Control Through Contact using Mixture of Deep Neural-Net Experts

Aykut C. SATICI

***Abstract*—We provide a data-driven control design framework for hybrid systems, with a special emphasis on contact-rich robotic systems. These systems exhibit continuous state flows and discrete state transitions, which are governed by distinct equations of motion. Hence, it may be impossible to design a single policy that can control the system in all modes. Typically, hybrid systems are controlled by multi-modal policies, each manually triggered based on observed states. However, as the number of potential contacts increase, the number of policies can grow exponentially and the control-switching scheme becomes too complicated to parameterize. To address this issue, we design contact-aware data-driven controllers given by deep-net mixture of experts (MoE). This architecture automatically learns switching-control scheme that can achieve the desired overall performance of the system, and a gating network, which determines the region of validity of each expert, based on the observed states.**

***Index Terms*—Robotics; nonlinear control; machine learning.**

I. Introduction

A power grasp [1] involves firmly grasping the object on many sides, often with large contact forces, to immobilize the object relative to the gripper; immobilization simplifies object manipulation. This approach is widespread in factories and warehouses.

In household or agricultural settings, several tasks cannot be solved by power grasps. The reasons for this limitation include the object being fragile (fruit picking, contact with humans), constraints on its motion (transporting a bowl of soup that should not be tilted), or size (moving an exercise ball or large box). In effect, the gripper can no longer apply a net wrench (force-torque pair) in any direction on the object, and so the object is no longer guaranteed to move rigidly with the gripper. This scenario, known as *nonprehensile manipulation*, is critical for many tasks humans easily perform such as buttoning a shirt, making a salad, or spreading peanut butter on toast, but cannot be robustly performed by state-of-the-art robots.

Successful task execution under nonprehensile manipulation requires complex reasoning about how wrenches applied on the object interact with its natural dynamics. This reasoning is challenging to carry out due to the multiplicity of possible contact modes and the uncertain contact dynamics governing each mode. Recent advances automate the reasoning over multiple modes, resulting in high-quality plans for achieving a task using a sequence of contact modes [2]. However, to achieve computational tractability, these methods simplify the robot-object dynamics by assuming that the robot-and-object are either always in equilibrium (quasi-static), or moving with constant velocity (quasi-dynamic), which constrain the versatility of nonprehensile manipulation. Moreover, the execution of this sequence typically relies on applying low-level controllers designed for single-mode tasks.

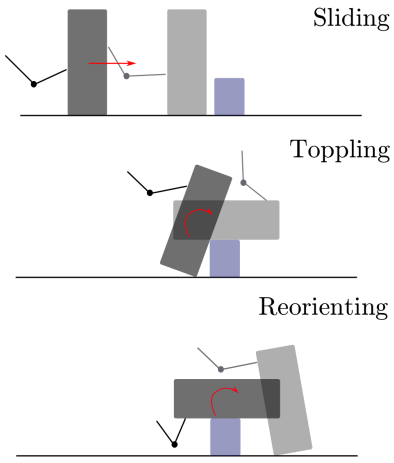


Fig. 1: Nonprehensile manipulation of a box via primitives

Another approach to nonprehensile manipulation is splitting tasks into *manipulation primitives* [3], [4], [5], such as rolling, sliding, throwing or toppling, where each primitive has a corresponding dynamics and is assigned its own controller. Consider the task of moving a box past an obstacle using a series of primitives such as sliding, toppling and reorienting as shown in Figure 1. Each primitive has a region of applicability in the state space, where the dynamics of that primitive describes the flow of the system [6]. Thus, manipulation planning involves identifying a successful primitive sequence, such as the order of primitives shown in Figure 1, and stabilizing the system under each primitive [7], [8]. This approach can be viewed as partitioning the state space and allocating a control law in each subdomain that results in a successful transition to the desired primitive until the goal state is reached[9]. However, the task of ordering the primitives and identifying the distinct controllers in each state partition is done manually [7], [9], and the control design problem is handled in a case-by-case basis.

Data-driven methods to nonprehensile object manipulation have been proposed in several recent works such as [10], which utilizes reinforcement learning ideas, [11], which leverages the recent advances on diffusion models. These approaches

Aykut C. Satici is with the Mechanical and Biomedical Engineering Department, Boise State University, Boise, ID, 83725 USA e-mail: aykutsatici@boisestate.edu
Manuscript received Jul 13, 2024; accepted Jan 3, 2025.
DOI: 10.17694/bajece.1515854

are promising, the former suffering from typical reinforcement learning pitfalls, such as sample inefficiency and the latter typically requiring an imitation trajectory to learn from.

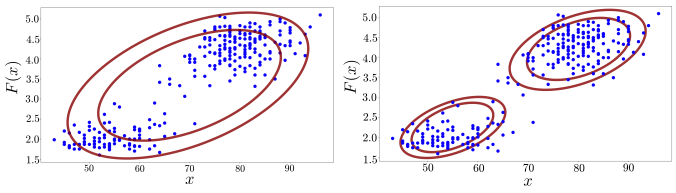
In this work, we present a data-driven approach for constructing dynamic motion plans and stabilizing control laws for complex locomotion and manipulation tasks that make and break contact. Our framework leverages the *mixture of experts* architecture from supervised learning to infer multi-modal controllers for contact-rich systems. This approach *automatically* learns the boundaries of the state partitions and allocates the appropriate expert controller to each partition in order to drive the system to the desired mode, and overall to the goal state. We demonstrate the efficacy of this technique on the swing-up task of the cartpole enclosed by wall barriers, both in simulation and real-world experiments.

II. Background: Mixture of Experts

The mixture of experts (MoE) framework is a technique primarily used to learn an ensemble of regression models (experts) that best fit high variance or multi-modal datasets, such as the one shown in Figure 2a[12]. This technique provides a way to train several specialized expert models simultaneously, where each expert is well curated for a cluster of datasets as seen in Figure 2b. The MoE architecture uses a routing function called a *gating network* to allocate the appropriate local expert for each input data[13]. The objective is to learn the parameters of each local experts and the gating network to best fit the dataset.

