SABANCI UNIVERSITY

Markov Chain Monte Carlo Algorithm for Bayesian Policy Search

by

Vahid Tavakol Aghaei

Supervisor:
Ahmet Onat, Sinan Yıldırım

Submitted to the Graduate School of Engineering and Natural Sciences in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
Mechatronics Engineering

September 2019

MARKOV CHAIN MONTE CARLO ALGORITHM FOR BAYESIAN POLICY SEARCH

APPROVED BY:

Assoc. Prof. Dr. Ahmet Onat

Prof. Dr. Kürşat Şendur

Asst. Prof. Dr. Öznur Taştan

Prof. Dr. İlker Birbil

Prof. Dr. Taylan Cemgil

DATE OF APPROVAL: 24/06/2019

ABSTRACT

Markov Chain Monte Carlo Algorithm for Bayesian Policy Search

Vahid Tavakol Aghaei

Mechatronics Engineering, PhD Dissertation, August 2019

Keywords: Reinforcement Learning; Markov Chain Monte Carlo; Particle filtering; Risk sensitive reward; Policy search; Control

The fundamental intention in Reinforcement Learning (RL) is to seek for optimal parameters of a given parameterized policy. Policy search algorithms have paved the way for making the RL suitable for applying to complex dynamical systems, such as robotics domain, where the environment comprised of high-dimensional state and action spaces. Although many policy search techniques are based on the wide spread policy gradient methods, thanks to their appropriateness to such complex environments, their performance might be affected by slow convergence or local optima complications. The reason for this is due to the urge for computation of the gradient components of the parameterized policy. In this study, we avail a Bayesian approach for policy search problem pertinent to the RL framework, The problem of interest is to control a discrete time Markov decision process (MDP) with continuous state and action spaces. We contribute to the field by propounding a Particle Markov Chain Monte Carlo (P-MCMC) algorithm as a method of generating samples for the policy parameters from a posterior distribution, instead of performing gradient approximations. To do so, we adopt a prior density over policy parameters and aim for the posterior distribution where the 'likelihood' is assumed to be the expected total reward. In terms of risk-sensitive scenarios, where a multiplicative expected total reward is employed to measure the performance of the policy, rather than its cumulative counterpart, our methodology is fit for purpose owing to the fact that by utilizing a reward function in a multiplicative form, one can fully take sequential Monte Carlo (SMC), known as the particle filter within the iterations of the P-MCMC. it is worth mentioning that these methods have widely been used in statistical and engineering applications in recent years. Furthermore, in order to

deal with the challenging problem of the policy search in large-dimensional state spaces an Adaptive MCMC algorithm will be proposed.

This research is organized as follows: In Chapter 1, we commence with a general introduction and motivation to the current work and highlight the topics that are going to be covered. In Chapter 2ö a literature review pursuant to the context of the thesis will be conducted. In Chapter 3, a brief review of some popular policy gradient based RL methods is provided. We proceed with Bayesian inference notion and present Markov Chain Monte Carlo methods in Chapter 4. The original work of the thesis is formulated in this chapter where a novel SMC algorithm for policy search in RL setting is advocated. In order to exhibit the fruitfulness of the proposed algorithm in learning a parameterized policy, numerical simulations are incorporated in Chapter 5. To validate the applicability of the proposed method in real-time it will be implemented on a control problem of a physical setup of a two degree of freedom (2-DoF) robotic manipulator where its corresponding results appear in Chapter 6. Finally, concluding remarks and future work are expressed in chapter 7.

ÖZET

Bayes Politika Arama için Markov Zinciri Monte Carlo Algoritması

Vahid Tavakol Aghaei

Mekatronik Mühendisliği, Doktora Tezi, Ağustos 2019

Anahtar Kelimeler: Takviyeli öğrenme, Markov zinciri Monte Carlo, Parçacık filtre, Riske duyarlı ödül, Politika araması, kontrol

Takviye Öğrenimindeki temel amaç, belirli bir parametrelenmiş kontrol politikanın en uygun parametrelerini aramaktır. Politika arama algoritmaları, ortamın yüksek boyutlu durum ve eylem alanlarından oluştuğu robotik alan gibi karmaşık dinamik sistemlere uygulanmaya uygun hale getirmenin yolunu açmıştır. Birçok politika arama tekniği geniş çaplı politika gradyan yöntemlerine dayanmasına rağmen, bu tür karmaşık ortamlara uygun olmaları nedeniyle performansları yavaş yakınsama veya yerel optima komplikasyonlarından etkilenebilir. Bunun nedeni, parametreleştirilmiş politikanın gradyan bileşenlerinin hesaplanma dürtüsünden kaynaklanmaktadır. Bu çalışmada, Takviye Öğrenme çerçevesine uygun politika arama problemi için bir Bayesian yaklaşımı elde ettik. İlgilendiğimiz konu, sürekli durum ve eylem alanları ile ayrık zaman bir Markov karar sürecini (MDP) kontrol etmektir. Gradyan yaklaştırmaları yerine, bir Posterior Dağılımından politika parametreleri için numune üretme yöntemi olarak bir Parçacık Markov Zinciri Monte Carlo (P-MCMC) algoritması geliştirerek bu alana katkıda bulunuyoruz. Bunu yapmak için, politika parametreleri üzerinde önceden bir yoğunluğu benimsiyoruz ve 'olasılığın' beklenen toplam ödül olduğu varsayılan posterior dağıtımı hedefliyoruz. Politikanın kümülatif muadili yerine performansını ölçmek için çoklayıcı beklenen toplam bir ödülün kullanıldığı riske duyarlı senaryolar açısından, metodolojimiz bir ödül fonksiyonunu çarpımcı bir formda kullanmaktan dolayı amaca uygundur. P-MCMC'nin yinelemelerinde parçacık filtresi olarak bilinen sıralı Monte Carlo'yu (SMC) tamamen kullanılabilir. Bu yöntemlerin son yıllarda istatistik ve mühendislik uygulamalarında yaygın olarak kullanıldığını belirtmekte fayda var. Ayrıca, politika araştırmasının bir başka zorlayıcı sorununu büyük boyutlu uzaylarda ele almak için, bir Uyarlamalı MCMC algoritması önerilecektir.

To my mother, Pari and my nephew, Alpay ...

Acknowledgements

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Ahmet Onat for his patience, support, assistance and friendship. He was not only a supervisor, but also a father for me. I am very glad to be his student during these years and have benefited from his immense knowledge.

I would also like to express my sincere gratitude to my adviser Dr. Sinan Yıldırım who was very supportive and patiently guided me to complete this thesis. I am delighted for working under his supervision.

Many thanks to Prof. Volkan Patoğlu who provided us the experimental setup for our real-time evaluations.

I would like to thank my fellow labmates Dr. Mustafa Yalçın, Arda Ağababaoğlu, Umut Çalışkan, Ali Khalilian, Özge Orhan and Fatih Emre Tosun for the friendly atmosphere that they created for me during the stressful period of my Phd career.

Besides thanks to all my friends especially Dr. Morteza Ghorbani, Sina Rastani, Amin Ahmadi, Siamak Naderi, Araz Sheibani, Reza Pakdaman, Ali Asgharpour, Sahand Faraji, Sonia Javadi, Kaveh Rahimzadeh, Faraz Tehranizadeh and Nasim Barzegar, I owe a debt of gratitude to Nasim Tavakkoli who shared all the happiness and excitement with me.

I must thank my friends Hamed Taham, Hessam Jafarpour, Mohammad Hossein Eskandani, Mohammad Mousavi, Nasser Arghavani and Dr. Reza Vafaei for providing the best moments and the support whenever I need them.

Finally, I greatly appreciate the patience and the endless support which my family and my parents Hamid and Pari showed to me throughout my life, especially my dear brothers Mehdi and Hossein.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Without any doubt, reinforcement learning (RL) can be recognized as the most propitious framework for the experts in the machine learning, control and robotics community; see Sutton and Barto (1998) for an introduction. In RL problem, an agent interacts sequentially and autonomously with an unknown environment to collect some data samples called trajectories or rollouts. It then utilizes the generated data to search for a policy; mapping from states to actions; that maximizes a performance criterion i.e, an expected total reward (objective function), in the long run. Most of the methods available in the literature are concerned with providing online policy parameter estimates. The fundamental concept of these paradigms is that, an ongoing approximation of the parameters acquired using the available data sets, could be updated when a new collection of data is received. The choice of a policy plays a crucial role in data collection part. Different policies result in different data collection patterns, which, in turn, affect how the policy is updated. Therefore, effective and precise parameter estimation methods for a policy are of significant importance specially in real time applications for example when commanding robotic

tasks, an unfavorable deviation in the policy parameters can cause disastrous outcomes. Policy search can be done either in a model-free or model-based fashion. The former case focuses on generating samples directly from the real robot or simulation platform where there is no model of the system at hand and agent learns an optimal control policy from these collected data samples. In the latter agent attempts to construct a model of the system's dynamics and subsequently employs the obtained model to learn the policy. Among the existing policy search RL methods, closer attention is paid to the gradient based algorithms where improvements in the policy parameters are pursuant to the gradient ascent approach that is followed by the gradient of the expected total reward with a predefined learning step size. These methods have been shown to be successful in dealing with high-dimensional continuous state spaces. Since such complex large scale environments are inherent in the robotic domain policy gradient (PG) methods are attracting widespread interest among the researchers in the context of robotics. Despite the fact that the PG techniques bear numerous advantages, it should be noted that they are prone to some weaknesses, as well. The challenges that one may encounter when using these methods are quality of the estimated gradient of the objective function, furthermore, scaling the learning rate. A major drawback of these algorithms is that a local search of the policy space is performed during the learning process which may lead to either being trapped in a local optimal point or poor convergence speed. A different approach to cope with these problems is a Bayesian inference method. Bayesian parameter estimators demand that a prior distribution for the unknown policy parameter is assigned. The target is then to determine the posterior distribution of the policy parameter given the observed data. Moreover it is possible to provide some characteristics of this posterior distribution in order to produce a point estimate of the parameter. The posterior mean, posterior median and the maximum a-posteriori probability (MAP) estimators can be pointed out as some popular Bayesian estimators in the field. Besides the aforementioned approximators, there also exist Monte Carlo based methods for Bayesian parameter inference when an

accurate evaluation of the posterior distribution is not feasible. As an alternative approach, the maximum likelihood estimation (MLE) method incorporates the likelihood of the sampled data to include all the appropriate information for calculating policy parameter. To summarize, Bayesian inference with roots in statistics, takes into account a-priori knowledge in the form of a probabilistic distribution of the past experience of the agent in interacting with the environment and incorporates it in the learning procedure by modeling the distribution over policy parameters.

1.2 Contribution

The major contribution of the present study is to perform Bayesian inference for the policy search method in RL by using a Markov chain Monte Carlo (MCMC) algorithm. Specifically, our algorithm is a particle MCMC algorithm (P-MCMC), involving a Sequential Monte Carlo (SMC), also known as particle filters, within its learning iterations. SMC methods are special cases of Monte Carlo algorithms in which the idea behind just depends on sampling from complex distributions, whenever an analytic computation can not be carried out. The novelty of our approach is due to a formulation of the policy search problem in a Bayesian framework where the expected total reward, $J(\theta)$, is formed by the product of exponential rewards and is treated as a pseudo-likelihood function. We propose the multiplicative formulation as the notion of *risk-sensitivity* in the structure of the reward function to use the SMC algorithm in the proposed method. Combined with an uninformative prior, $\mu(\theta)$, this leads to a pseudo-posterior distribution for the policy parameter $\pi(\theta)$.

$$\pi(\theta) \propto \mu(\theta) J(\theta).$$

This pseudo-posterior distribution can be utilized to identify promising regions for the policy parameter. Our dominant observations that follow the Bayesian formulation above are:

- Unbiased estimate of the expected total reward for a given set of policy parameters via SMC is possible
- It can be used within an MCMC algorithm that targets $\pi(\theta)$.

With regard to handling the drawbacks of gradient based methods, four main claims about our proposed method can be expressed:

1. The presented approach in this thesis is not based on estimating the gradient information of the expected total reward, instead it aims to employ the estimates of the expected multiplicative reward via MCMC algorithm. Its structure is very straightforward to implement and therefore, reduces the computation load by omitting the need for gradient calculations.
2. The proposed method is less likely to get stuck around an optimum solution point since it does not produce a point estimate of the parameters. Instead, it generates samples from $\pi(\theta)$ which can then be used to explore the surface of $J(\theta)$. In particular, those samples can be applied to identify favorable regions for the policy parameters, consequently keeping the agent away from getting trapped by some data samples which might distract the learning from its main objective.
3. Our proposed algorithm can have comparable, if not better, convergence properties in terms of computation time.
4. It can be dedicated to both robotic control problems and statistical Bayesian learning domains in which the proposed method can simultaneously learn and

predict the model and control gains in uncertain environments with high precision and facilitates the procedure of adapting the controller to changing conditions by using data sampling techniques. By explicitly putting prior distributions on unknown policy parameters, Bayesian methods provide a promising technique for handling parameter uncertainty.

Therefore, the scope of this research is to use the proposed MCMC methodology in policy search problems as an alternative to gradient-based methods. The claims on robustness and convergence time are supported with our numerical experiments throughout the thesis where we compare our proposed method with three state-of-the-art gradient based methods.

1.3 Outline

This thesis focuses on the parameter estimation problem for stochastic parameterized policies in the robotics and control domains in an offline manner using data driven methods. The main contents of this thesis is divided into six chapters laid out as follows:

Chapter 2: Literature Review

A literature review of the available studies relevant to the scope of the thesis is presented. It covers the research done in the area of Reinforcement Learning and Bayesian inference. Furthermore, it touches upon the recent works which ignited the spark to determine the approach of the thesis.

Chapter 3: Reinforcement Learning

The RL problem is introduced and a review of some well established policy gradient based RL methods such as REINFOCE, GPOMDP and episodic Natural Actor Critic (eNAC) algorithms is presented. We cast the idea from the robotics domain to the fuzzy control and describe the usage of these algorithms in tuning the control gains of a general Proportional-Derivative (PD) fuzzy controller. The results of this chapter led to the publication of one conference paper which are presented in section 5.1.

Chapter 4: Markov Chain Monte Carlo Methods

The main contributions of this thesis are highlighted here. In the first part, a novel gradient-free algorithm, which is based on the Bayesian RL framework with MCMC algorithm for the policy search problem in continuous MDPs, is proposed. To capture the essence, some MCMC methods are reviewed.

In the second part, to enhance the capability of the proposed MCMC algorithm and make it well suited for high-dimensional state spaces, where the number of unknown parameters to be learned are increasing with each dimension, an adaptive MCMC algorithm is proposed. This chapter contributed to the publication of a journal article and a conference paper which their outcomes are incorporated in section 5.2 and section 5.3. Not to mention that another paper is currently under preparation regarding the proposed adaptive MCMC algorithm and its initial results are presented in section 5.4.

Chapter 5: Numerical experiments and Simulations

This chapter is dedicated to the numerical simulations of the methods described in chapters 3 and 4 where learning control policies for three different nonlinear systems are studied. The chapter is split into four sections: In the first part, PG RL methods will be used to tune the control gains of a PD-type Fuzzy Logic Controller (FLC). The next section manifests the benefits of our proposed MCMC algorithm over PG RL ones. An extensive comparison is made through the control of a nonlinear model of an Inverted Pendulum. Further, the trajectory control of a planar manipulator is taken up as a more complex, nonlinear control problem in the robotics domain. The chapter is concluded by the extension of the proposed MCMC algorithm to an adaptive form and is applied to the control problem of a Ballbot model.

Chapter 6: Experimental results

In order to validate the proposed MCMC algorithm in the real-time applications we furnish our theoretical accomplishments by a real-time implementation. For this purpose, a modified formation of the MCMC algorithm has been derived resulting in an MCMC algorithm which does not include the SMC, anymore. This algorithm is tested on a physical setup of a 2-DoF planar manipulator with the goal of trajectory tracking. This chapter has been prepared as a journal manuscript paper and is ready for submission.

Chapter 7: Conclusions

The overall concept of the thesis is summarized and our work is concluded by supplying final observations. A discussion has been provided to express the open research problems that can be contemplated as our future work both in theory and practice.

Chapter 2

Literature Review

Summary: In this chapter we will have an overview to the available literature related to the research domain of this thesis. It comprises both the available papers in the domain of RL and Bayesian inference methods in policy search.

2.1 Reinforcement Learning

Reinforcement learning (RL) initiated in the primary work of Sutton and Barto (1998), is a significantly auspicious learning mechanism specially in the robotics branch where an agent (controller) interacts with its environment in order to obtain an optimal policy (action selection scheme). The attempt to get the optimal policy is carried out with respect to a cost function so that the principal goal of it is to optimize the performance measure over a long period of time. In general, RL problems include value function, policy search and actor-critic methods. In the value function RL the agent tries to get an optimal policy by first assigning a value to the action that is resulted in moving to a new state then picks the action that maximizes the value function. These methods are not usually well qualified for

discrete state spaces and thus demand some function approximators to map the discrete states to continuous ones as done by Gu et al. (2016) where a normalized advantage function (NAF) is used to obtain the maximum value of the Q-function analytically. In order to alleviate the need for estimating or learning the value function, policy search methods are used. These methods, which use parameterized policy, depend on maximizing the expected cumulative rewards. It is common to use a Gaussian probability distribution for the parameterized policy when dealing with continuous state spaces in which its mean and standard deviation can be considered as the parameters. A detailed review to these methods can be found in Kober et al. and Deisenroth et al. (2013). Benefiting from parameterized policies, have facilitated employment of RL to dynamical systems, such as control of robots as studied by Levine et al. (2016). Policy search methods can generally be divided into gradient-free and gradient-based methods (known as policy gradient). The former are usually suitable for low-dimensional spaces while its successful extensions for high-dimensional spaces are found such as Koutník et al. (2013) where a compressed large-scale network search for the optimal policy is performed. Evolutionary search algorithms have also been used as gradient-free methods in policy search problems as did by Salimans et al. (2017). Policy gradient methods are effectively used in high-dimensional state spaces. The works carried out by Peters and Schaal (2006) and Peters and Schaal (2007) have shown their functionality in the robotics domain. Among the existing RL approaches, a large portion of the policy search methods has been dedicated to PG ones which captivated an enormous interest between the researchers in the field due to their efficient applications. Despite this PGs are guaranteed to converge to a local optimal policy. However, estimated gradients suffer from high variance and this makes the convergence of the PG policy search methods slow. Most recently, Pajarinen et al. (2019) have proposed to use Kullback-Leibler (KL)-divergence and entropy bounds to update the natural gradients in policy search problem. Opposed to Kober and Peters (2008) where policy gradients are approximated for policy update step, Deisenroth and Rasmussen (2011) proposed

a policy search method called PILCO where the transition model of the system is modeled by Gaussian processes and policy improvement is done by analytically calculating policy gradients. To estimate the gradients in PG methods, a recent paper by Ciosek and Whiteson (2018) suggest summing over the chosen actions by the stochastic policy rather than using the action selected during the sampled trajectory.

Policy search methods which does not explicitly depend on a model of a system are called model-free approaches where the required stochastic trajectories are provided by drawing state action samples from the robot. In the model-based scenario, instead of using real robots, simulation environments are hired and the learned model dynamics are used for observing samples to create robot paths. A good example for this case is done by Tangkaratt et al. (2014) where first a state space model of the system is learned by using least square estimation method and then the policy is obtained by policy gradients with parameter-based exploration method (PGPE) which is already proposed by Sehnke et al. (2010). For an extensive study regarding the model-based policy search please refer to Polydoros and Nalpantidis (2017). Although working with simulations are easy in comparison to real robots, learning a forward model of a system is challenging than learning a policy mapping. On the other side, working with real robots is challenging due to the iterative interactions that can result in probable damages that may occur to the robot.

The third category unifies the advantages of the the value function based and policy search methods where the parameterized policy plays the role of an actor and the critic is considered to be the learned value function. The parameterized policy (here called actor) is advantageous since it can cope with continuous state action spaces without any need for function approximators. On the other side, the critic has the ability to calculate lower variance gradient estimates of the expected total rewards for the actor. This property makes them superior over the other classes by speeding up the convergence. A class of actor-critic methods with natural gradients

can be found in Peters and Schaal (2008) and Schulman et al. (2015) where the value of the critic is used to learn the actor. A comprehensive survey about the actor-critic methods can be found in Grondman et al. (2012).

In recent years, as opposed to classical RL, Deep RL (DRL) algorithms are introduced. For instance, the prominent and pioneering ones preceded in the work done by Mnih et al. (2015) which extends the Q-learning algorithm to deep neural networks. Tangkaratt et al. (2018) propose a unique actor-critic method called guided actor-critic (GAC) and claim that the deterministic policy gradient (DPG) is actually a special case of their algorithm. Another formulation of the DPG, is suggested by Lillicrap et al. (2015) as deep deterministic PG (DDPG), which relies on the PG methods. For learning robust control policies a category of model-based DRL methods are used by Finn et al. (2016) and Tzeng et al. (2015). The extension of the actor-critic methods in the domain of DRL can be found in Wu et al. (2017).

2.2 Bayesian Inference and Markov Chain Monte Carlo Methods

Bayesian optimization (BO) concept, which is dealt with completely in Brochu et al. (2010), is considered as a useful tool for learning in an uncertain environment while making decisions. It takes unknown parameters as random variables and assumes some distributions over them to explicitly incorporate uncertainty. This uninformative prior information over the parameters quantify the uncertainty to balance the exploration-exploitation tradeoff. Although the primary goal in RL is to opt for actions which makes the future rewards maximum according to the available estimates of the model (exploitation), exploring the areas of the parameter space to get possible high rewards are also inevitable. Bayesian inference for RL problems are classified to model-free and model-based ones where for the former prior distribution

is maintained over the parameters of the policy (or value function) whereas for the latter the prior is taken over parameters of the state transition or reward function, as discussed in Ghavamzadeh et al. (2015). The early examples of the Bayesian model-free method can be found in Dearden et al. (1998) where they used prior distribution for the Q-values in choosing the suitable action for discrete states. To extend the idea to the continuous problems Engel et al. (2005) took Gaussian processes to model Q-functions. Unlike the PG algorithms in RL which natural gradient methods have been used to estimate the gradients in policy improvement step Ghavamzadeh and Engel (2006) substituted the natural gradient approaches with a Bayesian structure and modeled the policy gradients with Gaussian processes. They use a Bayesian PG algorithm to estimate the posterior mean of the gradient of the expected return.

For the model-based case an explicit model of the system dynamics and the structure of the reward function can be learned. For instance, Wilson et al. (2014) put GPs on the model dynamics and attempt to learn their approximate mean function leading in learning the dynamical model of the system and the reward structure. Their approach was somehow limited to low dimensional state spaces. In order to develop it and scale to high-dimensional states instead of using GPs, Gal and Ghahramani (2016) and Higuera et al. (2018) propose to use Bayesian Neural Networks (NN). A sample work which uses BO for modeling the expected reward has been presented by Marco et al. (2017). They use a Bayesian method based on entropy search to target parameters which maximally decrease the uncertainty about the location of the minimum of the expected return. In order to speed up the learning procedure in BO one solution is to take advantage of the prior information. To reach this goal, Pautrat et al. (2018) leverage the existing multiple prior information about the expected reward by proposing a method called Most Likely Expected Improvement (MLEI).

All the above mentioned works in the Bayesian RL domain have concentrated on putting prior distributions either over the system dynamics model or expected

return and value function. In contrast to them we will attain priors over the policy parameters and considering the stochastic optimization tacked in Hoffman et al. (2008) to cast the policy search problem to the Bayesian inference. Instead of targeting the expected return, we define a posterior distribution which is proportional to the expected return and try to sample the policy parameters form this posterior distribution by using the MCMC algorithms covered in Andrieu et al. (2003) and Cemgil (2013). Similar to our work Wilson et al. (2010) have proposed putting priors over the policy parameters but they have used a hybrid MCMC method based on importance sampling which benefits from the gradient information approximated via trajectories. Prior to them Hoffman et al. (2008) used the same idea as an alternative to policy search using expectation maximization. Subsequently, they made some modifications to their approach for dealing with general MDPs by using reversible jump MCMC algorithm; see Fan and Sisson (2011), simulated annealing and clustering techniques in Hoffman et al. (2012). Unlike them, we will employ particle MCMC methodology explained in Andrieu et al. (2010) which replaces the intractable likelihood with an unbiased estimator given by a particle filter. Recently a policy guided Monte Carlo method (PGMC) has been proposed by Bojesen (2018) to improve the performance of the MCMC. Although most of the MCMC algorithms successfully can deal with the policy search problem, in high-dimensional state spaces or when the number of parameters are high an adaptive structure would be beneficial as considered by Andrieu and Robert (2001), Haario et al. (2005), Nguyen et al. (2018).

Chapter 3

Reinforcement Learning in Continuous State Spaces

Summary: In this chapter an overview to the RL structure, based on policy search techniques, in the domain of learning continuous and high dimensional state space dynamic systems driven by continuous input signals will be presented. General notations for the RL problem will be introduced in 3.1. Subsequently, some well-known policy gradient methods will be introduced.

3.1 Problem Statement and Background

In general, RL problems can be specified with the notion of Markov decision processes (MDP). An MDP is defined by the tuple

$$(\mathcal{S}, \mathcal{A}, g, \eta, r)$$

Here, $\mathcal{S} \subseteq \mathbb{R}^{d_s}$, where $d_s > 0$ represents a set of d_s-dimensional continuous state space, and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$, $d_a > 0$, stands for a continuous action space (control command).

We treat the state and action variables at time t as random variables $S_t \in \mathcal{S}$ and $A_t \in \mathcal{A}$ whose realizations will be denoted as s_t and a_t, respectively. At time $t > 0$, a transition from current state s_t to the next state s_{t+1} as a result of taking an action a_t admits a transition law described by the transition density function $g(s_{t+1}|s_t, a_t)$. We are interested in finite time horizon settings with a time length of n which episodically restart from an initial state. We denote the probability density for the initial state by $\eta(s_1)$. The reward function $r : \mathcal{A} \times \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ assigns an instantaneous real-valued scalar reward for the state transition from current state s_t to the next state s_{t+1} with action a_t, represented as $r(a_t, s_t, s_{t+1})$.

Let the control policy $h_\theta(a_t|s_t)$ be a stochastic parameterized policy with parameter $\theta \in \Theta \subseteq \mathbb{R}^{d_\theta}$ for some $d_\theta > 0$ and policy space Θ. The stochastic definition permits a characteristic resulting in exploration of the state space which is useful for hidden MDPs where the optimal policy is proven to be stochastic (Sutton et al. (2000)). The policy parameter θ corresponds to a probability density function $h_\theta(a_t|s_t)$ for the randomized action A_t at time t given state $S_t = s_t$. Letting $X_t = (S_t, A_t)$ taking values in $\mathcal{X} = \mathcal{S} \times \mathcal{A}$, this induces a Markov chain for $\{X_t\}_{t\geq 1}$ with transition law

$$f_\theta(x_t|x_{t-1}) := g(s_{t+1}|s_t, a_t)h_\theta(a_t|s_t) \tag{3.1}$$

where $x_t = (s_t, a_t)$. Therefore, the joint probability density of a trajectory (also called path, or rollout) $x_{1:n}$ until time n is

$$p_\theta(x_{1:n}) := f_\theta(x_1) \prod_{t=2}^{n} f_\theta(x_t|x_{t-1}) \tag{3.2}$$

where $f(x_1) = \eta(s_1)h_\theta(a_1|s_1)$, the initial distribution for X_1.

The objective of policy search in RL is to seek optimal or plausible policy parameters θ with respect to some expected performance of the trajectory $X_{1:n}$. A trajectory (path or rollout) is defined to be a collection of states $s_{1:n+1} = [s_1, s_2, \ldots, s_{n+1}]$

and actions $a_{1:n} = [a_1, a_2, \ldots, a_n]$. Define $R_n : \mathcal{X}^n \to \mathbb{R}$ to be the discounted sum of the immediate rewards up to time n

$$R_n(x_{1:n}) := \sum_{t=1}^{n-1} \gamma^{t-1} r(a_t, s_t, s_{t+1}) \tag{3.3}$$

where $\gamma \in (0, 1]$ is a discount factor, and let $U : \mathbb{R} \to \mathbb{R}$ be a monotonically increasing and continuous utility function with inverse U^{-1}. In a finite horizon reinforcement learning setting, the performance of a certain policy, $J_n(\theta)$ based on U and R_n is given by

$$J_n(\theta) = \mathbb{E}_\theta[U(R_n(X_{1:n}))] = \int p_\theta(x_{1:n}) U(R_n(x_{1:n})) dx_{1:n}. \tag{3.4}$$

where $p_\theta(x_{1:n})$ is the trajectory distribution. Various works formulate reinforcement learning as an inference problem for the policy parameter θ that is based on either maximizing (some function of) $J_n(\theta)$ with optimization techniques such as Gullapali et al. (1994), Dayan and Hinton (1997), Kimura and Kobayashi (1998), Toussaint and Storkey (2006), Peters (2005), Mitsunaga et al. (2005), Kappen et al. (2012), Maddison et al. (2017), or exploring admissible regions of $J_n(\theta)$ via some Bayesian approach as done by Hoffman et al. (2008) and Wingate et al. (2011).

3.2 Gradient based Algorithms for Policy Optimization

In some classical RL problems which are based on temporal differences as discussed in Sutton and Barto (1998), the expected reward of a policy for each individual state per time step i.e. s_t is calculated. This quantity which is known as *value function* $V^h(s_t)$, at each time step t, assesses the quality of each action a_t in the state s_t. Then this value is used to compute and subsequently update the policy h. However, since this value must be calculated for each state, it demands

filling the whole state-action space with the corresponding information for the value function. To explore the action space in order to find the one leading to an optimal value is therefore computationally hard, especially when the action space is continuous. Therefore, utilizing these methods in high-dimensional continuous state spaces are challenging. Policy search methodologies which will be discussed next have been proposed as alternatives to deal with the problems involved in value-based RL methods.

