# Probabilistic Inference for Movement Planning in Humanoids

**Probabilistische Planungsmethoden für humanoide Roboter**
Master-Thesis von Max Mindt
Tag der Einreichung:

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Elmar Rückert

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Intelligent Autonomous Systems

Probabilistic Inference for Movement Planning in Humanoids
Probabilistische Planungsmethoden für humanoide Roboter

Vorgelegte Master-Thesis von Max Mindt

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Elmar Rückert

Tag der Einreichung:

# Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter 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.

Darmstadt, den 4th November 2014

______________________________

(Max Mindt)

# Abstract

For controlling robots with many actuators, most stochastic optimal control algorithms use approximations of the system dynamics and of the cost function (e.g., using linearizations and Taylor expansions). These approximations are typically only locally correct, which might cause instabilities in the greedy policy updates, lead to oscillations or the algorithms diverge. To overcome these drawbacks, we add a regularization term to the cost function that punishes large policy update steps in the trajectory optimization procedure. In the first part of this thesis, we applied this concept to the Approximate Inference Control method (AICO), where the resulting algorithm guarantees convergence for uninformative initial solutions without complex hand-tuning of learning rates. We evaluated our new algorithm on two simulated robotic platforms. A robot arm with five joints was used for reaching multiple targets while keeping the roll angle constant. On the humanoid robot Nao, we showed how complex skills like reaching and balancing can be inferred from desired center of gravity or end effector coordinates.

In these tasks we assumed a known forward dynamic model. Typically, inaccurate model predictions have catastrophic effects on the numerical stability of SOC methods. In particular, if the model predictions are poor, the SOC method should not further explore but collect more data around the current trajectory. Therefore, we investigated in the second part of this thesis how to learn such a forward dynamics model with Gaussian processes in parallel to movement planning. The trade off between exploration and exploitation can be regularized with the model uncertainty, which was introduced as an additional objective in AICO. We evaluated the simultaneous model learning and movement planning approach on a simple pendulum toy task. We achieved safe planning and stable convergence even with inaccurate learned models. The planned trajectories, torques and end effector positions converges to local optimal solutions during the learning process. The model prediction error of the forward dynamics model converges to zero.

# Contents

# 1 Introduction

Planning whole body motor control tasks of humanoid robots, like reaching for objects while walking, avoiding obstacles during motion, or maintaining balancing during movement execution, is necessary for robots that can interact with us humans. Humanoid robotic platforms, like Nao[1] or iCub[2] in Figure 1.1, have many actuators and the sensor readings are noisy. Controlling these robots that have to fulfil these multiple objectives (e.g., balancing and walking) is challenging. It is not obvious how to solve such kind of problems. Therefore, it becomes crucial to develop robust methods, which can handle multiple objective problems.

Most real robotic applications involve interactions with the physical world, where the planned policy of our robot is hard to predict due to noise and model errors. One way to fix this issue is to use a feedback control law that tries to follow the computed trajectory as close as possible. However, a more promising way is to model such uncertainties, i.e. it is not known what will happen in the future when certain actions are applied. To achieve our plan under these uncertainty we have to make an optimal decision for every state. The planning problem becomes a decision-making-problem, which is computed by our policy $\pi$. This policy makes use of the observations at the current state and chooses an (optimal) action. Thus, a plan can be formulated as a sequence of decision problems. Futhermore, for multiple objective problems, the planning problem can be characterized as an optimization problem with multiple criteria of optimality or objectives. The objectives may be specified in the robot's configuration space (e.g., joint angles, joint velocities and base reference frame), in task space (where objectives such as desired end effector coordinates or center of gravity positions are specified), or in combinations of both.

## 1.1 Robust movement planning

In this thesis, we consider control problems in nonlinear systems with multiple objectives in combinations of these spaces. Unfortunately, many planning algorithms tend to diverge with multiple task objectives, high-dimensional state space and in particular under the assumption of a analytical known forward dynamics models. Moreover, they do not provide a feedback controller which becomes necessary for controlling compliant robots. Therefore, it is important to develop robust control methods with guaranteed convergence, which can achieve these goals and computes a feedback controller. In this thesis, we use a Bayesian inference approach. In this approach, a distribution over trajectories and control sequences is computed, where the trajectory that maximizes the probability of receiving a reward is defined as policy. These rewards are coupled with our objectives (e.g. achieving a desired end effector position). This has the advantage that multiple objectives can be defined in arbitrary spaces. However, we assume knowledge about the kinematic models that map objectives to states (e.g., end effector Cartesian coordinates to joint positions).

[1] The source of the picture is www.aldebaran.com
[2] The source of the picture is www.ias.tu-darmstadt.de

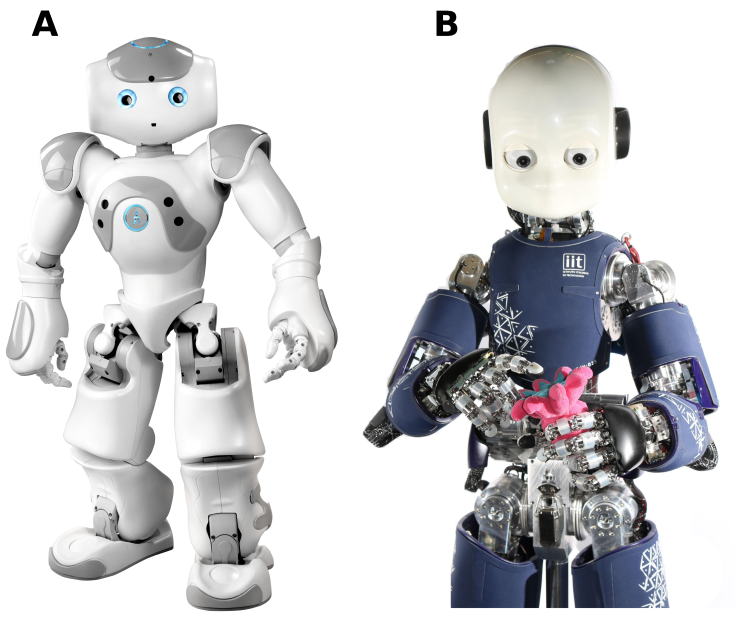


**Figure 1.1.:** Two examples of humanoid robotic platforms. In (A), the humanoid robot Nao is shown with 25 degrees of freedom. Nao is equipped with two cameras, four directional microphones, sonar rangefinder, two IR emitters, one inertial board, nine tactile sensors, and eight pressure sensors. Nao was used for two experiments, namely the arm-reaching task and the balancing task. In (B), the humanoid robot iCub with 53 degrees of freedom is depicted. iCub is equipped with actuated cameras for stereo-vision, inertial sensor, whole-body skin, tactile elements on the fingertips, 6 axis force/torque sensors, and variable-impedance actuation in the legs.

## 1.2 Simultaneous movement planning and model learning

As mentioned, the assumption of a good known forward dynamics model is often hard to fulfil, e.g., to define such a model analytically is often impossible. However, using a learned forward dynamics model during the planning process can have catastrophic effects if the model error is high and the prediction is wrong due to an insufficient learned model. Therefore, it becomes crucial to consider the model uncertainty to regulate the policy update steps, which is the reason why we use Gaussian processes. In our Bayesian inference approach we introduce the model uncertainty as an additional objective. By that the tradeoff between exploration and exploitation can be controlled in a principal manner. The goal is to have a robust planning method for multiple objective problems with stable convergence properties, using imprecise models.

## 1.3 Outlook of this thesis

In Section 2, existing approaches to solve multiple objective planning problems are discussed and we outline differences to our approach. We introduce in Section 3 the probabilistic planning method Approximate Inference Control (AICO), analyze its convergence properties in a reaching task in a light-weight robot arm and introduce the proposed regularization on the policy updates. The resulting algorithm is evaluated on the humanoid robot Nao, where in first results, arm reaching and balancing skills are inferred from desired center of gravity or end effector coordinates. In Section 4, we describe how to learn a forward dynamics model with Gaussian processes and how to consider the model uncertainty as feature. We evaluate the model learning and movement planning algorithm in a simple pendulum toy task. The computational time of our approach was considered in a light-weight robot arm. In Section 5, we conclude.

# 2 Related work

This thesis focuses on movement planning in high-dimensional nonlinear systems and simultaneous movement planning with model learning. The following Subsection 2.1 describes current approaches for movement planning. Thereafter, existing approaches for model learning are discussed in Subsection 2.2.

## 2.1 Existing approaches for movement planning

A common strategy to whole body motor control is to separate the redundant robot's configuration space into a task space and an orthogonal null space. Objectives or optimality criteria of motion are implemented as weights or priorities [2] to the redundant solutions in the null space. While these approaches have been successfully applied to a variety of tasks, including reaching, obstacle avoidance, walking and maintaining stability [5],[25],[8],[17], the application of these methods is typically limited to one step motor control, where information about future actions is not considered.

Alternatively, in Stochastic Optimal Control (SOC) problems [24],[3],[27],[32], a movement policy is optimized with respect to a cost function, which combines the different criteria of optimality in the form of different weights. For nonlinear systems, SOC methods use approximations of the system dynamics and of the cost functions, e.g., through linearizations and 2nd order Taylor expansions. These approximations are only locally correct and the updates of the policy may become unstable if the minima is not close to the points of the linearizations, or may oscillate in the case of multiple solutions. On the other hand, global planning does not need any initial solution but has much higher computational demands [12].

Many SOC methods address this issue and implement regularizations on the *algorithmic level*. E.g., in the *iLQG* method [28] a diagonal regularization term is added to the control cost Hessian[1], and in an extension [26], it was suggested to penalize deviations from the state trajectory used for linearization rather than controls. A drawback of this approach is that the additive regularization term needs rapid re-scaling to prevent divergence and accurate fine-tuning of a learning rate to find good solutions, which is challenging and increases the computational time of the algorithm.

Probabilistic planning methods that translate the SOC problem into an inference problem [10], typically implement learning rates in their belief updates [29] or in the feedback controller [22]. However, in nonlinear systems, both strategies are suboptimal in the sense that even with a small learning rate on the beliefs the corresponding control updates might be large and vice-versa, respectively.

We propose to regularize the policy updated on the *cost function level* for probabilistic plan-

[1] The update step in the trajectory optimizer corresponds to a Gauss-Newton Hessian approximation [26]

ning. We also penalize large distances between two successive trajectories in the iterative trajectory optimization procedure. In [26], the regularization term is only used for the control gains and not for the updates of the value function. However, the deviation from the linearization point can still be high if small regularization terms are used. In our approach, we always want to stay close to the linearization point as the used approximations are only locally correct. Hence, using too large update steps by greedily exploiting the inaccurate models might again be dangerous, leading the instabilities or oscillations. The scaling parameter of our punishment term serves as step size of the policy update. Due to the use of probabilistic planning, the need of an additional learning rate and complex update strategies of this learning rate can be avoided. Moreover, we will demonstrate that this type of regularization results in more robust policy updates in comparison to [26]. We choose the Approximate Inference Control (AICO) algorithm as probabilistic planning method [29] to discuss and analyze the proposed regularization, however, the same "*trick*" can be applied to large variety of SOC methods.

## 2.2 Existing approaches for model learning

To define a correct analytical forward dynamics model is often impossible or challenging for high-dimensional robots. Therefore, learning approaches can be used to learn such a forward dynamics model.

Model learning combined with dynamic programming was done by [23], where the forward dynamics was learned with Bayesian locally weighted regression. The approach is based on stochastic dynamic programming in discretized spaces. The model uncertainty was treated as noise. However, as this approach discretizes the state space and is only applicable to low-dimensional systems.

Model learning with AICO was used in [35], where the forward and inverse dynamic are learned with locally weighted projection regression (LWPR). The evaluation was done for a seven degrees of freedom (DOF) robot, where the trajectory was planned in joint space. However, to tune the parameters in LWPR is not easy. In addition, AICO deverges if the prediction is bad, gets slow with many data points as it stores and compares to all points.
To overcome these drawbacks, we use for movement planning in task space our new regularized version of AICO with Gaussian Processes for model learning. As Gaussian processes estimate the model uncertainty, we integrate in the planning as an additional feature.

In [6] Gaussian processes are used to learn a forward dynamics model, where again the model uncertainty is treated as noise. PILCO (Probabilistic Inference for Learning Control) is a model-based policy search method that uses approximate inference for long-term predictions and policy evaluation. The analytical policy gradients are used for the policy improvement. It was shown that PILCO learns in a few seconds a swing-up in a cart-pole task, which is a continuous state-action planning problem. A further successful experiment was done in a five degree of freedom robot with a state space of twelve dimensions and a control space of two dimensions.
In our approach, we regularize how to explore in uncertain regions by using the model uncertainty as an additional feature. By setting the desired model variance to zero we directly

learn the task and by increasing the desired model uncertainty the algorithm starts to explore. Thus, the executed movements are in areas with a known model uncertainty. We apply our approach in a simple pendulum toy task and in the biorobo platform, which has a state space of ten dimensions (position and velcoity are included) and a control space of five dimensions. However, it is not clear how to executed movements in known areas or how to define a model uncertainty feature in PILCO.

# 3 Robust policy updates for stochastic optimal control

Controlling high-dimensional robots can be formalized as solving a stochastic optimal control problem (SOC), which is done in Subsection 3.1. Deriving an analytical optimal solution requires certain assumptions, namely Linear system dynamic, Quadratic costs, and Gaussian noise (LQG). The classical way to solve a SOC problem in the LQG case is described in Subsection 3.2. Another approach to solve a SOC problem can be implemented in the probabilistic inference framework. The relation of these approaches is considered in Subsection 3.3. Unfortunately, the convergence of this approach is not guaranteed, which is shown in Subsection 3.4. Therefore, we add in Subsection 3.5 a regularization term that punishes large policy update steps in the trajectory optimization procedure. The evaluation for the resulting algorithm on for two tasks is shown in Subsection 3.6.

## 3.1 Problem formulation as a stochastic optimal control problem

For optimal decision making we consider finite horizon Markov Decision Processes (MDP)[1]. Finite horizon means that the length of the movement trajectory is finite. A MDP is completely defined by its state space $\mathbf{q}_t \in \mathbb{Q}$, its action space $\mathbf{u}_t \in \mathbb{U}$, its transition dynamics $P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t)$, its cost function $C_t(\mathbf{q}_t, \mathbf{u}_t) \in \mathbb{R}^1$, and its initial state probabilities $P(\mathbf{q}_0)$.