Let $F(x;\theta)$ denote a collection of N_F expert models $F(x;\theta) := \{F_1(x;\theta_1), \dots, F_{N_F}(x;\theta_{N_F})\}$, whose parameters are given by the set $\theta = \{\theta_1, \dots, \theta_{N_F}\}$. The gating network is responsible for dividing the input space $\mathcal{X} \subset \mathbb{R}^m$ into *state partitions*, and assigning local expert models capable of providing specialized predictions for each partition. We represent the gating network with the discrete probability distribution $\mathbf{P}(x|\psi) := (P_1(x|\psi), \dots, P_{N_F}(x|\psi))$, where $P_i(x|\psi)$ denotes the probability of state x belonging to the state partition $\mathcal{X}^i \subset \mathcal{X}$ with the index $i \in \{1, \dots, N_F\}$. In the standard MoE framework[14], the prediction $u(x)$ of the MoE is given by

$$u(x) = \sum_{i=1}^{N_F} F_i(x;\theta_i) P_i(x|\psi), \tag{1}$$



(a) Multi-modal dataset fit with one model (b) Multi-modal dataset fit with MoE

Fig. 2: Comparison of multi-modal dataset fit with one regression model and MoE

which requires evaluating all the experts for each input x. We can reduce the computation cost of (1) by utilizing the output of the single best expert as determined by the gating network[15]

$$u(x) = \{F_a(x;\theta_a) \mid a = \underset{i}{\text{argmax}}\{P_i(x|\psi)\}\}. \tag{2}$$

Model Structure: The expert models and the gating network can take several forms. Gaussian process (GP) models are commonly used in the MoE framework to infer a multi-modal probabilistic model from a small amount of data[13]. Despite the expressive power and tractability of GP experts, the inference procedure requires repeated matrix inversions that scale cubically with the size of the dataset[16]. In order to circumvent the large computational and memory overhead while also preserving the expressive power of GP experts, we leverage the universal approximation capabilities of neural networks for both the experts and the gating network. For regression problems that require the flexibility of nonlinear models, the experts can be given by deep neural-nets with point-estimate parameters, which can be extended to probabilistic models with the use of Bayesian neural networks, whose weights and biases are given by probability distributions[17]. Similarly, the gating network can be given by a neural network $\mathbf{P}(x|\psi) : \mathcal{X} \to \mathbb{R}^{N_F}$ with parameters ψ, and the output corresponds to the vector $[P_1(x|\psi), \dots, P_{N_F}(x|\psi)]$. In order to ensure that the probabilities $P_i(x|\psi)$ over all state partitions i sum to one, we use the SOFTMAX activation function[18] on the last layer of the gating network.

Training: Given the training dataset $\mathbb{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ with N input state-label pairs, we can use gradient-based techniques to find the optimal parameters (ψ, θ) that best fit the dataset[15]. In such techniques, we construct the cost function we wish to minimize as

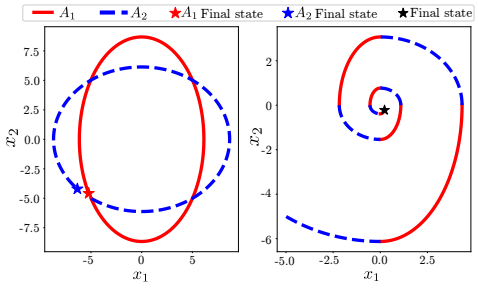
$$\mathbb{L}(\mathbb{D}) = \sum_{j=1}^{N} \sum_{i=1}^{N_F} \|F_i(x_j;\theta_i) - y_j\| \; P_i(x_j, \psi), \tag{3}$$

where $\|F_i(x_j;\theta_i) - y_j\|$ is the error in the prediction made by the expert i. Notice that the cost function (3) is minimum when the parameter θ_i has the lowest prediction error and the highest probability of getting selected by the gating network. So long as the complexity of the experts and the cost function allow for the pertinent gradients $\partial\mathbb{L}/\partial\psi, \partial\mathbb{L}/\partial\theta$ to be evaluated, we can invoke stochastic gradient descent (SGD) to update the decision parameters as follows:

$$\psi \leftarrow \psi - \frac{\partial \mathbb{L}}{\partial \psi},$$
$$\theta \leftarrow \theta - \frac{\partial \mathbb{L}}{\partial \theta}.$$

III. Motivating Application: *Switching Linear System*

In the following discussion, we present an example to motivate and lay the foundation for the use of MoE in the control design problem. In particular, we propose a data-driven technique to automatically seek switching controllers



(a) Two marginally stable closed-loop systems (b) Asymptotically stable switching systems

Fig. 3: Stable switching between two marginally stable systems

for multi-modal systems. Suppose we have two linear systems of the form

$$\begin{aligned} \dot{x} &= A_1 x = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix} x, \\ \dot{x} &= A_2 x = \begin{bmatrix} 0 & -2 \\ 1 & 0 \end{bmatrix} x, \end{aligned} \tag{4}$$

where each system is marginally stable as shown in Figure 3a. Although the individual systems are not asymptotically stable, it is possible to find a state-dependent switching rule that makes the resulting switched system stable[19] (Figure 3b). We aim to learn a gating network $\mathbf{P}(x|\psi)$ to automatically divide the state space into partitions and identify which of the two systems to execute in each state partition, with the goal of asymptotically stabilizing the origin.

Akin to the regression problem in Section II, the training dataset consists of *input state-label* pairs, where the labels are the performances of the trajectories generated under the current control law. In the case of the switching-control problem, we generate a trajectory and the corresponding performance metric (labels) as follows. Starting from some initial state $x(t = 0)$, we sample a state partition index i from the categorical distribution, whose probabilities are provided by the gating network:

$$i \sim \text{Categorical}(\mathbf{P}(x(t)|\psi)).$$

Given the partition index i, the expert (control law) is given by a sample from the Bernoulli probability distribution

$$F_i(\theta_i) = \begin{cases} 0, & \theta_i > \frac{1}{2}, \\ 1, & \theta_i \leq \frac{1}{2}, \end{cases} \tag{5}$$

where $F_i = 0$ corresponds to the first dynamics $\dot{x} = A_1 x$ and $F_i = 1$ corresponds to $\dot{x} = A_2 x$. The parameter θ_i of the expert is to be learned, and it determines which of the two experts to execute in each partition. In order to ensure that the parameter θ_i of the expert serves as the probability of the Bernoulli distribution, we use the SIGMOID function[18] to limit θ_i between 0 and 1. The next state $x(t + \Delta t)$ in the trajectory is obtained from the following integration scheme:

$$x(t + \Delta t) = (1 - F_i)A_1 x(t) + F_i A_2 x(t).$$

We repeat this process to generate a trajectory for the time-horizon T. The performance of the trajectory generated under the current parameters (ψ, θ) can be quantified by the metric ℓ as

$$\ell(x(t + \Delta t)) := \frac{1}{2}\|x(t + \Delta t)\|^2.$$

In Section IV-A, we generalize the performance metrics to be applicable to various dynamical systems and discuss how we can encode desired characteristics of the controller. From the performance metric ℓ, we can construct the cost function $\mathbb{L}$ similar to the standard MoE framework in (3) as

$$\mathbb{L}\Big(\{x(0), \ldots, x(T)\}\Big) = \sum_{t=0}^{T} \sum_{i=1}^{N_F} \ell_i\Big(x_i(t + \Delta t)\Big)\ P_i\Big(x(t), \psi\Big).$$

In the upcoming sections, we generalize the MoE control-search problem and provide techniques to efficiently learn the optimal decision parameters from appropriate cost functions.

