Curious iLQR: Resolving Uncertainty in Model-based RL

Sarah Bechtle[1], Akshara Rai[2], Yixin Lin[2], Ludovic Righetti[1,3] and Franziska Meier[2]

***Abstract*— Curiosity as a means to explore during reinforcement learning problems has recently become very popular. However, very little progress has been made in utilizing curiosity for learning control. In this work, we propose a model-based reinforcement learning (MBRL) framework that combines Bayesian modeling of the system dynamics with *curious iLQR*, a risk-seeking iterative LQR approach. During trajectory optimization the *curious iLQR* attempts to minimize both the task-dependent cost and the uncertainty in the dynamics model. We scale this approach to perform reaching tasks on 7-DoF manipulators, to perform both simulation and real robot reaching experiments. Our experiments consistently show that MBRL with *curious iLQR* more easily overcomes bad initial dynamics models and reaches desired joint configurations more reliably and with less system rollouts.**

I. Introduction

Curiosity has repeatedly been recognized as a fundamental building block of human behaviour [13]. Some researchers go as far as considering curiosity essential for the development of autonomous behaviour in humans [23]. The concept of intrinsically motivated, curious, behavior has also been explored within the reinforcement learning literature from various angles. For example, a first attempt towards intrinsically motivated agents consisted in rewarding agents to minimize prediction errors of sensory events [2], [19], [20]. This initial work on curiosity-driven agents was designed for low-dimensional and discrete state-and-action spaces. Recently, curiosity as a means to better explore was also investigated for high-dimensional continuous state spaces [3], [15]. Most of this work, including recent efforts towards curiosity driven robot learning [12], [21], has defined curiosity as a function of model prediction error.

In this work, we take inspiration from Kagan et al. [10], who define curiosity as motivation to resolve uncertainty in a model. Following this definition, our goal is to develop a model-based reinforcement learning (MBRL) algorithm that is aware of its model uncertainty, and optimizes action sequences to not only maximize a reward but also to reduce model uncertainty. MBRL comes with a lot of promise for sample-efficient learning on real robots [1]. However, one of the key challenges centers around making use of a bad model to generate useful data to improve the model and eventually achieve the task. We believe, that curiosity can help with this challenge. Specifically, in this work, our goal is to learn motor control tasks via MBRL, and thus the model represents the system dynamics. Our MBRL algorithm uses a Gaussian process [24] to represent the dynamics and a *curious* version

[1]Max Planck Institute for Intelligent Systems
[2]Facebook AI Research
[3]New York University

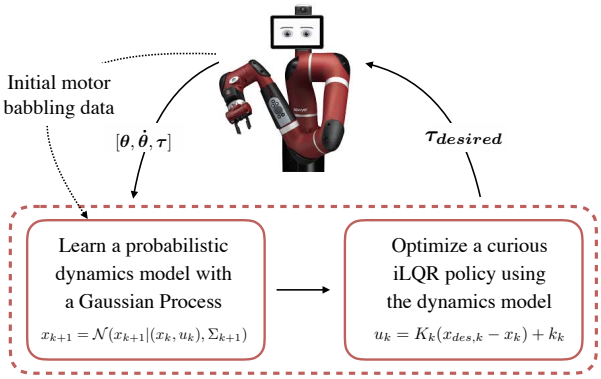


Fig. 1: An overview of our approach for model-based reinforcement learning. We start with motor babbling data to initialize the dynamics model, followed by an iterative loop of learning the model and updating an iLQR policy.

of the iterative Linear Quadratic Regulator (iLQR) [22] for trajectory optimization, as summarized in Figure 1. In a nutshell, our curious iLQR aims at optimizing trajectories that minimize the cost as well as the uncertainty in the dynamics model. Several model-based RL algorithms that take uncertainty into account have already been proposed [5], [6], [25]. They differ in their choice of policy optimization, dynamics model representation and how they incorporate uncertainty.

While [5], [6] utilize model uncertainty to generate trajectory distributions, the uncertainty does not play an explicit role in the cost. Because of this, these approaches do not explicitly optimize actions that resolve uncertainty in the current model of the dynamics. In order to generate higher rewards for actions that explore the uncertain parts of the dynamics model, we seek to include the variance of the models prediction into the reward function evaluation. To achieve this, our work leverages results on risk-sensitive optimal control [7], [9]. Specifically, [9] showed that optimizing actions with respect to the expected exponentiated cost directly takes into account higher order moments of the cost distribution while affording a computation of the optimal control through simple Ricatti equations. These results were extended to the nonlinear case in [7] which proposes a risk-sensitive version of iLQR which has the advantage of computational simplicity. The risk-sensitive control approach [7], [9] was presented on the premise that we want to avoid high variance states when we have a model of the system dynamics, and process noise. Here, we show that in the model-based RL setting, where we aim to learn a dynamics model from scratch, it is in fact beneficial to seek out actions which create state-action pairs for which the current dynamics model is

uncertain. More concretely, the contributions of this work are as follows: We combine risk-seeking – curious – iLQR with Gaussian process regression into a model-based RL loop that learns a model of the system dynamics from scratch, while trying to achieve a task. We demonstrate that such optimal control algorithms can be scaled to seven Degree of Freedom (DoF) manipulation platforms. Finally, we present a thorough analysis on utilizing curiosity for MBRL via optimal control, and show that curiosity helps to achieve the task faster, and more reliably, when starting from a bad model of the dynamics.

II. Background: Uncertainty in Optimal Control

In this section we review the fundamentals on which our work is built on. An approach enabling the inclusion of higher order statistics in the performance measure while keeping computations tractable, at least in the linear case, is to use exponential costs, as introduced by Jacobson [9]. [7] extended this work by deriving an iterative algorithm for continuous-time stochastic nonlinear optimal control problems called iterative Linear Exponential-Quadratic Optimal Control under Gaussian Process Noise (iLEG). Ponton et al. [16] extended the work from [7] to cases where not only process noise is present but also measurement noise has to be taken into consideration. Next, we briefly present the details of the risk-sensitive iLQR algorithm, following [7], [16] and [9].

