# Learning Stable Normalizing-Flow Control for Robotic Manipulation

Shahbaz Abdul Khader*[1,2], Hang Yin*[1], Pietro Falco[2] and Danica Kragic[1]

***Abstract*— Reinforcement Learning (RL) of robotic manipulation skills, despite its impressive successes, stands to benefit from incorporating domain knowledge from control theory. One of the most important properties that is of interest is control stability. Ideally, one would like to achieve stability guarantees while staying within the framework of state-of-the-art deep RL algorithms. Such a solution does not exist in general, especially one that scales to complex manipulation tasks. We contribute towards closing this gap by introducing *normalizing-flow* control structure, that can be deployed in any latest deep RL algorithms. While stable exploration is not guaranteed, our method is designed to ultimately produce deterministic controllers with provable stability. In addition to demonstrating our method on challenging contact-rich manipulation tasks, we also show that it is possible to achieve considerable exploration efficiency–reduced state space coverage and actuation efforts–without losing learning efficiency.**

## I. INTRODUCTION

Reinforcement Learning (RL) promises the possibility of robots to autonomously acquire complex manipulation skills. With the advent of deep learning, neural networks have been used as policies of a rich form that can even connect input, e.g. visual images, to output, e.g. motor torque or velocity, in an end-to-end manner [1], [2]. Although such general purpose application of RL is interesting, methods that benefit from prior domain knowledge, either as special structures for the policy or general properties from control theory, are gaining momentum. Employing structure on polices include trajectory-based shallow policy search such as [3], [4], [5] and also embedding variable impedance controller (VIC) structure into a deep neural policy [6], [7]. Such methods typically show a faster learning curve when compared to more general deep RL methods. What is more interesting is the trend to push the state-of-the-art by incorporating the property of control stability into RL as a means to guarantee safe and predictable behavior [8], [9], [10], [11], [12]. Unfortunately, with exception of one [12], no such methods have been applied in the context of robotic manipulation.

Stability is one of the first properties to be ensured whenever a closed-loop control law is synthesized. Despite this, stability guaranteed deep RL algorithms that actually scale to manipulators have proven to be elusive. Part of the reason is that stability analysis assumes knowledge of the dynamics model, something that is unavailable in an RL setting. Therefore, an RL algorithm would either need to learn a dynamics model first and then update policy [9], assume prior knowledge of nominal or partial models in order to reason about stability [10], [11], or assume prior stabilizing controllers [8], [11]. These methods seldom scale to manipulation control which often involves contact with the environment. Learning [13] or even offline modeling of contact dynamics is notoriously difficult. Our previous work [12] made progress by exploiting the passive interaction property between the manipulator and its environment that allowed us to skip learning of contact dynamics. However, this work employed specialized policy structure and Cross Entropy Method (CEM) based policy search that can only use one entire trajectory as a data point; whereas, state-of-the-art policy gradient methods can utilize all state-action samples as data points along with gradient information. Therefore, achieving stability in deep RL methods and its potential implications on learning efficiency remains an open question.

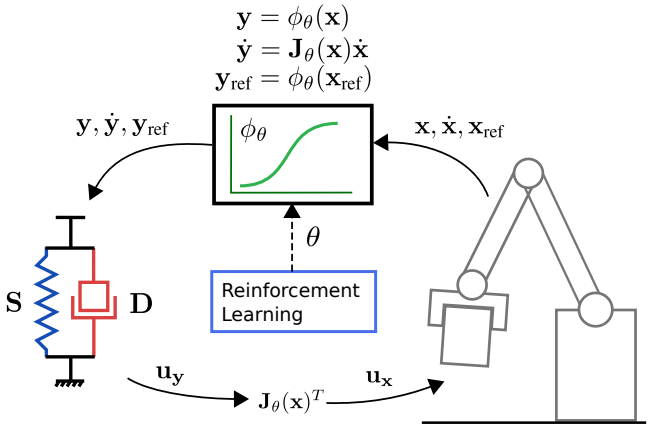


Fig. 1. **Learning Normalizing-flow Controllers** A 'normal' spring-damper system operating in a transformed space regulates the transformed image of the manipulation task dynamics. The transformation is through a bijection $\phi_\theta(.)$ (*normalizing-flow*), with Jacobian $\mathbf{J}_\theta$, that is parameterized as a neural network and learned through reinforcement learning. The optimum $\phi(.)$ that is just enough for the fixed spring-damper system to solve the desired task is sought. Such a construction imparts the stability properties of the spring-damper system to the entire system. $\mathbf{x}, \dot{\mathbf{x}}, \mathbf{x}_{\text{ref}}$ are the position, velocity and regulation point in the real system, and $\mathbf{y}, \dot{\mathbf{y}}, \mathbf{y}_{\text{ref}}$ its counterparts in the transformed space. $\mathbf{u_x}$ and $\mathbf{u_y}$ are control action forces.

We present an approach to learn stable *normalizing-flow* controllers for manipulation tasks. The focus is put on contact-rich tasks. Unlike the prior work [12], we do not target stable exploration but aim for stability guaranteeing deterministic controllers as a final result of RL. The proposed method uses neural network parameterized policies with additional structure and is suitable for any state-of-the-

*These authors contributed equally (listed in alphabetical order).