IV. METHODS

Based on the motivating example provided in Section III, we present a generalized data-driven control design framework for hybrid dynamical systems. In this framework, the controller is given by deep-net mixture of experts $F(x; \theta)$, and the control switching scheme is governed by the gating network $\mathbf{P}(x|\psi)$. This technique allows us to observe the effects of mode changes from the closed-loop trajectories and learn a switching mechanism to best control the hybrid system across modes. The objective is to learn the parameters θ_i of each expert and the gating network ψ that can achieve the desired performance.

Let $\phi(x_0, u, T)$ denote a closed-loop trajectory generated from a hybrid dynamical model starting from initial state x_0. We represent the dynamics of a hybrid system with the differential inclusions[20]

$$\begin{cases} \dot{x} \in f(x, u), & x \in C, \\ x^+ \in g(x, u), & x \in D, \end{cases} \tag{6}$$

where $x \in \mathbb{R}^m$ is the state vector, and $u \in \mathbb{R}^n$ is the input. The set-valued mappings $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ denote the flow and jump maps, respectively, where C and D are subsets of $\mathbb{R}^m$ consisting of the feasible states under the flow and jump rules, respectively. The notation x^+ indicates the state resulted by the jump rule g.

Remark *Another popular formulation for hybrid systems (e.g. mechanical systems that undergo contact) is through the utilization of measure differential inclusions (MDI)[21], which leads to a more succinct representation in terms of differential measures. This formulation represents equations* (6) *in the form*

$$\mathrm{d}x = f(x, u)\,\mathrm{d}t + \mathrm{d}R,$$

where $\mathrm{d}x$, $\mathrm{d}t$ *and* $\mathrm{d}R$ *represent various measures. The term* $\mathrm{d}R$ *is responsible for the changes in the system's continuous vector fields and state jumps whenever a contact is made or broken. We have used this formulation in Section V-B1 in order to model an example system on which we apply our data-driven controllers for verification. Readers who are interested in the theory of the MDI formulation are referred to the piece[21].*

For every state x in a trajectory, the control law first samples state partition index i from a categorical distribution and evaluates the corresponding expert as

$$u(x;\psi,\theta) = \{F_i(x;\theta_i) \mid i \sim \text{Categorical}(\mathbf{P}(x|\psi))\}. \quad (7)$$

We use the metric $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ to measure the performance of the sampled experts, which we discuss in depth in Section IV-A. The goal is to learn the decision parameters (ψ, θ) that minimize the metric ℓ for all initial states in the state space. We pose the search over the parameters of the experts and the gating network as the following optimization problem.

$$\begin{aligned} \underset{\psi,\theta}{\text{minimize}} \quad & \int_0^T \ell(x(t),u)\,\mathrm{d}t, \\ \text{subject to} \quad & \begin{cases} \dot{x} \in f(x,u), & x \in C, \\ x^+ \in g(x,u), & x \in D, \end{cases} \\ & u = \{F_i(x;\theta_i) \mid i \sim \text{Categorical}(\mathbf{P}(x|\psi))\}. \end{aligned} \quad (8)$$

In Section IV-C, we provide a procedure to solve the optimization problem (8) via stochastic gradient descent.

Remark *Without prior knowledge injected to the gating network, the samples from the categorical distribution in* (7) *initially explore the performance of most, if not all, of the expert controllers. As the parameters converge to their optimal values, the samples from the categorical distribution correspond to the indices of the single best experts and the control law in* (7) *is equivalent to* (2)*.*

A. Performance Metrics

We present two viable choices for the performance metric ℓ.

1) **Accumulated cost**: is the total quadratic loss between the desired state x^* and the states generated under the current control law. We can also enforce control saturations for underactuated systems by incurring a cost on the control input as follows:

$$\ell(x,u) = \frac{1}{2}(x-x^*)^\top \mathcal{Q}(x-x^*) + \frac{1}{2}u^\top \mathcal{R} u, \quad (9)$$

where $\mathcal{Q}$ and $\mathcal{R}$ represent a positive definite and positive semi-definite gain matrices, respectively. This construction encourages trajectories to reach the desired equilibrium with minimum effort and shortest time. We modify the cost function $\mathbb{L}$ presented in (3) to incorporate the quadratic loss $\ell(x,u)$ as follows:

$$\mathbb{L}(\phi) = \sum_{t=0}^{T}\sum_{i=1}^{N_F} \ell_i\Big(x_i(t+\Delta t), F_i\Big) P_i\Big(x(t)|\psi\Big). \quad (10)$$

Similar to the regression problem provided in Section II, the accumulated cost (10) is minimum when the metric ℓ_i achieved by expert i is low and the responsibility $P_i(x(t)|\psi)$ of the expert is high. Notice that the accumulated cost checks the performance of each expert at every state. When training for few experts, this cost function provides ample exploration, resulting in fast convergence to an optimal control strategy. However, for numerous experts, the accumulated cost incurs large computational overhead.

2) **Minimum trajectory loss (MTL)**: is designed to minimize the computational complexities of the accumulated cost. Compared to (10), MTL may also better represent the desired behavior of some dynamical systems. Consider the classical control problem of swinging-up the simple pendulum to the upright equilibrium. For an underactuated pendulum, a successful controller needs to swing the pendulum clockwise and counterclockwise, passing through the downward equilibrium point multiple times until enough kinetic energy is built up to reach the upward equilibrium. Accumulated loss incurs high cost in such scenarios and the control search would get stuck in a local minimum. In such cases, a successful cost function encourages trajectories that *eventually* lead to the goal state. This is achieved by MTL, which is composed of the lowest cost incurred across the entire trajectory and the responsibilities of the experts that led to the minimum cost. The resulting cost function $\mathbb{L}$ is given by

$$\begin{aligned} t_{\min} &= \inf_t \{\ell(x(t),u) : x(t) \in \phi(x_0,u,T)\}, \\ \mathbb{L}(\phi) &= \frac{\ell(x(t_{\min}),u)}{C} \sum_{t=0}^{t_{\min}} P_i(x(t)|\psi), \end{aligned} \quad (11)$$

where $C > 0$ is a normalization factor. Unlike the accumulated cost, MTL does not particularly reward low effort or short time trajectories, but it equally rewards two trajectories as long as they both reach the goal state within the time horizon T.

