

Discrete-Time Maximum Likelihood Neural Distribution Steering

George Rapakoulias¹ and Panagiotis Tsiotras²

Abstract—This paper studies the problem of steering the distribution of a discrete-time dynamical system from an initial distribution to a target distribution in finite time. The formulation is fully nonlinear, allowing the use of general control policies, parametrized by neural networks. Although similar solutions have been explored in the continuous-time context, extending these techniques to systems with discrete dynamics is not trivial. The proposed algorithm results in a regularized maximum likelihood optimization problem, which is solved using machine learning techniques. After presenting the algorithm, we provide several numerical examples that illustrate the capabilities of the proposed method. We start from a simple problem that admits a solution through semidefinite programming, serving as a benchmark for the proposed approach. Then, we employ the framework in more general problems that cannot be solved using existing techniques, such as problems with non-Gaussian boundary distributions and non-linear dynamics.

I. INTRODUCTION

The problem of controlling distributions of dynamical systems has attracted increased attention in the past few years, mainly due to its practical applications in machine learning and generative AI [1]. Compared to standard optimal control problems, where a feedback policy that minimizes a functional in the presence of uncertainty is sought, distribution control aims at directly steering the distribution of the system's state while minimizing a cost function [2], [3]. Apart from generative AI applications, this approach is suitable for controlling systems whose behavior is better captured by a distribution rather than a deterministic state variable. Such applications include, for example, swarm and multiagent control [4], [5], mean field games [6], [7], opinion dynamics [8], and safe stochastic model predictive control [9], [10], among many others.

This work focuses on the problem of controlling the distribution of a deterministic, discrete-time, control-affine system with nonlinear drift. The problem of interest can be cast as the following infinite-dimensional optimization problem

$$\min_{x_k, u_k} J = \sum_{k=0}^{N-1} \mathbb{E} [\|u_k\|^2 + V_k(x_k)], \quad (1a)$$

$$x_{k+1} = f_k(x_k) + B_k u_k, \quad (1b)$$

$$x_0 \sim \rho_i, \quad (1c)$$

$$x_N \sim \rho_f, \quad (1d)$$

where ρ_i, ρ_f are the boundary state distributions, f_k, B_k describe the system's prior dynamics, and $V_k(x_k)$ is a state-dependent cost that penalizes potentially undesirable regions of the state space. Problem (1) is an Optimal Mass Transport problem with (nonlinear) prior dynamics (OMTwpd). For continuous linear prior dynamics and $V(x_k) \equiv 0$, this problem is equivalent to an Optimal Transport (OT) problem in a transformed set of coordinates defined by the linear dynamics [11]. Furthermore, for a quadratic state cost $V(x_k) = x_k^\top Q_k x_k$, linear prior dynamics, and for Gaussian initial and terminal distributions, globally optimal solutions exist through the Covariance Steering (CS) framework for both continuous and discrete time settings. These can be computed efficiently using semidefinite programming (SDP) [12]–[17]. For general, nonlinear systems or systems with non-Gaussian boundary distributions, a globally optimal solution is difficult to obtain. Local solutions through linearization have been explored in [18], while solutions utilizing characteristic functions, for non-Gaussian boundary distributions and noise, have been proposed in [19]. However, both of these methods utilize linear feedback policies and therefore, may be suboptimal. Furthermore, the authors of [20] propose a randomized policy for steering between Gaussian Mixture Models (GMM) with deterministic linear prior dynamics using randomized linear policies. While the algorithm results in numerically efficient solutions with guaranteed convergence leveraging analytical results from the Covariance Steering theory, the optimality of the policy compared to more general nonlinear policies is not explored in [20].

Focusing on formulations with nonlinear dynamics, most existing works concern systems in continuous time. In the simplest case where $\dot{x}_t = u_t$, the problem has been studied in the context of Continuous Normalizing Flows (CNFs) [21], [22] due to its applications in generative AI [1]. Its stochastic counterpart, referred to as the Schrödinger Bridge Problem (SPB), has also been explored in a similar context [23]. Works that account for more complicated prior dynamics usually impose some assumptions on their structure. For example, in [8], [24] the authors account for dynamics with a nonlinear drift term but otherwise require control in all states by restricting their analysis to cases where $B_k = I$, and focus on mean field games and generative AI applications. A more general framework, allowing dynamics with fewer control channels than states is considered in [25] where the authors extended the results of [11] to feedback linearizable systems. Finally, a framework that does not require feedback linearizable dynamics is explored in [2] but the deterministic drift term is required to be the gradient of a potential function.