Let $\mathbf{q}_t \in \mathbb{Q}$ denote the current robot's state in configuration space (e.g., a concatenation of joint angles, joint velocities and reference coordinates in floating base systems) and let vector $\mathbf{x}_t \in \mathbb{X}$ denote task space features like end effector positions or the center of gravity of a humanoid (these features will be used to specify a cost function later). At time $t$, the robot executes the action $\mathbf{u}_t \in \mathbb{U}$ according to the movement policy $\pi(\mathbf{u}_t|\mathbf{q}_t)$[2]. The chosen action at the current state is evaluated by the cost function $C_t(\mathbf{q}_t, \mathbf{u}_t) \in \mathbb{R}^1$ and results in a state transition characterized by the probability $P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t)$. In stochastic optimal control (SOC), the goal is to find a optimal stochastic policy $\pi^*$ that minimizes the expected cost. Therefore, we define the expected cost as a value function given the policy, the initial state distribution, and the transition dynamics. Obviously, for the last time step $T$ we cannot perform any action, i.e. our value function is $J_T(\mathbf{q}_T) = C_T(\mathbf{q}_T)$. For the previous time steps we can use dynamic programming and iterate backwards in time, i.e. our value function is

$$J_t^\pi(\mathbf{q}_t) = C_t(\mathbf{q}_t, \mathbf{u}_t) + \mathbb{E}_P\left[J_{t+1}^\pi(\mathbf{q}_{t+1})|\mathbf{q}_t, \mathbf{u}_t\right].$$

[1] Note that the same principle of regulating the update steps in trajectory optimization can also be applied to planning algorithms in infinite horizon problems such as [9, 32]

[2] Note that the policy is time-dependent because we are in the finite horizon case. Hence, the reward function and the probabilistic transition model are also time-dependent.

But as described above, the optimal policy minimizes the expected cost. To do so, the optimal value function is computed by taking the optimal action at each time step. This leads to

$$J_t^{\pi^*}(\mathbf{q}_t) = \underset{\mathbf{u}_t}{\operatorname{argmin}} \left(C_t(\mathbf{q}_t, \mathbf{u}_t) + \mathbb{E}_P\left[J_{t+1}^{\pi^*}(\mathbf{q}_{t+1}) | \mathbf{q}_t, \mathbf{u}_t\right]\right), \tag{3.1}$$

$$= \underset{\mathbf{u}_t}{\operatorname{argmin}} \left(C_t(\mathbf{q}_t, \mathbf{u}_t) + \int_{q_{t+1}} P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t) J_{t+1}^{\pi^*}(\mathbf{q}_{t+1})\, \mathrm{d}\mathbf{q}_{t+1}\right). \tag{3.2}$$

Now the problem is to solve the expectation and the minimization operator. Only under special assumption we can solve analytically the stochastic optimal control problem, which is described in the next subsection.

## 3.2 Solving a SOC problem for the Linear-Quadratic-Gaussian case

Solving Equation (3.2) analytically requires special assumptions, namely Linear system dynamic, Quadratic costs, and Gaussian noise (LQG). To do so, the system dynamic becomes

$$P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t),$$

where $\mathbf{A}_t$ is called state transition matrix, $\mathbf{a}_t$ is the linear drift term, $\mathbf{B}_t$ the control matrix, and $\mathcal{N}$ denotes a normal distribution. In addition, we assume quadratic a cost function

$$C_t(\mathbf{q}_t, \mathbf{u}_t) = \mathbf{q}_t^T\mathbf{R}_t\mathbf{q}_t - 2\mathbf{r}_t^T\mathbf{q}_t + \mathbf{u}_t^T\mathbf{H}_t\mathbf{u}_t.$$

This results in a quadratic value function for the following time step

$$J_{t+1}^{\pi^*}(\mathbf{q}_{t+1}) = \mathbf{q}_{t+1}^T\mathbf{V}_{t+1}\mathbf{q}_{t+1} - 2\mathbf{v}_{t+1}^T\mathbf{q}_{t+1}.$$

With these assumptions it is possible to reformulate Equation (3.2)

$$\begin{aligned} J_t^{\pi^*}(\mathbf{q}_t) &= \underset{\mathbf{u}_t}{\operatorname{argmin}} \left(C_t(\mathbf{q}_t, \mathbf{u}_t) + \int_{q_{t+1}} P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t) J_{t+1}^{\pi^*}(\mathbf{q}_{t+1})\, \mathrm{d}\mathbf{q}_{t+1}\right), \\ &= \underset{\mathbf{u}_t}{\operatorname{argmin}} \Big(\mathbf{q}_t^T\mathbf{R}_t\mathbf{q}_t - 2\mathbf{r}_t^T\mathbf{q}_t + \mathbf{u}_t^T\mathbf{H}_t\mathbf{u}_t + \\ &\int_{q_{t+1}} \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t)(\mathbf{q}_{t+1}^T\mathbf{V}_{t+1}\mathbf{q}_{t+1} - 2\mathbf{v}_{t+1}^T\mathbf{q}_{t+1})\, \mathrm{d}\mathbf{q}_{t+1}\Big), \end{aligned}$$

which can be solved in closed form (c.f Appendix B). Note that the optimal trajectory is only independent of the model noise $\mathbf{Q}_t$, if the maximum a posteriori solution is used. These results in the Riccaty Equations, where the optimal value function is

$$J_t^{\pi^*}(\mathbf{q}_t) = \mathbf{q}_t^T\mathbf{V}_t\mathbf{q}_t - 2\mathbf{v}_t^T\mathbf{q}_t,$$

with the following entries

$$\mathbf{V}_t = \mathbf{R}_t + \left(\mathbf{A}_t^T - \mathbf{K}\right)\mathbf{V}_{t+1}\mathbf{A}_t,$$
$$\mathbf{v}_t = \mathbf{r}_t + \left(\mathbf{A}_t^T - \mathbf{K}\right)(\mathbf{v}_{t+1} - \mathbf{V}_{t+1}\mathbf{a}_t),$$
$$\mathbf{K} = \mathbf{A}_t^T\mathbf{V}_{t+1}\left(\mathbf{V}_{t+1} + \mathbf{B}_t\mathbf{H}_t^{-1}\mathbf{B}_t^T\right)^{-1}.$$

Another way to solve a stochastic optimal control problem is discussed in the next Subsection.

## 3.3 Probabilistic inference approach to solve a SOC problem

An interesting class of algorithms to SOC problems have been derived by reformulating the original Bellman formulation in Equation (3.2) as a Bayesian inference problem [30],[7], [31], [11]. Instead of minimizing costs, the idea is to maximize the probability of receiving a reward event ($z_t = 1$) at every time step

$$P(z_t = 1|\mathbf{q}_t, \mathbf{u}_t) \propto \exp\{-C_t(\mathbf{q}_t, \mathbf{u}_t)\}. \tag{3.3}$$

Note that the idea of turning the cost function in Equation (3.3) into a reward signal was also used in operational space control approaches [18], [19]. In general, minimizing the expected cost is not equivalent to maximizing the probability of receiving a reward as

$$\begin{aligned}\log P(z_{0:T} = 1) &= \log \mathbb{E}_{\mathbf{q}_{0:T},\mathbf{u}_{0:T-1}}\left(P(z_{0:T} = 1|\mathbf{q}_{0:T}, \mathbf{u}_{0:T-1})\right),\\ &\underset{\geq}{(1)} \mathbb{E}_{\mathbf{q}_{0:T},\mathbf{u}_{0:T-1}}\left(\log P(z_{0:T} = 1|\mathbf{q}_{0:T}, \mathbf{u}_{0:T-1})\right),\\ &= -\mathbb{E}_{\mathbf{q}_{0:T},\mathbf{u}_{0:T-1}}\left(C_t(\mathbf{q}_{0:T}, \mathbf{u}_{0:T-1})\right).\end{aligned}$$

Note that (1) is applicable with Jensen's inequality[3]. Only in the LQG case they are coincide, i.e.

$$\log P(z_{0:T} = 1) = -\mathbb{E}_{\mathbf{q}_{0:T},\mathbf{u}_{0:T-1}}\left(C_t(\mathbf{q}_{0:T}, \mathbf{u}_{0:T-1})\right).$$

In the probabilistic framework, we want to compute the posterior over state and control sequences, conditioning on observing a reward at every time step

$$P(\mathbf{q}_{0:T}, \mathbf{u}_{0:T-1}|z_{0:T} = 1) = P(\mathbf{q}_0)\prod_{t=0}^{T-1}\pi(\mathbf{u}_t|\mathbf{q}_t)P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t)\prod_{t=1}^{T}P(z_t = 1|\mathbf{q}_t, \mathbf{u}_t),$$

where $P(\mathbf{q}_0)$ is the initial state distribution, $P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t)$ the state transition model, and the policy $\pi(\mathbf{u}_t|\mathbf{q}_t)$ chooses the control $\mathbf{u}_t$ for a given state $\mathbf{q}_t$. For efficient implementations of this inference problem, a number of algorithms have been proposed that apply iterative policy updates assuming that all probability distributions can be modeled by an instance of the family of *exponential* distributions [29],[20], [34]. We will restrict our discussion on the Approximate Inference Control (AICO) algorithm with Gaussians [29]. A more detailed de-

[3] Jensen' inequality $f(\mathbb{E}(X)) \geq \mathbb{E}(f(X))$ is applicable for an expectation if the function $f$ is concave and $X$ is a integrable real-valued random variable.

scription of message passing in general factor graphs is given in [36],[16],[15].

Most real robotic systems are non-LQG and therefore our assumptions are violated. Hence, in AICO (with Gaussians), the system dynamics are linearized through 1st order Taylor expansion, where the state transition matrix $\mathbf{A}_t$, the linear drift term $\mathbf{a}_t$ and the control matrix $\mathbf{B}_t$ are often computed with derivatives simulated through finite differences. The numerical stability of AICO depends on the accuracy of the linearized model, we will therefore additionally compare to an approximation of the system dynamics, where controls $\mathbf{u}_t$ correspond directly to joint accelerations (compare Appendix C). We will refer to this approximation as *pseudo-dynamic* model. We propose to add a regularization term to the cost function. Before explaining the regularization term in more detail, we briefly discuss how different objectives are implemented in AICO. In the simplest case, the task-likelihood function in Equation (3.3) can be split into separate state and a control dependent terms, i.e.,

$$P(z_t = 1|\mathbf{q}_t, \mathbf{u}_t) = \mathcal{N}[\mathbf{q}_t|\mathbf{r}_t, \mathbf{R}_t]\mathcal{N}[\mathbf{u}_t|\mathbf{h}_t, \mathbf{H}_t] \enspace , \tag{3.4}$$

where, for analytical reasons, the Gaussians are given in *canonical* form, i.e.,

$$\mathcal{N}[\mathbf{u}_t|\mathbf{h}_t, \mathbf{H}_t] \propto \exp(-1/2\,\mathbf{u}_t^T\mathbf{H}_t\mathbf{u}_t + \mathbf{u}_t^T\mathbf{h}_t).$$

By inserting Equation (3.4) in Equation (3.3) we obtain the quadratic costs,

$$C_t(\mathbf{q}_t, \mathbf{u}_t) = \mathbf{q}_t^T\mathbf{R}_t\mathbf{q}_t - 2\mathbf{r}_t^T\mathbf{q}_t + \mathbf{u}_t^T\mathbf{H}_t\mathbf{u}_t - 2\mathbf{h}_t^T\mathbf{u}_t \enspace . \tag{3.5}$$

The state dependent costs, encoded by $\mathcal{N}[\mathbf{q}_t|\mathbf{r}_t, \mathbf{R}_t]$, can be defined in configuration space, in task space, or even in combinations of both spaces [31]. To be precise, reaching a goal state $\mathbf{g}^* \in \mathbb{Q}$ in configuration space can be encoded by $\mathbf{r}_t = \mathbf{R}_t\mathbf{g}^*$ ,where the precision matrix $\mathbf{R}_t$ scales the importance of different dimensions. In task space, let be $\mathbf{x}^* \in \mathbb{X}$ a desired end effector position and let $\mathbf{x} = f(\mathbf{q})$ be the forward kinematics mapping and $J(\mathbf{q}_t) = \partial f/\partial\mathbf{q}|\mathbf{q} = \mathbf{q}_t$ its Jacobian. We can now obtain a Gaussian task likelihood by approximating the forward kinematics by its linearization through the Jacobian, i.e., $\mathbf{x} \approx f(\mathbf{q}_0) + \mathbf{J}(\mathbf{q} - \mathbf{q}_0)$. The parameters of the Gaussian are then given by $\mathbf{r}_t = \mathbf{J}^T C\big(f(\mathbf{q}_0) - \mathbf{x}^*\big)$ and $\mathbf{R}_t = \mathbf{J}^T\mathbf{C}\mathbf{J}$, where the diagonal elements of the matrix $\mathbf{C}$ specify the desired precision in task space.

On the algorithmic level, AICO combines forward messages and backward messages to compute the belief over trajectories. We represent these Gaussian forward messages by $N[\mathbf{q}_t|\mathbf{s}_t, \mathbf{S}_t]$, the backward messages by $N[\mathbf{q}_t|\mathbf{v}_t, \mathbf{V}_t]$, and the belief by $N[\mathbf{q}_t|\mathbf{b}_t, \mathbf{B}_t]$. The recursive update equations are given in [29] and in [22], where an implementation which additionally implements control constraints (otherwise $\mathbf{h}_t = 0$) is given.

We can also compute the most likely action given the task constraints. By doing so, in the case of AICO with Gaussians, we obtain a time-varying linear feedback controller with time-varying offset

$$\mathbf{u}_t^{[n]} = \mathbf{o}_t + \mathbf{O}_t\mathbf{q}_t \enspace , \tag{3.6}$$

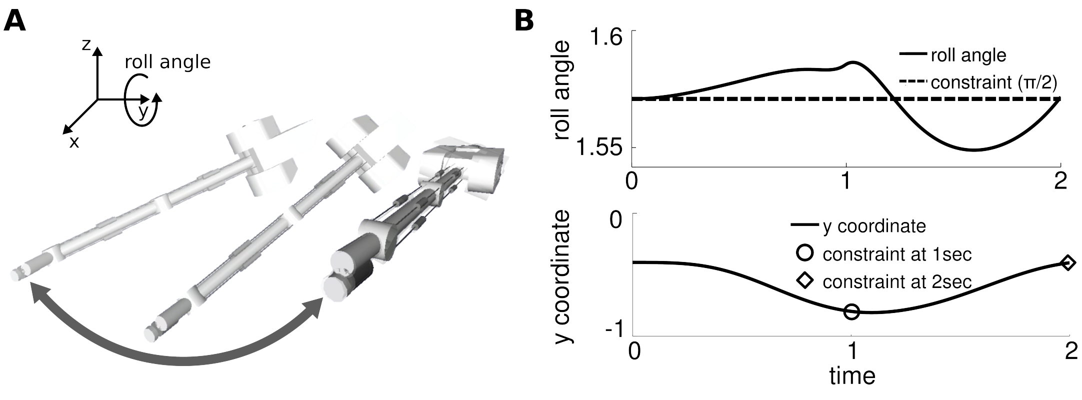


**Figure 3.1.:** A 5-degree-of-freedom robot arm has to reach for a via-point (the posture on the left in A) and return to its initial pose (the posture on the right in A). The reaching task is encoded in four task objectives, i.e., three cartesian coordinates and the roll angle of the end effector. The inferred trajectories for the y coordinate and the roll angle, including the objectives, are shown in (B).

where $\mathbf{o}_t$ is an open loop gain and $\mathbf{O}_t$ denotes the feedback gain matrix ($n$ denotes the iteration). The gains are given with

$$\begin{aligned}\mathbf{o}_t &= \mathbf{M}_t^{-1}\left(\mathbf{B}_t^T\mathbf{V}_*\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1} - \mathbf{B}_t^T\mathbf{V}_*\mathbf{a}_t + \mathbf{h}_t\right),\\ \mathbf{O}_t &= -\mathbf{M}_t^{-1}\mathbf{B}_t^T\mathbf{V}_*\mathbf{A}_t.\end{aligned}$$

The controller can be computed in closed form, where the solution is given in the Appendix D. Also the derivation of AICO is listed in the Appendix D.

