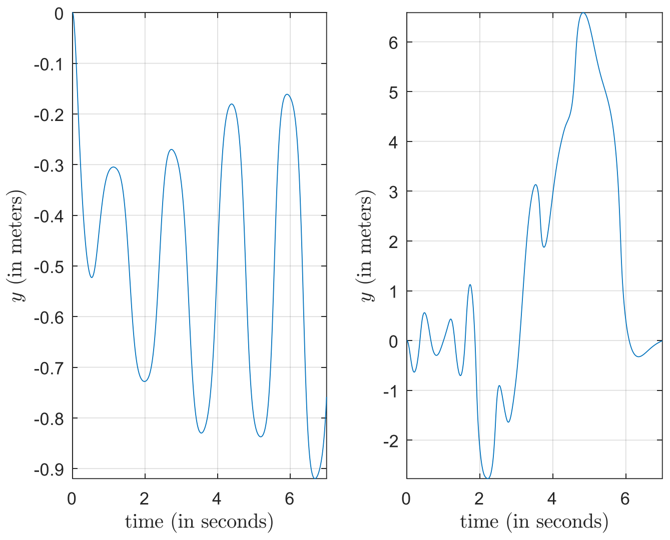

FIGURE 20. Results with objective f_1: y trajectory (in $meters$). Left: Before controller evolution. Right: After controller evolution.

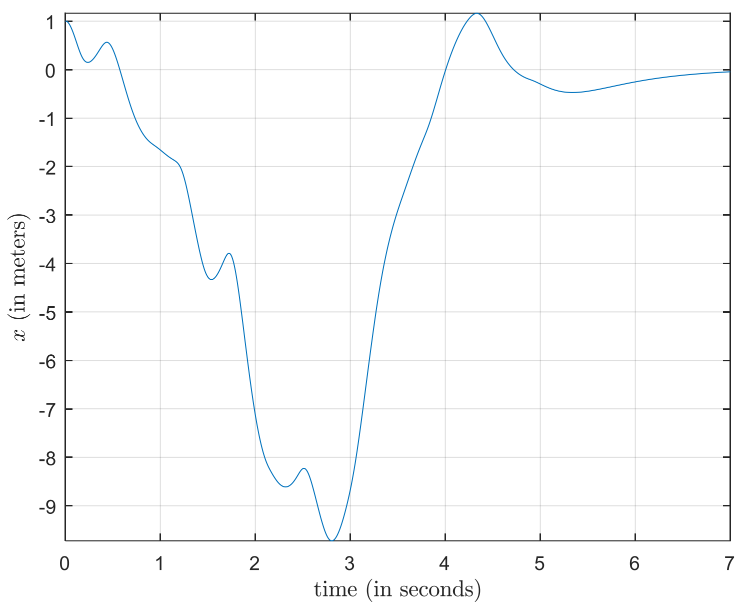

FIGURE 21. Results with objective f_1: position trajectory of the evolved DMP.

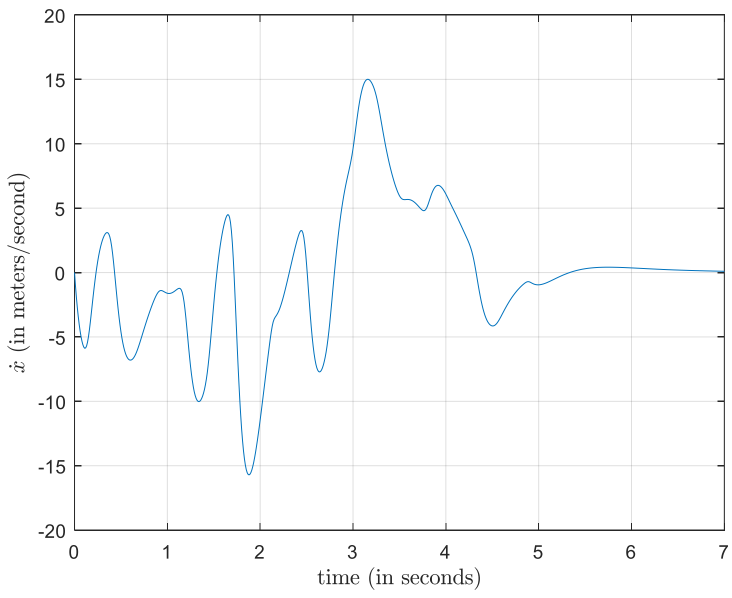

FIGURE 22. Results with objective f_1: velocity trajectory of the DMP.

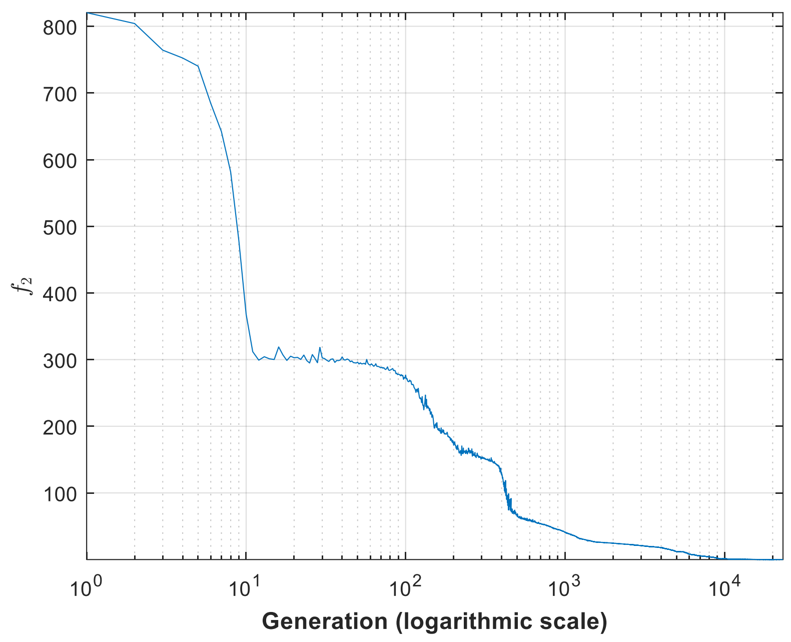

FIGURE 23. Results with objective f_2: evolution rate using square of the final value of Lagrangian (in $Joule^2$) as objective function. x axis is in logarithmic scale.

