ImitationFlow: Learning Deep Stable Stochastic Dynamic Systems by Normalizing Flows

Julen Urain[1], Michele Ginesi[2], Davide Tateo[1], and Jan Peters[1,3]

Abstract— We introduce ImitationFlow, a novel Deep generative model that allows learning complex globally stable, stochastic, nonlinear dynamics. Our approach extends the Normalizing Flows framework to learn stable Stochastic Differential Equations. We prove the Lyapunov stability for a class of Stochastic Differential Equations and we propose a learning algorithm to learn them from a set of demonstrated trajectories. Our model extends the set of stable dynamical systems that can be represented by state-of-the-art approaches, eliminates the Gaussian assumption on the demonstrations, and outperforms the previous algorithms in terms of representation accuracy. We show the effectiveness of our method with both standard datasets and a real robot experiment.

I. INTRODUCTION

To have fully autonomous robots in real-world scenarios, robots need to be able to perform a wide variety of complex tasks and skills. However, programming a robot to perform complex motion is often a time-consuming task, which requires both expertise and domain knowledge. Imitation Learning tackles this problem and provides robotics non-expert users the capability to teach robots complex trajectories providing a few demonstrations [1], [2].

Given a set of demonstrations, the objective of Imitation learning is to find a generative model that produces trajectories similar to the demonstrated ones. A correct selection of the model will lead to a successful and safe imitation. These generative models are called Movement Primitives. Movement primitives can be divided into three different categories: time-dependant, state-dependant, and time-state dependant movement primitives.

In this work, we focus on state dependant motion primitives as they are robust to both spatial and temporal perturbation. This class of motion primitives is complementary to the time-dependent motion primitives approaches, such as the Dynamic Movement Primitives (DMP) [3] and the Probabilistic Movement Primitives (ProMP) [4] approaches, that are particularly well suited when the movement presents a clear temporal (or phase) dependency, while robustness to perturbation is not a major concern.

One of the main issues of state-dependent movement primitives is to learn stable dynamics: While several models

[1]Intelligent Autonomous Systems, TU Darmstadt
[2]Department of Computer Science, University of Verona
[3]MPI for Intelligent Systems, Tuebingen
{urain,tateo,peters}@ias.tu-darmstadt.de
michele.ginesi@univr.it

Fig. 1. Above, robot is taught to perform a drums playing task. Below, *ImitationFlows* receives the recorded trajectories and learns to morph the latent dynamics in the left to the dynamics in the right. The generated dynamics in the right are given back to the robot to perform the task.

exist [5], [6], [7]; most of them are not good ensuring stable dynamics out of the region of the demonstrated trajectories. This issue limits the applications of these methods to real-world scenarios, as it can be difficult to ensure the correct execution of the learned motion. This issue has been faced by the Stable Estimator of Dynamical Systems (SEDS) algorithm [8]. In this model, the global asymptotically stability is ensured by a quadratic Lyapunov function. Due to the proposed Lyapunov function, the learned dynamics are restricted to continuously decreasing distance towards the attractor. In order to overcome the limitation of SEDS, different approaches were proposed [9], [10], [11], [12], [13], [14], [15], [16]. However, except for [17], all these models consider deterministic models. Also, all these works consider only point-to-point dynamics.

Contributions In this paper, we present *ImitationFlow*, a novel approach for learning stable stochastic nonlinear dynamical systems. Our methodology merges Deep generative models like Normalizing Flows [18], [19], [20] with stable dynamical systems modeling.

Our approach not only is capable of representing a wider class of dynamical systems w.r.t. previous works in the field, but it can also represent arbitrary complex densities, removing the Gaussian noise assumption of the demonstrated trajectories. The proposed model can describe both strike-

based and periodic movements in a single framework, without changing the core learning algorithm.

We also formally prove that *ImitationFlow* can learn globally asymptotically stable stochastic dynamics. This property ensures global convergence to the goal for point-to-point motions, ensuring that the learned movement can be applied from any starting point of the workspace. We have released a Pytorch Implementation at https://github.com/TheCamusean/iflow

II. BACKGROUND & NOTATION

A. Problem Statement

Let $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{n-1}, \tau_n\}$ be a set of n expert demonstrated trajectories, where each trajectory $\tau_i = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{T_i}\}$ of length l_i is a sequence of observations $\boldsymbol{y}_i \in \mathbb{R}^d$. For clarity of presentation, we assume that each element of the trajectory is generated with a fixed sampling time ΔT, but our derivations can be extended to the variable sampling time scenario.