## 3.4 Evaluation of the convergence properties of AICO

To investigate the convergence properties of AICO, we use a simulated light-weight robot arm [13] with five joints. The robot has to reach a desired end effector position in cartesian space and subsequently has to return to its initial pose. To increase the complexity, we define a second task, where the robot should additionally keep the roll angle of the end effector constant. For this task, we used the cost function

$$C_t(\mathbf{q}_t,\mathbf{u}_t) = \begin{cases} 10^4(\mathbf{x}^i - \mathbf{x}_t)^T(\mathbf{x}^i - \mathbf{x}_t) + 10^{-2}\mathbf{u}^T\mathbf{u} & \text{if } t = T^i \\ 10^{-2}\mathbf{u}^T\mathbf{u} & \text{else} \end{cases}, \tag{3.7}$$

where $\mathbf{x}^i$ denotes the desired robot postures in task space at times $T^1 = 500$ and $T^2 = 10^3$ (the planning horizon is 2 seconds with a time step of 2ms) with $\mathbf{x}^1 = [1, -0.4, 0, 0, \pi/2, 0]^T$ and $\mathbf{x}^2 = [1, 0, 0, 0, \pi/2, 0]^T$. Note that we do not assume any initial solution to initialize the planner, solely the initial posture of the robot in configuration space is used as initial 'trajectory'. An example movement is shown in Figure 3.1.

Using the *pseudo-dynamics* approximation of the system dynamics, the convergence rate of the costs per iteration of both tasks are shown in Figure 3.2A,B. For the simple task in Figure

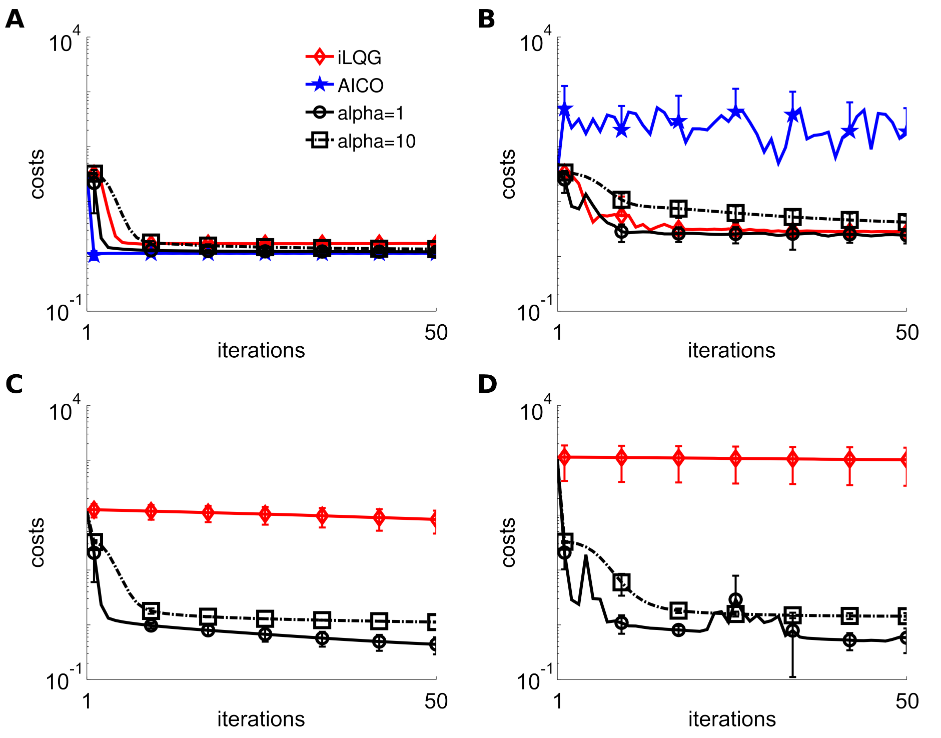


**Figure 3.2.:** Comparison of the convergence properties of *iLQG*, AICO and our regularized variant, where the rate of convergence is controlled via the parameter $\alpha$. In the top row (A-B), the model of the forward dynamics was approximated by a *pseudo dynamics* model [33]. In the bottom row, an analytic forward dynamics model of a 5-degree-of-freedom robot arm was used. The panels in the first column denote the costs of the planning algorithms applied to a simple task, where the robot arm has to reach for an end effector target and return to the initial state. In the second column (B,D), the robot has to keep additionally the roll angle constant (at $\pi/2$). Shown are the mean and the standard deviations for 10 initial states '$\mathbf{q}_0$ sampled from a Gaussian with zero mean and a standard deviation of 0.05.

3.2A the inferred cost values converge fast for all algorithms, with the standard AICO algorithm showing the best performance. However, the fast convergence also comes with the costs of a reduced robustness of the policy update as can be seen from the results in the second scenario illustrated in Figure 3.2B, where AICO is unstable and cannot infer solutions with low costs. When we used the analytic forward dynamics model (where the linearizations are computed through finite differences) instead of the pseudo dynamics model, computing the messages in AICO became numerically unstable and no solutions could be inferred. Therefore, the panels in Figure 3.2C,D do not include results of AICO. We also evaluated the *iLQG* method [28] that implements an adaptive regularization schedule and line search to prevent divergence [26]. While the *iLQG* algorithm performed well for the pseudo dynamcis model, the learning rate was automatically decreased to almost zero for the analytical dynamics model. Our regularization method for AICO, that we will present in the next Section, considerably outperformed both competing methods.

**Algorithm 1:** Approximate Inference Control with Regularized Update Steps

```
Data: initial state $\mathbf{q}_0$, parameter $\alpha^{[0]}$, threshold $\theta$
Result: feedback control law $\mathbf{o}_{0:T-1}$ and $O_{0:T-1}$
initialize $\mathbf{q}^{[0]}_{1:T} = \mathbf{q}_0$, $S_0 = 1e10 \cdot \mathbf{I}$, $\mathbf{s}_0 = S_0\mathbf{q}_0$, $n = 1$
while not converged do
    $\mathbf{q}^{[n-1]}_{0:T} = \mathbf{q}^{[n]}_{0:T}$
    for $t \leftarrow 1$ to $T$ do
        linearize model: $A_t, \mathbf{a}_t, B_t$
        compute: $H_t, \mathbf{h}_t, R_t, \mathbf{r}_t$
        update: $\mathbf{s}_t$, $S_t$, $\mathbf{v}_t$, $V_t$, $\mathbf{b}_t$, and $B_t$
        if $||\mathbf{b}_t - \mathbf{q}^{[n]}_t|| > \theta$ then
            repeat this time step
            $t \leftarrow t - 1$
        end
        $\mathbf{q}^{[n]}_t = B^{-1}_t \mathbf{b}_t$
    end
    for $t \leftarrow T - 1$ to $0$ do
        ..same updates as above...
    end
    for $t \leftarrow 0$ to $T - 1$ do
        compute feedback controller: $\mathbf{o}_t, O_t$
        $\mathbf{u}^{[n]}_t = \mathbf{o}_t + O_t \mathbf{q}_t$
        $\mathbf{q}^{[n]}_{t+1} = A_t \mathbf{q}^{[n]}_t + \mathbf{a}_t + B_t \mathbf{u}^{[n]}_t$
    end
    $n = n + 1$
    $\alpha^{[n]} = \alpha^{[n-1]}\gamma$
end
return $\mathbf{o}_{0:T-1}$ and $O_{0:T-1}$
```

## 3.5 Regulating the policy updates in AICO

To regularize the policy update steps in Equation (3.2), we add an additional cost term to the task-likelihood function, i.e.,

$$P(z_t = 1|\mathbf{q}^{[n]}_t, \mathbf{u}^{[n]}_t) \propto \exp\{-C_t(\mathbf{q}^{[n]}_t, \mathbf{u}^{[n]}_t) - \alpha^{[n]}(\mathbf{q}^{[n]}_t - \mathbf{q}^{[n-1]}_t)^T(\mathbf{q}^{[n]}_t - \mathbf{q}^{[n-1]}_t)\} \ , \qquad (3.8)$$

which punishes the distance of the state trajectories of two successive iterations of the algorithm ($n$ denotes the iteration). The parameter $\alpha$ controls the size of the update step. For large $\alpha$, the trajectory update will be conservative as the algorithm will stay close to the previous trajectory that has been used for linearization. For small $\alpha$ values, the new trajectory will directly jump to the LQG solution given the linearized dynamics and the approximated costs. Hence, $\alpha$ is inverse proportional to the step size. The value of $\alpha$ is updated after each iteration according to $\alpha^{[n]} = \alpha^{[n-1]}\gamma$. For $\alpha^{[0]} \geq 1$ and $\gamma > 1$, convergence is guaranteed as the regularization term will dominate with an increasing number of iterations.

The algorithm is listed in Algorithm 1. An interesting feature of this algorithm is that no learning rate is needed as $\alpha$ is used directly to implement a step size. In the original formulation of AICO the learning rate is either applied to the state update (in Line 13 in Algorithm 1) [29] or to the feedback controller (in Line 18 in Algorithm 1) [22]. However, neither implementation can guarantee convergence in nonlinear systems or in tasks with costs inducing a nonlinear mapping from $\mathbb{Q}$ to $\mathbb{X}$.

We evaluate the resulting algorithm on the same robot arm reaching tasks. For both tasks, the cartesian planning task in Figure 3.2A,C and the extension with the additional roll angle objective in Figure 3.2B,D, we evaluated AICO with the regularization parameter $\alpha \in \{1, 10\}$ (we did not increase $\alpha$ and $\gamma = 1$). For both models of the system dynamics, the *pseudo-dynamics* approximation (shown in Figure 3.2A,B) and the analytic model (illustrated in Figure 3.2C,D), AICO benefits from the regularization term and the costs decay exponentially fast. Interestingly, without "good" initial solutions, the differential dynamic programming method *iLQG* [26] that implements a sophisticated regularization scheme cannot generate movement policies with low costs when using the analytic model. This is shown in Figure 3.2C,D. To proof our new algorithm we investigate more complex tasks. Therefore, we develop two further task on the Nao platform, which are desribed in the next Subsections.

## 3.6 Results of R-AICO in humanoids

We evaluated the proposed planning method in simulation with the humanoid robot Nao. The Nao robot has 25 degrees-of-freedom. In first experiments, we investigated the performance of the planner with a *pseudo-dynamics* model of the robot. The humanoid had to reach for an end effector target using the right arm while maintaining balance. In a second experiment, Nao had to shift the x-coordinate of the center of gravity while maintaining balance.

### 3.6.1 Arm Reaching with a Humanoid Robot

The humanoid has to reach for the end effector target $\mathbf{x}^* = [0, 0.2, 0.06]^T$, where only the y- and the z-Cartesian coordinates are relevant. Additionally, the robot has to maintain balance, which is implemented as deviation of the center of gravity vectors from its initial values $\mathbf{x}_{\text{CoG}}(t = 0)$, i.e., we specify the desired center of gravity as $\mathbf{x}^*_{\text{CoG}} = \mathbf{x}_{\text{CoG}}(t = 0)$. The same cost function as in the experiments for the light weight robot arm in Equation (3.7) is used. For this task, however, only a single via-point is defined that is used for the desired end effector target and the center of gravity, i.e., $\mathbf{x}^1 = [\mathbf{x}^{*T}, \mathbf{x}^{*}_{\text{CoG}}{}^T]^T$.

Only by specifying two scalars in $\mathbf{x}^*$ (the scaling parameters in (3.7) are constants that take the values $10^4$ or $10^{-2}$), the planning algorithm infers 50-dimensional state trajectories (the state $\mathbf{q}_t$ at time $t$ encodes the joint angles and the joint velocities, ignoring the base frame). This is shown in Figure 3.3A for the proposed planning algorithm with the regularization parameter $\alpha = 1$. As in the robot arm experiments, the Approximate Inference Control algorithm (AICO) benefits from the regularization. As can be seen in Figure 3.3B, AICO cannot infer movement solutions with low costs without regularization.