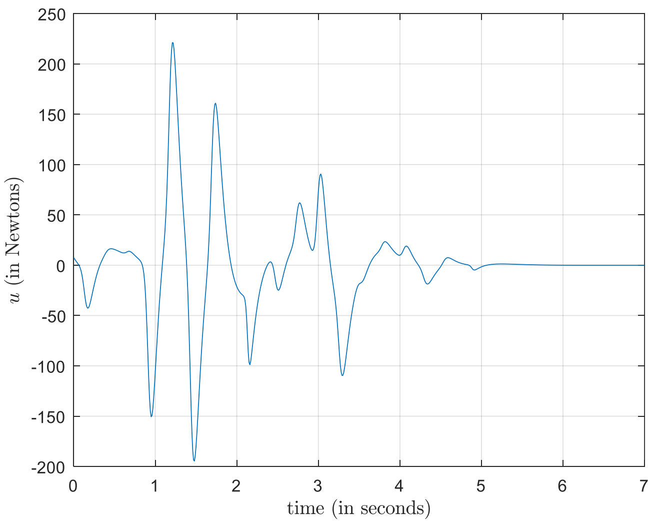

FIGURE 24. Results with objective f_2: evolved feedforward control input u.

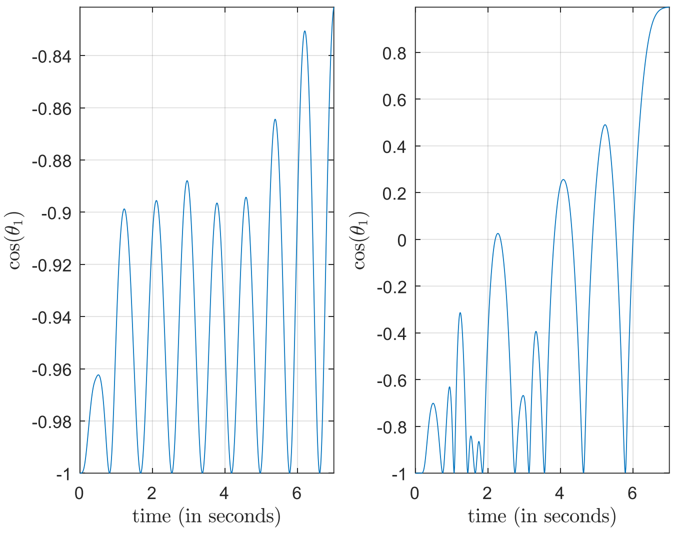

FIGURE 25. Results with objective f_2: $cos(\theta_1)$ trajectory. Left: Before controller evolution. Right: After controller evolution.

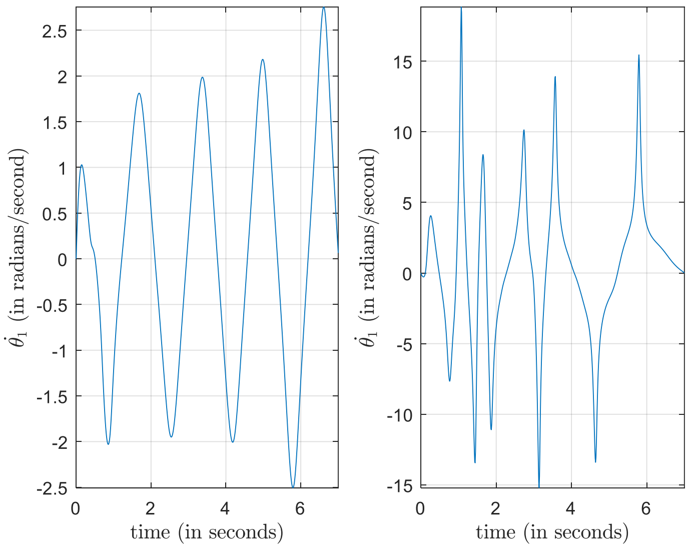

FIGURE 26. Results with objective f_2: $\dot{\theta}_1$trajectory (in $radians/second$). Left: Before controller evolution. Right: After controller evolution.

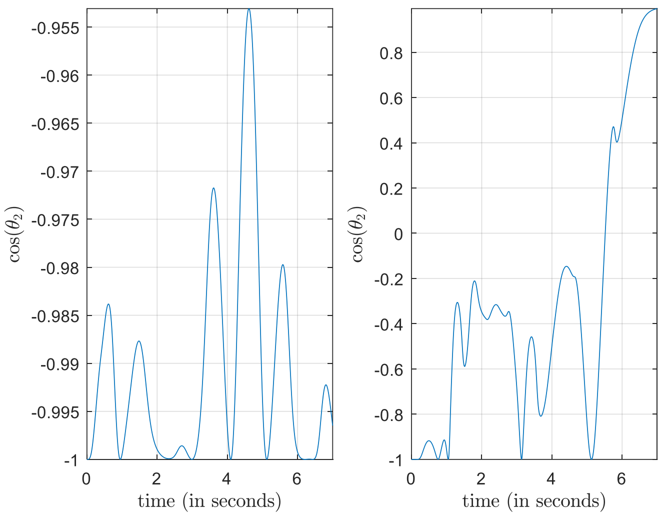

FIGURE 27. Results with objective f_2: $cos(\theta_2)$ trajectory. Left: Before controller evolution. Right: After controller evolution.

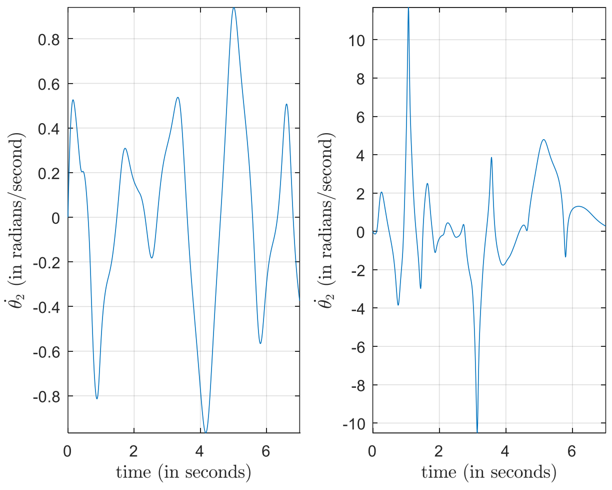

FIGURE 28. Results with objective f_2: $\dot{\theta}_2$ trajectory (in $radians/second$). Left: Before controller evolution. Right: After controller evolution.

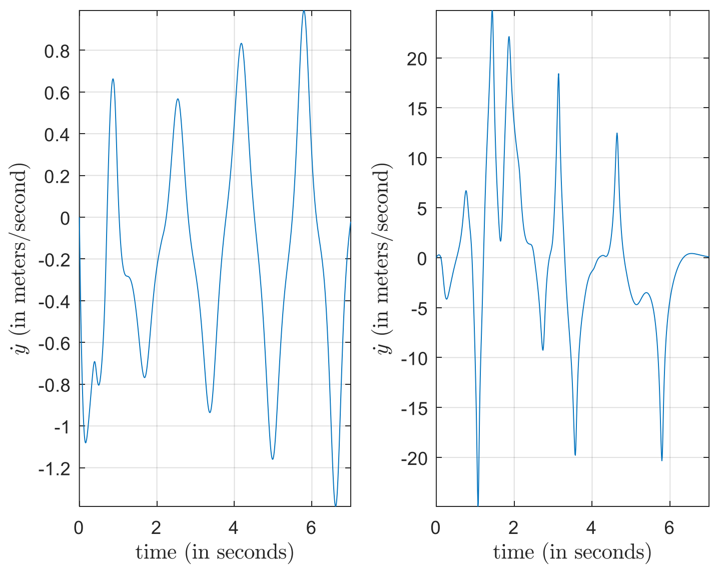

FIGURE 29. Results with objective f_2: $\dot{y}$ trajectory (in $meters/second$). Left: Before controller evolution. Right: After controller evolution.