Let $p(y_0)$ be the distribution of the trajectory starting point. We assume that each trajectory element is generated by the following Stochastic Differential Equation (SDE):

$$d\boldsymbol{z}(t) = f(\boldsymbol{z}(t))dt + \boldsymbol{g}(\boldsymbol{z}(t))d\boldsymbol{B}(t), \tag{1}$$

where $\boldsymbol{z} \in \mathbb{R}^d$ is the state, $f : \mathbb{R}^d \to \mathbb{R}^d$ is a continuous function, $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^{d \times d}$ is a continuous matrix function, and $\boldsymbol{B} : \mathbb{R} \to \mathbb{R}^d$ is a $d-$dimensional *Brownian motion* (also called *Wiener process*).

We remark that (1) is written in autonomous form. The general case, in which f and $\boldsymbol{g}$ are functions of both the state and time can be easily recovered by defining $\tilde{\boldsymbol{z}} = [\boldsymbol{z}^\intercal, t]^\intercal$, $\tilde{f}(\tilde{\boldsymbol{z}}) = [f(\boldsymbol{z}(t))^\intercal, 1]^\intercal$.

Our problem is to maximize the likelihood of the unknown model parameters $\boldsymbol{\psi} = \{\boldsymbol{\theta}, \boldsymbol{\phi}\}$ w.r.t. the set of observed trajectories $\mathcal{T}$. We frame our learning problem as the following optimization problem

$$\boldsymbol{\psi}^* = \arg\max_{\boldsymbol{\psi}} \mathcal{L}_{\boldsymbol{\psi}}(\mathcal{T})$$
$$s.t. \lim_{t \to \infty} \mathbb{E}[\boldsymbol{y}_t] \in \Omega, \forall y_0 \in \mathbb{R}^d \tag{2}$$

where $\Omega \subset \mathbb{R}^d$ and $\mathcal{L}_{\boldsymbol{\psi}}$ is defined as

$$\mathcal{L}_{\boldsymbol{\psi}}(\mathcal{T}) = \prod_{i=1}^{n} p(\tau_i; \boldsymbol{\psi}) = \prod_{i=1}^{n} p(y_0) \prod_{j=0}^{l_i} p(y_{j+1}|y_j).$$

B. Stability for Stochastic Dynamical systems

Lyapunov stability has various definitions in the case of stochastic dynamical systems. One option is to study the stability in probability, in which the stability is studied replacing $\dot{V}$ by its expected value [21].

By Itô's formula, the time derivative of the Lyapunov function, $V(\boldsymbol{y}, t)$, is expressed by the following differential equation

$$dV(\boldsymbol{y}, t) = LV(\boldsymbol{y}, t)dt + V_{\boldsymbol{y}}(\boldsymbol{y}, t)\boldsymbol{g}(\boldsymbol{y}, t)d\boldsymbol{B}(t)$$

$$LV(\boldsymbol{y}, t) = V_t + V_{\boldsymbol{y}}f(\boldsymbol{y}, t) + \frac{1}{2}\text{Tr}(\boldsymbol{g}^\intercal(\boldsymbol{y}, t))V_{\boldsymbol{y}\boldsymbol{y}}\boldsymbol{g}(\boldsymbol{y}, t), \tag{3}$$

Fig. 2. Imitation Flow architecture as a graphical model

where $LV(\boldsymbol{y}, t)$, is known as the diffusion equation. The expected value of $\dot{V}$ is

$$\mathbb{E}\left[\dot{V}(\boldsymbol{y}, t)\right] = LV(\boldsymbol{y}, t). \tag{4}$$

Assume there exist a $V(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}, (\boldsymbol{y}, t) \mapsto V(\boldsymbol{y}, t)$, and strictly increasing functions μ_1, μ_2, μ_3 such that

$$\mu_1(|\boldsymbol{y}|) \leq V(\boldsymbol{y}, t) \leq \mu_2(|\boldsymbol{y}|), \tag{5a}$$
$$LV(\boldsymbol{y}, t) \leq -\mu_3(|\boldsymbol{y}|) \forall \boldsymbol{y} \in \mathbb{R}^d. \tag{5b}$$

Then, the trivial solution for our SDE is stochastically asymptotically stable [21].

C. Normalizing Flows

Normalizing Flows provide a method for expressive density approximation. [18]. Requiring only a base distribution, usually uniform or normal distribution, and a set of bijective transformations, Normalizing Flows allows explicit density estimation and, in most cases, to sample from these complex distributions [20], [19].

