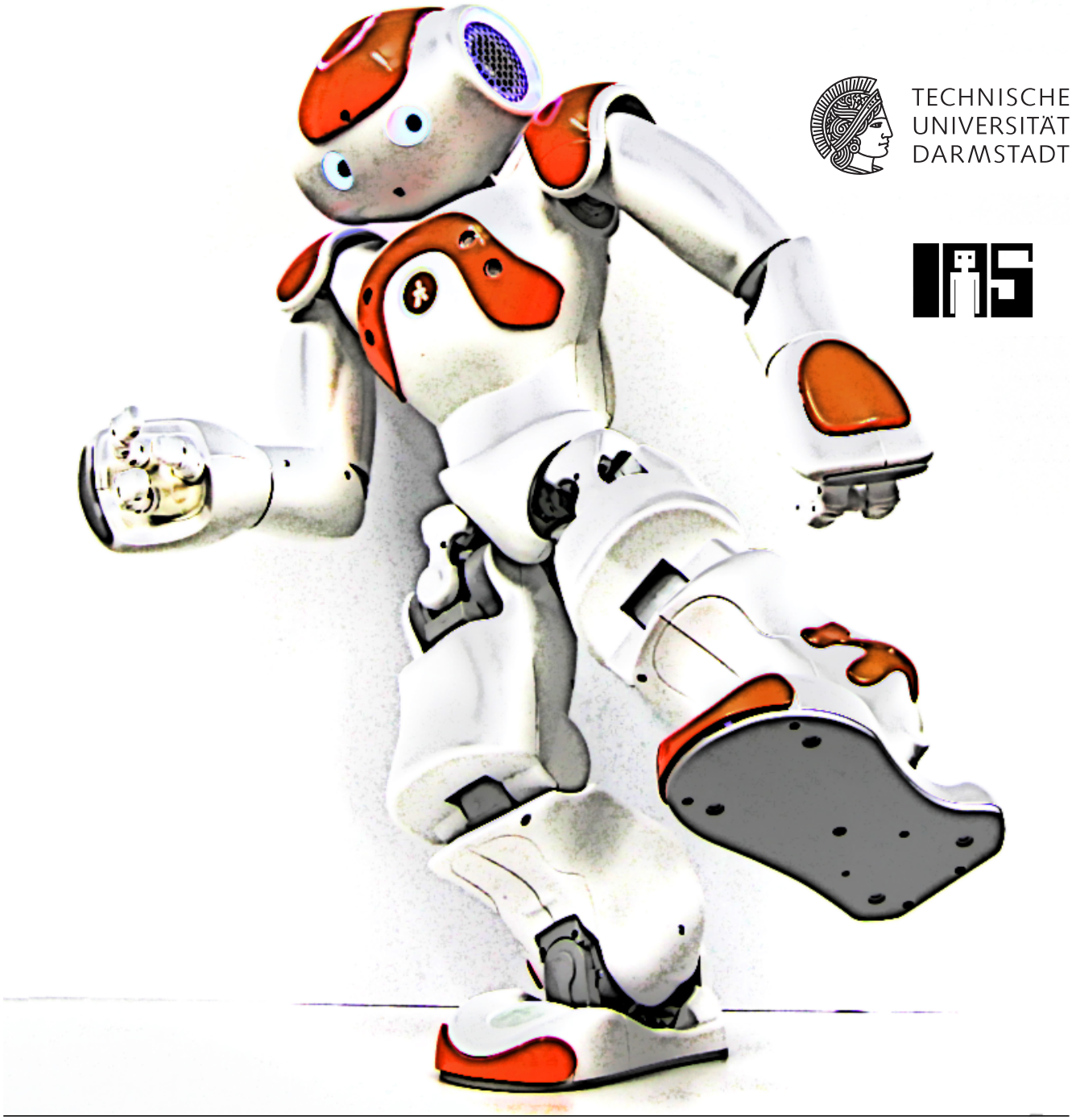
Latent Space Reinforcement Learning

Selbstverstärkendes Lernen in latenten Vektorräumen

Bachelor-Thesis von Kevin Sebastian Luck aus Hannoversch Münden

Mai 2014



TECHNISCHE UNIVERSITÄT DARMSTADT

IAS

Latent Space Reinforcement Learning
Selbstverstärkendes Lernen in latenten Vektorräumen

Vorgelegte Bachelor-Thesis von Kevin Sebastian Luck aus Hannoversch Münden

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Heni Ben Amor
3. Gutachten: Dr. Gerhard Neumann

Tag der Einreichung:

In memory of Cornelia Lutz

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 6. Mai 2014

(Kevin S. Luck)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, May 6, 2014

(Kevin S. Luck)

Abstract

Often we have to handle high dimensional spaces if we want to learn motor skills for robots. In policy search tasks we have to find several parameters to learn a desired movement. This high dimensionality in parameters can be challenging for reinforcement algorithms, since more samples for finding an optimal solution are needed with every additional dimension. On the other hand, if the robot has a high number of actuators, an inherent correlation between these can be found for a specific motor task, which we can exploit for a faster convergence. One possibility is to use techniques to reduce the dimensionality of the space, which is used as a pre-processing step or as an independent process in most applications. In this thesis we present a novel algorithm which combines the theory of policy search and probabilistic dimensionality reduction to uncover the hidden structure of high dimensional action spaces. Evaluations on an inverse kinematics task indicate that the presented algorithm is able to outperform the reference algorithms PoWER and CMA-ES, especially in high dimensional spaces. Furthermore we evaluate our algorithm on a real-world task. In this task, a NAO robot learns to lift his leg while keeping balance. The issue of collecting samples for learning on a real robot in such a task, which is often very time and cost consuming, is considered in here by using a small number of samples in each iteration.

Zusammenfassung

Versucht man für Roboter Bewegungsabläufe mithilfe von Verstärkendem Lernen, dem sogenannten Reinforcement Learning, zu lernen, muss man sich häufig mit hochdimensionalen Räumen auseinandersetzen. So hat beispielsweise bereits der NAO Roboter 26 Freiheitsgrade, hier Gelenkwinkel, mithilfer derer eine Bewegung umgesetzt werden kann. Zusätzlich kommen bei Policy Search Problemen noch Parameter für sog. Features hinzu, welche beispielsweise Gauss-Kurven sein können, die abhängig von der Zeit sind. Versuchen wir nun einen Bewegungsablauf zu erlernen, erreichen wir mit steigender Zahl der einzelnen Bewegungen relativ schnell eine sehr hohe Anzahl an Parametern aus der die Bewegungen abgeleitet werden. Diese Parameter müssen für einen optimalen Bewegungsablauf geschätzt werden und spannen einen hochdimensionalen Raum auf.

In hochdimensionalen Räumen, in denen die einzelnen (Gelenk-) Konfigurationen als Punkte liegen, kann man jedoch oft Unterräume finden, die eine stark reduzierte Anzahl an Dimensionen aufweisen. Die Dimensionsachsen dieser niedrig dimensionalen Räume kodieren hier Korrelationen zwischen den verschiedenen Parametern, wobei wir unser Augenmerk auf Korrelationen zwischen den Aktoren, z.B. Gelenke, legen werden. Ein intuitives Beispiel für solche Korrelationen sind z.B. Bewegungen einer (menschlichen) Hand, bei der die Gelenke der Finger oftmals in Abhängigkeit zueinander stehen.

In dieser Abschlussarbeit wird ein neuartiger Policy Search Algorithmus vorgestellt, der die versteckte latente Struktur in einem solchen hochdimensionalen Parameterraum ausnutzt und damit Reinforcement Learning und Dimensionsreduktion in einer Theorie vereint. Im Gegensatz zu früheren Ansätzen wird hierbei die Dimensionsreduktion nicht als ein Vorverarbeitungsschritt oder als ein unabhängiger Prozess eingesetzt, sondern direkt im Lernalgorithmus durchgeführt. Wie die durchgeführten Evaluationen mit einer Aufgabe der inversen Kinematik zeigen, kann insbesondere bei einer hohen Anzahl an Gelenkwinkel der vorgestellte Algorithmus eine deutlich bessere Konvergenz in Richtung eines optimalen Ergebnisses vorweisen als die zur Referenz verwendeten Algorithmen PoWER und CMA-ES. Weiterhin wird die Möglichkeit der Anwendung in realen Lern-Szenarien anhand eines Experiments mit einem NAO Roboter aufgezeigt, bei dem der Roboter die Fähigkeit erwerben soll, auf einem Bein zu stehen. In beiden Evaluationen wurde der normalerweise begrenzten Möglichkeit zur Erzeugung von Testläufen auf Robotern Rechnung getragen, indem eine möglichst kleine Anzahl an Testläufen pro Iteration gewählt wurde.

Acknowledgements

First, I want to thank my supervisor Heni Ben Amor, who gave me the opportunity and the freedom to work on the topic of this thesis and to develop my ideas. Especially for the transatlantic supervising, which was sometimes difficult but beneficial in the end.
I would also like to thank Gerhard Neumann, who was open for mathematical discussions all the time and has given me plenty of ideas for the future.
I owe the idea for the Lift-One-Leg Task on the NAO Robot to Erik Berger, who helped me with the simulation tool and did the physically test runs on the NAO robot in Freiberg.
Jan Peters for establishing my first contact to Heni when I was new to the group.
My two colleagues Hongh Linh Thai and especially Thomas Hesse, for all the discussions late in the night, when we had to work in the office to solve our problems for the deadlines. Actually, the name of the PePP$_C$Er Algorithm can be traced back to one of this conversations with Thomas.
Furthermore I gratefully acknowledge the many helpful suggestions of Annemarie Arnold, Judith Bönnighausen, Inga Schmidt and Wolfgang Heenes during the preparation of this thesis.
And all my friends, my family and especially my sister Susan, that they have let me work in solitude and privacy in the past months with my formulas.

Contents

Figures

List of Figures

Abbreviations, Symbols and Operators

List of Abbreviations

CMA-ES	Covariance Matrix Adaption Evolution Strategy
DOF	degrees-of-freedom
EM	Expectation Maximization
KL	Kullback-Leibler
ML	Machine Learning
MLE	Maximum Likelihood Estimation
PCA	Principle Component Analysis
p.d.f.	probability density function
PePPEr	'**P**olicy **Se**arch with **P**robabilistic **P**rinciple **C**omponent **Explor**ation in the Action Space'
PL	Policy Learning
PoWER	'**Po**licy learning by **W**eighting **E**xploration with the **R**eturns'
PPCA	Probabilistic Principal Component Analysis
RL	Reinforcement Learning

List of Symbols

Notation	Description
a	a single action
$\mathbf{a}$	the action vector
$\boldsymbol{\epsilon}$	a normal distributed (isotropic) exploration
$\mathbf{E}$	a matrix variate normal distributed exploration
$\boldsymbol{\phi}$	the feature vector
$\mathbf{I}$	the identity matrix

$\mathbf{Z}$	the latent matrix which contains different latent vectors
$\mathbf{z}$	the latent vector or variable
$\mathbf{W}$	the basis transformation matrix from the latent subspace into original
$\boldsymbol{\mu}$	the mean vector of a probability distribution
$\mathbf{M}$	the mean matrix of a probability distribution
$\mathbf{D}$	diagonal covariance matrix of a probability distribution
σ^2	variance of a probability distribution
$\boldsymbol{\theta}$	a set of parameters
r	a single reward for one step
$\mathbf{s}$	the current state
$\boldsymbol{\tau}$	a trajectory
$\mathbf{0}$	a matrix or vector which only contains zeros

List of Operators

Notation	Description	Operator
det	the determinant of a matrix	$\det(\cdot)$
diag	delivers the diagonal matrix of a matrix	$\operatorname{diag}(\cdot)$
$\mathbb{E}$	the expactation operator, it will be used if the context is clear	$\mathbb{E}[\cdot]$
$\mathbb{E}_x$	the expectation in respect of a random variable $\mathbf{x}$	$\mathbb{E}_x[\cdot]$
$\mathbb{E}_{x\|y}$	the expectation in respect of a random variable $\mathbf{x}$ conditioned on $\mathbf{y}$	$\mathbb{E}_{x\|y}[\cdot]$
$\otimes$	the kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$	$\mathbf{A} \otimes \mathbf{B}$
KL	Kullback-Leibler divergence	$\mathrm{KL}(\cdot \parallel \cdot)$
L	the likelihood function	$\mathrm{L}(\cdot)$
$\mathcal{L}$	the lower bound of a likelihood	$\mathcal{L}_\theta(\cdot)$
$\mathcal{N}$	(multivariate) normal distribution with mean $\boldsymbol{\mu}$ and covariance Σ	$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$
$\mathcal{N}_{p,n}$	matrix variate normal distribution with mean matrix $\mathbf{M}$ and covariance matrix Σ	$\mathcal{N}_{p,n}(\mathbf{M}, \Sigma)$
$\sim$	the random variable x is distributed with distribution $\mathcal{N}$	$x \sim \mathcal{N}$
π	stochastic policy	$\pi(\cdot\|\cdot)$
p	the probability of $\mathbf{x}$ (conditioned on $\mathbf{y}$)	$p(\mathbf{x}\|\mathbf{y})$
Q_t^π	the weightening based on the reward	$Q_t^\pi(\mathbf{s}_t, \mathbf{a})$
trace	delivers the trace of a matrix	$\operatorname{trace}(\cdot)$
vec	the vectorization of a matrix	$\operatorname{vec}(\cdot)$
var	the variance operator	$\operatorname{var}(\cdot)$
cov	the covariance operator	$\operatorname{cov}(\cdot, \cdot)$

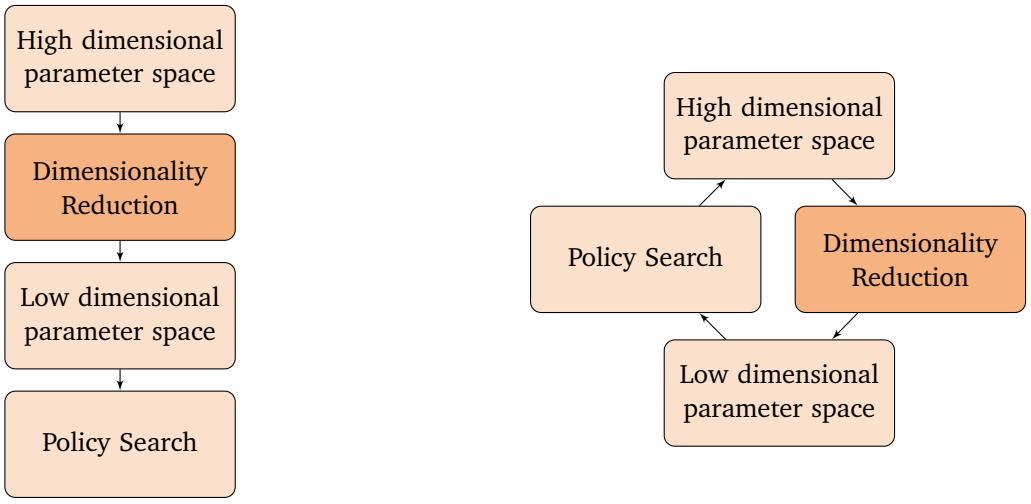
1 Introduction

Closely related to artificial intelligence is the vision of fully autonomous robots, that can perform tasks without the help or supervision of humans. A core component of this autonomous behavior is the ability of the robot to improve itself during the execution of tasks based on a reward. Developing algorithmic approaches that realize such learning abilities is the goal of the reinforcement learning (RL) community. Reinforcement learning has been applied to a broad field of applications in recent years including helicopter flight [1], robot table tennis [2], or quadruped locomotion [3].