Interestingly, to maintain balance, the humanoid utilizes its head and its left arm for which no objectives were explicitely specified. This effect is a feature of model-based planning methods

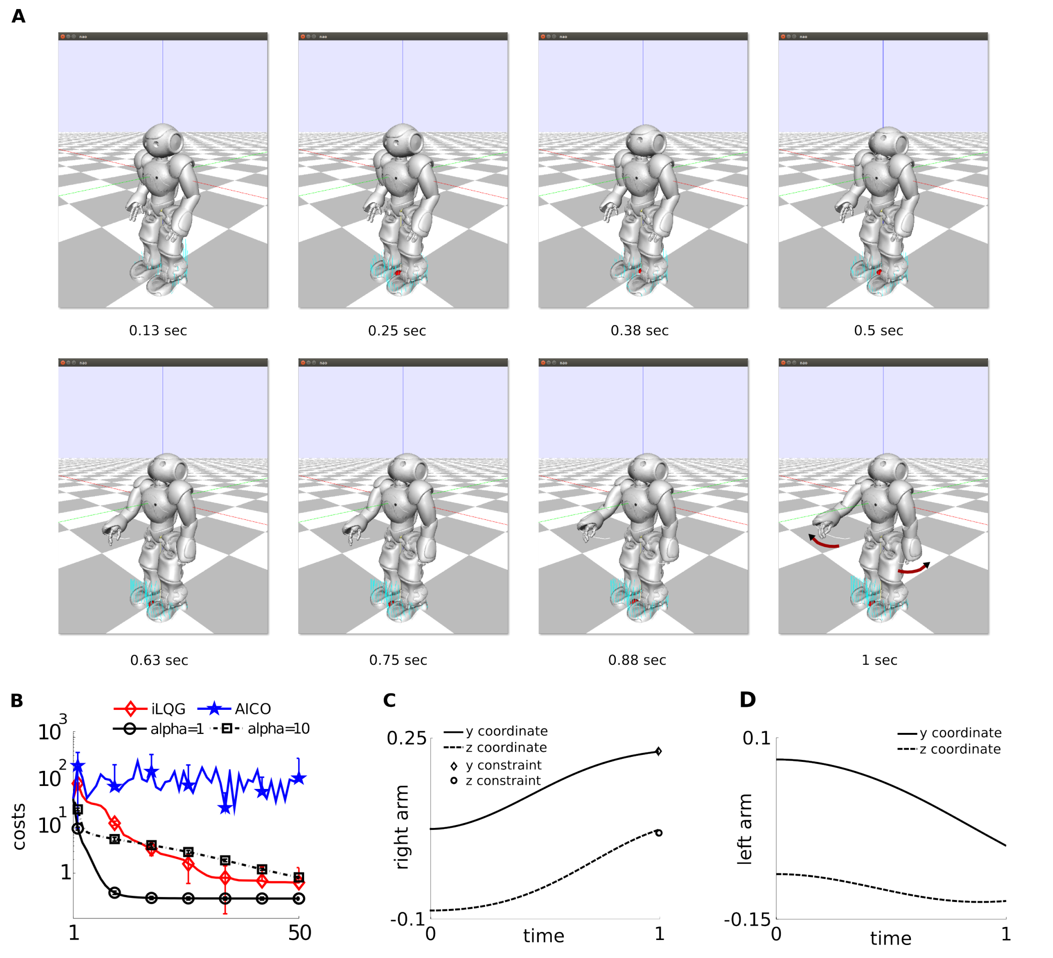


**Figure 3.3.:** Reaching task with the humanoid robot Nao. The robot has to reach a desired end effector position with the right arm while maintaining balance. Eight snapshots of the inferred movement are shown in (A). In (B), the convergence of the costs of the optimization procedure is shown, where we compare *iLQG*, the standard implementation of AICO and the regularized variant. The mean and the standard deviations for 10 initial states ‘$\mathbf{q}_0$ are sampled from a Gaussian with zero mean and a standard deviation of 0.05. The movement objectives for the right arm are shown in (C). To counter balance, the robot moves its left hand and the head, which is shown in (D).

that consider the coupled dynamics and is best illustrated in Figure 3.3C,D, where the end effector trajectories of both arms and the desired target values are shown.

### 3.6.2 Balancing with a humanoid

In this task the humanoid has to balance on one foot by moving its center of gravity. In this experiment, we specify three desired via-points for the center of gravity, i.e., $\mathbf{x}^i = \mathbf{x}^i_{\text{CoG}}$ with $i = 1, ..., 3$. The last via-point is set to the initial center of gravity $\mathbf{x}_{\text{CoG}}(t = 0)$. The first via-point has an offset of 0.1m in the x-coordinate of $\mathbf{x}_{\text{CoG}}(t = 0)$ to force the robot to move its center of gravity to the right. The second viapoint has the same negative offset in the x-direction to exhibit a movement to the left. The planning horizon was three seconds ($T^1 = 100, T^2 = 200$ and $T^3 = 300$ with $\tau = 10$ms) and the distance matrix $C$ in (3.7) was scaled with the importance weights $[10^6, 10, 10]^T$ for the x,y, and z coordinate of $\mathbf{x}^i_{\text{CoG}}$.

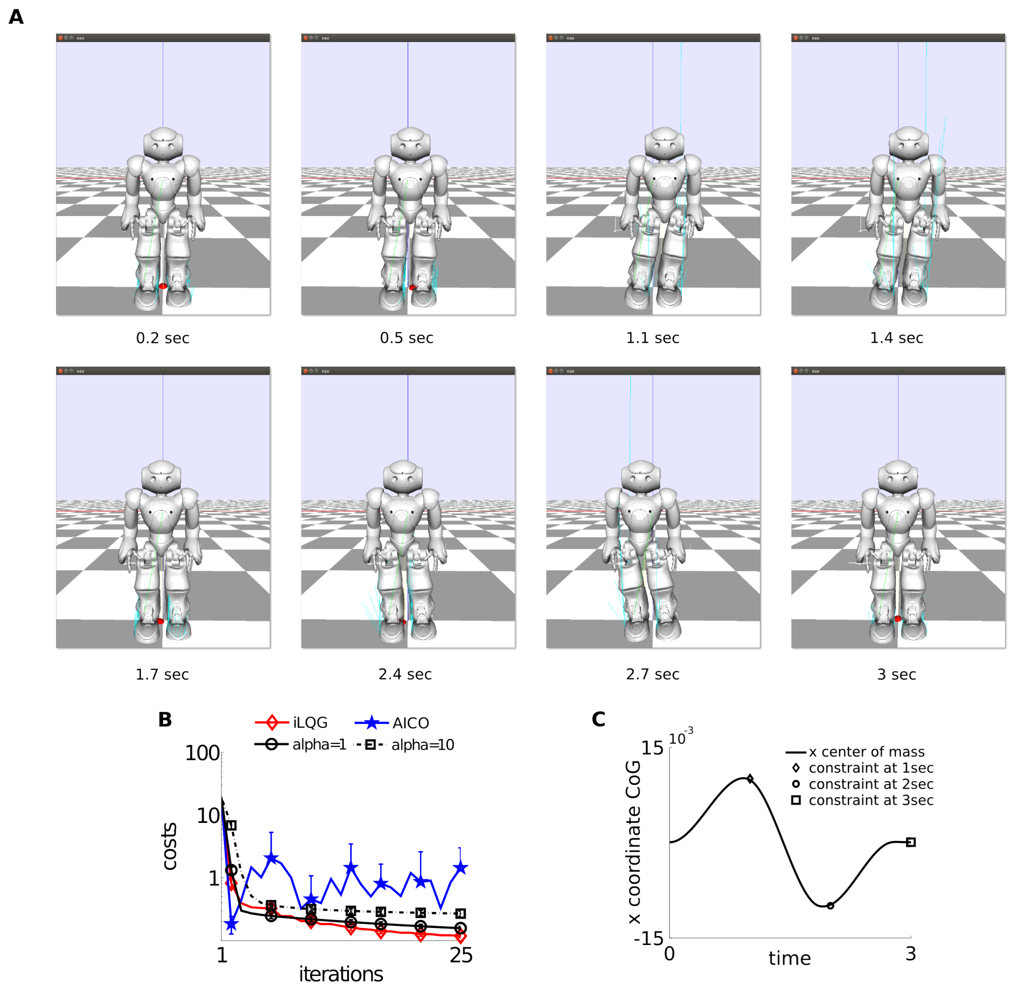


**Figure 3.4.:** Balancing task in the humanoid robot Nao. The robot should *swing* its hips, which is encoded by adding an offset scalar to the x-coordinate of the center of gravity vector. In (A) 10 snapshots of the resulting movement for an increasing planning horizon are shown for $\alpha = 1$. The convergence properties of *iLQG*, the standard AICO and its regularized variants are shown in (B). The mean and the standard deviations for 10 initial states $'\mathbf{q}_0$ are sampled from a Gaussian with zero mean and a standard deviation of 0.05. In (C) the x-coordinate of the center of gravity of the Nao is illustrated. The large dots denote the objectives.

For $\alpha = 1$, the resulting movement is illustrated in Figure 3.4A. Illustrated are 10 snapshots. Nao first moves its hip to the right (with respect to the robot frame) and thereafter to the left. This movement is the result of an inference problem encoded in mainly two scalars, i.e., the offsets.

The standard implementation of AICO was not able to infer successful balancing solutions, which is illustrated in Figure 3.4B. In contrast, the regularized variant using $\alpha \in \{1, 10\}$ converged after 25 iterations of the trajectory optimization procedure. For $\alpha = 1$, the x-coordinate of the center of gravity and the implemented objectives are shown in 3.4C.

## Computational time of R-AICO

The computational time of the proposed planning algorithm is the same as for the standard implementation of AICO. If the algorithm is implemented in C-code it achieves real time performance in humanoid planning problems [29]. However, for our experiments we used a Matlab implementation on a standard computer (2.4GHz, 8GB RAM), where, e.g., the computation of the balancing movements in Figure 3.4 took less then 50 seconds (which includes all 25 iterations of the optimization process). The movement duration of the executed trajectory was three seconds.

# 4 Simultaneous movement planning and model learning

In Section 3 we assumed a known analytical forward dyanmics model. For real robotic systems it is hard to define a sufficient accurate forward dynamics model, especially for compliant robots. Section 3 showed that even with an analytical model planning algorithms tend to diverge, e.g. AICO and iLQG. Because they have to deal with noisy data, high-dimensional space and many objectives. In this section we want to learn a forward dynamics model that can be used for planning with few data points and an inaccurate model. Therefore, we discuss how to learn a model, how we treat the model uncertainty and how to collect new data. In our model learning with parallel movement planning approach the initial model is learned through motor babbling. Hence, the Subsection 4.1 discusses how to generate such initial data points. In addition, we explain in Subsection 4.2 how we learn a model with Gaussian processes that estimate the model uncertainty. Subsection 4.3 explains how the model uncertainty is incorporated in AICO as an additional objective. The approach is evaluated on the simple pendulum as toy task in Subection 4.4 and on the biorob platform in Subsection 4.5.

## 4.1 Generating initial data points with motor babbling

Learning an initial forward dynamics model requires initial data points, which we collect through motor babbling. Starting in a stable initial robot posture with zeros torques we compute some random trajectories. In the simple pendulum setup the initial posture is the stable equilibrium point.

For the simple pendulum toy as task, we used as analytical forward model

$$\ddot{q} = -\frac{1}{m \cdot l^2}\left(\mu \dot{q} - mgl \sin q + u\right),$$

where $\mu = 1$ is the friction term, $m = 1kg$ the mass, $l = 1m$ the length, and $g = 9.81m/s^2$ the gravity. Runge-Kutta-Integration was used for the computation of the acceleration. The torques are computed with a PD-Controller

$$u_t = K_P(q_{des,t} - q_t) + K_D(\dot{q}_{des,t} - \dot{q}_t),$$

where the control gains are $K_P = 50$ and $K_D = 5$. Note that this PD-Controller is only used for motor babbling. If we execute a planned movement, we use the feedback controller in Equation (3.6). However, for both cases the controls are limited to $\pm 10$, where we defined control limits as additional objective in AICO [21]. In addition, the computed states are restricted to the range $[0, 2\pi]$. For a single random generated trajectory we get tuples in form of $(q_t, \dot{q}_t, \ddot{q}_t, u_t)$ for every time step $t$. For learning an initial model, we used 15 data points, which is dicussed in the next subsection.

## 4.2 Model learning with Gaussian processes

AICO uses a linearization of the transition dynamics

$$P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t),$$

through 1st order Taylor expansion. Therefore, we have to specify the state transition matrix $\mathbf{A}_t$, the control matrix $\mathbf{B}_t$, and the linear drift term $\mathbf{a}_t$ becomes

$$\begin{aligned}\mathbf{a}_t &= \left(f_t - \frac{\partial f_t}{\partial \mathbf{q}_t}\Delta t - \frac{\partial f_t}{\partial \mathbf{u}_t}\Delta t\right)\Delta t,\\ \mathbf{A}_t &= \left(\mathbf{I} + \frac{\partial f_t}{\partial \mathbf{q}_t}\Delta t\right),\\ \mathbf{B}_t &= \frac{\partial f_t}{\partial \mathbf{u}_t}\Delta t,\end{aligned}$$

where $f_t$ is the forward dynamics model, with $\ddot{\mathbf{q}} = f(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{u}_t) = f_t$. The time step is denoted by $t$. We want to learn the forward dynamics model $f_t$ with Gaussian processes. The training inputs are defined as tuples $\hat{\mathbf{x}}_t = (\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{u}_t) \in \mathbb{R}^{2D+F}$. The number of DOF is denoted by $D$ and $F$ is the dimension of the torques. The training target is defined as $\ddot{\mathbf{q}}_t \in \mathbb{R}^D$. A Gaussian process is completely defined by a mean function and a positive semidefinite covariance function. We use zero mean and the squared exponential covariance function

$$k(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\hat{\mathbf{x}}_p - \hat{\mathbf{x}}_q)^T \mathbf{\Lambda}^{-1}(\hat{\mathbf{x}}_p - \hat{\mathbf{x}}_q)\right) + \delta_{pq}\sigma_w^2.$$

The hyper-parameters $\boldsymbol{\theta}$ are the length-scales $l_i$, signal variance $\sigma_f^2$, and noise variance $\sigma_w^2$ with $\mathbf{\Lambda} = \mathrm{diag}([l_1^2, ..., l_{2D+F}^2])$. The hyper-parameter optimization was done by maximizing the log likelihood function $\ln P(\mathbf{y}|\boldsymbol{\theta})$. Given $N$ data points (which can represent several trajectories), the training set is defined as $\mathbf{X} = [\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_N]$ and $\mathbf{y} = [\ddot{\mathbf{q}}_1, ..., \ddot{\mathbf{q}}_N]$. For a new input $\mathbf{x}_*$, the mean and variance of the GP prediction are

$$\begin{aligned}\mathrm{m}(\hat{\mathbf{x}}_*) &= k(\mathbf{X}, \hat{\mathbf{x}}_*)^T\left(\mathbf{K} + \sigma_w^2\mathbf{I}\right)^{-1}\mathbf{y},\\ \mathrm{var}(\hat{\mathbf{x}}_*) &= k(\hat{\mathbf{x}}_*, \hat{\mathbf{x}}_*) - k(\mathbf{X}, \hat{\mathbf{x}}_*)^T\left(\mathbf{K} + \sigma_w^2\mathbf{I}\right)^{-1}k(\mathbf{X}, \hat{\mathbf{x}}_*),\end{aligned}$$

where $\mathbf{K}$ is the kernel matrix with entries $K_{pq} = k(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q)$. We train for every output dimension a single GP. AICO uses a linearization of the transition dynamics it is also necessary to compute the derivative of the predicted mean,

$$\frac{\partial\,\mathrm{m}(\hat{\mathbf{x}}_*)}{\partial \hat{\mathbf{x}}_*} = -2(\Lambda^{-1}\begin{bmatrix}\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_*\\ \vdots\\ \hat{\mathbf{x}}_N - \hat{\mathbf{x}}_*\end{bmatrix} \odot k(\mathbf{X}, \hat{\mathbf{x}}_*))^T\left(\mathbf{K} + \sigma_w^2\mathbf{I}\right)^{-1}, \tag{4.1}$$

where $\odot$ is a pointwise product. The derivatives $\partial f_t / \partial \mathbf{q}_t$ and $\partial f_t / \partial \mathbf{u}_t$ can be computed with Equation (4.1). A derivation of Equation (4.1) can be found in the Appendix E.