Given a latent variable $\boldsymbol{z} \in \mathbb{R}^d$, sampled from a certain distribution $z \sim p_z(\boldsymbol{z})$ and a diffeomorphic transformation $h : \mathbb{R}^d \to \mathbb{R}^d$, such that $\boldsymbol{y} = h(\boldsymbol{z})$, we can compute the distribution of $\boldsymbol{y}$, $p_y(\boldsymbol{y})$, in terms of $\boldsymbol{z}$, by the change of variable rule.

$$p_y(\boldsymbol{y}) = p_z(\boldsymbol{z})\left|\det\frac{\partial\boldsymbol{z}}{\partial\boldsymbol{y}}\right| = p_z(h^{-1}(\boldsymbol{y}))\left|\det\frac{\partial h^{-1}(\boldsymbol{y})}{\partial\boldsymbol{y}}\right|. \tag{6}$$

III. PROPOSED METHOD

ImitationFlow model extends the concept of Structured Inference Networks for Nonlinear State Space Models [22] to Normalizing Flows. Considering Normalizing Flows as emission function allows exact inference at the cost of imposing a deterministic emission.

The proposed model's architecture is presented in Fig. 2, and the dynamics are modelled using (7). Our model is composed of two main components. In the latent space $\mathcal{Z}$, the transition model follows some stable stochastic dynamics parameterized by $\boldsymbol{\phi}$. Then, we use, as emission function, a bijective, continuous and differentiable transformation $h_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^d$ transforming the data from the latent space $\mathcal{Z}$ to the observation space $\mathcal{Y}$.

$$d\boldsymbol{z}(t) = f_{\boldsymbol{\phi}}(\boldsymbol{z})dt + \boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{z})d\boldsymbol{B}(t)$$
$$\boldsymbol{y} = h_{\boldsymbol{\theta}}(\boldsymbol{z}), \tag{7}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are the learnable parameters of the model. Given that the Jacobian of $h_{\boldsymbol{\theta}}$, $J_{\boldsymbol{\theta}} = \frac{d\boldsymbol{y}}{d\boldsymbol{z}}$, is easy to compute, we can reframe (7) to compute the stochastic dynamic model for $\boldsymbol{y}(t)$

$$d\boldsymbol{y}(t) = J_{\boldsymbol{\theta}}(\boldsymbol{y})f_{\boldsymbol{\phi}}(h_{\boldsymbol{\theta}}^{-1}(\boldsymbol{y}))dt + J_{\boldsymbol{\theta}}(\boldsymbol{y})g_{\boldsymbol{\phi}}(h_{\boldsymbol{\theta}}^{-1}(\boldsymbol{y}))d\boldsymbol{B}(t)$$
(8)

A. Learning Algorithm

In this section, we describe how we solve the optimization problem described in (2). In order to estimate correctly the likelihood of the trajectories, we should estimate the probability density of initial states $p(\boldsymbol{y}_0)$. However, if a few trajectories are available, this estimation can be problematic. We leverage on the fact that the system represents stable dynamics: this means that we will have a stationary distribution in the limit

$$\lim_{t \to \infty} p(\boldsymbol{y}_t) = p(\boldsymbol{y}_\infty).$$

We exploit this property by assuming that the distribution of the last point of the trajectory is the stationary one. Differently from most of others probabilistic estimation algorithms, where the structural density is considered with forward conditioning, $p(\boldsymbol{y}_{j+1}|\boldsymbol{y}_j)$, in our approach we consider a backward conditioning with $p(\boldsymbol{y}_j|\boldsymbol{y}_{j+1})$. It is straightforward to prove that this view is equivalent under Bayes's rule.

Given the model proposed in (7), we can rewrite the probability distributions $p(\boldsymbol{y})$ in terms of $p(\boldsymbol{z})$. By applying the change of variable rule we obtain

$$p(\boldsymbol{y}_n) = p(\boldsymbol{z}_n)\left|\det\frac{\partial\boldsymbol{z}_n}{\partial\boldsymbol{y}_n}\right| = p(f^{-1}(\boldsymbol{y}_n))\left|\det J^{-1}(\boldsymbol{y}_n)\right|.$$
(9)

Due to the fact that $\boldsymbol{y}_{i+1}$ and $\boldsymbol{z}_{i+1}$ are the same event, it follows that $p(\boldsymbol{y}_i|\boldsymbol{y}_{i+1}) = p(\boldsymbol{z}_i|\boldsymbol{z}_{i+1})$. Applying again the change of variable rule we obtain

$$p(\boldsymbol{y}_i|\boldsymbol{y}_{i+1}) = p(\boldsymbol{z}_i|\boldsymbol{z}_{i+1})\left|\det\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{y}_i}\right|$$
$$= p(f^{-1}(\boldsymbol{y}_i)|f^{-1}(\boldsymbol{y}_{i+1}))\left|\det J^{-1}(\boldsymbol{y}_i)\right|. \quad (10)$$

To optimize (10) we first select a simple stochastic dynamical system for the latent dynamics. We choose a parameterization such that the asymptotic behavior of the system e.g., stable equilibrium point or limit cycle, is enforced. Then we select a suitable class of normalizing flows as emission function h. In this work, we will use the Euler-Maruyama [23] integration scheme to integrate the SDE that describes the dynamics of the latent space.