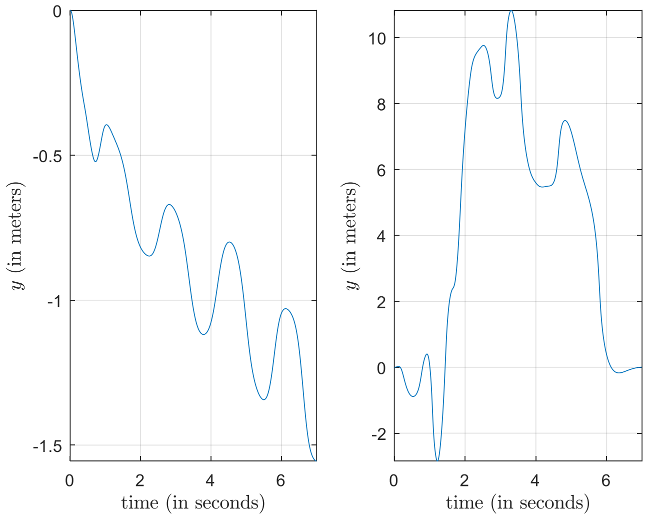

FIGURE 30. Results with objective f_2: y trajectory (in $meters$). Left: Before controller evolution. Right: After controller evolution.

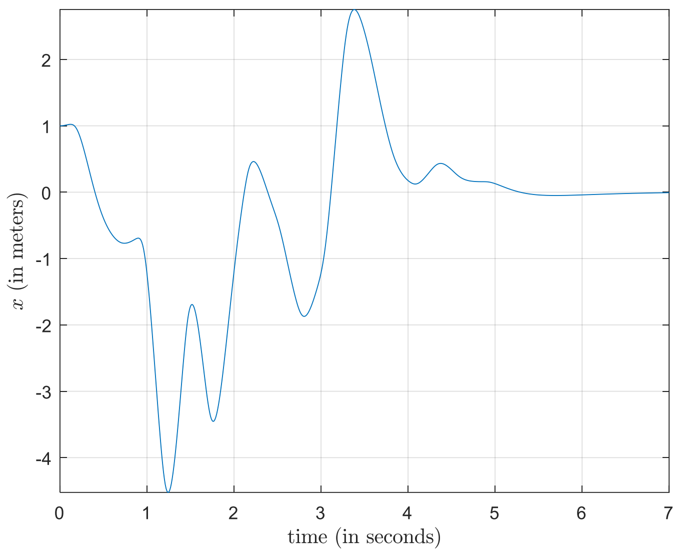

FIGURE 31. Results with objective f_2: position trajectory of the evolved DMP.

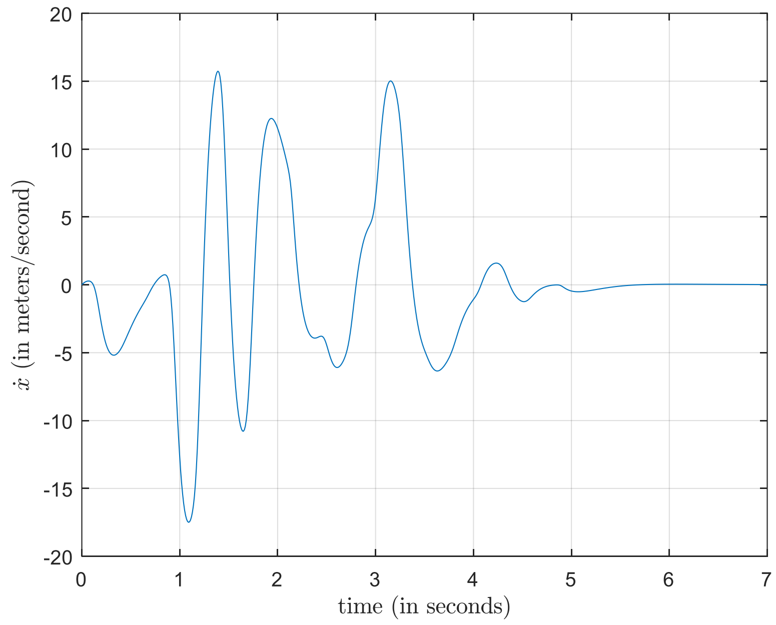

FIGURE 32. Results with objective f_2: velocity trajectory of the evolved DMP.

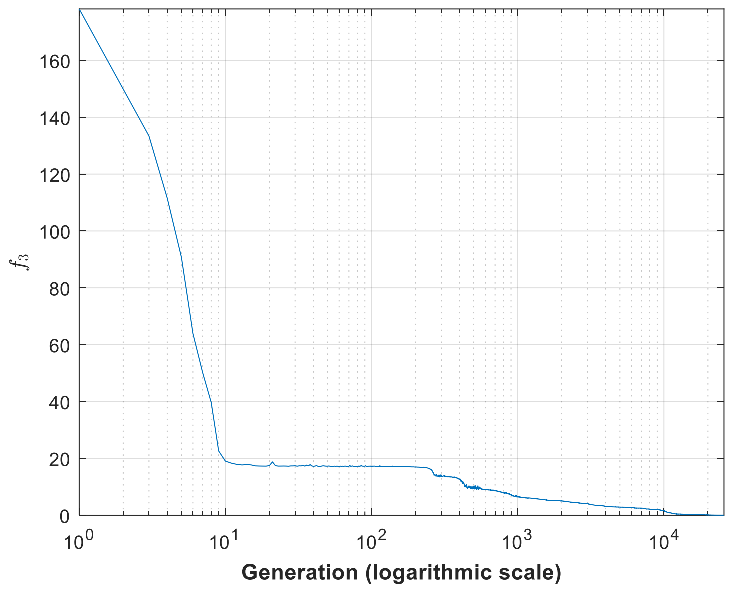

FIGURE 33. Results with objective f_3: evolution rate using final value of Lagrangian (in $Joules$) as objective function. x-axis is in Logarithmic scale.