An instance of RL-based techniques is *policy search*. Policy search methods try to find a policy π, which maximizes the expected long-term reward. In the case of a parameterized policy, i.e. linear Gaussian Models or Dynamic Motor Primitives [4], we have to determine several parameters for each degree-of-freedom. Thus, there is a strong connection between the number of parameters and degrees-of-freedom of a robot, which results in high dimensional parameter spaces in most real world tasks.

Admittedly, high dimensional parameter spaces often introduce problems, namely the *curse of dimensionality* [5]. With every new dimension, more trials need to be made to determine an optimal policy with policy search. This is due to the reinforcement learning nature of policy search, where we have to explore the parameter space to find improvements for the policy. In case of a naïve approach we need exponentially more samples. Furthermore, the simulation or the real trial on a robot is very time consuming and costly. Hence, we want to minimize the number of trials.

This gap between efficiency in terms of the number of trials and the effort for exploration in high dimensional spaces can be closed by the usage of dimensionality reduction techniques. The idea behind such techniques is the assumption that there are correlations between the axes of the high dimensional space such that the valid parameter vectors lie on a low dimensional manifold. A well-known method for finding low dimensional manifolds this is the Principle Component Analysis (PCA) [6] and its probabilistic version the Probabilistic Principal Component Analysis (PPCA) [7]. For example, by using PCA we can find such a manifold with 8 dimensions for the human hand, which has around 23 degrees-of-freedom (DOF). Solving optimization problems on this manifold is significantly easier than on the original high dimensional space.



(a) Pre-processing step **(b)** Independent process

Figure 1.1.: Two different and common used approaches to use dimensionality reduction techniques in policy search.

However, in current applications these techniques are usually used as a pre-processing step (see Fig. 1.1(a)), before reinforcement learning or as an independent process (see Fig. 1.1(b)): In [8], Deisenroth et al. projected the original 25-dimensional training inputs onto an 8-dimensional latent space for predicting a walking gait for a biped robot. For the approach presented in [9], Reduced Rank Regression was used to identify a low dimensional

subspace, which was used for learning on the real robot. In a similar manner, Bitzer et al. [10] used user-provided training data to learn a latent space with linear as well as nonlinear dimensionality reduction for robot learning.

However, this use of dimensionality reduction has serious limitations. First, using dimensionality reduction as a pre-processing step requires a significantly large training set of solutions for an accurate approximation of the assumed manifold. Often, such a large training set is not available or requires human demonstrations. Furthermore, we have to deal with the problem that if the latent space is chosen wrongly for the given task, the reinforcement algorithm may not be able to find a satisfying solution since it is not possible to change the parameters of the latent space anymore. Moreover, if we use human demonstrations, e.g., recorded joint configurations, we may have to deal with over-fitting if we assume a one-to-one mapping between the human and the robot body and hence neglect the *correspondence problem* [11]. On the other hand, using the dimensionality reduction as an independent process can decrease the possible learning efficiency if we are not using the reward information for the identification of the subspace. Furthermore it is debatable how to consider the dimensionality reduction in the policy model such that overfitting does not occur while learning in the low dimensional subspace. This can be a problem if we use only the low dimensional manifold for sampling such that the following adaptions of the manifold would depend on the initialization.

1.1 Motivation

The issues described naturally lead to the question, whether it is possible to merge the idea of policy search and dimensionality reduction to one technique. This is the main question which shall be answered by this thesis. Our aim is to find a theory, based on Expectation Maximization (EM) policy search, which allows us to develop a reinforcement learning algorithm able to find a latent space to perform a directed exploration. Thus the samples, which are used for policy search, lie on the manifold created by the previous samples with the highest reward.

This means that the dimensionality reduction is no longer a pre-processing step or done independently (see Fig. 1.2). Thus our policy search algorithm would be able to find a satisfying policy and a low dimensional manifold together in one process. The ability to find such a latent space online is satisfying the constraint that in the most cases we do not have a training set of possible solutions for initializations and that we are uncertain about the integrity of our current samples with respect to the representation of the possible manifold. Additionally, we will be able to use the rewards in a direct way for suggesting the latent space.

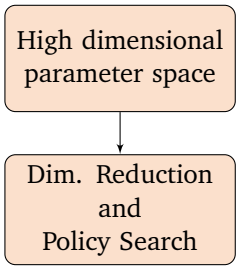


Figure 1.2.: The presented approach combines dimensionality reduction and policy search in one method.

1.2 Outline

The first chapter of this thesis will describe the required mathematical foundations and starts with a short introduction into the matrix variate normal distributions and EM algorithm. Subsequently, the theory behind PPCA will be highlighted, which serves as the main idea for the dimensionality reduction technique used later in this thesis and is based on EM. The discussion of the reinforcement learning framework of PoWER follows an overview of policy search from a probabilistic point of view and will be the basis of our theory for latent space reinforcement learning. The derivation of this theory is given in the second chapter, where the PePP$_{\text{C}}$Er Algorithm is introduced. The chapter is closed with a diagonal matrix extension for the PePP$_{\text{C}}$Er Algorithm. In the fourth chapter, we evaluate the introduced algorithm on a toy task where we need to learn inverse kinematics of a high-dimensional robot. Subsequently, we present the results for a real-world task on a NAO robot with the application of the PePP$_{\text{C}}$Er Algorithm. The goal was to let the robot lift one leg without loosing balance. This thesis is closed with a conclusion and a short description of potential future work.

Parts of this thesis were submitted on the International Conference on Intelligent Robots and Systems (IROS) 2014 under the title *Latent Space Policy Search for Robotics* [12].

2 Foundation

2.1 Matrix Variate Normal Distributions

In this section we introduce matrix variate normal distributions such that we are able to use them for the formulation of the PePP$_{\text{C}}$Er Algorithm later on. Furthermore, the use of them leads to an elegant way for the solve of several expectations that occur while using the EM algorithm.

The standard normal distribution $\mathcal{N}\left(\mu,\sigma^2\right)$, given by the probability density function (p.d.f.)

$$f(x)=\left(2\pi\sigma^2\right)^{-\frac{1}{2}}\exp\left\{-\left(2\sigma^2\right)^{-1}(x-\mu)^2\right\}, \tag{2.1}$$

can be extended to vectors and is then called the *multivariate* normal distribution. The p.d.f. for a vector $\mathbf{x}\in\mathbb{R}^n$ is given by

$$f(\mathbf{x})=(2\pi)^{-\frac{n}{2}}\det(\boldsymbol{\Sigma})^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\right\} \tag{2.2}$$

with a mean vector $\boldsymbol{\mu}\in\mathbb{R}^n$ and a covariance matrix $\boldsymbol{\Sigma}\in\mathbb{R}^{n\times n}$, which is symmetric and positive definite. We denote this distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$. Now let us assume, that we have a set of p samples $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_p$, each vector sampled from such a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$. If we organize this sample set as a matrix like

$$\left(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_p\right)^{\mathrm{T}}=\begin{pmatrix}x_{11}&x_{12}&\cdots&x_{1n}\\x_{21}&x_{22}&\cdots&x_{2n}\\\vdots&\vdots&\ddots&\vdots\\x_{p1}&x_{p2}&\cdots&x_{pn}\end{pmatrix} \tag{2.3}$$

it is possible to regard this as one random matrix instead of several multivariate samples. Furthermore, it is possible to describe a distribution based on random matrices like a multivariate distribution. In fact the theory of matrix variate distributions can be traced back to multivariate distributions.

The following definitions and theorems are taken from [13], where the interested reader can find the proofs and a deeper discussion of the theory of matrix variate distributions.

To introduce the matrix variate normal distribution, we need first the definitions of the Kronecker product between two matrices and the vectorization of a matrix.

Definition 1 *Given two matrices $\mathbf{A}\in\mathbb{R}^{m\times n}$ and $\mathbf{B}\in\mathbb{R}^{p\times q}$ the Kronecker product $\mathbf{A}\otimes\mathbf{B}$ is defined by*

$$\mathbf{A}\otimes\mathbf{B}=\begin{pmatrix}a_{11}\mathbf{B}&a_{12}\mathbf{B}&\cdots&a_{1n}\mathbf{B}\\a_{21}\mathbf{B}&a_{22}\mathbf{B}&\cdots&a_{2n}\mathbf{B}\\\vdots&\vdots&\ddots&\vdots\\a_{m1}\mathbf{B}&a_{m2}\mathbf{B}&\cdots&a_{mn}\mathbf{B}\end{pmatrix}\in\mathbb{R}^{mp\times nq}.$$

Definition 2 *Given a matrix $(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n)=\mathbf{A}\in\mathbb{R}^{m\times n}$, the vectorization of this matrix $\operatorname{vec}(\mathbf{A})$ is defined by*

$$\operatorname{vec}(\mathbf{A})=\begin{pmatrix}\mathbf{a}_1\\\vdots\\\mathbf{a}_n\end{pmatrix}\in\mathbb{R}^{nm}.$$

Now we can use the definition of the multivariate normal distribution to extend it to the matrix variate normal distributions.

Definition 3 *We call a random matrix $\mathbf{X} \in \mathbb{R}^{p\times n}$ distributed with a matrix variate distribution $\mathcal{N}_{p,n}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$, specified by the mean matrix $\mathbf{M} \in \mathbb{R}^{p\times n}$ and the covariance matrix $\mathbf{\Sigma}\otimes\mathbf{\Psi}$ with positive definite matrices $\mathbf{\Sigma} \in \mathbb{R}^{p\times p}$ and $\mathbf{\Psi} \in \mathbb{R}^{n\times n}$, if* $\operatorname{vec}\left(\mathbf{X}^{\mathrm{T}}\right) \sim \mathcal{N}\left(\operatorname{vec}\left(\mathbf{M}^{\mathrm{T}}\right), \mathbf{\Sigma}\otimes\mathbf{\Psi}\right)$ *holds.*

In the case that $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ are positive semidefinite matrices, an alternative definition has to be used which can be found in [13]. To distinguish between the multivariate normal distribution and the matrix variate distribution we will write $\mathbf{X} \sim \mathcal{N}_{p,n}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$ in the case of matrix variate normal distributions for random matrices $\mathbf{X} \in \mathbb{R}^{p\times n}$. Furthermore it is worthwhile to see that in addition to the given definition above a random matrix $\mathbf{X}$ is called matrix variate distributed with $\mathcal{N}_{p,n}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$ if we have $\operatorname{vec}(\mathbf{X}) \sim \mathcal{N}(\operatorname{vec}(\mathbf{M}), \mathbf{\Psi}\otimes\mathbf{\Sigma})$ under the described conditions.

Theorem 1 *Let $\mathbf{X}$ be a random matrix distributed with $\mathbf{X} \sim \mathcal{N}_{p,n}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$, then the p.d.f. is defined by*

$$f(\mathbf{X}) = (2\pi)^{-\frac{np}{2}} \det(\mathbf{\Sigma})^{-\frac{n}{2}} \det(\mathbf{\Psi})^{-\frac{p}{2}} \exp\left\{-\frac{1}{2}\operatorname{trace}\left(\mathbf{\Sigma}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{\Psi}^{-1}(\mathbf{X}-\mathbf{M})^{\mathrm{T}}\right)\right\}$$

The proof for this theorem can be found in [13, Thm. 2.2.1]. In this thesis, we will also need the following theorem for the matrix variate second order expectation.

Theorem 2 *Given a random matrix $\mathbf{X} \sim \mathcal{N}_{p,n}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$ with $\mathbf{X} \in \mathbb{R}^{p\times n}$, where the distribution is specified by the mean matrix $\mathbf{M} \in \mathbb{R}^{p\times n}$ and the covariance with $\mathbf{\Sigma} \in \mathbb{R}^{p\times p}$ and $\mathbf{\Psi} \in \mathbb{R}^{n\times n}$. For the expectation of the term $\mathbf{X}^{\mathrm{T}}\mathbf{A}\mathbf{X}$ with a matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$ we can find that*

$$I\!E\left[\mathbf{X}^{\mathrm{T}}\mathbf{A}\mathbf{X}\right] = \operatorname{trace}\left(\mathbf{\Sigma}\mathbf{A}^{\mathrm{T}}\right)\mathbf{\Psi} + \mathbf{M}^{\mathrm{T}}\mathbf{A}\mathbf{M}.$$

This theorem simplifies various steps in the later proofs, as there is no need to consider each dimension of an action vector independently with the theory of multivariate normal distributions.