The resulting algorithm is summarized in Alg. 1. On each iteration, a random trajectory $\tau_{\boldsymbol{y}}$ and a sampling time ΔT are selected. By the inverse of the bijective transformation $h^{-1}(\cdot)$ the latent trajectory $\tau_{\boldsymbol{z}}$ is computed. The determinant of the inverse jacobian $|J|^{-1}$ is also computed as we require it for computing the probability in (9) and (10). The latent trajectory $\tau_{\boldsymbol{z}}$ is split to be able to compute the conditional probability $p(\boldsymbol{z}_i|\boldsymbol{z}_{i+\Delta T})$ and the probability in the end $p(\boldsymbol{z}_n)$.

Algorithm 1 ImitationFlow Learning

Input: $\mathcal{T}$ trajectories
Parameters: $\boldsymbol{\phi}$ dynamics, $\boldsymbol{\theta}$ NormalizingFlow
while not converged **do**
    $\tau_{\boldsymbol{y}} \leftarrow \{\mathcal{T}\}$
    $\Delta T \leftarrow \{\text{Get a sampling time}\}$
    $\tau_{\boldsymbol{z}}, |\mathbf{J}_{\tau_{\boldsymbol{z}}}|^{-1} \leftarrow h_{\boldsymbol{\theta}}^{-1}(\tau_{\boldsymbol{y}})$
    $\boldsymbol{z}_{(0:T-\Delta T)}, \boldsymbol{z}_{(\Delta T:T)}, \boldsymbol{z}_n \leftarrow \text{SplitTime}(\tau_{\boldsymbol{z}}, \Delta T)$
    $p(\cdot|\boldsymbol{z}_{i+\Delta T}; \boldsymbol{\phi}), p_n(\cdot; \boldsymbol{\phi}) \leftarrow \text{GetDensFunc}(\boldsymbol{z}_{(\Delta T:T)}, \boldsymbol{z}_n)$
    $\mathcal{L} = p_n(\boldsymbol{z}_n; \boldsymbol{\phi})|J_n|^{-1}\prod p(\boldsymbol{z}_i|\boldsymbol{z}_{i+\Delta T}; \boldsymbol{\phi})|J_i|^{-1}$
    $\Delta\boldsymbol{\theta}, \Delta\boldsymbol{\phi} \propto -\nabla\boldsymbol{\theta}\mathcal{L}, -\nabla\boldsymbol{\phi}\mathcal{L}$
end while

B. Latent Stochastic Linear Dynamics

A simple stochastic stable dynamic system is the stochastic Linear dynamics

$$d\boldsymbol{z}(t) = A_{\boldsymbol{\phi}}\boldsymbol{z}(t)dt + K_{\boldsymbol{\phi}}d\boldsymbol{B}(t)$$
(11)

where $A_{\boldsymbol{\phi}}$ and $K_{\boldsymbol{\phi}}$ are the learnable parameters of the latent dynamics. We impose stability by constraining the eigenvalues real part to be smaller than zero, $\mathcal{R}(\boldsymbol{\lambda}_A) < 0$.

Besides, we require to compute the stationary distribution of $\boldsymbol{z}$. To simplify the problem, we consider the stationary distribution of the discretized system. Let $F = A\Delta T + I$ and $\Sigma = KK^T\Delta T$, then, we know that

$$\lim_{t \to \infty} p(\boldsymbol{z}_t) = \mathcal{N}(\mathbf{0}, \Sigma_\infty), \qquad \Sigma_\infty = \sum_{i=0}^{\infty} F^i\Sigma F^{i\mathsf{T}},$$

where Σ_∞ can be computed in closed form in the vectorized form as

$$\text{vec}(\Sigma_\infty) = \text{vec}(\Sigma)(I_{n^2} - F \otimes F).$$
(12)

To learn the model it is also required to compute the backward conditional probability $p(\boldsymbol{z}_i|\boldsymbol{z}_{i-1})$. Linear stochastic dynamics are invertible. Therefore, we can compute the conditional backward probability, which is normally distributed

$$p(\boldsymbol{z}_i|\boldsymbol{z}_{i+1}) = \mathcal{N}(\boldsymbol{z}_i|F^{-1}\boldsymbol{z}_{i+1}, F^{-1}\Sigma F^{-\mathsf{T}}).$$

C. Stability analysis for latent stable dynamics

In the following, we prove the asymptotic stability in probability of the learned system, under the assumption of a stable latent dynamic.