There are different choices for the function U depending on the nature of the reinforcement learning problem in terms of dealing with risk. A common choice is $U(x) = x$, which corresponds to the risk-neutral case where the performance measure of a policy reduces to its expected total reward. This case has been most extensively studied in the literature. Especially, among the available RL methods, the policy gradient algorithms which have drawn the most attention can be implemented in high-dimensional state-action spaces. This makes them well-suited in the robotics domain, where indeed problems usually involve coping with aforementioned spaces. Usage of policy gradient algorithms has been pioneered in the works by Gullapali et al. (1994). These methods have been employed to deal with complex control and robotics problems, such as those dealt in Kimura and Kobayashi (1998), Peters (2005), Mitsunaga et al. (2005), and Tavakol Aghaei and Onat (2017).

Specifically, the goal of policy optimization in RL is to quest optimal policy parameters θ that maximize the expected value of some function of R_n.

$$\hat{\theta} = \arg\max_{\theta \in \Theta} J_n(\theta). \tag{3.5}$$

Although it is hardly ever possible to evaluate $\hat{\theta}$ directly with this choice of R_n, maximization of $J_n(\theta)$ can be performed with *policy gradient* (PG) methods that

utilize the steepest ascent rule to update their parameters at iteration i as

$$\theta^{(i+1)} = \theta^{(i)} + \beta \nabla J_n(\theta^{(i)}), \tag{3.6}$$

where β is a learning rate and

$$\nabla J_n(\theta) = \left(\frac{\partial J_n(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J_n(\theta)}{\partial \theta_{d_\theta}} \right)^T$$

is the gradient of the expected total reward with respect to $\theta = (\theta_1, \ldots, \theta_{d_\theta})$. However, unless the state and the action spaces are finite or the Markov chain $\{X_t\}_{t \geq 1}$ admit linear and Gaussian transitional laws, it is impossible or too difficult to evaluate the gradient $\nabla J_n(\theta)$ in (3.6). In the sequel, we will review three main policy gradient methods that are proposed to efficiently approximate $\nabla J_n(\theta)$.

3.2.1 The REINFORCE Algorithm

One of the very first methods in estimating $\nabla J_n(\theta)$ in (3.6) is the REINFORCE algorithm introduced by Williams (1992) which exploits the idea of *likelihood ratio methods*. Since R_n does not depend on θ, $\nabla J_n(\theta)$ can be written as:

$$\nabla J_n(\theta) = \int \nabla p_\theta(x_{1:n}) R_n(x_{1:n}) dx_{1:n}. \tag{3.7}$$

Next, by using (3.2) as well as the 'likelihood trick' identified by $\nabla p_\theta(x_{1:n}) = p_\theta(x_{1:n}) \nabla \log p_\theta(x_{1:n})$, where the product converted to summation according to logarithm's specifications, we can rewrite

$$\nabla J_n(\theta) = \int p_\theta(x_{1:n}) \left[\sum_{t=1}^{n} \nabla \log h_\theta(a_t | s_t) \right] R_n(x_{1:n}) dx_{1:n}. \tag{3.8}$$

(3.8) includes the log-derivative of the policy distribution. Since the derivative of the logarithm of the policy solely depends on the policy parameter, estimating a gradient from paths is possible without an explicit model by swapping the expectation with summation. This policy estimator is known as episodic REINFORCE. Due to the lack of exact information about the trajectory distribution $p_\theta(x_{1:n})$ or non-linearity in $p_\theta(x_{1:n})$, the integration over this probability distribution may not be possible. The REINFORCE algorithm approximates $\nabla J_n(\theta)$ by producing $N \geq 1$ generated trajectories of length n from $p_\theta(x_{1:n})$,

$$x_{1:n}^{(i)} = (s_1^{(i)}, a_1^{(i)}, \ldots, s_n^{(i)}, a_n^{(i)}) \overset{\text{i.i.d.}}{\sim} p_\theta(x_{1:n}), \quad i = 1, \ldots, N,$$

and then performing the Monte Carlo estimate

$$\nabla J_n(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{n} \nabla \log h_\theta(a_t^{(i)} | s_t^{(i)}) \right] R_n(x_{1:n}^{(i)}). \tag{3.9}$$

In Williams (1992), it is shown that this estimator suffers from high variance. In order to reduce this variance, he proposes to make use of a baseline $b \in \mathbb{R}^{d_\theta}$ to modify (3.9) as

$$\frac{\partial J_n(\theta)}{\partial \theta_j} \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{n} \frac{\partial \log h_\theta(a_t^{(i)} | s_t^{(i)})}{\partial \theta_j} \right] \left(R_n(x_{1:n}^{(i)}) - b_j \right),$$

$$j = 1, \ldots, d_\theta. \tag{3.10}$$

This baseline $b = (b_1, \cdots, b_{d_\theta})$ is adaptively calculated during iterations from sample trajectories $x_{1:n}^{(i)}$ in a heuristic manner so that the variance of the approximation is minimized and it is adapted during the iterations.

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta \left[\nabla_\theta \log p_\theta(x_{1:n}) \left(R(x_{1:n}) - b \right) \right] \tag{3.11}$$

This is an unbiased estimator as

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta \left[\nabla_\theta \log p_\theta(x_{1:n}) b\right] = b \int \nabla_\theta p_\theta(x_{1:n}) dx_{1:n}$$

$$= b \nabla_\theta \int p_\theta(x_{1:n}) dx_{1:n} = b \nabla_\theta [1] = 0 \tag{3.12}$$

In order to minimize the variance one should calculate the optimal baseline by equating the gradient of the variance of (3.10) with respect to baseline b to zero as

$$\frac{\partial}{\partial b_j} Var\left[\nabla_{\theta_j} J_n(\theta)\right] = \frac{\partial}{\partial b_j} (\mathbb{E}[(\nabla_{\theta_j} J_n(\theta))^2] - \mathbb{E}[\nabla_{\theta_j} J_n(\theta)]^2) = 0 \tag{3.13}$$

According to (3.12) the second expression in (3.13) would be zero. Putting the equivalent expression for $\nabla_\theta J(\theta)$ denoted by (3.11) in the first term of the right hand side of (3.13) will yield in

$$\frac{\partial}{\partial b_j} \left(\mathbb{E} \left(\frac{\partial}{\partial \theta_j} \log h_\theta(a_t|s_t)(R(x_{1:n}) - b_j) \right)^2 \right) = 0 \tag{3.14}$$

Thus by solving this equation with respect to b the optimal baseline will be obtained:

$$b = \frac{\mathbb{E}\left(\left(\sum_{t=0}^{n-1} \nabla_\theta \log h_\theta(a_t|s_t) \right)^2 R(x_{1:n}) \right)}{\mathbb{E}\left(\left(\sum_{t=0}^{n-1} \nabla_\theta \log h_\theta(a_t|s_t) \right)^2 \right)} \tag{3.15}$$

The resulting REINFORCE algorithm is given in Algorithm 1. The REINFORCE algorithm utilizes the return of the entire episode to assess the quality of the taken actions during each trajectory. The variance of the returns depends on the length of the paths and may increase henceforth, the performance of the approximated gradient may get worse, regardless of whether it is utilized with the baseline. One way to overcome this weakness is to use the rewards earned in each time step.

This perception inspired the proposal of a new algorithm called Gradient in Partially Observable Markov Decision Processes (GPOMDP); see Baxter and Bartlett (2000).

Algorithm 1: REINFORCE Algorithm

Input: Number of time steps n, Number of episodes N, Initial parameter θ with dimension d_θ

Output: Gradient estimate for expected return $\nabla_\theta J(\theta)$

for *each episode e = 1, 2, . . . , N* **do**

Collect the data set as trajectories $\{x_{1:n}, u_{1:n-1}, r_{1:n}\}_{e=1:N}$

Compute the expected discounted reward

$$R_e(x_{1:n}) = \sum_{t=1}^{n} \gamma^{t-1} r(a_t, s_t, s_{t+1})$$

end

Calculate the optimal baseline

$$b = \frac{\sum_{e=1}^{N} \left(\left(\sum_{t=0}^{n-1} \nabla_\theta \log h_\theta(a_t^{(e)} | s_t^{(e)}) \right)^2 R_e(x_{1:n}) \right)}{\sum_{e=1}^{N} \left(\left(\sum_{t=1}^{n} \nabla_\theta \log h_\theta(a_t^{(e)} | s_t^{(e)}) \right)^2 \right)}$$

Approximate the gradient for each d_θ

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_{e=1}^{N} \sum_{t=1}^{n} \nabla_\theta \log h_\theta(a_t^{(e)} | s_t^{(e)}) (R_e(x_{1:n}) - b)$$

3.2.2 The GPOMDP Algorithm

As previously stated the variance of the REINFORCE algorithm is directly dependent on the number of visits to the state. Thus, its value is prone to get larger whenever the state space is high-dimensional. The gradient estimation in GPOMDP algorithm is done according to the instantaneous rewards r_t which are being assigned to the agent in each time step. It hinges on the fact that rewards are not correlated

with future actions.

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{k=1}^{n}\sum_{t=1}^{k}\nabla_\theta \log h_\theta(a_t|s_t)(r_k - b_k)\right] \tag{3.16}$$

Here expectation is over the samples in each episode and b_k is the calculated baseline for each time step similarly according to (3.15) with a difference that here for GPOMDP the immediate rewards are used. The resulting algorithm is given in Algorithm 2

Algorithm 2: GPOMDP Algorithm

Input: Number of time steps n, Number of episodes N, Initial parameter θ with dimension d_θ

Output: Gradient estimate for expected return $\nabla_\theta J(\theta)$

for *each episode e = 1, 2, . . . , N* **do**

  Collect the data set as trajectories $\{x_{1:n}, u_{1:n-1}, r_{1:n}\}_{e=1:N}$

end

for *each time step t = 1,2, . . . , n* **do**

  Calculate the optimal baseline for each time step

$$b = \frac{\sum_{e=1}^{N}\left(\sum_{t=1}^{j}\nabla_\theta \log h_\theta(a_t^{(e)}|s_t^{(e)})\right)^2 r_j}{\sum_{e=1}^{N}\sum_{t=1}^{n}\left(\sum_{t=1}^{j}\nabla_\theta \log h_\theta(a_t^{(e)}|s_t^{(e)})\right)^2}$$

  Approximate the gradient for each d_θ

$$\nabla_\theta J(\theta) = \sum_{e=1}^{N}\sum_{j=1}^{n}\left(\sum_{t=1}^{j}\nabla_\theta \log h_\theta(a_t^{(e)}|s_t^{(e)})\right)(r_j^{(e)} - b_j)$$

end

3.2.3 The eNAC Algorithm

The policy gradient theorem given by Sutton et al. (1999) states that in the approximated gradient of the cost function, instead of using the total reward of a trajectory, the quality function of the state and action at a time step $Q(s_t, a_t)$ can

be used. Then the calculated gradient regarding the baseline can be given by:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=1}^{n} \nabla_\theta \log h_\theta(a_t|s_t)(Q(s_t,a_t) - b_t)\right] \tag{3.17}$$

In order to estimate the quality function $Q(s_t, a_t)$ function approximation methods have been proposed in Sutton et al. (1999). This function which is known as an advantage function is composed of the combination of some basis functions ϕ with parameter weight vectors w and given as:

$$A_w(s_t,a_t) = \phi(s_t,a_t)^T w \approx Q(s_t,a_t) - b_t \tag{3.18}$$

For the sake of simplicity, it is assumed that the baseline is zero. Now we should find a parameter vector that can minimize the squared error between the quality function and the advantage function:

$$\frac{\partial}{\partial w}\mathbb{E}\left[\sum_{t=1}^{n} \left(Q(s_t,a_t) - A_w(s_t,a_t)\right)^2\right] = 0 \tag{3.19}$$

By taking the derivative in (3.19) we will get:

$$2\mathbb{E}\left[\sum_{t=1}^{n} \left(Q(s_t,a_t) - A_w(s_t,a_t)\right) \frac{\partial}{\partial w} A_w(s_t,a_t)\right] = 0 \tag{3.20}$$

By subtracting the resulting equation from equation (3.17) by the assumption that the baseline is zero and considering $\frac{\partial}{\partial w} A_w(s_t,a_t) = \phi(s_t,a_t)$ one can get:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=1}^{n} \nabla_\theta \log h_\theta(a_t|s_t) A_w(s_t,a_t)\right] \tag{3.21}$$

By taking basis functions to be $\phi(s_t, a_t) = \nabla_\theta \log h_\theta(a_t|s_t)$ and considering the fact that $A_w(s_t, a_t) = \phi(s_t, a_t)^T w$ then the gradient can be rewritten as:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=1}^{n} \nabla_\theta \log h_\theta(a_t|s_t) \nabla_\theta \log h_\theta(a_t|s_t)^T\right] w = G_\theta w \tag{3.22}$$

It is shown by Peters and Schaal (2008) that the matrix G_θ cancels out for the natural gradient and thus the approximated gradient of the cost function needs solely the calculation of parameter vector w. They proposed the episodic Natural Actor Critic (eNAC) algorithm to obtain these parameters where the problem has been treated as a regression one. The complete derivations regarding the eNAC algorithm can be found in Peters (2007). This algorithm is summarized in Algorithm 3. Note that we limit ourselves to the survey of just those strategies which are firmly related to the work in this thesis.

Algorithm 3: The eNAC Algorithm

Input: Number of time steps n, Number of episodes N, Initial parameter θ with dimension d_θ

Output: Gradient estimate for expected return $\nabla_\theta J(\theta) = w$

for *each episode e = 1, 2, . . . , N* **do**

Collect the data set as trajectories $\{x_{1:n}, u_{1:n-1}, r_{1:n}\}_{e=1:N}$

Copmute the expected discounted reward

$$R_e(x_{1:n}) = \sum_{t=1}^{n} \gamma^{t-1} r(a_t, s_t, s_{t+1})$$

Compute the feauture matrix

$$\psi^e = \begin{bmatrix} \sum_{t=1}^{n} \nabla_\theta \log h_\theta(a_t^{(e)} | s_t^{(e)}) \\ \phi(s^{(e)}) \end{bmatrix}$$

end

Establish the feature and return matrices

$$R = \left[R^1, R^2, \ldots, R^N\right], \ \psi = \left[\psi^1, \psi^2, \ldots, \psi^N\right]$$

Using regression problem calculate the vector of parameters w

$$\begin{bmatrix} w \\ v \end{bmatrix} = \left(\psi^T \psi\right)^{-1} \psi^T R$$

Chapter 4

Monte Carlo Methods

Summary: In this chapter the principle ideas of the Monte Carlo methods including importance sampling, Markov Chain Monte Carlo (MCMC), Metropolis-Hastings, Sequential Monte Carlo will be sketched first. Then these methods in the context of the Bayesian inference will be extended to the reinforcement learning setting. The objective is to propose an MCMC algorithm for the problem of policy search in the RL paradigm.

4.1 Monte Carlo Approximation

The motivation is to take expectation over a measurable function which is defined by some random variable $X \rightarrow \mathcal{X}$ and denoted by $\omega : \mathcal{X} \rightarrow \mathbb{R}^{d_\omega}$ with a probability distribution as $p(x)$:

$$\Omega = \mathbb{E}\left[\omega(X)\right] = \int_{\mathcal{X}} p(x)\omega(x)dx \tag{4.1}$$

Solving this integral is possible by using analytical integration methods under the condition that both the kernel distribution $p(x)$ and the measurable function $\omega(x)$

are completely available and known. However, in most cases, an explicit model of a system is not at hand and thus these situations do not hold. Another challenge arises when the dimensionality of the variable space $d_{\mathcal{X}}$ is large i.e., the computation complexity explodes exponentially with the dimension of $\mathcal{X}$. This phenomenon is known as *curse of dimensionality* (Bellman (1957)). Robotic frameworks frequently need to manage these high-dimensional states and actions because of the numerous degrees of freedom (DoFs) of modern robots. Therefore, numerical methods can hardly be applied to these problems. A promising option in contrast to deterministic techniques for integration issues is Monte Carlo method, where random samples are drawn from some artificial distributions (easy to sample from) and then these data samples are used to estimate the integral.

In the Monte Carlo methods, the distribution which is going to be approximated Ω is targeted with N independent, identically distributed (i.i.d) samples $X^{[1]}, X^{[2]}, \ldots, X^{[N]}$ which are either directly drawn from Ω or from an instrumental expert designed distribution. By averaging over these N examples the distribution Ω will be approximated.

$$\Omega \approx \frac{1}{N} \sum_{i=1}^{N} \omega(X^{[i]}) \tag{4.2}$$

It is proven that this estimator is an unbiased estimator and when the number of samples N approaches infinity, the convergence of the estimator is guaranteed (Rosenthal (2006)).

4.2 Importance Sampling

As it maybe obvious from its name, Importance Sampling (IS) puts weights on samples depending on their significance and similarity degree between them. IS is regularly exhibited as a strategy for decreasing the variance of an approximated

expectation via cautiously picking a distribution. In general, the goal is to evaluate the integral $\int f(x)p(x)dx$ by $i.i.d \sim p(x)$. Now instead of taking samples from $p(x)$ a new distribution which is easy to sample from namely $q(x)$ is introduced. Then approximating the integral changes to estimating the following one:

$$P(f) = \mathbb{E}_P\left[f(X)\right] = \int f(x)\frac{p(x)}{q(x)}q(x)dx, \quad i.i.d \sim q(x) \tag{4.3}$$

The ratio $p(x)/q(x)$ is called the weights $W(x)$ of the IS. This estimator is always unbiased if and only if both $p(x)$ and $q(x)$ own the same support. One modification to the IS formulation is normalized IS where it is divided by sum of the weights applied to the samples:

$$P_{IS}(f) = \frac{\sum_{i=1}^{N} f(x^{[i]})W(x^{[i]})}{\sum_{i=1}^{N} W(x^{[i]})} \tag{4.4}$$

Importance sampling in the subject of RL has been incorporated as a function approximation tool to estimate the Q-values (Precup et al. (2001) and Precup et al. (2000)). It has also been used to improve the REINFORCE algorithm for the partially observable MDP problems (Meuleau et al. (2001)).

4.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) algorithms are alternatives for sampling from distributions which are complex. MCMC is dependent on the creation of an ergodic Markov Chain where its samples are able to imitate the ones drawn from the objective distribution. It should be noted that, MCMC demands a stationary distribution π to converge to.

A chain with a Markovian property is a sequence of samples $\{X_n\}_{n\geq 1}$ drawn from a transition probability P where n is an index for the sample number which

has the following attribute:

$$p(X_n|X_{1:n-1} = x_{1:n-1}) = p(X_n|X_{n-1} = x_{n-1}) \tag{4.5}$$

expressing that the present state of the chain at time n given all the preceding states relies only on the previous state at time $n-1$.

MCMC methods have drawn attention in various branches such as machine learning, image processing and statistics (Andrieu et al. (2003) Erdil et al. (2016) Yıldırım et al. (2015)).

4.3.1 Metropolis-Hastings Algorithm

A very simple yet applicable class of MCMC algorithms is Metropolis-Hastings (MH) discussed in Hastings (1970) and Metropolis et al. (1953). The overall insight of the MH algorithm consists of proposing a new value x' conditional on its previous value x from a candidate proposal kernel $q(x'|x)$. The Chain then either admits it and the next state exploration happens around the accepted value or denies the proposed value x' and the current value does not encounter a change. The acceptance ratio of the MH algorithm is according to:

$$\alpha(x', x) = \min\left\{1, \frac{q(x|x')\pi(x')}{q(x'|x)\pi(x)}\right\} \tag{4.6}$$

A general form of the MH algorithm is summarized in Algorithm 4.

4.4 Sequential Monte Carlo

A recursive type of the importance sampling is known as Sequential Monte Carlo (SMC) which is useful for the situations with sequential interactions. Lets

Algorithm 4: The MH Algorithm

Input: Number of iterations i, Initial value for sample $x^{(1)}$
for *each episode iteration i = 1, 2, . . . , N* **do**
  Sample a uniform random number $u \sim \mathcal{U}[0, 1]$
  Sample a candidate value from the proposal kernel given the current value $x' \sim q(x'|x^{(i)})$
  Compare the acceptance probability with the drawn uniform random number
  if $u \leq \alpha(x', x) = \min\left\{1, \frac{q(x|x')\pi(x')}{q(x'|x)\pi(x)}\right\}$
    proposal is accepted ; $x^{(i+1)} = x'$
  else
    proposal is rejected ; $x^{(i+1)} = x^{(i)}$
  end
end

assume a sequence of random variables $\{X_n\}_{n\geq 1}$ over a space $\mathcal{X}$ with a sequence of distributions $\{\pi_n\}_{n\geq 1}$. A sequence of real-valued functions $\{\phi_n\}_{n\geq 1}$ are also defined. The objective is to compute the expectation over the real-valued function in a sequential manner as:

$$\pi_n(\phi_n) = \mathbb{E}_{\pi_n}\left[\phi_n(X_{1:n})\right] = \int \phi(x_{1:n})\pi_n(x_{1:n})dx_{1:n} \tag{4.7}$$

In the sequel we will describe how to deal with this integral.

4.4.1 Sequential Importance Sampling and resampling

The proposal importance distributions for the $\pi_n(x_{1:n})$ can be defined in a sequential form $\{q_n(x_{1:n})\}_{n\geq 1}$ as importance weights defined by the following term:

$$w_n(x_{1:n}) = \frac{\pi_n(x_{1:n})}{q_n(x_{1:n})} \tag{4.8}$$

since the importance density $q_n(x_{1:n})$ has a sequential behavior it can be written in the following form:

$$q_n(x_{1:n}) = q(x_1) \prod_{t=2}^{n} q(x_t|x_{1:t-1}) \tag{4.9}$$

with an initial distribution of $q(x_1)$. In this direction, the equation (4.8) can be rearranged as:

$$w_n(x_{1:n}) = \frac{\pi_n(x_{1:n})\pi_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})\pi_{n-1}(x_{1:n-1})} = w_{n-1}(x_{1:n-1})\frac{\pi_n(x_{1:n})}{\pi_{n-1}(x_{1:n-1})q(x_n|x_{1:n-1})} \tag{4.10}$$

The calculated importance weight then is used to obtain the normalized version of it which is utilized in the sequential importance sampling (SIS) algorithm which is given in Algorithm 5

$$W_n^{(i)} = \frac{w_n(X_{1:n}^{(i)})}{\sum_{i=1}^{N} w_n(X_{1:n}^{(i)})} \tag{4.11}$$

An issue with IS is that, except if the proposal distribution $q(x_{1:n})$ is very similar to the objective true distribution $\pi_n(x_{1:n})$, the normalized weights will ordinarily put their significance in just a single particle leading to a small number of particles with a large weight in comparison to the others. This phenomenon is called the weight degeneracy problem. To overcome this problem a resampling step is introduced. In the resampling procedure, samples are drawn from the weighted distribution $X_{1:n-1}^{(i)}$ (with different weights) and are substituted with the particles which are equally weighted $\widetilde{X}_{1:n-1}^{(i)}$. For performing resampling, first we approximate $\pi_{n-1}(x_{1:n-1})$ as:

$$\widehat{\pi}_{n-1}(x_{1:n-1}) = \sum_{i=1}^{N} W_{n-1}^{(i)} \delta(x_{1:n-1}^{(i)}) \tag{4.12}$$

Algorithm 5: The SIS Algorithm

Input: Number of iterations i, number of time steps n

for *n = 1, 2, ...* **do**

  for *iterations i = 1,2,...,N* **do**

    if $n = 1$

      Sample from initial density $X^{(i)} \sim q(X_1)$

      Compute its corresponding importance weight $w_1(X_1^{(i)}) = \frac{\pi_1(X_1^{(i)})}{q_1(X_1^{(i)})}$

    else

      Sample $X_n^{(i)} \sim q(X_n^{(i)}|X_{1:n-1}^{(i)})$, Compose the data $X_{1:n}^{(i)} = \left(X_{1:n-1}^{(i)}, X_n^{(i)}\right)$

      Compute the importance weight

$$w_n(X_{1:n}^{(i)}) = w_{n-1}(X_{1:n-1}^{(i)}) \frac{\pi_n(X_{1:n}^{(i)})}{\pi_{n-1}(X_{1:n-1}^{(i)}) q(X_n^{(i)}|X_{1:n-1}^{(i)})}$$

  end

  for *iterations i = 1,2,...,N* **do**

    calculate the normaliozing importance weight

$$W_n^{(i)} = \frac{w_n(X_{1:n}^{(i)})}{\sum_{i=1}^{N} w_n(X_{1:n}^{(i)})}$$

  end

end

Now N particles are independently drawn from $\widehat{\pi}_{n-1}(x_{1:n-1})$ according to the following probability density:

$$\widetilde{X}_{1:n-1}^{(i)} \sim \mathcal{P}(\widetilde{X}_{1:n-1}^{(i)} = X_{1:n-1}^{(i)}) = W_{n-1}^{(j)} \tag{4.13}$$

Then these particles are used to approximate an equally weighted function with weights $1/N$

$$\widehat{\pi}_{n-1}(x_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\tilde{x}_{1:n-1}^{(i)}) \tag{4.14}$$

In the next step we have $\widehat{\pi}_{n-1}(x_{1:n-1})$ and can estimate $\pi_n(x_{1:n})$ by drawing samples $X_n^{(i)}$ from the resampled distribution as $X_n^{(i)} \sim q(X_n^{(i)}|\widetilde{X}_{1:n-1}^{(i)})$ and after constructing

the data samples as $X_{1:n}^{(i)} = \left(\widetilde{X}_{1:n-1}^{(i)}, X_n^{(i)}\right)$, weights are assigned to them as following:

$$W_n(X_{1:n}^{(i)}) = \frac{\pi_n(X_{1:n}^{(i)})}{\pi_{n-1}(\widetilde{X}_{1:n-1}^{(i)})q(X_n^{(i)}|\widetilde{X}_{1:n-1}^{(i)})} \tag{4.15}$$

4.5 Sequential Markov Chain Monte Carlo Algorithm for Reinforcement Learning

4.5.1 Introduction

Policy search approaches have encouraged the use of reinforcement learning (RL) to dynamic frameworks, for example, control of robots. Numerous policy search algorithms depend on the gradient based methods, and in this manner may encounter the ill effects of slow convergence or local optima difficulties. In the presented thesis, we adopt a gradient-free Bayesian inference strategy to the policy search problem under RL setting, for the case of controlling a discrete time Markov Decision process with continuous state and action spaces. The method consists of accepting a prior density over unknown policy parameters and then targeting the posterior distribution where the likelihood is considered as the expected return. We propose a Markov chain Monte Carlo (MCMC) algorithm as a strategy for creating samples for the policy parameters from the objective posterior function. The proposed algorithm is compared with certain outstanding gradient based RL techniques and shows progressively proper performance regarding time response accomplishments and convergence speed.

We advocate a unique RL policy search technique utilizing Particle Markov chain Monte Carlo (P-MCMC), an ongoing and effective group of MCMC strategies for complex distributions. Our algorithm is best pertinent for risk-sensitive

situations, where a multiplicative expected total reward is used to quantify the performance of the executed actions, as opposed to the more typical additive one; since with a multiplicative structure for the return function, one can completely use sequential Monte Carlo (SMC), known as the standard particle filters (Doucet et al., (2001)) inside the iterations of the P-MCMC.

The proposed algorithm does not require gradient computations and along these lines it does not create a point-wise estimate for the policy, rather, it appraises the approximations done over the expected reward straightforwardly and supplies samples from the policy density which would then be able to be utilized to investigate the surface of the policy performance to recognize the ideal areas throughout the policy space. In this way, it does not face the danger of stalling out in nearby local optimal points or veer from its desired solution because of the bad choices of learning rate for the gradient ascent laws. In that sense, our proposed strategy might be invaluable, in any event, in terms of breadth of applicability, over techniques that do require gradient calculations. The claims on robustness and convergence are bolstered with given numerical investigations and simulations in Chapter 5.

4.5.2 Policy Search Based on Reward Assessment

Here we take the structure of the Markov Decision Process (MDP) the same as that discussed in Section 3.1. The term policy search in RL refers to an ultimate goal of an agent (controller) for finding optimal policy parameters θ considering the quality of the expected cost of a given trajectory $x_{1:n}$. Assume a discounted summation of the instantaneous rewards over a trajectory in each step time given as:

$$R\left(x_{1:n}\right) = \sum_{t=1}^{n-1} \gamma^{t-1} r\left(s_t, a_t\right) \tag{4.16}$$

where $x_{1:n} = [(s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n)]$ is a trajectory of state-action pairs obtained from the state space transition kernel, γ is a discount factor and r is a real-valued reward at each time. Consider a monotonically increasing and continuous utility function (Howard and Matheson (1972)) as $U : \mathbb{R} \to \mathbb{R}$ with its inverse denoted as U^{-1}. In a finite horizon RL setting, the performance of a certain policy is given as:

$$J(\theta) = \mathbb{E}_\theta \left[U(R(X_{1:n})\right] = \int p_\theta(x_{1:n}) U\left(R(X_{1:n})\right) dx_{1:n} \tag{4.17}$$

where $p_\theta(x_{1:n})$ is the path transition density. Some papers focus on RL problem as an inference mechanism which try to solve the policy search problem by maximizing $J(\theta)$ via hiring optimization tools such as Toussaint and Storkey (2006), Kappen et al. (2012) Kimura and Kobayashi (1998). Some other works use Bayesian methods to search for favorable portions of $J(\theta)$ such as Hoffman et al. (2008) and Wingate et al. (2011). In this study we extend the idea to the risk-sensitivity in RL and incorporate Bayesian inference to cope with the policy search dilemma.

We go for a specific alternative for the utility function U in the form of an exponential one:

$$U(X) = \frac{1}{\kappa} \exp(\kappa X) \tag{4.18}$$

where $\kappa > 0$ is a risk component. By this choice we can rewrite the relation for the cost function $J(\theta)$ given in equation 4.17:

$$\begin{aligned} J(\theta) &= \mathbb{E}_\theta \left[\frac{1}{\kappa} \exp(\kappa R(X_{1:n}))\right] \\ &= \frac{1}{\kappa} \int p_\theta(x_{1:n}) \prod_{t=1}^{n-1} \exp\left\{\kappa r(a_t, s_t, s_{t+1}) \gamma^{t-1}\right\} dx_{1:n}. \end{aligned} \tag{4.19}$$

where in the structure of the return function instead of additive return we used a multiplicative reward function. Then the classical policy search problem in the finite time horizon can be modified to a risk-sensitive framework where the goal is to

seek for policy parametrization $\theta \in \Theta$ that maximizes $U^{-1}(J(\theta))$ which is the same problem as to maximizing $J(\theta)$, see Osogami (2012) and Marcus et al. (1997). The risk-sensitivity can manage the uncertainty occupied in the return function in the RL framework. The source of the uncertainties are naturally because of the either characteristic stochastic model dynamics or parameters involved (Tamar (2015)).

