Model-based Reinforcement Learning with Parametrized Physical Models and Optimism-Driven Exploration

Chris Xie Sachin Patil Teodor Moldovan Sergey Levine Pieter Abbeel

***Abstract*— In this paper, we present a robotic model-based reinforcement learning method that combines ideas from model identification and model predictive control. We use a feature-based representation of the dynamics that allows the dynamics model to be fitted with a simple least squares procedure, and the features are identified from a high-level specification of the robot's morphology, consisting of the number and connectivity structure of its links. Model predictive control is then used to choose the actions under an optimistic model of the dynamics, which produces an efficient and goal-directed exploration strategy. We present real time experimental results on standard benchmark problems involving the pendulum, cartpole, and double pendulum systems. Experiments indicate that our method is able to learn a range of benchmark tasks substantially faster than the previous best methods. To evaluate our approach on a realistic robotic control task, we also demonstrate real time control of a simulated 7 degree of freedom arm.**

I. Introduction

Model-based control of robotic systems requires model identification to be performed before the system can be effectively controlled, particularly for dynamic, high-speed motions. One way to tackle this challenge is to first perform model identification, and then use the identified model to control the system [16], [20]. However, this approach requires a dedicated model identification step, which can become inefficient if the dynamics change frequently or suddenly, or if the robot interacts with unfamiliar physical objects that must each be identified.

Reinforcement learning (RL) offers a framework for automatically trading off exploration and exploitation to complete the task as quickly as possible. Model-based RL reduces the required system interaction time by learning a model of the dynamics, while still trading off exploration and exploitation to learn a model that is just detailed enough to succeed at the task. General-purpose statistical models are often used to represent the dynamics, but such models can require a substantial number of samples to acquire a sufficiently accurate dynamics estimate [13], [21].

In this paper, we combine concepts from model identification and model-based reinforcement learning to complete the task as quickly as possible while identifying the system online to acquire a model that is sufficient for task completion. In contrast to prior methods that use generic statistical models of the dynamics, we use a feature-based least-squares formulation of the model identification problem, which allows the model to be identified extremely quickly by bringing in our prior knowledge about the robot's morphology and physics. Exploration is performed using an optimistic model-predictive control (MPC) framework [21], which determines the optimal trajectory under an optimistic formulation of the system dynamics. As more interaction samples are gathered, the amount of optimism is reduced, until the method converges to the true dynamics.

Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA.

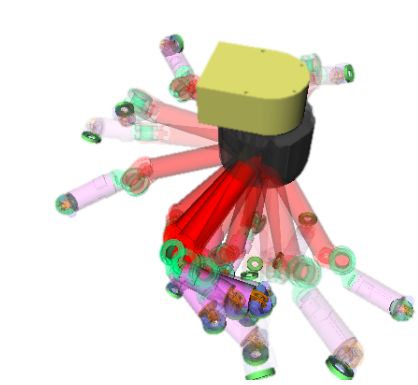

Fig. 1. 7 DoF arm learning to reach a target pose using our method. Later time steps shown with higher opacity, with the target pose fully opaque.

Our main contribution is a method for combining optimism-driven exploration with simple least-squares model fitting based on physical features of the dynamics that can be extracted automatically from a high-level specification of the system morphology (e.g., the number and connectivity of the links, which is readily available for most robotic systems). These features can be obtained manually for simple systems, or can be computed automatically using existing packages such as SymPyBotics [26]. We first present experimental results on the pendulum, double pendulum, and cartpole benchmarks. Compared to prior methods, our approach is able to solve these tasks using the lowest amount of system interaction time. In comparison to a prior method based on optimism-driven exploration [21], our approach also requires much less computation time, making it suitable for real-time online learning. To evaluate our method on a realistic robotic control task, we also demonstrate real time control of a simulated 7 degree of freedom arm, shown in Figure 1.

II. Related Work

System and model identification has been explored extensively in the context of robotics [16], [20]. Several methods have been proposed in the literature for finding good excitation trajectories for model identification [5], [14], [25], [28], [31], [32]. The physical feasibility of parameters during the identification process has also been considered [15],

[26]. However, model identification is typically performed as a separate process from control. In contrast, our work addresses the problem of controlling a robotic system to complete a task as quickly as possible, while learning a model sufficient for task completion.

One alternative to offline model identification is provided by adaptive control [6]. Adaptive control offers compelling convergence and stability guarantees, but is typically concerned with stabilizing around a target trajectory under a linear model. This makes it difficult to apply to more complex, nonlinear robotic systems with high-level goals defined by arbitrary cost functions.

Reinforcement learning (RL) [18], [27] tackles control problems with nonlinear dynamics in a more general framework, which can be either model-based or model-free. Model-based RL reduces the required interaction time by learning a model of the system during execution, and optimizing the control policy under this model, either offline in an episodic setting, or online. In the context of RL, exploration refers to intentionally taking suboptimal actions to improve future performance. Although many methods have been proposed for model-based RL with efficient exploration in discrete MDPs, data-efficient model-based RL for continuous systems remains a challenging problem despite substantial recent advances [1], [13], [19], [21].

Although several very successful model-based RL methods have been proposed recently [8], [13], [21], such methods typically use general-purpose statistical models of the dynamics. Such models are very flexible, but require a substantial number of samples to learn models that are accurate enough to succeed at the task. Several prior methods have suggested incorporating knowledge about the dynamics as a prior on the dynamics model [11], [12], [23]. However, these methods typically assume that prior knowledge comes in the form of predictions about the next state, which is a very specific and quantitative type of prior. Our approach incorporates prior knowledge about the dynamics of rigid-body systems, but does not assume knowledge of system parameters, such as masses and link lengths. Without these parameters, this prior model cannot give reasonable predictions. However, it tells us a great deal about the *structure* of the dynamics. The equations of motion can be written such that the parameters and model features decompose linearly, providing for a very efficient learning algorithm.