Lemma 1. *For any diffeomorphic transformation $\boldsymbol{y}(t) = h(\boldsymbol{z}(t))$, if the dynamics of $\boldsymbol{z}(t)$ are stochastically asymptotically stable, then the dynamics of $\boldsymbol{y}(t)$ are also stochastically asymptotically stable.*

Proof. We follow a derivation similar to the one proposed in [12] for Lyapunov stability analysis in Ordinary Differential Equation (ODE). Let $U(\cdot)$ be a Lyapunov function for the latent dynamics. We define the following Lyapunov candidate for the observed dynamics:

$$V(\boldsymbol{y}) = U(h^{-1}(\boldsymbol{y})).$$

From this definition it follows that

$$U(\boldsymbol{z}) = V(h(\boldsymbol{z})). \qquad (13)$$

From the equality in (13) and by the fact that U is a valid Lyapunov candidate we get that the condition in (5a) is also satisfied.

To satisfy the condition in (5b), we compute the diffusion (3) of the Lyapunov function LV. Considering (3) and (8)

$$LV(\boldsymbol{y}, t) = V_t + V_{\boldsymbol{y}} J(\boldsymbol{y}) f(h^{-1}(\boldsymbol{y})) +$$
$$\frac{1}{2} \mathrm{Tr}((J(\boldsymbol{y})\boldsymbol{g}(h^{-1}(\boldsymbol{y})))^{\mathsf{T}} V_{\boldsymbol{yy}} (J(\boldsymbol{y})\boldsymbol{g}(h^{-1}(\boldsymbol{y})))). \qquad (14)$$

Moreover, we can rewrite V_t, $V_{\boldsymbol{y}}$ and $V_{\boldsymbol{yy}}$ in terms of $U_{\boldsymbol{z}}$ and $U_{\boldsymbol{zz}}$.

$$V_t(\boldsymbol{y}) = \frac{\partial}{\partial t} V(\boldsymbol{y}) = 0, \qquad (15)$$

as we don't have explicit time dependency on $V(\boldsymbol{y})$.

$$V_{\boldsymbol{y}}(\boldsymbol{y}) = \frac{\partial}{\partial \boldsymbol{y}} V(\boldsymbol{y}) = \frac{\partial}{\partial \boldsymbol{z}} U(\boldsymbol{z}) \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} = U_{\boldsymbol{z}}(\boldsymbol{z}) J^{-1}(\boldsymbol{y}), \quad (16)$$

where $U_{\boldsymbol{z}}(\cdot) : \mathbb{R}^d \to \mathbb{R}^{1 \times d}$. Finally, we can rewrite $V_{\boldsymbol{yy}}$

$$V_{\boldsymbol{yy}}(\boldsymbol{y}) = \frac{\partial^2}{\partial \boldsymbol{y}^2} V(y) = \frac{\partial}{\partial \boldsymbol{y}^{\mathsf{T}}} \left(\frac{\partial}{\partial \boldsymbol{y}} V(\boldsymbol{y}) \right) =$$
$$= \frac{\partial}{\partial \boldsymbol{y}^{\mathsf{T}}} \left(U_{\boldsymbol{z}}(\boldsymbol{z}) J^{-1}(\boldsymbol{y}) \right)$$
$$= \frac{\partial \boldsymbol{z}^{\mathsf{T}}}{\partial \boldsymbol{y}^{\mathsf{T}}} \frac{\partial}{\partial \boldsymbol{z}^{\mathsf{T}}} \left(U_{\boldsymbol{z}}(\boldsymbol{z}) J^{-1}(\boldsymbol{y}) \right)$$
$$= J^{-\mathsf{T}}(\boldsymbol{y}) U_{\boldsymbol{zz}}(\boldsymbol{z}) J^{-1}(\boldsymbol{y}). \qquad (17)$$

Introducing (15), (16) and (17) in the diffusion equation (14)

$$LV(\boldsymbol{y}) = U_{\boldsymbol{z}} f(\boldsymbol{z}) + \frac{1}{2} \mathrm{Tr}(\boldsymbol{g}^{\mathsf{T}}(\boldsymbol{z}) U_{\boldsymbol{zz}} \boldsymbol{g}(\boldsymbol{z})) = LU(\boldsymbol{z}). \quad (18)$$

By hypothesis $LU(\boldsymbol{z})$ satisfies the condition in (5b), therefore the condition is also satisfied by $LV(\boldsymbol{y})$. □