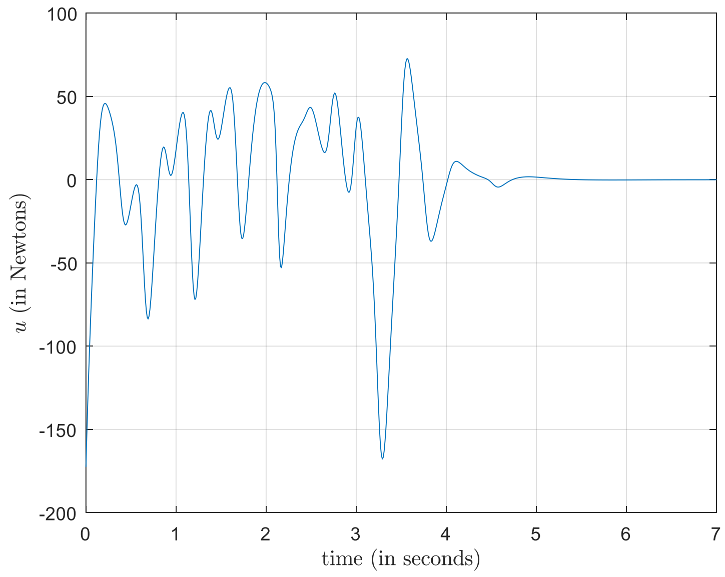

FIGURE 34. Results with objective f_3: evolved feedforward control input u.

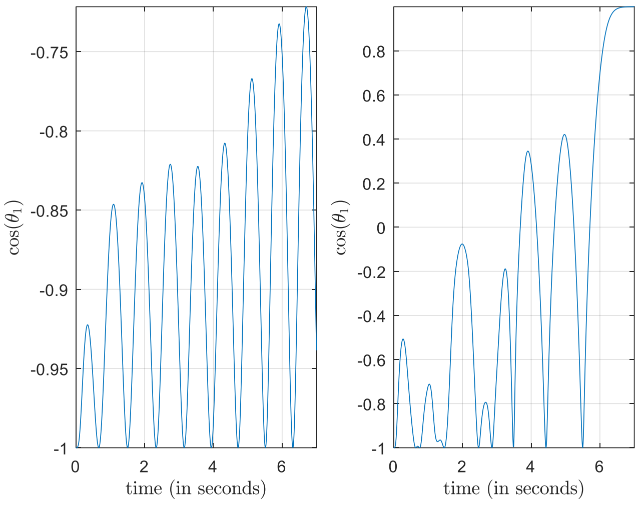

FIGURE 35. Results with objective f_3: $cos(\theta_1)$ trajectory. Left: Before controller evolution. Right: After controller evolution.

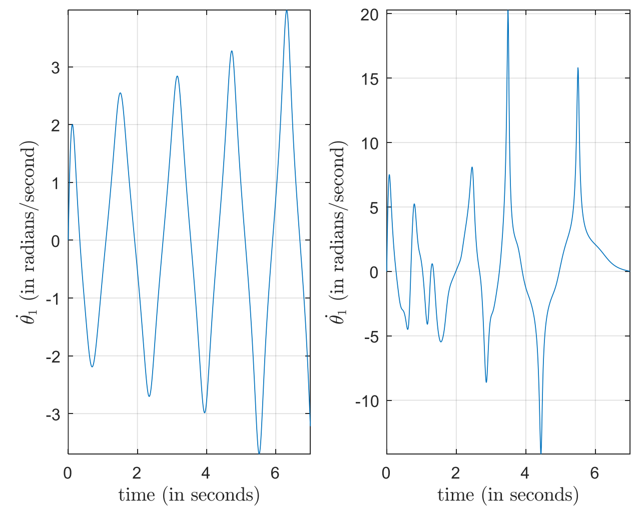

FIGURE 36. Results with objective f_3: $\dot{\theta}_1$ trajectory (in $radians/second$). Left: Before controller evolution. Right: After controller evolution.

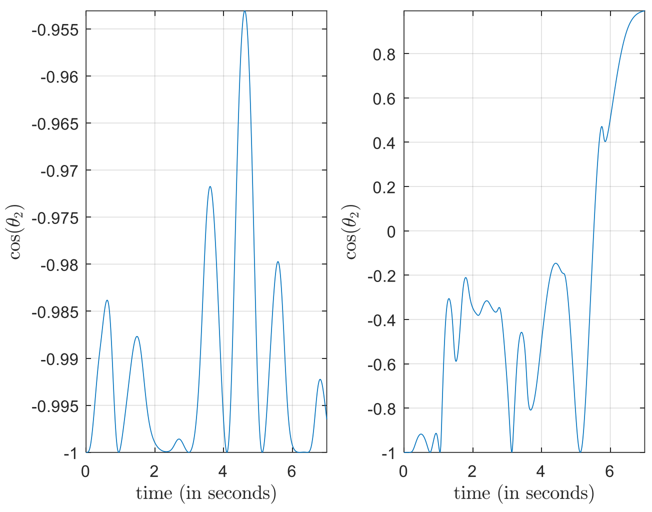

FIGURE 37. Results with objective f_3: $cos(\theta_2)$ trajectory. Left: Before controller evolution. Right: After controller evolution.

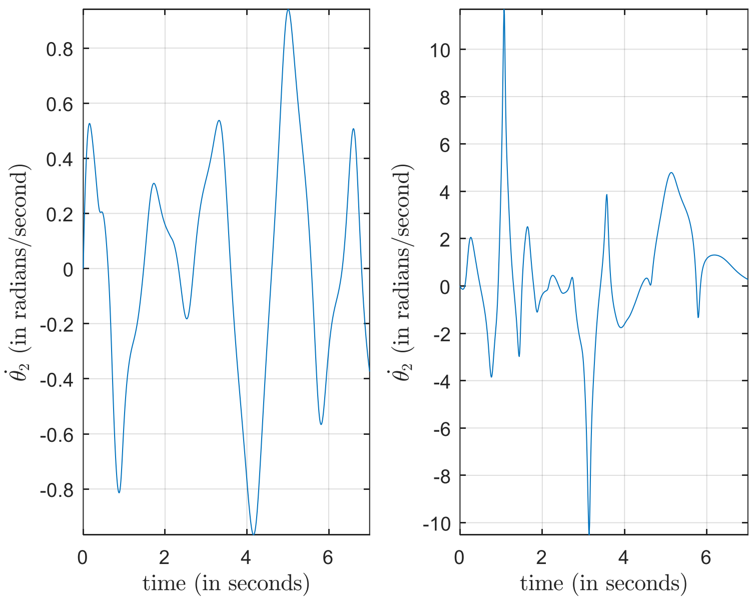

FIGURE 38. Results with objective f_3: $\dot{\theta}_2$trajectory (in $radians/second$). Left: Before controller evolution. Right: After controller evolution.