In order for the algorithm to learn the model at the same time as it performs the task, we use an optimism-driven exploration strategy combined with model-predictive control to continuously replan the next action. Prior work has proposed to use optimism-driven exploration mainly for discrete systems [4], [19]. A recent extension to continuous nonlinear systems provides for sample-efficient learning [21], but does not run in real time, making it impractical for real applications. We use an efficient MPC method based on differential dynamic programming (DDP) [17], which allows us to achieve real-time performance even while continuously refitting the model parameters with each new sample.

Algorithm 1 Model-based RL with MPC and optimistic exploration

Require: Start state $\mathbf{x}_{\text{start}} = [\dot{\mathbf{q}}, \mathbf{q}]_{\text{start}}$, cost function $l(\mathbf{x}, \tau)$, sampling frequency ν_s, control frequency ν_c

$\tau \leftarrow$ Random controls
$\mathbf{o} \leftarrow$ Empty list of observations
repeat
  Execute τ for $1/\nu_c$ seconds
  Append current $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \tau]$ to $\mathbf{o}$ every $1/\nu_s$ seconds
  Estimate dynamics $\hat{f}(\mathbf{q}, \dot{\mathbf{q}}, \tau)$ using samples in $\mathbf{o}$
  Construct optimistic dynamics $\tilde{f}(\mathbf{q}, \dot{\mathbf{q}}, \tau, \boldsymbol{\xi}) = \ddot{\mathbf{q}}$
  Optimize a trajectory from $[\dot{\mathbf{q}}, \mathbf{q}]$ using $\tilde{f}(\mathbf{q}, \dot{\mathbf{q}}, \tau, \boldsymbol{\xi})$
  Update τ to be the first action along this trajectory
until task completion

III. Optimistic Exploration with Continuous Model Identification

Our method combines feature-based model identification with optimistic exploration in an online model-based reinforcement learning algorithm. An outline of the method is presented in Algorithm 1. Here, $\mathbf{q}$ denotes the configuration of the robot, $\mathbf{x} = [\dot{\mathbf{q}}, \mathbf{q}]$ is the state of the system, and τ is the commanded action (e.g. the joint torques and forces). The task is specified by a cost function $l(\mathbf{x}, \tau)$, which typically depends on the distance between the current state $\mathbf{x}$ and some target state $\mathbf{x}_{\text{goal}}$. We begin with a random initial action and empty list of observations $\mathbf{o}$. We then repeatedly execute the current action τ for $1/\nu_c$ seconds, while collecting new observations $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \tau]$ at a frequency of ν_s. These observations are appended to $\mathbf{o}$. This list of observations is used to estimate the system dynamics $\hat{f}(\mathbf{q}, \dot{\mathbf{q}}, \tau) = \ddot{\mathbf{q}}$, as described in Section III-A. These estimated dynamics are then converted into optimistic dynamics $\tilde{f}(\mathbf{q}, \dot{\mathbf{q}}, \tau, \boldsymbol{\xi}) = \ddot{\mathbf{q}}$ by including a set of *virtual controls* $\boldsymbol{\xi}$ to allow the MPC algorithm to take optimistic action, as described in Section III-B. Using these estimated dynamics, the method then plans a fixed-horizon trajectory that minimizes the cost $l(\mathbf{x}, \tau)$ from the current state $\mathbf{x}$ by using differential dynamic programming (DDP), as described in Section III-C. The next action τ is then set to the first action along this trajectory, following the model predictive control (MPC) paradigm [10]. This process is repeated until a specified goal condition is reached.

A. Model Identification via Least Squares

In this section, we describe how the approximate dynamics $\hat{f}(\mathbf{q}, \dot{\mathbf{q}}, \tau)$ are fitted to samples $\mathbf{o} = \{[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \tau]_{\mathbf{i}}\}$. This method assumes that we know the morphology of the robot (the number and connectivity of its links), and therefore can write down its equations of motion. However, we do not necessarily know the physical parameters, such as the masses and lengths of the links. This assumption is reasonable for many physical systems, since the morphology and connectivity of the links can easily be ascertained from observation, but the physical parameters require a complex system identification procedure. For a robot consisting of a system of articulated

rigid bodies, the equations of motion can be decomposed such that model identification can be formulated as linear regression, making the dynamics fitting simple and efficient. While this technique is known in the model identification literature [16], we present it here for completeness.

Using $M(\mathbf{q})$ to denote the mass matrix, $C(\mathbf{q},\dot{\mathbf{q}})$ to represent the Coriolis and centripetal forces, $g(\mathbf{q})$ to represent gravity, and $\boldsymbol{\tau}$ the forces and torques, the equations of motion are given by

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}}) + g(\mathbf{q}) = \boldsymbol{\tau}. \tag{1}$$

These dynamics equations can be written as:

$$H(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) \cdot \Delta = \boldsymbol{\tau},$$

where the vector Δ depends only on system parameters and the matrix $H(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$, also referred to as the regressor matrix, does not depend on the system parameters. Model identification can then performed by estimating Δ. For instance, the vector $\Delta = \begin{bmatrix}\boldsymbol{\delta}_1^\intercal & \dots & \boldsymbol{\delta}_K^\intercal\end{bmatrix}^\intercal$ for K-link manipulators consists of the inertial parameters $\boldsymbol{\delta}_k$ for each link. For under-actuated systems, the dynamics can be expressed in the same form. A key difference is that we have zeroes in the $\boldsymbol{\tau}$ vector corresponding to the unactuated degrees of freedom. We describe how this decomposition is performed for specific robotic systems in Section IV.