D. Latent Stochastic Limit Cycles

In two dimensional space, attractive limit cycles can be represented by Linear Dynamics in polar coordinates $\{\rho, \psi\}$, and then apply the change of Variable rule to move to cartesian coordinates. Given $\rho \in \mathbb{R}$, the radius and $\psi \in \{-\pi, \pi\}$, the angle as the latent state, the dynamics can be represented by stochastic linear dynamics

$$d\rho(t) = a_\phi(\rho(t) - \rho_\phi^*)dt + \sigma_{1\phi} dB(t) \qquad (19)$$
$$d\psi(t) = b_\phi dt + \sigma_{2\phi} dB(t), \qquad (20)$$

where $\{a_\phi, b_\phi, \rho_\phi^*\}$ are learnable linear dynamics parameters and $\{\sigma_{1\phi}, \sigma_{2\phi}\}$ are learnable standard deviation parameters. The state is transformed from polar coordinates to cartesian coordinates and back with a diffeomorphic transformation, so it allows us to apply the change of variable rule

$$x = \rho \cos(\psi) \qquad \rho = \sqrt{x^2 + y^2}$$
$$y = \rho \sin(\psi) \qquad \psi = \arctan\left(\frac{y}{x}\right).$$

Similarly to section III-B, in order to compute the loss, we need the stationary distribution $p(x_\infty, y_\infty)$. Applying the change of Variable rule, we obtain

$$p(x_\infty, y_\infty) = p(\rho_\infty, \psi_\infty) \left| \frac{\partial(\rho_\infty, \psi_\infty)}{\partial(x_\infty, y_\infty)} \right|,$$

where the probability is split between the determinant of the Jacobian for the polar coordinates and the probability of the polar coordinates in ∞. The determinant of the Jacobian is

$$\det J = \det \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{-y}{x^2+y^2} \\ \frac{y}{\sqrt{x^2+y^2}} & \frac{x}{x^2+y^2} \end{bmatrix} = (x^2 + y^2)^{-\frac{1}{2}}. \quad (21)$$

The density of the stationary distribution is the product of two separate distribution families. While, ρ evolves with a linear dynamic towards ρ^*, leading to a normal distribution, the stationary distribution for ψ depends on the initial density, $p(\psi_0)$. Unfortunately, in full generality, the density at the limit is not uniquely given, as it depends on the initial density distribution. For instance, if we assume that the initial phase distribution is uniform, the stationary density will be

$$p(\rho_\infty, \psi_\infty) = p(\rho_\infty) p(\psi_\infty) = \mathcal{N}(\rho^*, \Sigma_\infty) \mathcal{U}(-\pi, \pi).$$

where Σ_∞ is computed the same way we did in (12). We compute the conditional probability as in the previous section.

For higher dimensions, we propose to maintain the limit cycle in a two-dimensional plane and add further linear attractor dynamics towards zero for the extra coordinates.

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate the learning quality of our model in different datasets. We compare *ImitationFlow* with SEDS and Control Lyapunov Function-based Dynamic Movements (CLF-DM) in the LASA dataset [8]. Then, we show that our model is capable of representing limit cycle dynamics, for both generation and discrimination of trajectories. Finally, we show that our algorithm scales to real-world problems by learning drums playing motion in a real robot.

A. Point-to-Point trajectories

LASA dataset is a set of two-dimensional point-to-point trajectories with different shapes. This dataset is widely used for testing stable dynamics models. We learn the dynamics and we compare our results with baseline methods such as SEDS and CLF-DM in 4 metrics: SEDS error [8], Swept Area Error [10], Frechet distance and DTW error [24].

As we deal with a small set of demonstrations, the mean velocity of the demonstrated trajectories is set as the initial velocity of the latent linear dynamics. With random initialization parameters, while the model can still match the trajectory closely, it struggles to learn the proper magnitudes of velocities. For these experiments, we model the emission function as a concatenation of 10 coupling layers [19] and 10 orthogonal transformations [25].

To compute the metrics, we generate the expected trajectory from the same starting point of demonstrations and we

Fig. 3. Comparison of the performances of our method w.r.t. different state of the art algorithms. Boxplots are obtained using the LASA dataset.

Fig. 4. Examples of learned dynamics in LASA dataset. In dark blue, the expected trajectory, in light blue, noisy trajectories, in red, given trajectories

compute the similarity w.r.t the demonstrations. We compare the obtained similarity measurements with the ones obtained for SEDS and CLF-DM.

Fig. 3 shows that *ImitationFlow* outperforms the other algorithms in all the metrics, providing more accurate results. Moreover, we can see that our model is more robust than SEDS and CLF-DM: it is clear that *ImitationFlow* is the less variant and produces fewer outliers.

The generated trajectories are shown in Fig. 4. *Imitation-Flow* can learn stochastic dynamic systems that follow the trajectories provided by the user while maintaining stability. The vector field in Fig. 4 has been computed with the expected value and we can observe the variance in our model in the light blue trajectories.