A. Risk-sensitive iLQR

To include stochastic processes when optimizing a trajectory, it is necessary to consider a nonlinear optimal control problem where the system dynamics are defined by the following stochastic differential equation

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{f}\left(\mathbf{x_k}, \mathbf{u_k}\right)\Delta t + \mathbf{g}\left(\mathbf{x_k}, \mathbf{u_k}\right)\Delta\omega \tag{1}$$

where $\mathbf{f}$ represents the dynamics of the system and $\mathbf{g}$ the stochasticity of the problem. $\Delta\omega$ is a Brownian motion with zero mean and covariance $(\mathbf{\Sigma}\cdot\Delta t)$. Following the idea of [9] to include higher order momenta of the cost function, the objective function takes the form of an exponential transformation of the performance criteria J:

$$J = \min_{\pi} \mathbb{E}\left\{\exp[\sigma\mathcal{J}\left(\pi\right)]\right\} \tag{2}$$

where $\mathcal{J}(\pi)$ is the performance index, which is a random variable, and a functional of the policy π. $\mathbb{E}\,\mathbb{E}$ is the expected value of $\mathcal{J}$ over stochastic trajectories induced by the policy π. $\sigma \in \mathbb{R}$ accounts for the sensitivity of the cost to higher order moments (variance, skewness, etc). Notably from [7], the cost is

$$\frac{1}{\sigma}\log(J) = \mathbb{E}(\mathcal{J}^*) + \frac{\sigma}{2}\mathrm{var}(\mathcal{J}^*) + \frac{\sigma^2}{6}\mathrm{sk}(\mathcal{J}^*) + \cdots \tag{3}$$

where var and sk stand for variance and skewness and $\mathcal{J}^*$ is the optimal task cost. When $\sigma > 0$ the optimal control will be risk-averse, favoring low costs with low variance but when $\sigma < 0$ the optimal control will be risk-seeking, favoring low costs with high variance. $\sigma = 0$ reduces to the standard, risk-neutral, optimal control problem. We will exploit this property to create policies that explore regions with high uncertainty in the next sections.

Let the performance criteria $\mathcal{J}$ be

$$\mathcal{J} = \Phi_f\left(\mathbf{x_T}\right) + \sum_{\mathbf{k=0}}^{\mathbf{T}} L(\mathbf{x_k}, \mathbf{u_k}) \tag{4}$$

where

$$L(\mathbf{x_k}, \mathbf{u_k}) = \Phi(\mathbf{x_k}) + \frac{1}{2}\mathbf{u_k^T}\mathbf{R}(\mathbf{x_k})\mathbf{u_k} + \mathbf{u_k^T}\mathbf{r}(\mathbf{x_k}) \tag{5}$$

L is composed of the state cost $\Phi(\mathbf{x_k})$ and a quadratic control cost. $\Phi_f\left(\mathbf{x_T}\right)$ is the cost at the final state.

B. Algorithm derivation

The algorithm begins with a nominal state and control input trajectory $\mathbf{x^n}$ and $\mathbf{u^n}$. The dynamics are linearized and the cost is quadratized along $\mathbf{u_k^n}$, $\mathbf{x_k^n}$ in terms of state and control deviations $\delta\mathbf{x_k} = \mathbf{x_t} - \mathbf{x_k^n}$, $\delta\mathbf{u_k} = \mathbf{u_k} - \mathbf{u_k^n}$ leading to the linear dynamics approximation:

$$\delta\mathbf{x_{k+1}} = \mathbf{A_k}\delta\mathbf{x_k} + \mathbf{B_k}\delta\mathbf{u_k} + \mathbf{C_k}\omega_k \tag{6}$$

where $\mathbf{A_k} = \Delta t\frac{\partial\mathbf{f}}{\partial\mathbf{x_k}}$ and $\mathbf{B_k} = \Delta t\frac{\partial\mathbf{f}}{\partial\mathbf{u_k}}$

$\mathbf{C_k}$ represents how the uncertainty propagates through the system. And the quadratic approximation of the cost

$$\begin{aligned}\tilde{l}(\mathbf{x}, \mathbf{u}, \mathbf{k}) = q_\mathbf{k} + \mathbf{q_k^T}\delta\mathbf{x_k} + \mathbf{r_k^T}\delta\mathbf{u_k} + \frac{1}{2}\delta\mathbf{x^T}\mathbf{Q_k}\delta\mathbf{x_k} + \\ \delta\mathbf{x^T}\mathbf{P_k}\delta\mathbf{u_k} + \frac{1}{2}\delta\mathbf{u^T}\mathbf{R_k}\delta\mathbf{u_k}\end{aligned} \tag{7}$$

with final cost

$$\tilde{l}(\mathbf{x}, \mathbf{N}) = q_\mathbf{N} + \mathbf{q_N^T}\delta\mathbf{x_k} + \frac{1}{2}\delta\mathbf{x_k^T}\mathbf{Q_N}\delta\mathbf{x_k} \tag{8}$$

where $q_\mathbf{k}$ $\mathbf{q_k}$, $\mathbf{r_k}$, $\mathbf{Q_k}$, $\mathbf{R_k}$ and $\mathbf{P_k}$ are the coefficients of the Taylor expansion of the cost function around the nominal trajectory. Given the quadratic control cost, the locally optimal control law will be of the form $\delta\mathbf{u_k} = \mathbf{k_k} + \mathbf{K_k}\delta\mathbf{x_k}$ and the underlying optimal control problem can be solved by using Bellman equation