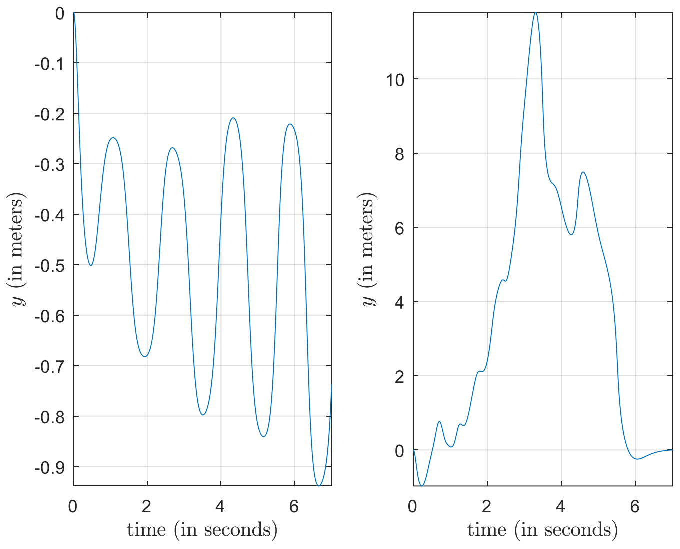

FIGURE 39. Results with objective f_3: y trajectory (in $meters$). Left: Before controller evolution. Right: After controller evolution.

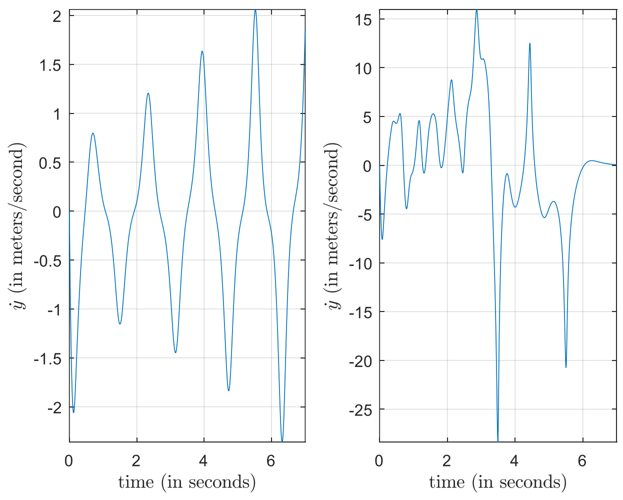

FIGURE 40. Results with objective f_3: $\dot{y}$ trajectory (in $meters/second$). Left: Before controller evolution. Right: After controller evolution.

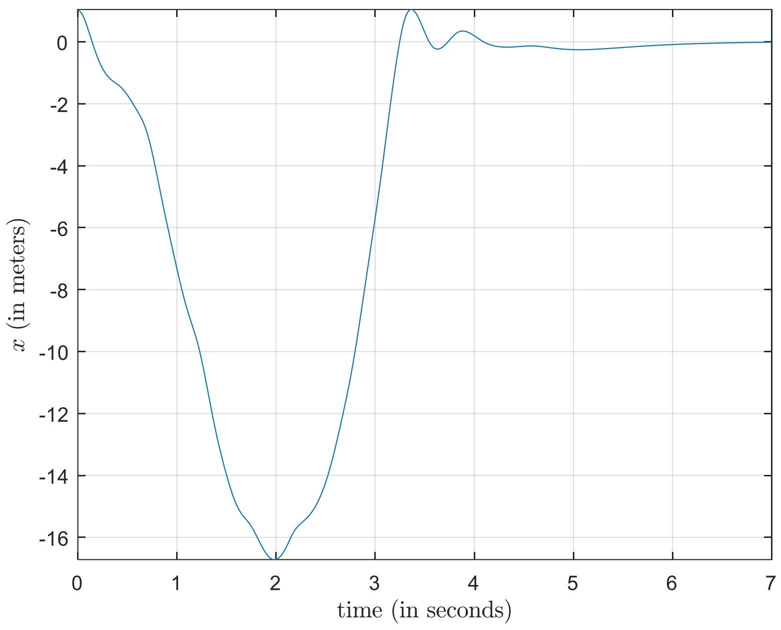

FIGURE 41. Results with objective f_3: position trajectory of the evolved DMP.

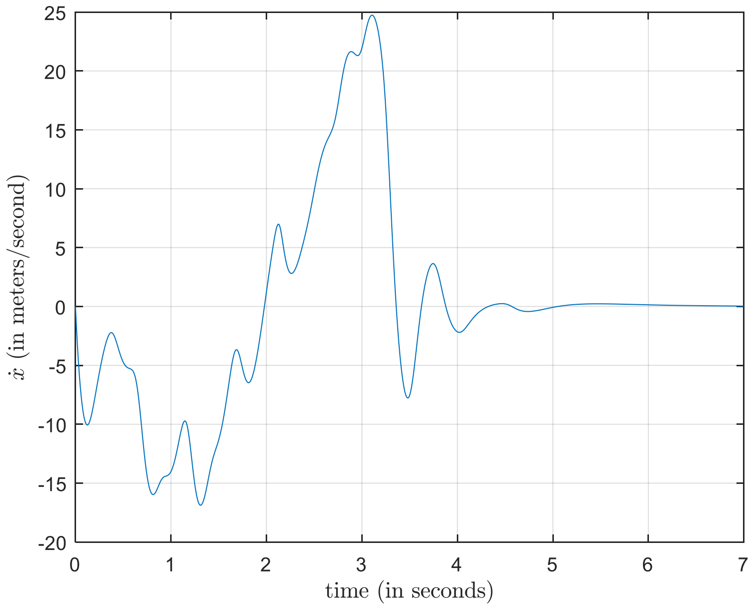

FIGURE 42. Results with objective f_3: velocity trajectory of the evolved DMP.

TABLE 3. Generation Data for the Experiments Performed

Objective Function	**Value of Objective Function at the First Generation**	**Value of Objective Funtion at final Generation**	**No. of Generations**
f_1	1.3994×10^4	0.01 (Reached Termination value)	20082
f_2	820.4718	0.01 (Reached Termination Value)	23218
f_3	178.0993	0.0156 (Termination due to flat evolution rate)	25761

CHAPTER 8

ANALYSIS

In all the chosen cases, from Figures 13 – 42, it was observed that the evolutionary controller was able to bring the energy of the double inverted pendulum close to zero, i.e., the double inverted pendulum was brought close to the upright position. Figure 13, Figure 23 and Figure 33 show the evolution rates of with objective functions f_1, f_2 and f_3. These figures provide a summary to the evolutionary process. Figure 14, Figure 24, and Figure 34 show the complicated feedforward control input that the evolutionary controller has effectively learnt. The visualization of the evolution process and some observations that could be useful for future research are presented here.

Visualization of the Evolution

Figure 43 presents the visualization of the evolution that occurs in the proposed method for the first 5000 generations with objective function f_1 in Equation (46). The axes represent the energy of each subsystem of the double inverted pendulum at the final state of each trial. For every stage of the controller evolution, there exists a point in the space represented in Figure 43 that corresponds to the energy that the final configuration of the double inverted pendulum can have. Evaluation of the objective function at each of these points represents the fitness or effectiveness of the controller. Clearly, effectiveness of the controller increases as the distance from the origin decreases. Thus, the origin is the desired energy configuration representing the minimum of the chosen objective function. It is shown that the energy of the final configuration gradually moves closer to the origin with each generation. Also, it is observed that exploration is controlled adaptively by the evolution algorithm based on the distance from the desired solution. The exploration decreases as we get closer to the minimum.