¹ Ph.D. student, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA. Email: grap@gatech.edu

² David and Andrew Lewis Chair and Professor, School of Aerospace Engineering, and Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, 30332, USA. Email: tsiotras@gatech.edu

The discrete-time problem has been studied much less. The case of degenerate prior dynamics, i.e., $x_{k+1} = x_k + u_k$ has been addressed within the framework of Discrete Normalizing Flows (DNFs) [26]–[28]. To the best of the authors’ knowledge, the more general problem with nonlinear prior dynamics has not been addressed in the literature. This problem is of interest for several reasons. First, it captures the case where the system dynamics are inherently discrete, as in the digital implementation of control algorithms. Furthermore, most types of neural networks can be analyzed through the lens of discrete dynamical systems, [29]–[31], providing a significant drive towards more research in the discrete setting. Finally, from a computational point of view, solving the discrete-time steering problem requires storing the state vector at a constant number of intervals, requiring, therefore, a fixed memory budget, compared to continuous formulations which perform a temporal discretization based on the stiffness of the trained dynamics [26].

To this end, in this work, we study Problem (1) using tools from machine learning and the DNF literature, specifically combining control-theoretic ideas and tools from [26]. To bring Problem (1) to the framework of Normalizing Flows, we first relax the terminal distribution constraint (1d) to a Kullback–Leibler (KL) divergence soft constraint, giving rise to the problem

$$\min_{x_k, u_k} J = \sum_{k=0}^{N-1} \mathbb{E} [\|u_k\|^2 + V_k(x_k)] + \lambda \text{D}_{\text{KL}}(\rho_N \| \rho_f), \quad (2a)$$

$$x_{k+1} = f_k(x_k) + B_k u_k, \quad (2b)$$

$$x_0 \sim \rho_i, \quad (2c)$$

where $\lambda > 0$. Henceforth, this is the general problem formulation we will use in this paper.

II. NOTATION

We use lowercase letters to denote vectors and vector random variables and capital letters to denote matrices. Given a function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, its Jacobian is denoted by $\nabla F : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$. Distributions in $\mathbb{R}^n$ are denoted by ρ and probability density functions (PDFs) by p . Given a random variable $x \sim \rho_x$ and a transformation $y = F(x)$, the pushforward of x is denoted by $\rho_y = F_{\#} \rho_x$.

III. PRELIMINARIES

A. Normalizing flows

Let ρ_1, ρ_2 be two distributions with probability density functions $p_1(x), p_2(x)$ respectively. Their KL divergence is given by

$$\text{D}_{\text{KL}}(\rho_1 \| \rho_2) = \int_{-\infty}^{\infty} p_1(x) \log \frac{p_1(x)}{p_2(x)} dx \quad (3a)$$

$$= \mathbb{E}_{x \sim \rho_1} \left[\log \frac{p_1(x)}{p_2(x)} \right] \quad (3b)$$

$$= \mathbb{E}_{x \sim \rho_1} [\log p_1(x)] - \mathbb{E}_{x \sim \rho_1} [\log p_2(x)]. \quad (3c)$$

Equation (3) suggests that calculating the KL divergence requires the analytic expressions of the PDFs of the two

functions ρ_1 and ρ_2 . In the context of Problem (2), ρ_1 would be a target distribution, which we know explicitly, while ρ_2 would correspond to the distribution of the state at some time step of interest. The calculation of ρ_2 would therefore require propagating the initial state distribution through the nonlinear dynamics, which is generally intractable. To overcome this issue, one can use the change of variables formula connecting the PDFs of two random variables that are linked through a diffeomorphic (invertible and differentiable) transformation. This is summarized in the following lemma:

Lemma 1. [32] (*Change of Variables*) *Let x be an n -dimensional random variable with known PDF, denoted $p_x(x)$, and let $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a diffeomorphism. The PDF of $y = F(x)$, denoted $p_y(y)$, can be calculated using the formula*

$$p_y(y) = p_x(F^{-1}(y)) \det \nabla F^{-1}(y). \quad (4)$$

Transformations that correspond to flow maps of continuous-time systems are invertible, as far as unique solutions exist for the differential equation describing their dynamics. In CNF problems without prior dynamics, this condition is satisfied because neural networks with finite weights and Lipschitz nonlinearities result in Lipschitz ODE dynamics, and the uniqueness of solutions is guaranteed through Picard’s existence theorem [21]. In the discrete-time case, however, the invertibility of the network needs to be addressed explicitly. In this paper, we make use of the following lemma:

Lemma 2. [26] (*Flow Invertibility*) *Consider the discrete-time nonlinear system described by*

$$x_{k+1} = x_k + f_k(x_k), \quad k = 0, 1, \dots, N-1, \quad (5)$$

and the transformation $x_N = F(x_0)$. The transformation F is invertible if, for all $k = 0, 1, \dots, N-1$, the mappings f_k are contractive.