B. State Sampling

We intend to find a solution to the optimization problem in (8) for all initial states x_0 in the state space. To do so, we compose the performance metric ℓ from *a batch of initial states* and update the parameters (ψ, θ) *iteratively* via stochastic gradient descent (SGD). To efficiently sample the initial states, we use a combination of greedy and explorative state sampling techniques. An example of greedy state sampling technique, commonly known as *Dataset Aggregation* (DAGGER), is a method adapted from imitation learning[22]. This technique collects states most visited under the current parameters (ψ, θ) and concentrates on refining the performance of the controller on these states. In detail, we first discretize the state space and uniformly sample several initial states. Starting from those initial states, we generate trajectories using the current parameters. In order to improve the controller at the states favored by the current policy, we draw N_d initial state samples from the states visited in the trajectories. This efficient state exposition is pivotal for a fast convergence to the optimal parameters.

The explorative state sampling technique exposes the training to the rewards of approaching and remaining close to x^*. It also uses random sampling to explore new control strategies and recover from locally optimal solutions. This method collects N_r initial states around the neighborhood of the desired equilibrium by drawing samples from the normal distribution $x_0 \sim \mathcal{N}(x^*, \Sigma)$, with mean x^* and the variance Σ is kept small. For each parameter update in SGD, we compute

Algorithm 1 Solution to the Optimization Problem (8)

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while !(`is converged`) **do**
  $J \leftarrow 0$ ▷ Average cost function
  for $x_0 \in \mathcal{D}_N$ **do**
    $\mathbb{L}$ = `Performance metric`(x_0, ψ, θ) ▷ Section IV-A
    $J \leftarrow J + \mathbb{L}/N_\mathcal{D}$
  $\theta \leftarrow \theta - \alpha^{\partial J}/_{\partial \theta}$ ▷ SGD step
  $\psi \leftarrow \psi - \alpha^{\partial J}/_{\partial \psi}$
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples (Section IV-B)
  $i \leftarrow i + 1$
return θ

the performance metric as an expectation over $N_\mathcal{D} = N_d + N_r$ samples as follows:

$$J(\phi, u) = \mathbb{E}_{x_0 \in \mathcal{D}_N}[\mathbb{L}(\phi(x_0, u, T))], \tag{12}$$

where $\mathcal{D}_N$ is a *replay buffer* consisting of $N_\mathcal{D}$ initial state samples.

C. Training Mixture of Experts Controller

We solve the optimization problem in (8) following the procedure outlined in Algorithm (1). At the beginning of the training, we collect $N_\mathcal{D}$ initial states samples using the greedy and explorative state sampling techniques discussed in Section IV-B and save them in the replay buffer $\mathcal{D}_N$. For every initial state in the replay buffer, we generate a trajectory using the current decision parameters (ψ, θ) and assign the cost function $\mathbb{L}$. The average cost incurred by the current policy is given by J in (12), from which we compute the pertinent gradients $^{\partial J}/_{\partial \psi}, ^{\partial J}/_{\partial \theta}$ via forward-mode auto-differentiation techniques[23]. We invoke a variant of stochastic gradient descent (SGD), known as ADAM[24] to efficiently update the parameters with adaptive learning rates α. The training is terminated when the average running cost J is below a small threshold for trajectories generated from various initial states.

V. Case Studies

We demonstrate the efficacy of the MoE controller in simulation and real-world experiments. In the first case study, we learn a gating network that switches between two marginally stable closed-loop systems to result in a piecewise-asymptotically-stable system. Then, we find switching MoE controller to swing up the classical cartpole mechanism enclosed with wall barriers.

A. Switching Linear System

We find the stable switching scheme through the MoE framework discussed in Section III. We aim to learn the parameters ψ of the gating network $\mathbf{P}(x|\psi)$ and the expert parameters θ_i such that the switching system converges to the desired equilibrium $x^* = (0, 0)$. The gating network is a fully-connected neural net with one hidden layer (2 input states $\rightarrow$ 6 neurons $\rightarrow$ 4 outputs) and an ELU activation function[25]. We constrain the maximum number of state partitions to 4. Each state partition has a corresponding controller parameter $\theta_i \in \mathbb{R}$.

The response of the learned switching system is shown in Figure 4. Figure 4a shows the single best expert F_a given by (2) in each state partition, where purple corresponds to $F_a = 0$ or $\dot{x} = A_1 x$ and yellow corresponds to $F_a = 1$ or $\dot{x} = A_2 x$. The sample trajectory starts at $x_0 = [-5, -5]$ and successfully converges to the origin shown by the red star. The state partition index of the single best expert is shown in Figure 4b, and it depicts that the training uses only 3 out of the 4 state partitions available. The partitions in Figure 4b matches the analytical solution to the successful stable switching system given by[19]

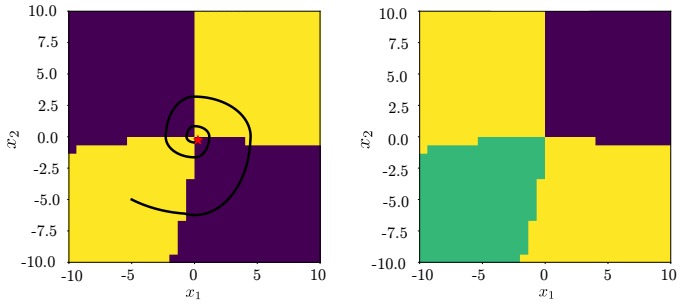
$$\dot{x} = \begin{cases} A_1 x, & x_1 x_2 \leq 0, \\ A_2 x, & x_1 x_2 > 0, \end{cases}$$

where $x = [x_1, x_2]$.

The training progress is shown in Figure 5. The three rows in the figure depict the performance of the training after 0, 200 and 1400 parameter updates, respectively. The sample trajectory in Figure 5a shows that the initial parameters result in unstable switching between the two systems. After only 200 parameter updates, the training finds a stable switching mechanism, but it does not yet converge to the desired equilibrium x^*. In order to create an asymptotically stable system, the corners of each state partition must intersect at the origin, which the training finds successfully after 2000 parameter updates (Figure 4). This is thanks to the explorative state sampling technique, which samples states close to the desired equilibrium, assisting the training in finding the distinct boundaries of each partition at the origin.