[1]Robotics, Perception, and Learning lab, KTH Royal Institute of Technology, Stockholm, Sweden. {shahak, hyin, dani}@kth.se.

[2]ABB Corporate Research, Vasteras, Sweden. pietro.falco@se.abb.com.

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and European Research Council (ERC) grant agreement No. [884807].

art policy search method. Our results show that not only is learning stable controllers feasible, without sacrificing learning efficiency, but also that it is possible to achieve significant reduction in state space coverage and actuation efforts–both desirable in real-world RL. We reason that this improvement in exploration efficiency is due to the fact that the stability property, while acting as a bias to the policy, is able to direct the exploration towards the goal. The results are supported with extensive simulation as well as a real-world robotic experiments on contact-rich manipulation tasks.

## II. Preliminaries

### A. Reinforcement Learning

Reinforcement learning addresses solving a Markov decision process problem that is described by an environment model (or dynamics) $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, a state space $\mathbf{s} \in \mathcal{S}$, an action space $\mathbf{a} \in \mathcal{A}$ and a reward function $r(\mathbf{s}_t, \mathbf{a}_t)$. The environment is generally assumed to be unknown. The solution to the problem is reached when the policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ of the agent, that acts upon the environment, is able to maximize the accumulated reward, usually along the horizon of a fixed length $T$:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{s}_0,\mathbf{a}_0,\ldots,\mathbf{s}_T}[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t)],$$

where $\{\mathbf{s}_0, \mathbf{a}_0, ..., \mathbf{s}_T\}$ is a sample trajectory from the distribution induced in the stochastic system.

Although on-policy RL methods typically require a stochastic policy, continuous control problems such as manipulator control usually expects a deterministic policy as the ultimate result. With this in mind, we reserve the term controller exclusively for a deterministic map of state to action, $\mathbf{a} = \pi(\mathbf{s})$, and the term policy for the stochastic counterpart, $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$. The quantity $\pi$, therefore, is interpreted based on context. We shall use the notation $\mathbf{u}$ also for action to maintain conformity to control literature.

### B. Robot Manipulator Dynamics

The state of a robot manipulator is defined by a generalized coordinate $\mathbf{x} \in \mathbb{R}^d$, and its time derivative $\dot{\mathbf{x}}$ ($\mathbf{s} = [\mathbf{x}^T\ \dot{\mathbf{x}}^T]^T$). The robot dynamics can be written as:

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{g}(\mathbf{x}) = \mathbf{u} + \mathbf{f}_{\text{ext}} \tag{1}$$

with $\mathbf{M}(\mathbf{x}) \in \mathbf{R}^{d\times d}$, $\mathbf{C}(\mathbf{x}, \ddot{\mathbf{x}}) \in \mathbf{R}^{d\times d}$ and $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^d$ denoting the inertia matrix, Coriolis term and the gravitational force. The manipulator is actuated by control efforts $\mathbf{u} \in \mathbb{R}^d$ and subject to external generalized force $\mathbf{f}_{\text{ext}} \in \mathbb{R}^d$.

In the context of RL, the coupled system of a manipulator interacting with its environment is considered the environment of the RL problem. Equation (1) represents all the involved quantities: the policy acts through $\mathbf{u}$, the manipulator part of the environment is represented by the LHS and the robot-environment interaction is accounted by $\mathbf{f}_{\text{ext}}$. In this paper, we further assume a passive environment for the robot, where the robot is the only actuated entity. This has implications for the stability property and will be explained in Sec. III-B.

### C. Lyapunov Stability

Lyapunov stability considers the temporal behaviour of nonlinear control systems such as robotic manipulators [14]. The system is stable if the state trajectories that start close enough are bounded around an equilibrium point, and is asymptotically stable if the trajectories converge to the equilibrium point. Such properties are the first necessary steps towards a guarantee that the system does not diverge rapidly causing potential harm. Lyapunov analysis proves stability by constructing a Lyapunov function $V(\mathbf{x}, \dot{\mathbf{x}})$, a scalar function of the state, with the property that it decreases as the system evolves. See [14] for a formal definition.

In the manipulator case, stability implies that any trajectory that is realized by the controller is bounded in state space and converges to an equilibrium point. The equilibrium point is usually designed to be a unique task relevant goal position, for example the final position in an episodic task.

### D. Invertible Neural Networks and Normalizing-Flows

In normalizing-flows [15], complex probability distribution models are constructed by learning a sequence of invertible and differentiable neural network transformations that maps simple probability distributions into the required complex ones. The overall transformation is a bijection that maps from one domain to another.

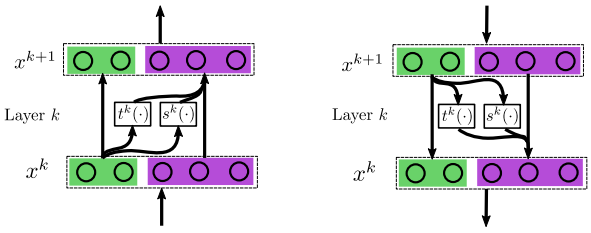


Fig. 2. **Normalizing-flow transformations** The layers can be stacked to build bijective neural network transformations.