For a different technique that requires modeling the individual state transition functions as gradients of a convex potential, we refer the reader to [27]. Finally, one result that facilitates the derivation of the proposed algorithm is the equivalence of the KL-divergence before and after a diffeomorphic transformation.

Lemma 3. [33] (*Forward and Backward KL divergence*) *Let $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a diffeomorphism and let ρ_1, ρ_2 be two distributions. Then*

$$\text{D}_{\text{KL}}(\rho_1 \| \rho_2) = \text{D}_{\text{KL}}(F_{\#} \rho_1 \| F_{\#} \rho_2). \quad (6)$$

B. Exact Steering Using Semidefinite Programming

For the special case of a system with linear dynamics of the form

$$x_{k+1} = Ax_k + Bu_k, \quad (7)$$

and for Gaussian initial and terminal state distributions, the optimal policy for Problem (1) is parametrized by an affine feedback controller of the form [13]

$$\pi_k(x_k) = K_k(x_k - \mu_k) + v_k, \quad (8)$$

where $\mu_k = \mathbb{E}[x_k]$, while its solution, i.e., the calculation of $\{K_k, v_k\}$ for $k = 0, 1, \dots, N-1$, can be attained efficiently through semidefinite programming. The soft-constrained version (2) has only been studied for a Wasserstein-2 soft constraint penalty function in [14]. The KL-divergence soft-constrained version can be solved similarly. Although this paper focuses on the nonlinear version of the problem, the case of linear prior dynamics with Gaussian boundary conditions will be used as a benchmark, to validate the accuracy of the proposed algorithm. To this end, we briefly present how one can calculate the optimal solution to Problem (2) with a controller of the form (8) using semidefinite programming. Although we do not explicitly prove that this family of policies is globally optimal for the problem in this paper, this has been shown to be the case for the hard constrained Problem (1) in [13].

C. KL-divergence Covariance Steering

When the dynamics of the system are linear and a control policy of the form (8) is used, the first two moments of the state can be calculated explicitly at any time instant k in the steering horizon. The corresponding equations are

$$\mu_{k+1} = A\mu_k + Bv_k, \quad (9a)$$

$$\Sigma_{k+1} = (A + BK_k)\Sigma_k(A + BK_k)^\top, \quad (9b)$$

where $\Sigma_k = \text{cov}(x_k)$. The KL-divergence between the terminal distributions $\rho_N = \mathcal{N}(\mu_N, \Sigma_N)$ and $\rho_f = \mathcal{N}(\mu_f, \Sigma_f)$ can be calculated via [34]

$$\begin{aligned} D_{\text{KL}}(\rho_N \parallel \rho_f) = & \frac{1}{2}(\text{tr}(\Sigma_f^{-1}\Sigma_N) + (\mu_f - \mu_N)^\top \Sigma_f^{-1}(\mu_f - \mu_N) \\ & + \log \det(\Sigma_f) - \log \det(\Sigma_N) - n), \end{aligned}$$

where n is the dimension of the state vector. Using equations (9), the change of variables $K_k = \Sigma_k^{-1}U_k$, and $Y_k = U_k \Sigma_k^{-1}U_k$ in Problem (2) yields

$$\min J = \sum_{k=0}^{N-1} \text{tr}(Y_k) + \|v_k\|^2 + \lambda D_{\text{KL}}(\rho_N \parallel \rho_f), \quad (11a)$$

$$U_k \Sigma_k^{-1}U_k = Y_k, \quad (11b)$$

$$\Sigma_{k+1} = A\Sigma_k A^\top + BU_k + U_k^\top B^\top + BY_k B^\top, \quad (11c)$$

$$\mu_{k+1} = A\mu_k + Bv_k, \quad (11d)$$

$$\Sigma_0 = \Sigma_i, \quad (11e)$$

$$\mu_0 = \mu_i, \quad (11f)$$

$$\Sigma_N = \Sigma_f, \quad (11g)$$

$$\mu_N = \mu_f. \quad (11h)$$

Relaxing (11b) to the semidefinite inequality $U_k \Sigma_k^{-1}U_k \preceq Y_k$ turns Problem (11) into a semidefinite program. This relaxation has been proven to be lossless in [13], [14]. Finally, we note that the term $-\log \det(\Sigma_f)$ in the KL-divergence is convex with respect to Σ_f and can be added to the cost function using appropriate slack variables accompanied by an LMI constraint [35].