We assume that we have noisy observations of the features $[\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}]$. Given an observation vector $\mathbf{z}$ of the features $[\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}]$ and generalized joint forces/torques $\boldsymbol{\tau}$ for N samples, the vector Δ may be inferred by least squares regression as:

$$\hat{\Delta} = \underset{\Delta}{\operatorname{argmin}} \|A\Delta - \mathbf{b}\|^2$$

where

$$A = \begin{bmatrix} H(\mathbf{q}_1,\dot{\mathbf{q}}_1,\ddot{\mathbf{q}}_1) \\ \vdots \\ H(\mathbf{q}_N,\dot{\mathbf{q}}_N,\ddot{\mathbf{q}}_N) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \boldsymbol{\tau}_1 \\ \vdots \\ \boldsymbol{\tau}_N \end{bmatrix}.$$

Since $H(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ is typically not full rank, A is not full rank and the solution is an affine subspace. Thus, we use the Moore-Penrose pseudo-inverse to get our solution $\hat{\Delta} = A^\dagger \mathbf{b}$. This gives us the least norm solution in the affine subspace. Once we estimate $\hat{\Delta}$, we can recover the forward dynamics equation $\hat{f}(\mathbf{q},\dot{\mathbf{q}},\tau) = \ddot{\mathbf{q}}$ by solving the equations of motion in (1) with respect to $\ddot{\mathbf{q}}$. This forward dynamics estimate can then be used with any model predictive control method to choose locally optimal actions. However, acting greedily with respect to this dynamics estimate is not always desirable. When the dynamics are incorrect, it may be preferable to instead take actions that are suboptimal under the estimated model, but that are more effective at exploring the state space of the system, in order to acquire a better estimate of the dynamics that can allow the method to more quickly reach the goal. In the next section, we describe one particular exploration method that involves constructing an *optimistic* estimate of the dynamics.

B. Optimistic Exploration

Our method uses model predictive control (MPC) to choose the actions. In order to perform the task quickly while identifying the model to a sufficient degree for task completion, we augment MPC with optimistic exploration. This combination of MPC with exploration allows for the exploration strategy to change online as the model is updated. The intuition behind this exploration strategy is that, when the dynamics are uncertain, the algorithm is allowed to choose which of the dynamics models in prefers, among those models that are highly probable given the data. If the algorithm chooses an accurate dynamics model, it will complete the task. If it chooses an inaccurate model, it will receive observations that show that this model is inaccurate, and the dynamics estimate will be improved. In prior work, this type of exploration strategy was shown to substantially improve the sample-efficiency of model-based RL [21]. However, this prior method suffered from very long computation times, which made it impractical for real-time online control. In this section, we present a simplified variant of the optimistic exploration framework suitable for real-time applications. In the next section, we show how it can be incorporated into a simple and efficient DDP algorithm to allow for efficient, real-time control.

In order to allow MPC to choose among the likely dynamics models, we introduce slack variables $\boldsymbol{\xi}_t$ into the dynamics, such that $\ddot{\mathbf{q}}_t = \hat{f}(\mathbf{q}_t,\dot{\mathbf{q}}_t,\tau_t) + \boldsymbol{\xi}_t = \tilde{\mathbf{f}}(\mathbf{q}_t,\dot{\mathbf{q}}_t,\tau_t,\boldsymbol{\xi}_t)$, where $\tilde{f}$ is the new optimistic dynamics model. When these slack variables are treated as virtual controls by MPC, they enable optimistic exploration. Intuitively, they account for uncertainty about the dynamics due to imprecise estimates of the vector Δ. To keep MPC from choosing highly improbable dynamics, the slacks are penalized quadratically during MPC with a penalty of the form $\frac{1}{m}\|\boldsymbol{\xi}_t\|^2$. The magnitude of exploration is controlled by m, which should be proportional to the amount of uncertainty about the current dynamics.[1] Previous work used Bayesian models to accurately estimate this uncertainty [21]. In this work, we simply decrease m as the number of samples N increases. While this approach is somewhat simplistic, we found that it works well in practice. Establishing a formal bound on m in terms of the number of samples N is difficult due to the complexity of the physical model. However, we can roughly estimate this bound by considering a simplified linear-Gaussian model of the dynamics. Given a multivariate Gaussian with mean μ_0 and covariance Σ_0, the variance of the posterior estimate of the mean after the update is given by $(\Sigma_0^{-1} + N\Sigma^{-1})^{-1}$, where Σ is the sample variance [22]. This suggests that, for large N, the variance of the mean decreases roughly as $1/N$ with the number of samples N. For simplicity, we use only a single exploration hyper-parameter c, using $m = \frac{c}{N}$ as an estimate of the uncertainty about the model. This makes it easier to adjust the amount of exploration by tweaking a single

[1] Note that the uncertainty about the model is not the same as dynamics noise. In this work, we assume deterministic dynamics, though stochastic dynamics could also be handled in this framework.

parameter.

This optimistic exploration scheme has the effect that the system is steered into taking actions that either move it toward the goal, or else update the model if the previously chosen path to the goal is incorrect, so that another route is attempted on the next replanning step. In the case of linear dynamics or discrete systems, this optimistic exploration scheme has a number of desirable theoretical properties that make it a good choice [1], [3], [4], [7]. Although such results do not exist for the general continuous nonlinear case, we observed that the optimistic exploration strategy empirically achieves effective exploration in practice.

C. Model Predictive Control

To achieve real-time control for online reinforcement learning, we use a simple and efficient differential dynamic programming (DDP) algorithm to choose locally optimal actions $\mathbf{u} = [\tau, \xi]$, which include both real and virtual controls, with respect to the optimistic dynamics $\tilde{f}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \tau_t, \xi_t) = \ddot{\mathbf{q}}$. The actions are optimized with respect to an augmented cost function of the form $\bar{l}(\mathbf{x}, \mathbf{u}) = l(\mathbf{x}, \tau) + \frac{1}{m}\|\xi_t\|^2$, which includes both the actual cost of the task and the penalty on the virtual controls. We first convert the optimistic forward dynamics into a discrete dynamics equation of the form $\mathbf{x}_{t+1} = \bar{f}(\mathbf{x}_t, \mathbf{u}_t)$ by using a fourth order Runga-Kutta integrator, and then supply these dynamics and cost function to a DDP algorithm, which we summarize in this section for completeness. Once this algorithm determines a sequence of locally optimal actions, we extract τ from the first action and apply this control to the system.