In a specific design (see Fig. 2), a normalizing-flow network was constructed by stacking invertible affine coupling layers [16]:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \phi^k(\mathbf{x}^k) \\
\mathbf{x}^{k+1}_{1:d^k} &= \mathbf{x}^k_{1:d^k} \\
\mathbf{x}^{k+1}_{d^k+1:d} &= \mathbf{x}^k_{d^k+1:d} \odot \exp(\mathbf{s}^k(\mathbf{x}^k_{1:d^k})) + \mathbf{t}^k(\mathbf{x}^k_{1:d^k})
\end{aligned}$$

where $\mathbf{x}^k$ is the input to the layer, $D$ is the dimension of the input and $d^k < D$ is the index where the input dimensions are partitioned at layer $k$. The notation $\odot$ denotes elementwise multiplication and $\mathbf{s}^k(\cdot)$ and $\mathbf{t}^k(\cdot)$ are nonlinear transformations. The entire network is composed as $\phi = \phi^k \circ \phi^{k-1} \circ ... \circ \phi^1$ with the invertibility cascaded. $\mathbf{s}^k(\cdot)$ and $\mathbf{t}^k(\cdot)$ are parameterized as neural networks and may be made continuously differentiable as was done in [17].

## III. Stable Normalizing-Flow Policy

To realize the goal of RL with stability properties, we expect the policy to be: composed of a stable controller,

differentiable, parameterized as a neural network for the most parts and amenable to unconstrained policy search.

In this section, we first draw some parallels between density estimation and dynamical systems to motivate the notion of normalizing-flow controller. This is followed by the formal stability proof and also formalizing the RL problem.

### A. Normalizing-Flow for Second-order Dynamical Systems

The main idea is transforming a simple regulation control system with desirable properties, such as a spring-damper system with stability, to a more complex one without losing those properties. This is analogous to normalizing-flow models that use bijective mappings to construct rich probability densities from a normal distribution, see Fig. 3. While these models transform random variables back and forth, our model is designed to transform the state variables. In both cases, the bijective transformations are learned from data. Our method is inspired from [17], but unlike it, we use a second-order spring-damper system and learn the transformation using RL.

Given a position space $\mathbf{x} \in \mathcal{X}$, a transformed position space $\mathbf{y} \in \mathcal{Y}$ and a nonlinear bijection $\mathbf{y} = \phi(\mathbf{x})$, consider the spring-damper regulation control:

$$\mathbf{u_y} = -\mathbf{S}\mathbf{y} - \mathbf{D}\dot{\mathbf{y}} \implies \mathbf{u_y} = -\mathbf{S}\phi(\mathbf{x}) - \mathbf{D}\mathbf{J}(\mathbf{x})\dot{\mathbf{x}}$$

where $\mathbf{J}$ is the Jacobian of $\phi$. From the principle of virtual work, we expect control in the $\mathbf{x}$ space $\mathbf{u_x}$ to yield $\mathbf{u_x}^T\delta\mathbf{x} = \mathbf{u_y}^T\delta\mathbf{y}$. Using $\delta\mathbf{y} = \mathbf{J}(\mathbf{x})\delta\mathbf{x}$, we have:

$$\mathbf{u_x} = \mathbf{J}^T(\mathbf{x})\mathbf{u_y} = -\mathbf{J}^T(\mathbf{x})\mathbf{S}\phi(\mathbf{x}) - \mathbf{J}^T(\mathbf{x})\mathbf{D}\mathbf{J}(\mathbf{x})\dot{\mathbf{x}} \quad (2)$$

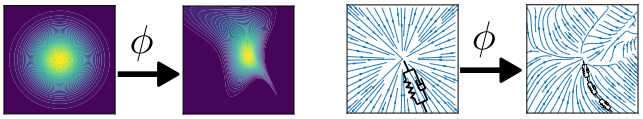


Fig. 3. **Left**: modeling complex probability distribution with normalizing-flow, e.g. by transforming from a simple normal distribution. **Right**: modeling complex attractor fields through a similar process of nonlinear transformation from 'normal' spring-damper system (e.g. $\mathbf{S} = \mathbf{D} = \mathbf{I}$).

We call the form in Eq. (2) a *normalizing-flow controller*. Note that the transformation is comparable to general coordinate transformation in task space.

### B. Stable Normalizing-Flow Controller

We present our formal theoretical result of the stable controller as following:

*Theorem*: Given a differentiable and bijective function $\phi : \mathbb{R}^d \to \mathbb{R}^d$ with its Jacobian $\mathbf{J} : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, positive definite matrices $\mathbf{S} \succ 0$ and $\mathbf{D} \succ 0$, the robot dynamics Eq. (1) driven by the controller

$$\mathbf{u} = -\mathbf{J}(\mathbf{x})^T\mathbf{S}[\phi(\mathbf{x}) - \phi(\mathbf{x}_{\text{ref}})] - \mathbf{J}^T(\mathbf{x})\mathbf{D}\mathbf{J}(\mathbf{x})\dot{\mathbf{x}} \quad (3)$$

is 1) globally asymptotically stable at the unique attractor $\mathbf{x}_{\text{ref}}$ if $\mathbf{f}_{\text{ext}} = \mathbf{0}$; and 2) a passive dynamical system if $\mathbf{f}_{\text{ext}} \neq \mathbf{0}$. We assume gravity is compensated externally.