IV. MAXIMUM LIKELIHOOD DISTRIBUTION STEERING

This section contains the main results of the paper. To this end, consider Problem (2) with a policy parametrized by a neural network, i.e., $u_k = \pi_k(x_k; \theta)$ where θ corresponds to the trainable policy parameters. To optimize (2), tractable expressions for the cost function (2a) must be developed. The first step is arguably the calculation of the KL divergence. Calculating it directly would involve the explicit calculation of the probability density of the state at the end of the steering horizon, p_N , which is challenging due to the nonlinearities of the system. Instead of calculating p_N directly, let $F(x_0) = \Phi_{N-1} \circ \Phi_{N-2} \circ \dots \circ \Phi_0(x_0)$ denote the transformation linking the initial and terminal states under the discrete dynamic model (1b), where

$$\Phi_k(x_k) = f_k(x_k) + B_k \pi_k(x_k), \quad (12)$$

is the closed-loop state transition function at time step k . Under certain conditions on the control policy that will be specified later in the section, this transformation is diffeomorphic. Therefore, its inverse $x_0 = F^{-1}(x_N)$ satisfies the conditions of Lemma 3. Applying this result to the KL divergence yields

$$D_{\text{KL}}(\rho_N \parallel \rho_f) = D_{\text{KL}}(\rho_i \parallel F_{\#}^{-1} \rho_f).$$

Further expanding the second term using the definition of the KL divergence, yields

$$D_{\text{KL}}(\rho_i \parallel F_{\#}^{-1} \rho_f) = \mathbb{E}_{x \sim \rho_i} [\log p_i(x)] - \mathbb{E}_{x \sim \rho_i} [\log p_0(x)],$$

where $p_i(x)$ is the PDF of ρ_i and $p_0(x)$ is the PDF of the distribution $F_{\#}^{-1} \rho_f$, that is, the density of a random variable sampled from ρ_f and pushed through the inverse transformation F^{-1} . Notice that the term $\mathbb{E}_{x \sim \rho_i} [p_i(x)]$ does not depend on the control policy parameters, and can therefore be omitted from the cost function of (2).

The calculation of $\log p_0(x)$ can be facilitated through Lemma 1. Specifically, one can link the density p_0 with the density of p_f using

$$\log p_0(x) = \log p_f(F(x)) + \log \det \nabla F(x).$$

The second term can also be efficiently calculated using the chain rule as follows

$$\log \det \nabla F(x) = \sum_{k=0}^{N-1} \log \det \nabla \Phi_k(x_k).$$

In our implementation, the Jacobian of the state transition functions $\nabla \Phi_k(x_k)$ were calculated using automatic differentiation.

Finally, we discuss conditions for $\pi_k(x_k; \theta)$ that preserve the invertibility of the discrete-time dynamics. Without loss of generality, we assume that the state transition functions (12) are of the form

$$x_{k+1} = x_k + \phi_k(x_k) + B_k \pi_k(x_k; \theta). \quad (13)$$

Proposition 1. *Let the system dynamics be described by (13), and let L_{ϕ_k}, L_{π_k} be the Lipschitz constants of ϕ_k, π_k ,*

respectively, and let σ_{B_k} be the spectral norm of the matrix B_k . Then, if $L_{\pi_k} < (1 - L_{\phi_k})/\sigma_{B_k}$, the state transition function defined in (13) is a diffeomorphism.

Proof. Based on Lemma 2, it suffices to show that $\phi(x_k) + B_k\pi_k(x_k)$ is a contraction. To upper bound its Lipschitz constant note that $\|\nabla(\phi_k + B_k\pi_k(x_k))\|_2 \leq \|\nabla\phi_k\|_2 + \|B_k\nabla\pi_k\|_2 \leq L_{\phi_k} + \sigma_{B_k}L_{\pi_k}$, due to the subadditivity and submultiplicativity of the spectral norm [36]. Constraining this upper bound yields the desired result. $\square$

Remark. In the case where f_k, B_k in (1b) are discretized versions of the continuous dynamics $\dot{x}_t = f_t(x_t) + B_t u_t$, then the first order approximation of the terms in (13) are $\phi_k(x_k) = \Delta T f_t(x_k)$ and $B_k = \Delta T B_t$, where ΔT is the discretization step size. Therefore, for Lipschitz continuous-time dynamics, L_{ϕ_k} and σ_{B_k} can be made sufficiently small by reducing the discretization step ΔT .

Note that training Neural Networks with bounded Lipschitz constants can be achieved using spectral normalization [37]. In this work, we use $\pi_k = \alpha L_{\pi_k} \hat{\pi}_k$ where $L_{\pi_k} = (1 - L_{\phi_k})/\sigma_{B_k}$, $\alpha \in (0, 1)$ and $\hat{\pi}_k$ is a Multilayer Perceptron (MLP) with spectral normalization in all of its weights, having therefore a Lipschitz constant of 1.

After calculating the loss function, optimization is carried out using standard gradient-based optimizers. For our implementation, we used the AdamW [38] scheme, implemented in pytorch [39].