$$\Psi_\sigma(\delta\mathbf{x_k}, \mathbf{k}) = \min_{\boldsymbol{u}}\{\tilde{l}(\mathbf{x}, \mathbf{u}, \mathbf{k}) + \mathbb{E}[\Psi_\sigma(\delta\mathbf{x_{k+1}}, \mathbf{k+1})]\} \tag{9}$$

and making the following quadratic approximation of the value function Ψ:

$$\Psi(\delta\mathbf{x_k}, \mathbf{k}) = \frac{1}{2}\delta\mathbf{x_k^T}\mathbf{S_k}\delta\mathbf{x} + \delta\mathbf{x_k^T}\mathbf{s_k} + s_k \tag{10}$$

where $\mathbf{S_k} = \nabla_{\delta x\delta x}\Psi$ and $\mathbf{s_k} = \nabla_{\delta x}\Psi - \mathbf{S_k}\delta\mathbf{x_k}$ are functions of the partial derivatives of the value function. Using the linear dynamics, the quadratic cost and the quadratic approximation of Ψ, and solving for the optimal control, we get

$$\begin{aligned}\delta\mathbf{u_k} &= \mathbf{k_k} + \mathbf{K_k}\delta\mathbf{x_k} \\ \mathbf{k_k} &= -\mathbf{H_k^{-1}}\mathbf{g_k} \\ \mathbf{K_k} &= -\mathbf{H_k^{-1}}\mathbf{G_k}\end{aligned} \tag{11}$$

where $\mathbf{H_k}$, $\mathbf{g_k}$, $\mathbf{G_k}$ are given by

$$\begin{aligned}\mathbf{H_k} &= \mathbf{R_k} + \mathbf{B_k^T S_k B_k} + \sigma \mathbf{B_k^T S^T C \Sigma C^T S_k B_k} \\ \mathbf{g_k} &= \mathbf{r_k} + \mathbf{B_k^T s_k} + \sigma \mathbf{B_k^T S_k^T C \Sigma C^T s_k} \\ \mathbf{G_k} &= \mathbf{P_k^T} + \mathbf{B_k^T S_k A_k} + \sigma \mathbf{B_k^T S_k^T C \Sigma C^T S_k A_k}\end{aligned} \tag{12}$$

The corresponding backward recursions are

$$\begin{aligned}\mathbf{s_k} = \mathbf{q_k} + \mathbf{A_k^T s_{k+1}} + \mathbf{G_k^T k_k} + \mathbf{K_k^T H_k k_k} + \\ \sigma \mathbf{A_k^T S_{k+1}^T C \Sigma C^T s_{k+1}}\end{aligned} \tag{13}$$

$$\begin{aligned}\mathbf{S_k} = \mathbf{Q_k} + \mathbf{A_k^T S_{k+1} A_k} + \mathbf{K_k^T H_k K_k} + \mathbf{G_k^T K_k} + \\ \mathbf{K_k^T G_k} + \sigma \mathbf{A_k^T S_{k+1}^T C \Sigma C^T S_{k+1} A_k}\end{aligned} \tag{14}$$

With $\sigma = 0$ the recursions revert to the usual Ricatti recursions for iLQR [22].

III. MBRL via Curious iLQR

In this section we present how we can incorporate curious behaviour into a robot's learning control loop and how it can affect it's capability of achieving a task. Rooted in the literature [10], we believe that by seeking out uncertainties, a robot is able to solve a reinforcement learning task faster and therefore gain higher rewards sooner compared to a non curious robot. Keeping real robot experiments in mind, we design an approach that explores while optimizing a trajectory to achieve a particular task. The exploration should reduce uncertainty about the robots dynamics model, while staying close to the trajectory that is being optimized for the task. More concretely, a result of having high uncertainty in the model is a high variance in an optimized state sequence. In our work we want to seek out actions that reduce this variance. In the following we are going to outline our approach in more detail, and we start by reviewing the general model-based reinforcement learning setting.

A. Model-Based Reinforcement Learning

The goal of model based reinforcement learning (MBRL) is to solve a task through learning a model of the dynamics. The dynamics are described through $x_{k+1} = f(x_k, u_k)$ were x_k and u_k are the state and action of the current time step, and x_{k+1} the state at the next time step. f represents the learned model of the dynamics. The goal is to find a policy $u_k = \pi(x_k)$ s.t $\min_u\{\mathcal{J}(x_k, u)\}$ where $\mathcal{J}$ is the cost associated with taking action u when in state x_k. In our work we initialize our dynamics model with data collected through motor babbling, followed by an optimal control loop to optimize for a locally optimal policy for a fixed time horizon, using the current model. Next the optimal policy under the current dynamics is executed on the robot and the resulting data is used to update the model. In model based reinforcement learning the model and the optimizer are task independent. This independence promises sample efficiency as a already learned model can be reused for new tasks. Specifically in our work, we can learn new reaching positions in as few as three iterations.

B. Curious iLQR via model uncertainty

We now move on to describing the details of our instantiation of a risk-seeking – curious – iLQR approach. In the background Section II-A we recalled the backward recursions for a risk sensitive optimizer. This optimizer works with the assumption that a known dynamics model is disturbed by a stochastic noise process. In our work, in contrast, we do not assume a known dynamics model but learn and improve the dynamics model as we are trying to achieve a task. The quality of our current model affects the quality of the solution an optimizer can find. This can lead to MBRL loops getting stuck in 'local optima' or converging very slowly. Our hypothesis is that, by trying to explore uncertainties in the model, our MBRL loop can escape these local minima and find better solutions, faster, to a task. On a high level we can summarize our approach as follows: To model uncertainties we take a probabilistic approach to modeling the forward dynamics of the system. The optimizer reasons about uncertainties in the dynamics model to find solutions to a nonlinear optimal control problem. The algorithm alternates between improving the probabilistic dynamics of the robot model from the recently unrolled trajectory, optimizing the trajectory using the current model, and rolling out the current locally optimal policy on the system. See Figure 1 for an overview of the approach.