2.2 Expectation Maximization

In this section a short introduction is given into the Expectation Maximization (EM) algorithm based on the introduced notations which will follow closely [14, pp.450] and [1, pp.43]. The EM algorithm is used to determine the maximum likelihood solution of a stochastic model with latent variables, which are here denoted by $\mathbf{z}$. Let a data set be given by $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots \mathbf{a}_T]$ then the goal of EM is to maximize the probability of the likelihood

$$\mathrm{p}_{\theta}(\mathbf{A}) = \prod_{t=1}^{T} \mathrm{p}_{\theta}(\mathbf{a}_t) = \prod_{t=1}^{T} \int_{\mathbb{Z}} \mathrm{p}_{\theta}(\mathbf{a}_t, \mathbf{z})\, \mathrm{d}\mathbf{z}, \tag{2.4}$$

where we assume independent and identically distributed data points $\mathbf{a}_t$. The distribution is determined by the parameters given by θ. Instead of the maximization of Eq. 2.4 we can maximize the log-likelihood given by

$$\ln \mathrm{p}_{\theta}(\mathbf{A}) = \sum_{t=1}^{T} \ln \int_{\mathbb{Z}} \mathrm{p}_{\theta}(\mathbf{a}_t, \mathbf{z})\, \mathrm{d}\mathbf{z}, \tag{2.5}$$

because the logarithm is a monotonically increasing function and so the maximum of $\mathrm{p}_{\theta}(\mathbf{A})$ is the same as for $\ln \mathrm{p}_{\theta}(\mathbf{A})$. Solving this equation for the parameters is difficult, while the logarithm is inside of the sum. Introducing the distribution $q(\mathbf{Z})$ and using the identity $\mathrm{p}_{\theta}(\mathbf{A}) = \frac{\mathrm{p}_{\theta}(\mathbf{A},\mathbf{Z})}{\mathrm{p}_{\theta}(\mathbf{Z}|\mathbf{A})}$ we get that

$$\begin{aligned} \ln \mathrm{p}_{\theta}(\mathbf{A}) &= \int_{\mathbb{Z}} q(\mathbf{Z}) \ln \mathrm{p}_{\theta}(\mathbf{A})\, \mathrm{d}\mathbf{Z} = \int_{\mathbb{Z}} q(\mathbf{Z}) \ln \frac{q(\mathbf{Z})\,\mathrm{p}_{\theta}(\mathbf{A},\mathbf{Z})}{q(\mathbf{Z})\,\mathrm{p}_{\theta}(\mathbf{Z}|\mathbf{A})}\mathrm{d}\mathbf{Z} \\ &= \int_{\mathbb{Z}} q(\mathbf{Z}) \ln \frac{\mathrm{p}_{\theta}(\mathbf{A},\mathbf{Z})}{q(\mathbf{Z})}\mathrm{d}\mathbf{Z} - \int_{\mathbb{Z}} q(\mathbf{Z}) \ln \frac{\mathrm{p}_{\theta}(\mathbf{Z}|\mathbf{A})}{q(\mathbf{Z})}\mathrm{d}\mathbf{Z} \\ &= \mathcal{L}_{\theta}(q) + \mathrm{KL}\left(q(\mathbf{Z}) \,\|\, \mathrm{p}_{\theta}(\mathbf{Z}|\mathbf{A})\right) \end{aligned} \tag{2.6}$$

holds (the verification can be found in [14, p.451]). The Kullback-Leibler (KL) divergence has the property that $\mathrm{KL}(q \parallel \mathrm{p}) \geq 0$ and $\mathrm{KL}(q \parallel \mathrm{p}) = 0$ if, and only if, the distributions q and p are equal. With this property we can now argue that the term $\mathcal{L}_\theta(q)$ is a lower bound of $\ln \mathrm{p}_\theta(\mathbf{A})$. The two iterative steps of the EM algorithm is first to minimize the KL divergence by setting it to zero and then to maximize the lower bound $\mathcal{L}_\theta(q)$, to increase the probability of the data set $\mathrm{p}_\theta(\mathbf{A})$.

To minimize the term $\mathrm{KL}(q(\mathbf{Z}) \parallel \mathrm{p}_\theta(\mathbf{Z}|\mathbf{A}))$ in the *expectation step* with respect to the old parameters, we have to update the distribution $q(\mathbf{Z})$ which means to equal $q(\mathbf{Z})$ and $\mathrm{p}_\theta(\mathbf{Z}|\mathbf{A})$. Therefore the KL divergence is zero and $\ln \mathrm{p}_\theta(\mathbf{A}) = \mathcal{L}_\theta(q)$. Now, in the *maximization step*, we can estimate the new parameters with

$$\begin{aligned} \theta_{new} &= \arg\max_\theta \mathcal{L}_\theta(q) = \arg\max_\theta \int_{\mathbb{Z}} q(\mathbf{Z}) \ln \mathrm{p}_\theta(\mathbf{A}, \mathbf{Z}) \, \mathrm{d}\mathbf{Z} - \int_{\mathbb{Z}} q(\mathbf{Z}) \ln q(\mathbf{Z}) \, \mathrm{d}\mathbf{Z} \\ &= \arg\max_\theta \mathbb{E}_{q(\mathbf{Z})}\left[\ln \mathrm{p}_\theta(\mathbf{A}, \mathbf{Z})\right] + \mathrm{const} = \arg\max_\theta \mathcal{Q}_\theta(q) + \mathrm{const} \end{aligned} \tag{2.7}$$

where we can drop the constant term. From the *expectation step* we can now be substitute $q(\mathbf{Z})$ with $\mathrm{p}_\theta(\mathbf{Z}|\mathbf{A})$ and so the estimation of the new parameters can be done with Maximum Likelihood Estimation (MLE). Because the KL divergence can only be positive or zero, we can be sure that this maximization of the lower bound leads to an increase of the probability $\ln \mathrm{p}_\theta(\mathbf{A})$ or at least to the same probability as before. This is due to the fact that we chose the KL divergence to be zero before maximization.

2.3 Dimensionality Reduction with Probabilistic Principal Component Analysis

A standard technique to handle high dimensional spaces is Principle Component Analysis (PCA) [6] which is able to find a low dimensional subspace for a data set. This method has become popular in a broad field of applications where spaces with the above described property have to be handled, i.e., meteorology [15], medicine [16] or face recognition [17]. A good example for the application of PCA is grasping in robotics. The human hand has around 25 DOF, so each hand configuration lies in a 25 dimensional space. But it has been found that about 7 dimension retain enough information to represent each hand configuration with only a small amount of information loss [18]. In this section, a brief overview of the probabilistic version of PCA is given. For the standard PCA we refer the reader to [19].

In contrast to PCA, where we assume to have the full data set, the Probabilistic Principal Component Analysis (PPCA) [7] is able to model the uncertainty in the case of missing data. Furthermore, the PPCA is formulated as a sampling technique from the low dimensional into the high dimensional space.

Assume a d-dimensional data point $\mathbf{x} \in \mathbb{R}^d$ in a high dimensional space, we can find a linear Gaussian model

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{2.8}$$

where the low dimensional latent variable $z \in \mathbb{R}^n$ is Gaussian distributed according to $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. This latent variable is projected with the transformation matrix $\mathbf{W} \in \mathbb{R}^{d \times n}$ into the high dimensional space. The columns of $\mathbf{W}$ span the low-dimensional subspace, but these basis vectors do not have to be orthogonal to each other like in PCA. The mean of this low dimensional subspace is given by the vector $\boldsymbol{\mu} \in \mathbb{R}^d$. To model the uncertainty about the missing data, the isotropic error term $\boldsymbol{\epsilon} \in \mathbb{R}^d$ is used with the Gaussian distribution $p(\boldsymbol{\epsilon}) = \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right)$. This isotropic error leads to a noise added to the principle components, in contrast to PCA where these projections would lie directly on them. So the resulting Gaussian distribution of Eq. 2.8 is given by

$$p(\mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2 \mathbf{I}\right). \tag{2.9}$$

As our model contains latent variables, the parameters $\mathbf{W}, \sigma^2$ and $\boldsymbol{\mu}$ can be determined for the maximization of the probability $p(\mathbf{X})$ with the iterative EM algorithm.

Because the ability to project samples from the manifold to the high dimensional space, PPCA seems to be ideal to be used as underlying theoretical foundation for the exploration method in our desired latent space RL method.

2.4 Policy Search

Most tasks for a robot can be formulated as a policy search problem. The goal of policy search is to find a control policy $\pi(\mathbf{s}_t, t)$ that provides an action $\mathbf{a}_t \in \mathsf{A}$ for every given state $\mathbf{s}_t \in \mathsf{S}$ and time t of the system. The execution of

this action on the robot results in a new state $\mathbf{s}_{t+1}$ and a reward $r_t(\mathbf{s}_t, \mathbf{a}_t)$. Instead of a deterministic policy we will use a stochastic policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t)$ given as a conditional probability distribution with parameters θ. Furthermore, we assume that the policy is linear in the form of $\mathbf{a}_t = \mathbf{M}_\theta \boldsymbol{\phi}(\mathbf{s}_t, t)$ where the matrix $\mathbf{M}_\theta$ depends on the given parameters and the time t. The feature vector $\boldsymbol{\phi}(\mathbf{s}_t, t) = \left(\boldsymbol{\phi}_1(\mathbf{s}_t, t), \boldsymbol{\phi}_2(\mathbf{s}_t, t), \ldots, \boldsymbol{\phi}_p(\mathbf{s}_t, t)\right)^{\mathrm{T}}$ contains basis functions $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_p$ which depend on the current time and the state. Examples for such features are time-dependent Gaussians.The goal of policy search is to determine the parameters θ s.t. the *expected return*

$$J(\theta) = \mathbb{E}_{p_\theta(\boldsymbol{\tau})}[R(\boldsymbol{\tau})] = \int_{\mathbb{T}} p_\theta(\boldsymbol{\tau}) R(\boldsymbol{\tau}) \, \mathrm{d}\boldsymbol{\tau} \tag{2.10}$$

is maximized. This expectation integrates over all possible trajectories $\boldsymbol{\tau}$, given by the set $\mathbb{T}$. The trajectories are given by the sequence of states and actions $\boldsymbol{\tau} = [\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}]$. The reward of each trajectory is given by $R(\boldsymbol{\tau})$, which we assume to be the *accumulated immediate rewards* r_t with

$$R(\boldsymbol{\tau}) = \sum_{t=1}^{T} r_t(\mathbf{s}_t, \mathbf{a}_t) + r_{T+1}(\mathbf{s}_T), \tag{2.11}$$

where the reward r_{T+1} denotes the final reward for reaching the state $\mathbf{s}_{T+1}$. For the probability of a trajectory, we assume the *Markov property* and so we get

$$p_\theta(\boldsymbol{\tau}) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \, \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t). \tag{2.12}$$

On the right side of the term we can find our policy that is to be determined, while the state probabilities on the left side will disappear in Eq. 2.14 later on.

2.5 The Reinforcement Learning Framework of PoWER

Policy Search methods, which are based on EM, formalize the problem statement as an inference problem with latent variables. In order to do so, the rewards have to be transformed into (improper) probability distributions such that each binary reward event is an (unnormalized) probability. In the case of a reward signal where a small number is good and a large bad, this can be easily done by using the exponential function with

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = \exp\left\{-\lambda r'_t(\mathbf{s}_t, \mathbf{a}_t)\right\}, \tag{2.13}$$

where $r'_t(\mathbf{s}_t, \mathbf{a}_t)$ is the actual reward and λ a (temperature) parameter to adjust the exponential function. Here the exponential function ensures that the reward r_t is non-negative. From now on we will assume that our reward r_t is such a transformed reward. With EM we can now find a lower bound of the expected return from Eq. (2.10), which is, according to Kober and Peters [20], given by

$$\begin{aligned} L_{\theta_{\text{old}}}(\theta) &= \int_{\mathbb{T}} p_{\theta_{\text{old}}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) \ln p_\theta(\boldsymbol{\tau}) \, \mathrm{d}\boldsymbol{\tau} \\ &= \mathbb{E}_{p_{\theta_{\text{old}}}(\boldsymbol{\tau})}\left[\left(\sum_{t=1}^{T} \ln \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t)\right) R(\boldsymbol{\tau})\right] \\ &= \mathbb{E}_{p_{\theta_{\text{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T} Q^\pi(\mathbf{s}_t, \mathbf{a}_t, t) \ln \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t)\right], \end{aligned} \tag{2.14}$$

where θ_{old} are the old parameters and θ the new ones,which we have to determine with Maximum Likelihood Estimation (MLE). The trajectories are here the latent variables while the rewars be the observations. The function $Q^\pi(\mathbf{s}_t, \mathbf{a}_t, t)$ is defined as the state-action value function given by

$$Q^\pi(\mathbf{s}_t, \mathbf{a}, t) = \mathbb{E}\left[\sum_{\tilde{t}=t}^{T} r_{\tilde{t}}(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}) \, |\mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right]. \tag{2.15}$$

We can estimate $Q^\pi(\mathbf{s}_t, \mathbf{a}, t)$ by a single rollout with

$$Q^\pi\left(\mathbf{s}_t^i, \mathbf{a}_t^i, t\right) \approx \sum_{\tilde{t}=t}^{T} r_{\tilde{t}}^i \tag{2.16}$$

where i denotes the index of the rollout. For the sake of simplicity, we will write Q_t^π instead of $Q^\pi(\mathbf{s}_t, \mathbf{a}, t)$. An advantage of this approach is, that we can obtain the parameters by weighted MLE update and so there is no need for a user-specific *learning rate*, which can be a crucial parameter in policy gradient algorithms [21]. The policy $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t, t)$ can be modelled as a linear Gaussian model. Such policies are, for example, used in the '**Po**licy learning by **W**eighting **E**xploration with the **R**eturns' (PoWER) algorithm [20].

In the stochastic policy of PoWER, given by $a_t = (\mathbf{m} + \boldsymbol{\epsilon}_t)^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}, t)$ for each dimension of the control action, a Gaussian noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is added to the current solution $\mathbf{m} \in \mathbb{R}^d$. The use of Eq. (2.14) leads to the update rule for PoWER given by

$$\mathbf{m}_{t+1} = \mathbf{m}_t + \left(\sum_{t=1}^{T} \mathbf{B}(\mathbf{s}_t, t) Q_t^\pi\right)^{-1} \cdot \left(\sum_{t=1}^{T} \mathbf{B}(\mathbf{s}_t, t) \boldsymbol{\epsilon}_t Q_t^\pi\right) \tag{2.17}$$

with $\mathbf{B}(\mathbf{s}_t, t) = \boldsymbol{\phi}(\mathbf{s}_t, t) \boldsymbol{\phi}(\mathbf{s}_t, t)^{\mathrm{T}} \left(\boldsymbol{\phi}(\mathbf{s}_t, t)^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\phi}(\mathbf{s}_t, t)\right)^{-1}$. In the case of a non-static $\boldsymbol{\Sigma}$, a similar update rule for $\boldsymbol{\Sigma}$ can be found in [20]. With the current framework we are not able to introduce latent variables in our stochastic policy so far. With these latent variables we would be able to use a (stochastic) linear model of a low dimensional subspace for the exploration in policy search. This extension, which is the main contribution of this thesis, will be discussed in the next chapter.