V. NUMERICAL EXAMPLES

In the first numerical example, we study the problem of driving a double integrator system from an initial to a terminal Gaussian distribution. Since an exact solution can be obtained for this problem using the results from Section III-C, we use this example as a benchmark. To this end, consider the discrete-time deterministic dynamics of the form (7) with

$$A = \begin{bmatrix} I_2 & \Delta T I_2 \\ 0_2 & I_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0_2 \\ \Delta T I_2 \end{bmatrix}, \quad \Delta T = 0.1,$$

a horizon of $N = 30$ time steps, $V_k(x_k) \equiv 0$ and $\lambda = 60$, which is equivalent with normalizing $\|u_k\|^2$ with $1/(Nm)$ where m is the number of input channels. We use this value for λ for all the subsequent examples. The boundary distributions are $x_0 \sim (\mu_i, \Sigma_f)$ and $x_N \sim \mathcal{N}(\mu_f, \Sigma_f)$ with parameters $\mu_i = [0, 0, 5, 8]$, $\Sigma_i = \text{blkdiag}(1, 1, 0.2, 0.2)$, $\mu_f = [0, 0, 10, 0]$, $\Sigma_f = 0.4I_4$. For this example, the Lipschitz constants of the prior dynamics are $L_\phi = \sigma_B = \Delta T$. To this end, we set $L_\pi = 9$, $\alpha = 0.9$ and $\pi_k = \alpha L_{\pi_k} \hat{\pi}_k$. Each policy $\hat{\pi}_k(\cdot)$ is modeled using a fully connected MLP with spectral normalization, and five layers with $\{4, 64, 64, 64, 2\}$ neurons per layer. The convergence plot, along with the optimal solution calculated using the SDP technique described in Section III-C can be viewed in Figure 1.

In the second numerical example, we use double integrator dynamics, a horizon of $N = 40$, but this time, we opt to steer from a Gaussian Mixture Model with 8 modes to

the Normal distribution in the presence of obstacles. The obstacles are modeled using appropriate potential fields with Gaussian kernels [40] of the form

$$V_k(x) = \lambda_{\text{obs}} \exp\left(-\frac{(x - x_0)^2}{r_{\text{obs}}^2}\right). \quad (14)$$

The policy at each time step has the same structure as in the first example but with 128 neurons in the hidden layers. The results are illustrated in Figure 2.

Another, more complicated example, which capitalizes on the fact that only a batch of samples is required for the initial distribution rather than its PDF, is depicted in Figure 3, where the initial distribution is arbitrary and the terminal is the normal distribution. The prior dynamics and policy parametrization are identical to Example 2. We note that the inverse problem, i.e., steering from a Gaussian distribution to an arbitrary distribution for which only samples are available, would be significantly harder and cannot currently be solved with the proposed approach, since explicit information about the PDF of the terminal distribution is required for the maximum likelihood training. We leave the investigation of this case as part of future work.

Finally, we test the proposed method in the 2D nonlinear model

$$x_{k+1} = x_k + 0.1\sqrt{1 + y_k^2} + u_k, \quad (15a)$$

$$y_{k+1} = y_k + 0.1x_k, \quad (15b)$$

for $N = 40$. By bringing these equations in the form of (13) with $L_B = \sigma_B = 0.1$, we may use the same policy parametrization as in the first example. The results are illustrated in Figure 4. Since this is a two-dimensional example, we can overlay the samples with the contour lines of the PDF at each time step to validate the precision of the solution.

Table I demonstrates a quantitative evaluation of the algorithm's performance. The first column corresponds to the experiment number. To measure the distance between the distribution of the state at the final time step of the steering horizon and the target distribution, we calculate the 2-Wasserstein distance using discrete Optimal Transport [32] and report its value in the second column. We use the 2-Wasserstein distance because it can be computed exactly on empirical distributions that are available only through samples and accurately reflects the actual distance between the continuous distributions given enough samples. In the third column, we report the minimum value of the log-determinant of the Jacobian of the optimal map linking the initial and final state in order to validate the invertibility of the computed map and finally provide the total training time in minutes in the last column. Training was performed on an Nvidia RTX-3070 GPU.