There exist some papers which focused on risk sensitive RL. For example, Geibel and Wysotzki (2011) have made use of some error states to create the risk and then applied it to the value function based RL. Another work done by Mihatsch and Neuneier (2002) assigned temporal differences (TD) errors to be the risk-sensitive element instead of changing the structure of the return and established the risk-sensitive category for the Q-learning algorithm. In a recent attempt by Shen et al. (2013), a risk-sensitive Q-learning algorithm for unknown state-spaces have been extracted. In fact, these algorithms are applicable for discrete state spaces and therefore in a general sense differ from our structure where the control problem in MDPs with continuous state and action spaces are concerned.

The cumulative expected reward leads to approximations with large bias and variance as elaborately argued by Maddison et al. (2017). To lighten this issue, we will transform the return function to an exponential multiplicative total reward. The contribution of this step emerges in the reformulation of the policy search in a Bayesian inference form where the expected multiplicative return, is handled as if it is a likelihood function. The supporting idea to opt for an expectation of a multiplicative return is the ability to employ unbiased lower variance estimators of $J(\theta)$.

4.5.3 Bayesian inference as a tool for policy search

The primary commitment in this work is to propose a Bayesian methodology for RL that is executed by means of MCMC. In particular, our procedure is a particle

MCMC calculation including a SMC (particle filter) in its iterations. The novelty of the presented approach is due to a formulation of the policy search problem in a Bayesian inference where the normal expected multiplicative reward is treated as a pseudo-likelihood function. The purpose behind taking $J(\theta)$ as an expectation of a multiplicative reward is the capacity to utilize lower variance and unbiased estimators of it, in spite of the techniques that employ the additive reward functions which lead in approximations with high variance. Rather than attempting to think of a single policy parametrization, the problem is extended into a Bayesian setting where the performance function $J(\theta)$ is considered as a likelihood function of θ. Then a posterior distribution $\pi(\theta)$ is constructed which is proportional to the multiplication of an informative prior function of policy parameters $\mu(\theta)$ with the cost function as:

$$\pi(\theta) \propto \mu(\theta)J(\theta) \tag{4.20}$$

in which instead of dealing directly with $J(\theta)$, we choose $\pi(\theta)$ as an artificial distribution and perform sampling through MCMC algorithm. By this method we can benefit from the fact that the areas with high reward can be thoroughly investigated by the agent until it is adequately sure that no other policy improvement will occur which leads to ending up with the optimal parameters among the solution space.

There exist likewise Bayesian techniques for policy learning proposed for a particular choice of the utility function as $U(x) = x$. Like our structure, these approaches put a prior distribution with probability density $\mu(\theta)$ over the policy parameter θ and employ some reasonable function of $J(\theta)$ as the likelihood function to establish an instrumental posterior distribution to target. For instance, a trans-dimensional MCMC algorithm which is essentially based on Expected Maximization (EM) algorithm is presented for sampling from $\pi(\theta)$ in the work done by Hoffman et al. (2007). Another option for tackling the posterior density is proposed in Wilson et al. (2010) where a hybrid MCMC algorithm based on importance sampling is used. Unlike our work Wingate et al. (2011) allow for negative rewards in the structure of

the value function but use the exponential form of it as likelihood and then employ an approximate MCMC algorithm to draw samples from the posterior function. Additionally, there are hybrid methods as the EM algorithm, where the aim is to maximize the value function $V(\pi)$ with respect to policy parameters by applying forward-backward algorithms as shown by Toussaint and Storkey (2006). Lately in Maddison et al. (2017), similar to our work, an exponential utility function has been chosen for the risk-sensitivity case for the return function but they tackle the policy search problem in RL with particle filter based value function and then update the polices with policy gradient algorithms.

4.5.4 Policy search via MCMC

In this part, the advantages of the Bayesian approach will be clarified. The problem of our interest is focused on the control of an MDP with continuous state and action space in discrete time.

In the situation where the analytical evaluation of the integrals mentioned in equation (4.17) is not easy to handle (intractable), approximation methods can play an essential role. Although, as an estimator a suitable alternative would be the use of classical numerical integration methods, they may face the so called *curse of dimensionality* problem. It is due to the explosion of the computation load with the number of dimensions. A promising choice for the numerical integration can be Monte Carlo estimators. A famous category of these estimators is MCMC algorithms which have drawn interest among the experts ranging from statistics to Machine Learning. MCMC methods are well-suited for high-dimensional complex distributions where sampling are difficult. It is a sampling approach constructed for a deep search of the parameter space and puts its concentration more on the most prominent parts of the space where the value of the total expected reward is large.

Taking into account the relation (4.20), it is computationally hard to handle the posterior distribution $\pi(\theta)$ which includes an intractable integral derived from $J(\theta)$.

MCMC techniques are based on creating an ergodic Markov chain $\left\{\theta^{(m)}\right\}_{m\geq 0}$ which is guaranteed to finally provide samples distributed with respect to a target distribution, starting from an initial point $\theta^{(0)}$, which has the true target distribution as the stationary distribution, for our situation $\pi(\theta)$. Statistically, en ergodic Markov Chain stands for a chain which has the property of being irreducible and aperiodic. For finite state Markov chains, irreducibility implies that each state can be visited beginning from each of them and aperiodicity signifies that each state can be visited any time step n greater than some fixed number. In the event that one reproduces such a Markov chain, after a sufficiently long time the samples of the Markov chain will concede $\pi(\theta)$. Generally speaking, sampling from a target distribution with a transition kernel P considering the samples $\left\{X^{(i)}\right\}_{i\geq 0}$ from Markov Chain occurs in the following manner. Given a current initial sample $x^{(1)}$, a next sample state $x^{(2)}$ is drawn. Then $x^{(3)}$ is sampled conditioned on the previous sample $x^{(2)}$. This process continues sequentially according to the transition probability $P(x^{(n)}|x^{(n-1)})$. Then the target distribution at time n denoted as $\pi_n(x)$ with respect to the transition kernel P becomes:

$$\pi_n(x) = \int P(x|x')\pi_{n-1}(x')dx' \tag{4.21}$$

which results in a sequence of marginal distributions $\{\pi_n\}_{n\geq 1}$.

4.5.4.1 Metropolis-Hastings algorithm for policy search

The main question here is how to find a transition kernel P that is easy to sample from, and target density $\pi(\theta)$ is its invariant distribution i.e., in the long run when n approaches infinity it should meet $\pi = P\pi$. The answer is to use the simple yet famous Metropolis-Hastings (MH) algorithm. Apparently, the most widely used MCMC method is the MH methodology where most of the useful MCMC algorithms

can be conceived as special cases or expansions of this method. While, this approach does not present any method to determine an explicit mathematical expression to the distribution which is typically difficult to obtain even for discrete state spaces case, it just conveys a procedure to draw samples given the past ones.

The strategy of the MH algorithm for the invariant density $\pi(\theta)$ incorporates sampling a candidate proposal θ' given the present one $\theta^{(m-1)} = \theta$ for trials $m = 1, 2, \ldots$ as indicated by a proposal density $q(\theta'|\theta)$ as

$$\theta' \sim q(\theta'|\theta).$$

steering the chain towards the proposal indicating that the proposed value θ' is accepted with an acceptance probability of $\alpha(\theta, \theta') = \min\{1, \rho(\theta, \theta')\}$, (for which $0 \leq \alpha(\theta, \theta') \leq 1$) and thus the current quantity of the parameter changes as $\theta^{(m)} = \theta'$ or it remains the same as its previous value $\theta^{(m)} = \theta$ expressing that the proposed value is dismissed. The acceptance ratio is formulated as:

$$\rho(\theta, \theta') = \frac{q(\theta|\theta')}{q(\theta'|\theta)} \frac{\pi(\theta')}{\pi(\theta)} = \frac{q(\theta|\theta')}{q(\theta'|\theta)} \frac{\mu(\theta')J(\theta')}{\mu(\theta)J(\theta)}. \tag{4.22}$$

when a symmetrical proposal distribution is chosen the acceptance rate will be simplified to:

$$\rho(\theta, \theta') = \frac{\pi(\theta')}{\pi(\theta)} \tag{4.23}$$

The transition kernel P in the MH step is defined as:

$$P(\theta'|\theta) = q(\theta'|\theta)\alpha(\theta, \theta') + \delta(\theta - \theta')\beta(\theta') \tag{4.24}$$

where $\delta(.)$ is the Delta Dirac function and β is the rejection probability with $0 \leq \beta(\theta') \leq 1$ which is given as:

$$\beta(\theta') = \int \left(1 - \alpha(\theta, \theta')\right) q(\theta'|\theta) d\theta \tag{4.25}$$

Samples are not directly picked from the transition kernel P, they are instead sampled from the proposal distribution $q(\theta'|\theta)$, yet dismissing a portion of the samples drawn. Usually, evaluating P is not feasible since most of the time it is intractable. It is worth noting that the choice of the proposal distribution is essential to the extent that the statistical features of the Markov chain strongly depend on this decision. An inadequate decision brings about potentially poor performance of the Monte Carlo estimators. To be specific, the efficiency of the MH algorithm depends on the quality of the selected standard deviation of the proposal distribution. On the unlikely case that it is excessively limited, it will generate samples which are closely correlated and mixing slowly. Then again, in case it is excessively wide, the rejection rate can be remarkably high resulting in the repetition of the old parameters during the new iterations. Generally, samples created by the MCMC should be mixed well such that they effectively overlook their previous values. Specifically, it is necessary to choose a logical proposal variance to guarantee well-mixing. It is simple to demonstrate that the samples produced by MH algorithm will converge asymptotically to those drawn from the target distribution, as shown by Andrieu et al. (2003).

The MH algorithm requires the ability of calculating the acceptance ratio $\rho(\theta, \theta')$ for any given $\theta, \theta' \in \Theta$. For our case, this is not possible since calculation of $\rho(\theta, \theta')$ demands computation of both $J(\theta)$ and $J(\theta')$ which is not generally feasible due to the complex formulation of $J(\theta)$ in (4.19). Despite this, we are still able to target $\pi(\theta)$ using an MCMC algorithm in case unbiased and non-negative estimates of $J(\theta)$ can be found i.e,. obtain random variables $\widehat{J}(\theta) \geq 0$ such that we have:

$$\mathbb{E}[\widehat{J}(\theta)] = J(\theta)$$

The MH algorithm that uses such estimate $\widehat{J}(\theta)$ instead of $J(\theta)$ is called the

pseudo-marginal MH algorithm (PMMH) (Andrieu and Roberts (2009)). The application of the proposed PMMH algorithm for our RL problem is given in Algorithm 6. For a broader family of such algorithms for hidden Markov models (HMM), namely particle MCMC algorithms; refer to Andrieu et al. (2010). Algorithm 6 reveals that dealing with non-negative unbiased estimators of $J(\theta)$ denoted as $\widehat{J}$ instead of $J(\theta)$ itself is possible, and we can still target $\pi(\theta)$ directly.

Algorithm 6: Pseudo-marginal Metropolis-Hastings for reinforcement learning

Input: Number of time steps n, initial value and estimate of expected performance $(\theta^{(0)}, \widehat{J}^{(0)})$, proposal distribution $q(\theta'|\theta)$, number of particles N

Output: Samples $\theta^{(k)}$, $k = 1, 2, \ldots$

for *k = 1, 2, ...* **do**

Given $\theta^{(k-1)} = \theta$ and $\widehat{J}^{(k-1)} = \widehat{J}$, sample a proposal value $\theta' \sim q(\theta'|\theta)$.

Obtain an unbiased estimate $\widehat{J}'$ of $J(\theta')$ by using Algorithm 7 with N particles.

Accept the proposal and set $\theta^{(k)} = \theta'$ and $\widehat{J}^{(k)} = \widehat{J}'$ with probability $\min\{1, \widehat{\rho}(\theta, \theta')\}$ where

$$\widehat{\rho}(\theta, \theta') = \frac{q(\theta|\theta')}{q(\theta'|\theta)} \frac{\mu(\theta')}{\mu(\theta)} \frac{\widehat{J}'}{\widehat{J}},$$

otherwise reject the proposal and set $\theta^{(k)} = \theta$ and $\widehat{J}^{(k)} = \widehat{J}$.

end

4.5.4.2 SMC for approximating the cost function $J(\theta)$

The challenge now is getting unbiased estimators of $J(\theta)$ for any given policy parameter θ. Considering the point that a multiplicative expected reward function is used we can come up with a solution for this problem by employing an SMC algorithm. The basic concepts of the SMC method have already been covered in section 4.4.1. For sake of simplicity (and without loss of generality regarding the validity of our methodology), we will assume from now on that $\kappa = 1$ and the

immediate reward function only depends on $x_t = (a_t, s_t)$. With this assumption, we can rewrite the equation for $J(\theta)$

$$J(\theta) = \int f_\theta(x_1) \exp\{r(a_1, s_1)\} \prod_{t=1}^{n-1} f_\theta(x_t|x_{t-1}) \exp\left\{\gamma^{t-1} r(a_t, s_t)\right\} dx_{1:n}, \qquad (4.26)$$

which suggests that we may extend the problem of estimating the expected return in our RL setting into the framework of *Feynman-Kac formulae* stated in Del Moral (2004), in which $\{X_t\}_{t\geq 1}$ is the Markov chain and $\exp\{\gamma^{t-1} r(a_t, s_t)\}$ is the potential function at time t. In other words, we can think of $\{X_t\}_{t\geq 1}$ as the latent Markov process of a hidden Markov model, and $\gamma^{t-1} r(a_t, s_t)$ as the conditional observation density at time t for some fixed observation. It was shown in Del Moral (2004) that a non-negative and unbiased estimator of $J(\theta)$ can be obtained by running a SMC algorithm, known as particle filter, as in Algorithm 7.

Algorithm 7 includes propagation, weighting, and re-sampling of N particles (samples) at a time step. When only one particle is used i.e. $N = 1$, the algorithm is simplified to sampling a single trajectory $X_{1:n} \sim p_\theta(x_{1:n})$ and computing $\exp\{R(X_{1:n})\} = \prod_{t=1}^{n} e^{\gamma^{t-1} r(X_t)}$ as an estimate of the expected total reward.

We hereby point out some results which connect RL and hidden Markov models in the context of Feynman-Kac formulae by implementing inference methods used for hidden Markov models and RL. For example, an Expectation Maximization (EM) algorithm has been used for discrete and Gaussian RL settings in Toussaint and Storkey (2006). An approximate inference tool based on EM used for more complex situations in Rawlik et al. (2010). Likewise, in a most recent paper by Maddison et al. (2017), Algorithm 7 has been used for an unbiased estimator of $J(\theta)$. While they employ this estimator to update the policy parameters by using policy gradient algorithm, we implement an MCMC algorithm by traversing the space of the posterior distribution in a Bayesian learning framework.

Algorithm 7: SMC algorithm for an unbiased estimate of $J(\theta)$

Input: Policy θ, number of time steps n, discount factor γ
Output: Unbiased estimate of $\widehat{J}$
Start with $\widehat{J}_0 = 1$.
for $t = 1, \ldots, n$ **do**
  for $i = 1, \ldots, N$ **do**
    if $t = 1$ **then**
      Sample $\widehat{X}_1^{(i)} \sim \eta_\theta(x_1)$.
    else
      Sample $\widehat{X}_t^{(i)} \sim f_\theta(\cdot|X_{t-1}^{(i)})$ using the transition density
    end
    Calculate $W_t^{(i)} = e^{\gamma^{t-1} r(\widehat{X}_t^{(i)})}$.
  end
  if $t < n$ **then**
    Resample from $\{\widehat{X}_t^{(i)}, i = 1, \ldots, N\}$ with probabilities proportional to $\{W_t^{(i)}, i = 1, \ldots, N\}$ to obtain resampled particles $\{X_t^{(i)}, i = 1, \ldots, N\}$, i.e.

$$P(X_t^{(i)} = \widehat{X}_t^{(j)}) = \frac{W_t^{(j)}}{\sum_{j'=1}^{N} W_t^{(j')}}, \quad i = 1, \ldots, N.$$

  end
  Update the estimate: $\widehat{J}_t = \widehat{J}_{t-1} \times \frac{1}{N} \sum_{i=1}^{N} W_t^{(i)}$ **return** $\widehat{J}$.
end

4.6 Adaptive MCMC for Policy Search in High-dimensional State Spaces

4.6.1 Introduction

Dealing with nonlinear and complex systems in case of the parameter estimation is conceived to be a challenging task in different aspects, specially control and robotics systems due to the inherent large scale dimensions. Three common sense solutions to this problem are categorized as Expectation Maximization (EM) (see for example the method taken by Fearnhead et al. (2010) for state space parameter

estimation), gradient based (such as policy gradient RL performed by Peters (2005), where the gradients are estimated and PILCO where the gradients are calculated analytically as introduced in Deisenroth and Rasmussen (2011)) and Bayesian inference methods (a current complete review by Kantas et al. (2015)). Although the advantage of the Bayesian methods over both EM and gradient-based is that it can be computationally cheaper, the other two may better be scaled to high-dimensional state spaces. One common disadvantage for the gradient based and EM is that they are prone to get stuck in local optima, whereas the Bayesian inference can skip local solutions and come up with a global one. In this section, we intend to improve the performance of the proposed MCMC algorithm to approximate the policy parameters of a nonlinear state-space model in a way that it becomes well-suited for high-dimensions. For this purpose, the proposed MCMC algorithm will be modified to an adaptive form. (Adaptive-MCMC). The reason for this is that tuning the parameters of the proposal distribution by hand is not practical due to the large number of variables. To show the effectiveness of the proposed Adaptive-MCMC algorithm, it will be applied on a nonlinear model of a *Ballbot* system to control the stabilization of the yaw, pitch and roll. The resulting successful simulations will be presented in Chapter 5.

4.6.2 Bayesian inference for general state space framework

State space models (generally Hidden Markov models (HMM)), constitute a sequence of latent Markov Decision Process (MDP) state models $\{X_t\}_{t\geq 1}$ taking values in some measurable space $\mathcal{X} \subseteq \mathbb{R}^{d_x}$. The MDP can be categorized by its initial distribution $\eta(.)$ and corresponding state transition density $f_\theta(.)$ where $\theta \in \Theta \subset \mathbb{R}^{d_\theta}$ is the unknown parameter vector. Assuming a sequence of observations $\{Y_t\}_{t\geq 1}$ made for the latent variables from the measurement distribution $g_\theta(.)$ we have

$$X_1 \sim \eta(.)$$

$$x_t \sim f_\theta(x_t|x_{t-1}); \qquad t = 2, 3, \dots$$

$$y_t \sim g_\theta(y_t|x_t)$$

Bayesian method aims to find the latent HMM states which are conditioned on some observations up to a final time T i.e., calculating the marginal posterior distribution $p(x_{1:T}|y_{1:T})$ as following:

$$p(x_t|y_{1:t}) = \int \frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{p(y_t|y_{1:t-1})} p(x_{t-1}|y_{1:t-1}) dx_{t-1} \tag{4.27}$$

Generally speaking, unless the state space model is either discrete (finite) or it obeys a linear Gaussian model (Kalman Filter can deal with), that integral in equation (4.27) turns in to an intractable one which can not be evaluated. This methodology approximates the state variables given the observations with the assumption that the parameter vector θ is known. But for the problem of our interest it is not the case. In order to cast this idea to approximating the policy parameters, we take θ as random variables and create a prior distribution over them as $\mu(\theta)$. Then a posterior distribution is constructed where the latent variables are considered as parameter vector θ (which are going to be estimated) conditional on some observations regarding the parameter θ. The sequence of observed data to be used are assumed to be the measure of some performance of the system for each parameter denoted as $J(\theta)$. Since we are dealing with the RL framework which is based on the policy search with respect to the expected total rewards, the performance function $J(\theta)$ (as the observed data) is assumed to be the expected value of multiplicative rewards. Thus the posterior distribution will have the form $p(\theta_{1:N}|J_{1:N}(\theta))$ where N is the number of total iterations for collecting data. According to equation (4.27), the resulting posterior distribution is almost impossible to calculate and sample from. To alleviate this, an artificial distribution which is proportional to the combination of the observations and the priors introduced for parameters is proposed as $p(x_{1:T}|y_{1:T}) \propto \mu(x_{1:T}) p(y_{1:T}|x_{1:T})$, where from now on we replace the notation of

latent variables $x_{1:T}$ and observations $y_{1:T}$ with their corresponding variables as parameter vector θ and expected cost function of parameter vector $J(\theta)$, respectively i.e., $p(\theta|J(\theta)) \propto \mu(\theta)p(J(\theta)|\theta)$. So rather than directly evaluating the marginal posterior conditioned on $J(\theta)$, we target its equivalent proportional expression. For continuous non-Gaussian state space models sampling from full posterior $p(\theta|J(\theta))$ or calculating marginal likelihood $p(J(\theta)|\theta)$ is not possible directly. Despite this, except if we can find an unbiased and non-negative estimate of $J(\theta)$, it is still practical to sample from the posterior. One approach for targeting this problem is to leverage the PMMH where employs an SMC algorithm to estimate the marginal likelihood as $\widehat{p}(J(\theta)|\theta)$. Then this estimate will be used to compute the acceptance rate for the Metropolis-Hastings (MH) algorithm. One advantage that can be stated about PMMH algorithm is its capability to target the posterior distribution whatever the number of particles is. However, when it comes to the high-dimensioanl problems where the number of parameters θ is relatively large, it demands precise tuning of all the proposal distributions which is not cost-effective. Specially, when the problem at hand is beyond the statistical problems, for example in control and robotics domain where tuning multi-dimensional parameter of policy plays a crucial role in the safety and performance of the system. Therefore, letting the standard deviations of the proposal distributions as a fixed value during the whole learning iterations is not reasonable. The algorithm traverses the parameter space in each iteration but assigning constant values for the proposal distribution may restrict it for some specific regions leading to a repetitive excessive acceptance or rejection of the policy parameters (updating all the parameters at the same time would not be possible), whereas modifying the proposal distribution according to the system's new situation (considering the stability and logical working point at each area of the policy space) can cause a remarkable improvement in the learning performance of the PMMH algorithm.

4.6.3 Adaptive MCMC

The concept of the Adaptive MCMC (A-MCMC) has faced great enthusiasm due to some developments on its hypothetical facts (Haario et al. (2001), Roberts and Rosenthal (2008)). One of the advantages of the A-MCMC algorithm is that it employs all of the samples from the beginning of the sampling procedure to tune the covariance matrix of the proposal density. Assume a random walk Metropolis for the proposal distribution of the MCMC algorithm:

$$X_{i+1} = X_i + \mathcal{N}(\mu, s\Sigma) \tag{4.28}$$

which is performed by using a multivariate Gaussian density identified by:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d_x/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{4.29}$$

where d_x is the dimensionality of the state-space and μ is the mean and s is an scaling factor for the covariance matrix. As shown by Roberts and Rosenthal (2001) and Roberts et al. (1994), the optimal value for the scaling factor s is equal to $2.38/\sqrt{d_x}$ for large scale problems. Goal here is to tune the covariance matrix Σ by employing an adaptive method. We dedicate our focus to the adaptive method introduced by Robbins and Monro (1951). In fact we are looking for a method which is mathematically reasonable and on the other side is computationally cheap. Considering the expected acceptance probability $\bar{\alpha}$ which is usually a non-increasing monotonic function and optimal acceptance as α^* the Robbins-Monro update rule can be stated as: if $\bar{\alpha} > \alpha^*$ then Σ is too small and need to be increased and vice versa. According to this intuition the following updating law is given:

$$\Sigma_{i+1} = \Sigma_i + \lambda_{i+1}(\bar{\alpha}_i - \alpha^*) \tag{4.30}$$

where $\lambda > 0$ is a stochastic step size which guarantees that in the long run the changes occurred in the value of Σ will eventually disappear meaning that:

$$|\Sigma_{i+1} - \Sigma_i| \leq \lambda_{i+1} \to 0$$

The value of the step size λ in the Robbins-Monro formula should satisfy the following conditions:

$$\sum_{i=1}^{\infty} \lambda_i = \infty$$

$$\sum_{i=1}^{\infty} \lambda_i^{1+\epsilon} < \infty; \quad \epsilon > 0$$

Since the derivations of the A-MCMC algorithm is out of the scope of this study, we just refer to the A-MCMC algorithm that has been used for our problem. This algorithm was elaborately outlined in the works done by Atchade et al. (2010), Bai et al. (2008). An adaptive metropolis algorithm introduced by Haario et al. (2001) which will be used during the iterations of our proposed MCMC algorithm is given in Algorithm 8. As mentioned earlier, although an acceptable scaling factor

Algorithm 8: Adaptive Metropolis algorithm

Input: initial parameter, coavariance, mean and scaling factor of the random walk Metropolis $\theta_0, \mu_0, \Sigma_0, s_0$

Output: updated covariance matrix Σ

for *iteration i = 1, 2, ...* **do**

Sample a proposal value $\theta'_{i+1} \sim q(\theta'|\theta_{i-1}) = \mathcal{N}(\theta_{i-1}, s_{i-1}\Sigma_{i-1})$ from the random walk metropolis proposal and compute the average acceptance rate $\bar{\alpha}$

Update the covariance

$$\log(s_{i+1}) = \log(s_{i-1}) + \lambda_{i+1}(\bar{\alpha} - \alpha^*)$$

$$\mu_{i+1} = \mu_{i-1} + \lambda_{i+1}(\theta'_{i+1} - \mu_{i-1})$$

$$\Sigma_{i+1} = \Sigma_i + \lambda_{i+1}\left((\theta'_{i+1} - \mu_{i-1})^T(\theta'_{i+1} - \mu_{i-1}) - \Sigma_{i-1}\right)$$

end

s for the covariance matrix is taken to be $2.38/\sqrt{d_x}$, to enforce it to a logical one at the first iterations of the adaptation it is suggested to modify accordingly, as well. The algorithm implies that starting with large Σ_0 will result in large acceptance ratio and the other way around which reflects the fact that the learning convergence for the covariance parameters would be slow when using a constant scaling factor. For a comprehensive study related to the different adaptive MCMC algorithms refer to Andrieu and Thoms (2008). In the context of high-dimensioanl state spaces, the effectiveness of the covariance matrix Σ in the performance of the MCMC algorithm is constrained by its computational complexity. To put it better, optimizing and upgrading the covariance matrix in a sequential manner can cause computational load as shown by Nishihara et al. (2014). Consequently, the urge for comprehensive research in generating more sophisticated algorithms about the covariance matrix adaptation for the MCMC method is strong to enhance its productivity.

Chapter 5

Numerical Simulations

Summary: In this Chapter first we will contribute to the concept of the fuzzy control by using the PG RL algorithms in order to tune the scaling factors of fuzzy controllers. In the second part the proposed MCMC algorithm will be implemented on a nonlinear model of an Inverted pendulum in a continuous MDP framework. In this direction, a linear feedback control is used to stabilize the inverted pendulum in the upright position. To exhibit the efficiency of the proposed MCMC algorithm it will be compared with the discussed PG RL algorithms in chapter 3. The outcomes show that proposed MCMC algorithm either outperforms the performance of that of PG methods both in convergence and time-response or can have a similar performance. And finally, in the last part of this chapter, proposed MCMC algorithm will be applied on a nonlinear model of a 2-Degree of Freedom (DoF) robotic manipulator to perform a reference trajectory tracking control. The results obtained, will be compared with a gradient based Adaptive PD method and eNAC algorithm.

5.1 Tuning Fuzzy Logic Controllers (FLC) by Using PG RL Algorithms

The early successful implementations of FLCs has been completed by Mamdani (1974). Their promising results has turned them into an option to classical control methods. FLCs have the advantage of consolidating expert knowledge into the classical control problems in order to deal with complex control problems. The most well-known category of FLCs are Proportional Integral Derivative (PID) controllers. Other than the current established gain tuning algorithms, various tuning methodologies based on optimization for both traditional and FLCs can be exemplified, such as Big Bang-Big Crunch (BB-BC) applied by Wang and Kumbasar (2018), Genetic Algorithm studied in Ko et al. (2006) and ant colony used in Duan et al. (2006).

The present part dedicates its emphasis to explore the possibility of applying PG RL algorithms discussed in section 3.2 on FLCs, due to the fact that their contribution in this domain have not been remarkably considered. Although there exist some works related to gain tuning of the FLCs, for example, Boubertakh et al. (2010) Aghaei et al. (2015) and some which improve the quality of Q-learning algorithm by fuzzy logic such as Busoniu et al. (2010), these attempts mostly focus on discrete state spaces. Instead, we utilize PG RL techniques to tune the parameters of the FLCs. This is helpful in the light of the fact that value function based RL methods (such as Q-learning) need to construct a look-up table for each individual state-action pair which is prohibitively difficult in state-spaces with large dimensions and demands using appropriate function approximators to establish a mapping between states. Without loss of generality, the employed PG RL algorithms, minimize the expected cumulative return of the learning algorithm, which evaluates the precision of the step response of a closed loop system with a fuzzy controller, and succeeds in finding the optimal gain values of the FLC. The obtained closed loop time response quality satisfies the aimed specifications, as shown by our simulations.