3 Reinforcement Learning in Spaces with Low Intrinsic Dimensionality

3.1 Using the PoWER Framework for Estimating Policies with Latent Variables

The EM framework of PoWER, given in Eq. (2.14), in association with weighted MLE can be exploited for the use of stochastic policies with latent variables. In spirit of the common notations, $\mathbf{z}$ is used for latent variables and $\mathbf{Z} \in \mathbb{R}^{m \times n}$ for latent matrices. Furthermore, for a consistent notation, $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t)$ is written as $\mathrm{p}_\theta(\mathbf{a}_t|\mathbf{s}_t, t)$. In order to add latent variables into the policy , we make use of the marginalization rule (see [14]) and define $\mathrm{p}_\theta(\mathbf{a}_t|\mathbf{s}_t, t) = \int_{\mathbb{Z}} \mathrm{p}_\theta(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)\, \mathrm{d}\mathbf{z}$. If EM is applied a new lower bound for Eq. (2.14) can be found with

$$\begin{aligned} \mathcal{L}_{\theta_{\text{old}}}(\theta) &= \mathbb{E}_{\mathrm{p}_{\theta_{\text{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T} Q^\pi(\mathbf{s}_t, \mathbf{a}_t, t) \ln \int_{\mathbb{Z}} \mathrm{p}_\theta(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)\, \mathrm{d}\mathbf{z}\right] \\ &\geq \mathbb{E}_{\mathrm{p}_{\theta_{\text{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T} Q^\pi(\mathbf{s}_t, \mathbf{a}_t, t)\, \mathbb{E}_{q(\mathbf{z}|\mathbf{a}_t, \mathbf{s}_t)}\left[\ln \mathrm{p}_\theta(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)\right]\right] \\ &= \mathcal{L}'_{\theta_{\text{old}}, q}(\theta), \end{aligned} \tag{3.1}$$

where the distribution

$$q(\mathbf{z}|\mathbf{a}_t, \mathbf{s}_t) = \frac{\mathrm{p}_{\theta_{\text{old}}}(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)}{\mathrm{p}_{\theta_{\text{old}}}(\mathbf{a}|\mathbf{s}_t, t)} \tag{3.2}$$

is given by the posterior distribution $\mathrm{p}_{\theta_{\text{old}}}(\mathbf{z}|\mathbf{a}_t, \mathbf{s}_t)$ of the latent variables. It is worth to recognize that we have applied EM twice in this lower bound. First to derive the policy update by weighted MLE and second, to update the joint distribution $\mathrm{p}_\theta(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)$. This lower bound can be used for any latent variable model for stochastic policies. In the next sections, a model for the stochastic policy, which uses latent variables, to perform dimensionality reduction will be discussed.

3.2 Low Dimensional and Stochastic Policy Models

If we investigate the PoWER algorithm, we notice that this algorithms handles each dimension of the action space separately. But this seems only reasonable if we have to fulfill a task where the actions, i.e. the angles of a robot arm or hand, are independent for each time step. The fact is, that in reality we have a lot of dependencies between actions for the most common tasks, however this dependencies may not be obvious. In our algorithm, we want to take advantage of this hidden structure in the action space for a faster reinforcement learning in policy search tasks. Since we assume that we start learning without any knowledge or already sampled configurations, we have to learn the structure of this hidden space online. With the described exploitation for latent variables in Eq.(3.1) we are able to learn the properties of the hidden subspace and the optimal policy for the robot with one unified algorithm. Revisiting the model of PPCA given by

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{3.3}$$

where the axes of the low dimensional subspace are given by the columns of $\mathbf{W}$, we can find a stochastic policy for uncovering a hidden structure in the action space. This stochastic policy can be found with

$$\mathbf{a} = \left(\mathbf{W}\mathbf{Z}^{\mathrm{T}} + \mathbf{M} + \mathbf{E}\right)\boldsymbol{\phi} = \mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}, \tag{3.4}$$

and the random matrices $\mathbf{E} \sim \mathcal{N}_{d,p}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$ and $\mathbf{Z}^{\mathrm{T}} \sim \mathcal{N}_{n,p}(\mathbf{0}, \mathbf{I})$, while $\mathbf{M} \in \mathbb{R}^{d \times p}$ is the mean matrix. From now on we will use d for the number of dimensions of the high dimensional action space, n for the number of the low

dimensional space and p for the number of parameters respectively the size of the feature vector $\boldsymbol{\phi} \in \mathbb{R}^p$. As was mentioned before we will write $\boldsymbol{\phi}$ instead of $\boldsymbol{\phi}(\mathbf{s}_t, t)$ for simplification, while keeping the dependencies in mind. Now let us take a closer look at the matrices and their intuition: Like in PPCA the matrix $\mathbf{W} \in \mathbb{R}^{d \times n}$ is a linear transformation from the hidden subspace into the high dimensional action space. The term $\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)$, which corresponds to a low dimensional action vector, can be seen as the representation of the high dimensional action in the subspace, whose mean action is given by $\mathbf{M}\boldsymbol{\phi}$. Finally, the term $\mathbf{E}\boldsymbol{\phi}$ encodes the uncertainty as an isotropic noise. In a later section, we will introduce a generalization of this model where this term is modelled by a diagonal covariance matrix where we use one parameter per action dimension. In terms of policy search, we can see the term $\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)$ as an directed exploration noise and $\mathbf{E}\boldsymbol{\phi}$ as an undirected exploration noise. The parameters for this model are $\mathbf{W}, \mathbf{M}$ and σ^2.

3.3 Linear Model Properties

In this section we investigate the properties of the probability distributions which arise as a result from the above given model. The expectation of our model in Eq. (3.3) can be found with

$$\begin{aligned}\mathbb{E}[\mathbf{a}] &= \mathbb{E}\left[\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}\right] \\ &= \underbrace{\mathbb{E}\left[\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)\right]}_{=0} + \mathbf{M}\boldsymbol{\phi} + \underbrace{\mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\right]}_{=0} \\ &= \mathbf{M}\boldsymbol{\phi}\end{aligned} \tag{3.5}$$

while the covariance of the model can be written as

$$\begin{aligned}\mathrm{cov}(\mathbf{a}) &= \mathbb{E}\left[\left(\left(\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}\right) - \mathbb{E}[\mathbf{a}]\right)\left(\left(\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}\right) - \mathbb{E}[\mathbf{a}]\right)^{\mathrm{T}}\right] \\ &= \mathbb{E}\left[\left(\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{E}\boldsymbol{\phi}\right)\left(\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{E}\boldsymbol{\phi}\right)^{\mathrm{T}}\right] \\ &= \mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\right] + \underbrace{\mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\right]}_{=0} + \underbrace{\mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{E}^{\mathrm{T}}\right]}_{=0} + \underbrace{\mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\right]}_{=0} + \mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{E}^{\mathrm{T}}\right] \\ &= \mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\right] + \mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{E}^{\mathrm{T}}\right].\end{aligned} \tag{3.6}$$

At this point, we have to use Thm. 2 for the expectation of matrix variate distribution. If we apply this theorem we get

$$\begin{aligned}\mathrm{cov}(\mathbf{a}) &= \mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\right] + \mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{E}^{\mathrm{T}}\right] \\ &= \mathbf{W}\,\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\,\mathbf{W}^{\mathrm{T}} + \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\sigma^2\mathbf{I} \\ &= \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right)\end{aligned} \tag{3.7}$$

for the covariance of our linear model. Thus the distribution for $\mathbf{a}$ is given as

$$\mathrm{p}(\mathbf{a}) = \mathcal{N}\left(\mathbf{M}\boldsymbol{\phi}, \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right)\right). \tag{3.8}$$

If we revisit our model equation, we can find that our latent variables are given by the random matrix $\mathbf{Z}$. At a first glance, it seems obvious to use the distribution $\mathrm{p}(\mathbf{a}|\mathbf{Z})$ as posterior distribution for EM. But in order to apply Bayes theorem for Gaussian variables (see or Appendix A.2 or [14, p.93]) in an easy way, we need vectors as latent variables instead of matrices. For this simplification, we can use the term $\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}$, whose result is a vector, as latent variable. Hence we have to estimate the posterior distribution $\mathrm{p}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}\right)$. For the properties of the distribution $\mathrm{p}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)$, we can use the fact that $\mathbb{E}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right] = 0$ holds and so the covariance of this distribution is reduced to the term $\mathbb{E}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\right]$, where Thm. 2 can be applied again and we get

$$\mathrm{p}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) = \mathcal{N}\left(\mathbf{0}, \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\right) \tag{3.9}$$

as a multivariate normal distribution. Given the model Eq. and its distribution in (3.8) we can find that

$$\mathrm{p}\left(\mathbf{a}|\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) = \mathcal{N}\left(\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi}, \sigma^2\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\right) \tag{3.10}$$

holds. With these distributions and Bayes theorem for Gaussian variables we can derivate the posterior distribution with

$$\mathrm{p}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}\right) = \mathcal{N}\left(\mathbf{C}\mathbf{W}^{\mathrm{T}}\left(\mathbf{a} - \mathbf{M}\boldsymbol{\phi}\right), \mathbf{C}\sigma^2\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right) \tag{3.11}$$

where $\mathbf{C} = \left(\sigma^2\mathbf{I} + \mathbf{W}^{\mathrm{T}}\mathbf{W}\right)^{-1}$.

3.4 Expectation and Maximization of the Parameters

With the distributions given in the previous section, we can now define the expectation and the maximization steps. Instead of the so far used notation $\mathbb{E}_{p_{\theta_{old}}(\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a})}\left[\mathbf{Z}^T\boldsymbol{\phi}\right]$ for the expectation of the latent variables , we will use the notation $\mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^T\boldsymbol{\phi}\right]$.

3.4.1 Expectation

In the expectation step of the EM algorithm, we have to estimate the parameters of the probability distribution $p_\theta\left(\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}\right)$. To do so, we can use the old parameters θ_{old} from the previous EM iteration. We can directly estimate this step, because we only need the expectation of this distribution, with

$$\mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^T\boldsymbol{\phi}\right] = \mathbf{C}\mathbf{W}^T\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right) \tag{3.12}$$

and

$$\mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^T\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T\right] = \mathbf{C}\sigma^2\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right) + \mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^T\boldsymbol{\phi}\right]\cdot\mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^T\boldsymbol{\phi}\right]^T \tag{3.13}$$

with $\mathbf{C} = \left(\sigma^2\mathbf{I}+\mathbf{W}^T\mathbf{W}\right)^{-1}$ like above.

3.4.2 Maximization

Now we have to find the parameters for which we maximize the expected return. To do so, we have to maximize our lower bound given in Eq. (3.1) which is equivalent to doing a weighted MLE for our parameters θ. The parameters of our model are $\mathbf{W}, \mathbf{M}$ and σ^2 like mentioned in Sec. 3.2.

A common way to estimate the mean parameter, here $\mathbf{M}$, is to do a MLE. Thus, $\mathbf{M}$ is given by

$$\begin{aligned}
\frac{\partial \ln p(\mathbf{a})}{\partial \mathbf{M}} &= \frac{\partial}{\partial \mathbf{M}}\left(-\frac{1}{2}\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^T\mathbf{D}^{-1}\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)\right)\\
&= \frac{\partial}{\partial \mathbf{M}}\left(-\frac{1}{2}\left(-2\mathbf{a}^T\mathbf{D}^{-1}\mathbf{M}\boldsymbol{\phi}+\boldsymbol{\phi}^T\mathbf{M}^T\mathbf{D}^{-1}\mathbf{M}\boldsymbol{\phi}\right)\right)\\
&\overset{A.1}{=\!=} \left(-\frac{1}{2}\left(-2\mathbf{D}^{-1}\mathbf{a}\boldsymbol{\phi}^T+\frac{\partial \boldsymbol{\phi}^T\mathbf{M}^T\mathbf{D}^{-1}\mathbf{M}\boldsymbol{\phi}}{\partial \mathbf{M}}\right)\right)\\
&\overset{A.2}{=\!=} \left(-\frac{1}{2}\left(-2\mathbf{D}^{-1}\mathbf{a}\boldsymbol{\phi}^T+2\mathbf{D}^{-1}\mathbf{M}\boldsymbol{\phi}\boldsymbol{\phi}^T\right)\right)\\
&= \left(\mathbf{D}^{-1}\left(\mathbf{a}\boldsymbol{\phi}^T-\mathbf{M}\boldsymbol{\phi}\boldsymbol{\phi}^T\right)\right)
\end{aligned} \tag{3.14}$$

where $\mathbf{D} = \left(\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)\left(\mathbf{W}\mathbf{W}^T+\sigma^2\mathbf{I}\right)\right) = \mathbf{D}^T$ holds. In this derivatives we made use of given rules for matrix differentiation at scalar forms, which can be found in the Appendix A.1. Now we can insert this result into the term

$$0 = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{\partial \ln p(\mathbf{a})Q_t^\pi}{\partial \mathbf{M}}\right] \tag{3.15}$$

which is the weighted MLE step and if we solve this equation for $\mathbf{M}$ we find

$$\begin{aligned}
&0 = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{\partial \ln p(\mathbf{a})Q_t^\pi}{\partial \mathbf{M}}\right]\\
\Leftrightarrow &0 = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbf{D}^{-1}\left(\mathbf{a}\boldsymbol{\phi}^T-\mathbf{M}\boldsymbol{\phi}\boldsymbol{\phi}^T\right)Q_t^\pi\right]\\
\Leftrightarrow &0 = \left(\mathbf{W}\mathbf{W}^T+\sigma^2\mathbf{I}\right)^{-1}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)}\left(\mathbf{a}\boldsymbol{\phi}^T-\mathbf{M}\boldsymbol{\phi}\boldsymbol{\phi}^T\right)Q_t^\pi\right]\\
\Leftrightarrow &\mathbf{M}\cdot\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)}\boldsymbol{\phi}\boldsymbol{\phi}^TQ_t^\pi\right] = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)}\mathbf{a}\boldsymbol{\phi}^TQ_t^\pi\right]\\
\Leftrightarrow &\mathbf{M} = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)}\mathbf{a}\boldsymbol{\phi}^TQ_t^\pi\right]\cdot\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)}\boldsymbol{\phi}\boldsymbol{\phi}^TQ_t^\pi\right]\right)^{-1}.
\end{aligned} \tag{3.16}$$