B. Limit Cycle behaviours

We can model a complex limit cycle in the observation space by switching the latent linear dynamic model with a simple limit cycle behavior. To prove this, we have recorded two-dimensional handwritten data, representing different letters. The recorded demonstrations maintain similar shape, orientation, and velocity in the drawings.

Similarly to the previous section, proper initialization of the latent dynamics increases the performance of our model. We apply Principal Component Analysis (PCA) over the trajectories and we extract the main frequency of the cycle through Fourier transform. This frequency is set as the initial angular velocity. For the nonlinear emission, we considered a concatenation of ten Coupling layers with ten orthogonal transformations. The dynamics generated by *ImitationFlows* are shown in Fig. 5. The model can perfectly represent the complex demonstrated dynamics.

C. Classification for IROS letters

Normalizing Flows compute the exact distribution of the data. This property distinguishes them from other density approximators [26] that lack the normalization term. Due to this fact, we can compare the obtained probabilities and use them for classification.

We show this by applying the previously learned letter models for the limit cycles as classification modules. A simple classifier is built by computing the probability of the given trajectory and selecting the class with the highest probability

$$k^* = \arg\max_{k \in \mathcal{K}} p(\tau|k). \tag{22}$$

We have recorded ten new handwritten trajectories for each class, and we use them for testing. As shown in Fig. 6, the classifier gets a high rate of correct classification for all the shapes, with a small prediction error for R and S. The classifier is not considering any scaling or rotating term, which could improve our results. Failure mostly occurs when there is a scale or velocity discrepancy between training and testing trajectories.

D. Learning drums playing on a robot

We study the applicability of *ImitationFlow* in a real robotics application, with higher dimensionality(Cartesian Position with fixed orientation). We have recorded trajectories while doing kinesthetic teaching with a DLR-KUKA lightweight robot in drums playing task, presented in Fig. 1.

To deal with a higher dimensional problem, we switch the Coupling layer [19] with a Masked Autoregressive Flow (MAF) [20]. The orthogonal transformation remains on each output of the MAF layer. We initialize the model in a similar way to the previous section. We apply a Fourier transform in the first dimension of PCA space and the main frequency is set as the initial angular velocity.

We show in Fig. 7, a generated trajectory compared with the demonstrated data in cartesian space. We consider a starting configuration that is far from the limit cycle and generate a full trajectory offline. We show that the model can generate a stable trajectory towards the attractor.

V. RELATED WORK

Learning (globally) stable dynamical systems is a crucial topic in the Learning from Demonstration (LfD) community, as these kinds of systems provide motion generation controllers that are robust to perturbations and can generalize the

Fig. 5. Examples of learned limit cycles. The color in the vector fields represent the magnitude of the velocity, the brighter the faster. In green, trajectories generated by Imitation Flows.

Fig. 6. Confussion Matrix for I,R,O,S letters. The values in each cell, represent the probability of a test trajectory(vertical) to be classified as class(horizontal)

Fig. 7. In Blue, the generated trajectory by *ImitationFlows*. In red, samples from given demonstrations.

motion in regions of the state space where no demonstration is available. Previous works on this topic can be divided into two different categories.

The first set of approaches is based on Lyapunov stability analysis. The common idea behind these methods is to find a candidate function from a family of Lyapunov functions. One of the first examples of these approaches is the SEDS algorithm [8], that can learn globally stable dynamics towards a goal position. The candidate function is selected from the family of quadratic Lyapunov functions, while the dynamical system is learned using a finite mixture of Gaussian functions. The optimal solution is obtained by maximizing the log-likelihood of the demonstration under the stability constraint using Successive Quadratic Programming (SQP). The main limitation of SEDS is that the dynamics are restricted to contractive ones due to the limitation imposed by the quadratic Lyapunov function family. To overcome this issue, and improve the class of representable movements, the CLF-DM approach [10] has been proposed. To do so, the authors propose a richer class of Lyapunov functions, the Weighted Sum of Asymmetric Quadratic Function) (WSAQF). The method uses a Control Lyapunov Function (CLF) to find stable dynamics. Dynamics are modeled using an unstable dynamical system that is stabilized using an additional control signal. While CLF-DM is more expressive than the SEDS algorithm, the learning process can be problematic [12]. An alternative solution to the lack of representation power of SEDS is the $\tau-$SEDS algorithm [12]. Differently from the CLF-DM approach, this algorithm adds a diffeomorphic transformation on top of the stable dynamics learned by SEDS. However, an application-dependent diffeomorphic transformation must be provided by the designer. Similar in spirit to our work is [13], which also used a diffeomorphic transformation to inherit the stability properties. While in their case, a kernel-based local translation is used, in our case, we use invertible networks as bijective transformation. In [17], the learned stable dynamics were also stochastic. Anyway, the distribution family of the model was imposed and not learned from the data. The Lyapunov stability principle has been studied also in combination with neural networks. In [27], the authors learn stable dynamics and the Lyapunov candidate from data using a neural network. While they can find accurate results, stability is not guaranteed globally.