1) Learning a Bayesian dynamics model: Because of their intuitive uncertainty behavior, the Bayesian dynamics model is implemented as a Gaussian Process (GP). Let $\mathbf{x_k}$ denote the state of the system at time step k, where $\mathbf{x_k} = [\boldsymbol{\theta_k}, \dot{\boldsymbol{\theta}}_\mathbf{k}]$ and $\boldsymbol{\theta_k}$, $\dot{\boldsymbol{\theta}}_\mathbf{k}$ are joint positions and velocities respectively. Furthermore, let $\mathbf{u_k}$ denote commanded torques $\mathbf{u_k} = [\tau_1, \tau_2, ..., \tau_N]$ where $N =$ DoF. Then tuples of states and actions $(\mathbf{x_k}, \mathbf{u_k}) \in \mathbb{R}^{F+A}$ are used as training inputs and $\ddot{\boldsymbol{\theta}}_{\mathbf{k+1}} \in \mathbb{R}^F$, joint accelerations at the next time step, as training outputs. Once trained, the GP delivers one step predictions of the form

$$p(\ddot{\boldsymbol{\theta}}_{\mathbf{k+1}} | \mathbf{x_t}, \mathbf{u_t}) = \mathcal{N}(\ddot{\boldsymbol{\theta}}_{\mathbf{k+1}} | \mathbf{f}\left(\mathbf{x_k}, \mathbf{u_k}\right) \Delta t, \boldsymbol{\Sigma}_{\boldsymbol{k+1}}) \tag{15}$$

where $\mathbf{f}$ is the mean vector and $\boldsymbol{\Sigma}_{\mathbf{k+1}}$ the covariance matrix of the predictive distribution evaluated at $(\mathbf{x_k}, \mathbf{u_k})$. The GP outputs the acceleration at the next time step $\ddot{\boldsymbol{\theta}}_{\mathbf{t+1}}$ which is numerically integrated to velocity

$$\ddot{\boldsymbol{\theta}}_{\mathbf{k+1}} \Delta t + \dot{\boldsymbol{\theta}}_\mathbf{k} = \dot{\boldsymbol{\theta}}_{\mathbf{k+1}} \tag{16}$$

and position

$$\dot{\boldsymbol{\theta}}_{\mathbf{k+1}} \Delta t + \boldsymbol{\theta_k} = \boldsymbol{\theta_{k+1}}. \tag{17}$$

The prior mean of the GP is considered to be zero and the chosen kernel is problem dependent. Given a test input $\mathbf{x_*} = (\mathbf{x_k}, \mathbf{u_k})$, the posterior distribution, under the prior, of the GP takes the form of

$$p(\ddot{\boldsymbol{\theta}}_{\mathbf{k+1}} | \mathbf{x_*}, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\ddot{\boldsymbol{\theta}}_{\mathbf{k+1}} | \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{y}, k_{**} - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k_*}) \tag{18}$$

where $\mathbf{X}$ and $\mathbf{y}$ are the training inputs and training outputs collected while running the policy on the robot. κ is the kernel function, $\mathbf{k_*} = [\kappa(x_*, x_1), ..., \kappa(x_*, x_N)]$, $k_{**} =$

$\kappa(x_*, x_*)$ and $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$. The use of a probabilistic method to model the dynamics of the system has the benefit of naturally including a notion of uncertainty in the predictions, in contrast to other methods, like neural networks, where this can only be achieved by emulating predictions of distributions, see [11] for an overview. Next we outline how model uncertainty can be included during policy optimization.

2) Seeking out uncertainties: When referring back to equation (1), our system dynamics can be defined as

$$\mathbf{x_{k+1}} \sim \mathcal{N}(\mathbf{x_{k+1}} | \mathbf{h}\left(\mathbf{x_k}, \mathbf{u_k}\right), \mathbf{\Sigma}) \tag{19}$$

where $\mathbf{x_{k+1}}$ is a random variable, $\mathbf{h}$ is the mean prediction of the GP integrated as in equations (16) and (17) to the future state $\mathbf{x_{k+1}}$. Σ is the covariance matrix of the posterior distribution of the GP predictions. When linearising our dynamics around a nominal trajectory

$$\delta\mathbf{x_{k+1}} = \mathbf{A_k}\delta\mathbf{x_k} + \mathbf{B_k}\delta\mathbf{u_k} + \mathbf{C_k}\omega_k \tag{20}$$

we have $\mathbf{A_k} = \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{x_k}}$, $\mathbf{B_k} = \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{u_k}}$ and $\omega_k \sim \mathcal{N}(\omega_k | 0, \mathbf{\Sigma_{k+1}})$, where $\mathbf{A_k}$ and $\mathbf{B_k}$ are the analytical gradients of the GP prediction at each time step.

For our formulation, the covariance matrix is of particular interest, as it provides the information that is essential for our algorithm to explore. The uncertainty of the Gaussian process model when evaluating the posterior distribution at $(\mathbf{x_k}, \mathbf{u_k})$ is the notion of uncertainty that can be used in order to encourage the agent to choose actions such that the uncertainty of the model will be resolved in the future. It is important to note now that $\mathbf{\Sigma_{k+1}}$ in equations (12) and (13) represents the uncertainty of the model when predicting $\mathbf{x_{k+1}}$ from $(\mathbf{x_k}, \mathbf{u_k})$. As can be seen from equations (13) and (14), with a negative σ value, the cost explicitly favors for the uncertainty in the dynamics. As a result, the agent is encouraged to select actions that include some level of uncertainty while still trying to reduce the task specific error. With $\sigma = 0$ the agent will ignore any uncertainty in the environment and therefore not explore around the trajectory that should be optimized. In this case the optimizer ignores any higher order statistics of the cost function and only resolves the optimal control problem by minimizing the expectation over the cost.