For the maximization of $\mathbf{W}$ and σ^2, we have to use our lower bound, where we can find for $\ln p(\mathbf{a}, \mathbf{Z})$ that

$$\begin{aligned}
\frac{\partial \ln p\left(\mathbf{a}, \mathbf{Z}^T\boldsymbol{\phi}\right)}{\partial \boldsymbol{\theta}_{new}} &= \frac{\partial}{\partial \boldsymbol{\theta}_{new}} \left(\ln p\left(\mathbf{a}\middle|\mathbf{Z}^T\boldsymbol{\phi}\right) + \ln p\left(\mathbf{Z}^T\boldsymbol{\phi}\right)\right) \\
&= \frac{\partial}{\partial \boldsymbol{\theta}_{new}} \left(\ln\left(\sqrt{\det\left(2\pi\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)\mathbf{I}\right)}^{-1}\right) + \right. \\
&\qquad \left. -\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^T \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\right) \\
&= \frac{\partial}{\partial \boldsymbol{\theta}_{new}} \left(-\frac{d}{2}\ln\left(2\pi \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)\right) - \frac{d}{2}\ln\left(\sigma^2\right)\right. \\
&\qquad \left. -\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^T \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\right)
\end{aligned} \tag{3.17}$$

holds if we differentiate this term w.r.t. one of the parameters. Doing so for the parameter $\mathbf{W}$ and using the derivatives of the scalar forms we get

$$\begin{aligned}
\frac{\partial \ln p\left(\mathbf{a}, \mathbf{Z}^T\boldsymbol{\phi}\right)}{\partial \mathbf{W}} &\overset{3.17}{=\!=} \frac{\partial}{\partial \mathbf{W}} \left(-\frac{d}{2}\ln\left(2\pi \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)\right) - \frac{d}{2}\ln\left(\sigma^2\right) + \right. \\
&\qquad \left. -\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^T \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\right) \\
&= \frac{\partial}{\partial \mathbf{W}} \left(-\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^T \left(\mathbf{a} - \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\right) \\
&= \frac{\partial}{\partial \mathbf{W}} \left(-\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \cdot \right. \\
&\qquad \left. \left(-\mathbf{a}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} - \boldsymbol{\phi}^T\mathbf{Z}\mathbf{W}^T\mathbf{a} + \boldsymbol{\phi}^T\mathbf{Z}\mathbf{W}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} + \boldsymbol{\phi}^T\mathbf{Z}\mathbf{W}^T\mathbf{M}\boldsymbol{\phi} + \boldsymbol{\phi}^T\mathbf{M}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi}\right)\right) \\
&= \frac{\partial}{\partial \mathbf{W}} \left(-\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(-2\mathbf{a}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} + \boldsymbol{\phi}^T\mathbf{Z}\mathbf{W}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi} + 2\boldsymbol{\phi}^T\mathbf{M}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi}\right)\right) \\
&\overset{A.1}{=\!=} -\frac{1}{2\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(-2\mathbf{a}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T + \frac{\partial \boldsymbol{\phi}^T\mathbf{Z}\mathbf{W}^T\mathbf{W}\mathbf{Z}^T\boldsymbol{\phi}}{\partial \mathbf{W}} + 2\mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T\right) \\
&\overset{A.2}{=\!=} -\frac{1}{\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(-\mathbf{a}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T + \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T + \mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T\right).
\end{aligned} \tag{3.18}$$

When we put this result into the weighted MLE for our lower bound we have

$$\begin{aligned}
0 &= \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \frac{\partial}{\partial \mathbf{W}} \mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\ln p\left(a, \mathbf{Z}^T\boldsymbol{\phi}\right)\right] Q_t^{\pi}\right] \\
&= \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[-\frac{1}{\sigma^2 \text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(-\mathbf{a}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T + \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T + \mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T\right)\right] Q_t^{\pi}\right] \\
&= \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \left(\mathbf{a}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T - \mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T\right)\right] Q_t^{\pi}\right] - \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathbb{E}_{\mathbf{Z}^T\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\text{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^T\right)} \mathbf{W}\mathbf{Z}^T\boldsymbol{\phi}\left(\mathbf{Z}^T\boldsymbol{\phi}\right)^T\right] Q_t^{\pi}\right]
\end{aligned} \tag{3.19}$$

where we can eliminate σ^2 because it does not depend on an index,i.e., it is a constant. Solving this equation for $\mathbf{W}$, results in the following update equation

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}\right]=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}-\mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right)\right]Q_t^{\pi}\right]\\
\Leftrightarrow&\mathbf{W}\,\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}\right]=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}-\mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right)\right]Q_t^{\pi}\right]\\
\Leftrightarrow&\mathbf{W}=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}-\mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right)\right]Q_t^{\pi}\right]\cdot\\
&\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}\right]\right)^{-1}\\
\Leftrightarrow&\mathbf{W}=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}-\mathbf{M}\boldsymbol{\phi}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}\right)Q_t^{\pi}\right]\cdot\\
&\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}\right]\right)^{-1}\\
\Leftrightarrow&\mathbf{W}=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}Q_t^{\pi}\right]\cdot\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}\right]\right)^{-1}.
\end{aligned}
\tag{3.20}
$$

In an analogous manner, we can find the derivative w.r.t. σ^2 with

$$
\begin{aligned}
\frac{\partial\ln\mathrm{p}\left(\mathbf{a},\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)}{\partial\sigma^2}&\overset{3.17}{=\!=}\frac{\partial}{\partial\sigma^2}\left(-\frac{d}{2}\ln\left(\sigma^2\right)-\frac{1}{2\sigma^2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)\right)\\
&=-\frac{d}{2\sigma^2}+\frac{1}{2\left(\sigma^2\right)^2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)
\end{aligned}
\tag{3.21}
$$

and if we plug this result into the lower bound we get

$$
\begin{aligned}
0&=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{\partial}{\partial\sigma^2}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\ln\mathrm{p}\left(\mathbf{a},\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)\right]Q_t^{\pi}\right]\\
&=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[-\frac{d}{2\sigma^2}+\frac{1}{2\left(\sigma^2\right)^2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)\right]Q_t^{\pi}\right]\\
\Leftrightarrow\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{d}{2\sigma^2}Q_t^{\pi}\right]&=\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{2\left(\sigma^2\right)^2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)\right]Q_t^{\pi}\right]\\
\Leftrightarrow\frac{d}{2\sigma^2}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]&=\frac{1}{2\left(\sigma^2\right)^2}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)\right]Q_t^{\pi}\right]\\
\Leftrightarrow\sigma^2\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]&=\frac{1}{d}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)\right]Q_t^{\pi}\right]\\
\Leftrightarrow\sigma^2\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]&=\frac{1}{d}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)-2\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}+\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]Q_t^{\pi}\right].
\end{aligned}
\tag{3.22}
$$

The update equation for this parameter is given by

$$
\begin{aligned}
\sigma^2=&\frac{1}{d}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)-2\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{W}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]\right.\right.\\
&\left.\left.+\operatorname{trace}\left(\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\right]\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)\right)Q_t^{\pi}\right]\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]\right)^{-1}.
\end{aligned}
\tag{3.23}
$$

3.4.3 Complete Algorithm

With the results given above we can find the '**P**olicy **Se**arch with **P**robabilistic **P**rinciple **C**omponent **Explo**ration in the Action Space' (PePP$_c$Er) algorithm which can be found in Alg. 1. The initial parameters ${\sigma^2}_0$,$\mathbf{W}_0$ and $\mathbf{M}_0$ can either be chosen randomly or with a pre-analysis with PCA or PPCA. Furthermore, as an additional parameter, the algorithm needs the number of latent dimension n as input. This number depends on the given (robot) system and the task. A pre-analysis on a few samples may help to uncover this property of the hidden subspace. As already mentioned, a feature vector $\boldsymbol{\phi}\left(\mathbf{s}_t, t\right) \in \mathbb{R}^p$ is needed. An example for this vector is given later in the experiment section. The result of this algorithm is the matrix $\mathbf{M}$ which contains weights for the basis functions.

Input: Initialized parameters ${\sigma^2}_0, \mathbf{W}_0$ and $\mathbf{M}_0$ and the number n of the low intrinsic dimension. The function $\boldsymbol{\phi}\left(\mathbf{s}_t, t\right)$ serves the basic functions as features.

repeat

Sampling:

for h=*1:H* **do** # Sample the H rollouts

for t=*1:T* **do**

$\mathbf{a}_t^h = \mathbf{W}_i \mathbf{Z}^{\mathrm{T}} \boldsymbol{\phi} + \mathbf{M}_i \boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}$ with $\mathbf{Z} \sim \mathcal{N}_{p,n}(\mathbf{0}, \mathbf{I})$ and $\mathbf{E} \sim \mathcal{N}_{d,p}\left(\mathbf{0}, {\sigma^2}_i \mathbf{I}\right)$

Execute action $\mathbf{a}_t^h$

Observe and store reward $r_t\left(\mathbf{s}_t, \mathbf{a}_t^h\right)$

Calculate weights:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}, t) = \mathbb{E}\left[\sum_{\tilde{t}=t}^{T} r_{\tilde{t}}\left(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}\right) | \mathbf{s}_t = s, \mathbf{a}_t = \mathbf{a}\right]$$

Expectation:

$$\mathbf{C} = \left({\sigma^2}_i \mathbf{I} + \mathbf{W}_i^{\mathrm{T}} \mathbf{W}_i\right)^{-1}$$

foreach $\mathbf{a}_t^h$ **do**

$$\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right] = \mathbf{C}\mathbf{W}_i^{\mathrm{T}}\left(\mathbf{a}_t^h - \mathbf{M}_i\boldsymbol{\phi}\right)$$

$$\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right] = \mathbf{C}{\sigma^2}_i \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right) + \mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right] \cdot \mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}$$

Maximization:

$$\mathbf{M}_{i+1} = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1} \mathbf{a}_t^h \boldsymbol{\phi}^{\mathrm{T}} Q_t^{\pi}\right]\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1} \boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}} Q_t^{\pi}\right]\right)^{-1}$$

$$\mathbf{W}_{i+1} = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1} \left(\mathbf{a}_t^h - \mathbf{M}_{i+1}\boldsymbol{\phi}\right)\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}} Q_t^{\pi}\right]$$
$$\cdot \left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1} \mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right] Q_t^{\pi}\right]\right)^{-1}$$

$${\sigma^2}_{i+1} = d^{-1}\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \mathrm{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\left(\left(\mathbf{a}_t^h - \mathbf{M}_{i+1}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}_t^h - \mathbf{M}_{i+1}\boldsymbol{\phi}\right) - 2\left(\mathbf{a}_t^h - \mathbf{M}_{i+1}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{W}_{i+1}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]\right.\right.$$
$$\left.\left. + \mathrm{trace}\left(\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\right]\mathbf{W}_{i+1}^{\mathrm{T}}\mathbf{W}_{i+1}\right)\right) Q_t^{\pi}\right]\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} Q_t^{\pi}\right]\right)^{-1}$$

until $\mathbf{M}_i \approx \mathbf{M}_{i+1}$

Output: Policy $\mathbf{M}$ for the feature vector $\boldsymbol{\phi}$

Algorithm 1: Policy Search with Probabilistic Principle Component Exploration in the Action Space (PePP$_c$Er)

3.5 Diagonal Matrix Extension

Revisiting the stochastic model of PePP$_c$Er given by

$$\mathbf{a} = \left(\mathbf{W}\mathbf{Z}^{\mathrm{T}} + \mathbf{M} + \mathbf{E}\right)\boldsymbol{\phi} = \mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}, \tag{3.24}$$

we can find a slightly different model if we choose the covariance matrix of $\mathbf{E}$ as a diagonal matrix with several parameters for each dimension in the action space. Thus we have $\mathbf{E} \sim \mathcal{N}_{d,p}(\mathbf{0}, \mathbf{D} \otimes \mathbf{I})$ with $\mathbf{D} \in \mathbb{R}^{d,d}$ as such a diagonal covariance matrix with respect to the action space and the other distributions as before. For Eq. (3.7) we can now find that

$$\begin{aligned} \operatorname{cov}(\mathbf{a}) &= \mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\right] + \mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{E}^{\mathrm{T}}\right] \\ &= \mathbf{W}\ \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\ \mathbf{W}^{\mathrm{T}} + \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{D} \\ &= \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \mathbf{D}\right) \end{aligned} \tag{3.25}$$

because of Thm. 2 and $\mathbf{E}^{\mathrm{T}} \sim \mathcal{N}_{p,d}(\mathbf{0}, \mathbf{I} \otimes \mathbf{D})$. Therefore the property

$$\mathrm{p}(\mathbf{a}) = \mathcal{N}\left(\mathbf{M}\boldsymbol{\phi}, \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \mathbf{D}\right)\right) \tag{3.26}$$

exists for the distribution of a $\mathbf{a}$ and we can find for $\ln \mathrm{p}(\mathbf{a}, \mathbf{Z})$ that

$$\begin{aligned} \frac{\partial \ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)}{\partial \theta_{new}} = \frac{\partial}{\partial \theta_{new}}\Bigg(&-\frac{d}{2}\ln\left(2\pi \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right) - \frac{1}{2}\ln\det(\mathbf{D}) \\ &- \frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\Bigg). \end{aligned} \tag{3.27}$$

While the maximization term for $\mathbf{M}$ remains the same it is important to ensure that the maximization for $\mathbf{W}$ does not depend on $\mathbf{D}$ and is still solvable. We can find that this is still true with

$$\begin{aligned} \frac{\partial \ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)}{\partial \mathbf{W}} &\overset{3.27}{=\!=} \frac{\partial}{\partial \mathbf{W}}\Bigg(-\frac{d}{2}\ln\left(2\pi \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right) - \frac{1}{2}\ln\det(\mathbf{D}) + \\ &\qquad - \frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\Bigg) \\ &= \frac{\partial}{\partial \mathbf{W}}\left(-\frac{1}{2\sigma^2 \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\right) \\ &= \frac{\partial}{\partial \mathbf{W}}\left(-\frac{1}{2\sigma^2 \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(-2\mathbf{a}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} + \boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} + 2\boldsymbol{\phi}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)\right) \\ &\overset{A.1}{=\!=} -\frac{1}{2\sigma^2 \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(-2\mathbf{D}^{-1}\mathbf{a}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}} + \frac{\partial \boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}}{\partial \mathbf{W}} + 2\mathbf{D}^{-1}\mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right) \\ &\overset{A.2}{=\!=} -\frac{1}{\sigma^2 \operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(-\mathbf{D}^{-1}\mathbf{a}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}} + \mathbf{D}^{-1}\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}} + \mathbf{D}^{-1}\mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right). \end{aligned} \tag{3.28}$$

because the matrix derivations ensure that the matrix $\mathbf{D}$ can be found on the left side of each term and so Eq. 3.19 still holds.