## 4.3 Model uncertainty as additional objective in R-AICO

In AICO, it is possible to define objectives in arbitrary spaces. To incorporate the model uncertainty $\sigma_t$ as an additional objective. To do so, it is necessary to define a mapping $\sigma_t = g(\hat{\mathbf{x}}_t)$ and its Jacobian $J(\hat{\mathbf{x}}_t) = \partial g / \partial \hat{\mathbf{x}}_t$ with $\hat{\mathbf{x}}_t = (\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{u}_t)$. For a new point $\hat{\mathbf{x}}_*$ the function $g$ is given by the predicted variance $\text{var}(\hat{\mathbf{x}}_*)$ and its Jacobian is given by the derivative of the predicted variance

$$\frac{\partial \text{var}(\hat{\mathbf{x}}_*)}{\partial \hat{\mathbf{x}}_*} = -2(\Lambda^{-1} \begin{bmatrix} \hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_* \\ \vdots \\ \hat{\mathbf{x}}_N - \hat{\mathbf{x}}_* \end{bmatrix} \odot k(\mathbf{X}, \hat{\mathbf{x}}_*))^T \left(\mathbf{K} + \sigma_w^2 \mathbf{I}\right)^{-1}. \tag{4.2}$$

The additional objective is incorporated by reformulation $\mathbf{r}_t, \mathbf{R}_t$ with

$$\begin{aligned} \hat{\mathbf{r}}_t &= \mathbf{r}_t + \mathbf{J}^T \mathbf{C}_\sigma \left(\sigma^* - g(\hat{\mathbf{x}}_t)\right), \\ \hat{\mathbf{R}}_t &= \mathbf{R}_t + \mathbf{J}^T \mathbf{C}_\sigma \mathbf{J}. \end{aligned}$$

The same cost function as in the experiments for the light weight robot arm in Equation (3.7) was used with the following extension

$$\hat{C}_t(\mathbf{q}_t, \mathbf{u}_t) = C_t(\mathbf{q}_t, \mathbf{u}_t) + \left(\sigma^* - g(\hat{\mathbf{x}}_t)\right)^T \mathbf{C}_\sigma \left(\sigma^* - g(\hat{\mathbf{x}}_t)\right), \tag{4.3}$$

where the diagonal elemets of the matrix $\mathbf{C}_\sigma$ specify the desired precision and $\sigma_*$ specifies the desired model uncertainty. The tradeoff between exporation and exploitation can be adjust by the desired model uncertainty $\sigma_*$, i.e. AICO tries to match this defined value. This is evaluated on a simple pendulum as toy task in Subsection 4.4. Note that a derivation of Equation (4.2) can be found in the Appendix E.

## 4.4 Results on the simple pendulum task

The simple pendulum has to reach for the end effector target $\mathbf{x}^* = [0, 1]^T$, which represents the upright posture. The desired model uncertainty $\sigma_*$ was evaluated for the values $\{0, 0.2, 1\}$ and without model uncertainty feature. The cost function, which was described in Equation (4.3) was used. The precision matrix was set to $\mathbf{C}_\sigma = \text{diag}(w_1, ..., w_D])$ with diagonal elements $w_i$, which are evaluated for the values $10^{-2}$ and $10^2$. The planning horizon was 300 ms with a time step of 2 ms. Note that we do not assume any initial solution to initialize the planner, solely the initial posture of the pendulum in configuration space was used as initial 'trajectory', which was used for every evaluation. The proposed regularization parameter in Equation (3.8) was set to $\alpha = 0$, i.e. fast convergence and allowing the risk of jumps. Note

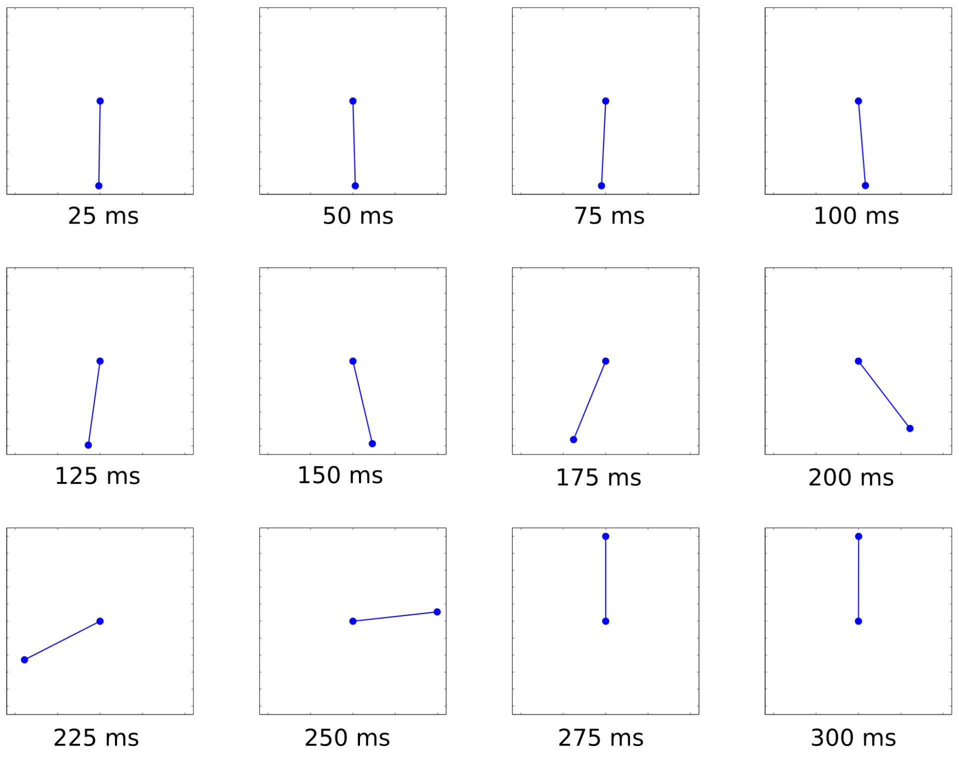


**Figure 4.1.:** Swing-Up of the simple pendulum as toy task, where the best solution was found with model uncertainty feature and $\sigma_* = 0$. The reach the upper stable posture, the algorithm has to plan several swings to the left - and right direction.

that the original implementation of AICO was not able to find an appropriate solution, without specifying additional objectives such as via-points.

In the following preliminary results are discussed, where we interpret results from a simple pendulum planning task. The best solution of the swing-up is illustrated in Figure 4.1, which was obtained with the model uncertainty feature and $\sigma_* = 0$. To reach the upper stable posture, the algorithm has to plan several swings to the left - and right direction. Convergence is guaranteed for all desired model uncertainty values if the diagonal elements of the precision matrix $\mathbf{W}_\sigma$ are small enough, which is shown in Figure 4.2A,B. In Figure 4.2A, the prediciton error of the Gaussian processes is shown and in (B) the root mean squared error (RMSE) of the task performance is shown, i.e. the RMSE of the planned and executed trajectory. The influence of the different desired model uncertainty values can be seen in Figure 4.2B, where for a small $\sigma_* = 0$ the convergence is faster than for a higher $\sigma_* = 1$. As said, allowing the risk of jumps with a regularization parameter $\alpha = 0$ is obtained for no model uncertainty feature, compare Figure 4.2B. The algorithm explores to much if the diagonal elements of the precision matrix $\mathbf{W}_\sigma$ are $10^2$ for $\sigma^* = 1$ and $\sigma^* = 0.1$. However, for small precision weights the planned trajectories, torques and end effector positions converges to the optimal during the learning process and the model prediction error converges to zero. Evaluations on multiple planning tasks are part of ongoing work.

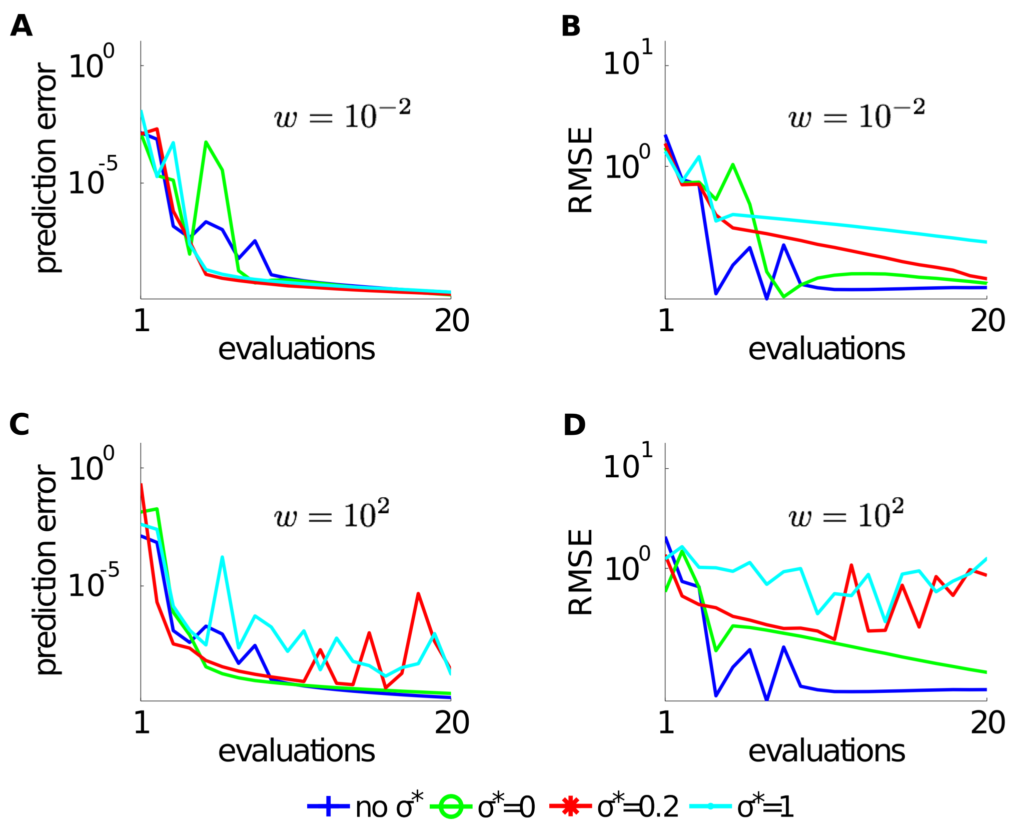


**Figure 4.2.:** Convergence properties of the simple pendulum task. In (A) and (B), the prediction error of the Gaussian processes and the RMSE of the task performance converges for all desired model uncertainty values with the weight $w = 10^{-2}$. In (C) and (D), convergence is guaranteed for the model uncertainty value $\sigma_* = 0$ and if no model uncertainty feature was used. For a high weight and the model uncertainty values $\sigma_* = 1$ and $\sigma_* = 0.2$ the algorithm explores to much.

## 4.5 The computational time issue

Gaussian processes tend to be slow with a large number of training points. To investigate the computational time we used a simulation of the robot platform biorob, which has to reach a end effector target $\mathbf{x}^* = \mathbf{x}_0 + [0.5, 0, 0.5]^T$, where $\mathbf{x}_0$ was the initial posture. The same cost function as in the experiments for the light weight robot arm in Equation (3.7) was used. The planning horizon is 1 second with a time step of 2 ms. Note that we do not assume any initial solution to initialize the planner, solely the initial posture $\mathbf{x}_0$ of the pendulum in configuration space was used as initial 'trajectory'. After a planning evaluation, AICO was initialized with the last planned trajectory. The proposed regularization parameter was $\alpha = 2$.

The resulting planned movement is shown in Figure 4.3A, which was obtained after 20 evaluations. The prediction of the Gaussian processes are depicted in Figure 4.3B for two hyper-parameter optimization procedures, where the optimization was done after each evaluation (denoted by 1:1), or after adding more than 100 hundret data points (denoted by $>$100). The prediction error of the Gaussian processes is for both cases similar. But the computation time differs, which is depicted in Figure 4.3C. The algorithm achieves to reach the desired end effector position, what is shown in Figure 4.3D. Thus, the computational time

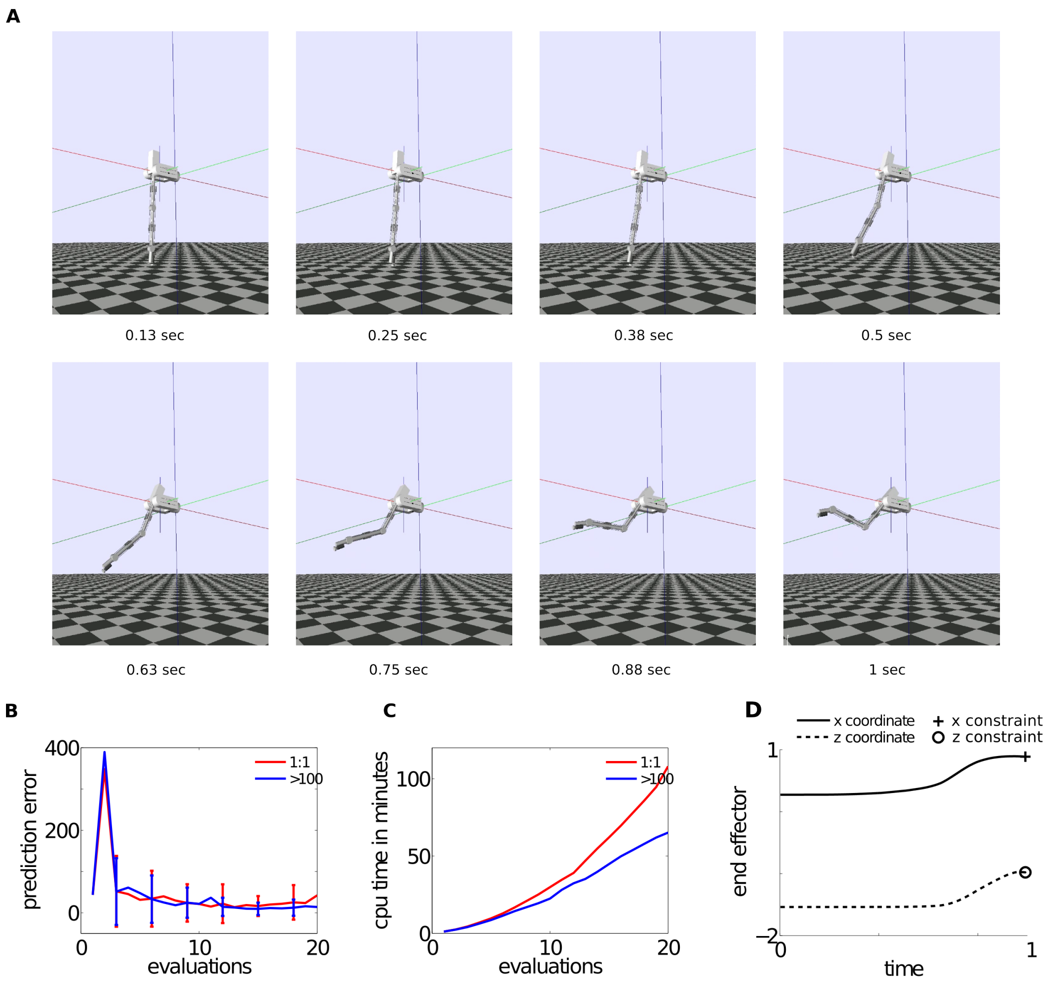


**Figure 4.3.:** Reaching task in the biorob platform. The robot has to reach a desired end effector position. Eight snapshots of the inferred movement are shown in (A). In (B), the convergence of the Gaussian processes prediction is shown, where the hyper-parameter optimization was done after each evaluation (denoted by 1:1) or after adding of more than 100 data points (denoted by >100). In (C), the computation time is shown for the different hyper-parameter optimization procedures. The resulting end effector trajectory including the constraints is depicted in (C).

can be decreased in this experiment using a simple heuristic achieving the same performance of the predicted Gaussian process error.