C. Challenges in High-dimensional State space

Previous work such as [7] and [16] using risk-sensitive control variations of iLQR primarily deal with simplified, low dimensional problems. Our experiments are conducted on a 7 degree of freedom Sawyer robot, and the higher dimensional system adds some complexities to the approach. Firstly, we approximate our dynamics using 7 Gaussian Process models, which model each joint's dynamics independently. This makes a strong assumption about dimensional independence, but also makes our analysis much simpler. Moreover, vanilla Gaussian Processes do not scale to large number of datapoints, and we use a sparse GP representation from the GPy [8] library. This approximation allows us to work with fairly high-dimensional input spaces (21 dimensional in experiments with a 7 DoF robot arm) and use large number of datapoints for training the dynamics. In Section II-A, the gradients of the value function (Equations (13), (14)) tend to suffer from numerical ill-conditioning in high-dimensions. We account for this issue with Tikhonov regularization: before inversion for calculating the optimal control we add a diagonal constant matrix to $\mathbf{H_k}$ from Equation (12). Before we describe our high dimensional experiments, we illustrate our approach with two simple toy examples.

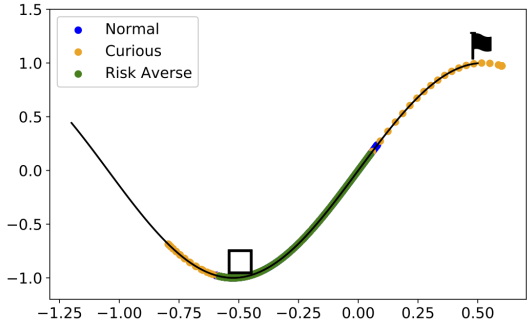


Fig. 2: Explored states of the environment in the mountain car problem by risk-averse, normal and curious iLQR.

IV. Illustration: Curious iLQR

In this section, we show some toy examples that illustrate the advantages of using curiosity – the motivation to resolve uncertainty – as an exploration tool. The objectives of these toy experiments are twofold (1) they should provide an analysis of whether it is possible to escape local minima with curiosity, that a locally optimal optimizer, can potentially get stuck in [22]. And (2) if by trying to resolve the uncertainty in the model, the agent explores parts of the state space that are less known but contain relevant information for solving potential future tasks and is therefore able to transfer easier. In the following, and throughout the paper, we will refer to the agent that tries to resolve the uncertainty in its environment as curious and the one that is not following the uncertainty as normal.

A. Mountain Car Example

In this experiment, we analyze the benefit of including model uncertainty during trajectory planning to find a globally optimal solution, instead of converging to a local optimal solution, when using a locally optimal feedback controller. The mountain car problem [14] consists of an under-actuated car that should be parked at the top of a steep hill when starting from the valley. In this example gravity is stronger then the cars engine so the car cannot simply accelerate to drive up the hill and must gain momentum by driving up the opposite hill. This problem is representative for how an optimizer, that finds locally optimal solutions, is not able to compute a trajectory to actually solve the task. Locally the best action would always be to drive up the goal hill and not drive away from the goal, to leverage potential energy, and finally reach the desired goal position. We learn

a probabilistic model of the system dynamics and a policy as presented in III.

To show that the curious agent, in fact, explores a wider portion of the state space, Fig. 2 shows the position of the mountain car over the course of the experiment. The curious agent is extensively exploring the state space, while the normal agent, as well as the risk averse agent, suffers from being trapped in its locally optimal solution. The behaviour of driving away from the goal is only observed in the curious agent, and thus this agent is the only one receiving a reward by the end of the episode.

B. Learning model faster by following its own uncertainty

The second experimental platform is the OpenAI gym Reacher environment [4], a two degrees of freedom arm attached at the center of the scene. The goal of the task is to reach a target placed in the environment. In the experiments presented below, actions were optimized as described in section III. The intuition behind this experiment is that, if an agent is driven to resolve uncertainty in its model, by exploring less known states, a better model of the system dynamics can be learned and therefore used to optimize a control sequence more reliably. Our hypothesis is that the model learned by the curious agent is more certain by the end of learning. As it was driven to resolve uncertainties in the environment we expect it to be more robust when solving the reaching task for new targets. In Figure 3 the end-effector

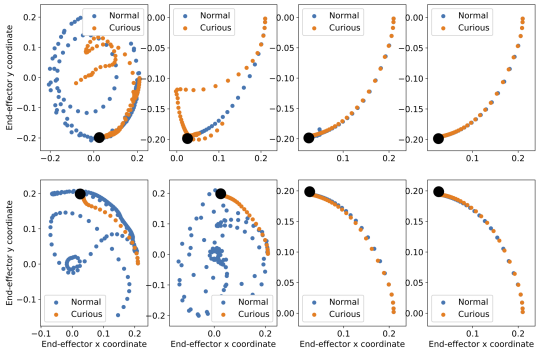


Fig. 3: End-effector position of curious and not curious agent for 4 learning iterations on 2 different targets. The targets are represented by the black dots.

position in euclidean space shows the behaviour of the 2D arm of the curious and normal agent. Four outer learning iterations are shown for two desired target positions. Each row depicts a different target position. The curious agent tries to resolve the uncertainty within the model, the normal agent does not consider the uncertainty in the model but optimizes only for the task related reward. For the first target position the normal agent seemingly reaches the target after the second learning iteration, the curious agent only manages to reach the target during the third iteration. Interestingly, the exploration of the curious agent leverages the arm to reach the second target position immediately and continues to reach it consistently thereafter. Figure 4 confirms the assumption that the curious agent is more certain than the normal agent, as it is certain about the predictions it is making to reach the second target. The figure shows the uncertainty and the prediction error of the model for the normal and the curious agent respectively. The values are reported in end-effector space. The curious model overall has lower uncertainty and prediction error values compared to the normal agent, over the whole state space.