Now we have to determine the maximization term of $\mathbf{D}$. First we can find for the differentiation of $\ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)$ w.r.t. $\mathbf{D}$ that

$$\begin{aligned} \frac{\partial \ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)}{\partial \mathbf{D}} &\overset{3.27}{=\!=} \frac{\partial}{\partial \mathbf{D}}\left(-\frac{1}{2}\ln\det(\mathbf{D}) - \frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\right) \\ &\overset{A.3}{=\!=} -\frac{1}{2}\mathbf{D}^{-1} - \frac{\partial}{\partial \mathbf{D}}\frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right) \\ &\overset{A.2}{=\!=} -\frac{1}{2}\mathbf{D}^{-1} + \frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbf{D}^{-1}\operatorname{diag}\left\{\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\right\}\mathbf{D}^{-1} \end{aligned} \tag{3.29}$$

holds and if we plugging this result into the lower bound we get that

$$\begin{aligned} 0 &= \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{\partial}{\partial \mathbf{D}}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)\right]Q_t^{\pi}\right] \\ \Longleftrightarrow 0 &= \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} -\frac{1}{2}\mathbf{D}^{-1}Q_t^{\pi} + \sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbf{D}^{-1}\operatorname{diag}\left\{\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\right\}\mathbf{D}^{-1}\right]Q_t^{\pi}\right] \\ \Longleftrightarrow \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\frac{1}{2}\mathbf{D}^{-1}Q_t^{\pi}\right] &= \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{2\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbf{D}^{-1}\operatorname{diag}\left\{\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\right\}\mathbf{D}^{-1}\right]Q_t^{\pi}\right]. \end{aligned} \tag{3.30}$$

If we now multiply $\mathbf{D}$ from the left and the right side we find the maximization with

$$\mathbf{D} = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\operatorname{diag}\left\{\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)\left(\mathbf{a}-\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\right\}\right]Q_t^{\pi}\right]\cdot\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]\right)^{-1}$$

$$\begin{aligned}\mathbf{D} = \mathbb{E}_{\boldsymbol{\tau}}\Bigg[\sum_{t=1}^{T}\frac{1}{\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\operatorname{diag}\Big\{&\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}-\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}\mathbf{W}-\mathbf{W}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]\left(\mathbf{a}-\mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}+\\ &\mathbf{W}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]\mathbf{W}^{\mathrm{T}}\Big\}Q_t^{\pi}\Bigg]\cdot\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]\right)^{-1}.\end{aligned} \tag{3.31}$$

The resulting algorithm can be found in Alg. 2, where we can find the new update for the diagonal covariance matrix and slightly different terms for the expectation step.

Input: Initialized parameters $\mathbf{D}_0, \mathbf{W}_0$ and $\mathbf{M}_0$ and the number n of the low intrinsic dimension. The function $\boldsymbol{\phi}\left(\mathbf{s}_t, t\right)$ serves the basic functions as features.

repeat

Sampling:

for h=*1:H* **do** # Sample the H rollouts

for t=*1:T* **do**

$\mathbf{a}_t^h = \mathbf{W}_i\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} + \mathbf{M}_i\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}$ with $\mathbf{Z} \sim \mathcal{N}_{p,n}\left(\mathbf{0}, \mathbf{I}\right)$ and $\mathbf{E} \sim \mathcal{N}_{d,p}\left(\mathbf{0}, \mathbf{D}_i \otimes \mathbf{I}\right)$

Execute action $\mathbf{a}_t^h$

Observe and store reward $r_t\left(\mathbf{s}_t, \mathbf{a}_t^h\right)$

Calculate weights:

$$Q^{\pi}\left(\mathbf{s}, \mathbf{a}, t\right) = \mathbb{E}\left[\sum_{\tilde{t}=t}^{T} r_{\tilde{t}}\left(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}\right) | \mathbf{s}_t = s, \mathbf{a}_t = \mathbf{a}\right]$$

Expectation:

$$\mathbf{C} = \left(\mathbf{I} + \mathbf{W}_i^{\mathrm{T}}\mathbf{D}_i^{-1}\mathbf{W}_i\right)^{-1}$$

foreach $\mathbf{a}_t^h$ **do**

$$\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right] = \mathbf{C}\mathbf{W}_i^{\mathrm{T}}\mathbf{D}_i^{-1}\left(\mathbf{a}_t^h - \mathbf{M}_i\boldsymbol{\phi}\right)$$

$$\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right] = \mathbf{C}\cdot\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right) + \mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]\cdot\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}$$

Maximization:

$$\mathbf{M}_{i+1} = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\mathbf{a}_t^h\boldsymbol{\phi}^{\mathrm{T}}Q_t^{\pi}\right]\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}Q_t^{\pi}\right]\right)^{-1}$$

$$\begin{aligned}\mathbf{W}_{i+1} = \mathbb{E}_{\boldsymbol{\tau}}\Bigg[&\sum_{t=1}^{T}\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\left(\mathbf{a}_t^h - \mathbf{M}_{i+1}\boldsymbol{\phi}\right)\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}Q_t^{\pi}\Bigg]\\ &\cdot\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}\right]\right)^{-1}\end{aligned}$$

$$\begin{aligned}\mathbf{D}_{i+1} = \mathbb{E}_{\boldsymbol{\tau}}\Bigg[&\sum_{t=1}^{T}\operatorname{trace}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\operatorname{diag}\Big\{\left(\mathbf{a}-\mathbf{M}_{i+1}\boldsymbol{\phi}\right)\left(\mathbf{a}-\mathbf{M}_{i+1}\boldsymbol{\phi}\right)^{\mathrm{T}}-2\mathbf{W}_{i+1}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]\left(\mathbf{a}-\mathbf{M}_{i+1}\boldsymbol{\phi}\right)^{\mathrm{T}}+\\ &\mathbf{W}_{i+1}\mathbb{E}_{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]\mathbf{W}_{i+1}^{\mathrm{T}}\Big\}Q_t^{\pi}\Bigg]\cdot\left(\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]\right)^{-1}\end{aligned}$$

until $\mathbf{M}_i \approx \mathbf{M}_{i+1}$

Output: Policy $\mathbf{M}$ for the feature vector $\boldsymbol{\phi}$

Algorithm 2: Policy Search with Probabilistic Principle Component Exploration in the Action Space (PePP$_{\mathrm{C}}$Er) with Diagonal Matrix Extension.

4 Experiments

We conducted a set of experiments in order to compare PePP$_c$Er to other existing algorithms for policy search. In this chapter we will present the setups under which the experiments were carried out. Both results on simulation and with a physical robot will be reported. All experiments were done with the Diagonal Matrix Extension for the PePP$_c$Er Algorithm.

4.1 Learning Inverse Kinematics

For the comparison of PePP$_c$Er with other algorithms we set up an experiment with a simulated robot arm. The arm has 26 hinge-joints and 27 segments, an example of such an arm for 8 joints can be found in Fig. 4.1. The goal is to track a trajectory using the end-effector of the robot arm, which we choose to be an ellipse. This ellipse, which is visible as a red circle in Fig. 4.1, follows the equation

$$f(t) = \begin{pmatrix} \cos(t \cdot 0.15) + DOF - 2.5 \\ \sin(t \cdot 0.15) \cdot 1.5 + 2 \\ 0 \end{pmatrix}, \tag{4.1}$$

where $1 \leq t \leq 50$ is the current time step and DOF the degrees-of-freedom of the robot arm. This forms an ellipse at the upper end of the robot arm. In the start configuration all joints were set to zero degree and the position of the end-effector was $(DOF, 0, 0)^{\mathrm{T}}$ since each segment is one unit long.

For each of the 50 time steps, we have to learn parameters to bring the end-effector to touch the current point of this trajectory. In our case, the parameters are weights for the summation of Gaussian features, which depends on time. The activations of these Gaussians can be found in Fig. 4.2, where we can see that each Gaussian has a variance of three and the distance between two means is three time steps. To learn the inverse kinematics, we set the reward function to

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = \exp(-\mathrm{D}), \tag{4.2}$$

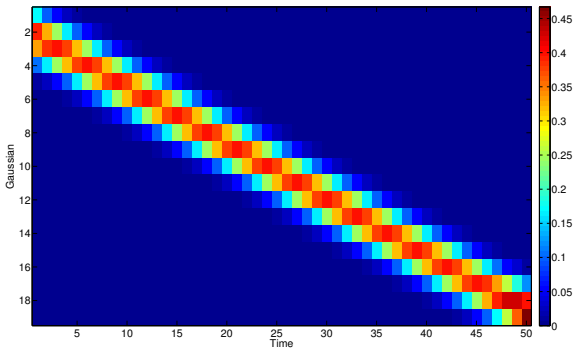


Figure 4.2.: The rows represent the Gaussians and the columns the specific time step. The values of the Gaussians are normalized for each time step.

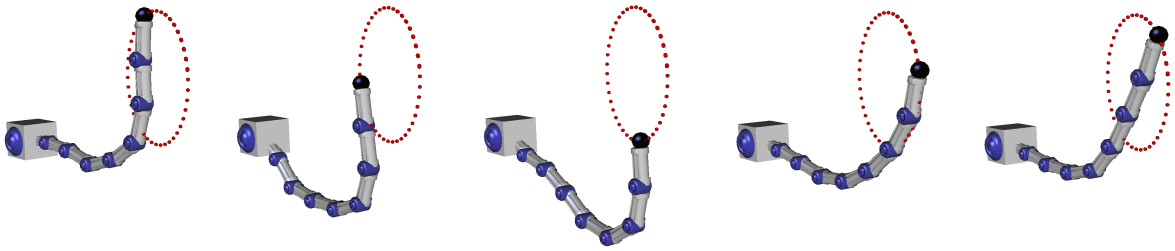


Figure 4.1.: A robot arm with 8 DOF in a 2D Tracking Task. These movements were learned with PePP$_c$Er within 1000 iterations and have an error near zero. The red circle is the desired trajectory of the end-effector.

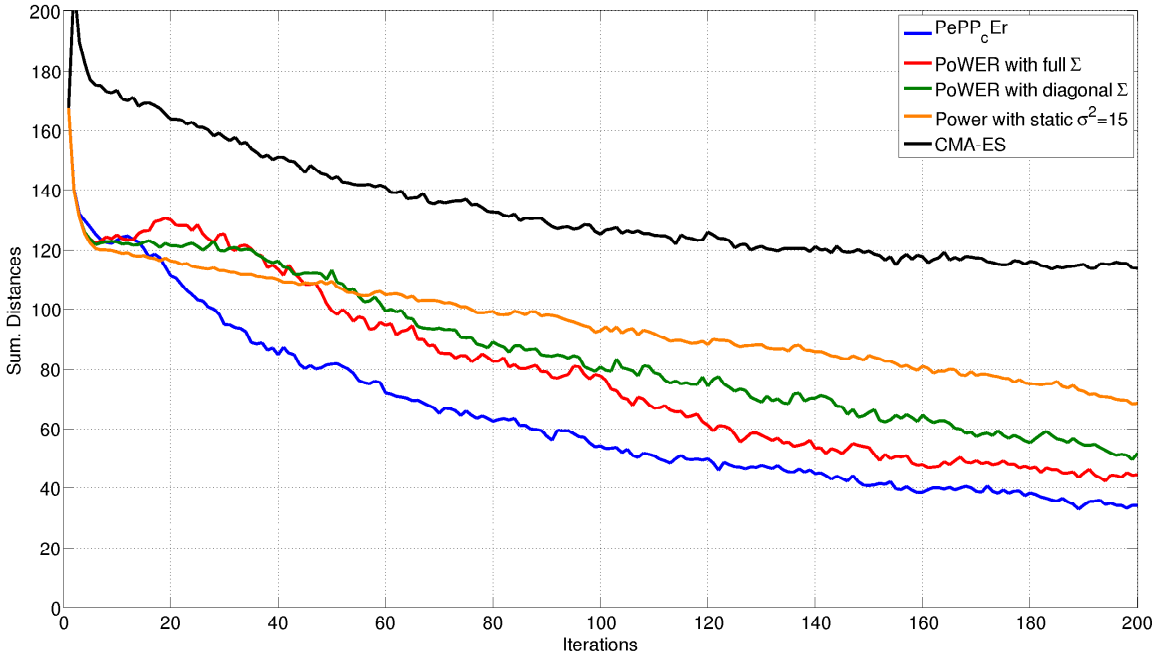


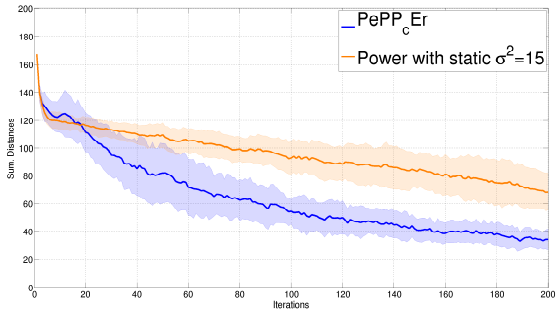
Figure 4.3.: Comparison between PePP$_c$Er , PoWER and CMA-ES on the inverse kinematic task with a 26-linked robot arm. In each iteration we generated 50 parameterized joint configurations and used the 25 best samples. Every Algorithm was performed with 30 trials and plotted with his mean.

where D is the distance between the end-effector and the target at time step t and the executed action $\mathbf{a}_t$. The state of the system is not relevant in this function.