5.1.1 Structure of the fuzzy controller

The general form of a fuzzy rule, which is designed by an expert to control the system, is identified by defining the output for some specific inputs as

$$If\ x_1\ is\ A_1\ \dots x_n\ is\ A_n\ then\ O\ is\ B$$

in which x_i, are crisp inputs, A_i are fuzzy sets and O is the output of the system located at point B. This rule in general consists of two parts, the *If* part is called *premise* and *then* part is called *consequent*. Each rule has a degree of firing which signifies its applicability as:

$$\nu_k = \prod_{k=1}^{n} \nu_{A_k} \tag{5.1}$$

here ν_k is the firing strength of the k^{th} rule and whenever $\nu_k > 0$ its corresponding rule is activated. Considering the current situation of the system, the decision making part tries to find the set of rules which are activated at that time. For our setting the inputs to the FLC are taken as the state error and its derivative $(e, \dot{e})$. The output of the fuzzy rule is control signal. The fuzzy inputs for the FLC rules can have Triangular, Gaussian or Trapezoidal forms with some linguistic variables to distinguish the range of applicability of each set from others. The output of the system can also be either a fuzzy set or a *singleton*. This output needs to be a crisp value so a mechanism called as *defuzzification* is used to convert the fuzzy inputs to numeric values (methods such as center of gravity, weighted mean). It has been shown that for a FLC with a product-sum inference, center of gravity defuzzification and triangular uniformly distributed Membership Functions (MF) for the input fuzzy sets, the output can be computed as:

$$U = A + Pk_e e + Dk_d \dot{e} \tag{5.2}$$

where k_e and k_d are the gains for error and its derivative. For more details about the derivation and the A, P and D components refer to Qiao and Mizumoto (1996). The system to be controlled is an inverted pendulum which its equations are explicitly given in section 5.2.1; see Dann et al. (2014). The goal is to control the angle of the pendulum and its angular velocity $(\phi, \dot{\phi})$. The governing rules (consisting of a total of 9 rules) which display how to best control this system under the fuzzy logic framework is given in Table. 5.1, where N, Z and P terms stand for Negative, Zero and Positive linguistic terms, respectively. For each of the error, error derivative

$e/\dot{e}$	derivative of error		
error	**N**	**Z**	**P**
N	C_1	C_1	C_0
Z	C_1	C_0	C_{-1}
P	C_0	C_{-1}	C_{-1}

TABLE 5.1: Symmetrical rule-base of a FLC for controlling the inverted pendulum.

and output of the system three symmetrical triangular MFs are used as illustrated in Fig. 5.1. The universe of discourses of the input MFs are as:

$$E_N = -\frac{\pi}{2}, \quad E_P = \frac{\pi}{2}, \quad D_N = -\frac{\pi}{4}, \quad D_P = \frac{\pi}{4}$$

and for the output MF these values are defined as following:

$$u_N = -20, \quad u_{C_{-1}} = -10, \quad u_{C_0} = 0, \quad u_{C_1} = 10, \quad u_P = 20$$

The crisp control output value is calculated by using the center of gravity method which is:

$$u = \frac{\sum_{i=1}^{M} b_i \int \nu_i}{\sum_{i=1}^{M} \int \nu_i} \tag{5.3}$$

where b_i is the center of the MF in the consequent part of the i^{th} rule and M is the total number of rules.

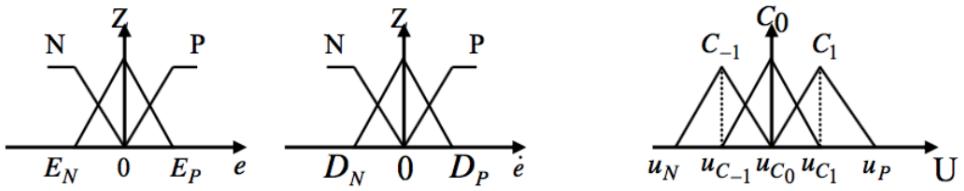


FIGURE 5.1: FLC Membership functions for the input-output.

5.1.2 PG RL settings

Tuning the scaling gains of the PD-type FLC will be performed by eNAC and GPOMDP RL algorithms discussed earlier. The output signal of the PD-type FLC which will be applied to the plant (here inverted pendulum) can be written as:

$$U = A + Pk_e e + Dk_d \dot{e} + \epsilon \tag{5.4}$$

in which ϵ is a zero mean white Gaussian noise as $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and applied to the control signal during the learning process to ensure the stochastic nature of the policy. Throughout the testing policy, this value would be zero (deterministic). The objective of the control problem is to tune the parameter vector $\theta = [k_e \quad k_d]$. The parametric policy π_θ is selected to be a Gaussian one with standard deviation σ_{π_θ}:

$$\pi_\theta(a|s) = \frac{1}{\sigma_{\pi_\theta}\sqrt{2\pi}} \exp\left(-\frac{(U - \theta s)^2}{2\sigma_{\pi_\theta}^2}\right) \tag{5.5}$$

where s is the continuous state of the systems as the error for the angle ϕ and its velocity $\dot{\phi}$ i.e., $s = [e \quad \dot{e}]^T$. U is given by equation (5.4). After determining the policy, it is required to compute the gradient of the logarithm of the parameterized policy with respect to the given parameters θ, in order to approximate $\nabla_\theta J(\theta)$. This

is done by taking the derivative from π_θ as following:

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{a - \theta s}{\sigma^2_{\pi_\theta}} s \tag{5.6}$$

If one considers learning σ_{π_θ} as well, then taking gradient of π_θ will be performed with respect to σ_{π_θ} as:

$$\nabla_{\sigma_{\pi_\theta}} \log \pi_\theta(a|s) = \frac{(a - \theta s)^2 - \sigma^2_{\pi_\theta}}{\sigma^3_{\pi_\theta}} \tag{5.7}$$

note that in our simulations for the sake of simplicity, σ_{π_θ} is considered as a constant value of 2, therefore we have just made use of equation (5.6).

For the reward function, two general structures are assumed where we call the first one as *interval based reward* and the second one as *absolute value reward* and define them in the following:

$$r(s,a) = \begin{cases} 0, & -8^\circ \leq \phi \leq 8^\circ \\ -10, & \text{otherwise.} \end{cases} \tag{5.8}$$

$$r = -w_\phi |\phi_{ref} - \phi| - w_{\dot{\phi}} |\dot{\phi}_{ref} - \dot{\phi}| - w_a |a|. \tag{5.9}$$

where ϕ_{ref} and $\dot{\phi}_{ref}$ are the desired pendulum angles and angular velocity (for the stabilization issue, the desired corresponding values for these variables are zero). The assigned weights for the vertical angle (pendulum), angular velocity and force applied to the horizontal cart are $w_\phi, w_{\dot{\phi}}$ and w_a, respectively. We took these values to be $w_\phi = 3$, $w_{\dot{\phi}} = 0.85$ and $w_a = 0.1$. Note that in equation (5.9), in case of failure (angle is out of its accepted range), the simulation will stop and the agent will be assigned a negative value of -1000. During the simulations, the discounted factor $\gamma = 0.9$ and the nonlinear model of the inverted pendulum is simulated with the *Fourth-Order Runge-Kutta* method in MATLAB. Number of episodes for the REINFORCE, eNAC and GPOMDP algorithms is selected to be 100 where in each

episode the inverted pendulum is run for $10s$ with a sampling time of $0.01s$. To measure the performance of each learning algorithm, they have been tested and averaged over 20 experiments in which REINFORCE algorithm could only manage to get an optimal solution for the parameter vector θ under the absolute value reward function with $\sigma_{\pi_\theta} = 0.001$, and thus we only considered its resulting time response figure.

The average return plots for the eNAc and GPOMDP algorithms are covered in Fig. 5.2 which are normalized in the range $[0, 1]$ to capture a justified comparison between both rewards. It should be noted that the difference in the ranges of the horizontal axis is due to the fact that the GOPMDP algorithm converges almost six times earlier than the convergence iteration of the eNAC.

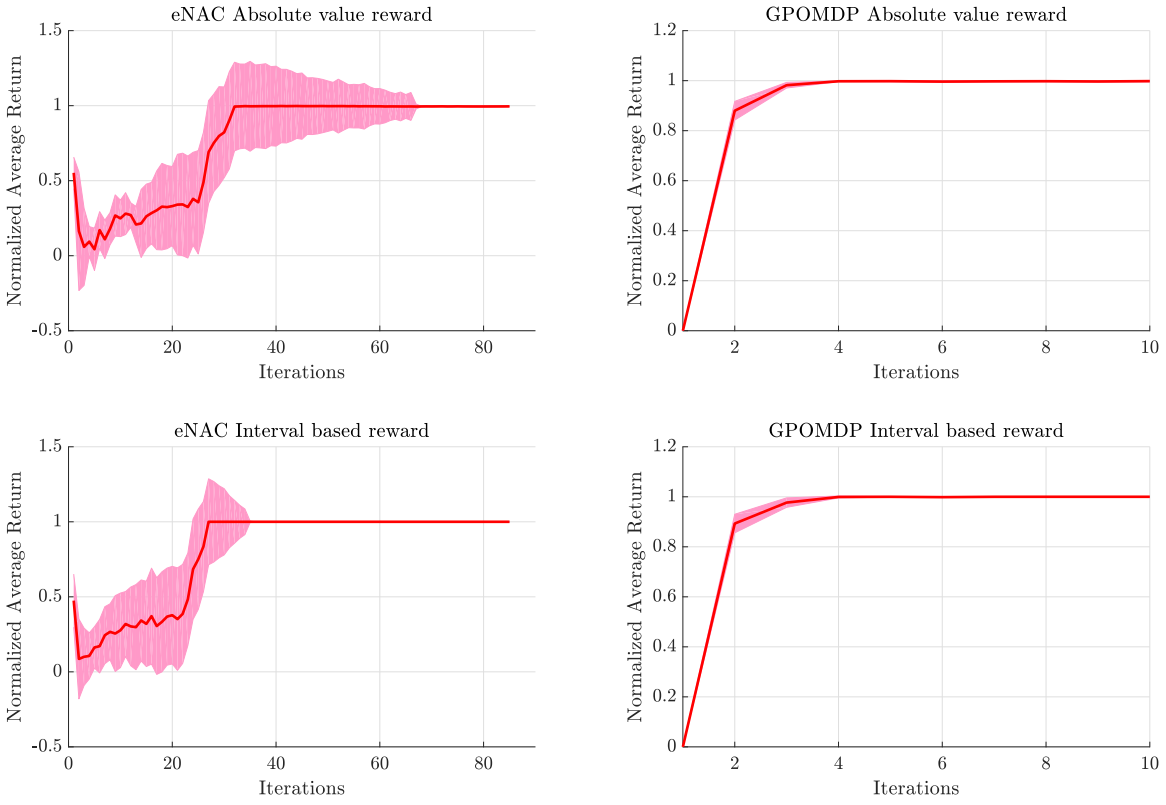


FIGURE 5.2: Normalized average return for eNAC and GPOMDP.

According to Fig. 5.2, although GPOMDP's convergence speed is higher than eNAC, an overall observation of the time response in Fig. 5.3 and Fig. 5.4, suggests that, eNAC resulted in better time response specifications than GPOMDP. The fast convergence of the GPOMDP algorithm is because of falling in the local optimum, while eNAC with Absolute value reward attempted to skip the local solution for the parameter vector and finally came up with a promising one. The higher standard deviation in the performance of the eNAC algorithm reveals that behavior of the performed 20 simulations are not very similar to each other. Considering ϕ in Fig. 5.3, eNAC algorithm with the Absolute value reward excels its counterpart in closed loop time response by rejecting overshoot in the response. For the Interval based reward, settling time is less than that of Absolute value reward but it shows almost a small undershoot. Similarly, for the angular velocity $\dot{\phi}$, eNAC with the Absolute value reward exhibits better performance in case of rejecting overshoot (while it bears an undershoot) and for the Interval based reward, the settling time is less. It is clear from Fig. 5.4 that GPOMDP with the Absolute value reward demonstrates a better settling time by almost three times less than that of Interval based reward for both ϕ and $\dot{\phi}$.

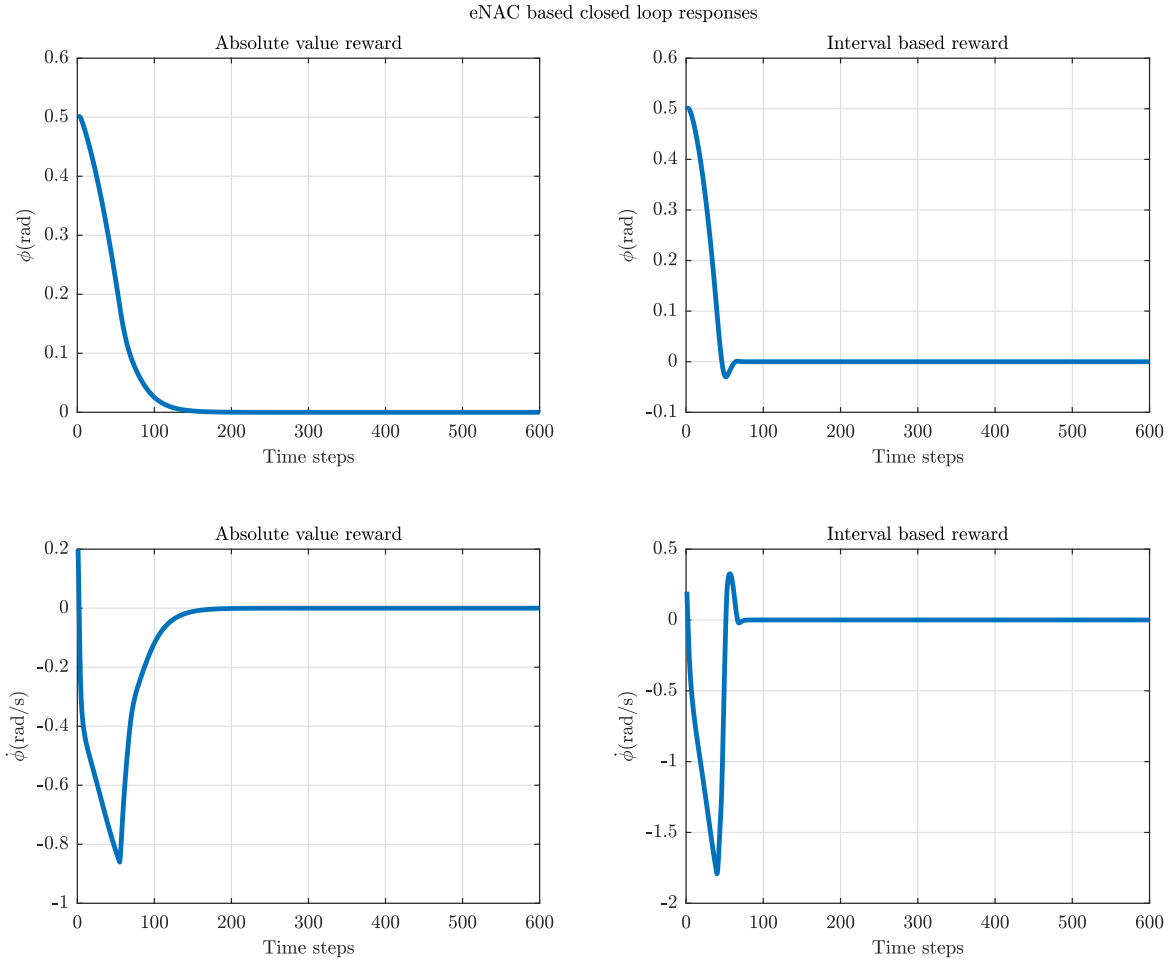


FIGURE 5.3: Closed loop responses for eNAC algorithm.

Since eNAC algorithm had an early settling time in the closed loop responses, the range of the horizontal axis in its corresponding time domain responses is considered to be less than the other two algorithms. Therefore, for a better illustration of the time domain plots their horizontal axis is taken to be in different ranges.

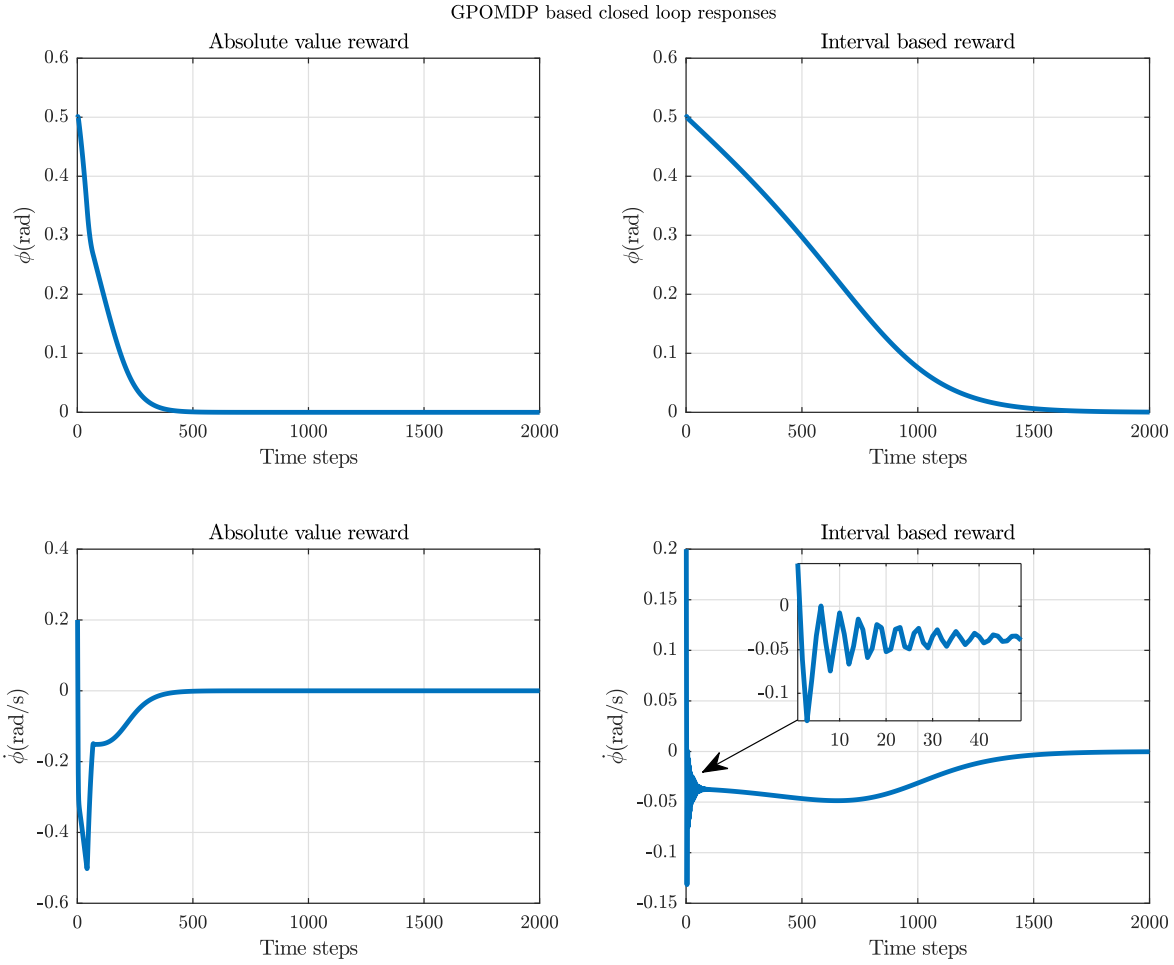


FIGURE 5.4: Closed loop responses for GPOMDP algorithm.

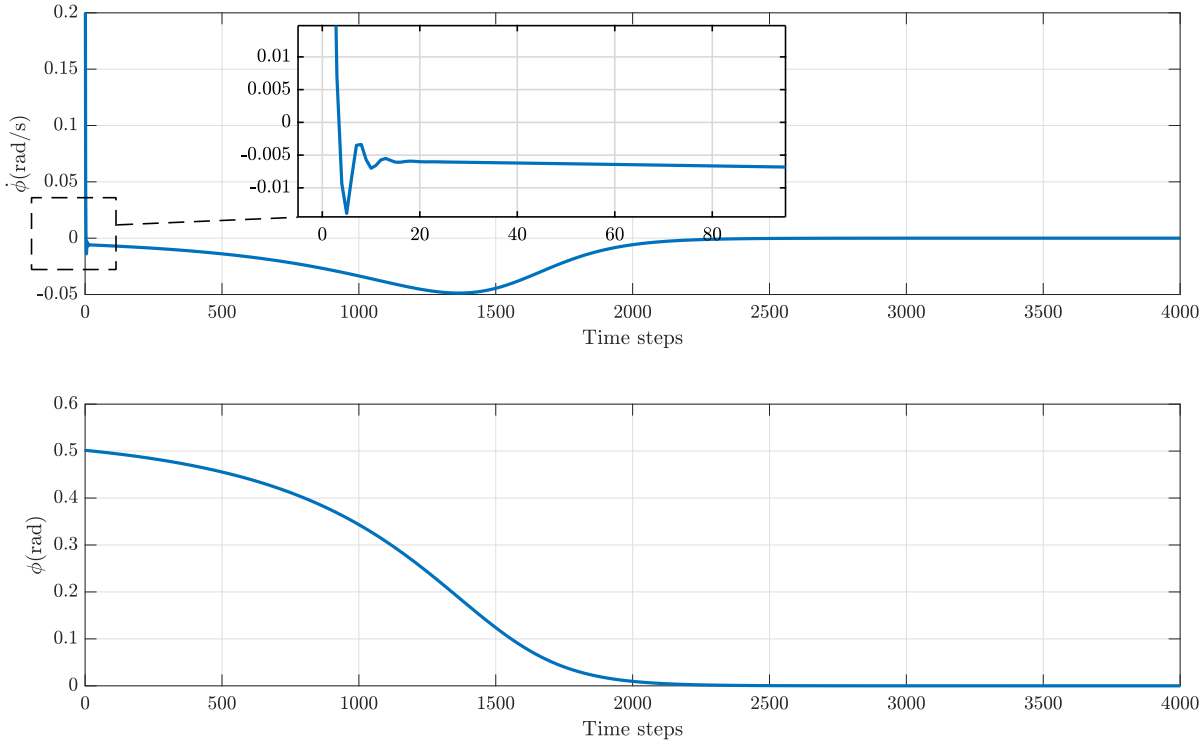


FIGURE 5.5: Closed loop responses for REINFORCE algorithm.

Finally, Fig. 5.5 displays the time responses of the REINFORCE algorithm with the Absolute value reward (the only successful reward with $\sigma_{\pi_\theta} = 0.001$) and by looking at this figure it is apparent that the total time response of it is inferior to the other PG algorithms by inheriting a large settling time. Note that, illustrated time responses are all one sample representative closed loop response for each individual algorithm.

5.2 Application of MCMC Method to a Nonlinear Model of an Inverted Pendulum

Given a nonlinear model of a continuous MDP which is here an Inverted Pendulum, objective is the stabilization problem of the Inverted Pendulum. To tackle this issue, a linear feedback controller with some controller gains corresponding to a state will be considered. The available gain tuning methods based on learning frameworks use gradient calculations to gradually update these gains either analytically or using system's trajectory information. In contrast to them, we handle the problem by taking a Bayesian inference approach which relies on MCMC sampling methods. The motivation originates from the PG RL setting where the goal is to search for optimal control policy parameters by maximizing the expected value of the immediate rewards during a given time horizon. The problem emerges when the evaluation of this expected return becomes a matter of challenge due to the intractability of the resulting integral calculation. To tackle this concern, we extend it to the parameter estimation in the Bayesian framework and build a posterior distribution with respect to the unknown parameters which is proportional to the expected returns. Consequently, rather than dealing directly with the expected return, we target the posterior distribution and attempt to draw samples (unknown policy parameters) from. This method essentially entails a PMMH algorithm which employs SMC (particle filtering) throughout its iterations.

5.2.1 The inverted pendulum model

The state space of the inverted pendulum comprises of four continuous elements as angle, angular velocity, position and velocity of the classical cart-pole system i.e. $s_t = (\phi_t, \dot{\phi}_t, p_t, \dot{p}_t)$. The action space $a \in A$ is of one dimensional and continuous. As a result of applying the action (control signal) a to the pendulum at state $s =$

$(\phi, \dot{\phi}, p, \dot{p})$, the nonlinear dynamics of the system are represented by the following equations:

$$\ddot{\phi} = \frac{-3ml\dot{\phi}^2 \sin\phi\cos\phi + 6(M+m)g\sin\phi - 6(a - b\dot{\phi})\cos\phi}{4l(M+m) - 3ml\cos\phi} \tag{5.10a}$$

$$\ddot{p} = \frac{-2ml\dot{\phi}^2 \sin\phi + 3mg\sin\phi\cos\phi + 4a - 4b\dot{\phi}}{4(M+m) - 3m\cos\phi} \tag{5.10b}$$

where the parameter setting for the model of the system is summarized in table 5.2.

The control aim to satisfy is to take the pendulum in the upright situation and

TABLE 5.2: Parameter setting for the inverted pendulum model

m	Mass of the Pole (kg)	0.5
M	Mass of the Cart (kg)	0.5
g	Gravitational Constant (m/s^2)	9.81
b	Friction Coefficient $(N/m \times s)$	0.1
l	Length of The Pole (m)	0.6

bring the cart to the origin from a given random state. Given a state s_t at time t, the action a_t is taken from a stochastic policy as Gaussian density $\mathcal{N}(\theta^T s_t, \sigma^2)$ with mean $\theta^T s_t$ and variance σ^2 such that the conditional distribution of the action is,

$$h_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(a - \theta^T s)^2}{2\sigma^2}\right\}. \tag{5.11}$$

Assuming a constant variance, the goal of policy search is to learn the parameter vector θ.

5.2.2 Reward structure and parameter setting for the algorithms

The performance of the proposed MCMC algorithm will be compared with algorithms discussed in Chapter 3, eNAC, REINFORCE and GPOMDP. For all the simulations, two different reward functions are used. The first reward function which we call an *Interval based reward* is a delayed reward expressed as:

$$r(a,s) = \begin{cases} 0, & -12^\circ \leq \phi \leq 12^\circ, \text{ and } -1.5 \text{ m} \leq p \leq 1.5 \text{ m} \\ -10, & \text{otherwise.} \end{cases} \tag{5.12}$$

The second one is a quadratic reward function defined in the following form:

$$r(a,s) = -s^T Q s - ca^2 \tag{5.13}$$

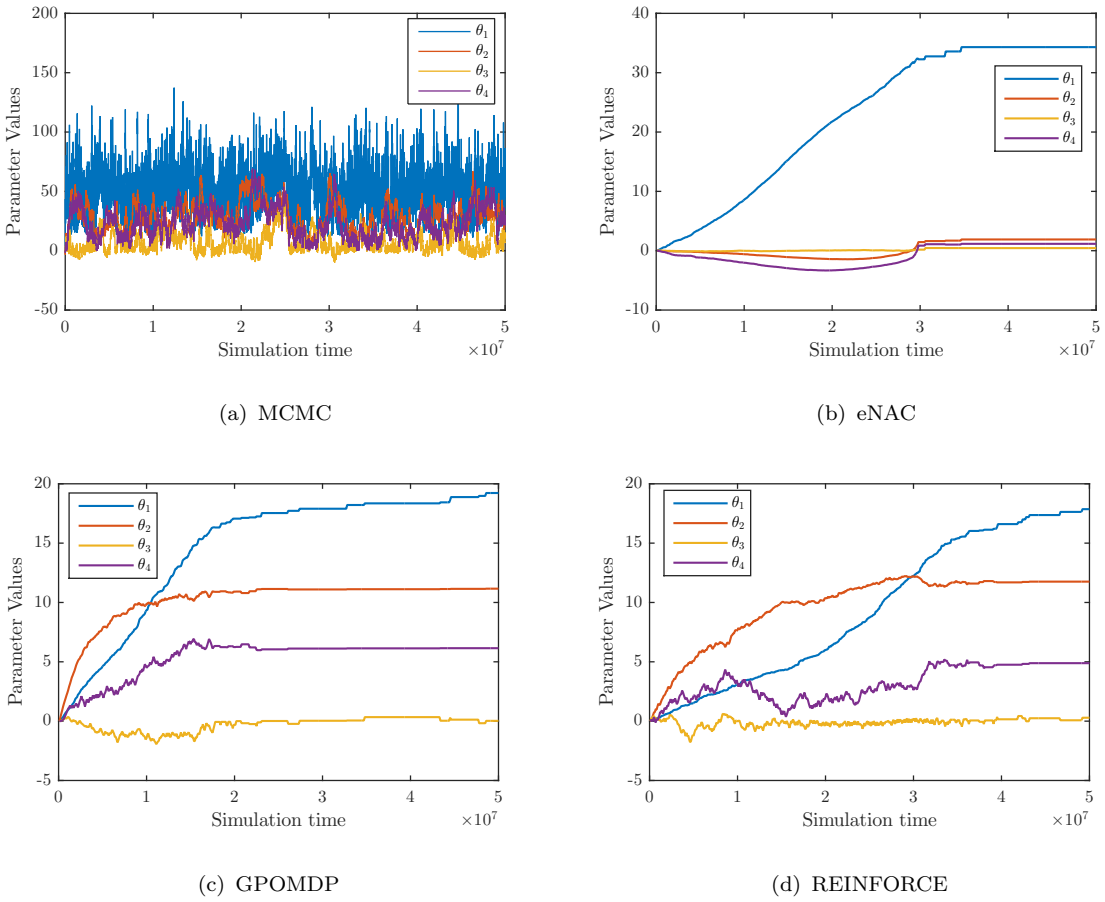
where Q is a positive definite matrix and c is a positive constant value. For this experiment Q is a diagonal matrix denoted as $diag([12, 1.25, 2, 0.25]^T)$ and $c = 0.01$. Selection of these values depend on the perceived importance of each state and is used to balance the contributions of the state and action to the reward function. Discounted value for the return function R is $\gamma = 0.99$. The length of simulations for all of them is $n = 1000$ time steps with a $0.01s$ sampling time. While the learning is in progress, the action is treated as a noisy one with an additive white Gaussian noise with a variance of $\sigma^2 = 4$; but, for testing the quality of the obtained policy, it is treated as a deterministic one which is noise-free.

PG algorithms (eNAC, REINFORCE and GPOMDP) run for 500 iterations with $H = 100$ episodes at each iteration to get an estimation of the policy parameter gradient. For the MCMC method in Algorithm 6, 50000 iterations are run, with $N = 1$ particle for each iteration. This choice of the number of iterations for the compared algorithms guarantees an equal computational load for them.

For the MCMC parameter updates, a Gaussian random walk is taken as a proposal density $q(\theta'|\theta) = \mathcal{N}(\theta'; \theta, \Sigma_q)$, where Σ_q is a diagonal covariance matrix with $diag([10, 1, 1, 1]^T)$. The uninformative prior distribution for parameter θ is a normal distribution $\mathcal{N}(0, 10^3 I_4)$ where I_4 is a 4×4 identity matrix. The importance of the choice of Σ_q for an amenable performance of the MCMC algorithm will be discussed in the upcoming sections where two different values (small and large) for Σ_q will be considered to observe how this selection can affect the algorithm.