The above theorem can be proved by performing a Lyapunov analysis with a function:

$$V(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2}[\phi(\mathbf{x}) - \phi(\mathbf{x}_{\text{ref}})]^T\mathbf{S}[\phi(\mathbf{x}) - \phi(\mathbf{x}_{\text{ref}})] + \frac{1}{2}\dot{\mathbf{x}}^T\mathbf{M}(\mathbf{x})\dot{\mathbf{x}}$$

Given $\mathbf{S} \succ 0$ and $\mathbf{M}(\mathbf{x}) \succ 0$, we have $V \geq 0$. Note that $\phi(\cdot)$ is bijective so the equality holds only when $\mathbf{x} = \mathbf{x}_{\text{ref}}$. By taking the time-derivative of the Lyapunov function,

$$\dot{V} = [\mathbf{J}(\mathbf{x})\dot{\mathbf{x}}]^T\mathbf{S}[\phi(\mathbf{x}) - \phi(\mathbf{x}_{\text{ref}})] + \dot{\mathbf{x}}^T\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \frac{1}{2}\dot{\mathbf{x}}^T\dot{\mathbf{M}}(\mathbf{x})\dot{\mathbf{x}}$$

Substitute $\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}}$ with Eq. (1), (3) and $\mathbf{f}_{\text{ext}} = \mathbf{0}$:

$$\begin{aligned}\dot{V} &= \dot{\mathbf{x}}^T[\mathbf{J}(\mathbf{x})^T\mathbf{S}(\phi(\mathbf{x}) - \phi(\mathbf{x}_{\text{ref}})) + \mathbf{u} \\ &\quad + \frac{1}{2}(\dot{\mathbf{M}}(\mathbf{x}) - 2\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}}))\dot{\mathbf{x}}] \\ &= -\dot{\mathbf{x}}^T\mathbf{J}^T(\mathbf{x})\mathbf{D}\mathbf{J}(\mathbf{x})\dot{\mathbf{x}} < 0\end{aligned}$$

where the skew-symmetric property of $\dot{\mathbf{M}} - 2\mathbf{C}$ and $\mathbf{D} \succ 0$ is used. From Eq. (3) the only invariant set for $\dot{V} = 0$ is $\mathbf{x} = \mathbf{x}_{\text{ref}}$ and $\dot{\mathbf{x}} = \mathbf{0}$. This is because Jacobian $\mathbf{J}(\mathbf{x})$ of bijective $\phi(\cdot)$ is full rank at any $\mathbf{x}$. According to LaSalle's invariant set theorem [18], $\mathbf{x}$ tends to $\mathbf{x}_{\text{ref}}$ when $\dot{V} = 0$ and the system is globally asymptotically stable at $\mathbf{x}_{\text{ref}}$.

By inserting $\mathbf{f}_{\text{ext}}$ to the equation above and observing that

$$\dot{V} = \dot{\mathbf{x}}^T\mathbf{f}_{\text{ext}} - \dot{\mathbf{x}}^T\mathbf{J}^T(\mathbf{x})\mathbf{D}\mathbf{J}(\mathbf{x})\dot{\mathbf{x}} < \dot{\mathbf{x}}^T\mathbf{f}_{\text{ext}},$$

we conclude that the controlled dynamical system is passive under $\mathbf{f}_{\text{ext}}$. The significance of passivity is that when two stable and passive systems interact physically, the coupled system is also stable [19], [20]. In other words, by ensuring the robot to be passive in isolation, we can conclude that all physical interaction with unknown but passive environments are automatically stable. This property can be exploited to avoid modeling or learning unknown interaction dynamics.

For the Cartesian space of a redundant manipulator, where the position coordinate $\mathbf{x}$ is not a generalized coordinate, we propose to implement the Jacobian transpose control presented in Sections 8.6.2 and 9.2.2 of [21]. As explained in Section 8.6.2 of [21], when a slightly reformulated Lyapunov analysis that involves joint space kinetic energy term and joint space viscous friction is considered, it guarantees energy decrease for any joint space velocity and thus ensures all null-space motions to eventually cease and reach an equilibrium position.

### C. RL with Normalizing-Flow Controller

The controller in Eq. (3) is an instance of the deterministic controller $\mathbf{a} = \pi(\mathbf{s})$ and can be easily converted into its stochastic counterpart $\pi(\mathbf{a}|\mathbf{s})$ by incorporating an additive Gaussian noise. Furthermore, by implementing $\phi(\mathbf{x})$ as a normalizing-flow neural network according to Sec. II-D, all the variable terms in Eq. (3) including $\mathbf{J}(\mathbf{x})$ are differentiable. Thus, it is only required to drop our policy into any policy gradient model-free RL algorithm, such as Proximal Policy Optimization (PPO) [22], that is implemented in an automatic differentiation framework.

Recall that only the controller has stability guarantee and not the noise-incorporated policy. We expect the RL

algorithm to eventually refine the noise covariance to low values such that the final controller is good enough for the task. Regardless of obtaining a stable controller, it is also important to investigate the effect of the proposed policy structure on RL. This is because the stability property acts as an attractor toward the goal position ($\mathbf{x}_{\text{ref}}$) and thus can be seen as a bias to the policy. Whereas, in standard deep RL no such bias exists for the policy.