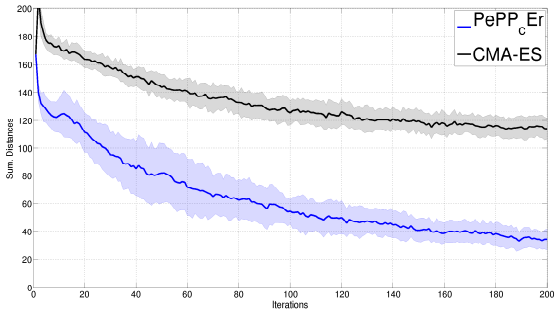
While our task space is only two dimensional, the robot arm meanwhile has 26 DOF which implies that we have a high redundancy in our system. PePP$_c$Er tries to uncover the latent structure of this 26-dimensional space by determining the mean and the principal axes of this subspace, where each axis represents the co-articulation of different links. Algorithms like PoWER are not able to identify these dependencies between the dimensions of the actions. The Covariance Matrix Adaption Evolution Strategy (CMA-ES) Algorithm can find a full covariance matrix for all parameters, but it will be slower than an Algorithm that has a strong exploration in the principal components concerning tasks where only a few dimensions are relevant. We ran the explained setup with different policy search algorithms, the results can be found in Fig. 4.3. We can see that after 19 iterations PePP$_c$Er significantly outperforms the different PoWER implementations and the CMA-ES Algorithm. For a better comparison between PePP$_c$Er and the different algorithms we show in Fig. 4.4 only PePP$_c$Er and one other algorithm per graph. As mentioned above we used 19 time-dependent Gaussians over 50 time steps (see Fig. 4.2 for an intuition) so we have to estimate 19 weights for one action dimension to approximate the desired trajectory. For 26 dimensions this results in a total number of 494 parameters which have to be estimated for this task. The number of assumed latent dimensions for the PePP$_c$Er Algorithm was set to $n = 11$ in this evaluation. Furthermore we used the Diagonal Matrix Extension. The algorithms runs 200 iterations and in each iteration 50 samples were created. From this set the 25 samples with highest rewards were chosen for updating the specific parameters of the algorithms. This was performed to have a better distinction between good and bad samples.

4.2 Learning to Stand on One Leg

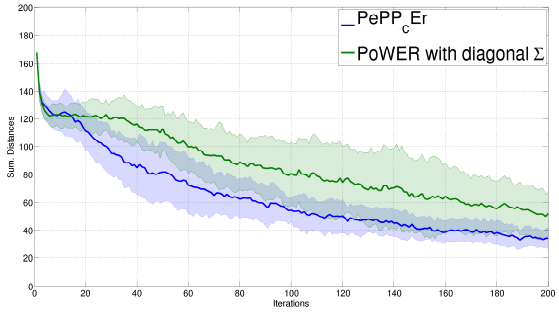
To show that the PePP$_c$Er Algorithm can be used in real-world tasks, we performed a learning task on the NAO robot. Our reward function measured the height of the left leg and the head of the robot, in order to learn to stand on one leg. This task requires the co-articulation of different body parts and is often used in biomechanical studies on synergies and low-dimensional control in humans, such as in [22]. Learning was performed in a physics-based simulator instead of the real robot, in order to prevent damages of the robot and to automate the process. Because normally sampling on real robots is



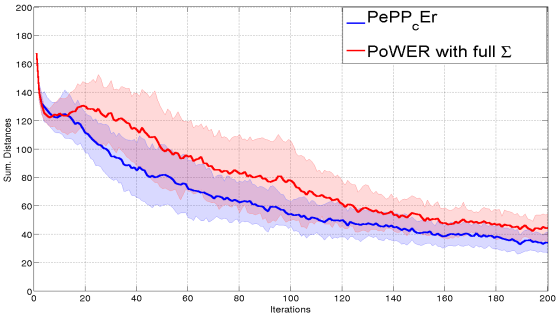
(a) Comparison between PePP$_c$Er and PoWER with a static diagonal covariance matrix $\mathbf{I} \cdot 15$.



(b) Comparison between PePP$_c$Er and CMA-ES, where the parameters of CMA-ES are automatically chosen.



(c) Comparison between PePP$_c$Er and PoWER with a diagonal covariance matrix, which is adapted in each iteration.



(d) Comparison between PePP$_c$Er and the PoWER with an adaption of a full covariance matrix in each iteration.

Figure 4.4.: Different comparisons between PePP$_c$Er and one algorithm from Fig. 4.3, each algorithm result plotted with its mean and variance.

cost and time intensive, the goal was to use only a few samples in each iteration to learn a stable and good policy. Thus, it is still possible to do learning directly on the robot, if no simulation is available.

In each iteration we executed 20 movements and used the 5 best executions for learning. Each movement consists of 5 time steps and was created by two time-dependent Gaussian features, so we have 52 parameters to estimate due to the 26 DOF of the NAO robot. As can be seen in Fig. 4.6 only a small number of samples is needed to learn a stable and smooth solution. Policy 1 was learned within 5 iterations, so 100 executions on the simulated robot in total.

In Policy 2 we used 600 samples in 30 iterations, which results in a movement where the robot can lift the leg even higher. Because of the stochastic nature of the PePP$_c$Er Algorithm, we can find very broad variations of leg lifting movements, two extreme examples can be found in Fig. 4.5. The shown movements seems extreme, but they are still stable and smooth, so that we can execute them on the real robot. Fig. 4.7 shows the evolution of a policy during the policy search process. Initially the leg is not lifted. However, after several iterations the algorithm identifies a stable lifting-up movement. If we compare the results of the first iterations for different policies in Fig. 4.8 we can see that the found solutions have a strong dependency on the result of the first iteration. This behaviour can be traced back to the fact that the EM Algorithm is a local search algorithm. The results of Policy 4 are particularly insightful since in the solution, which is found in the first iteration of this trial, lifts the right leg instead of the left one. Even if this was the only

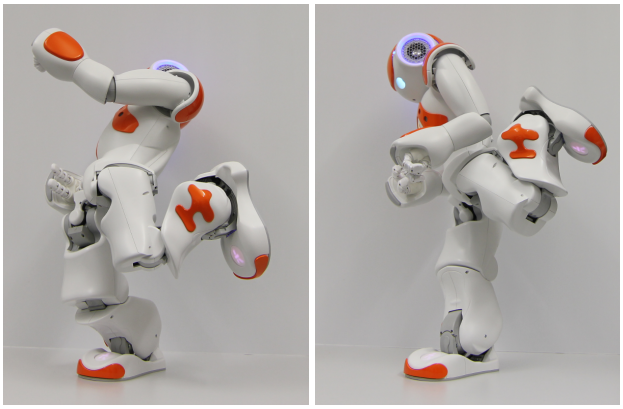


Figure 4.5.: The final poses of two extreme solutions learned by the PePP$_c$Er Algorithm, which are still stable.

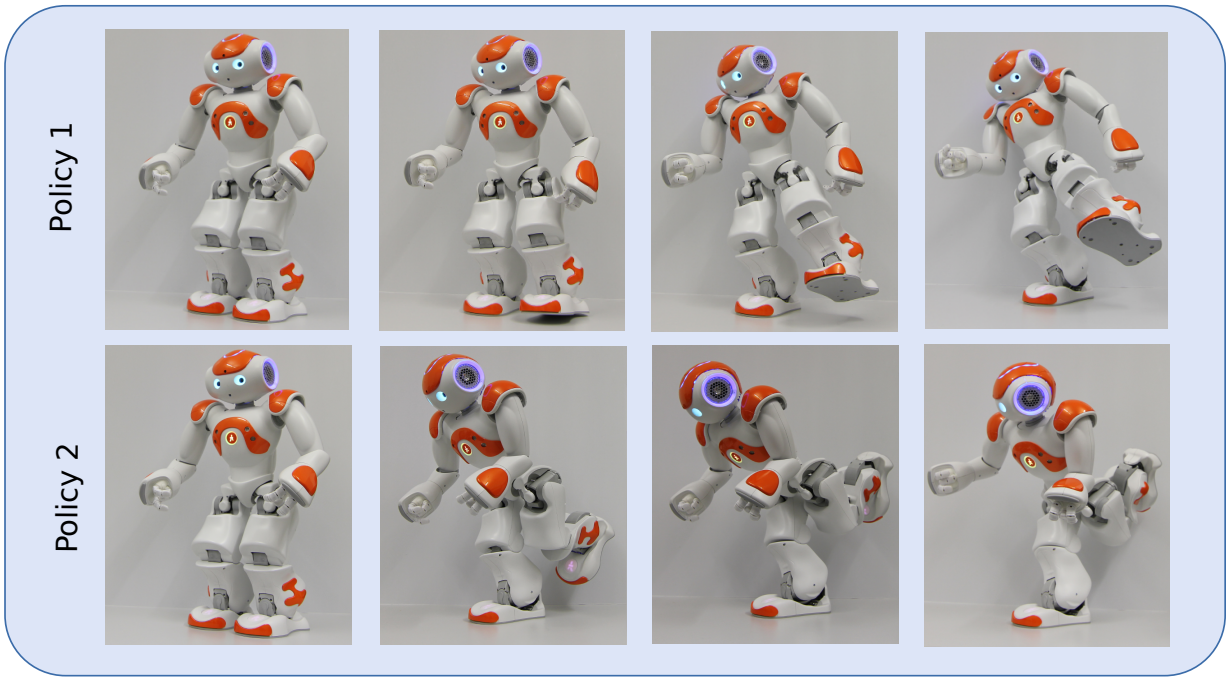
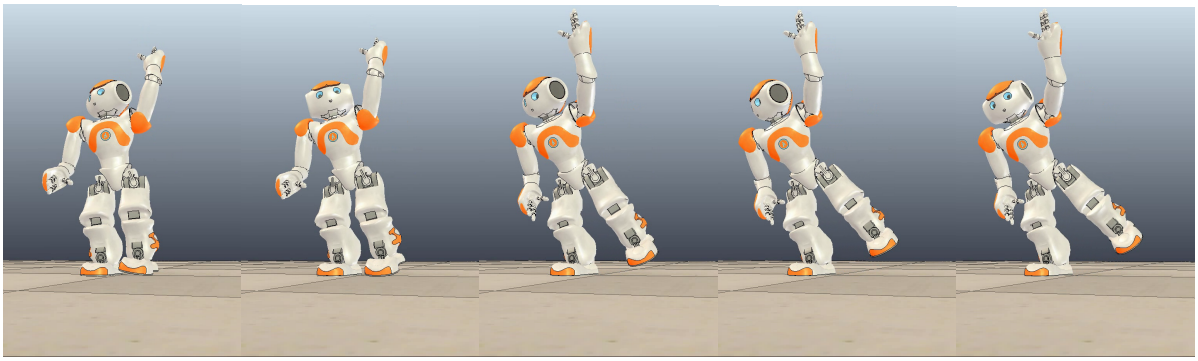


Figure 4.6.: Two different policies for standing on one leg learned using latent space policy search. The PePP$_c$Er algorithm needed only 100 executions for the first policy.



(a) 1 iteration **(b)** 2 iterations **(c)** 3 iterations **(d)** 4 iterations **(e)** 5 iterations

Figure 4.7.: The results of the first five iterations of Policy 5 learned by the PePP$_c$Er Algorithm

stable position found in the first iteration, the PePP$_c$Er Algorithm is able to find a smooth and stable movement for lifting the left leg within five iterations, therefore performing 100 executions.

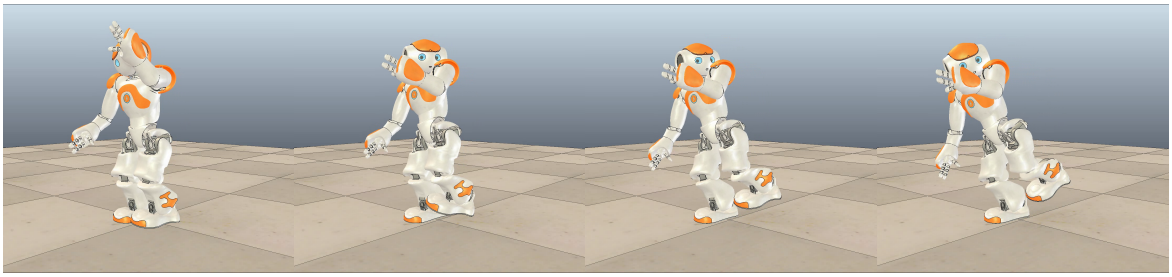
4.3 Experimental Intuition

To give an intuition for the PePP$_c$Er Algorithm, we performed and visualized a trial on an optimization task. The test function was the sixth Bukin function [23], given by

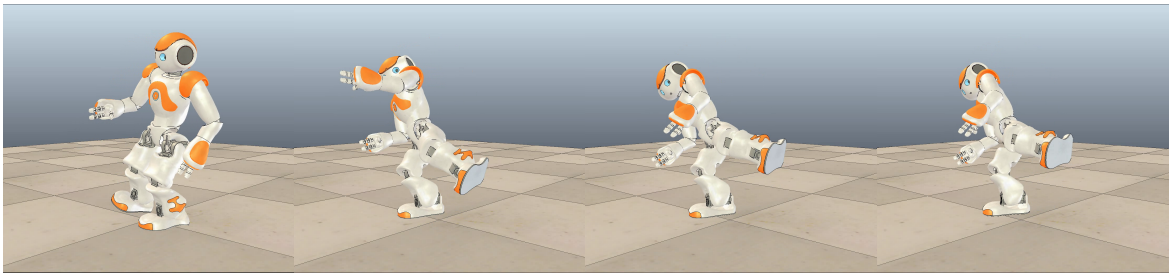
$$f(x,y) = 100\sqrt{|y - 0.01x^2|} + 0.01|x + 10|, \ \min f(x,y) = f(-10,1) = 0 \tag{4.3}$$

as a minimization problem. This function has one global minima, but many local minima that lie all in a ridge, which can be seen in Fig. 4.9, due to this the function models global optimization problem.

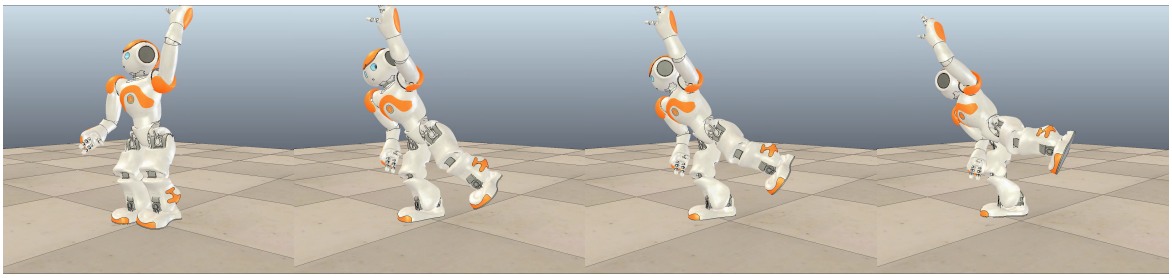
In the given plots in Fig. 4.10 we can see the solution and principal component for each iteration of the PePP$_c$Er Algorithm. For the sake of visibility, we show the principal component ten times larger. We can see that after a few iterations the mean point reaches the ridge and the principal component lies in the direction of the ridge as expected, because the five best samples lie along the ridge. Especially in the last 10 iterations, before the global minima is reached, the principal component leads to a sampling along the ridge to reach the global minima.