The optimal control problem we aim to solve can be formulated as

$$\min_{\mathbf{x}_{1:T}, \mathbf{u}_{0:T-1}} \quad \sum_{t=0}^{T-1} \bar{l}(\mathbf{x}_t, \mathbf{u}_t) \tag{2a}$$

$$\text{subject to:} \quad \mathbf{x}_t = \bar{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}),\ \forall t \in 1, \ldots, T \tag{2b}$$

The goal is to find the set of controls $\mathbf{u}_{0:T-1}$ that minimizes the cost function starting from the current state $\mathbf{x}_0$. We use a variant of DDP called iterative LQR (iLQR), which requires only a first order expansion of the dynamics [29]. This method is particularly fast, making it well suited for MPC. The rest of this section summarizes this method. The algorithm iteratively computes first order expansions of the dynamics and second order expansions of the cost around the current trajectory, and then analytically computes the sequence of optimal controls with respect to this approximation. This sequence of controls is then executed to obtain a new trajectory, and the process repeats until convergence or for a fixed number of iterations. The controls are computed by a dynamic programming procedure that consists of recursively updating the value function and Q-function, defined as

$$V_t(\mathbf{x}_t) = \min_{\mathbf{u}_{t:T-1}} \sum_{i=t}^{T-1} \bar{l}(\mathbf{x}_i, \mathbf{u}_i)$$

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \bar{l}(\mathbf{x}_t, \mathbf{u}_t) + V_{t+1}(\bar{f}(\mathbf{x}_t, \mathbf{u}_t)).$$

Under the LQR assumptions, both of these functions are quadratic, and can be expressed up to a constant as

$$V_t(\mathbf{x}_t) = \frac{1}{2}\mathbf{x}_t^\intercal V_{xx,t}\mathbf{x}_t + \mathbf{x}_t^\intercal V_{x,t}$$

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix}^\intercal \begin{bmatrix}Q_{x,t} & Q_{xu,t} \\ Q_{ux,t} & Q_{uu,t}\end{bmatrix}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix}^\intercal \begin{bmatrix}Q_{x,t} \\ Q_{u,t}\end{bmatrix}.$$

Let $\bar{l}_{x,t}, \bar{l}_{u,t}, \bar{l}_{xx,t}, \bar{l}_{ux,t}, \bar{l}_{uu,t}$ denote the first and second derivatives of the cost function $\bar{l}(\mathbf{x}_t, \mathbf{u}_t)$, and $\bar{f}_{x,t}, \bar{f}_{u,t}$ denote the derivatives of the discretized dynamics. The coefficients can be written as a recurrence described by

$$\begin{aligned}
Q_{x,t} &= \bar{l}_{x,t} + \bar{f}_{x,t}^\intercal V_{x,t+1} \\
Q_{u,t} &= \bar{l}_{u,t} + \bar{f}_{u,t}^\intercal V_{x,t+1} \\
Q_{xx,t} &= \bar{l}_{xx,t} + \bar{f}_{x,t}^\intercal V_{xx,t+1}\bar{f}_{x,t} \\
Q_{uu,t} &= \bar{l}_{uu,t} + \bar{f}_{u,t}^\intercal V_{xx,t+1}\bar{f}_{u,t} \\
Q_{ux,t} &= \bar{l}_{ux,t} + \bar{f}_{u,t}^\intercal V_{xx,t+1}\bar{f}_{x,t} \\
V_{x,t} &= Q_{x,t} - Q_{ux,t}^\intercal Q_{uu,t}^{-1} Q_{u,t} \\
V_{xx,t} &= Q_{xx,t} - Q_{ux,t}^\intercal Q_{uu,t}^{-1} Q_{ux,t}.
\end{aligned}$$

With this recurrence, we can obtain the optimal policy $g(\mathbf{x}_t) = \hat{\mathbf{u}}_t + \mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t)$, where $\mathbf{k}_t = -Q_{uu,t}^{-1}Q_{u,t}$ is the open loop term and $\mathbf{K}_t = -Q_{uu,t}^{-1}Q_{ux,t}$ is the closed loop feedback gain term. Because we are using an MPC framework, we only execute a small portion of the converged optimal control policy. This makes it very convenient to use the previous found solution as a warm-start, which allows for fast convergence.

One final detail in this framework is that, in the early stages of learning, the estimate of the model parameters $\hat{\Delta}$ may be too inaccurate to perform stable forward and backward passes with MPC. If we detect that the forward pass diverges, we revert to a simple double-integrator dynamics model. Typically, this stage of learning lasts less than one second.

IV. Experiments

We evaluated our method on a number of standard robotic control benchmarks: the pendulum, cartpole, and double pendulum, as shown in Figure 3, as well as on a 7 degree of freedom arm, shown in Figure 5. For each system, our method obtains a noisy observation of the features $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]^\intercal$, where the noise is additive and drawn from a zero-mean Gaussian, and is tasked with reaching a target state as quickly as possible.