## IV. Experimental Results

The proposed method is validated on a number of episodic tasks with fixed goal positions. While the low-dimensional block-insertion task (Fig. 4a) provides a ground for better visualization of the basic principles involved, the simulated peg-in-hole (Fig. 4d) and the real-world gear assembly (Fig. 4e) tasks help validate complex manipulation tasks that we care about. Furthermore, they are also contact-rich applications that provide the additional challenge of learning the force interaction behaviour–not just the motion profile–that suits well the second-order setting of force based policies.

Important hyperparameters and some experiment details are described below. The number of flow elements is indicated by $n_{\text{flow}}$, where a flow element is defined as two successive normalizing-flow layers (Sec. II-D) with swapped partitioning of input dimensions. This is to avoid one partition always passing the identity mapping, as suggested in [16]. The number of units in the hidden layer of $\mathbf{s}^k(.)$ and $\mathbf{t}^k(.)$, both of which are identically defined as fully-connected networks with only one hidden layer, is denoted as $n_h$. The stiffness and damping matrices, $\mathbf{S}$ and $\mathbf{D}$, are desired to be kept as identity matrices as per our interpretation of 'normal' spring-damper system. As mentioned earlier, the normalizing-flow controller is converted to a stochastic policy by incorporating an additive Gaussian noise, whose value is initialized with a spherical standard deviation defined by the scalar $\sigma_{\text{init}}$. Note that $\sigma_{\text{init}}$ is only relevant where the standard deviation of the policy is directly implemented as trainable parameters and not as a neural network. We use the deep RL framework *garage* [23] along with its default implementation of PPO. We used the reward function suggested in [24] for all our experiments.

### A. 2D Block-Insertion

The 2D block-insertion task was first introduced in [12] as a simplified representation of contact-rich manipulation. A block of mass, that can only slide on a surface without being able to rotate, is controlled by a pair of orthogonal forces acting on it with the goal of inserting it into a slot structure. Naturally, the 2D position $\mathbf{x}$ and velocity $\dot{\mathbf{x}}$ together forms the state variable and the 2D force $\mathbf{u}$ is the action variable. The final goal position, when inserted into the slot, is the regulation reference $\mathbf{x}_{\text{ref}}$. The initial position $\mathbf{x}_0$ is sampled from a wide normal distribution with the elements of its diagonal standard deviations as 50 mm and 100 mm, corresponding to the horizontal and vertical axes, respectively. We used 15 rollouts, each with an episode length of 2 seconds, in one iteration of RL. The

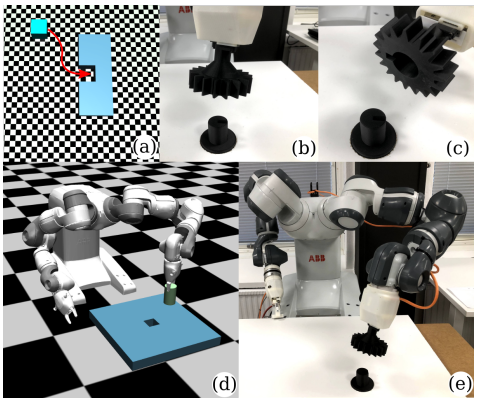


Fig. 4. **Experimental setup (a)** 2D Block-insertion: the block has dimensions of $50 \times 50 \times 50$ mm, weight of 1 Kg and insertion clearance of 2 mm. An example path is indicated from a representative initial position. **(b, c & e)** Real-world gear assembly: a gear with a cylindrical hole of diameter 30 mm and depth 20 mm is to be assembled onto a cylindrical shaft of 28 mm diameter. **(d)** Simulated peg-in-hole: a cylindrical peg of 23 mm diameter is to be inserted into a square hole of width 25 mm and depth 20 mm. An insertion depth of 25 mm for block-insertion and 16 mm for the robot assembly tasks are considered successful.

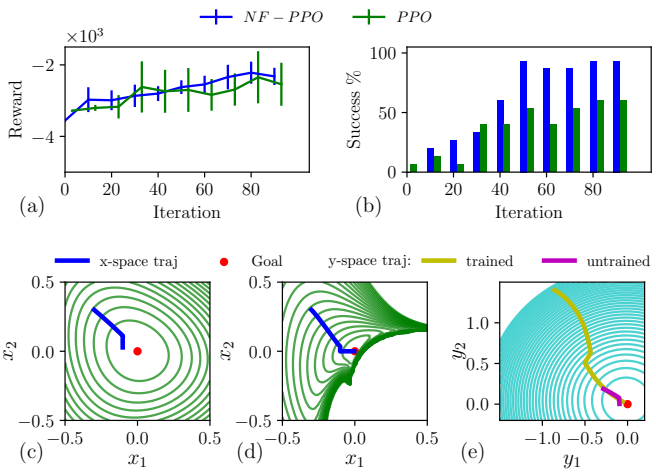


Fig. 5. **Learning Normalizing-flow policies for 2D Block-insertion** **(a)** RL reward **(b)** Success rate of insertions **(c)** A deterministic rollout, overlaid on the energy function $V_{|\dot{\mathbf{x}}=0}$ contour plot, in $\mathbf{x}$ space for randomly initialized $\mathbf{s}^k(.)$ and $\mathbf{t}^k(.)$ (iteration 0). **(d)** The same as **c**, but after RL (after iteration 99) **(e)** Plots corresponding to **c** and **d**, but in $\mathbf{y}$ space.

remaining hyperparameters were set as $n_{\text{flow}} = 2$, $n_h = 8$, $\sigma_{init} = 2.0$ N, and $\mathbf{S} = \mathbf{D} = \mathbf{I}$.