5.2.3 Assess the proposed MCMC and PG algorithms with respect to different reward functions

Here, the comparison is made using the reward given in equation (5.12). The related policy parameter evolution plots (trace plots) are depicted in Fig. 5.6. It is obvious from the figure that, the PG methods are point-wise estimators which converge to optimal points in the parameter space, whereas the proposed MCMC algorithm targets the posterior distribution $\pi(\theta)$ and produces samples distributed according to this target distribution in a fashion that spaces with high expected rewards have been investigated. This phenomenon can be better understood form Fig. 5.7 where their histograms are given. To have an insight about the quality of the convergence speed of the compared algorithms, a set of simulations composed of 5 runs are carried out and approximated expected returns of them are averaged. Then, the performance of the MCMC is computed as the logarithm of the expected returns as $\log \widehat{J}(\theta)$. Resulting convergence plot is illustrated in Fig. 5.8 where the shaded areas reveal the standard deviation over the 5 runs.



(a) MCMC

(b) eNAC

(c) GPOMDP

(d) REINFORCE

FIGURE 5.6: Trace plots for MCMC and gradient based algorithms - Interval based reward

According to Fig. 5.8, performance plot in MCMC has converged rapidly in contrast to PG algorithms (Note the different horizontal axis ranges, by up to two orders of magnitude). It should be pointed out that $J(\theta)$ is assumed to be different from that of PG algorithms (for MCMC with an exponential utility function $U(x) = \exp(x)$ and for PG methods a linear utility function $U(x) = x$), which indicates the difference between the y-axes values for MCMC and PG methods. And finally, for the interval valued reward case, to capture an idea about the resulting time response quality by using the learned policy parameters, Integral Absolute Error

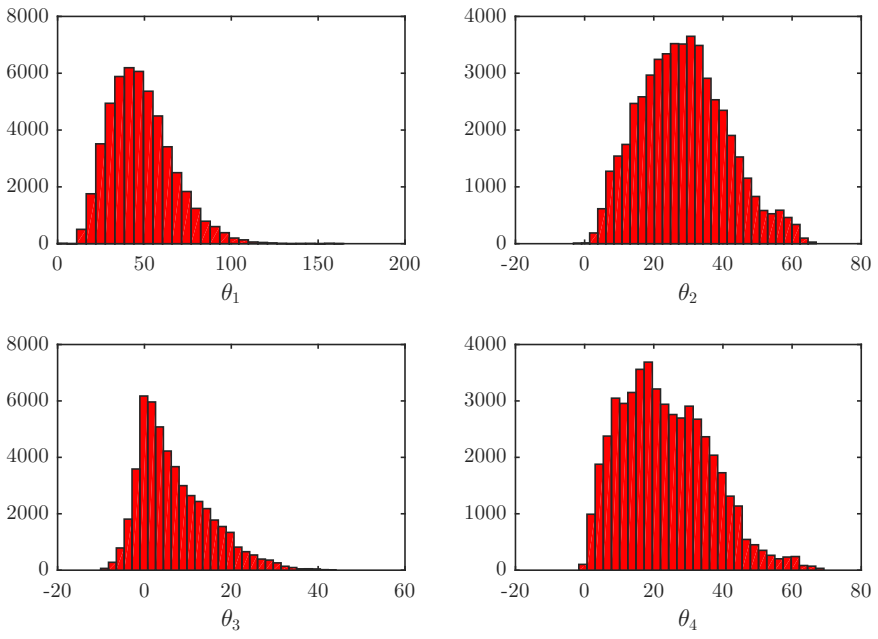
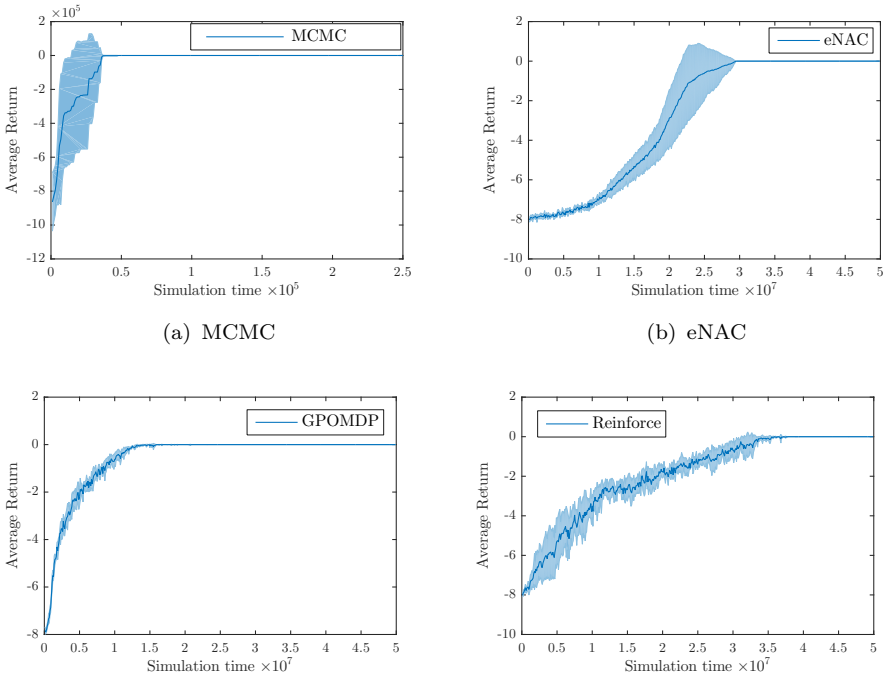


FIGURE 5.7: Histograms of MCMC policy parameter estimates - Interval based reward

(IAE) containing ϕ_t, p_t and a_t has been measured.

$$IAE = \int_0^\infty |e_t| dt \tag{5.14}$$

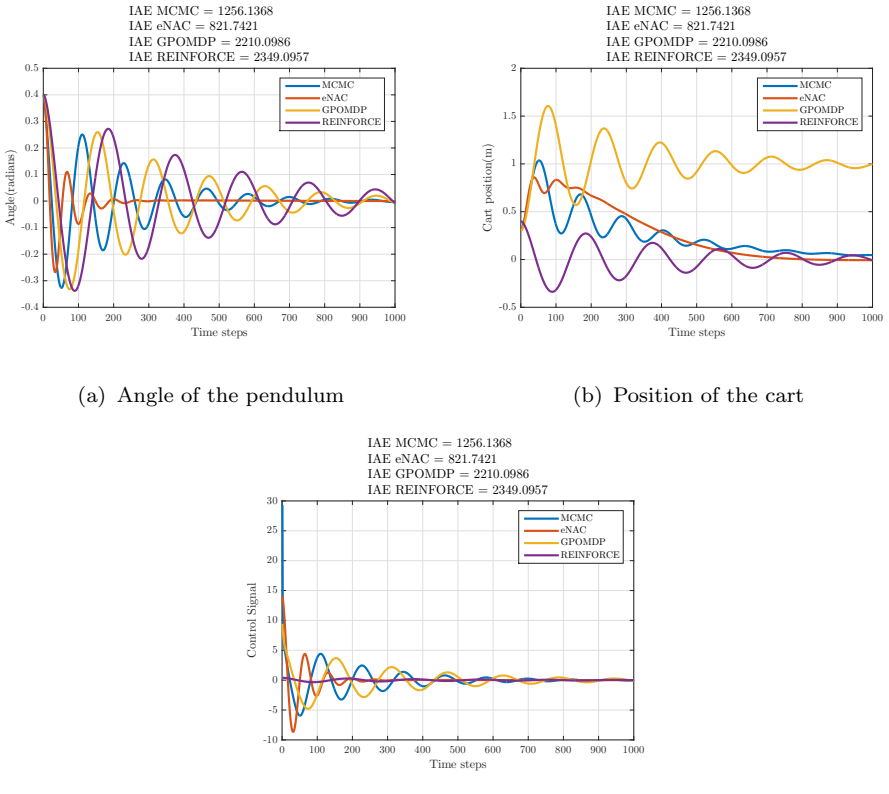
where e_t implies the resulting error between the desired values of ϕ_t, p_t and a_t and their actual values in the feedback control. As mentioned earlier since PG algorithms are point-wise estimators, in calculation of IAE term we used the parameters θ which are learned at the final iteration, whereas for MCMC method, it is averaged over the last quarter of the overall 50000 iterations, specifying that an approximation for the mean value of the target distribution $\pi_n(\theta)$ is considered. In Fig. 5.9 a test trial with the obtained policy parameters is shown for the time response of the given algorithms. Considering the IAE term, proposed MCMC algorithm performs better than GPOMDP and REINFORCE in the time domain but on the other hand, it can not defeat eNAC when an Interval based reward function is used.



(a) MCMC (b) eNAC

(c) GPOMDP (d) REINFORCE

FIGURE 5.8: Performance comparison with respect to convergence - Interval based reward

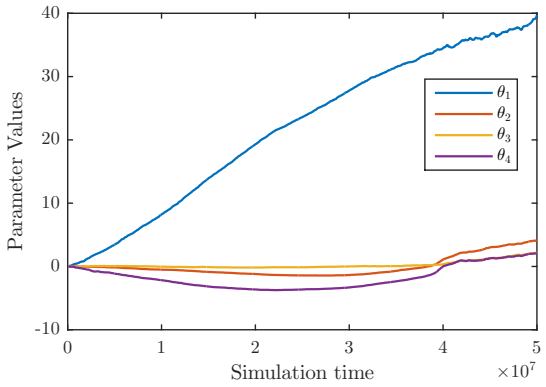


(a) Angle of the pendulum (b) Position of the cart

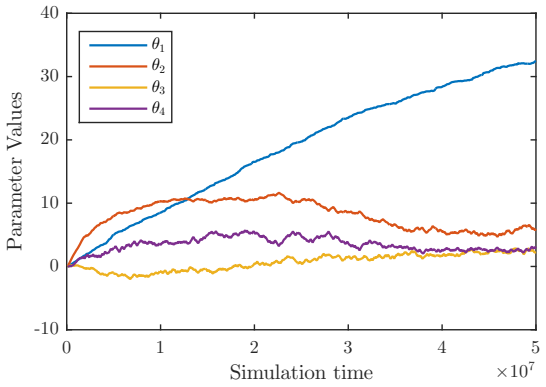
(c) Control signal

FIGURE 5.9: A sample time response for stabilization of the Cart-Pole - Interval based reward

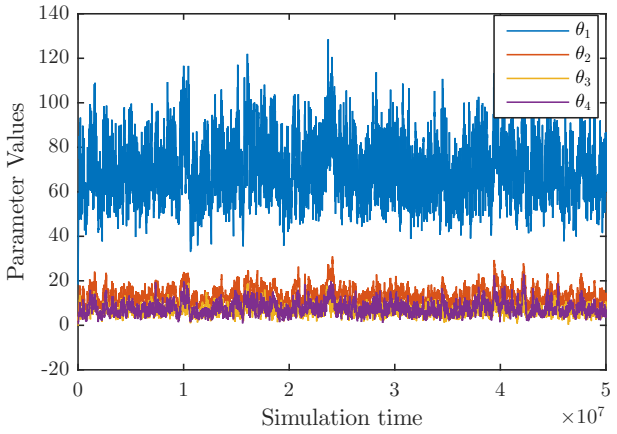
In the next part, the interval based reward function has been substituted with a quadratic one given in (5.13) and the simulations repeated. The reason for using a quadratic reward is that they are broadly utilized in learning the behavior of the dynamic systems. Likewise, the trace plots for MCMC and PG algorithms are depicted in Fig. 5.10, histograms for the generated samples from the posterior target distribution according to MCMC algorithm are presented in Fig. 5.11, approximated expected total rewards are given in Fig 5.12, and at last, a test trial displaying the time response specifications for the compared methods are provided in Fig. 5.13. It is worth noting that, for the quadratic reward case REINFORCE algorithm could not handle the problem of estimating a proper optimal solution for the policy parameters and due to this fact it is eliminated from the results. Similar to the conclusions derived from the interval based reward figures, the same interpretation is also valid for the quadratic reward, except as it is obvious from Fig. 5.13, MCMC outperformed eNAC and GPOMDP in terms of the IAE for the time responses. The control performance in terms of IAE is summarized in Table 5.3. Note that horizontal axes in part (a) for both Fig. 5.8 and Fig. 5.12 are in the same scale as other parts of these figures, however for a plain demonstration of the convergence of the performance plots of MCMC in the early stages of the iterations, we magnified the initial portion of the horizontal axes. Note that the type of the reward function affects the quality of the time responses of the control system. For instance, it is observed from Fig. 5.9 and Fig. 5.13 (where the former uses an interval based reward and the latter uses a quadratic one) that the employing a quadratic reward enhances the performance of the control time responses in terms of IAE by suppressing overshoots and decreasing the settling time since it presents better information to the agent about its performance.



(a) eNAC



(b) GPOMDP



(c) MCMC,

FIGURE 5.10: Trace plots for MCMC and gradient based algorithms - Quadratic reward

TABLE 5.3: Performance of the proposed MCMC algorithm in comparison to PG methods in terms of IAE.

	IAE for given methods			
Reward Type	MCMC	eNAC	GPOMDP	REINFORCE
Interval based	1256.1368	821.7421	2210.0986	2349.0957
Quadratic	378.1922	418.8248	454.8195	NA$^{(*)}$

$^{(*)}$For Quadratic reward, REINFORCE did not converge to an optimal solution.

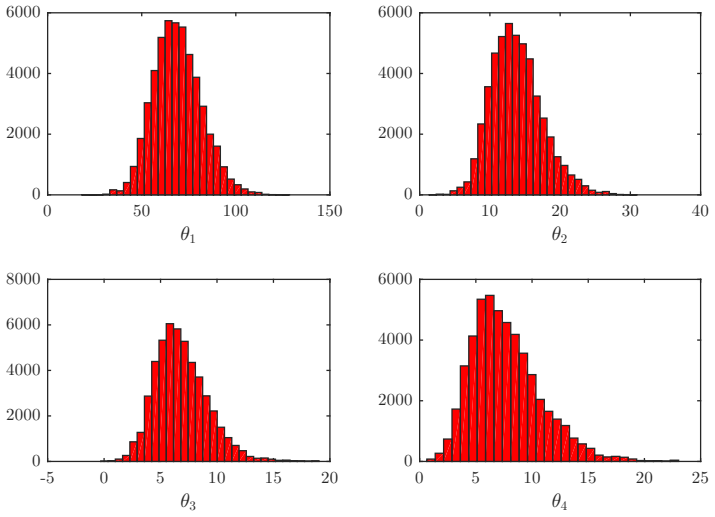
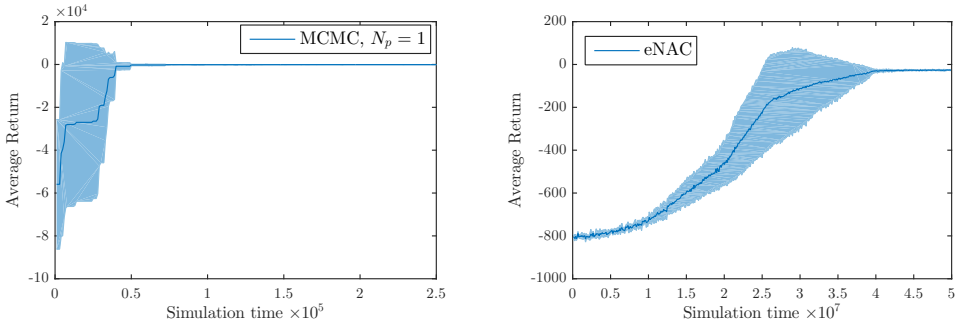
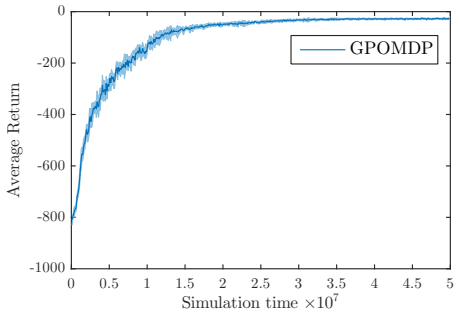


FIGURE 5.11: Histograms of policy parameter estimates for MCMC - Quadratic reward



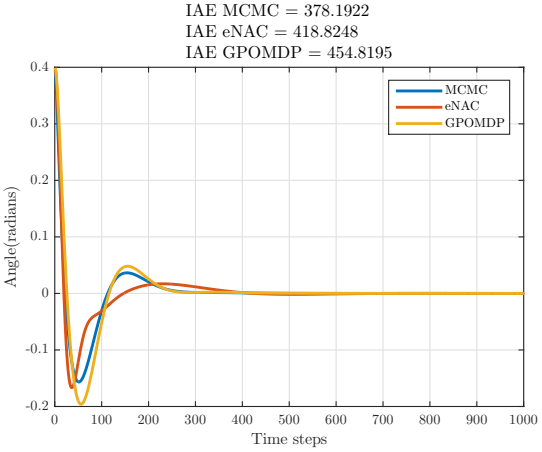
(a) MCMC

(b) eNAC

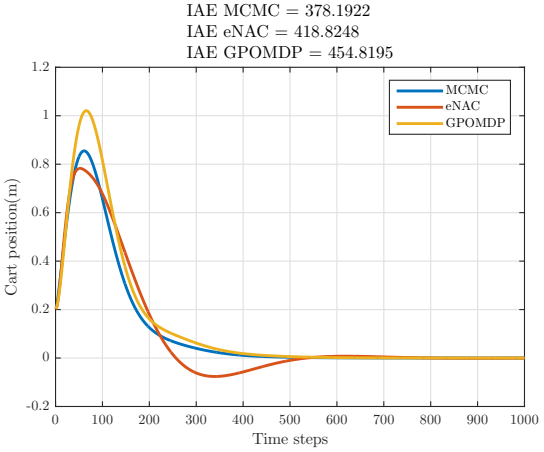


(c) GPOMDP

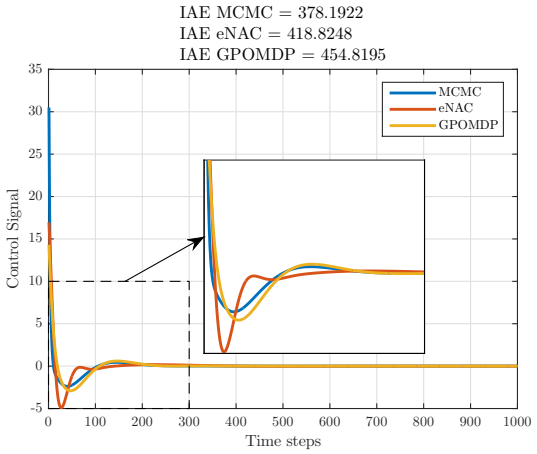
FIGURE 5.12: Performance comparison with respect to convergence - Quadratic reward



(a) Angle of the pendulum



(b) Position of the cart



(c) Control signal

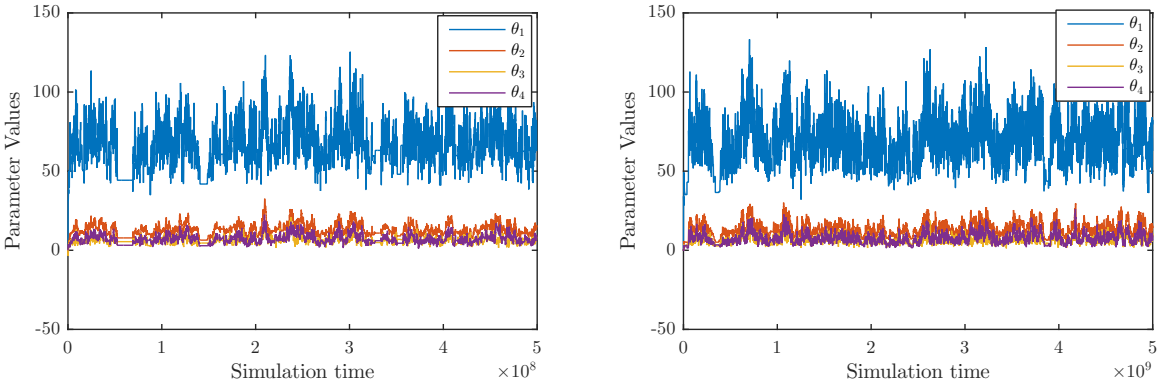
FIGURE 5.13: A sample time response for stabilization of the Cart-Pole - Quadratic reward

5.2.4 MCMC performance regarding adjustable parameters

In order to have an insight about how the performance of the proposed MCMC algorithm can be modified with some tunable parameters such as number of particles in the SMC or covariance matrix of the proposal distribution, we have done two more experiments which show the effects of these elements.

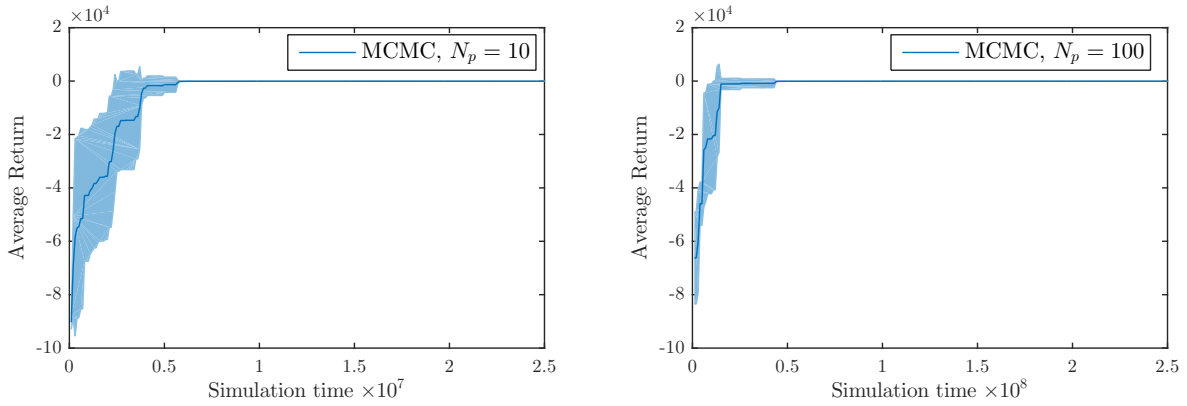
5.2.4.1 MCMC performance using different number of particles

All the previous results for the parameter estimation of the control policy have done considering the number of particle in the SMC algorithm as one i.e., $N_p = 1$. One alternative for boosting the performance of the MCMC algorithm is to use SMC with number of particles more than one. For this situation, simulations based on number of particles as $N_p = 10$ and $N_p = 100$ are carried under a quadratic reward function. The trace plots, overall return and learned policy parameters as histograms are represented in Fig. 5.14, 5.15 and 5.16, respectively. It is obvious from the generated results that for $N_p = 100$ MCMC algorithm turns out to have smoother histogram distributions. Despite this, convergence of the algorithm gets slow for $N_p = 100$ owing to a ten-fold growth in calculation time for each iteration, as it can be observed from Fig. 5.15.



(a) MCMC with $N_p = 10$ (b) MCMC with $N_p = 100$

FIGURE 5.14: Parameter update for MCMC with quadratic rewards regarding different number of particles



(a) MCMC with $N_p = 10$

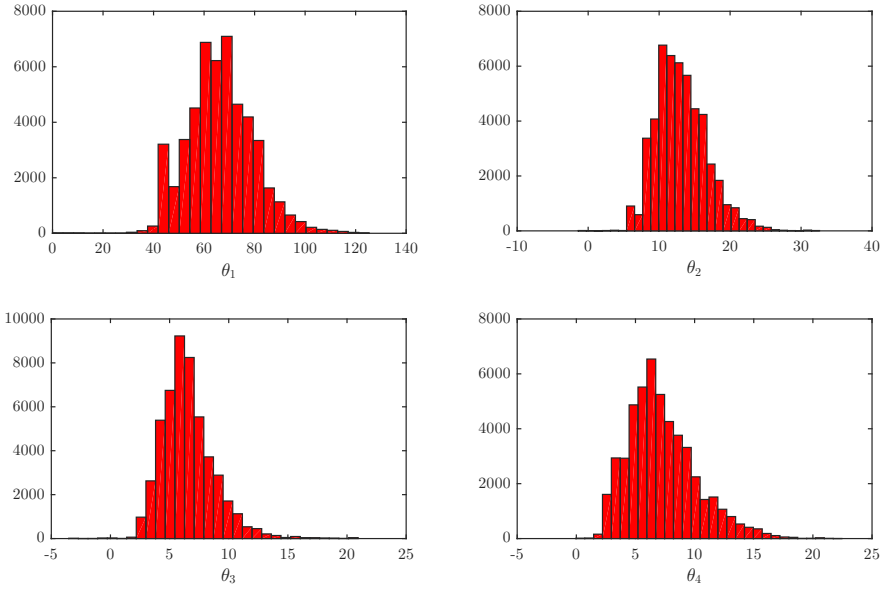
(b) MCMC with $N_p = 100$

FIGURE 5.15: Performance for MCMC with quadratic rewards regarding different number of particles

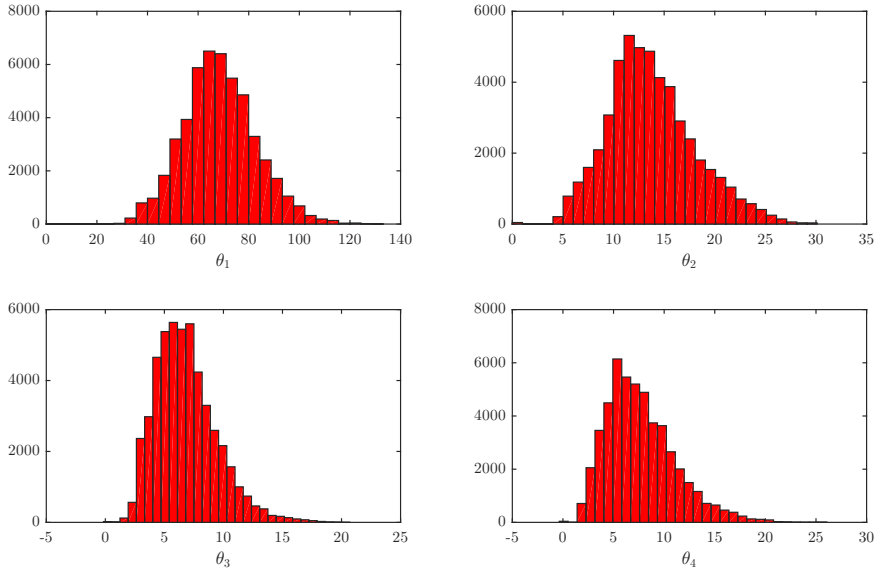
5.2.4.2 MCMC performance using different proposal covariance Σ_q and action variance σ^2

Samples drawn from the posterior distribution $\pi(\theta)$ using the MCMC algorithm should be in such a way that they mix well so that they do not stick to their previous values. Particularly, the choice of a suitable proposal variance Σ_q to guarantee well-mixing is important. Small values of this quantity Σ_q will cause samples which are closely correlated, thus mixing so slowly. On the other hand, large values for Σ_q makes the chain susceptible to remember its older parameters via dismissing the newly proposed ones. Moreover, the role of the action variance (as a Gaussian white noise) will be inspected. In the following, this situation will be clarified through simulations. As before, the quadratic reward function has been used. We take some test values for the tuple (σ, Σ_q) and consider their probable combinations:

- $\sigma_1 = 2.5$, $\sigma_2 = 1.5$ and
- $\Sigma_{q_1} = \text{diag}([100, 10, 10, 10])$, $\Sigma_{q_2} = \text{diag}([1, 0.1, 0.1, 0.1])$



(a) MCMC with $N_p = 10$



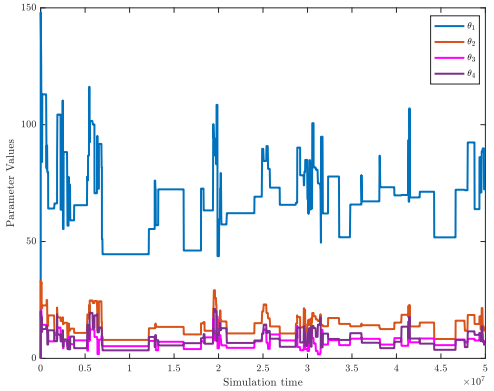
(b) MCMC with $N_p = 100$

FIGURE 5.16: Histogram of the estimated parameters for MCMC with quadratic rewards regarding different number of particles

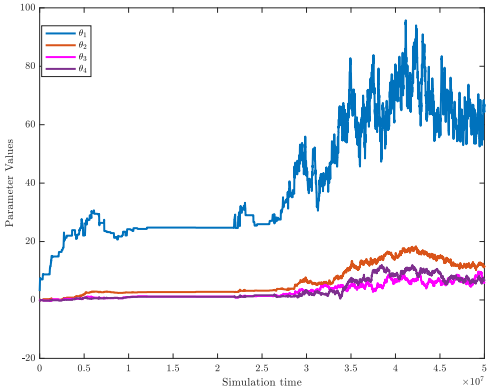
Obtained simulation results due to the above mentioned sets are shown in Fig. 5.17 and Fig. 5.18. If one compares the trace plots of these figures with that of given in Fig. 5.10(c), it is seen that a suitable selection of Σ_q in a way that it is neither very large nor very small, just like Fig. 5.10(c), leads to a well-mixed sample trace where it converges to the true target distribution. To be specific, part (a) of Fig. 5.17 and Fig. 5.18 imply that the large values for Σ_q may cause the rejection of the proposed candidate parameters for the policy and the provided samples would be reluctant to adopt new parameters. Similarly, in part (b) of these figures, the generated samples have a tendency to mix slow and therefore are highly correlated due to small Σ_q.

It is also notable that, in spite of poor choices of Σ_q, the proposed MCMC algorithm still can get an estimate for the policy parameter and manages to converge. The convergence performance plots for these cases are illustrated in parts (c) and (d) of Fig. 5.17 and Fig. 5.18.