Table 4 represents the data for the evolution process in terms of the initial and final generation of evolution.

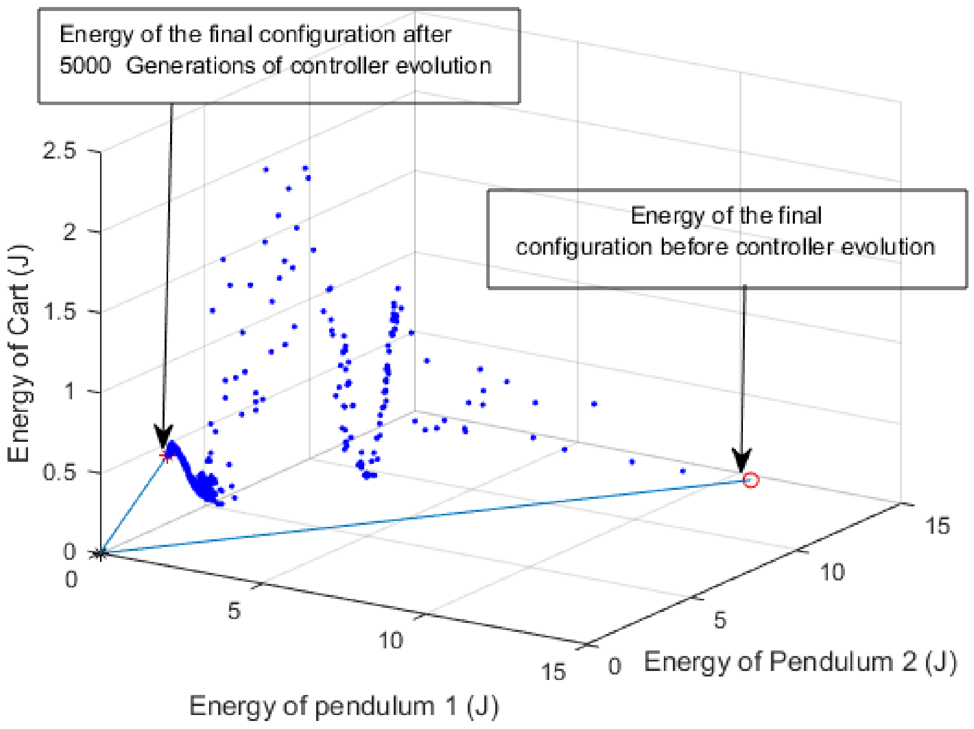

FIGURE 43. Visualization of the controller evolution.

TABLE 4. Generation Data Showing Controller Evolution

Objective Function	**Value of Objective Function at the First Generation**	**Value of Objective Function at final Generation**	**No. of Generations**
f_1	214	3.6	5000

Observations

As mentioned in the methodology above, the choice of objective function is important for the performance of the evolution algorithm as shown by the generation data given in Table 3.

It was found that squared energy based objective functions gave good results as compared to the simple energy based function from data in Table 3. In Figure 44, squared energy-like objective functions f_1 and f_2 that have the same units are compared. This shows that any of the energy-based Lyapunov-like functions can be used as the objective function to obtain comparable results.

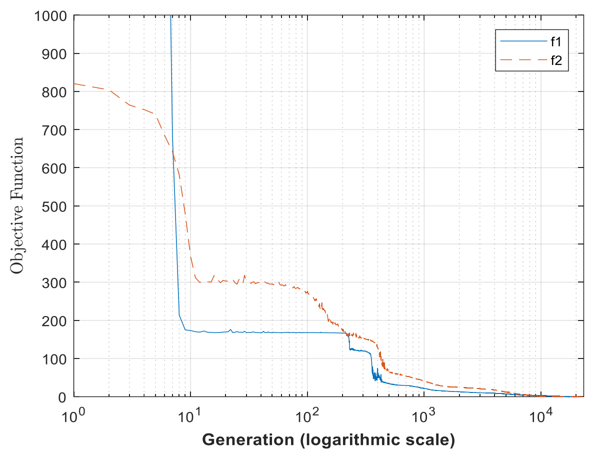

FIGURE 44. Comparison of evolution rate with objective functions f_1 and f_2 (in $Joule^2$). x-axis is in logarithmic scale.

The choice of DMP parameters is arbitrary. It was found that for different parameter values of the double inverted pendulum, the appropriate choice of DMP system parameters will give better results. From the experiments, for a reasonable choice of DMP system corresponding to the parameters of the double inverted pendulum, there is always a reduction in energy, showing that a theoretical inspection in future might provide more insight into evolutionary controllers of the form proposed in this thesis. The observations were made from experiments performed with three different sets of constant parameters for the double inverted pendulum and the DMP system. The first parameter set P_1 is described in Table 5. Table 6 shows the data corresponding to the evolution rate with this parameter set. Tables 7 describes the second parameter set P_2. The data corresponding to the evolution rate with P_2 is given in Table 8. Table 9 gives the description for the third parameter set P_3. Table 10 gives the data corresponding to evolution rate with P_3. The results are visualized in the evolution rates shown in Figures 45, 46, and 47. The objective function f_1 was used in all cases. The results hint at a model-free approach for control without the need of estimation of system parameters.

Obviously, the choice of evolution algorithm is vital for the development of an effective controller. This fact is elucidated by Figure 48 where the evolution rates for 10,000 generations using two different learning algorithms, the Particle Swarm Optimization (PSO) and the CMA-ES algorithm, were compared. Both used the same parameter set P_1 for the double inverted pendulum and DMP system. Also, the same objective function f_1 was used for both cases.

TABLE 5. Parameter Set P_1

Parameters of Inverted Pendulum and DMP	**Symbols**	**Values Chosen**
Mass of Cart (pivot point)	m_1	1 kg
Mass of each pendulum	m_2	0.3 kg
Acceleration due to Gravity	g	9.8 m/s^2
Length of First pendulum	l_1	1 m
Length of Second pendulum	l_2	2 m
Natural Frequency	ω_0	5 rad/s
Damping Ratio	ζ	1
Initial Deviation from relaxed length of the spring	x_0	1m
Relaxed length of the spring	x_g	0 m
Initial velocity	$\dot{x}_0$	0 m/s
Number of basis functions	N	25

TABLE 6. Evolution Data for Parameter Set P_1

Objective Function	**Value of Objective Function at the First Generation**	**Value of Objective Function at final Generation**	**No. of Generations**
f_1	7.0441×10^4	0.2173	12015