# 5 Conclusion

Stochastic Optimal Control (SOC) methods are powerful planning methods to infer high-dimensional state and control sequences [28],[26],[29], [20]. For real time applications in humanoids, efficient model predictive control variants have been proposed [26]. However, the quality of the generated solutions heavily depends on the initial movement policy and on the accuracy of the approximations of the system dynamics. Most methods use regularization to prevent numerical instabilities, but typically greedily exploit the approximated system dynamics model. The resulting trajectory update might be far from the previous trajectory used for linearization.

As the linearizations are only locally valid, we explicitly avoid large jumps in the trajectories by punishing large deviations from the previous trajectory. This is achieved by adding an regularization term as an additional objective. We demonstrated in this thesis that SOC methods can greatly benefit from such a regularization term. We used such regularization term for the Approximate Inference Control (AICO) algorithm [29]. Due to the regularization term, which implicitly specifies the step size of the trajectory update, no learning rate as in the standard formulation of AICO is needed. Our experiment shows that the used regularization term considerably outperforms existing SOC methods that are based on linearization, in particular if highly non-linear system dynamics are used.

We also investigate a combination of SOC and model learning. Typically, inaccurate model predictions have catastrophic effects on the numerical stability of SOC methods. In particular, if the model predictions are poor, the SOC method should not further explore but collect more data around the current trajectory. We showed that even with bad model predictions we achieved convergence in multiple objective planning problems. The tradeoff between exploration and exploitation was controlled via an additional task objective that is the distance to the desired model uncertainty. We tested the concept of simultaneous movement planning and model learning on a simple pendulum as toy task, where we achieved stable convergence even under insufficient learned models. The effect of the hyperparameter optimization on the computational time was investigated on a five link robot arm called biorob.

## 5.1 Future work

In future work we would like to test the presented approach on real robots. With the Nao platform, we would like to learn to walk with the center of gravity as input information. On the biorob platform we would like to study the proposed model uncertainty feature. For a high-dimensional robot platform, like the iCub, it is necessary to improve the computation speed and robustness due to the model uncertainty feature.

The assumption of known task objectives is not obvious and to infering the task objective from demonstrations could be future work. This could be done, e.g., with inverse reinforcement learning or with bayesian inverse reinforcement learning methods [1],[4].

The computation time of our proposed simultaneous movement planning with model learning could be reduced, e.g. through parallelization, or faster hyper-parameter optimization procedures, or the approach introduced in [14], where exploration solely based on empirical estimates of the learner's accuracy and learning progress. The idea is that not every data point increases the certainty about the learned model, e.g. if the transition dynamics change over time.

# A List of publications

Rueckert, E; Mindt M.; Peters, J.; Neumann, G. (2014). Robust Policy Updates for Stochastic Optimal Control. Accepted for *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*.

## A.1 Comments and contributions to publications

The paper *Robust Policy Updates for Stochastic Optimal Control* was written by Elmar Rueckert (ER), Gerhard Neumann (GN), and by myself (MM). (ER) developed the basic ideas used in this paper. Together, (ER) and (MM) implemented the algorithms and conducted the experiments. The experiments on the biorob were performed by (ER) while (MM) did the humanoid experiments on the Nao. (JP) contributed in writing. This paper builds the foundation for Section 3.

# B Solving a SOC problem for the LQG Case

The derivation of Equation (3.2) is given by

$$
\begin{aligned}
J_t^{\pi^*}(\mathbf{q}_t) &= \underset{\mathbf{u}_t}{\operatorname{argmin}} \left( C_t(\mathbf{q}_t, \mathbf{u}_t) + \int_{q_{t+1}} P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t) J_{t+1}^{\pi^*}(\mathbf{q}_{t+1}) \, \mathrm{d}\mathbf{q}_{t+1} \right), \\
&= \underset{\mathbf{u}_t}{\operatorname{argmin}} \Big( \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - 2\mathbf{r}_t^T \mathbf{q}_t + \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t + \\
&\int_{q_{t+1}} \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t)(\mathbf{q}_{t+1}^T \mathbf{V}_{t+1}\mathbf{q}_{t+1} - 2\mathbf{v}_{t+1}^T \mathbf{q}_{t+1}) \, \mathrm{d}\mathbf{q}_{t+1} \Big), \\
&= \underset{\mathbf{u}_t}{\operatorname{argmin}} \Big( \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - 2\mathbf{r}_t^T \mathbf{q}_t + \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t + \left(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t\right)^T \mathbf{V}_{t+1} \left(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t\right) \\
&\qquad -2\mathbf{v}_{t+1}^T \left(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t\right) + \mathbf{tr}(\mathbf{V}_{t+1}, \mathbf{Q}_t) \Big), \\
&= \underset{\mathbf{u}_t}{\operatorname{argmin}} \Big( \mathbf{q}_t^T \left(\mathbf{R}_t + \mathbf{A}_t^T \mathbf{V}_{t+1}\mathbf{A}_t\right)\mathbf{q}_t - 2\left(\mathbf{r}_t^T + (\mathbf{v}_{t+1} - \mathbf{V}_{t+1}\mathbf{a}_t)^T \mathbf{A}_t\right)\mathbf{q}_t + \mathbf{u}_t^T \left(\mathbf{H}_t + \mathbf{B}_t^T \mathbf{V}_{t+1}\mathbf{B}_t\right)\mathbf{u}_t \\
&\qquad +2\mathbf{u}_t^T \mathbf{B}_t^T \left(\mathbf{V}_{t+1}(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t) - \mathbf{v}_{t+1}\right) + \mathbf{a}_t^T \mathbf{V}_{t+1}\mathbf{a}_t - 2\mathbf{v}_{t+1}^T \mathbf{a}_t + \mathbf{tr}(\mathbf{V}_{t+1}, \mathbf{Q}_t) \Big).
\end{aligned}
$$

Minimize with respect to $\mathbf{u}_t$ by setting the gradient to zero yields in

$$
\begin{aligned}
0 = \nabla_{\mathbf{u}_t} J_t^{\pi^*}(\mathbf{q}_t) &= 2\left(\mathbf{H}_t + \mathbf{B}_t^T \mathbf{V}_{t+1}\mathbf{B}_t\right)\mathbf{u}_t^* + 2\mathbf{B}_t^T \left(\mathbf{V}_{t+1}(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t) - \mathbf{v}_{t+1}\right), \\
\mathbf{u}_t^* &= -\left(\mathbf{H}_t + \mathbf{B}_t^T \mathbf{V}_{t+1}\mathbf{B}_t\right)^{-1} \mathbf{B}_t^T \left(\mathbf{V}_{t+1}(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t) - \mathbf{v}_{t+1}\right).
\end{aligned}
$$

Note that the optimal control path is independent of $\mathbf{Q}_t$. Insert this result back into the value function results in the Ricatty Equations

$$
\begin{aligned}
J_t^{\pi^*}(\mathbf{q}_t) &= \mathbf{q}_t^T \mathbf{V}_t \mathbf{q}_t - 2\mathbf{v}_t^T \mathbf{q}_t, \\
\mathbf{V}_t &= \mathbf{R}_t + \left(\mathbf{A}_t^T - \mathbf{K}\right)\mathbf{V}_{t+1}\mathbf{A}_t, \\
\mathbf{v}_t &= \mathbf{r}_t + \left(\mathbf{A}_t^T - \mathbf{K}\right)(\mathbf{v}_{t+1} - \mathbf{V}_{t+1}\mathbf{a}_t), \\
\mathbf{K} &= \mathbf{A}_t^T \mathbf{V}_{t+1}\left(\mathbf{V}_{t+1} + \mathbf{B}_t\mathbf{H}_t^{-1}\mathbf{B}_t^T\right)^{-1}.
\end{aligned}
$$

# C Derivation of different pseudo-dynamic models

This section shows different derivations of pseudo-dynamic models. The most common numerical methods for solving an ordinary differential equation (ODE) are inter alia *Eulers method*, *backward Euler method*, and *Heuns method*, which will be discussed in the following subsections. We compared all methods, and could not find any significant difference in terms of accuracy. However, for our experiments we used the backward Euler method, as it is computationally the most efficient.

## C.1 Eulers method

Assume we have a full controlled system and $\mathbf{q}_t = [q_t, \dot{q}_t]^T$. With Eulers method, our state transitions are given by

$$P(q_{t+1}|\dot{q}_t, q_t) = \mathcal{N}(q_{t+1}|q_t + \tau\dot{q}_t, W_1),$$
$$P(\dot{q}_{t+1}|\dot{q}_t, q_t) = \mathcal{N}(\dot{q}_{t+1}|\dot{q}_t + \tau u_t, W_2),$$

where $\tau$ denotes the time step. Then we obtain the following reformulation

$$\mathbf{q}_{t+1} = \mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t,$$
$$\begin{bmatrix} q_{t+1} \\ \dot{q}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} + \begin{bmatrix} 0 \\ \tau \end{bmatrix} u_t.$$

## C.2 Backward Eulers method

Assume we have a full controlled system and $\mathbf{q}_t = [q_t, \dot{q}_t]^T$. With backward Eulers method, our state transitions are given by

$$P(q_{t+1}|\dot{q}_t, q_t) = \mathcal{N}(q_{t+1}|q_t + \tau\dot{q}_{t+1}, W_1),$$
$$P(\dot{q}_{t+1}|\dot{q}_t, q_t) = \mathcal{N}(\dot{q}_{t+1}|\dot{q}_t + \tau u_t, W_2),$$

where $\tau$ denotes the time step. It is possible to transform the state transitions as follows

$$\begin{aligned} P(q_{t+1}|\dot{q}_t, q_t) &= \mathcal{N}(q_{t+1}|q_t + \tau\dot{q}_{t+1}, W_1), \\ P(q_{t+1}|\dot{q}_t, q_t, u_t) &= \mathcal{N}\left(q_{t+1}|q_t + \tau(\dot{q}_t + \tau u_t), W_1\right), \\ P(q_{t+1}|\dot{q}_t, q_t, u_t) &= \mathcal{N}(q_{t+1}|q_t + \tau\dot{q}_t + \tau^2 u_t, W_1). \end{aligned}$$

Then we obtain the following reformulation

$$\begin{aligned} \mathbf{q}_{t+1} &= \mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \\ \begin{bmatrix} q_{t+1} \\ \dot{q}_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}\begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} + \begin{bmatrix} \tau^2 \\ \tau \end{bmatrix} u_t. \end{aligned}$$

## C.3 Heuns method

Assume we have a full controlled system and $\mathbf{q}_t = [q_t, \dot{q}_t]^T$. With Heuns method, our state transitions are given by

$$\begin{aligned} P(q_{t+1}|\dot{q}_t, q_t) &= \mathcal{N}\left(q_{t+1}|q_t + \frac{\tau}{2}(\dot{q}_t + \dot{q}_{t+1}), W_1\right), \\ P(\dot{q}_{t+1}|\dot{q}_t, q_t) &= \mathcal{N}(\dot{q}_{t+1}|\dot{q}_t + \tau u_t, W_2), \end{aligned}$$

where $\tau$ denotes the time step. It is possible to transform the state transitions as follows

$$\begin{aligned} P(q_{t+1}|\dot{q}_{t+1}, \dot{q}_t, q_t) &= \mathcal{N}\left(q_{t+1}|q_t + \frac{\tau}{2}(\dot{q}_t + \dot{q}_{t+1}), W_1\right), \\ P(q_{t+1}|u_t, \dot{q}_t, q_t) &= \mathcal{N}\left(q_{t+1}|q_t + \frac{\tau}{2}(\dot{q}_t + \dot{q}_t + \tau u_t), W_1\right), \\ P(q_{t+1}|u_t, \dot{q}_t, q_t) &= \mathcal{N}\left(q_{t+1}|q_t + \tau\dot{q}_t + \frac{\tau^2}{2}u_t, W_1\right). \end{aligned}$$

Then we obtain the following reformulation

$$\begin{aligned} \mathbf{q}_{t+1} &= \mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \\ \begin{bmatrix} q_{t+1} \\ \dot{q}_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}\begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} + \begin{bmatrix} \frac{\tau^2}{2} \\ \tau \end{bmatrix} u_t. \end{aligned}$$

An interessting issue is that for all proposed methods only the $\mathbf{B}_t$ matrix changes in the upper entries.