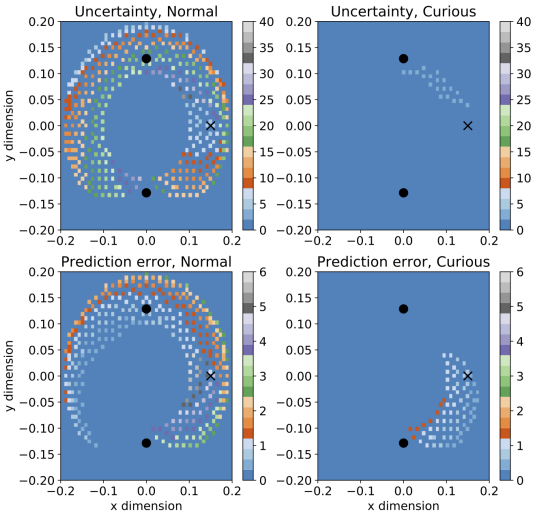


Fig. 4: The uncertainty and the prediction error in end effector space for the normal and curious agent. The dots represent the target positons, the first target is on the bottom, the second on the top. The cross is the initial position when starting the experiment. Regions that are not by the arm reachable are shown in dark blue.

V. Experiments on high-dimensional problems

Lastly the goal of this work is to perform experiments on torque-controlled manipulator. Our first experimental platform is a 7 degrees of freedom Kuka iiwa7 simulation in the PyBullet physics simulator [17]. In the next Section, we present results on a 7 degrees of freedom Sawyer robot arm hardware from Rethink Robotics [18]. The goal of these experiments is to reach a desired target joint configuration. The dynamics are learned with Gaussian process regression. For each output dimension a separate GP is trained and the algorithm used is the same as presented in section III. We believe that curiosity (on average) helps the optimizer to find better solutions faster, because it helps escape local optima within the MBRL loop. Curiosity also means we explore more and because of that we learn more about the dynamics which helps when using the model for new targets. Overall we believe that, as we improve our model while rolling out trajectories optimized under the current model, curiosity helps us escape the local minima given by the possibly bad quality of our model, by exploring more and learn more about the dynamics of the system. In simulation, we performed three different kinds of experiments that we will describe in detail below. Throughout all of the simulation

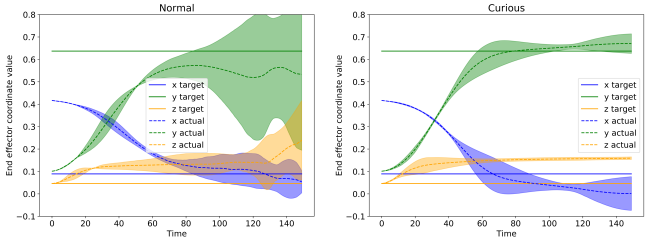


Fig. 5: Final performance in end-effector space after learning to reach one target position using MBRL with (left) normal, (right) curious iLQR. Solid lines represent targets, dashed lines actual trajectories in x, y, z. The mean and the standard deviation are computed across 10 different trials.

experiments the trajectory was $0.625s$ long at a sampling rate of 240 Hz. Motor babbling was performed at the beginning for $0.5s$ by commanding sine-trajectories in each joint.

A. Reaching task from scratch

During the first set of experiments, we wanted to compare the performance of both controllers when learning to reach a given target configuration from scratch. We performed ten experiments for each kind of controller while always perturbing the initial joint positions slightly. We also randomly initialize the torque trajectories for the optimizer to make sure the initial trajectory would not influence convergence to a solution. For a given target joint configuration, 20 iterations of optimizing the trajectory, running the trajectory on the system and updating the dynamics model, were performed. We perform this experiment for 3 different target joint configurations. For each target we run 10 trials, where the variability between trials is caused by the random initial torque trajectory, and by small perturbations to the starting configuration (simulating sensing noise). Figure 5 visualizes the mean and variance of the end-effector trajectories across the 10 trials on the first target. We observe that MBRL with the curious iLQR reaches the target end-effector position, by the end of learning, more consistently then with normal iLQR. Most notably is the variance of the final trajectories when using MBRL with regular iLQR. This suggests, that at least for a few trials MBRL with iLQR failed to converge to the target within 20 learning iterations. We summarize results of all 3 experiments, corresponding to 3 different target joint configurations in Table I. Here we present both the final end-effector distance, as well as how often the MBRL loop succeeded to reach the target with at least 10cm precision within 20 iterations. We see, that, independent of the target, on average, MBRL with curious iLQR reaches the target faster and more reliably.

	Precision		**Success**	
Experiment	Curious	Normal	Curious	Normal
1	0.07	0.20	80%	40%
2	0.17	0.60	80%	50%
3	0.04‘	0.062	100%	80%

TABLE I: Results for 3 set of experiments, each experiment had a different target configuration and was performed 10. Experiment 1 corresponds to Figure 5

B. Optimizing towards new targets after model learning

To confirm the hypothesis that the models learned by MBRL with curious iLQR generalize better, because they have explored the state space better, we decided to evaluate the learned dynamics models on a second set of experiments in which the robot tries to reach new, unseen targets. Thus, in this experiment with take the GP models learned during experiment 1 in Section V-A and use them to optimize trajectories to reach new targets that were not seen during training of the model. The results are shown in Figure 6, where three randomly chosen targets where set and the trajectory was optimized with regular iLQR. Note, that here we use regular iLQR to optimize for the trajectory so that we can better compare the models learned with/without curiosity in the previous set of experiments. The results are