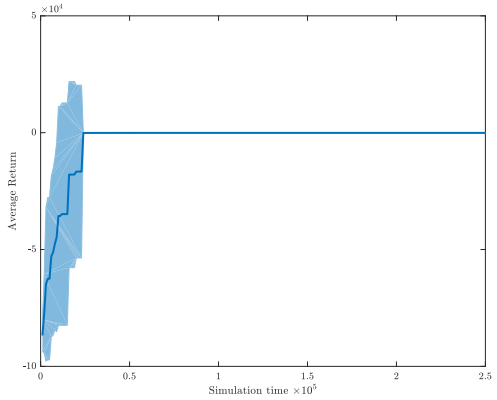
Lastly, the drawn conclusions are valid for the selected values of the action noise σ, meaning that the proposed MCMC algorithm entails a certain degree of robustness. But not to forget that, large values for the action noise are not tolerable since they may cause the system to become unstable despite an acceptable policy.



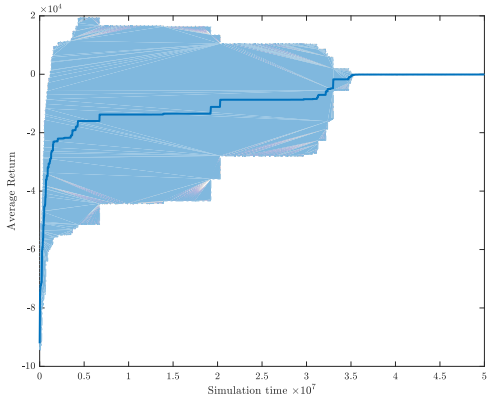
(a) Trace plots generated with a large Σ_q



(b) Trace plots generated with a small Σ_q



(c) Performance of the MCMC with a large Σ_q



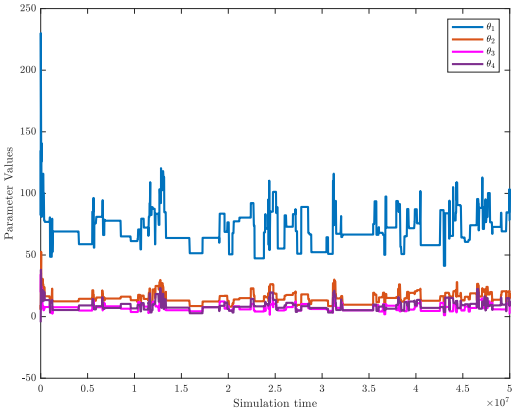
(d) Performance of the MCMC with a small Σ_q

FIGURE 5.17: Trace plots and average returns for MCMC using both small and large values of Σ_q with the quadratic reward when $\sigma = 2.5$

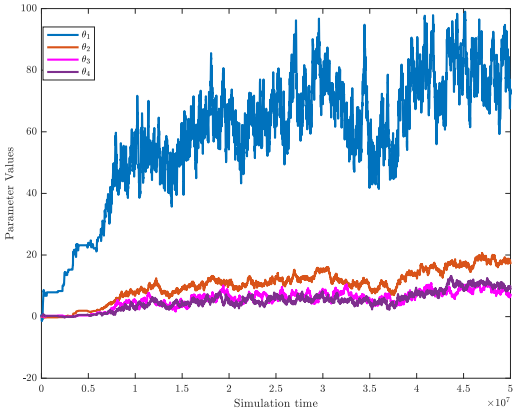
5.3 Trajectory Control of a Robotic Manipulator via Bayesian MCMC Method

In this section the focus is on the policy search concept based on the simplified version of the Bayesian MCMC algorithm. By the term simplified, we mean that the general form of the proposed MCMC algorithm, which uses particle filters in

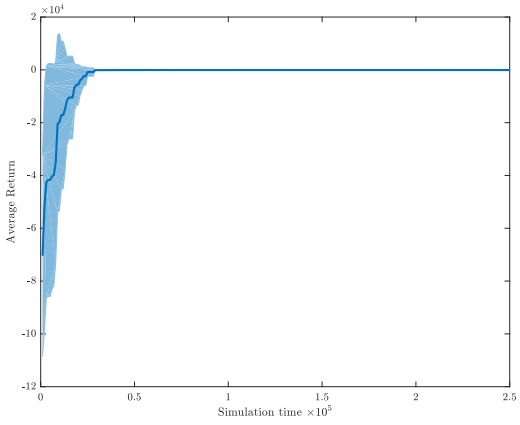
its iterations, is modified to a formation which is amenable for applying to real-time physical systems by excluding the particle filters from its structure. Although here we intend to demonstrate the outcomes of a simulation based comparative study between eNAC, gradient based Adaptive PD and MCMC algorithms, later we will use this simplified version of the MCMC algorithm to perform a real-time experiment.



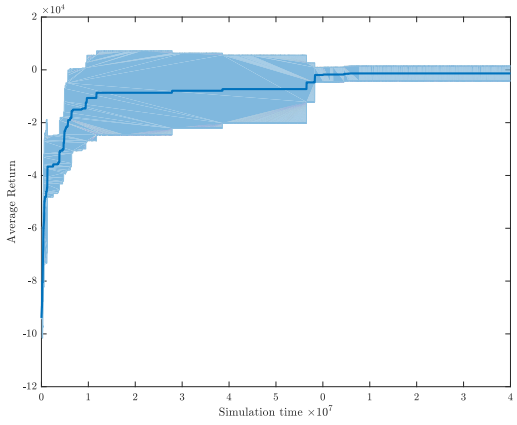
(a) Trace plots generated with a large Σ_q



(b) Trace plots generated with a small Σ_q



(c) Performance of the MCMC with a large Σ_q



(d) Performance of the MCMC with a small Σ_q

FIGURE 5.18: Trace plots and average return for MCMC using small and large values of Σ_q with the quadratic reward when $\sigma = 1.5$

Therefore, our goal is to verify the algorithm on the simulation environment and then validate it on physical systems. The system of our interest is a 2-Degree of Freedom (DoF) planar manipulator which is composed of nonlinear dynamics. Since we have already covered the subjects related to PG RL and Bayesian MCMC, we commence with a brief description of the Adaptive PD method, then give a model of the 2-DoF planar manipulator and conclude this section by expressing the obtained results.

5.3.1 Gradient based Adaptive PD method

Steepest descent methods (gradient descent) can be considered as a well-known optimization approach for dealing with dynamical systems. Despite this fact, these methods might not be an appropriate alternative for systems with complex dynamics. The reason is that they are bound to fall in local optimal points. Moreover, their performance can be highly affected by poor choices of the initial parameters which are going to be updated by gradient rules. Nevertheless, some successful applications of these methods to the dynamical systems can be found, for example WANG et al. (2007) and Liu (2007). The control problem for our setting is to follow a given reference trajectory. Here, the cost function of the adaptive PD algorithm can be constructed as:

$$J = \frac{1}{2}\left(e^{\mathrm{T}}e\right) \tag{5.15}$$

where e stands for the error between the given reference signal and the actual trajectory. Based on the gradient descent, parameter update rule is

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} g^{(t)} \tag{5.16}$$

where $g^{(t)}$ is the control signal in each time step and η is a learning rate. The Pseudo-code of the adaptive gradient based PD algorithm for our system is given in

Algorithm 9.

Algorithm 9: Pseudo-code for adaptive PD

Input: Number of time steps n, initial controller gain vector $\theta^{(0)}$, values for the stopping criteria(δ, ϵ), step size for the gradient rule $\eta = 0.001$
Output: Control signal g, $\theta^{(t)}$
for *t = 1, 2, ...* **do**
  Step 1: Calculate gradient vector of the control signal $\nabla_\theta g^{(t)}$
  Step 2: Calculate new controller gain $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta g^{(t)}$
  Step 3: Calculate new control signal g
  Step 4: Calculate stopping critaria
  if $\mid J^{(t+1)} - J^{(t)} \mid < \epsilon$ **then**
    terminate algorithm, **return** g, θ
  end
  if $\mid e^{(t+1)} - e^{(t)} \mid < \delta$ **then**
    terminate algorithm, **return** g, θ
  end
end

5.3.2 Structure of the two link planar manipulator

The two link planar manipulator's mechanism, which is generally known as Pantograph, is shown in Fig. 5.19. It consists of five links: the fixed link N which its length can be neglected and symmetric links l_1, l_2 and l_3, l_4 ($l_1 = l_2, l_3 = l_4$). The links l_1 and l_2 are actuated with torques τ_1 and τ_2 generated by two motors connected to these links. The end-effector is located at the conjunction of l_3 and l_4 and is denoted as point Q. Controlled system outputs are x_Q and y_Q position of end-effector with respect to center of workspace (cw), which is an origin point for initialization of the system. Control inputs of the system are τ_2 and τ_1. Generally speaking, equation of motion for a mechanical system can be described by *Lagrangian equations* as:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_\mathrm{i}} - \frac{\partial L}{\partial q_\mathrm{i}} = \tau_\mathrm{i} \tag{5.17}$$

Here L is the Lagrangian function which is the difference between the kinetic K and potential energy P as $L = K - P$, τ_i is the torque and q_i is the angle with respect to the x coordinate. A complete modeling procedure and derivation of the equations for the planar manipulator can be reached in Yu (2006).

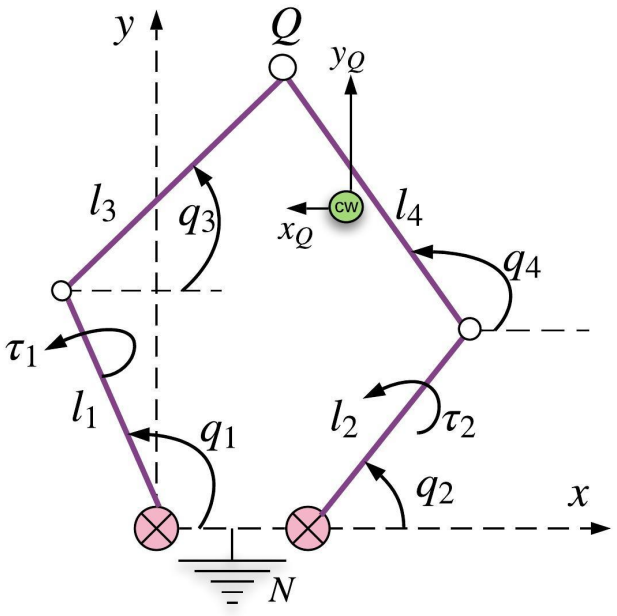


FIGURE 5.19: schematic representation of the planar manipulator

The main objective here is to learn an optimal control policy which enables the system to track a desired trajectory. In this direction a PD control is used. The given reference signal is assumed to be a circular one with a radius of 20 mm. The error signal is defined as the difference between the desired path and the actual output trajectory. So our policy is composed of the parameters of the PD controller in the $x - y$ direction:

$$\theta = \begin{bmatrix} k_{d_x} & k_{p_x} & k_{d_y} & k_{p_y} \end{bmatrix}$$

For the eNAC algorithm, the action is drawn from a stochastic policy which is here a Gaussian distribution as $\mathcal{N}(\theta^T s, \sigma^2)$ where the state of the system s is $(e, \dot{e})$ in the $x - y$ axis.

5.3.3 Numerical quantities of the compared algorithms for simulations

For MCMC and eNAC algorithms a quadratic reward function is hired as following:

$$r(a,s) = \begin{cases} -s^T Q s - Ca^2, & -150\text{mm} \leq x \leq 150\text{mm}, \text{ and } -25 \text{ mm} \leq y \leq 70 \text{ mm} \\ -10^8, & \text{otherwise;} \qquad \text{(Applied just for MCMC)} \end{cases} \tag{5.18}$$

where $Q = diag([1, 1000, 5, 1000])$ and $C = diag([10^{-5}, 10^{-5}])$ are weight matrices for the states and actions. Discounted factor for the return function is $\gamma = 0.99$. The proposal distribution for the MCMC algorithm is a zero mean Gaussian $\mathcal{N}(\theta'; \theta, \Sigma_q)$ with the covariance matrix $\Sigma_q = diag([1, 20, 1, 20])$. The uninformative prior distribution over parameters is $\mu(\theta) = \mathcal{N}(0, diag([500, 10000, 500, 10000]))$. For the eNAC algorithm parameterized policy is also assumed to be a Gaussian one with $\sigma = 158$. The simulation duration of system is $5s$ with a sampling rate of $0.001s$. Number of iterations for the eNAC is 3000 including 15 episodes for each of them. For the MCMC, we iterate the algorithm for 5000 iterations but since the trend of the trace plots does not change, similar to eNAC, we illustrated the results up to 3000^{th} iteration. By looking at Fig. 5.20 and Fig. 5.21, where the convergence of the MCMC and eNAC are shown based on total expected reward, it is observed that MCMC's performance goes beyond that of eNAC owing to its successful convergence happening near the 100^{th} iteration, while eNAC succeeded it around the 1500^{th} iteration. Note that the convergence plots are obtained by averaging the results over 5 different experiments.

The trace plots of the learned parameters for MCMC, eNAC and Adaptive PD are depicted in figures 5.22, 5.23 and 5.24, respectively. As mentioned earlier, obtained parameter vector from eNAC and Adaptive PD approaches an optimal

point at the end of iterations, but for the MCMC it expresses a distribution where one can draw parameters from. For our case, we consider the mean of the last quarter of the learned samples. In Table. 5.4 these parameter values are given. To get an idea of the drawn samples (last quarter) as policy parameters, a histogram representation is shown in Fig. 5.25.

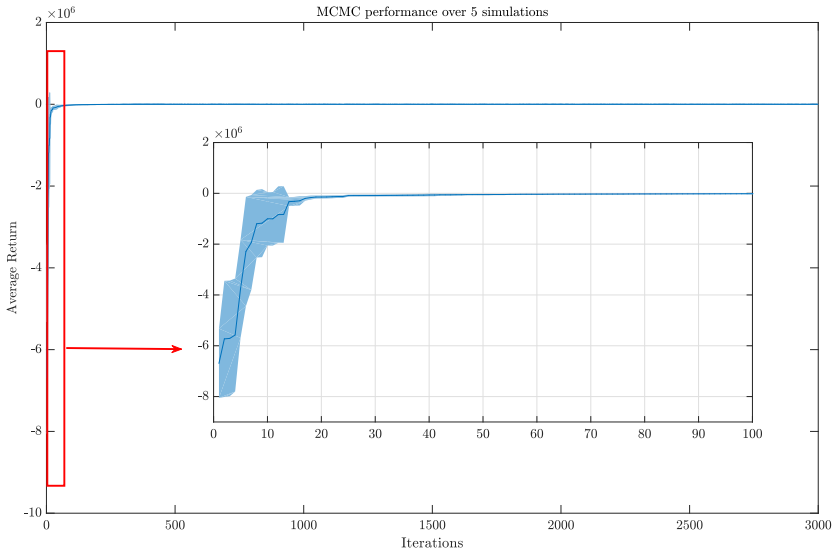


FIGURE 5.20: MCMC average return for 2-D manipulator.

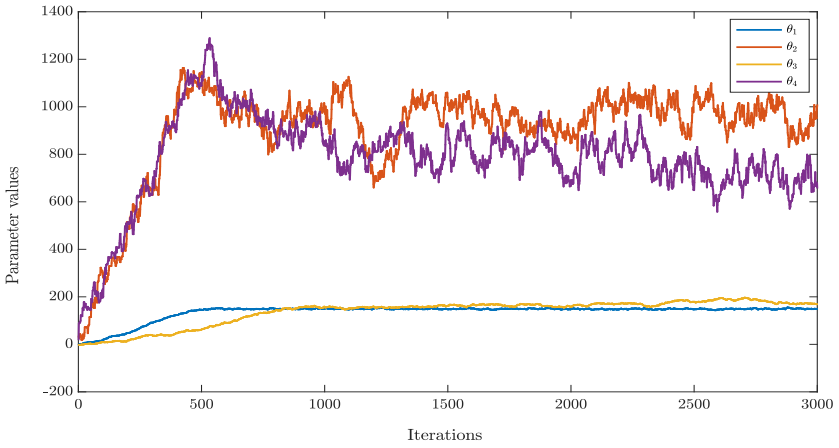


FIGURE 5.22: Trace plots for MCMC, 2-D manipulator

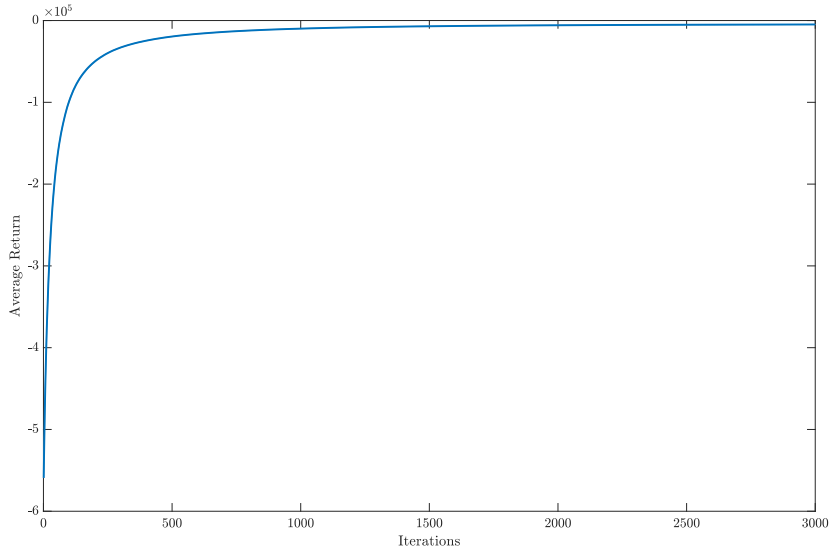


FIGURE 5.21: eNAC average return for 2-D manipulator.

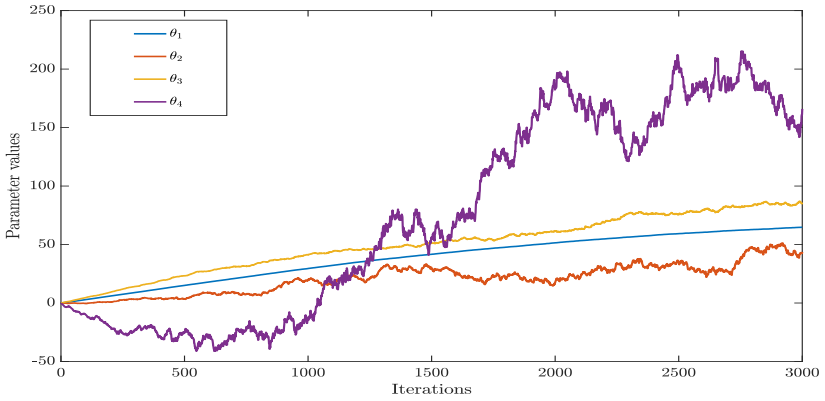


FIGURE 5.23: Trace plots for eNAC, 2-D manipulator

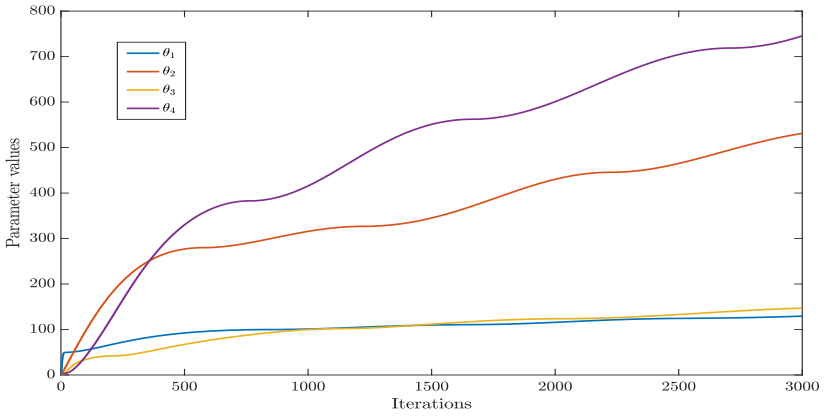


FIGURE 5.24: Trace plots for adaptive PD, 2-D manipulator

TABLE 5.4: Policy parameters learned for the 2-DoF planar manipulator.

	θ_1	θ_2	θ_3	θ_4
MCMC	149.37	955.15	185.78	722.95
eNAC	65.00	4.16	100.77	221.62
Adaptive PD	129.41	531.18	146.92	745.77

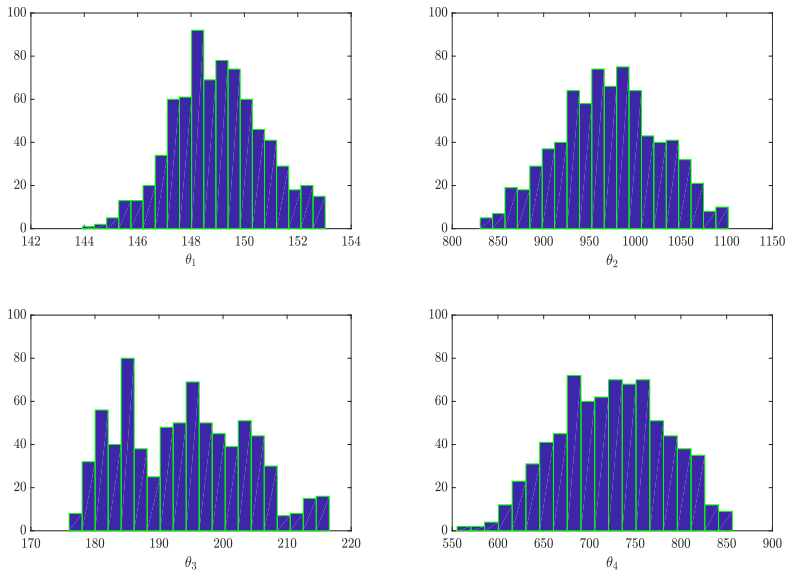


FIGURE 5.25: Histograms of policy parameter estimates for MCMC (last quarter).

For comparing the performance of the algorithms in the time response, their trajectory for the error in $x-y$ axis by using the learned parameters are demonstrated in figures 5.26 and 5.27 where in both x and y domains MCMC inherits the best performance (least error) in contrast to the other two. It is also seen that its behavior is very similar to Adaptive PD in the y axis. The main reason for comparing our method with Adaptive PD, which is a famous classical control method, is to verify MCMC's validity. To distinguish between the given algorithms from the error point of view, Integral Absolute Error (IAE) has been used as a metric and summarized in Table 5.5. Finallly, both the circular reference and the actual trajectory for the given algorithms are shown in Fig. 5.28.

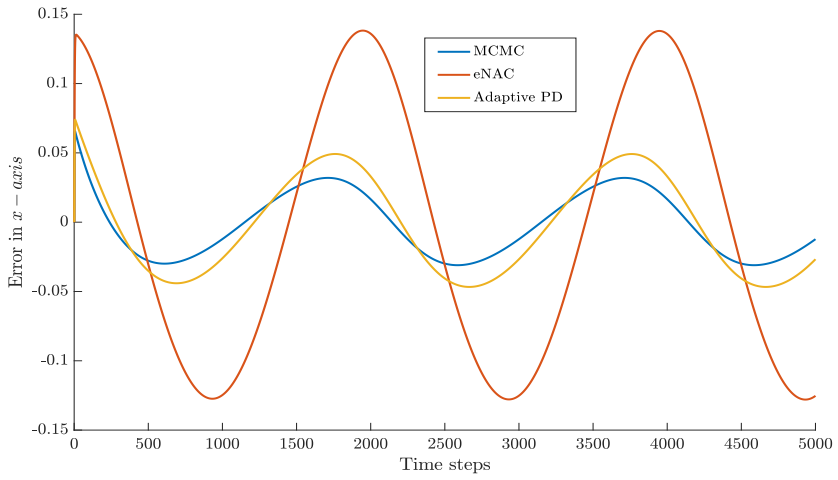


FIGURE 5.26: Error trajectory in the x-axis.

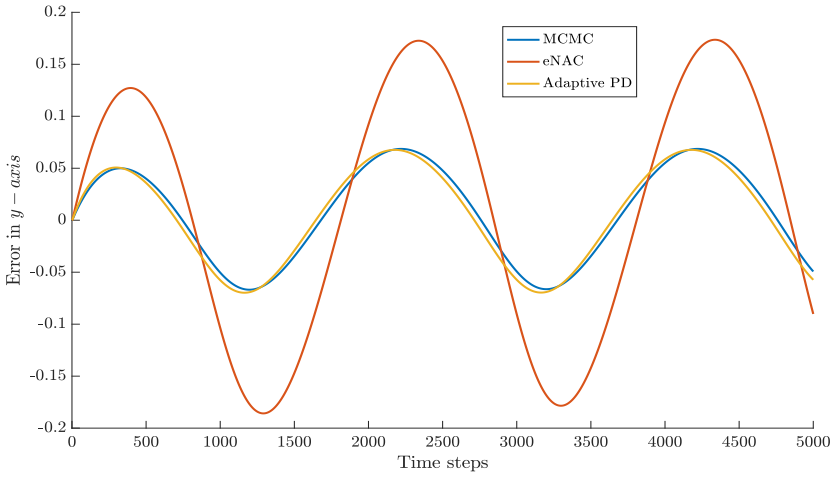


FIGURE 5.27: Error trajectory in the y-axis.

TABLE 5.5: IAE comparison for the given algorithms.

	MCMC	***eNAC***	***Adaptive PD***
IAE	0.2457 mm	0.7454 mm	0.2842 mm

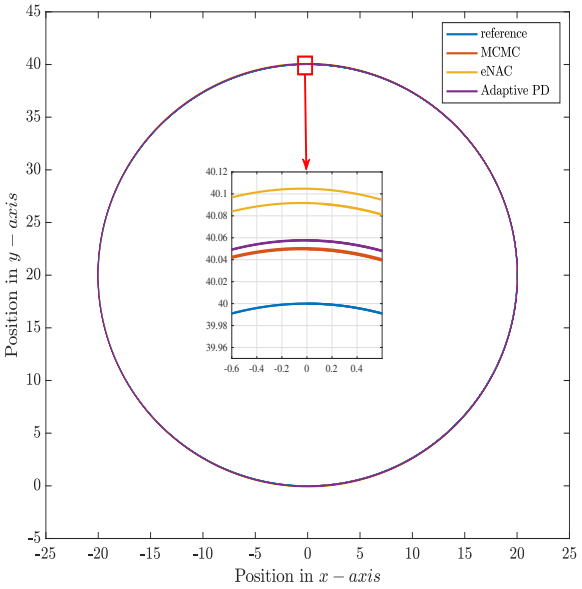


FIGURE 5.28: Circular reference and actual trajectory.

5.4 Policy Search for the Control Problem of a Ballbot via an Adaptive MCMC Algorithm

The main concept of a Ballbot system which its physical model is shown in Fig. 5.29 is similar to the inverted pendulum where a body is driven on a rolling ball. The ball itself is rotated by three motors mounted on the body. The model of the Ballbot system includes various set of nonlinear ordinary differential equations and composed of thousands of terms. Considering this higly nonlinear model, our aim is to stabilize the body of the system on the moving ball. To capture this control goal, a linear quadratic regulator (LQR) feedback controller is used. But the challenge here is that to be able to use an LQR controller the system is required to be linearized. Our attempt is to deal with the nonlinear model of the Ballbot system while using a feedback controller. For this purpose, we will apply an Adaptive MCMC algorithm to learn the gains of the feedback controller. We extend the idea

of the MCMC algorithm to an Adaptive form due to the fact that the number of the gain parameters to be learned are high.

Since the derivations of the equations of motion for the Ballbot system is out of the scope of this thesis, we just refer the readers to some successful works focused on modeling the system such as Bonci (2016), and Fankhauser and Gwerder (2010).

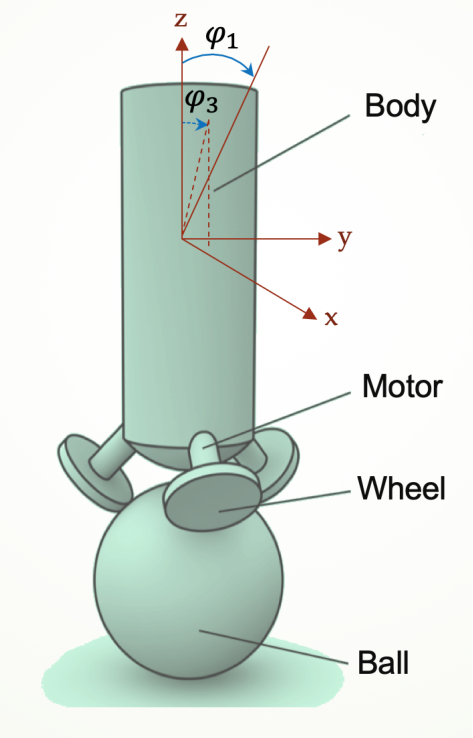


FIGURE 5.29: Model of a Ballbot system.

5.4.1 Parameter settings for the simulations

The model of the ballbot system that we have used in our simulations composed of ten continuous state spaces but the states that we are going to control are the angle of the body in the y and x directions and their velocities which are respectively denoted as $(\varphi_1,\ \dot{\varphi}_1,\ \varphi_3,\ \dot{\varphi}_3)$. The action space is three dimensional and continuous which consists of torques for three motors (τ_1, τ_2, τ_3). To stabilize the system we use

a feedback controller with 30 gain parameters which are going to be learned by the proposed Adaptive MCMC algorithm. The corresponding parameter gain matrix for the feedback controller is as following where the index i and j signify the actions and states, respectively.

$$\theta_{i,j} = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & k_{1,4} & k_{1,5} & k_{1,6} & k_{1,7} & k_{1,8} & k_{1,9} & k_{1,10} \\ k_{2,1} & k_{2,2} & k_{2,3} & k_{2,4} & k_{2,5} & k_{2,6} & k_{2,7} & k_{2,8} & k_{2,9} & k_{2,10} \\ k_{3,1} & k_{3,2} & k_{3,3} & k_{3,4} & k_{3,5} & k_{3,6} & k_{3,7} & k_{3,8} & k_{3,9} & k_{3,10} \end{bmatrix}$$

For the Adaptive MCMC algorithm a quadratic reward function is used as:

$$r(a,s) = \begin{cases} -s^T Q s - C a^2, & -20^\circ \leq \phi_1, \phi_3 \leq 20^\circ, \\ -10^8, & \text{otherwise;} \end{cases} \tag{5.19}$$

The weight matrix for the states is a positive definite one with ten elements in its diagonal as $Q = diag([1000, 50, 100, 50, 20, 10, 20, 10, 20, 10])$ and $C = 0.001$ for each element of action space. The discounted factor for the return function is $\gamma = 0.99$. For the Adaptive MCMC, the covariance matrix Σ_q of the proposal distribution which is a zero mean Gaussian $\mathcal{N}(\theta'; \theta, \Sigma_q)$, is considered to be fixed for 5000 iterations as pre-run, where a total of $50,000$ iterations is run. After a pre-run is completed the covariance matrix of the proposal distribution is modified accordingly. We iterate the learning procedure for our simulations with a run of the Ballbot system for $5s$ with a time step of $0.01s$. For a successful sample run, the trace plot of the last quarter of the learned parameters is shown in Fig. 5.30 where the first four at the top left correspond to our targeted states.