VI. CONCLUSIONS

This paper presents a method for solving the distribution steering problem for discrete-time nonlinear systems by

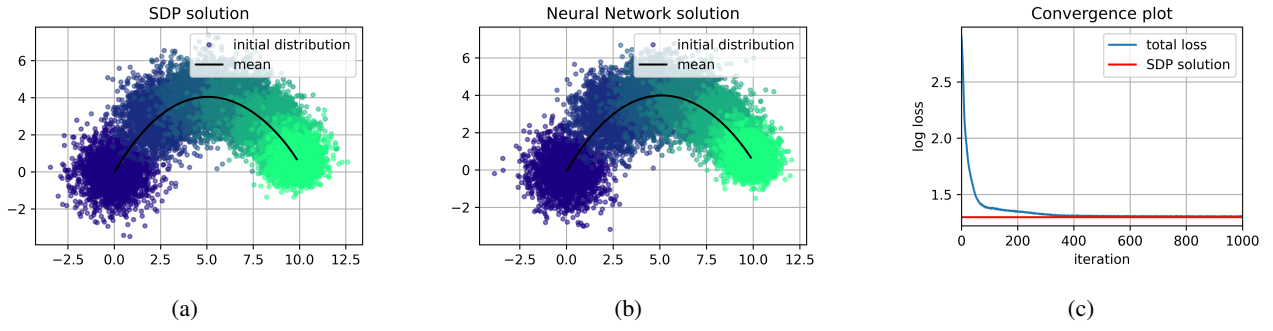


Fig. 1: (a) Exact SDP solution (b) Neural Network solution (c) Convergence plot along with optimal cost calculated using the SDP method. For figures (a), (b) the axes correspond to the first two states of the 2D double integrator.

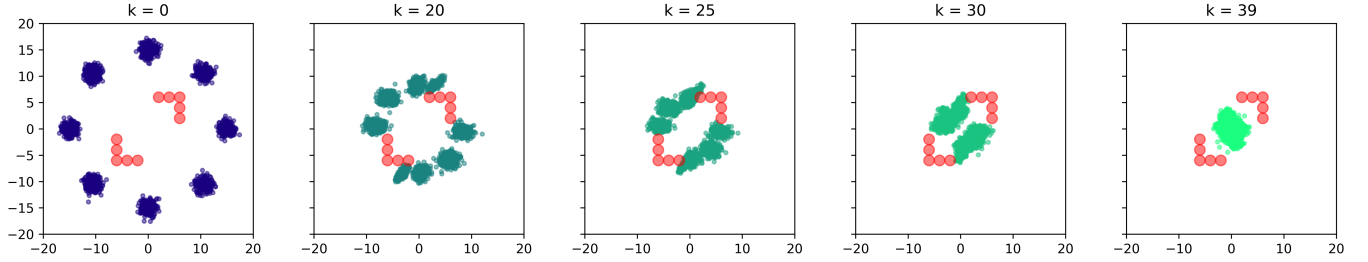


Fig. 2: GMM to Gaussian with given mean and covariance, double integrator prior dynamics and obstacles.

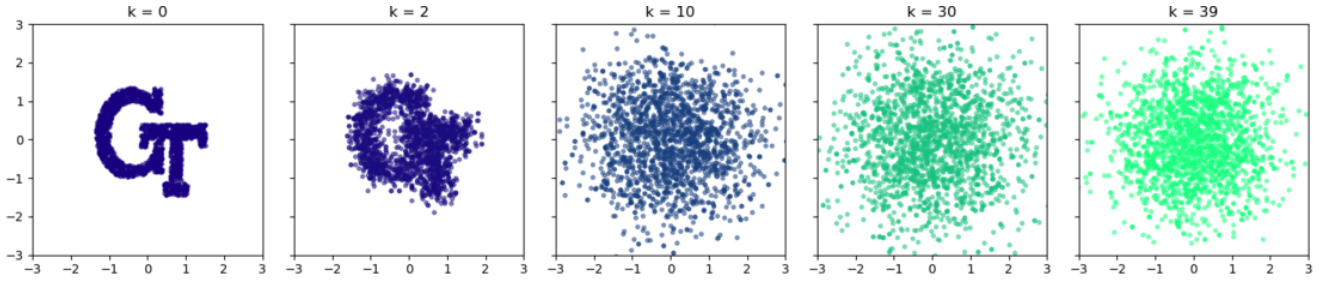


Fig. 3: GT to Gaussian distribution steering with given mean and covariance with double integrator prior dynamics.

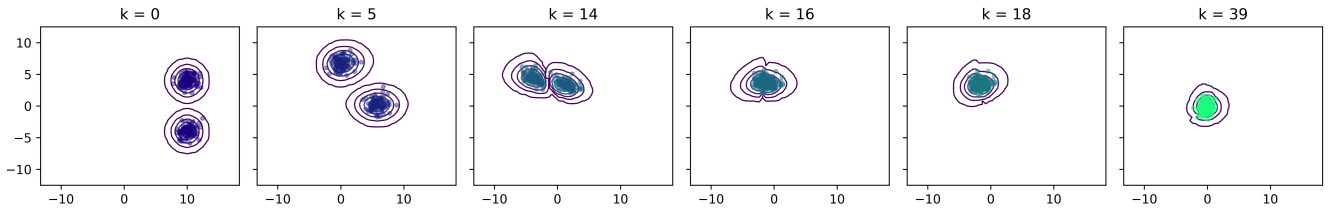


Fig. 4: GMM to Gaussian distribution steering with nonlinear prior dynamics. The axes correspond to the system states.