A. Benchmark Tasks

The pendulum, cartpole, and double pendulum benchmarks require controlling an underpowered system to swing up and place the endpoint of the last link at the target position. Control limits prevent each system from swinging up by continuous application of the same torque, requiring long-horizon planning. The cost function for our method consists of the distance between the endpoint of the last link and the target, as well as terms to penalize large velocities and controls. To impose control limits, we pass the controls

	pendulum	cartpole	double pendulum
DDP with known dynamics	3.04 ± 0.89s	7.44 ± 3.26s	3.7 ± 0.89s
our method	3.28 ± 1.17s	8.31 ± 3.15s	4.98 ± 1.83s
optimism-driven exploration [21]	3.9 ± 1s	10 ± 3s	17 ± 7s
Boedecker et al. [9]	—	12-18s	—
PILCO [13]	12s	17.5s	50s

Fig. 2. Interaction time required to successfully learn each benchmark task for our method, DDP with known dynamics, and the best prior methods.

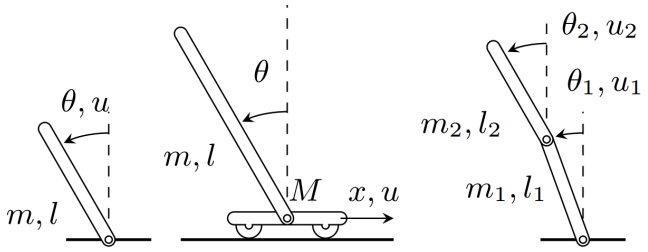

Fig. 3. Benchmark tasks: Pendulum (left), cartpole (center), double pendulum (right)

from DDP through a squashing function of the form $s(u) = 2c\,(\sigma(u) - 0.5)$, where $\sigma(.)$ is the logistic function and c is the control limit. The cost function therefore has the form

$$l(\mathbf{x}_t, \mathbf{u}_t) = \sum_{t=0}^{T} \sqrt{(p(\mathbf{x}_t) - \mathbf{p}^*)^\intercal Q_p (p(\mathbf{x}_t) - \mathbf{p}^*) + \alpha} + \frac{1}{2}\left[\mathbf{x}_t^\intercal Q_v \mathbf{x}_t + s(\mathbf{u}_t)^\intercal R s(\mathbf{u}_t) + \mathbf{u}_t^\intercal P \mathbf{u}_t\right],$$

where the first term is a Huber-like loss on the distance to the target endpoint position $\mathbf{p}^*$, Q_p is a diagonal weight matrix, and α is a smoothing constant. The velocity cost is weighted by a diagonal weight matrix Q_v, and the controls are penalized both after squashing under R and before the squashing, under P, as recommended in prior work [30]. Success at each task required reducing the distance between $p(\mathbf{x}_t)$ and $\mathbf{p}^*$ to less than 0.05 units.

The regressor matrix for the cartpole and the double pendulum systems was obtained by manually factoring the equations of motion into $H(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\Delta = \tau$, while the regressor matrix for the pendulum was obtained automatically using SymPyBotics [26]. Further details about each system are presented in Appendix A.

B. Benchmark Comparisons

The results for the pendulum, cartpole, and double pendulum tasks are shown in Table 2. The most sample-efficient previous results on these tasks were obtained using optimism-driven exploration with a Dirichlet process mixture model [21]. However, the computational requirements of this approach prevented it from running in real time, with most tasks running at less than one hundredth of real time. Our proposed method is able to complete each task in real time, by using DDP-based model predictive control and a dynamics model that can be refitted efficiently using least squares. Other state-of-the-art prior methods shown in our results table include PILCO, which uses an episodic formulation instead of learning online and therefore runs comfortably in real time [13], as well as Boedecker et al. [9], which uses Gaussian processes with MPC. We also include the time to completion for DDP using the true dynamics for each task, to provide a lower bound on the possible time to completion.

Our method achieves the best sample efficiency on each of the benchmark tasks. In fact, the time to completion on each task is very close to the time attained by DDP with known dynamics, indicating that our approach is able to identify a sufficiently accurate model of the system extremely rapidly. The advantage of our approach increases with system complexity, with the more complex double pendulum task attaining a time to completion that more than three times faster than the previous best approach [21], and ten times faster than the previous best approach that can run in real time [13]. Furthermore, unlike the previous optimism-driven method [21], the computational cost of our approach is well within the bounds required for real-time operation. The average wallclock computation time for each benchmark is shown below:

pendulum	cartpole	double pendulum
2.67 ± 1.06s	6.70 ± 4.49s	3.98 ± 1.66s

C. 7 Degree of Freedom Arm

Since the dynamics features for our method can be constructed automatically, for example by using the SymPyBotics package [26], we can extend it to more complex tasks that are representative of real-world robotic control problems. To evaluate this capability, we tested our method on a simulated Barrett WAM 7 degree of freedom arm. The goal of the task was to reach a target pose with zero velocity, starting with no prior knowledge about the physical parameters of the system, other than the dynamics features. Ten target poses were selected at random from a spherical Gaussian distribution with a covariance of 1, centered in the middle of the joint limits. A trial was considered complete when the L_∞ distance to the target pose was less than 0.05, and the velocity L_∞ norm was less than 0.1. The cost function for this task had the form

$$l(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2}\left[(\mathbf{x}_t - \mathbf{x}_t^*)^\intercal Q (\mathbf{x}_t - \mathbf{x}_t^*) + s(\mathbf{u}_t)^\intercal R s(\mathbf{u}_t) + \mathbf{u}_t^\intercal P \mathbf{u}_t\right],$$

where Q was set to be $20I$ for the velocities and $50000I$ for the joint angles. We chose $R = \text{diag}(0.08, 0.00004, 0.12, 0.04, 0.04, 0.04, 0.04)$, to account for the fact that the bigger shoulder pan joint needed to apply larger torques to raise the arm, and set $P = R/100$. We used torque limits of

target pose:	1	2	3	4	5
DDP with known dynamics	1.43 ± 0.03s	1.64 ± 0.02s	1.34 ± 0.02s	2.68 ± 0.84s	1.57 ± 0.03s
our method	5.84 ± 2.76s	9.11 ± 3.4s	10.9 ± 4.62s	9.14 ± 6.22s	3.61 ± 1.12s
target pose:	6	7	8	9	10
DDP with known dynamics	2.05 ± 0.0s	0.35 ± 0.09s	1.9 ± 0.0s	2.65 ± 0.0s	4.98 ± 3.32s
our method	6.15 ± 2.64s	4.6 ± 2.35s	3.71 ± 1.34s	7.77 ± 2.36s	9.99 ± 4.49s

Fig. 4. Results for ten randomly chosen target poses for 7 DoF arm for DDP with the true dynamics and our method, which learned the dynamics online from system interaction.