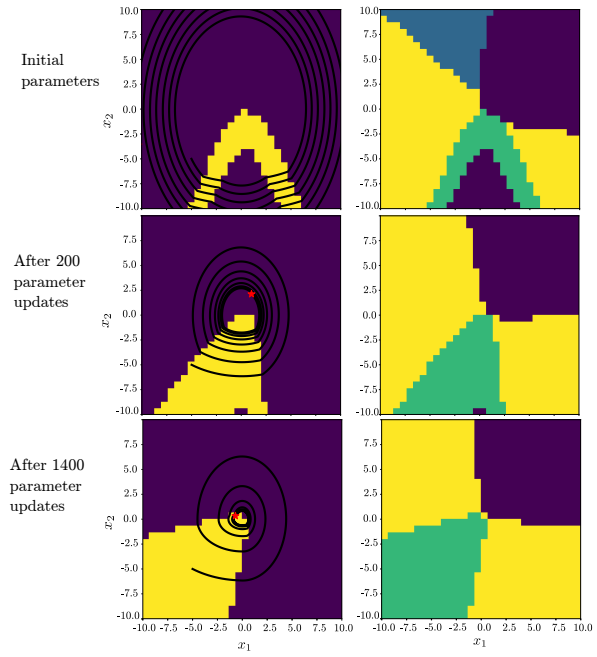
B. Cartpole with Wall Contacts

In this section, we take the classical cartpole swing-up problem and introduce potential contacts from two barriers as shown Figure 6. The potential contacts serve as a way to convert the standard cartpole system into a multi-modal dynamics. The objective is to swing-up the pendulum on the



(a) (b)

Fig. 4: Final stable switching system: (a) The single best expert F_a in each state partition, where purple corresponds to $F_a = 0$ or $\dot{x} = A_1 x$ and yellow corresponds to $F_a = 1$ or $\dot{x} = A_2 x$, (b) State partition index of the single best expert



(a) Control input (b) State partition

Fig. 5: Training progress. The final solution is shown in Figure 4

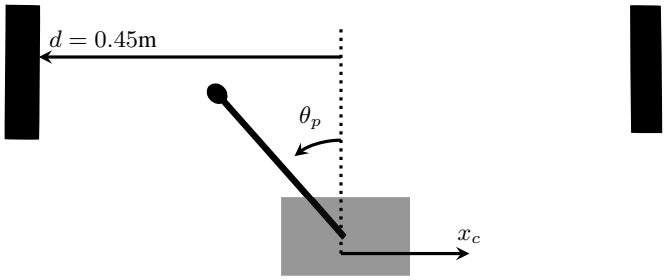


Fig. 6: Cartpole with wall contacts

cart in the presence of contacts and impacts. We apply the MoE framework to train switching expert controller and a gating network that governs the switching scheme. We demonstrate the performance of the MoE controller in simulation and real-world experiments. Lastly, we compare the performance of the MoE controller against a single swing-up controller.

1) System Model: The cartpole system consists of a freely rotating pendulum link hinged on an actuated cart. The setup is enclosed by two rigid walls hanging 0.2m from the bottom of the cart. The objective is to use the control authority on the cart in order to swing-up the pendulum to the upright. The pendulum spans length of $l = 0.31$m and its mass $m_p = 0.75$kg is concentrated at the distance $l_{cm} = 0.2$m from the hinge. The cart alone has a mass of $m_c = 0.165$kg and the moment of inertia of the pendulum is $I_p = 0.0022$ kg·m^2. The viscous friction in the bearings of the cart's wheels is characterized by the coefficient $b = 1.2$ N·sec/m.

Differentiable contact model: In order to infer the MoE controller for a contact-rich mechanism, we generate closed-loop trajectories from an accurate contact model given by measure differential inclusions[21], [26], [27]. The dynamics of the cartpole under impacts, contacts and Coulomb friction is given by

$$\begin{aligned} M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q})\,\mathrm{d}t - \mathrm{d}R = 0, \\ h(q,\dot{q}) := C(q,\dot{q})\dot{q} + G(q) - Bu(q,\dot{q}), \end{aligned} \tag{13}$$

where $q = (x_c, \theta_p)$, x_c is the location of the cart, θ_p is the angle of the pendulum from the vertical. From the Euler-Lagrange formulation, the positive definite mass matrix $M(q)$, the Coriolis and centripetal terms $C(q,\dot{q})$, the gravitational terms $G(q)$, and the input-to-state mapping B are given by

$$\begin{aligned} M(q) &= \begin{bmatrix} m_c + m_p & -m_p l_{cm}\cos(\theta_p) \\ -m_p l_{cm}\cos(\theta_p) & m_p l_{cm}^2 + I_p \end{bmatrix}, \\ C(q,\dot{q}) &= \begin{bmatrix} b & m_p l_{cm}\dot{\theta}_p \sin(\theta_p) \\ 0 & 0 \end{bmatrix}, \\ G(q) &= \begin{bmatrix} 0 & -m_p g l_{cm}\sin(\theta_p) \end{bmatrix}^\top, \\ B &= [1\ 0]^\top, \end{aligned} \tag{14}$$

where g is the acceleration due to gravity. The force measure $\mathrm{d}R$ contains the contact forces as

$$\mathrm{d}R = W_N\,\mathrm{d}\lambda_N + W_T\,\mathrm{d}\lambda_T,$$

where W_N and W_T are the projection matrices that map the effect of the normal and tangential contact forces, respectively, to the generalized coordinates. The vectors $\mathrm{d}\lambda_N$ and $\mathrm{d}\lambda_T$ consist of the normal and tangential contact impulse measures, respectively. In the presence of impacts, we integrate the contact measures over a singleton time t as $\int_{\{t\}}(\mathrm{d}\lambda_N, \mathrm{d}\lambda_T) = (\lambda_N(t), \lambda_T(t))$ in order to obtain the impulsive contact forces. In the case of persisting contact forces, the contact impulse measures evaluate to $(\mathrm{d}\lambda_N, \mathrm{d}\lambda_T) = (\dot{\lambda}_N, \dot{\lambda}_T)$, where $\dot{\lambda}_N$ and $\dot{\lambda}_T$ hold the normal and the tangential contact forces, respectively.

We use Moreau's time-stepping algorithm[27] to time-discretize and integrate the MDI. At each integration step, we compute the contact forces that enforce the geometric and kinematic constraints of rigid-bodies-in-contact via the *linear complementarity formulation*[27]. This formulation presents a linear complementarity problem (LCP) that searches for *contact force and post-impact velocity* pairs that obey the no-penetration conditions of rigid bodies. Without the presence of Coulomb friction, the LCP can be posed as a convex optimization problem, which can be solved analytically and can provide tractable gradients through the solution of the LCP. However, with the consideration of Coulomb friction, the LCP becomes a non-convex optimization problem, which may be intractable. In such scenarios, we solve the LCP through numerical methods, namely *Lemke's algorithm*[28], which allows us to use auto-differentiation techniques to evaluate the pertinent gradients during the training process.

2) Training: We aim to learn the parameters (ψ, θ) of the MoE controller in order to stabilize the cartpole system to the desired state $x^* = (q^*, \dot{q})^* = ((0,0),(0,0))$ under contacts, impacts and Coulomb friction. Once the system reaches within a small neighborhood of x^*, we employ Linear

TABLE I: Structure of the deep-net experts and the gating network.