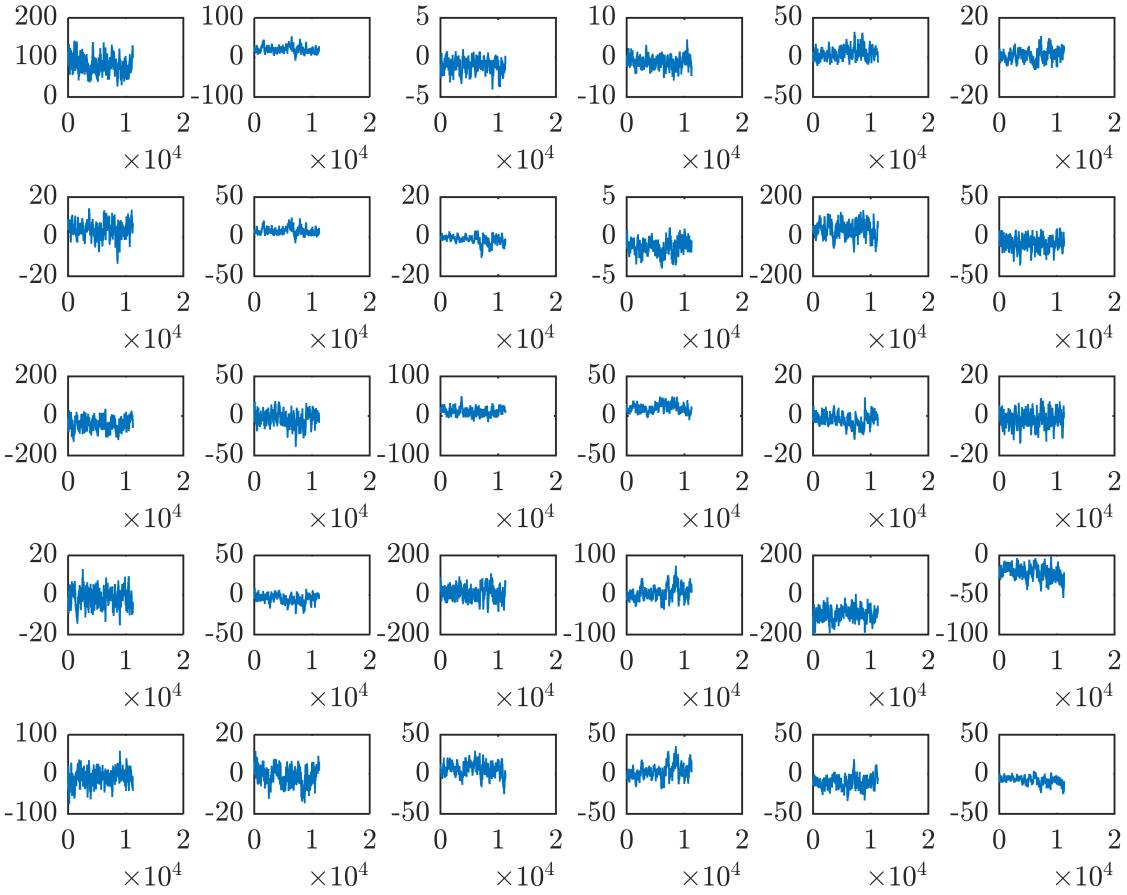
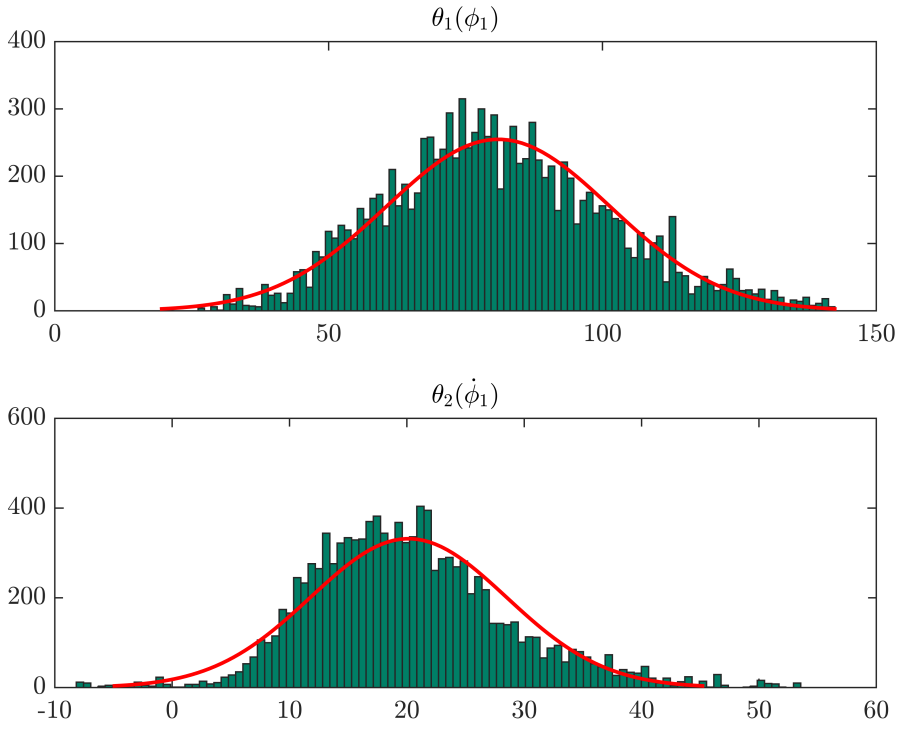
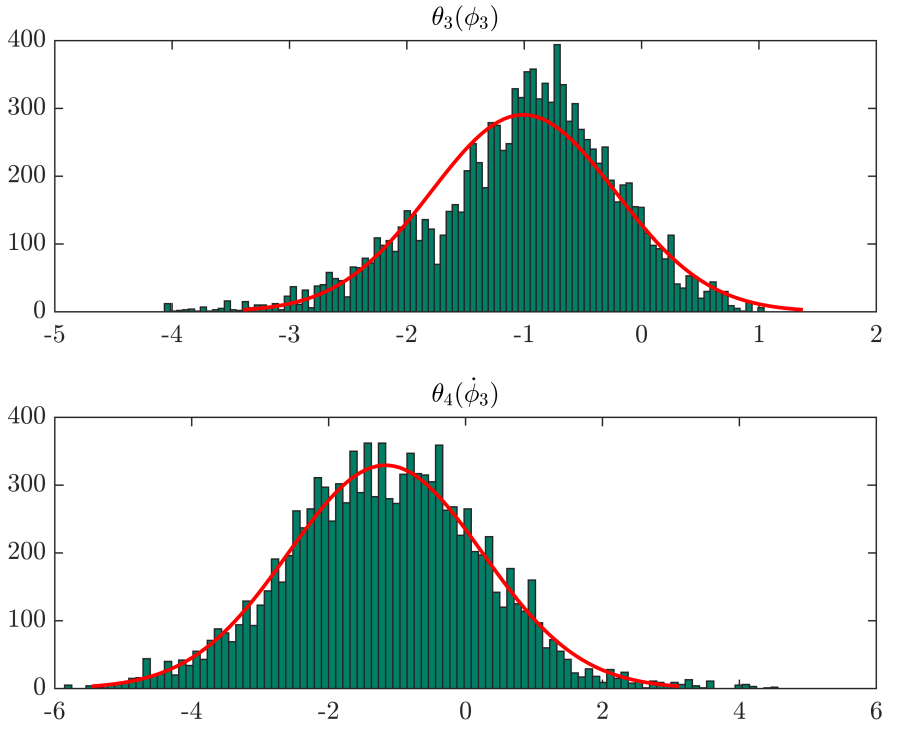


FIGURE 5.30: Trace plots of the adaptive MCMC for the Ballbot system.

To have a deeper insight of the quality of the learned parameters, we draw their histograms for the first four parameters and give them in Fig. 5.31. Furthermore, the aimed learned parameters as the angles of the body with respect to their angular velocities are presented as learned policies in Fig. 5.32 and Fig. 5.33.



(a) Histograms for φ_1 and $\dot{\varphi}_1$



(b) Histograms for φ_3 and $\dot{\varphi}_3$

FIGURE 5.31: Histograms of the adaptive MCMC for Ballbot system.

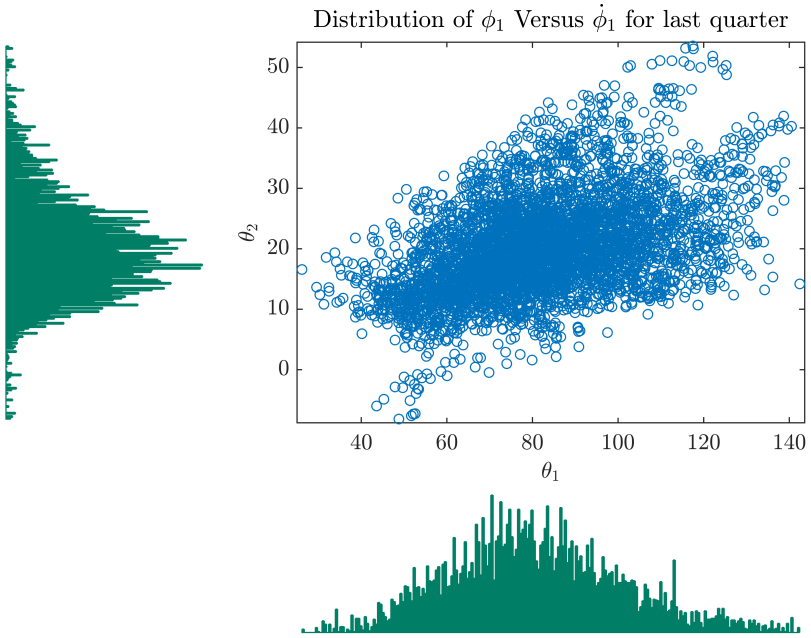


FIGURE 5.32: Learned policy (θ_1, θ_2) with adaptive MCMC for the Ballbot system.

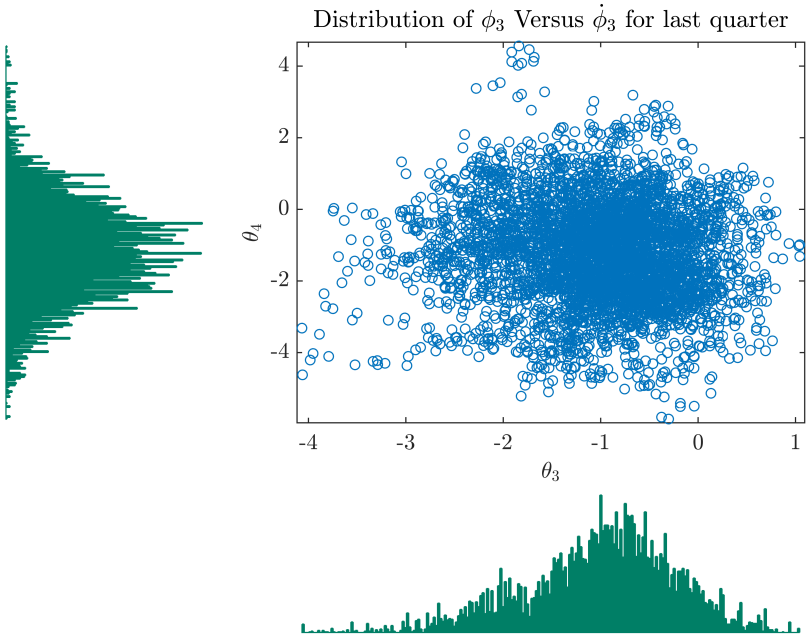


FIGURE 5.33: Learned policy (θ_3, θ_4) with adaptive MCMC for the Ballbot system.

The performance of the adaptive MCMC algorithm in terms of convergence is depicted in Fig. 5.34 in a logarithmic scale. For the time responses we have used the learned policy parameters to simulate a sample run of $10s$ for the Ballbot system. Their results for stabilization problem of the body of the Ballbot in the vertical position are shown in Fig. 5.36. The resulted torque profile for the motors which cause the ball to roll, is given in Fig. 5.35. It should be noted that for a better representation of the torque behavior we considered the first $1s$ of the simulation time.

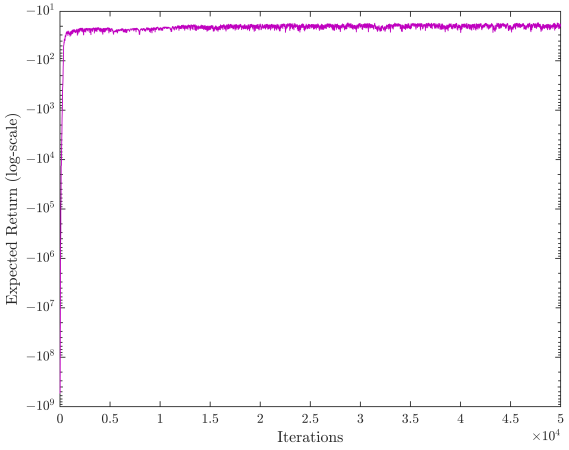


FIGURE 5.34: Expected return of the adaptive MCMC for Ballbot system.

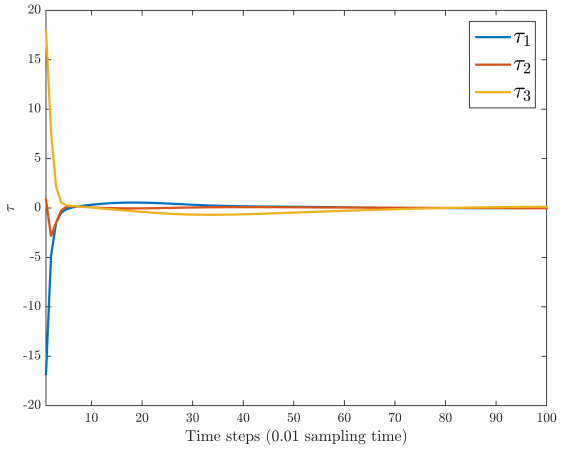
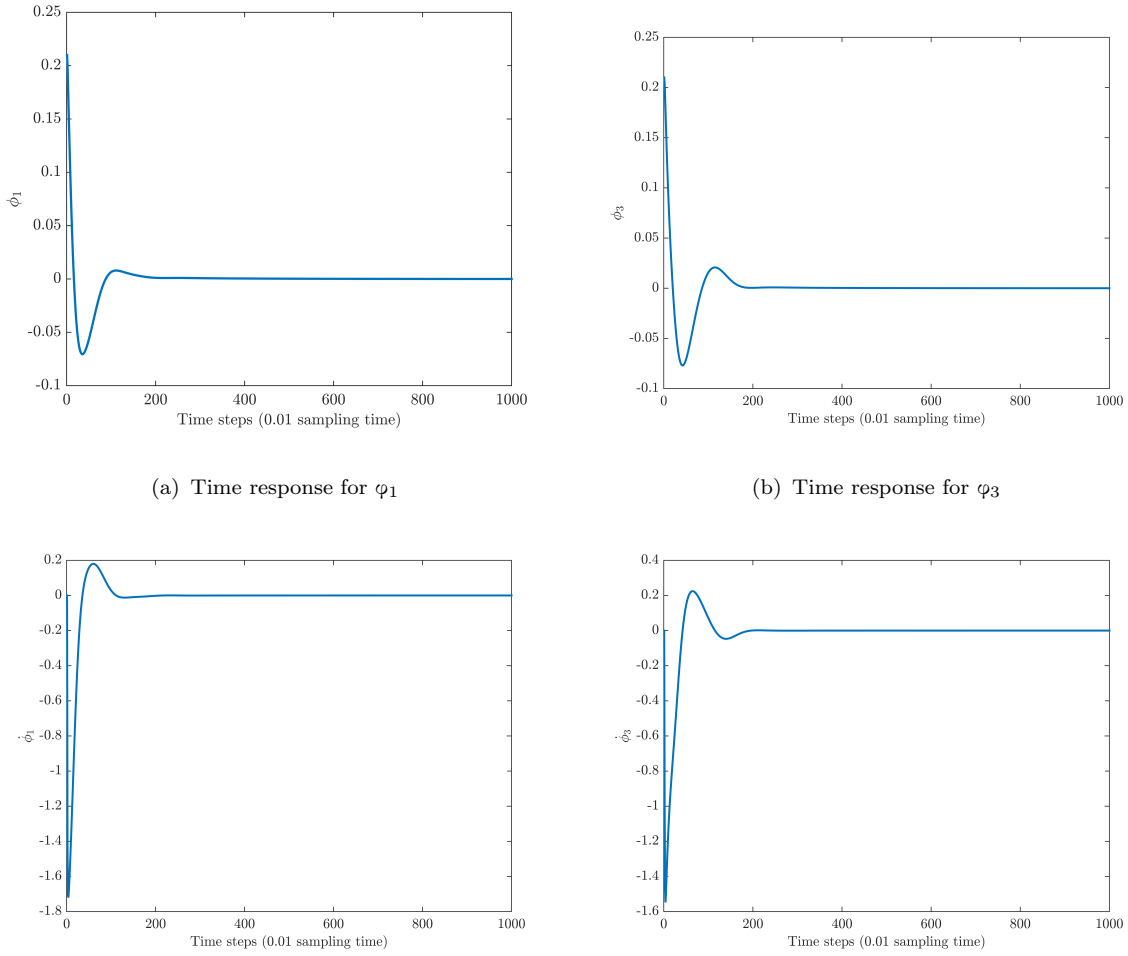


FIGURE 5.35: Time responses for the torque profiles of the adaptive MCMC for Ballbot system.



(a) Time response for φ_1

(b) Time response for φ_3

(c) Time response for $\dot{\varphi}_1$

(d) Time response for $\dot{\varphi}_3$

FIGURE 5.36: Time responses of the adaptive MCMC for Ballbot system.

Chapter 6

Experimental Results

Summary: The present chapter is intended to make the available gap between the theory and application and contribute to the verification of the proposed MCMC algorithm for policy search by carrying out real-time experiments on a physical system. An empirical assessment of the proposed algorithm will be performed on a 2-DoF planar manipulator setup the model of which was used for simulations in Section 5.3 . The control problem involves following a circular reference in the $x-y$ direction. For this purpose, the proposed MCMC algorithm will be manipulated and simplified in a way that it is suitable for real-time applications by excluding the resampling part in the framework of the algorithm. Taking into account the real-time experiments, it must be remembered that the Bayesian RL and specifically MCMC is still much in its infancy. In fact, the majority of the beneficial real-time implementations of RL do not belong to the Bayesian learning category and thus we will try to contribute to this part.

6.1 Physical setup of the planar manipulator

Since modeling and control of the kinematics and dynamics of the 2-DoF planar manipulator is out of the scope of this thesis, we refer the readers to some works which have extensively went through this topic such as Yu (2006), Campion et al. (2005) and Unal et al. (2008). The physical model is shown in Fig. 6.1 where the parallel arms are used to apply the required force for the translation of the end-effector, while they are built in a way that have low friction. The movement occurs by a capstan drive which is located at the base joints and enables the user to apply large torques regardless of the erosion. Two motors engaged in the body of the planar manipulator run it so that the end-effector can be controlled. Linear current drivers were designed and built to transfer the power to the motors. The position of the motors are measured precisely thanks to the high-resolution optical encoders. The control process of the overall system is done under a PC implementation of the VDAQ Hardware-In-the-Loop (HIL) control board which has the ability to utilize the MATLAB simulink's real-time kernels. A representation of the 2-Dof planar manipulator as a linkage systems is illustrated in Fig. 5.19.

6.2 Policy search with modified MCMC algorithm

As previously discussed, Algorithms 6 and 7 are using particle filters in their iterations and need to simultaneously propagate different samples which is suitable for simulation environment. To make them fit for real applications, which being in different states at an individual time step t is not possible, the sampling and resampling parts of these algorithms are excluded. This is equivalent to the situation where just one particle filter is hired in their structure. The resulting modified MCMC algorithms are then given in Algorithm 10 and Algorithm 11.

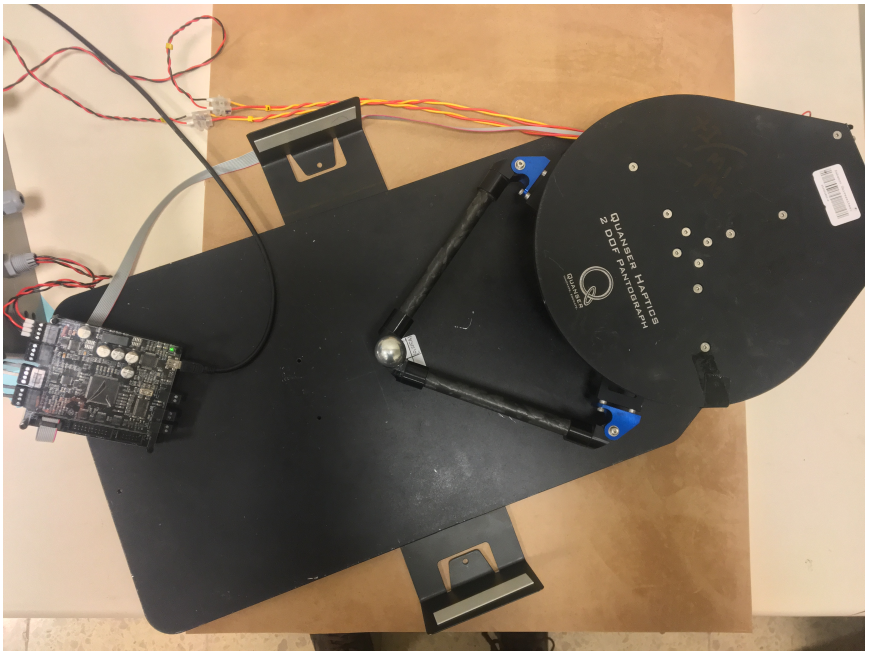


FIGURE 6.1: Physical setup for the two link manipulator.

Algorithm 10: PMMH for RL

Input: Iteration numbers m, initial guess of parameter and estimate of expected cost $(\theta^{(0)}, \hat{J}^{(0)})$, proposal distribution $q(\theta'|\theta)$

Output: Parameter samples $\theta^{(m)}$, $m = 1, 2, \ldots$

for *m = 1, 2, ...* **do**

Given $\theta^{(m-1)} = \theta$ and $\hat{J}^{(m-1)} = \hat{J}$, draw a candidate value from the proposal distribution $\theta' \sim q(\theta'|\theta)$.

Implement Algorithm 11 to get an unbiased estimate $\hat{J}'$ of $J(\theta')$

Accept the proposed parameter and set $\theta^{(m)} = \theta'$ and $\hat{J}^{(m)} = \hat{J}'$ with probability $\min\{1, \hat{\rho}(\theta, \theta')\}$ where

$$\hat{\rho}(\theta, \theta') = \frac{q(\theta|\theta')}{q(\theta'|\theta)} \frac{\mu(\theta')}{\mu(\theta)} \frac{\hat{J}'}{\hat{J}},$$

otherwise reject the proposal and set $\theta^{(m)} = \theta$ and $\hat{J}^{(m)} = \hat{J}$.

end

Algorithm 11: Modified SMC algorithm for an unbiased estimate of $J(\theta)$

Input: Policy θ, time steps n, discount factor γ for RL

Output: Unbiased estimate of J as $\hat{J}$

Start with $\hat{J}_0 = 1$.

for $t = 1, \ldots, n$ **do**

Draw sample trajectories from the transition density $x_t \sim p_\theta(x_t|\theta)$

Calculate the non-negative reward as $W_t = \exp\{r(x_t)\,\gamma^{t-1}\}$.

Update the estimate: $\hat{J}_t = \hat{J}_{t-1} \times W_t$

return $\hat{J}$.

end

6.3 Trajectory control and experimental results

The classical industrial robotic controllers which are normally designed for usual predefined tasks need to be substituted with intelligent ones, which are capable of making decisions based on data driven techniques, to best adopt to their most recent environment. They should have the potential to learn and perform different specific tasks with varying environments. Incorporating learning mechanisms in the standard stabilizing feedback control strategies can be an alternative for this. Trajectory control of a robotic manipulator, which enforces the system to track a given path precisely, can be achieved by utilizing policy search algorithms for feedback stabilization controllers. The reference tracking task here is to learn an optimal policy which enables the robot to follow a circular trajectory in the $x - y$ axis. Therefore, the goal is to minimize the corresponding errors in both x and y axes. The state space of 2-DoF planar manipulator is formalized as a function of

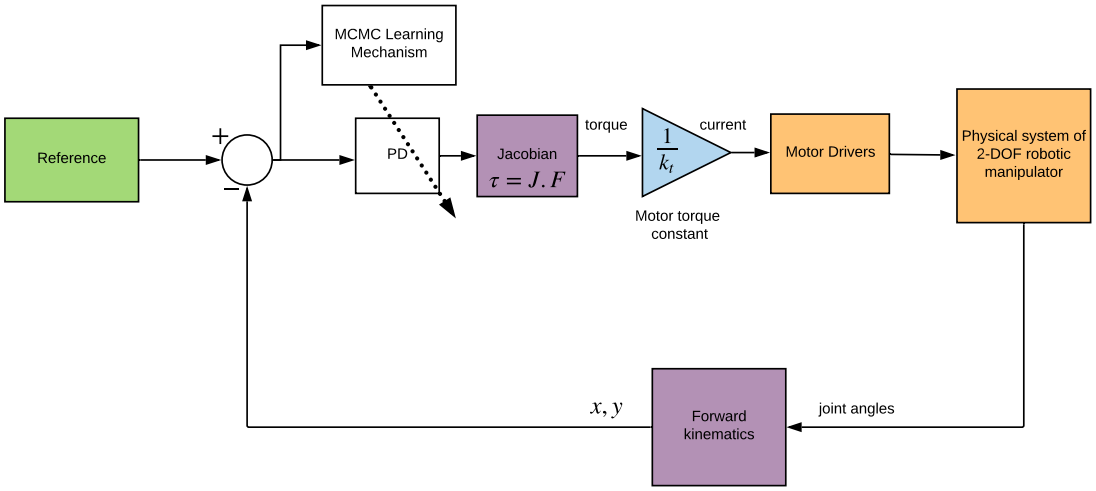


FIGURE 6.2: Learning paradigm for the trajectory control of the 2-DoF manipulator.

the end-effector position error and its derivative (velocity) as:

$$s_t = \begin{bmatrix} x_{ref} - x \\ y_{ref} - y \\ \dot{x}_{ref} - \dot{x} \\ \dot{y}_{ref} - \dot{y} \end{bmatrix} = \begin{bmatrix} e_x \\ e_y \\ \dot{e}_x \\ \dot{e}_y \end{bmatrix}$$

In order to use PD as the controller for our desired trajectory tracking, a computed-torque method has been employed to linearize the feedback. Jacobian and forward kinematics for the 2-DoF manipulator provides an opportunity to take advantage of the computed-torque technique. As it can be seen from the trajectory control schematic of the system in Fig. 6.2, the motor driver receives current through multiplying the resulted torque from the Jacobian by the inverse of motor torque coefficient as its input to operate. The inputs and outputs to the MCMC learning mechanism are motor torques (τ_1, τ_2) and position of the end-effector (x_Q, y_Q), respectively.

Gains of the PD controller comprises the parameter vector to be learned by the modified MCMC algorithm as:

$$\theta = \begin{bmatrix} k_{p_x} & k_{p_y} & k_{d_x} & k_{d_y} \end{bmatrix}^T$$

Reward function for the learning structure is taken as equation (6.1) in which $Q \in \mathbb{R}^{4\times4}$ and $C \in \mathbb{R}^{2\times2}$ are chosen to be $Q = \text{diag}([50, 50, 0.5, 0.5])$, $C = \text{diag}([0.01, 0.01])$.

$$r(a,s) = \begin{cases} -s^T Q s - Ca^2, & -110\text{mm} \leq x \leq 110\text{mm}, \text{ and } -170 \text{ mm} \leq y \leq 330 \text{ mm} \\ -10^8, & \text{otherwise} \end{cases} \tag{6.1}$$

The value of the discounted factor for the multiplicative total reward is $\gamma = 0.99$ and each iteration of the MCMC lasts for 5s with a 0.001s sampling time except in the situation where the agent leaves its allowable region and a failure happens (experiment time will be less than 5s). In each iteration of the MCMC, the end-effector required to be calibrated meaning that it has to be located in the initial point $cw = (0, 225.14)$ where the joint angles $q_2 = \pi/6$ and $q_1 = 5\pi/6$. The whole procedure of the experiment for policy search has been planned to be 900 iterations. In the experiments in 200^{th} iteration the optimal policy parameters were completely obtained. For the policy update by proposal distribution, a Gaussian Random Walk (GRW) is used where its covariance matrix is selected as $\Sigma_q = \text{diag}([0.001, 0.001, 0.0001, 0.0001])$. A zero mean Gaussian distribution with a covariance of $\Sigma_{\mu(\theta)} = \text{diag}([1.2, 1.2, 0.006, 0.006])$ is assigned for the prior density over parameter vector θ .

To capture the idea of the learning procedure for the robotic manipulator system, a video demonstration can be reached in our *Youtube* channel in *OnatSU*.