More recently in [16] also deep models have been proposed for learning stable dynamic systems. In their paper,

a deep neural network is proposed that learns the Lyapunov candidate and the system dynamics jointly by using an Input-Convex Neural Network (ICNN).

Recently, another set of approaches has been developed, building on the contraction analysis to find a globally stable solution. In [14], the authors propose a method that uses the same dynamic model as SEDS, but they exploit contraction analysis, instead of Lyapunov functions, to enforce global asymptotical stability. In [15], a contractive vector field is learned by Reproducing Kernel Hilbert Space (RKHS). This approach can learn a set of equilibrium points and constraint the local curvature using convex optimization.

There have been a few attempts of extending the Normalizing flows to time series [28], [29]. While these approaches benefit from the flexibility of normalizing flows and present remarkable results, they do not have any theoretical guarantees on the stability of the learned dynamical system.

To the best of our knowledge, our method is the only approach that has global stability guarantees for any distribution SDE, while being able to learn more complex attractors, such as limit cycles.

VI. DISCUSSION AND FUTURE WORK

In this paper, we presented a Deep Generative Model that extends the Normalizing Flows for learning stable SDEs. We have demonstrated the Lyapunov stability for a class of SDEs and propose a learning algorithm to learn them from a set of demonstrated trajectories.

We show that our model outperforms state-of-the-art algorithms for stable dynamics learning. Moreover, we have shown that our model is not limited to point-to-point dynamics, and we show how it can be extended to complex limit cycle behaviors by plugging a different latent dynamic. The density estimation property of our model has been applied in a classification task, and we show that our model can be extended to higher dimensions to solve a real robot task.

In future works, we will include conditioning to our model, to adapt the dynamics to different environments. Furthermore, we will improve the generalization capabilities of our model in areas without demonstration, to obtain smoother trajectories towards the stable attractor.

REFERENCES

[1] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems*, pp. 1040–1046, 1997.
[2] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," in *Lazy learning*, pp. 11–73, Springer, 1997.
[3] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*, pp. 261–280, Springer, 2006.
[4] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, pp. 2616–2624, 2013.
[5] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.
[6] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.
[6] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.
[8] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.
[9] A. Lemme, K. Neumann, F. Reinhart, and J. J. Steil, "Neurally imprinted stable vector fields," in *European Symposium on Artificial Neural Networks*, 2013.
[10] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.
[11] H. Ravichandar and A. Dani, "Learning contracting nonlinear dynamics from human demonstration for robot motion planning," in *ASME, Dynamic Systems and Control Conference*, 2015.
[12] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robotics and Autonomous Systems*, vol. 70, pp. 1–15, 2015.
[13] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Systems & Control Letters*, vol. 96, pp. 51–59, 2016.
[14] H. C. Ravichandar, I. Salehi, and A. P. Dani, "Learning partially contracting dynamical systems from demonstrations.," in *Conference on Robot Learning*, pp. 369–378, 2017.
[15] V. Sindhwani, S. Tu, and M. Khansari, "Learning contracting vector fields for stable imitation learning," *arXiv preprint arXiv:1804.04878*, 2018.
[16] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," in *Advances in Neural Information Processing Systems 32*, pp. 11126–11134, 2019.
[17] J. Umlauft and S. Hirche, "Learning stable stochastic nonlinear dynamical systems," in *International Conference on Machine Learning*, pp. 3502–3510, 2017.
[18] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, vol. 37, pp. 1530–1538, 2015.
[19] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," in *International Conference in Learning Representations*, 2017.
[20] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.
[21] X. Mao, *Stochastic differential equations and applications*. Elsevier, 2007.
[22] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *AAAI conference on artificial intelligence*, 2017.
[23] E. Platen and N. Bruti-Liberati, *Numerical solution of stochastic differential equations with jumps in finance*, vol. 64. Springer Science & Business Media, 2010.
[24] C. F. Jekel, G. Venter, M. P. Venter, N. Stander, and R. T. Haftka, "Similarity measures for identifying material parameters from hysteresis loops using inverse analysis," *International Journal of Material Forming*, may 2019.
[25] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.
[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Conference on Learning Representations*, 2014.
[27] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven lyapunov candidates," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1216–1222, IEEE, 2013.
[28] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma, "Videoflow: A conditional flow-based model for stochastic video generation," in *International Conference on Learning Representations*, 2020.
[29] H. S. Razaghi and L. Paninski, "Filtering normalizing flows," in *Bayesian Deep Learning Workshop at NeurIPS*, 2019.