Neural Network	Inputs	Number of neurons in hidden layers	Outputs
Expert $F_i(x;\theta_i)$	$[x_c, \cos(\theta_p), \sin(\theta_p), \dot{x}_c, \dot{\theta}_p]$	(10, 4)	$u \in \mathbb{R}$
Gating network $\mathbf{P}(x\|\psi)$	$[x_c, \cos(\theta_p), \sin(\theta_p), \dot{x}_c, \dot{\theta}_p]$	(4, 3)	$[P_1, P_2, P_3]$

Quadratic Regulator (LQR) to maintain the system at the desired equilibrium. The structures of the deep-net experts and gating network are provided in Table I. We constrain the maximum number of state partitions to 3, where each partition has a local expert F_i. The output of the experts correspond to the force applied on the cart. We use minimum trajectory loss (MTL) discussed in Section IV-A with time horizon $T = 1.5$s, where the performance metric ℓ is given by (9). In each parameter update, we sample $N_{\mathcal{D}} = 4$ initial states through greedy and explorative techniques.

3) Hardware: We demonstrate the performance of the MoE controller in simulation and hardware. The hardware (Figure 7), designed and built by QUANSER[29], uses a DC-motor to translate the cart on a track. The cart uses a rack-and-pinion mechanism to translate on the track with zero-slip. One of the wheels of the cart is attached to an optical encoder, from which we estimate the position and velocity of the cart. There is also an optical encoder rigidly attached to the pendulum link, reporting its orientation. We evaluate the experts and the gating network in MATLAB/Simulink and pass the corresponding voltage commands to the DC-motor via QUARC, QUANSER'S real-time control software.

Parameter Estimation: We use an adaptation law[30] to estimate the parameters of the hardware in Figure 7, namely the mass m_c of the cart, viscous friction in the bearings of the cart's wheels denoted by b, and the effects of the cable harness on the cart, which we denote by the spring constant k. Consider the continuous dynamics of the cart whose equations of motion are given by the ordinary differential equation

$$m_c\ddot{x}_c + b\dot{x}_c + kx_c = u(x_c, \dot{x}_c), \tag{15}$$

where $u_c(x_c, \dot{x}_c)$ is a control input that is to be determined for tracking. We are uncertain of the constant parameters $\xi = \begin{bmatrix} m_c & b & k \end{bmatrix}^\top$, whose estimates are denoted by $\hat{\xi} \in \mathbb{R}^3$. Let us introduce the errors in the position x_c, and parameters ξ to be

$$\tilde{x}_c = x_c - x_{c_r}, \quad \tilde{\xi} = \xi - \hat{\xi}, \quad e = \begin{bmatrix} \tilde{x}_c & \dot{\tilde{x}}_c \end{bmatrix}^\top,$$

where x_{c_r} is a reference signal for the motion of the mechanical system. Inspired by Spong[30] et al., let us choose the control input according to

$$\begin{aligned} u(x_c, \dot{x}_c) &= Y(x_c, \dot{x}_c, a, v)\hat{\xi} - cr, \\ Y(x_c, \dot{x}_c, a, v) &= \begin{bmatrix} a & v & x_c \end{bmatrix}, \end{aligned} \tag{16}$$

where the quantities v, a, and r are given as

$$\begin{aligned} v &= \dot{x}_{c_r} - \lambda\tilde{x}_c, \\ a &= \dot{v} = \ddot{x}_{c_r} - \lambda\dot{\tilde{x}}_c, \\ r &= \dot{x}_c - v = \dot{\tilde{x}}_c + \lambda\tilde{x}_c, \end{aligned}$$

where $c, \lambda > 0$ are constant gains. Substituting the control law (16) into the system model (15) leads to

$$m\dot{r} + (b + c)r = -Y\tilde{\xi}. \tag{17}$$

The parameter estimate $\hat{\xi}$ may be computed using standard methods of adaptive control such as gradients or least squares. For example, for a positive definite matrix Γ of appropriate dimensions, we can use the gradient update law

$$\dot{\hat{\xi}} = -\Gamma^{-1}Y^\top(x_c, \dot{x}_c, a, v)r. \tag{18}$$

Proposition *The control* (16) *and adaptation law* (18) *stabilize the system to a reference trajectory while the estimates of the parameters tend to their correct values.*

Proof 1 *See Appendix*

We can excite several frequencies by choosing

$$\begin{aligned} x_{c_r}(t) &= A\sin(\omega(t) + \phi) \\ \omega(t) &= \pi\sum_{k=1}^{M_r}\left(1 - \frac{k-1}{M_r}\right)\sin kt \end{aligned}$$

for some constants ϕ, $A > 0$, and a sufficiently large $M_r \in \mathbb{N}$. In Figure 8, we plot the response of the system to the control and adaptation laws (16, 18), implemented in simulation. The constants that are used are as follows: $(A, M_r, \phi) = (3/10, 3, 0°)$, $\Gamma = \text{diag}(1, 1, 1/10)$ and $(c, \lambda) = (1, 4)$. The real mass, damping and stiffness of the system are $(m_c, b, k) = (0.665, 1.819, 0)$ and their estimates start at $(\hat{m}_c, \hat{b}) = (-1/2, -1/4, -1)$.

4) Results: Figure 9 shows a successful swing-up trajectory generated by the MoE controller in simulation and hardware. The blue contours correspond to the level sets of the control input u during impact ($x_c = 0.36m, \dot{x}_c = 0^m/_s$), and the solid red lines depict the boundaries of the state partitions. Although the gating network can provide up to three state partitions, the training converges to utilizing only two. Figure 9 shows that the system successfully avoids contacts during the swing-up phase, which otherwise would have prevented the pendulum from pumping energy from the downward equilibrium. By the time the pendulum approaches the upright equilibrium, it is moving at such high speed ($\sim$-6$^{\text{rad}}/_{\text{s}}$) that LQR cannot stabilize the pendulum to the upright. However, we have observed from several trajectories that the system leverages the impact from