Trace plot and improvement of the return function during learning are both depicted in Fig. 6.3. From the figure, it is observed that the trend of the learned parameter trace for k_{d_x} and k_{d_y} are almost similar. The learned parameter vector for the PD controller is:

$$\theta = \left[k_{p_x} = 0.1285, k_{p_y} = 0.534, k_{d_x} = 0.0013, k_{d_y} = 0.000125\right]^T$$

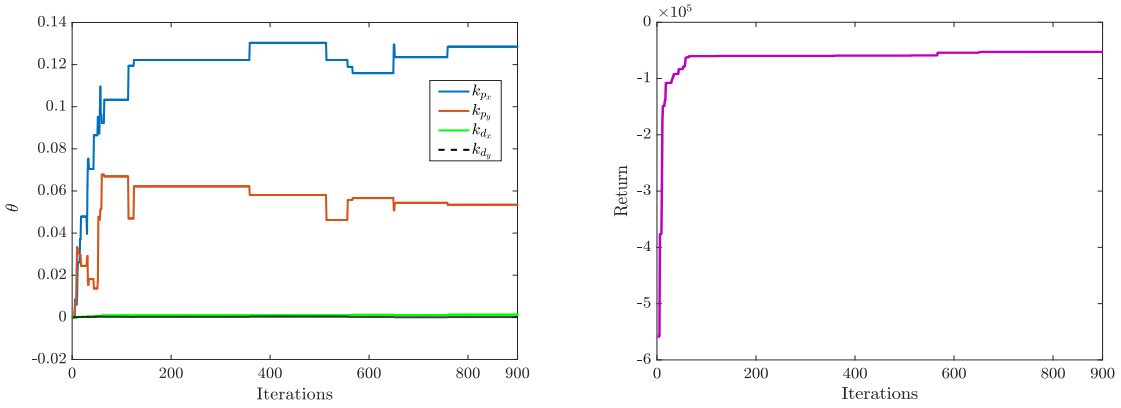


FIGURE 6.3: Trace plots and average return for physical setup of robotic manipulator: **Left**: Trace plots. **Right**: Total return

The position error of the reference trajectory for both x and y axes are shown in Fig. 6.4.

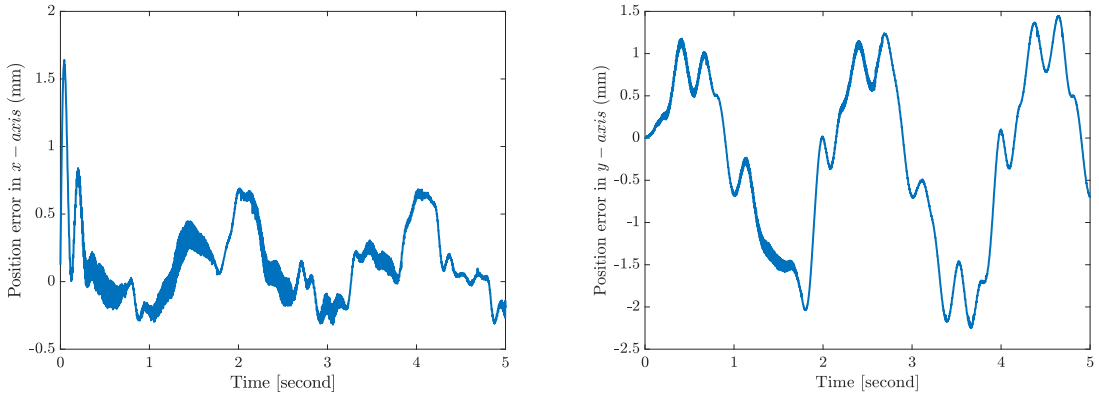


FIGURE 6.4: Position error for physical setup of robotic manipulator: **Left**: error in $x-$axis. **Right**: error in $y-$axis.

The resulting torques generated by the motors are represented in Fig. 6.5. As it is observed from Fig. 6.4 and Fig. 6.5, the learning controller experiences an unpleasant oscillatory behavior or jitters with high frequencies which is caused by the measurement noise or an uncertainty in the planar's system such as the inherited internal velocity controller. To eliminate this problem one can use a low-pass filter in the joint-space measurements but it will add undesired delays to the time response of the overall system. Because of this fact, we did not use it in our experiments.

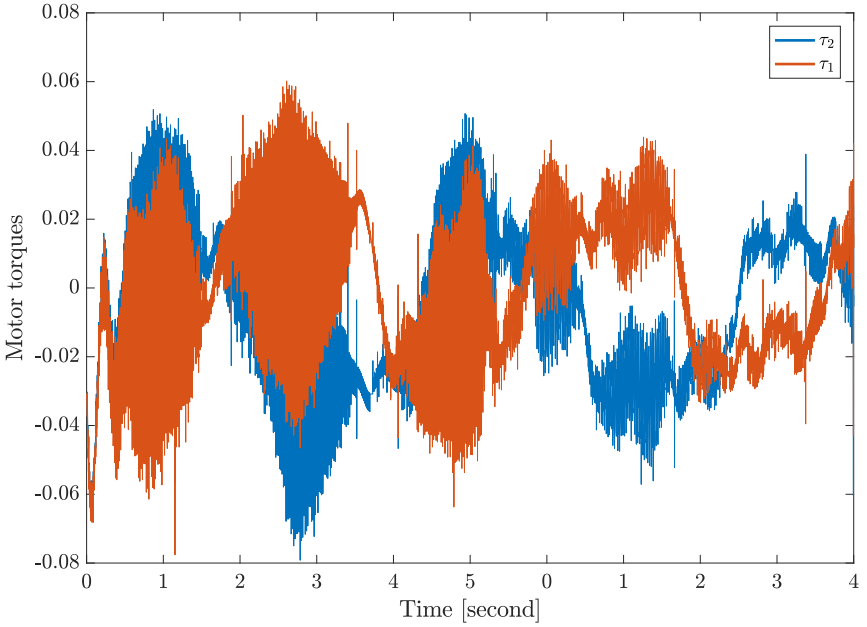


FIGURE 6.5: Torque profiles generated by motors.

Finally, in Fig. 6.6 trajectory of the desired circular reference and actual one are depicted. To have a precision measurement degree of the proposed learning algorithm, Root Mean Square Error (RMSE) has been used for our trajectory tracking task which is equal to 4.88 mm. (for 5s planar draws the circular trajectory two times).

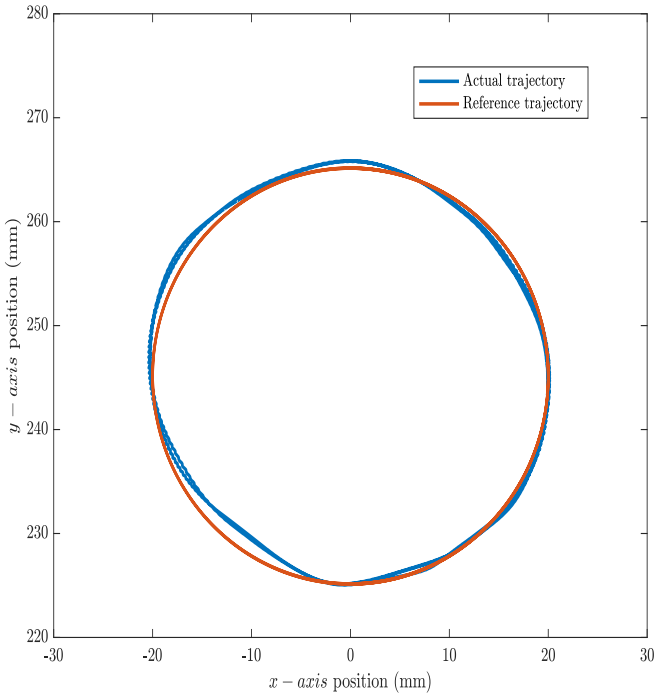


FIGURE 6.6: Desired reference and actual trajectory.

Chapter 7

Conclusions

Summary: In the presented thesis, we developed an offline SMC-based policy search algorithm according to the RL framework for the continuous high dimensional state spaces in the robotics domain. In the following we summarize the overall concept and contributions of the thesis and discuss the possible future work.

7.1 Thesis Focus

In the first part of the available thesis we explored the possibility of using PG RL algorithms on FLCs, owing to the fact that their contribution in this domain have not been remarkably considered. Although there exist some related attempts on either gain tuning of the FLCs or some which improve the quality of Q-learning algorithm in RL by fuzzy logic, these works mostly concentrate on discrete state spaces. Compared to them, we employ PG RL techniques to tune the parameters of the FLCs in the continuous state and action spaces where the need for the approximation of the value functions is alleviated. This part is also contributed to the hand-tuning challenges of FLC gains via applying a learning mechanism.

As a second contribution, a new gradient-free algorithm based on MCMC for the policy search problem in continuous state spaces, is proposed. These algorithms are best applicable for sampling from complicated distributions, when an analytic computation is hard to reach. The uniqueness of our proposed algorithm is based on the formulation of the policy search problem in a Bayesian framework where the expected total reward, is established by the product of exponential rewards and is treated as a likelihood function. We propose to use the risk-sensitivity idea in the structure of the reward function to facilitate the application of SMC algorithm in the iterations of the MCMC. Incorporating an uninformative prior density with a risk-sensitive cost function results in a posterior distribution for the policy distribution. Then one can use the proposed MCMC algorithm to identify promising areas for the optimal policy parameters. Considering the weaknesses of the gradient based approaches in RL, we can summarize the benefits of the proposed MCMC algorithm as following:

- The proposed MCMC is a gradient-free algorithm based on the estimates of the expected risk sensitive total reward. This specification makes Its structure easy to implement and therefore, lessens the computation complexity.
- Since the samples drawn by the MCMC algorithm from the posterior density are used to seek the high probable areas of the optimal policy parameters, it is less probable to get trapped in local solutions.
- We have compared our proposed MCMC algorithm with PG RL ones and showed by our simulations that it is superior to the gradient based algorithms both in terms of control time response and convergence.
- The proposed MCMC algorithm is contributed to both the statistical learning and robotics where it is capable of parameter learning of the controllers and the model of the system at the same time by using sampling methods in Bayesian

statistics. It is also able to deal with uncertainties by incorporating some uninformative prior densities on the policy parameters.

As a third contribution, we have developed a policy search algorithm which can be reformulated and be suitable for dealing with high-dimensional state spaces. In this direction, we modified our MCMC algorithm to an adaptive form where it has the ability to change the covariance matrix of the proposal Gaussian distribution for the parameter update rule. This is essential especially when the hand-tuning of these covariance matrices are costly for high-dimensional parameter spaces.

As the last contribution by considering the fact that the Bayesian MCMC learning methods are in their first stages of progress in real-time applications, we made a bridge between theory and application by carrying out experimental evaluations of the proposed MCMC algorithm on a 2-DoF robotic manipulator. Our problem was a trajectory control task in which the robotic manipulator was responsible for tracking a given circular reference precisely.

7.2 Future Works

One of the limitations of the proposed MCMC algorithm is that statistically for a better convergence of the algorithm more iterations of learning need to be done. This is challenging specially in real-time complex robotic applications. As a result, it would be more beneficial, if we could decrease the dependency of our algorithm to iterations by incorporating some a-priori knowledge to the learning algorithm for a fast convergence.

Since our method is an offline policy search method, one of the extensions of it would be an online parameter update scheme during the time steps of each learning iteration. Another modification can happen for the structure of the reward function

where for our case we used a multiplicative reward to take the advantage of SMC approach. Since in real-time implementations we can not use particle filters, Instead likewise to the general setting of the RL paradigm, we can employ an additive reward function.

For the A-MCMC algorithm, we could not manage to perform the algorithm on a physical system and our experiments were limited to simulation environments on a nonlinear model of a Ballbot system. Therefore, to demonstrate that our claims about the proposed A-MCMC algorithm is valid in real-time as well, we could carry its corresponding real-time evaluations.

In this thesis we used our algorithm for the parameter estimation of the controllers. To extend its depth of application, we could employ it on the problem of model learning of an unknown system. Although there have been some efforts in model learning problem by using gradient-based methods in the literature, it would be an effective gradient-free method in the model learning domain.

Bibliography

[1] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[2] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2829–2838. JMLR.org, 2016. URL `http://dl.acm.org/citation.cfm?id=3045390.3045688`.

[3] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.*, 32(11):1238–1274, September 2013. ISSN 0278-3649. doi: 10.1177/0278364913495721. URL `http://dx.doi.org/10.1177/0278364913495721`.

[4] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Found. Trends Robot*, 2(1–2):1–142, August 2013. ISSN 1935-8253. doi: 10.1561/2300000021. URL `http://dx.doi.org/10.1561/2300000021`.

[5] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373, January 2016. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2946645.2946684`.

[6] Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1061–1068, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1963-8. doi: 10.1145/2463372.2463509. URL http://doi.acm.org/10.1145/2463372.2463509.

[7] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017.

[8] J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[9] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 745–750, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273590. URL http://doi.acm.org/10.1145/1273496.1273590.

[10] Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible natural gradient policy search. *Machine Learning*, May 2019. ISSN 1573-0565. doi: 10.1007/s10994-019-05807-0. URL https://doi.org/10.1007/s10994-019-05807-0.

[11] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 849–856. Curran Associates, Inc., 2008. URL http://dblp.uni-trier.de/db/conf/nips/nips2008.html#KoberP08.

[12] Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 465–472, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104541.

[13] Kamil Ciosek and Shimon Whiteson. Expected policy gradients. In *AAAI*, pages 2868–2875. AAAI Press, 2018.

[14] Voot Tangkaratt, Syogo Mori, Tingting Zhao, Jun Morimoto, and Masashi Sugiyama. Model-based policy gradients with parameter-based exploration by least-squares conditional density estimation. *Neural Netw.*, 57:128–140, September 2014. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.06.006. URL http://dx.doi.org/10.1016/j.neunet.2014.06.006.

[15] F. Sehnke, A. Graves, C. Osendorfer, and J. Schmidhuber. Multimodal parameter-exploring policy gradients. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 113–118, Dec 2010. doi: 10.1109/ICMLA.2010.24.

[16] Athanasios S. Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent and Robotic Systems*, 86:153–173, 2017.

[17] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomput.*, 71(7-9): 1180–1190, March 2008. ISSN 0925-2312. doi: 10.1016/j.neucom.2007.11.026. URL http://dx.doi.org/10.1016/j.neucom.2007.11.026.

[18] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*

- *Volume 37*, ICML'15, pages 1889–1897. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045319.

[19] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, Nov 2012. ISSN 1094-6977. doi: 10.1109/TSMCC.2012.2218595.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

[21] Voot Tangkaratt, Abbas Abdolmaleki, and Masashi Sugiyama. Guide actor-critic for continuous control. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJk59JZ0b.

[22] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[23] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519, 2015.

[24] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Xingchao Peng, Sergey Levine, Kate Saenko, and Trevor Darrell. Towards adapting deep

visuomotor representations from simulated to real environments. *CoRR*, abs/1511.07111, 2015.

[25] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5285–5294, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL `http://dl.acm.org/citation.cfm?id=3295222.3295280`.

[26] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010. URL `http://dblp.uni-trier.de/db/journals/corr/corr1012.html#abs-1012-2599`.

[27] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8 (5-6):359–483, November 2015. ISSN 1935-8237. doi: 10.1561/2200000049. URL `http://dx.doi.org/10.1561/2200000049`.

[28] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 761–768, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL `http://dl.acm.org/citation.cfm?id=295240.295801`.

[29] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 201–208, New York, NY, USA, 2005.

ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102377. URL http://doi.acm.org/10.1145/1102351.1102377.

[30] Mohammad Ghavamzadeh and Yaakov Engel. Bayesian policy gradient algorithms. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 457–464, Cambridge, MA, USA, 2006. MIT Press. URL http://dl.acm.org/citation.cfm?id=2976456.2976514.

[31] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using trajectory data to improve bayesian optimization for reinforcement learning. *J. Mach. Learn. Res.*, 15(1):253–282, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2627443.

[32] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045502.

[33] Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. Synthesizing neural network controllers with probabilistic model-based reinforcement learning. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2538–2544, 2018.

[34] Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. Real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1563, Piscataway, NJ, USA, May 2017. IEEE.

[35] Rémi Pautrat, Konstantinos I. Chatzilygeroudis, and Jean-Baptiste Mouret. Bayesian optimization with automatic prior selection for data-efficient direct policy search. In *ICRA*, pages 7571–7578. IEEE, 2018.

[36] Matthew Hoffman, Arnaud Doucet, Nando D. Freitas, and Ajay Jasra. Bayesian policy learning with trans-dimensional MCMC. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 665–672. Curran Associates, Inc., 2008. URL `http://papers.nips.cc/paper/3343-bayesian-policy-learning-with-trans-dimensional-mcmc.pdf`.

[37] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1):5–43, Jan 2003. ISSN 1573-0565. doi: 10.1023/A:1020281327116. URL `https://doi.org/10.1023/A:1020281327116`.

[38] A. T. Cemgil. *A Tutorial Introduction to Monte Carlo methods, Markov Chain Monte Carlo and Particle Filtering.* Academic Press Library in Signal Processing, 2013.

[39] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Bayesian role discovery for multi-agent reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 1587–1588, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9826571-1-9. URL `http://dl.acm.org/citation.cfm?id=1838206.1838494`.

[40] Yanan Fan and Scott A Sisson. *Reversible jump MCMC.* Chapman and Hall/CRC.

[41] Matthias Hoffman, Hendrik Kück, Nando de Freitas, and Arnaud Doucet. New inference strategies for solving markov decision processes using reversible jump MCMC. *CoRR*, abs/1205.2643, 2012.

[42] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society Series B*, 72(3):269–342, 2010. URL `https://EconPapers.repec.org/RePEc:bla:jorssb:v:72:y:2010:i:3:p:269-342`.

[43] Troels Arnfred Bojesen. Policy-guided monte carlo: Reinforcement-learning markov chain dynamics. *Phys. Rev. E*, 98:063303, Dec 2018. doi: 10.1103/PhysRevE.98.063303. URL `https://link.aps.org/doi/10.1103/PhysRevE.98.063303`.

[44] Christophe Andrieu and Christian P Robert. Controlled mcmc for optimal sampling. Working Papers 2001-33, Center for Research in Economics and Statistics, 2001. URL `https://EconPapers.repec.org/RePEc:crs:wpaper:2001-33`.

[45] H. Haario, E. Saksman, and J. Tamminen. Componentwise adaptation for high dimensional MCMC. *Computational Statistics*, 20(2):265–273, 2005.

[46] Dao Nguyen, Perry de Valpine, Yves Atchade, Daniel Turek, Nicholas Michaud, and Christopher Paciorek. Automatic adaptation of MCMC algorithms. *arXiv e-prints*, art. arXiv:1802.08798, Feb 2018.

[47] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press. URL `http://dl.acm.org/citation.cfm?id=3009657.3009806`.

[48] V. Gullapali, J. Franklin, and H. Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems*, (39), 1994.

[49] P. Dayan and G.E. Hinton. Using Expectation-Maximization for reinforcement learning. *Neural Computation*, 9:271–278, 1997.

[50] H. Kimura and S. Kobayashi. Reinforcement learning for continuous action using stochastic gradient ascent. *The 5th International Conference on Intelligent Autonomous Systems*, 1998.

[51] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 945–952, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143963. URL `http://doi.acm.org/10.1145/1143844.1143963`.

[52] J. Peters. Machine learning of motor skills for robotics. *Phd Thesis Proposal USC Technical Report, Tech Rep*, 2005.

[53] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita. Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, pages 1594 –1601, 2005.

[54] Hilbert J. Kappen, V. Gomez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87:159–182, 2012.

[55] Chris J. Maddison, Dieterich Lawson, George Tucker, Nicolas Heess, Arnaud Doucet, Andriy Mnih, and Yee Whye Teh. Particle value functions. In *Workshop track - ICLR 2017*, Mar. 2017. doi: 10.1109/ASPAA.2007.4392996.

[56] David Wingate, Noah D. Goodman, Daniel M. Roy, Leslie P. Kaelbling, and Joshua B. Tenenbaum. Bayesian policy search with policy priors. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 16–22, July 2011.

[57] Vahid Tavakol Aghaei and Ahmet Onat. Tuning scaling factors of fuzzy logic controllers via reinforcement learning policy gradient algorithms. In *Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering*, ICMRE 2017, pages 146–151, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5280-2. doi: 10.1145/3068796.3068827. URL `http://doi.acm.org/10.1145/3068796.3068827`.

[58] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696.

[59] J. Baxter and P. L. Bartlett. Direct gradient-based reinforcement learning. In *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, volume 3, pages 271–274 vol.3, May 2000. doi: 10.1109/ISCAS.2000.856049.

[60] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press. URL `http://dl.acm.org/citation.cfm?id=3009657.3009806`.

[61] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomput.*, 71(7-9): 1180–1190, March 2008. ISSN 0925-2312. doi: 10.1016/j.neucom.2007.11.026. URL `http://dx.doi.org/10.1016/j.neucom.2007.11.026`.

[62] Jan Reinhard Peters. *Machine Learning of Motor Skills for Robotics*. PhD thesis, Los Angeles, CA, USA, 2007. AAI3262746.

[63] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN 9780486428093.

[64] Jeffrey S. Rosenthal. *A first look at rigorous probability theory*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, second edition, 2006. ISBN 978-981-270-371-2; 981-270-371-3.

[65] Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. Off-policy temporal difference learning with function approximation. In *ICML*, 2001.

[66] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 759–766, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2. URL `http://dl.acm.org/citation.cfm?id=645529.658134`.

[67] Nicolas Meuleau, Leonid Peshkin, and Kee-Eung Kim. Exploration in gradient-based reinforcement learning. Technical Report 2001-003, MIT, 2001.

[68] Ertunç Erdil, Sinan Yıldırım, Müjdat Çetin, and Tolga Taş dizen. Mcmc shape sampling for image segmentation with nonparametric shape priors. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 411–419, 2016.

[69] Sinan Yıldırım, Sumeetpal S. Singh, Thomas Dean, and Ajay Jasra. Parameter estimation in hidden markov models with intractable likelihoods using sequential monte carlo. *Journal of Computational and Graphical Statistics*, 24(3):846–865, 2015. doi: 10.1080/10618600.2014.938811. URL `https://doi.org/10.1080/10618600.2014.938811`.

[70] W. Larry Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[71] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21: 1087, 1953.

[72] A. Doucet, J.P.G. De Freitas, and N.L. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[73] Ronald A. Howard and James E. Matheson. Risk-sensitive markov decision processes. *Management Science*, 18(7):356–369, 1972. URL `https://EconPapers.repec.org/RePEc:inm:ormnsc:v:18:y:1972:i:7:p:356-369`.

[74] Takayuki Osogami. Robustness and risk-sensitivity in markov decision processes. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 233–241. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4693-robustness-and-risk-sensitivity-in-markov-decision-processes.pdf`.

[75] S. I. Marcus, E. Fernández-Gaucherand, D. Hernández-Hernández, S. Coraluppi, and P. Fard. *Risk Sensitive Markov Decision Processes*, pages 263–279. Birkhäuser Boston, Boston, MA, 1997. ISBN 978-1-4612-4120-1. doi: 10.1007/978-1-4612-4120-1_14. URL `http://dx.doi.org/10.1007/978-1-4612-4120-1_14`.

[76] Aviv Tamar. *Risk-Sensitive and Efficient Reinforcement Learning Algorithms*. PhD thesis, Israel Institute of Technology, Haifa, 2015.

[77] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *CoRR*, abs/1109.2147, 2011. URL `http://arxiv.org/abs/1109.2147`.

[78] Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Machine Learning*, 49(2):267–290, Nov 2002. ISSN 1573-0565. doi: 10.1023/A:1017940631555. URL https://doi.org/10.1023/A:1017940631555.

[79] Yun Shen, Michael J. Tobia, Tobias Sommer, and Klaus Obermayer. Risk-sensitive reinforcement learning. *CoRR*, abs/1311.2097, 2013. URL http://arxiv.org/abs/1311.2097.

[80] Chris J. Maddison, Dieterich Lawson, George Tucker, Nicolas Heess, Arnaud Doucet, Andriy Mnih, and Yee Whye Teh. Particle value functions. *CoRR*, abs/1703.05820, 2017. URL http://arxiv.org/abs/1703.05820.

[81] Matt Hoffman, Arnaud Doucet, Nando de Freitas, and Ajay Jasra. Trans-dimensional mcmc for bayesian policy learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 665–672, USA, 2007. Curran Associates Inc. ISBN 978-1-60560-352-0. URL http://dl.acm.org/citation.cfm?id=2981562.2981646.

[82] David Wingate, Noah D. Goodman, Daniel M. Roy, Leslie P. Kaelbling, and Joshua B. Tenenbaum. Bayesian policy search with policy priors. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011. URL http://www.mit.edu/~wingated/papers/ijcaipp.pdf.

[83] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 945–952, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143963. URL http://doi.acm.org/10.1145/1143844.1143963.

[84] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1):5–43, Jan 2003. ISSN 1573-0565. doi: 10.1023/A:1020281327116. URL `https://doi.org/10.1023/A:1020281327116`.

[85] Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient monte carlo computations. *Annals of Statistics*, pages 697–725, 2009.

[86] P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer-Verlag, New York, 2004.

[87] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. An approximate inference approach to temporal optimization in optimal control. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2011–2019. Curran Associates, Inc.

[88] Paul Fearnhead, David Wyncoll, and Jonathan Tawn. A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464, 06 2010. ISSN 0006-3444. doi: 10.1093/biomet/asq013. URL `https://doi.org/10.1093/biomet/asq013`.

[89] Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *In Proceedings of the International Conference on Machine Learning*, 2011.

[90] Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan M. Maciejowski, and Nicolas Chopin. On particle methods for parameter estimation in state-space models. 2015.

[91] Heikki Haario, Eero Saksman, and J Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7:223–242, 2001.

[92] G. Roberts and J. Rosenthal. Examples of adaptive mcmc. 2008.

[93] Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling for various metropolis-hastings algorithms, 2001.

[94] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms, 1994.

[95] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[96] Yves Atchade, Yves Atchade, and Gersende Fort. Limit theorems for some adaptive mcmc algorithms with subgeometric kernels. *Bernoulli*, 16(1):116–154, 2010.

[97] Yan Bai, Gareth O. Roberts, and Jeffrey S. Rosenthal. On the containment condition for adaptive markov chain monte carlo algorithms. Technical report, University of Toronto, 2008.

[98] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001. URL `https://projecteuclid.org:443/euclid.bj/1080222083`.

[99] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, December 2008. ISSN 0960-3174. doi: 10.1007/s11222-008-9110-y. URL `http://dx.doi.org/10.1007/s11222-008-9110-y`.

[100] Robert Nishihara, Iain Murray, and Ryan P. Adams. Parallel mcmc with generalized elliptical slice sampling. *Journal of Machine Learning Research*, 15 (1):2087–2112, 2014. URL `http://dblp.uni-trier.de/db/journals/jmlr/jmlr15.html#NishiharaMA14`.

[101] E. H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Proceedings of the Institution of Electrical Engineers*, 121(12):1585–1588, December 1974. ISSN 0020-3270. doi: 10.1049/piee.1974.0328.

[102] J. Wang and T. Kumbasar. Optimal pid control of spatial inverted pendulum with big bang?big crunch optimization. *IEEE/CAA Journal of Automatica Sinica*, pages 1–11, 2018. ISSN 2329-9266. doi: 10.1109/JAS.2018.7511267.

[103] C. Ko, T. Lee, H. Fan, and C. Wu. Genetic auto-tuning and rule reduction of fuzzy pid controllers. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1096–1101, Oct 2006. doi: 10.1109/ICSMC.2006.384546.

[104] Hai-bin Duan, Dao-bo Wang, and Xiu-fen Yu. Novel approach to nonlinear pid parameter optimization using ant colony optimization algorithm. *Journal of Bionic Engineering*, 3(2):73–78, Jun 2006. ISSN 2543-2141. doi: 10.1016/S1672-6529(06)60010-3. URL `https://doi.org/10.1016/S1672-6529(06)60010-3`.

[105] Hamid Boubertakh, Mohamed Tadjine, Pierre-Yves Glorennec, and Salim Labiod. Tuning fuzzy pd and pi controllers using reinforcement learning. *ISA transactions*, 49 4:543–51, 2010.

[106] V. T. Aghaei, A. Onat, I. Eksin, and M. Guzelkaya. Fuzzy pid controller design using q-learning algorithm with a manipulated reward function. In *2015 European Control Conference (ECC)*, pages 2502–2507, July 2015. doi: 10.1109/ECC.2015.7330914.

[107] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2010. ISBN 1439821089, 9781439821084.

[108] Wu Zhi Qiao and Masaharu Mizumoto. Pid type fuzzy controller and parameters adaptive method. *Fuzzy Sets Syst.*, 78(1):23–35, February 1996. ISSN 0165-0114. doi: 10.1016/0165-0114(95)00115-8. URL http://dx.doi.org/10.1016/0165-0114(95)00115-8.

[109] Christoph Dann, Gerhard Neumann, and Jan Peters. Policy evaluation with temporal differences: A survey and comparison. *J. Mach. Learn. Res.*, 15 (1):809–883, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2638563.

[110] Xue-Song WANG, Yu hu CHENG, and Wei SUN. A proposal of adaptive pid controller based on reinforcement learning. *Journal of China University of Mining and Technology*, 17(1):40 – 44, 2007. ISSN 1006-1266. doi: https://doi.org/10.1016/S1006-1266(07)60009-1.

[111] Ziqian Liu. Self-tuning control of electrical machines using gradient descent optimization. *Optimal Control Applications and Methods*, 28(2):77–93, 3 2007. ISSN 1099-1514. doi: 10.1002/oca.789.

[112] Hongnian Yu. Modeling and control of hybrid machine systems — a five-bar mechanism case. *International Journal of Automation and Computing*, 3(3): 235–243, Jul 2006. ISSN 1751-8520. doi: 10.1007/s11633-006-0235-1.

[113] A. Bonci. New dynamic model for a ballbot system. In *2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6, Aug 2016. doi: 10.1109/MESA.2016.7587176.

[114] Péter Fankhauser and Corsin Gwerder. Modeling and control of a ballbot. Master's thesis, ETH Zurich, Zürich, 2010. Bachelor Thesis ETH Zurich, 2010.

[115] G. Campion, Qi Wang, and V. Hayward. The pantograph mk-ii: a haptic instrument. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 193–198, Aug 2005. doi: 10.1109/IROS.2005.1545066.

[116] R. Unal, G. Kiziltas, and V. Patoglu. A multi-criteria design optimization framework for haptic interfaces. In *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 231–238, March 2008. doi: 10.1109/HAPTICS.2008.4479949.

[117] OnatSU. A video demonstration for policy learning by mcmc implemented on a 2-dof robotic manipulator. May 2019. URL `https://www.youtube.com/watch?v=0-pmAUC_x14`.