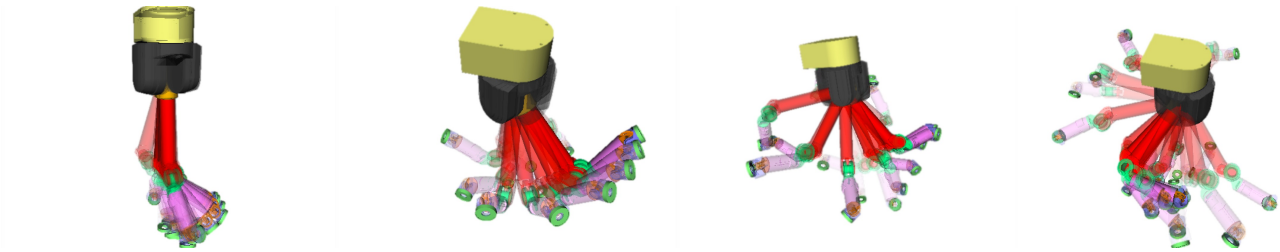

Fig. 5. Sample trajectories for the 7 DoF Barrett arm. The target pose is opaque, and the preceding poses become progressively more translucent. Each image shows an entire trajectory executed by our method, with poses sampled at 0.25 second intervals.

$[\pm 77.3, \pm 160.6, \pm 95.6, \pm 29.4, \pm 11.6, \pm 11.6, \pm 2.7]$, and $v_c = 20$ Hz, $v_s = 100$ Hz. The observation noise was set to have a standard deviation of 0.014 by analyzing the encoder accuracy of the Barrett WAM.[2]

The results of these experiments are shown in in Table 4. The results indicate that our method was successfully able to move the arm into the desired pose in each experiment. Although some of the target poses required more time to acquire a sufficiently accurate dynamics model, some of the targets were reached in time that was only 2-3 times slower than DDP with known dynamics. In all cases, the computation time required to find a solution was comparable to the interaction time, indicating that our method could run in real time.

Note that the dynamics model of this three-dimensional 7 degree of freedom arm is much more complex than any of the benchmarks in the previous section, and the weights Δ had a dimensionality of 70, compared to less than 10 for the benchmark tasks. Several sample trajectories obtained using our method are shown in Figure 5.

V. Discussion and Future Work

We presented a model-based reinforcement learning method that combines ideas from model identification, optimistic exploration, and model predictive control to quickly and efficiently learn to perform robotic control tasks under unknown dynamics. The key idea in our work is to combine efficient linear models of the dynamics, which are informed by domain knowledge of articulated physical systems, with optimism-driven exploration. The features for these linear models can be obtained automatically from the morphology of the robot, and the optimism-driven exploration can be performed using model predictive control.

[2]As with most encoders, position readings are substantially more accurate than velocity readings. For simplicity, we set the observation noise to correspond to the less accurate velocity readings for all entries of the state.

Our method achieves state-of-the-art sample efficiency on standard benchmark problems, including the pendulum, the double pendulum, and the cartpole tasks. Furthermore, unlike our previous MPC-based exploration method, which used a statistical model of the dynamics [21], our method achieves real-time performance, making it feasible for online reinforcement learning on real robotic platforms. Unlike the prior methods in our comparison, our approach leverages additional domain knowledge to greatly simplify the model learning problem. This prior knowledge is encapsulated in the dynamics features, which can be linearly combined to obtain the true dynamics. While this makes the comparison somewhat unequal, it serves to illustrate an important point in model-based reinforcement learning for robotic control: using freely available prior knowledge about the physical system can dramatically simplify the model learning and control problem. The prior knowledge we use is trivial to obtain for most robotic systems, since it consists of the number and connectivity of the robot's links, information that can be easily gathered from a cursory examination.

A number of future directions should be explored to make such applications effective and practical. First, although we demonstrate state-of-the-art results in simulation and evaluate some simulated unmodeled effects, we did not evaluate the robustness of our approach to unmodeled effects on a real system. The least-squares model identification procedure we use has been applied to real robotic systems in the past [16], so there is reason to believe that robustness to unmodeled effects may already be adequate. However, an interesting avenue for future work would be to combine our linear models with more expressive statistical models that can account for unmodeled effects, similar to prior work on autonomous vehicle control [2], [24].

We demonstrated our method on a variety of articulated systems, but the approach is general enough to apply to other kinds of robotic systems also, including autonomous vehicles

and aircraft. In fact, our prior work already demonstrated that optimism-driven exploration can achieve impressive results on simulated helicopter control, recovering from an engine failure with auto-rotation [21]. Extension of our proposed method to such applications requires only a method for constructing the corresponding dynamics features, which can be obtained from analyzing the equations of motion of the system. This can enable a range of applications in future work.

REFERENCES

[1] Y. Abbasi-Yadkori, C. Szepesvári, S. Kakade, and U. V. Luxburg, "Regret bounds for the adaptive control of linear quadratic systems," in *Proc. of the 24th Annual Conference on Learning Theory*, 2011.

[2] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems*, 2006.

[3] P. L. B. Ambuj Tewari, "Optimistic linear programming gives logarithmic regret for irreducible MDPs," in *Proc. of Neural Information Processing Systems Conference (NIPS)*, 2007.