TABLE I: Quantitative analysis of the proposed approach.

Exp. #	$\mathbb{W}(\rho_N \parallel \rho_f)$	$\min \log \det \nabla F $	Training time [m]
1	1.12	0.32	4.1
2	1.56	3.19	20.1
3	0.64	0.42	13.53
4	0.18	0.17	10.5

formulating it as a regularized maximum likelihood optimization problem. The control policies are parametrized using neural networks with appropriate Lipschitz constraints to ensure the invertibility of the discrete-time dynamics. A

general cost function is considered, allowing state-dependent terms to model obstacles in the state space using potential fields. In parallel, a KL-divergence soft constraint version of the Covariance Steering problem is developed as a benchmark to compare with the proposed nonlinear maximum likelihood methods. Finally, four comprehensive numerical examples are presented and analyzed with respect to how closely they achieve the target distribution, as well as in terms of run time. For the linear dynamics with Gaussian boundary distributions, the solution is also compared against the globally optimal solution calculated as the solution of a

semidefinite program.

ACKNOWLEDGMENT

The authors would like to sincerely thank Dr. Ali Reza Pedram for his comments in an initial version of the manuscript and for multiple fruitful discussions. Support for this work has been provided by ONR award N00014-18-1-2828 and NASA ULI award #80NSSC20M0163. This article solely reflects the opinions and conclusions of its authors and not of any NASA entity.