TABLE 7. Parameter Set P_2

Parameters of Inverted Pendulum and DMP	**Symbols**	**Values Chosen**
Mass of Cart (pivot point)	m_1	$1\ kg$
Mass of each pendulum	m_2	$0.3\ kg$
Acceleration due to Gravity	g	$9.8\ m/s^2$
Length of First pendulum	l_1	$1\ m$
Length of Second pendulum	l_2	$1.5\ m$
Natural Frequency	ω_0	$5\ rad/s$
Damping Ratio	ζ	1
Initial Deviation from relaxed length of the spring	x_0	$2\ m$
Relaxed length of the spring	x_g	$0\ m$
Initial velocity	$\dot{x}_0$	$0\ m/s$
Number of basis functions	N	25

TABLE 8. Evolution Data for Parameter Set P_2

Objective Function	**Value of Objective Function at the First Generation**	**Value of Objective Function at final Generation**	**No. of Generations**
f_1	2.9704×10^5	0.5545	22481

TABLE 9. Parameter Set P_3

Parameters of Inverted Pendulum and DMP	**Symbols**	**Values Chosen**
Mass of Cart (pivot point)	m_1	$1\ kg$
Mass of each pendulum	m_2	$0.2\ kg$
Acceleration due to Gravity	g	$9.8\ m/s^2$
Length of First pendulum	l_1	$0.5\ m$
Length of Second pendulum	l_2	$1m$
Natural Frequency	ω_0	$5\ rad/s$
Damping Ratio	ζ	1
Initial Deviation from relaxed length of the spring	x_0	$0.5\ m$
Relaxed length of the spring	x_g	$0\ m$
Initial velocity	$\dot{x}_0$	$0\ m/s$
Number of basis functions	N	25

TABLE 10. Evolution Data for Parameter Set P_3

Objective Function	Value of Objective Function at the First Generation	Value of Objective Function at final Generation	No. of Generations
f_1	9.7233×10^3	0.8572	9821

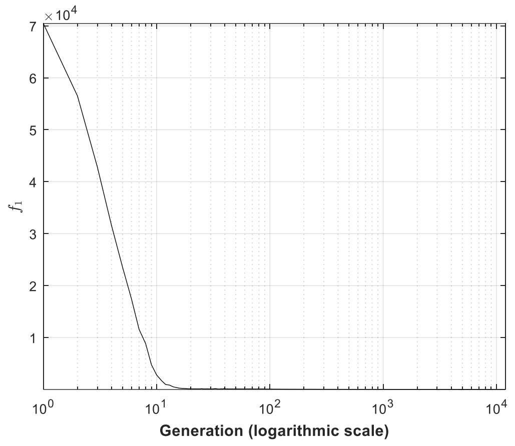

FIGURE 45. Evolution rate with parameter set P_1.

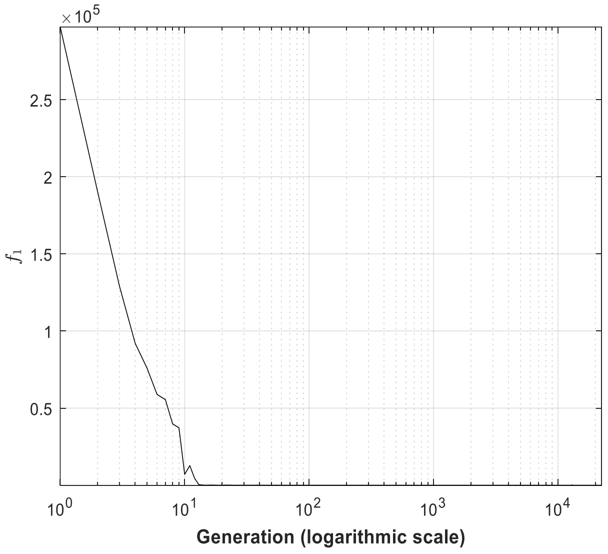

FIGURE 46. Evolution rate with parameter set P_2.

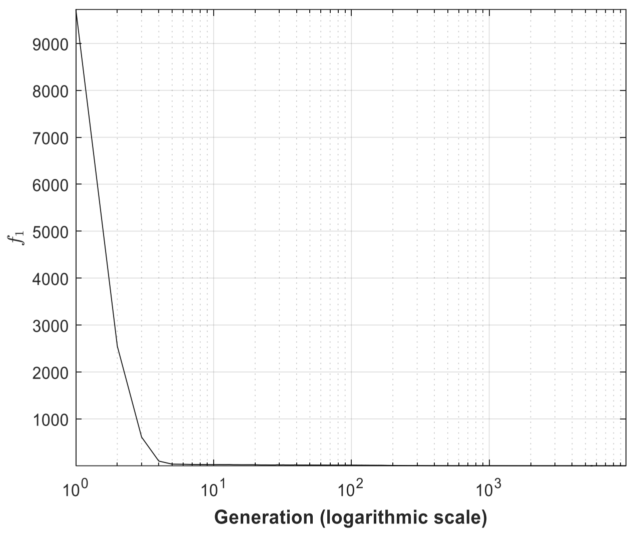

FIGURE 47. Evolution rate with parameter set P_3.

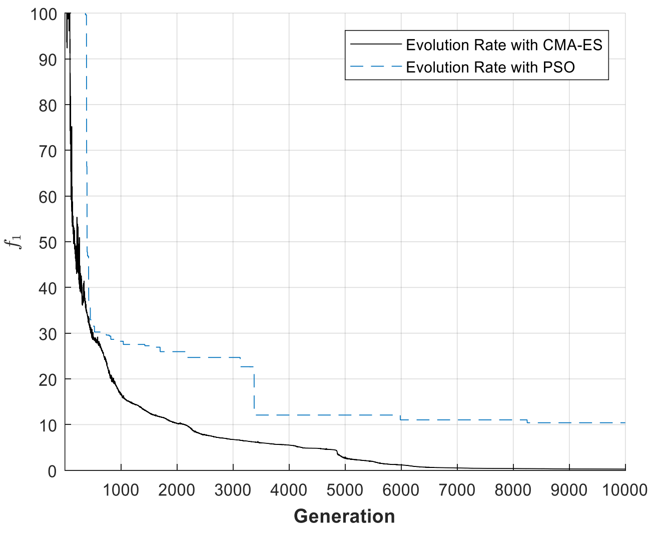

FIGURE 48. Comparison between evolution rates with CMA-ES and PSO.

The control of a six-dimensional system was achieved just by using controller on a single degree of freedom. The function approximator in the Dynamic Movement Primitive (DMP) system, shown in Equation (8) in Chapter 3, could structure a time-varying force that was able to make the experiment successful for the chosen set of parameters.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

This thesis examines the effectiveness of the CMA-ES evolutionary algorithm for learning a model-free controller.

An artificial mass-spring-damper system with parameterized time-varying applied force that is described as Dynamic Movement Primitive (DMP) in literature was used to define the structure of the controller. The evolution algorithm used was the Covariance Matrix Adaption Evolution Strategy (CMA-ES).