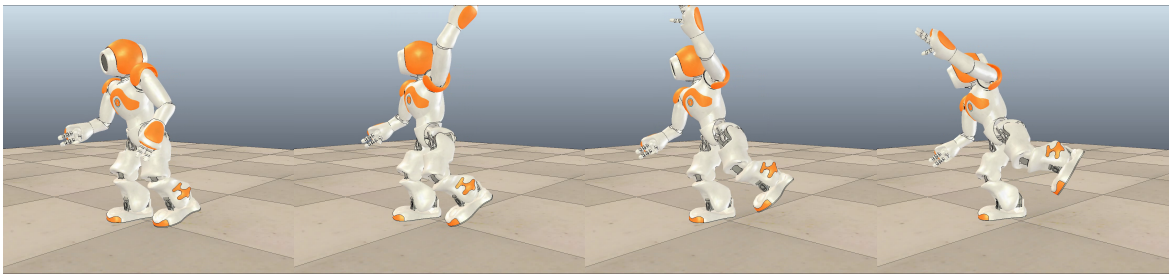
(a) 1st iteration of policy 3 **(b)** 5th iter. of policy 3 **(c)** 10th iter. of policy 3 **(d)** 15th iter. of policy 3



(e) 1st iteration of policy 4 **(f)** 5th iter. of policy 4 **(g)** 10th iter. of policy 4 **(h)** 15th iter. of policy 4



(i) 1st iteration of policy 5 **(j)** 5th iter. of policy 5 **(k)** 10th iter. of policy 5 **(l)** 15th iter. of policy 5



(m) 1st iteration of policy 6 **(n)** 5th iter. of policy 6 **(o)** 10th iter. of policy 6 **(p)** 15th iter. of policy 6

Figure 4.8.: The resulting poses after 1, 5, 10 and 15 iterations with the PePPEr Algorithm. Each row is one independent trial.

However, we can see that in the first two iterations it may be beneficial to use, in this case, both principal components for the first few iterations. When the ridge is reached, it is sufficient to use only one principal component for the exploration because of the almost linear structure of the ridge. While the adding and removing of columns to the transformation matrix $\mathbf{W}$ is already possible between two iterations, we need here a reasonable decision criterion for the number of dimensions. This possibility of using different numbers of principal components for various stages during an optimization process may be a crucial feature for a faster convergence and needs further investigations in the future.

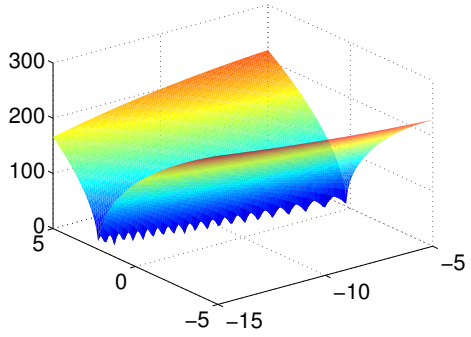
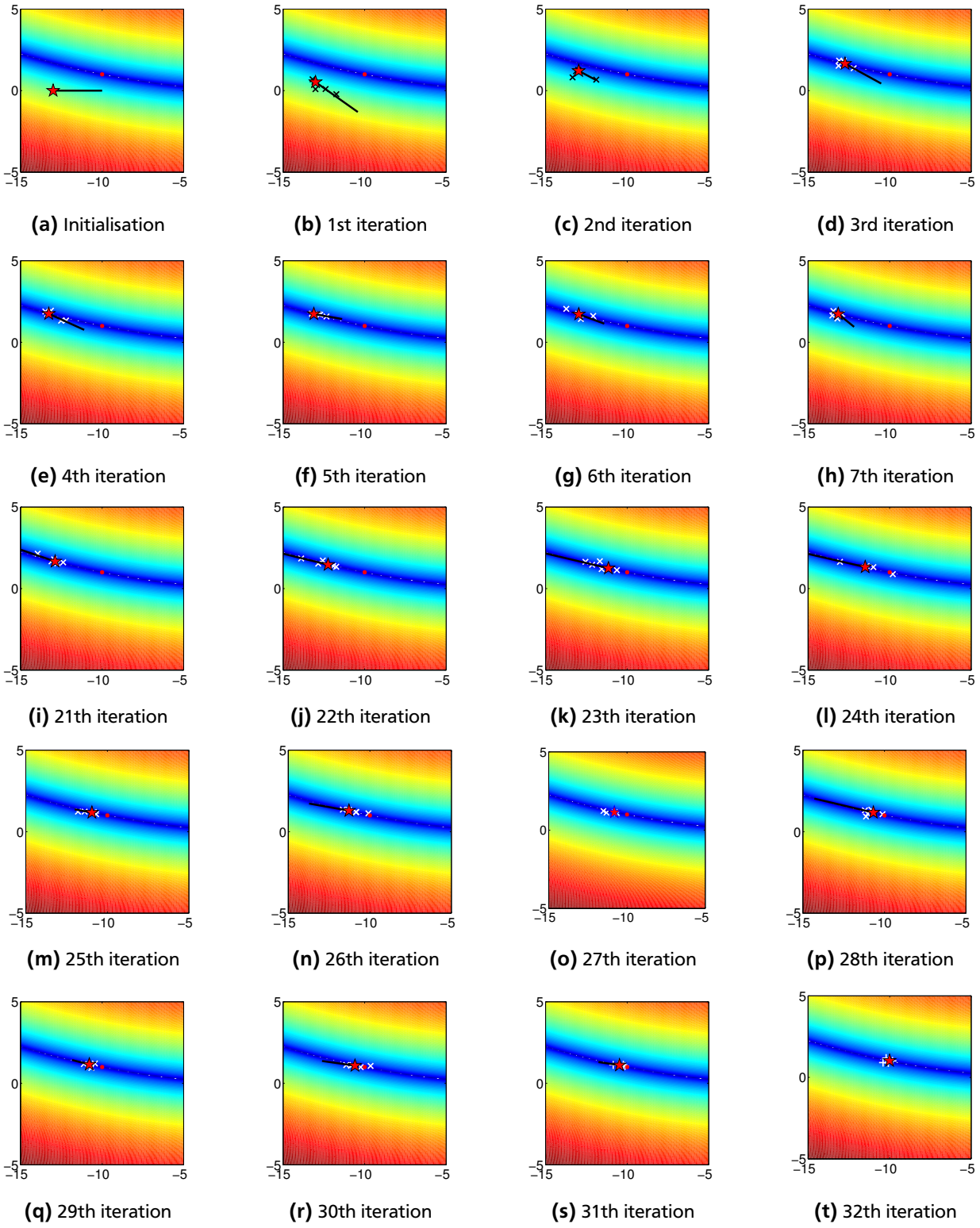


Figure 4.9.: The sixth Bukin function.



(a) Initialisation **(b)** 1st iteration **(c)** 2nd iteration **(d)** 3rd iteration

(e) 4th iteration **(f)** 5th iteration **(g)** 6th iteration **(h)** 7th iteration

(i) 21th iteration **(j)** 22th iteration **(k)** 23th iteration **(l)** 24th iteration

(m) 25th iteration **(n)** 26th iteration **(o)** 27th iteration **(p)** 28th iteration

(q) 29th iteration **(r)** 30th iteration **(s)** 31th iteration **(t)** 32th iteration

Figure 4.10.: The PePPęEr Algorithm performed on the sixth Bukin function. The red star is the current mean in the given iteration and the black line the principal component. Both are calculated from five samples, given by the white crosses. The global minima of the Bukin function is given by the red dot.

5 Conclusion and Future Work

In this thesis we presented a novel policy search algorithm for exploiting latent spaces in high dimensional action spaces. We combined the techniques of policy search and dimensionality reduction in an algorithm instead of using the dimensionality reduction as a pre-processing step.

The PePP$_C$Er Algorithm was introduced with an isotropic undirected exploration and a diagonal matrix extension, where we can determine a parameter for this exploration for each dimension. We showed with a evaluation on a simulated and a real-world task that the property of the PePP$_C$Er Algrothm, to uncover the hidden structure of a latent subspace, is able to speed up the convergence to an optimal solution in the given robot tasks. This indicates that our approach is suitable to do policy search for robots where we have a (high) redundancy in the actions and can only use a small set of samples due to costs and time.

However, so far the PePP$_C$Er algorithm is only able to uncover the latent space in the action space. Although the algorithm is still better as the PoWER Algorithm with a full covariance matrix per action dimension, we want to investigate the possibility to uncover latent spaces in the feature space too. This may not be reasonable for time-dependent Gaussian features, but if we have to handle Dynamic Motor Primitives (i.e. see [4]) or feedback features it is possible that with a growing number of features like this we can find such a low dimensional space. This leads to the possibility to use a larger number of features, which can be a crucial feature for an online learning or improving task.

Furthermore the problem to guess the number of dimensions of the latent space right for a maximal efficiency of the PePP$_C$Er Algorithm still exists. While it is possible to do this with a pre-analysis, we want do avoid this and instead guess the number of needed dimensions while learning, because the algorithm allows to change the dimensionality of the latent space while running.

A problem of the PoWER framework is to choose a right parameter for the transformed reward functions if we use the exponential function. So we have to adjust this reward function $\exp(-\lambda \cdot r_t)$ with a parameter $\lambda > 0$ to get a better distinction between good and bad samples. In contrast, the REPS [24] framework offers the possibility to adjust this parameter dynamically in each iteration. Because the PePP$_C$Er Algorithm can be used in any policy search framework where we have a stochastic policy $\pi(\mathbf{a}_t, \mathbf{s}_t)$, we want to implement PePP$_C$Er in the REPS Algorithm in the future.

Bibliography

[1] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Proceedings of the International Symposium on Experimental Robotics*, 2004, pp. 363–372.

[2] K. Muelling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, no. 3, pp. 263–279, 2013.

[3] J. Zico Kolter and A. Y. Ng, "The stanford littledog: A learning and rapid replanning approach to quadruped locomotion," *Int. J. Rob. Res.*, vol. 30, no. 2, pp. 150–174, Feb. 2011.

[4] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research*. Springer, 2005, pp. 561–572.

[5] R. E. Bellman, *Adaptive Control Processes: A guided tour*. Princeton University Press, 1961, ch. 5.16.

[6] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[7] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[8] M. P. Deisenroth, R. Calandra, A. Seyfarth, and J. Peters, "Toward fast policy search for learning legged locomotion," in *IEEE/RSJ International Conference on Intelligent Systems and Robots*, 2012.

[9] J. Z. Kolter and A. Y. Ng, "Learning omnidirectional path following using dimensionality reduction," in *in Proceedings of Robotics: Science and Systems*, 2007.

[10] S. Bitzer, M. Howard, and S. Vijayakumar, "Using dimensionality reduction to exploit constraints in reinforcement learning," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 3219–3225.

[11] C. L. Nehaniv and K. Dautenhahn, "Imitation in animals and artifacts," K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA, USA: MIT Press, 2002, ch. The Correspondence Problem, pp. 41–61.

[12] K. S. Luck, G. Neumann, E. Berger, J. Peters, and H. B. Amor, "Latent space policy search for robotics," 2014, *in review*.

[13] A. K. Gupta and D. K. Nagar, *Matrix variate distributions*. CRC Press, 2000, vol. 104.

[14] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.

[15] A. Hannachi, I. T. Jolliffe, and D. B. Stephenson, "Empirical orthogonal functions and related techniques in atmospheric science: A review," *International Journal of Climatology*, vol. 27, no. 9, pp. 1119–1152, 2007. [Online]. Available: http://dx.doi.org/10.1002/joc.1499

[16] M. Khezri and M. Jahed, "An exploratory study to design a novel hand movement identification system," *Computers in Biology and Medicine*, vol. 39, no. 5, pp. 433 – 442, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010482509000420

[17] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 696–710, Jul 1997.

[18] J. Lin, Y. Wu, and T. Huang, "Articulate hand motion capturing based on a monte carlo nelder-mead simplex tracker," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 4, 2004, pp. 975–978 Vol.4.

[19] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[20] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[21] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[22] G. Torres-Oviedo and L. H. Ting, "Subject-specific muscle synergies in human balance control are consistent across different biomechanical contexts," *Journal of neurophysiology*, vol. 103, no. 6, pp. 3084–3098, 2010.

[23] A. Bukin, *New Minimization Strategy For Non-Smooth Functions*. Budker Institute of Nuclear Physics preprint BUDKER-INP-1997-79, Novosibirsk, 1997.

[24] C. Daniel, G. Neumann, and J. Peters, "Hierarchical relative entropy policy search," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, 2012.

A Appendix

A.1 Derivatives of Scalar Forms

The following derivatives with respect to matrices are given without any proof.

$$\frac{\partial \mathbf{a}^{\mathrm{T}}\mathbf{X}\mathbf{b}}{\partial \mathbf{X}} = \mathbf{a}\mathbf{b}^{\mathrm{T}} \tag{A.1}$$

$$\frac{\partial \mathbf{b}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{D}\mathbf{X}\mathbf{c}}{\partial \mathbf{X}} = \mathbf{D}^{\mathrm{T}}\mathbf{X}\mathbf{b}\mathbf{c}^{\mathrm{T}} + \mathbf{D}\mathbf{X}\mathbf{c}\mathbf{b}^{\mathrm{T}} \tag{A.2}$$

If $\mathbf{X}$ is a diagonal matrix then we have

$$\frac{\partial \ln\det(\mathbf{X})}{\partial \mathbf{X}} = \mathbf{X}^{-1} \tag{A.3}$$

A.2 Marginalization Rule

This result is taken from [14, p. 93]:

If we have $\mathbf{x}$ given as a marginal normal distribution and $\mathbf{y}$ given $\mathbf{x}$ as a conditional normal distribution defined by

$$\mathrm{p}(\mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \tag{A.4}$$

$$\mathrm{p}(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \tag{A.5}$$

we are able to determine the normal distribution $\mathbf{x}$ given $\mathbf{y}$ with

$$\mathrm{p}(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\boldsymbol{\Sigma}\left(\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\right), \boldsymbol{\Sigma}\right) \tag{A.6}$$

where $\boldsymbol{\Sigma}$ is defined by $\boldsymbol{\Sigma} = \left(\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A}\right)^{-1}$.