REFERENCES

- [1] L. Ruthotto and E. Haber, “An introduction to deep generative modeling,” *GAMM-Mitteilungen*, vol. 44, no. 2, p. e202100008, 2021.
- [2] K. F. Caluya and A. Halder, “Wasserstein proximal algorithms for the Schrödinger bridge problem: Density control with nonlinear drift,” *Transactions on Automatic Control*, vol. 67, no. 3, pp. 1163–1178, 2021.
- [3] Y. Chen, T. T. Georgiou, and M. Pavon, “Controlling uncertainty,” *Control Systems Magazine*, vol. 41, no. 4, pp. 82–94, 2021.
- [4] A. D. Saravanos, Y. Li, and E. A. Theodorou, “Distributed hierarchical distribution control for very-large-scale clustered multi-agent systems,” in *Robotics: Science and Systems XIX*, (Daegu, Republic of Korea), July 2023.
- [5] A. D. Saravanos, A. Tzolovikos, E. Bakolas, and E. Theodorou, “Distributed Covariance Steering with Consensus ADMM for Stochastic Multi-Agent Systems,” in *Proceedings of Robotics: Science and Systems*, (Virtual), July 2021.
- [6] L. Ruthotto, S. J. Osher, W. Li, L. Nurbekyan, and S. W. Fung, “A machine learning framework for solving high-dimensional mean field game and mean field control problems,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 17, pp. 9183–9193, 2020.
- [7] Y. Chen, “Density control of interacting agent systems,” *Transactions on Automatic Control*, vol. 69, no. 1, pp. 246–260, 2024.
- [8] G.-H. Liu, T. Chen, O. So, and E. Theodorou, “Deep generalized Schrödinger bridge,” in *Advances in Neural Information Processing Systems*, vol. 35, (Louisiana, LA), pp. 9374–9388, Curran Associates, Inc., 2022.
- [9] J. Knaup, K. Okamoto, and P. Tsiotras, “Safe high-performance autonomous off-road driving using covariance steering stochastic model predictive control,” *Transactions on Control Systems Technology*, vol. 31, no. 5, pp. 2066–2081, 2023.
- [10] A. D. Saravanos, I. M. Balci, E. Bakolas, and E. A. Theodorou, “Distributed model predictive covariance steering,” *arXiv preprint arXiv:2212.00398*, 2022.
- [11] Y. Chen, T. T. Georgiou, and M. Pavon, “Optimal transport over a linear dynamical system,” *Transactions on Automatic Control*, vol. 62, no. 5, pp. 2137–2152, 2016.
- [12] E. Bakolas, “Finite-horizon covariance control for discrete-time stochastic linear systems subject to input constraints,” *Automatica*, vol. 91, pp. 61–68, 2018.
- [13] F. Liu, G. Rapakoulias, and P. Tsiotras, “Optimal covariance steering for discrete-time linear stochastic systems,” *arXiv preprint arXiv:2211.00618*, 2022.
- [14] I. M. Balci and E. Bakolas, “Exact SDP formulation for discrete-time covariance steering with Wasserstein terminal cost,” *arXiv preprint arXiv:2205.10740*, 2022.
- [15] Y. Chen, T. T. Georgiou, and M. Pavon, “Optimal steering of a linear stochastic system to a final probability distribution, part I,” *Transactions on Automatic Control*, vol. 61, no. 5, pp. 1158–1169, 2015.
- [16] Y. Chen, T. T. Georgiou, and M. Pavon, “Optimal steering of a linear stochastic system to a final probability distribution, part II,” *Transactions on Automatic Control*, vol. 61, no. 5, pp. 1170–1180, 2015.
- [17] G. Rapakoulias and P. Tsiotras, “Discrete-time optimal covariance steering via semidefinite programming,” in *62nd Conference on Decision and Control*, (Singapore), pp. 1802–1807, 2023.
- [18] J. Ridderhof, J. Pilipovsky, and P. Tsiotras, “Chance-constrained covariance control for low-thrust minimum-fuel trajectory optimization,” in *2020 AAS/AIAA Astrodynamics Specialist Conference*, pp. 9–13, 2020.
- [19] V. Sivaramakrishnan, J. Pilipovsky, M. Oishi, and P. Tsiotras, “Distribution steering for discrete-time linear systems with general disturbances using characteristic functions,” in *American Control Conference*, (Atlanta, GA), pp. 4183–4190, 2022.
- [20] I. Balci and E. Bakolas, “Density steering of gaussian mixture models for discrete-time linear systems,” *arXiv preprint arXiv:2311.08500*, 2023.
- [21] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” in *Advances in Neural Information Processing Systems*, vol. 31, (Montreal, Canada), pp. 6571–6584, Curran Associates, Inc., 2018.
- [22] D. Onken, S. W. Fung, X. Li, and L. Ruthotto, “OT-flow: Fast and accurate continuous normalizing flows via optimal transport,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, (held virtually), pp. 9223–9232, 2021.
- [23] Y. Chen, T. T. Georgiou, and M. Pavon, “On the relation between optimal transport and Schrödinger bridges: A stochastic control viewpoint,” *Journal of Optimization Theory and Applications*, vol. 169, pp. 671–691, 2016.
- [24] T. Chen, G.-H. Liu, and E. A. Theodorou, “Likelihood training of Schrödinger bridge using forward-backward SDEs theory,” in *International Conference on Learning Representations*, (held virtually), April 2022.
- [25] K. F. Caluya and A. Halder, “Finite horizon density steering for multi-input state feedback linearizable systems,” in *American Control Conference*, pp. 3577–3582, IEEE, 2020.
- [26] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, “Invertible residual networks,” in *International Conference on Machine Learning*, pp. 573–582, PMLR, 2019.
- [27] C.-W. Huang, R. T. Chen, C. Tsirigotis, and A. Courville, “Convex potential flows: Universal probability distributions with optimal transport and convex optimization,” in *International Conference on Learning Representations*, (held virtually), April 2020.
- [28] R. Van Den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling, “Sylvester normalizing flows for variational inference,” in *34th Conference on Uncertainty in Artificial Intelligence*, (Monterey, CA), pp. 393–402, Association For Uncertainty in Artificial Intelligence (AUAI), 2018.
- [29] E. Haber and L. Ruthotto, “Stable architectures for deep neural networks,” *Inverse Problems*, vol. 34, no. 1, p. 014004, 2017.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.
- [31] J. Pilipovsky, V. Sivaramakrishnan, M. Oishi, and P. Tsiotras, “Probabilistic verification of relu neural networks via characteristic functions,” in *Learning for Dynamics and Control Conference*, (Philadelphia, PA), pp. 966–979, PMLR, 2023.
- [32] G. Peyré and M. Cuturi, *Computational Optimal Transport: With Applications to Data Science*. Foundations and Trends in Machine Learning, 2019.
- [33] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing flows for probabilistic modeling and inference,” *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 2617–2680, 2021.
- [34] J. Duchi, “Derivations for linear algebra and optimization,” *Berkeley, California*, vol. 3, no. 1, pp. 2325–5870, 2007.
- [35] M. ApS, “Mosek modeling cookbook,” 2020.
- [36] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas*. Princeton University Press, 2009.
- [37] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *International Conference on Learning Representations*, (Vancouver, Canada), April 2018.
- [38] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, (Vancouver, Canada), April 2018.
- [39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, vol. 32, (Vancouver, Canada), 2019.
- [40] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, “A neural network approach for high-dimensional optimal control applied to multiagent path finding,” *Transactions on Control Systems Technology*, vol. 31, no. 1, pp. 235–251, 2022.