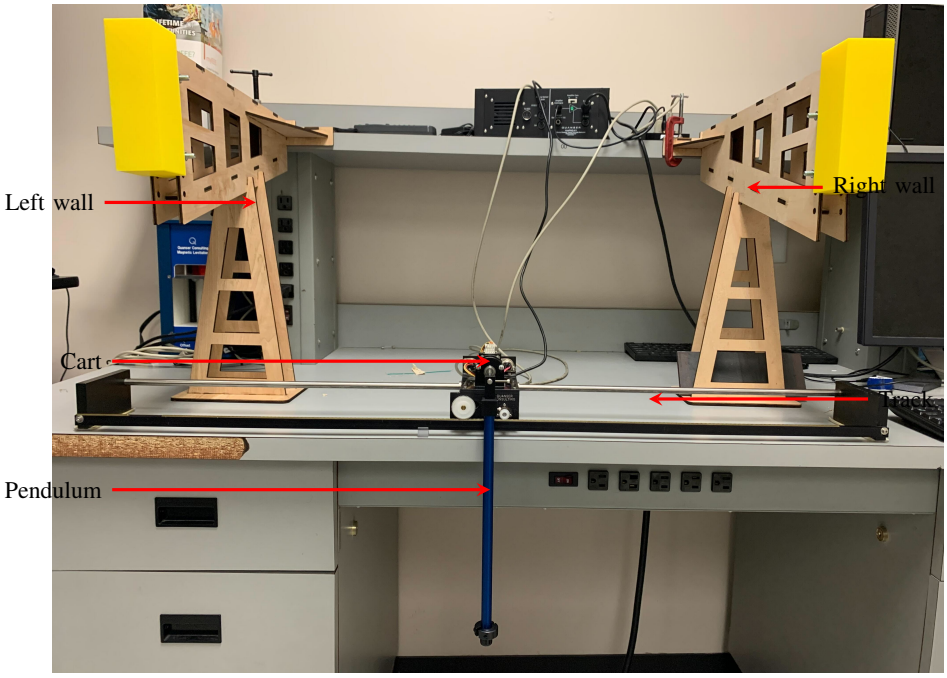


Fig. 7: Experimental setup of cartpole with wall contacts

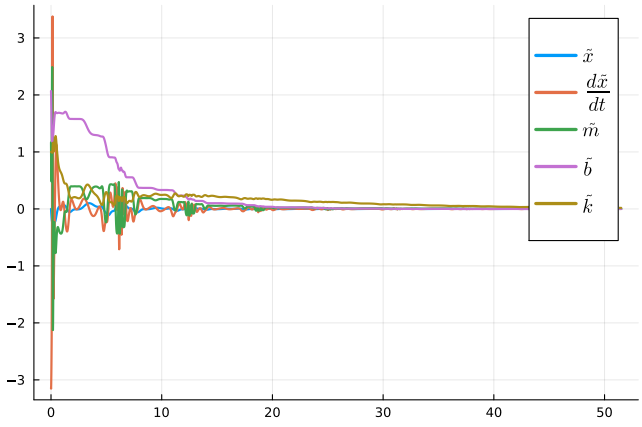


Fig. 8: Simulation showing the convergence of the system state and parameter estimates

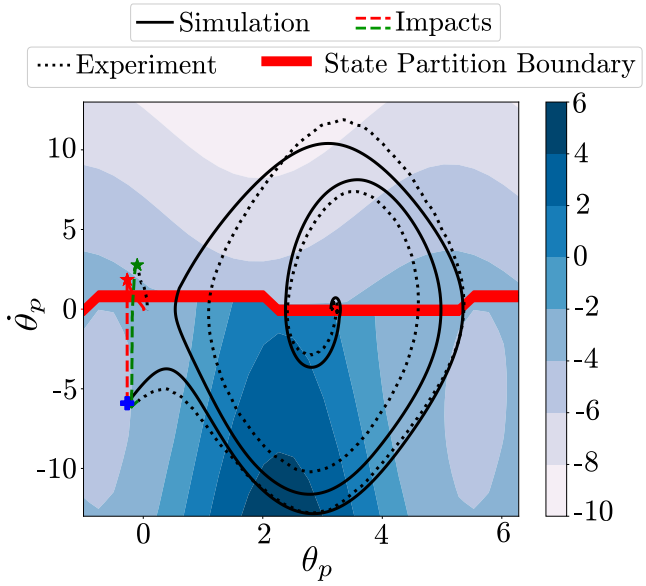
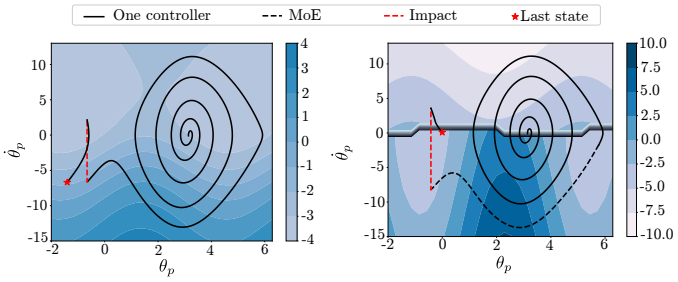


Fig. 9: A sample trajectory starting from downward equilibrium at rest. The blue contours represent the level sets of the control input at the pre-impact and post-impact states.

the wall to lower the speed of the pendulum.

During impact, the control law switches experts, where the new expert applies rapid braking allowing the LQR to catch the pendulum post-impact. The MoE controller achieves successful swing-up in simulation and real-world, proving the accuracy in the contact model and the robustness of the controllers.

Comparison between MoE and single controller: We compare the performance of the MoE controller against a single controller, which can be thought of as the MoE controller with $N_F = 1$. This controller is parameterized by a neural net, with a similar structure to the experts provided in Table I. We train the controller with the same minimum trajectory loss (MTL) and training parameters as the MoE. Once the controller swings the pendulum to the neighborhood of x^*, we use LQR to stabilize it to the upright. As shown in Figure 10a, the single controller successfully swings up the pendulum close to the upright. However, due to the length of the pendulum and the tight distance between the walls, the pendulum inevitably impacts one of the barriers. Unfortunately, the LQR is not able to catch the pendulum post-impact, due to the high velocity of the pendulum. On the other hand, Figure 10b shows the performance of the MoE controller in the same scenario. The MoE solution leverages the switching controllers



(a) Performance of a single controller (b) Performance of the MoE controller

Fig. 10: Comparison between MoE and a single controller

to apply rapid braking post-impact, which significantly lowers the velocity of the pendulum. This assists the LQR in catching the pendulum at the appropriate speed. This demonstrates the advantages of switching controllers in the presence of multi-modal contact-rich systems.

VI. Conclusion

We provide a data-driven control design that reasons about the effects of contact forces on the hybrid system. We incorporate accurate system model in the training via linear complementarity formulation, and infer mixture of experts controller. The learning framework also provides a gating network, which divides the state space into partitions and dedicates a specialized local expert in each partition. From simulation and real-world experiments, we demonstrate that the learned policy leverages the advantages of contact in some states and minimizes its adverse effects in others.

The framework that we propose is general and can be applied to a wide range of hybrid dynamical systems. Almost all robotic systems must interact with their environment in order to be useful: walking or running robots make and break contact with the ground in order to keep a stable gait, manipulators must grasp and manipulate objects. In all of these cases, the contact forces are critical to the system's behavior. Our MoE framework can be used to learn controllers that reason about the effects of contact forces on the system, leveraging the advantages of contact in some states and minimize its adverse effects in others.