In our 100 iterations long experiment, we compare the RL performance of the proposed policy parameterization and also a regular neural network policy. The former will be hereafter called normalizing-flow PPO (NF-PPO) and the latter just PPO. Note that it is only the policy parameterization that is different while the RL algorithms in both cases are PPO. In Fig. 5a, it can be seen that NF-PPO does have an advantage in learning efficiency as it achieves near 100% insertion success rate at about 50 iterations. Figures 5c-d show a single trajectory rollout of the deterministic controller before and after RL. The trajectory may be compared to the expected one in Fig. 4a. The trajectory is overlaid on the contour plot of the proposed Lyapunov energy function

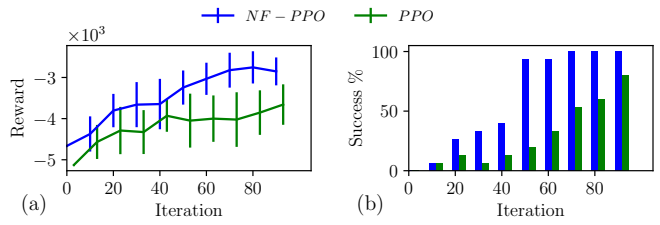


Fig. 6. **Learning Normalizing-flow policies for peg-in-hole (a)** RL reward **(b)** Success rate of insertions. Corresponds to $\sigma_{init} = 1.0$ in Table I.

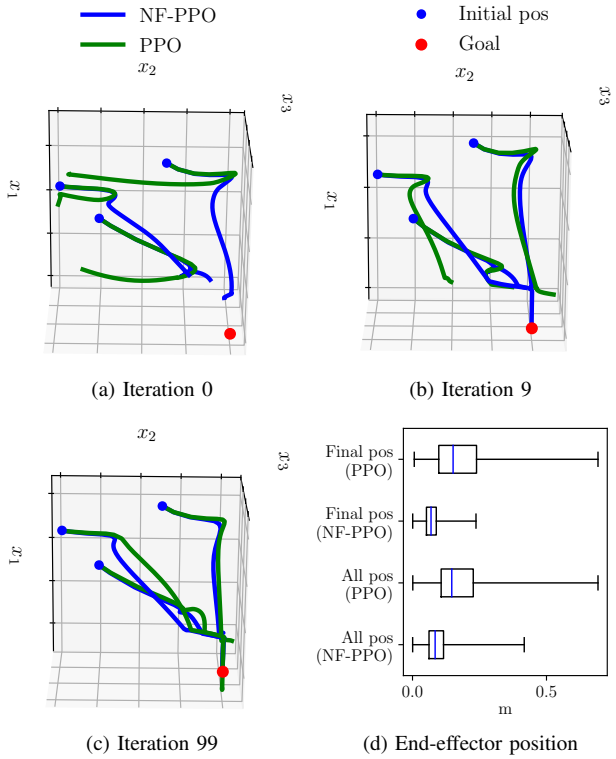


Fig. 7. **Effect of stable policy on learning peg-in-hole (a-c)** Deterministic rollouts of three randomly chosen initial positions at different stages of RL show consistent convergent behavior for NF-PPO. Note that these rollouts are only for evaluation and otherwise the policy is stochastic. **(d)** Distributions of end-effector distance $||\mathbf{x} - \mathbf{x_{ref}}||$ during the first 10 iterations of RL. The whiskers include all samples.

($V_{|\dot{\mathbf{x}}=0}$). When $\dot{\mathbf{x}} = 0$, the policy is the negative gradient of the energy function, which can therefore be used to partially visualize the policy. It is interesting to note that even with randomly initialized $\mathbf{s}^k(.)$ and $\mathbf{t}^k(.)$, the policy is well-behaved and acts as an attractor field toward the goal. The situation is similar in Fig. 5d except that the policy (and $V$) has been reshaped by RL into a richer one that not only behaves as an attractor but also accomplishes the task. Figure 5e tracks the trajectories before and after RL in the transformed $\mathbf{y}$ space. In this space, the trajectories may be severely deformed by the learned $\phi(.)$, but the energy function is a fixed independent quadratic function.

### *B. Simulated Peg-In-Hole*

In this experiment, we scale up to a 7-DOF manipulator but only consider the translation part of the operational space for policy implementation and RL. The rotation part is implemented as a fixed regulation control (stable impedance control) that achieves the desired orientation for insertion. Our method can be extended to include rotation also but is left out for the sake of simplicity. The task is the peg-in-hole problem that has long been considered as a representative problem for contact-rich manipulation [25], [26], [27]. The translation space position $\mathbf{x}$ and velocity $\dot{\mathbf{x}}$ form the state variable and the corresponding force $\mathbf{u}$ is the action variable. The initial position is sampled from a wide normal distribution in the joint space and then transformed into the operational space. The mean of the normal distribution roughly matches with the position in Fig. 4d and the spherical standard deviation is chosen to be 0.2 radians. The goal position, which is the inserted position, is set as the reference $\mathbf{x}_{\text{ref}}$. As in the previous case, here also we used 15 rollouts per iteration with each rollout having an episode time of 2 seconds. The remaining hyperparameters were set as $n_{\text{flow}} = 2$, $n_h = 16$, $\sigma_{init} = 1.0$ N, and $\mathbf{S} = \mathbf{D} = \mathbf{I}$.