The model-free controller was used for the swing-up task of a double inverted pendulum in a computer simulated experiment. The objective of the experiment was to swing- up the two pendulums as close to the upright position as possible from the pendant position just by controlling the pivot point (or cart). For the application of the evolution algorithm, energy-based objective functions were chosen. The justification of this choice is twofold. Firstly, energy-based Lyapunov functions are a popular choice for the design of swing-up controller. Secondly, energy of the double inverted pendulum at the final state of the dynamics provides a good measure of the final state of the pendulum and so could be used to describe the effectiveness of the controller.

The experiment was successful for the chosen DMP values and objective functions. The evolutionary controller was analyzed and was found to be effective.

This study seems to indicate that with further research into the theoretical framework, evolutionary algorithms could be used effectively to learn model-free controllers. It also shows that a framework for the application of advanced machine learning techniques to the control of nonlinear dynamical systems is possible and warrants further research.

REFERENCES

REFERENCES

[1] L. H. Keel and S. P. Bhattacharyya, “Controller synthesis free of analytical models: three term controllers,” *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1353-1369, 2008.

[2] J. T. H Chan, “Data based synthesis of a multivariable linear quadratic regulator,” *Automatica*, vol.32, no. 3, pp. 403-407, 1996.

[3] J. Peters and S. Schaal, “Learning operational space control,” paper presented at Robotics: Science and Systems (RSS), Philadelphia, PA, 2006.

[4] A. Ijspeert and S. Schaal, “Learning attractor landscapes for learning motor primitives,” in *Advances in Neural Information Processing Systems 15 (NIPS 2003)*. Cambridge, MA: MIT Press, 2003, pp. 1547-1554.

[5] N. Hansen and A. Ostermeier, “Completely derandomized self-adaption in evolution strategies,” *Evolutionary Computation*, vol. 9, no.2, pp.159-195, 2001.

[6] M. Spong, “Partial feedback linearization of underactuated mechanical systems,” in *Proceedings of International Conference on Intelligent Robots and Systems*, Munich, Germany, 1994, pp. 314–321.

[7] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Upper Saddle River, NJ: Prentice Hall, 1991.

[8] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York, NY: John Wiley and Sons, 2006.

[9] F. L. Lewis, S. Jagannathan, and A. Yesidirek, *Neural-network Control of Robot Manipulators and Nonlinear Systems.* Philadelphia, PA: Taylor and Francis, 1999.

[10] R. Sutton and A. Barto, *Reinforcement Learning*. Boston, MA: MIT Press, 1998.

[11] K. Doya, “Reinforcement leaning in continuous time and space,” *Neural Computation,* vol.01, no.12, pp. 243-269, 1999.

[12] J. Peters and S. Schaal, “Policy gradient methods for robotics,” in *Proceedings of the IEEE/RSJ2006, International Conference on Intelligent Robots and Systems (IROS),* Beijing, China, 2006, pp.2219-2225.

[13] J. Peters and S. Schaal, “Natural actor critic,” *Neurocomputing*, vol.71, pp.1180-1190, 2008.

[14] S. Amari, “Natural gradient works efficiently in learning,” *Neural Computation*, vol.10, pp. 251-276, 1998.

[15] J. Koeber and J. Peters,“Policy search for motor primitives”, in *Advances in Neural Information Processing Systems 21(NIPS 2008)*. Cambridge, MA: MIT Press, 2008, pp. 297–304.

[16] E. Theodorou, J. Buchli, and S. Schaal, “A generalized path integral control approach to reinforcement learning,” *Journal of Machine Learning Research*, vol.11, pp.3137-3181, 2010.

[17] D. E. Moriarty, A.C. Schultz, and J. J. Grefenstette, “Evolutionary algorithms for reinforcement learning,” *Journal of Artificial Intelligence (JAIR),* vol.11, pp. 241-276, 1999.

[18] C. Darwin, *The Origin of Species*. London: John Murray,1859.

[19] J. H. Holland, *Adaption in Natural and Artificial Systems*, Ann Arbor, MI: The University of Michigan Press, 1975.

[20] J. C. Spall, “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation,” *IEEE Transactions on Automatic Control*, vol.37, no.3, pp. 332-341, 1992.

[21] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in *Proceedings of IEEE International Conference on Neural-networks,* Piscataway, NJ, 1995, vol.4, pp. 1942-1948.

[22] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, “Genetic reinforcement learning for neurocontrol problems,” *Machine Learning*, vol.13, pp.259–284,1993.

[23] F. Stulp and O. Sigaud, “Path integral policy improvement with covariance matrix adaptation,” in *Proceedings of the 29th International Conference on Machine Learning* (ICML), 2012, pp.281-288.

[24] V. Heidrich-Meisnerm, and C. Igel, “Evolution strategies for direct policy search,” in *Proc. of the International Conference on Parallel Problem Solving from Nature*, 2008, pp.428-437.

[25] J. A. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA).* Washington, DC, 2002, vol.2, pp.1398-1403.

[26] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, “Learning to grasp under uncertainty,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 5703-5708.

[27] F. Stulp and S. Schaal, “Hierarchical reinforcement learning with movement primitives,” in *Proceedings of 11th IEEE-RAS International Conference on Humanoid Robots*, 2011, pp.231-238.

[28] D.H. Park, H. Hoffmann, P. Pastor, and S. Schaal, “Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields,” in *Proceedings of 8th IEEE-RAS International Conference on Humanoid Robots*, 2008, pp.91-98.

[29] N. Hansen and A. Ostermeier, “Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The (μ/μI , λ)-CMA-ES,” in *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, 1997, pp. 650-654.

[30] D. E. Moriarty and R. Miikkulainen, “Efficient reinforcement learning through symbiotic evolution,” *Machine Learning*, vol.22, pp.11-32,1996.

[31] C. Igel, “Neuroevolution for reinforcement learning using evolution strategies,” in *Congress on Evolutionary Computation (CEC)*, 2003, vol. 4, pp. 2588-2595.

[32] F. Hamano, “Review of classical mechanics,” Lecture notes for EE474/574, Department of Electrical Engineering, CSULB, 2015.

[33] K. Furuta, T. Okutani, and H. Sone, “Computer control of a double inverted pendulum,” *Computer and Electrical Engineering*, vol. 5, no. I, pp. 67-84, 1978.

[34] F. Cheng, G. Zhong, Y. Li, and Z. Xu, "Fuzzy control of a double inverted pendulum,” *Fuzzy Sets and Systems*, vol. 79, pp. 315-321, 1996.

[35] K. J. Åström and K. Furuta, “Swinging up a pendulum by energy control,” in *Proceedings of 13th IFAC World Congress*, San Francisco, USA, 1996, pp.37-42.

[36] M. Yamakita, M. Iwasaki, Y. Sugahara, and K. Furuta, “Robust swing up control of double pendulum,” in *Proceedings of the American Control Conference*, 1995, pp.290-295.