[4] M. Araya, O. Buffet, and V. Thomas, "Near-optimal BRL using optimistic local transitions," in *Proc. Int. Conf. on Machine Learning (ICML)*, ser. ICML '12, 2012, pp. 97–104.

[5] B. Armstrong, "On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics," *Int. Journal of Robotics Research*, vol. 8, no. 6, pp. 28–48, 1989.

[6] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[7] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration-exploitation tradeoff using variance estimates in multi-armed bandits," *Theoretical Computer Science*, vol. 410, no. 19, pp. 1876–1902, 2009.

[8] J. Boedecker, J. Springenberg, J. Wulfing, and M. Riedmiller, "Approximate real-time optimal control based on sparse gaussian process models," in *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2014.

[9] J. Boedecker, J. T. Springenberg, J. Wulfing, and M. Riedmiller, "Approximate real-time optimal control based on sparse gaussian process models," in *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium on*. IEEE, 2014, pp. 1–8.

[10] E. F. Camacho and C. B. Alba, *Model predictive control*, 2013.

[11] M. Cutler, T. Walsh, and J. How, "Reinforcement learning with multi-fidelity simulators," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3888–3895.

[12] M. Cutler and J. P. How, "Efficient reinforcement learning for robots using informative simulated priors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[13] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning*, L. Getoor and T. Scheffer, Eds. Bellevue, Washington, USA: Omnipress, 2011, pp. 465–472.

[14] M. Gautier and W. Khalil, "Exciting trajectories for the identification of base inertial parameters of robots," *Int. Journal of Robotics Research*, vol. 11, no. 4, pp. 362–375, 1992.

[15] M. Gautier, S. Briot, and G. Venture, "Identification of consistent standard dynamic parameters of industrial robots," in *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, 2013, pp. 1429–1435.

[16] J. Hollerbach, W. Khalil, and M. Gautier, "Model identification," in *Springer Handbook of Robotics*. Springer, 2008, pp. 321–344.

[17] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*, ser. Modern analytic and computational methods in science and mathematics. American Elsevier Pub. Co., 1970. [Online]. Available: http://books.google.com/books?id=tA-oAAAAIAAJ

[18] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotic Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[19] S. Kuindersma, R. Grupen, and A. Barto, "Variational bayesian optimization for runtime risk-sensitive control," in *Robotics: Science and Systems (RSS)*, 2013.

[20] L. Ljung, *System identification*. Springer, 1998.

[21] T. Moldovan, S. Levine, M. Jordan, and P. Abbeel, "Optimism-driven exploration for nonlinear systems," in *International Conference on Robotics and Automation (ICRA)*, 2015.

[22] K. P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," *def*, vol. 1, p. 16, 2007.

[23] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2677–2682.

[24] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments." in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014.

[25] W. Rackl, R. Lampariello, and G. Hirzinger, "Robot excitation trajectories for dynamic parameter estimation using optimized b-splines," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 2042–2047.

[26] C. D. Sousa and R. Cortesao, "Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach," *Int. Journal of Robotics Research*, vol. 33, no. 6, pp. 931–944, 2014.

[27] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems*, vol. 12, no. 2, pp. 19–22, 1992.

[28] J. Swevers, C. Ganseman, D. B. Tukel, J. De Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Trans. on Robotics and Automation (TRA)*, vol. 13, no. 5, pp. 730–740, 1997.

[29] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[30] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *International Conference on Robotics and Automation (ICRA)*, 2014.

[31] C. Wang, Y. Zhao, C.-Y. Lin, and M. Tomizuka, "Fast planning of well conditioned trajectories for model learning," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 1460–1465.

[32] J. Wu, J. Wang, and Z. You, "An overview of dynamic parameter identification of robots," *Robotics and computer-integrated manufacturing*, vol. 26, no. 5, pp. 414–419, 2010.

APPENDIX

A. Benchmark Details

In this appendix, we present details for each benchmark system in our evaluation.

1) Pendulum: The pendulum system is a simple single link system with these nonlinear dynamics [13]:

$$\mathbf{q} = \theta,$$
$$\ddot{\mathbf{q}} = \frac{u - b\dot{\theta} - \frac{1}{2}mlg\sin\theta}{\frac{1}{4}ml^2 + I},$$

where θ is the joint angle of the link, u is the torque applied at this joint, m and l are the mass and length of the link, respectively, g is the gravity constant $9.81(\mathrm{m/s^2})$, b is the friction coefficient, and I is the moment of inertia around the pendulum midpoint, which is equal to $\frac{1}{12}ml^2$. The parameters are $m = 1$ kg, $l = 1$ m, and constrained $u \in [-3,3]$ N·m. The goal state is $\begin{bmatrix}0 & \pi\end{bmatrix}^\intercal$, which has the pendulum standing up with no velocity.

For the cost function, we used $\alpha = 0.01$, $Q_p = 2I$, $Q_v = \mathrm{diag}([0.005, 0])$, $P = 0.01I$, where I is the identity matrix. Recall that R is a diagonal matrix that penalizes the squashed control and the virtual control, as mentioned before. For the squashed controls, the upper left block of R, which is the matrix of the quadratic penalty, is $0.01I$. We chose $T = 13, \delta = 0.1\mathrm{s}, v_c = 10$ Hz, $v_s = 100$ Hz.