TABLE I

IMPACT OF INITIALIZING POLICY VARIANCE

| | $\sigma_{\text{init}}$ | $Itr_{90}$ | $\|\|\mathbf{x} - \mathbf{x_{ref}}\|\|$ | $\|\|\tau\|\|$ |
|---|---|---|---|---|
| Units | N | | m | Nm |
| NF-PPO | **1.0** | **63** | $\mathbf{0.1 \pm 0.05}$ | $\mathbf{0.9 \pm 1.1}$ |
| | 1.5 | 67 | $0.1 \pm 0.05$ | $1.4 \pm 1.1$ |
| | 2.0 | 60 | $0.12 \pm 0.1$ | $2 \pm 5.1$ |
| | 3.0 | 56 | $0.14 \pm 0.1$ | $2.7 \pm 2.1$ |
| PPO | 1.0 | - | $0.17 \pm 0.1$ | $0.9 \pm 1.1$ |
| | 1.5 | 85 | $0.18 \pm 0.1$ | $1.3 \pm 1.2$ |
| | 2.0 | 73 | $0.18 \pm 0.12$ | $1.8 \pm 3.4$ |
| | **3.0** | **57** | $\mathbf{0.2 \pm 0.13}$ | $\mathbf{2.6 \pm 1.7}$ |

From the results of a 100 iterations long RL experiment in Fig. 6a-b, it can be seen that there is a clear advantage for our method in terms of learning efficiency, although PPO eventually catches towards the end of 100 iterations. To examine the effect of stability, we track deterministic rollouts from three randomly chosen initial positions at various stages of RL in Fig. 7a-c. At iteration 0, with randomly initialized policy, PPO causes unpredictable motion away from the goal while NF-PPO ensures convergence towards the goal. The situation has improved for PPO in iteration 9 but still not on par with NF-PPO. Finally, at iteration 99, PPO appears to have attained similar behavior as NF-PPO, although PPO only managed 2/3 insertions compared to 3/3 by NF-PPO. Apart from predictable motion behaviors, another advantage we expect from the stability property is efficient exploration that is directed more towards the goal. Such a trend is visible in Fig. 7d that presents the distributions of the end-effector distance relative to the goal during the first 10 RL iterations.

So far, we have not addressed the interplay between exploration and stability. In Table I, we present results of extensive simulation study that reveals the impact of initial policy variance $\sigma_{\text{init}}$. It turned out that $\sigma_{\text{init}}$, a hyperparameter, drastically affects the overall exploration behaviour and hence RL performance. Note that the initial policy variance is a trainable parameter and the RL algorithm is expected to refine this to smaller values progressively. As $\sigma_{\text{init}}$ is increased, both NF-PPO and PPO show improvements in RL performance, indicated by the number of iterations required

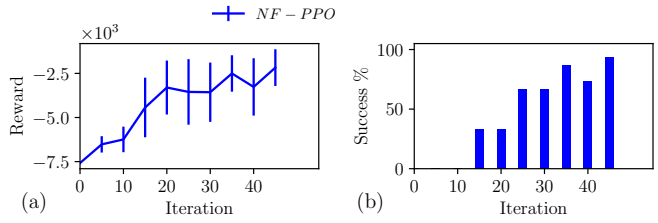


Fig. 8. **Learning Normalizing-flow policy for gear assembly (a)** RL reward. **(b)** Success rate of insertions.

for consistent $90\%$ success ($Itr_{90}$). This gain is, of course, at the expense of generally undesirable conditions such as wider state distribution and also higher joint torques. Although exploration is necessary for RL, the real-world manipulator case imposes constraints due to safety concerns. Joint torques can be considered as a proxy for acceleration/deceleration and also contact forces, both of which should be kept to a minimum. Interestingly, the general drop in state space coverage of NF-PPO is not accompanied by a reduced joint torque. This indicates that exploration is as energetic as PPO but more efficiently directed. A more specific and significant result is that the learning performance of our method with $\sigma_{init} = 1.0$ is closely matched by PPO only when $\sigma_{init} = 3.0$. In other words, our method is exploration efficient and as a result reduces state space coverage by half and joint torques by a third.

### C. Real-World Gear Assembly

In this experiment, we successfully attempted a real robot experiment on the gear assembly task. This task is in essence the inversion of the peg-in-hole case, where the peg and hole are reversed. All aspects of the system are identical to the peg-in-hole case except that initial position is fixed, episode length is set to five seconds, and the following changes in hyperparameters are made: $n_{\text{flow}} = 1$, $n_h = 16$, $\sigma_{init} = 5.0$ N, $\mathbf{S} = 4.0\mathbf{I}$ and $\mathbf{D} = \mathbf{I}$. $\sigma_{init}$ and $\mathbf{S}$ are increased significantly to overcome uncompensated joint friction. The non identity value for $\mathbf{S}$ is against our interpretation of 'normal' spring-damper system but was chosen for expediting real-world RL, which would otherwise be time consuming.