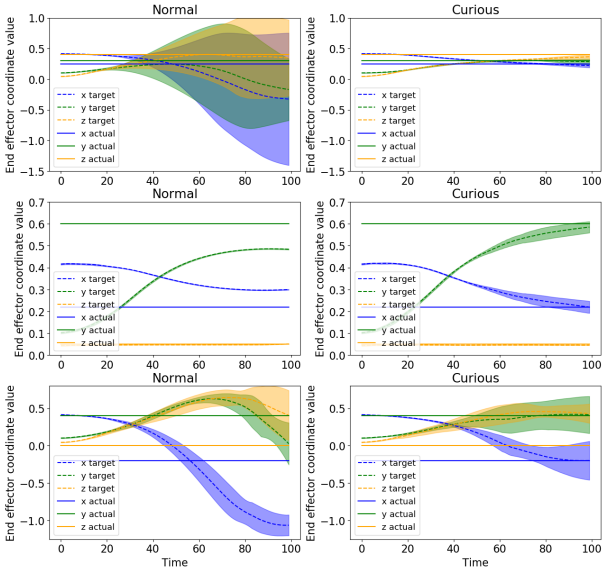


Fig. 6: Optimizing to reach new targets with regular iLQR after dynamics models were learned. 3 different targets (one per row) are evaluated and the final end-effector trajectories presented. The mean and the standard deviation of the optimized trajectories are computed across the 10 models learned via MBRL with normal iLQR (left col) and MBRL with curious iLQR (right col).

averaged across 10 trials. Here, the 10 trials correspond to using one of the 10 dynamics models at the end of Experiment 1 in Section V-A. For each trial, the initial torque trajectory was initialized randomly, and the initial joint configuration slightly perturbed. The mean trajectories and standard deviation of the end-effector trajectories across the 10 trials are shown. We see that MBRL with curious iLQR results in a model that performs better when presented with a new target. The new targets are reached more reliably and precisely. Here, we chose the new target location to be close to the initial target (from Experiment 1 in Section V-A) because we assume that the MBRL runs in the previous Section only learned reliable dynamics models locally.

C. Continual learning towards 2nd target

In an additional experiment, we test the continual updating of the dynamics models, when optimizing for a 2nd target

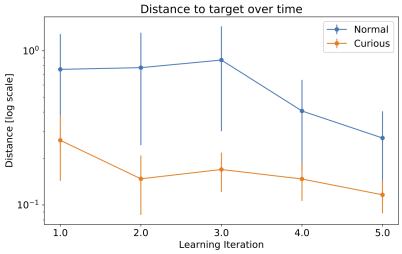


Fig. 7: Mean and standard deviation of the euclidean distance between the end-effector position and the target over the course of five model learning iterations.

(after target 1 was achieved in Experiment V-A). Here, we select a target end-effector position that is far away from the initial, and let MBRL with normal and with curious iLQR continue to update their respective models. Because here we already start with an initial dynamics model, we perform a maximum of five rollouts and learning updates. The mean distance to the desired end-effector position averaged across 5 trials is shown in Figure 7. The 5 trials correspond to 5 randomly chosen GP models from Experiment 1 in Section V-A, and randomly intialized torque trajectories. The plot shows, that MBRL with curiosity already get's closer to the target during the first rollout. This is likely to be a combination of both a better dynamics models learned with curious iLQR in the first experiment, and the continued use of curiosity to optimize to the new target. The final optimized trajectories using MBRL with curious iLQR reaches the target with high precision faster than continual MBRL with regular iLQR. We give a visual depiction of the end-effector trajectories for the 5 system rollouts in Figure 8, for one trial.

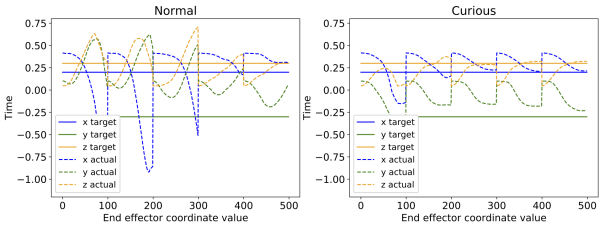
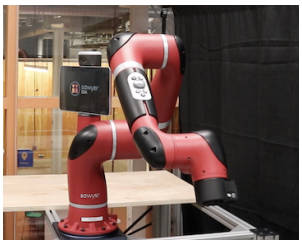


Fig. 8: Trajectories for end-effector positions over the course of the five rollouts and learning iterations are shown together with the desired end-effector position.

VI. Real Hardware Experiments

The experimental platform for our hardware experiments is the Sawyer Robot [18], again a 7 DoF manipulator. The purpose of the experiments was to demonstrate the applicability and the benefits of our algorithm on real hardware. We perform reaching experiments for 4 different target locations. Each experiment is started from scratch with no prior data, and the number of hardware experiments needed to reach the target are compared. The results are summarized in Table II and show the number of learning iterations needed in order to reach the target together with the precision in end-effector



(a) rest configuration

(b) target configuration

Fig. 9: Rest and target configuration of experiments on the Sawyer robot.

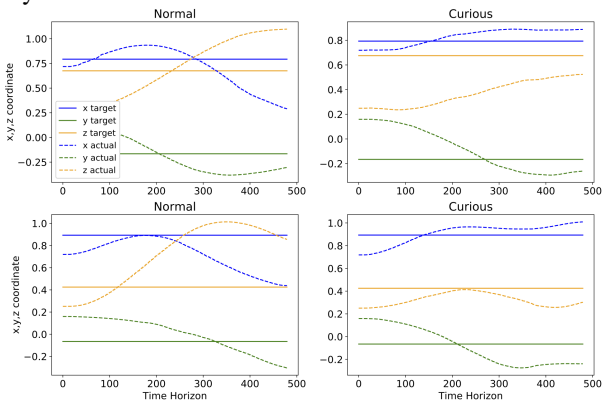


Fig. 10: Distance to target end-effector position after trajectory optimization for two targets.

space. If the target was reached with a precision of below 10 cm, we would consider the task as achieved, if the target was not reached after the 8th learning iteration we would stop the experiment and consider the last end-effector position. Running the experiment on hardware was a lengthy process, as the GP training and the rollout would happen iteratively and GP training time increases with growing amount of data. For this reason we decided to terminate our experiments after eight iterations. We repeated each experiment three times to demonstrate the repeatability of our method as we expected measurement noise to affect solutions. From the table we can see that MBRL with curious iLQR would reach a target on average after 3.5 iterations with a average precision of 7 cm compared to MBRL with regular iLQR that needed 5.9 iterations (often not ever reaching the target after eight iterations with the desired precision), with a precision of 14cm on average. As in simulation, similar to Experiment V-B we wanted to evaluate the quality of the learned models on new target positions. The results are summarized in Table III and are similar to what we observe in simulation, the models learned with curiosity, when used to optimize for new targets can achieve higher precision then when using the models learned without curiosity. Note, again here we utilize regular iLQR for the optimization for both types of learned models. In Figure 10 we show two example of end-effector trajectories when reaching two targets, for each approach.