# D Derivation of approximate inference control algorithm

For all derivation steps we used the rules in Gaussian identities[1] associated with their numbers. As described, we want to maximize the probability of receiving a reward, Equation (3.3) becomes

$$\begin{aligned}P(\mathbf{z}_t = 1|\mathbf{q}_t, \mathbf{u}_t) &\propto \exp\{-(\mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - 2\mathbf{r}_t^T \mathbf{q}_t + \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t)\},\\ &= \mathcal{N}[\mathbf{q}_t|\mathbf{r}_t, \mathbf{R}_t]\mathcal{N}[\mathbf{u}_t|0, \mathbf{H}_t],\\ &= P(\mathbf{z}_t = 1|\mathbf{q}_t)P(\mathbf{u}_t),\end{aligned}$$

where the reward probability is factorized into a state dependent term and $P(z_t = 1|\mathbf{q}_t) = \mathcal{N}[\mathbf{q}_t|\mathbf{r}_t, \mathbf{R}_t]$ and the action prior $P(\mathbf{u}_t) = \mathcal{N}[\mathbf{u}_t|0, \mathbf{H}_t]$. Before starting with the message parsing we have to marginalize out the controls in the state transition model

$$\begin{aligned}P(\mathbf{q}_{t+1}|\mathbf{q}_t) &= \int_{\mathbf{u}_t} \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{W}_t)\mathcal{N}[\mathbf{u}_t|0, \mathbf{H}_t]d\mathbf{u}_t,\\ &= \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{W}_t + \mathbf{B}_t\mathbf{H}_t^{-1}\mathbf{B}_t^T).\end{aligned}$$

This is an uncontrolled state space model, where we can use massage parsing. But first of all, we sum up our assumptions. Therefore, we have an initial state distribution

$$P(\mathbf{q}_0) = \mathcal{N}(\mathbf{q}_0|\mathbf{a}_0, \mathbf{Q}_0),$$

a state transition model

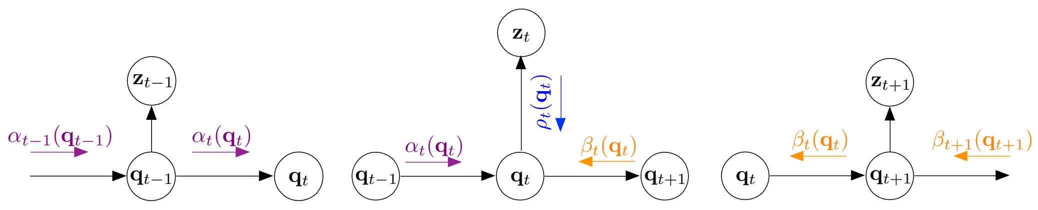
$$P(\mathbf{q}_{t+1}|\mathbf{q}_t) = \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t, \mathbf{Q}_t),$$

and an observation model

$$P(\mathbf{z}_t|\mathbf{q}_t) = \mathcal{N}(\mathbf{z}_t|\mathbf{D}_t\mathbf{q}_t + \mathbf{d}_t, \mathbf{C}_t).$$

To compute the posterior over state and control sequences, conditioning on observing a reward at every time step $P(\mathbf{q}_{0:T}, \mathbf{u}_{0:T-1}|z_{0:T} = 1)$ we can use Bayes' Theorem and it turns out that we need to compute the probability $P(\mathbf{q}_t)$. In Gaussian Network, this probability is the product of the forward message $\alpha_t(\mathbf{q}_t)$, the backward massage $\beta_t(\mathbf{q}_t)$, and the observation message $\rho_t(\mathbf{q}_t)$, which is illustrated in Figure D.1(b). Therefore, we define as forward message $\alpha_t(\mathbf{q}_t) = \mathcal{N}[\mathbf{q}_t|, \mathbf{s}_t, \mathbf{S}_t]$, the backward message as $\beta_t(\mathbf{q}_t) = \mathcal{N}[\mathbf{q}_t|, \mathbf{v}_t, \mathbf{V}_t]$, and the

[1] http://ipvs.informatik.uni-stuttgart.de/mlr/marc/notes/gaussians.pdf



(a) Forward Message (b) Belief (c) Backward Message

**Figure D.1.:** The influence of different Gaussian messages are depicted in this Figure. The forward -, backward -, and observation message are necessary to compute the belief $P(\mathbf{q}_t)$, which is depicted in Figure D.1(b). The observation message is given by the cost function. Figure D.1(a) shows the dependencies of the forward message, which are the previous forward message and the corresponding dependencies of the state transition model, namely $P(\mathbf{q}_t|\mathbf{q}_{t-1})$ and $P(\mathbf{z}_{t-1}|\mathbf{q}_{t-1})$. Figure D.1(c) shows the similar dependencies of the backward message.

observation message $\rho_t(\mathbf{q}_t) = \mathcal{N}[\mathbf{q}_t|\mathbf{r}_t, \mathbf{R}_t]$, which are all Gaussians. In the next subsections describe the derivations of these messages. We end up with,

$$\begin{aligned}
P(\mathbf{q}_t) = \mathcal{N}[\mathbf{q}_t|\mathbf{b}_t, \mathbf{B}_t] &= \mathcal{N}[\mathbf{q}_t|\mathbf{s}_t, \mathbf{S}_t]\mathcal{N}[\mathbf{q}_t|\mathbf{r}_t, \mathbf{R}_t]\mathcal{N}[\mathbf{q}_t|\mathbf{v}_t, \mathbf{V}_t],\\
&\overset{(21)}{\propto} \mathcal{N}(\mathbf{q}_t|\mathbf{s}_t + \mathbf{r}_t + \mathbf{v}_t, \mathbf{S}_t + \mathbf{R}_t + \mathbf{V}_t),\\
&= \mathcal{N}[\mathbf{q}_t|\mathbf{b}_t, \mathbf{B}_t],
\end{aligned}$$

where $\mathbf{B}_t = (\mathbf{S}_t + \mathbf{R}_t + \mathbf{V}_t)^{-1}$ and $\mathbf{b}_t = \mathbf{B}_t(\mathbf{s}_t + \mathbf{r}_t + \mathbf{v}_t)$. We can also compute the optimal feedback controller, which can be done by the conditional distribution $P(\mathbf{u}_t|\mathbf{q}_t) = P(\mathbf{u}_t, \mathbf{q}_t)/P(\mathbf{q}_t)$. Hence, we need to compute the joint state-control posterior

$$\begin{aligned}
P(\mathbf{u}_t, \mathbf{q}_t) &= P(\mathbf{u}_t, \mathbf{q}_t),\\
&= \int_{\mathbf{q}_{t+1}} \alpha_t(\mathbf{q}_t)\rho_t(\mathbf{q}_t)P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t)P(\mathbf{u}_t)\beta_{t+1}(\mathbf{q}_{t+1})\rho_{t+1}(\mathbf{q}_{t+1})d\mathbf{q}_{t+1},\\
&= P(\mathbf{q}_t)P(\mathbf{u}_t)\int_{\mathbf{q}_{t+1}} P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{u}_t)\mathcal{N}[\mathbf{q}_{t+1}|\bar{\mathbf{v}}_{t+1}, \bar{\mathbf{V}}_{t+1}]d\mathbf{q}_{t+1}.
\end{aligned}$$

The solution of the conditional distribution is

$$P(\mathbf{u}_t|\mathbf{q}_t) = \mathcal{N}[\mathbf{q}_t|\mathbf{M}_t^{-1}\left(\mathbf{B}_t^T\mathbf{V}_*\left(\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1} - \mathbf{A}_t\mathbf{q}_t - \mathbf{a}_t\right) + \mathbf{h}_t\right), \mathbf{M}_t^{-1}],$$

where $\mathbf{V}_* = (\mathbf{Q}_t + \bar{\mathbf{V}}_{t+1}^{-1})^{-1}$ and $\mathbf{M}_t = \mathbf{B}_t^T\mathbf{V}_*\mathbf{B}_t + \mathbf{H}_t$. This can be reformulate to a time-varying linear feedback controller

$$\mathbf{u}_t = \mathbf{o}_t + \mathbf{O}_t\mathbf{q}_t,$$

where the gains are

$$\begin{aligned}
\mathbf{o}_t &= \mathbf{M}_t^{-1}\left(\mathbf{B}_t^T\mathbf{V}_*\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1} - \mathbf{B}_t^T\mathbf{V}_*\mathbf{a}_t + \mathbf{h}_t\right),\\
\mathbf{O}_t &= -\mathbf{M}_t^{-1}\mathbf{B}_t^T\mathbf{V}_*\mathbf{A}_t.
\end{aligned}$$

## D.1 Update equations for forward message parsing

The dependencies to compute the forward message $\alpha_t(\mathbf{q}_t)$ are depicted in Figure D.1(a), which are the dependency of the previous state $P(\mathbf{q}_t|\mathbf{q}_{t-1})$, the corresponding task dependency $P(\mathbf{z}_{t-1}|\mathbf{q}_{t-1})$, and the the forward message $\alpha_{t-1}(\mathbf{q}_{t-1})$ of state $\mathbf{q}_{t-1}$. Integrating out the previous state $\mathbf{q}_{t-1}$ results in

$$
\begin{aligned}
\alpha_t(\mathbf{q}_t) &= \int_{\mathbf{q}_{t-1}} P(\mathbf{q}_t|\mathbf{q}_{t-1})P(\mathbf{z}_{t-1}|\mathbf{q}_{t-1})\alpha_{t-1}(\mathbf{q}_{t-1})d\mathbf{q}_{t-1},\\
&\overset{Def.}{=} \int_{\mathbf{q}_{t-1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1},\mathbf{Q}_{t-1}\right)\mathcal{N}\left(\mathbf{z}_{t-1}|\mathbf{D}_{t-1}\mathbf{q}_{t-1}+\mathbf{d}_{t-1},\mathbf{C}_{t-1}\right)\\
&\mathcal{N}\left[\mathbf{q}_{t-1}|\mathbf{s}_{t-1},\mathbf{S}_{t-1}\right]d\mathbf{q}_{t-1},\\
&= \int_{\mathbf{q}_{t-1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1},\mathbf{Q}_{t-1}\right)\mathcal{N}\left(\mathbf{D}_{t-1}\mathbf{q}_{t-1}|\mathbf{z}_{t-1}-\mathbf{d}_{t-1},\mathbf{C}_{t-1}\right)\\
&\mathcal{N}\left[\mathbf{q}_{t-1}|\mathbf{s}_{t-1},\mathbf{S}_{t-1}\right]d\mathbf{q}_{t-1},\\
&\overset{(35)}{\propto} \int_{\mathbf{q}_{t-1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1},\mathbf{Q}_{t-1}\right)\\
&\mathcal{N}\left[\mathbf{q}_{t-1}|\underbrace{\mathbf{D}_{t-1}^T\mathbf{C}_{t-1}^{-1}\left(\mathbf{z}_{t-1}-\mathbf{d}_{t-1}\right)}_{=\mathbf{r}_{t-1}},\underbrace{\mathbf{D}_{t-1}^T\mathbf{C}_{t-1}^{-1}\mathbf{D}_{t-1}}_{=\mathbf{R}_{t-1}}\right]\mathcal{N}\left[\mathbf{q}_{t-1}|\mathbf{s}_{t-1},\mathbf{S}_{t-1}\right]d\mathbf{q}_{t-1},\\
&\overset{(21)}{\propto} \int_{\mathbf{q}_{t-1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1},\mathbf{Q}_{t-1}\right)\mathcal{N}\left[\mathbf{q}_{t-1}|\underbrace{\mathbf{s}_{t-1}+\mathbf{r}_{t-1}}_{=\bar{\mathbf{r}}_{t-1}},\underbrace{\mathbf{S}_{t-1}+\mathbf{R}_{t-1}}_{=\bar{\mathbf{R}}_{t-1}}\right]d\mathbf{q}_{t-1},\\
&= \int_{\mathbf{q}_{t-1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1},\mathbf{Q}_{t-1}\right)\mathcal{N}\left[\mathbf{q}_{t-1}|\bar{\mathbf{s}}_{t-1},\bar{\mathbf{S}}_{t-1}\right]d\mathbf{q}_{t-1},\\
&\overset{(3)}{=} \int_{\mathbf{q}_{t-1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1},\mathbf{Q}_{t-1}\right)\mathcal{N}\left(\mathbf{q}_{t-1}|\bar{\mathbf{S}}_{t-1}^{-1}\bar{\mathbf{s}}_{t-1}\bar{\mathbf{S}}_{t-1}^{-1}\right)d\mathbf{q}_{t-1},\\
&\overset{(37)}{=} \mathcal{N}\left(\mathbf{q}_{t-1}|\mathbf{a}_{t-1}+\mathbf{A}_{t-1}\bar{\mathbf{S}}_{t-1}^{-1}\bar{\mathbf{s}}_{t-1},\mathbf{Q}_{t-1}+\mathbf{A}_{t-1}\bar{\mathbf{S}}_{t-1}^{-1}\mathbf{A}_{t-1}^T\right),\\
&\overset{(4)}{=} \mathcal{N}\left[\mathbf{q}_{t-1}|\mathbf{s}_t,\mathbf{S}_t\right],
\end{aligned}
$$

with $\mathbf{S}_t = \left(\mathbf{Q}_{t-1}+\mathbf{A}_{t-1}\bar{\mathbf{S}}_{t-1}^{-1}\mathbf{A}_{t-1}^T\right)^{-1}$ and $\mathbf{s}_t = \mathbf{S}_t\left(\mathbf{a}_{t-1}+\mathbf{A}_{t-1}\bar{\mathbf{S}}_{t-1}^{-1}\bar{\mathbf{s}}_{t-1}^T\right)^{-1}$. Reformulating $\mathbf{S}_t$ and $\mathbf{s}_t$ to be numerical stable leads to the AICO forward messages

$$
\begin{aligned}
\mathbf{S}_t &= \left(\mathbf{A}_{t-1}^{-T}-\mathbf{K}_s\right)\bar{\mathbf{S}}_{t-1}\mathbf{A}_{t-1}^{-1},\\
\mathbf{s}_t &= \left(\mathbf{A}_{t-1}^{-T}-\mathbf{K}_s\right)\left(\bar{\mathbf{s}}_{t-1}+\bar{\mathbf{S}}_{t-1}\mathbf{A}_{t-1}^{-1}\mathbf{a}_{t-1}\right)^{-1},\\
\mathbf{K}_s &= \mathbf{A}_{t-1}^{-T}\bar{\mathbf{S}}_{t-1}\left(\bar{\mathbf{S}}_{t-1}+\bar{\mathbf{A}}_{t-1}^{-T}\mathbf{Q}_{t-1}\mathbf{A}_{t-1}^{-1}\right)^{-1}.
\end{aligned}
$$