References

[1] M. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.

[2] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2730–2736, IEEE, 2022.

[3] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[4] F. Ruggiero, A. Petit, D. Serra, A. C. Satici, J. Cacace, A. Donaire, F. Ficuciello, L. R. Buonocore, G. A. Fontanelli, V. Lippiello, *et al.*, "Nonprehensile manipulation of deformable objects: Achievements and perspectives from the robotic dynamic manipulation project," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 83–92, 2018.

[5] K. M. Lynch and T. D. Murphey, "Control of nonprehensile manipulation," in *Control problems in robotics*, pp. 39–57, Springer, 2003.

[6] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 64–92, 1999.

[7] M. Erdmann, "An exploration of nonprehensile two-palm manipulation," *The International Journal of Robotics Research*, vol. 17, no. 5, pp. 485–503, 1998.

[8] M. Yashima, Y. Shiina, and H. Yamaguchi, "Randomized manipulation planning for a multi-fingered hand by switching contact modes," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2, pp. 2689–2694, IEEE, 2003.

[9] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4066–4073, IEEE, 2017.

[10] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov, "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system," in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 35–42, IEEE, 2018.

[11] X. Zhang, M. Chang, P. Kumar, and S. Gupta, "Diffusion meets dagger: Supercharging eye-in-hand imitation learning," *arXiv preprint arXiv:2402.17768*, 2024.

[12] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[13] T. Härkönen, S. Wade, K. Law, and L. Roininen, "Mixtures of gaussian process experts with smc^2," *arXiv preprint arXiv:2208.12830*, 2022.

[14] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[15] Z. Chen, Y. Deng, Y. Wu, Q. Gu, and Y. Li, "Towards understanding mixture of experts in deep learning," *arXiv preprint arXiv:2208.02813*, 2022.

[16] M. M. Zhang and S. A. Williamson, "Embarrassingly parallel inference for gaussian processes," *Journal of Machine Learning Research*, 2019.

[17] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

[18] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[19] D. Liberzon, *Switching in systems and control*, vol. 190. Springer, 2003.

[20] R. Goebel, R. G. Sanfelice, and A. R. Teel, "Hybrid dynamical systems," *IEEE control systems magazine*, vol. 29, no. 2, pp. 28–93, 2009.

[21] B. Brogliato and B. Brogliato, *Nonsmooth mechanics*. Springer, 1999.

[22] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," in *In AISTATS*, Citeseer, 2011.

[23] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in julia," *arXiv preprint arXiv:1607.07892*, 2016.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[26] J. J. Moreau, "Unilateral contact and dry friction in finite freedom dynamics," *Nonsmooth mechanics and Applications*, pp. 1–82, 1988.

[27] C. Glocker and C. Studer, "Formulation and preparation for numerical evaluation of linear complementarity systems in dynamics," *Multibody System Dynamics*, vol. 13, pp. 447–463, 2005.

[28] V. Acary and B. Brogliato, *Numerical methods for nonsmooth dynamical systems: applications in mechanics and electronics*. Springer Science & Business Media, 2008.

[29] Quanser, *Linear Servo Base Unit with Inverted Pendulum*. Apr 2021.

[30] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2020.

[31] H. Khalil, *Nonlinear Control*. Always Learning, Pearson, 2015.

VII. Appendix

Consider the Lyapunov function candidate

$$V(\tilde{x}_c, \dot{\tilde{x}}_c, \tilde{\xi}) = \frac{1}{2} m_c r^2 + c\lambda \tilde{x}_c^2 + \frac{1}{2} \tilde{\xi}^\top \Gamma \tilde{\xi}.$$

This is a positive definite function over the space of $(\tilde{x}_c, \dot{\tilde{x}}_c, \tilde{\xi})$. We take the time derivative of the Lyapunov function candidate

and substitute from the closed-loop system dynamics (17, 18). We suppress its functional dependence for brevity.

$$\begin{aligned}\dot{V} &= m_c r\dot{r} + 2c\lambda\tilde{x}_c\dot{\tilde{x}}_c + \tilde{\xi}^\top\Gamma\dot{\tilde{\xi}} \\ &= r\left(-(b+c)r - Y\tilde{\xi}\right) + 2c\lambda\tilde{x}_c\dot{\tilde{x}}_c + \tilde{\xi}^\top Y^\top r \\ &= -e^\top Qe \leq 0,\end{aligned} \tag{19}$$

where Q is a symmetric, positive-definite matrix given as

$$Q = \begin{bmatrix}(b+c)\lambda^2 & b\lambda \\ b\lambda & b+c\end{bmatrix}.$$

Integrating both sides of equation (19) gives

$$V(t) - V(0) = -\int_0^t e^\top(\sigma)Qe(\sigma)\,\mathrm{d}\sigma < \infty.$$

We observe that $\dot{\tilde{x}}_c$ is bounded because $\dot{V} \leq 0$ implies that the terms r, $\tilde{x}_c$ and $\tilde{\xi}$ are bounded functions of time. This allows us to invoke Barbalat's lemma[30] to deduce that $\tilde{x}_c \to 0$ as $t \to \infty$. Furthermore, using equation (17), we can readily see that $\ddot{x}_c$ is bounded. Another application of Barbalat's lemma shows that the velocity error $\dot{\tilde{x}}_c \to 0$ provided that the reference acceleration $\ddot{x}_{c_r}(t)$ is bounded. Since $x_c \to x_{c_r}$, we also have that $u(x_c, \dot{x}_c) \to \hat{m}_c\ddot{x}_{c_r} + \hat{b}\dot{x}_{c_r} + \hat{k}x_{c_r}$.

Remark *For any $\eta > 0$, the set $\Omega_\eta = \{(e, \tilde{\xi}) : V(\tilde{x}_c, \dot{\tilde{x}}_c, \tilde{\xi}_c) \leq \eta\}$ is positively invariant. The positive limit set of $(e(t), \tilde{\xi}(t))$ is a subset of $E = \{(e, \tilde{\xi}) : e = 0\}$. Unfortunately, for a general nonautonomous system, the positive limit set is not necessarily a positively invariant set, precluding us to invoke LaSalle's theorem[31] to conclude that $\tilde{\xi} \to 0$.*

BIOGRAPHIES

AYKUT C. SATICI received the B.Sc. and M.Sc. degrees in mechatronics engineering from Sabanci University, Istanbul, Turkey, in 2008 and 2010, respectively, and the Masters degree in mathematics from the University of Texas, Dallas, TX, USA, in 2013. He is currently with the Electrical Engineering Department, University of Texas at Dallas. His current research interests include robotics, geometric mechanics, and cooperative control.