VII. Conclusion and Future Work

In this work, we presented a model-based reinforcement learning algorithm that uses an optimal control framework

Target	Learning Iterations								Distance to Target							
	Curious				Normal				Curious				Normal			
1	6	2	3	**3.67**	8	3	8	**7.0**	0.05	0.09	0.09	**0.07**	0.37	0.08	0.18	**0.21**
2	3	4	4	**3.67**	8	3	5	**5.3**	0.05	0.09	0.09	**0.09**	0.20	0.08	0.09	**0.12**
3	6	4	3	**4.33**	8	8	8	**8.0**	0.09	0.09	0.09	**0.09**	0.17	0.16	0.11	**0.15**
4	3	2	2	**2.33**	3	3	3	**3.0**	0.04	0.07	0.07	**0.06**	0.04	0.08	0.05	**0.06**
				3.5				**5.9**				**0.07**				**0.14**

TABLE II: Results on a reaching task on the Sawyer robot. The experiment for each target was repeated three times, the number of learning iterations and the final end-effector distance to the target is reported.

	Reaching Precision (cm)	
Target	**Curious**	**Normal**
1	0.20	0.67
2	0.26	0.61
3	0.25	1.06
4	0.24	0.67
5	0.37	0.49
	0.26	**0.7**

TABLE III: Results on optimizing to reach a new target not seen during training. The distance of the endeffector to the target is reported in meters.

to trade-off between optimizing for a task specific cost, and exploring around a locally optimal trajectory. Our algorithm explicitly rewards actions that seek out uncertainties in our model, by incorporating them into the cost. By doing so, we are able to learn a model of the dynamics that achieves the task faster than MBRL with standard iLQR, and also transfers well to other tasks. We present experiments on Kuka iiwa7 arm in simulation, and a Sawyer robot on hardware. In both sets of experiments, MBRL with curious iLQR (our approach) not only learns to achieve the specified task faster, but also generalizes to new tasks and initial conditions. All this points towards the conclusion that resolving dynamics uncertainty during model-based reinforcement learning is indeed a powerful tool. As [13] states, curiosity is a superficial affection, it can arise, diverge and end promptly. We were able to observe similar behaviour as well in our experiments. Once the task was achieved, the robot would explore and deviate slightly from the task, and go back to exploiting once its fairly certain about the dynamics. In the future, we would like to explore this direction by considering how to maintain exploration strategies and potentially reuse them later, especially on new tasks. This could be helpful if the robot is still certain about a task, even though the environment or task has changed. Furthermore, we noticed that, with increasing amount of high dimensional data, the scalability of the Gaussian processes was reaching its limits. Therefore we would want to investigate the potential of Bayesian neural networks in our MBRL framework in the future.

REFERENCES

[1] Christopher G Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3557–3564. IEEE, 1997.

[2] Andrew G. Barto. Intrinsically motivated learning of hierarchical collections of skills. *International Conference on Developmental Learning and Epigenetic Robotic*, pages 112–119, 2004.

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[5] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[6] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011.

[7] Farbod Farshidian and Jonas Buchli. Risk sensitive, nonlinear optimal control: Iterative linear exponential-quadratic optimal control with gaussian noise. *arXiv preprint arXiv:1512.07173*, 2015.

[8] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[9] David H Jacobson. Optimal Stochastic Linear Systems with Exponential Performance Criteria and Their Relation to Deterministic Differential Games. *IEEE Transaction on Automatic Control*, 18, 1973.

[10] Jerome Kagan. Motives and development. *Journal of Personality and Social Psychology*, 22(1):51–66, 1972.

[11] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision?, 2017.

[12] Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arXiv preprint arXiv:1807.01521*, 2018.

[13] George Loewenstein. The psychology of curiosity: A review and reinterpretation. *Psychological bulletin*, 116(1):75, 1994.

[14] Andrew Moore. Efficient memory-based learning for robot control. *PhD thesis, University of Cambridge*, 1990.

[15] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July:488–489, 2017.

[16] Brahayam Ponton, Stefan Schaal, and Ludovic Righetti. Risk sensitive nonlinear optimal control with measurement uncertainty. *CoRR*, 2016.

[17] games Pybullet, a python module for physics simulation in robotics and machine learning., 2016–2017.

[18] Rethink Robotics.

[19] S. Singh, A.G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. *18th Annual Conference on Neural Information Processing Systems (NIPS)*, 2004.

[20] Satinder Singh, Richard L. Lewis, Andrew G. Barto, and Jonathan Sorg. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[21] Daniel Tanneberg, Jan Peters, and Elmar Rueckert. Intrinsic motivation and mental replay enable efficient online adaptation in stochastic recurrent networks. *Neural Networks*, 109:67–80, 2019.

[22] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. *IEEE International Conference on Robotics and Automation, ICRA*, 2014.

[23] Robert W White. Motivation reconsidered: The concept of competence. *Psychological review*, 66(5):297, 1959.

[24] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

[25] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James Rehg, Byron Boots, and Evangelos Theodorou. Information theoretic MPC for model-based reinforcement learning. In *ICRA*, 2017.