Figure 8 shows that the task was learned with $90\%$ success achieved at around 50 iterations. This is not directly comparable with the peg-in-hole simulation results in Fig. 6 since the initial position is randomly chosen there. Regardless, the results are satisfactory and not entirely different from the simulation results. As in the case of simulated peg-in-hole, stable behaviour was apparent right from the beginning of the RL process. We did not attempt PPO on the real robot as it is not obvious how to safeguard against the generally unpredictable behaviour at the beginning of RL.

## V. Discussions and Conclusions

In this paper, we sought out to investigate whether control stability can be incorporated into deep RL, and if so, what benefits would it entail? The approach was limited to formulating provably stable deterministic policies without ensuring stable exploration. Experimental evaluation focused on contact-rich tasks so that the force interaction aspect of manipulation is also included. Our results reveal that in addition to yielding a stable controller, the process of RL itself benefits from the approach.

Two concrete benefits can be identified for our method. First, the deterministic policy, which is usually the ultimate goal of RL, is provably stable for any possible instantiation of the policy, including randomly initialized or partially trained ones. This also includes the scenario where a fully trained policy visits a previously unseen region in the state space. In all such cases, trajectories will converge towards the goal providing a great deal of predictability and safety. Recall that stability is guaranteed even during contact with an unknown but passive environment. Second, the stability property has the potential to achieve efficient exploration. During random exploration, where stability is not technically guaranteed, the trajectories can still be directed towards the goal resulting in reduced state space coverage without affecting learning performance.

The most interesting outcome in our study is the concept of efficient exploration. Our results confirm that larger the exploration, the better it is in general for RL performance. However, this benefit comes with the generally undesirable cost of large coverage in state space and also high joint torques. Our results strongly indicate that by inducing stable behavior, state-action distribution can be drastically reduced without compromising the learning performance. This may be rephrased as: efficiently directing the exploration towards the general direction of the goal position can achieve considerable exploration efficiency. But, this advantage is lost when the exploration level is high enough to undermine the convergent behavior of stability, a possible explanation for the case $\sigma_{init} = 3.0$ in Table I.

The proposed method is not without limitations. The policy structure in Eq. (2) is likely to be constrained in its modeling capacity compared to a completely free neural network. Unfortunately, no formal notion of model capacity exists and quantitative comparison cannot be readily made. Another noteworthy point is the higher training and inference time of our method. Recall that the policy contains additional structures including the Jacobian of the bijection, which is expensive to compute during inference time. This forced us to downgrade $n_{\text{flow}}$ to 1 from a value of 2 to meet the realtime requirements in the real-world experiment. This can be easily addressed by more efficient implementation and faster computing hardware. The presence of Jacobian in the policy also means that it has to be further differentiated for computing the policy gradient during training. Based on our experience, a fourfold increase in overall computation time can be expected when compared to standard deep policies.

The results in this work may be considered as one step closer towards provable control stability in the context of deep RL of robotic manipulation. We showed that learning stable deterministic policies, or controllers, is both feasible and useful through a number of simulated and real-world RL of manipulation skills. As a next step, we intend to pursue deep RL methods that feature stable exploration as well.

## REFERENCES


[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[3] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[4] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.

[5] H. Yin, A. Paiva, and A. Billard, "Learning cost function and trajectory for robotic writing motion," in *Proceedings of IEEE International Conference on Humanoid Robots (Humanoids)*, Madrid, Spain, 2014.

[6] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1010–1017.

[7] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, 2020.

[8] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. null, p. 803–832, Mar. 2003.

[9] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, 2017, pp. 908–918.

[10] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, 2019, pp. 3387–3395.

[11] R. Cheng, A. Verma, G. Orosz, S. Chaudhuri, Y. Yue, and J. Burdick, "Control regularization for reduced variance reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning, ICML*, 2019, pp. 1141–1150.

[12] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Stability-guaranteed reinforcement learning for contact-rich manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 1–8, 2020.

[13] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Data-efficient model learning and prediction for contact-rich manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4321–4328, 2020.

[14] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991.

[15] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[16] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[17] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Proc. Conference on Learning for Dynamics and Control (L4DC)*, 2020.

[18] J. LaSalle and S. Lefschetz, *Stability by Liapunov's Direct Method: With Applications*, ser. Mathematics in science and engineering : a series of monographs and textbooks. Academic Press, 1961.

[19] N. Hogan and S. P. Buerger, "Impedance and interaction control," in *Robotics and automation handbook*. CRC press, 2018, pp. 375–398.

[20] J. E. Colgate and N. Hogan, "Robust control of dynamically interacting systems," *International journal of Control*, vol. 48, no. 1, pp. 65–88, 1988.

[21] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[23] T. garage contributors, "Garage: A toolkit for reproducible reinforcement learning research," https://github.com/rlworkgroup/garage, 2019.

[24] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 156–163.

[25] M. Nuttin and H. Van Brussel, "Learning the peg-into-hole assembly operation with a connectionist reinforcement technique," *Computers in Industry*, vol. 33, no. 1, pp. 101–109, 1997.

[26] S.-k. Yun, "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?" in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1647–1652.

[27] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8943–8950.