## D.2 Update equations for backward message parsing

The dependencies to compute the backward message $\beta_t(\mathbf{q}_t)$ are depicted in Figure D.1(c), which are the dependency of the next state $P(\mathbf{q}_{t+1}|\mathbf{q}_t)$, the corresponding task dependency $P(\mathbf{z}_{t+1}|\mathbf{q}_{t+1})$, and the the backward message $\beta_{t+1}(\mathbf{q}_{t+1})$ of state $\mathbf{q}_{t+1}$. Integrating out the next state $\mathbf{q}_{t-1}$ results in

$$\beta_t(\mathbf{q}_t) = \int_{\mathbf{q}_{t+1}} P(\mathbf{q}_{t+1}|\mathbf{q}_t)P(\mathbf{z}_{t+1}|\mathbf{q}_{t+1})\beta_{t+1}(\mathbf{q}_{t+1})\mathrm{d}\mathbf{q}_{t+1},$$

where we can use the same first rules as in the derivation of the forward message and begin with

$$\begin{aligned}
&\overset{compareD.1}{\propto} \int_{\mathbf{q}_{t+1}} \mathcal{N}\left(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t\mathbf{Q}_t\right)\mathcal{N}\left[\mathbf{q}_{t+1}|\underbrace{\mathbf{D}_{t+1}^T\mathbf{C}_{t+1}^{-1}(\mathbf{z}_{t+1}-\mathbf{d}_{t+1})}_{=\mathbf{r}_{t+1}}\underbrace{\mathbf{D}_{t+1}^T\mathbf{C}_{t+1}^{-1}\mathbf{D}_{t+1}}_{=\mathbf{R}_{t+1}}\right]\\
&\mathcal{N}\left[\mathbf{q}_{t+1}|\mathbf{v}_{t+1},\mathbf{V}_{t+1}\right]\mathrm{d}\mathbf{q}_{t+1},\\
&\overset{(21)}{\propto} \int_{\mathbf{q}_{t+1}} \mathcal{N}\left(\mathbf{A}_t\mathbf{q}_t|\mathbf{q}_{t+1}-\mathbf{a}_t,\mathbf{Q}_t\right)\mathcal{N}\left[\mathbf{q}_{t+1}|\underbrace{\mathbf{r}_{t+1}+\mathbf{v}_{t+1}}_{=\bar{\mathbf{v}}_{t+1}},\underbrace{\mathbf{R}_{t+1}+\mathbf{V}_{t+1}}_{=\bar{\mathbf{V}}_{t+1}}\right]\mathrm{d}\mathbf{q}_{t+1},\\
&\overset{(35)}{\propto} \int_{\mathbf{q}_{t+1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t),\mathbf{A}_t^{-1}\mathbf{Q}_t\mathbf{A}_t\right)\mathcal{N}\left[\mathbf{q}_{t+1}|\bar{\mathbf{v}}_{t+1},\bar{\mathbf{V}}_{t+1}\right]\mathrm{d}\mathbf{q}_{t+1},\\
&\overset{(3)}{=} \int_{\mathbf{q}_{t+1}} \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t),\mathbf{A}_t^{-1}\mathbf{Q}_t\mathbf{A}_t\right)\mathcal{N}\left(\mathbf{q}_{t+1}|\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1}\bar{\mathbf{V}}_{t+1}^{-1}\right)\mathrm{d}\mathbf{q}_{t+1},\\
&\overset{(37)}{=} \mathcal{N}\left(\mathbf{q}_t|-\mathbf{A}_t^{-1}\mathbf{a}_t+\mathbf{A}_t^{-1}\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1},\mathbf{A}_t^{-1}\mathbf{Q}_t\mathbf{A}_t+\mathbf{A}_t^{-1}\bar{\mathbf{V}}_{t+1}^{-1}\mathbf{A}_t^{-T}\right),\\
&= \mathcal{N}\left(\mathbf{q}_t|\mathbf{A}_t^{-1}\left(\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1}-\mathbf{a}_t\right),\mathbf{A}_t^{-1}\left(\mathbf{Q}_t+\bar{\mathbf{V}}_{t+1}^{-1}\right)\mathbf{A}_t^{-T}\right),\\
&\overset{(4)}{=} \mathcal{N}\left[\mathbf{q}_t|\mathbf{v}_t,\mathbf{V}_t\right],
\end{aligned}$$

with $\mathbf{V}_t = \left(\mathbf{A}_t^{-1}\left(\mathbf{Q}_t+\bar{\mathbf{V}}_{t+1}^{-1}\right)\mathbf{A}_t^{-T}\right)^{-1}$ and $\mathbf{v}_t = \mathbf{V}_t\mathbf{A}_t^{-1}\left(\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1}-\mathbf{a}_t\right)$. Reformulating $\mathbf{V}_t$ and $\mathbf{v}_t$ to be numerical stable leads to the AICO backward messages

$$\begin{aligned}
\mathbf{V}_t &= \left(\mathbf{A}_t^T-\mathbf{K}_v\right)\bar{\mathbf{V}}_{t+1}\mathbf{A}_t,\\
\mathbf{v}_t &= \left(\mathbf{A}_t^T-\mathbf{K}_v\right)\left(\bar{\mathbf{v}}_{t+1}-\bar{\mathbf{V}}_{t+1}\mathbf{a}_t\right),\\
\mathbf{K}_v &= \mathbf{A}_t^T\bar{\mathbf{V}}_{t+1}\left(\bar{\mathbf{V}}_{t+1}+\mathbf{Q}_t\right)^{-1}.
\end{aligned}$$

# E Gradient derivation of the predicted mean and the variance of the Gaussian process

The mean and the variance of the GP prediction are

$$\begin{aligned}\mathrm{m}(\mathbf{x}_*) &= k(\mathbf{X},\mathbf{x}_*)^T \left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}\mathbf{y},\\ \mathrm{var}(\mathbf{x}_*) &= k(\mathbf{x}_*,\mathbf{x}_*) - k(\mathbf{X},\mathbf{x}_*)^T\left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1} k(\mathbf{X},\mathbf{x}_*).\end{aligned}$$

For the derivations it is helpful to derivate first of all the squared exponential kernel

$$k(\mathbf{x}_p,\mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p-\mathbf{x}_q)^T\Lambda^{-1}(\mathbf{x}_p-\mathbf{x}_q)\right),$$

for both input arguments $\mathbf{x}_p$

$$\begin{aligned}\frac{\partial k(\mathbf{x}_p,\mathbf{x}_q)}{\partial \mathbf{x}_p} &= \frac{\partial}{\partial \mathbf{x}_p}\left(\sigma_f^2\exp\left(-\frac{1}{2}(\mathbf{x}_p-\mathbf{x}_q)^T\Lambda^{-1}(\mathbf{x}_p-\mathbf{x}_q)\right)\right),\\ &= \frac{\partial}{\partial \mathbf{x}_p}\left(-\frac{1}{2}(\mathbf{x}_p-\mathbf{x}_q)^T\Lambda^{-1}(\mathbf{x}_p-\mathbf{x}_q)\right)k(\mathbf{x}_p,\mathbf{x}_q),\\ &= -\frac{1}{2}\frac{\partial}{\partial \mathbf{x}_p}\left(\mathbf{x}_p^T\Lambda^{-1}\mathbf{x}_p - \mathbf{x}_q^T\Lambda^{-1}\mathbf{x}_p - \mathbf{x}_p^T\Lambda^{-1}\mathbf{x}_q + \mathbf{x}_q^T\Lambda^{-1}\mathbf{x}_q\right)k(\mathbf{x}_p,\mathbf{x}_q),\\ &= -\frac{1}{2}\left(2\Lambda^{-1}\mathbf{x}_p - 2\Lambda^{-1}\mathbf{x}_q\right)k(\mathbf{x}_p,\mathbf{x}_q),\\ &= -\Lambda^{-1}\left(\mathbf{x}_p-\mathbf{x}_q\right)k(\mathbf{x}_p,\mathbf{x}_q),\end{aligned}$$

and $\mathbf{x}_q$

$$\begin{aligned}\frac{\partial k(\mathbf{x}_p,\mathbf{x}_q)}{\partial \mathbf{x}_q} &= \frac{\partial}{\partial \mathbf{x}_q}\left(\sigma_f^2\exp\left(-\frac{1}{2}(\mathbf{x}_p-\mathbf{x}_q)^T\Lambda^{-1}(\mathbf{x}_p-\mathbf{x}_q)\right)\right),\\ &= \frac{\partial}{\partial \mathbf{x}_q}\left(-\frac{1}{2}(\mathbf{x}_p-\mathbf{x}_q)^T\Lambda^{-1}(\mathbf{x}_p-\mathbf{x}_q)\right)k(\mathbf{x}_p,\mathbf{x}_q),\\ &= -\frac{1}{2}\frac{\partial}{\partial \mathbf{x}_q}\left(\mathbf{x}_p^T\Lambda^{-1}\mathbf{x}_p - \mathbf{x}_q^T\Lambda^{-1}\mathbf{x}_p - \mathbf{x}_p^T\Lambda^{-1}\mathbf{x}_q + \mathbf{x}_q^T\Lambda^{-1}\mathbf{x}_q\right)k(\mathbf{x}_p,\mathbf{x}_q),\\ &= -\frac{1}{2}\left(-2\Lambda^{-1}\mathbf{x}_p + 2\Lambda^{-1}\mathbf{x}_q\right)k(\mathbf{x}_p,\mathbf{x}_q),\\ &= \Lambda^{-1}\left(\mathbf{x}_p-\mathbf{x}_q\right)k(\mathbf{x}_p,\mathbf{x}_q),\end{aligned}$$

which are nearly similar except the minus. Now the derivation of the mean and the variance becomes easier. We start with the derivation of the mean

$$
\begin{aligned}
\frac{\partial \mathrm{m}(\mathbf{x}_*)}{\partial \mathbf{x}_*} &= \frac{\partial}{\partial \mathbf{x}_*}\left(k(\mathbf{X},\mathbf{x}_*)^T \left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}\mathbf{y}\right),\\
&= \frac{\partial}{\partial \mathbf{x}_*}\left(k(\mathbf{X},\mathbf{x}_*)^T\right)\left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}\mathbf{y},\\
&= (\Lambda^{-1}\begin{bmatrix}\mathbf{x}_1-\mathbf{x}_*\\ \vdots \\ \mathbf{x}_N-\mathbf{x}_*\end{bmatrix} \odot k(\mathbf{X},\mathbf{x}_*))^T\left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}\mathbf{y},
\end{aligned}
$$

where $\odot$ is a pointwise product. The derivation of the variance is given by

$$
\begin{aligned}
\frac{\partial \mathrm{var}(\mathbf{x}_*)}{\partial \mathbf{x}_*} &= \frac{\partial}{\partial \mathbf{x}_*}\left(k(\mathbf{x}_*,\mathbf{x}_*)-k(\mathbf{X},\mathbf{x}_*)^T \left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}k(\mathbf{X},\mathbf{x}_*)\right),\\
&= -\frac{\partial}{\partial \mathbf{x}_*}\left(k(\mathbf{X},\mathbf{x}_*)^T\left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}k(\mathbf{X},\mathbf{x}_*)\right),\\
&= -2(\Lambda^{-1}\begin{bmatrix}\mathbf{x}_1-\mathbf{x}_*\\ \vdots \\ \mathbf{x}_N-\mathbf{x}_*\end{bmatrix} \odot k(\mathbf{X},\mathbf{x}_*))^T\left(\mathbf{K}+\sigma_w^2\mathbf{I}\right)^{-1}.
\end{aligned}
$$

# List of Figures

# Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1, 2004.

[2] Paolo Baerlocher and Ronan Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *IROS*, pages 13–17, 1998.

[3] Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends in cognitive sciences*, pages 485–488, 2012.

[4] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, pages 182–189, 2011.

[5] Su Il Choi and Byung Kook Kim. Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, pages 143–151, 2000.

[6] Marc Deisenroth, Dieter Fox, and Charles Rasmussen. Gaussian processes for data-efficient learning in robotics and control. 2013.

[7] Thomas Furmston and David Barber. Variational methods for reinforcement learning. In *AISTATS*, pages 241–248, 2010.

[8] Michael Gienger, Herbert Janssen, and Christian Goerick. Task-oriented whole body motion for humanoid robots. In *IEEE-RAS*, pages 238–244, 2005.

[9] Matthew Hoffman, Nando Freitas, Arnaud Doucet, and Jan Peters. An expectation maximization algorithm for continuous markov decision processes with arbitrary reward. In *AISTATS*, pages 232–239, 2009.

[10] Bert Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. 2007.

[11] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *JMLR*, pages 159–182, 2012.

[12] James Kuffner and Steven LaValle. Rrt-connect: An efficient approach to single-query path planning. In *ICRA*, pages 995–1001, 2000.

[13] Thomas Lens, Jürgen Kunz, Oskar von Stryk, Christian Trommer, and Andreas Karguth. Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications. In *ISR/ROBOTIK*, pages 1–6, 2010.

[14] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *NIPS*, pages 206–214, 2012.

[15] Thomas Minka. A family of algorithms for approximate bayesian inference. 2001.

[16] Kevin Murphy. Dynamic bayesian networks: representation, inference and learning. 2003.

[17] Koichi Nishiwaki, Mamoru Kuga, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Whole-body cooperative balanced motion generation for reaching. *IJHR*, pages 437–457, 2005.

[18] Jan Peters and Stefan Schaal. Learning operational space control. In *Robotics: Science and Systems*, 2006.

[19] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *ICML*, pages 745–750, 2007.

[20] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. An approximate inference approach to temporal optimization in optimal control. In *NIPS*, pages 2011–2019, 2010.

[21] Elmar Rückert and Gerhard Neumann. A study of morphological computation by using probabilistic inference for motor planning. 2011.

[22] Elmar Rückert and Gerhard Neumann. Stochastic optimal control methods for investigating the power of morphological computation. *Artificial Life*, pages 115–131, 2013.

[23] Jeff Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. *NIPS*, 1997.

[24] Robert Stengel. *Stochastic optimal control: theory and application*. John Wiley & Sons, Inc., 1986.

[25] Tomomichi Sugihara and Yoshihiko Nakamura. Whole-body cooperative balancing of humanoid robot using cog jacobian. In *IROS*, pages 2575–2580, 2002.

[26] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IROS*, pages 4906–4913, 2012.

[27] Emanuel Todorov. General duality between optimal control and estimation. In *IEEE*, pages 4286–4292, 2008.

[28] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC*, pages 300–306, 2005.

[29] Marc Toussaint. Robot trajectory optimization using approximate inference. In *ICML*, pages 1049–1056, 2009.

[30] Marc Toussaint and Christian Goerick. Probabilistic inference for structured planning in robotics. In *IROS*, pages 3068–3073, 2007.

[31] Marc Toussaint and Christian Goerick. A bayesian view on motor control and planning. In *From Motor Learning to Interaction Learning in Robots*, pages 227–252. Springer, 2010.

[32] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *ICML*, pages 945–952, 2006.

[33] Marc Toussaint, Nils Plath, Tobias Lang, and Nikolay Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *ICRA*, pages 385–391, 2010.

[34] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *IJRR*, pages 1263–1278, 2012.

[35] Sethu Vijayakumar, Marc Toussaint, Giorgio Petkos, and Matthew Howard. Planning and moving in dynamic environments. In *Creating Brain-Like Intelligence*. Springer, 2009.

[36] Jonathan Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, pages 236–239, 2003.