We can rewrite the dynamics in the form given in Eqn. 2:

$$H(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \begin{bmatrix}\ddot{\theta} & \dot{\theta} & \sin\theta\end{bmatrix}$$
$$\Delta = \begin{bmatrix}\frac{1}{3}ml^2 \\ b \\ \frac{1}{2}mgl\end{bmatrix}$$
$$\boldsymbol{\tau} = u$$

2) Cartpole: The cart-pole system is a nonlinear system described by the following dynamics [13]:

$$\mathbf{q} = [\theta \quad x]^\intercal$$
$$\ddot{\mathbf{q}} = \begin{bmatrix}\frac{-3ml\dot{\theta}^2\sin\theta\cos\theta - 6(M+m)g\sin\theta - 6(u-b\dot{x})\cos\theta}{4l(M+m)-3ml\cos^2\theta} \\ \frac{2ml\dot{\theta}^2\sin\theta + 3m_2g\sin\theta\cos\theta + 4u - 4b\dot{x}}{4(M+m)-3ml\cos^2\theta}\end{bmatrix}.$$

The state space is 4D and the control is 1D, which is the external force applied to the cart. M denotes the mass of the cart, m_2 denotes the mass of the pole, l denotes the length of the pole, θ denotes the angle of the pendulum, p denotes the position of the cart, b denotes the friction between the cart and the ground, and $g = 9.8$ (m/s^2) is acceleration due to gravity. We chose $M = .5$ kg, $m_2 = .5$ kg, $l = .5$ m, $b = .1$ N/m/s, and constrainted $u \in [-10, 10]$ $N \cdot m$. The goal state is $\begin{bmatrix}0 & 0 & \pi & 0\end{bmatrix}^\intercal$, which has the cartpole standing up at the origin with no velocity.

For the cost function, we used $\alpha = 0.1$, $Q_p = \text{diag}([1, 20])$, $Q_v = \text{diag}([0.07,\ 0.03,\ 0,\ 3])$, $P = 0.01I$. The upper left block of R, for the squashed controls, is $0.01I$. We chose $T = 8, \delta = 0.1\text{s}, v_c = 16.7$ Hz, $v_s = 50$ Hz.

Since the cartpole is underactuated, we moved a term to the right hand side of the dynamics to replace the zero due to the unactuacted degree of freedom. We can rewrite the dynamics in the form given in Eqn. 2:

$$H(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \begin{bmatrix}\ddot{p} & \ddot{\theta}\cos\theta & \dot{\theta}^2\sin\theta & \dot{p} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddot{p}\cos\theta & \ddot{\theta}\end{bmatrix}$$
$$\Delta = [M+m, \frac{1}{2}ml, -\frac{1}{2}ml, b, 3, 2l]^\intercal$$
$$\boldsymbol{\tau} = [u, -3g\sin\theta]^\intercal$$

3) Double Pendulum: The double pendulum system is a fully actuated two link system with applied torques at the joints. The system dynamics are:

$$\mathbf{q} = [\theta_1 \quad \theta_2]^\intercal$$
$$\begin{bmatrix}l_1^2(\frac{1}{4}m_1+m_2)+I_1 & \frac{1}{2}m_2l_2l_1\cos(\theta_1-\theta_2) \\ \frac{1}{2}l_1l_2m_2\cos(\theta_1-\theta_2) & \frac{1}{4}m_2l_2^2+I_2\end{bmatrix}\begin{bmatrix}\ddot{\theta} \\ \ddot{\theta}\end{bmatrix} =$$
$$\begin{bmatrix}-l_1\left(\frac{1}{2}m_2l_2\dot{\theta}_2^2\sin(\theta_1-\theta_2) - g\sin\theta_1(\frac{1}{2}m_1+m_2)\right)+u_1 \\ \frac{1}{2}m_2l_2\left(l_1\dot{\theta}_1^2\sin(\theta_1-\theta_2)+g\sin\theta_2\right)+u_2\end{bmatrix}$$

Here, θ_1 and θ_2 are the joint angles, m_1 and m_2 are the masses of link 1 and link 2, respectively. l_1, l_2 are the lengths of the links, g is the gravity constant, I_1 and I_2 are the moments of inertia of the links, and u_1, u_2 are the torques applied at the joints. We chose $m_1 = m_2 = 0.5$ kg and $l_1 = l_2 = 0.5$ m. u_1, u_2 were constrained to be in the range $[-2, 2]$ N·m. To compute the forward dynamics, we solve this linear equation for the second derivative of the joint angles. The goal state is $\begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^\intercal$, which has the double pendulum standing up with no velocity.

For the cost function, we used $\alpha = 0.05$, $Q_p = 5I$, $Q_v = \text{diag}([0.04,\ 0.04,\ 0,\ 0])$, $P = 0.01I$, where I is the identity matrix. The upper left block of R, for the squashed controls, is $0.01I$. In order to make the system stabilize near the goal, we increased the control penalty to 0.1 when the system was near the goal. We chose $T = 8, \delta = 0.08\text{s}, v_c = 16.7$ Hz, $v_s = 50$ Hz.

We can rewrite the dynamics in the form given in Eqn. 2:

$$H(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \begin{bmatrix}A & \mathbf{0} \\ \mathbf{0} & B\end{bmatrix}$$
$$A = \begin{bmatrix}\ddot{\theta}_1 & \ddot{\theta}_2\cos(\theta_1-\theta_2) & \dot{\theta}_2^2\sin(\theta_1-\theta_2) & \sin\theta_1\end{bmatrix}$$
$$B = \begin{bmatrix}\ddot{\theta}_1\cos(\theta_1-\theta_2) & \ddot{\theta}_2 & \dot{\theta}_1^2\sin(\theta_1-\theta_2) & \sin\theta_2\end{bmatrix}$$
$$\Delta = \begin{bmatrix}l_1^2(\frac{1}{4}m_1+m_2)+I_1 \\ \frac{1}{2}m_2l_2l_1 \\ \frac{1}{2}m_2l_2l_1 \\ -gl_1(\frac{1}{2}m_1+m_2) \\ \frac{1}{2}m_2l_2l_1 \\ \frac{1}{4}m_2l_2^2+I_2 \\ -\frac{1}{2}m_2l_2l_1 \\ -\frac{1}{2}m_2l_2g\end{bmatrix}$$
$$\boldsymbol{\tau} = [u_1, u_2]^\intercal.$$