Journal of Computational Physics 519 (2024) 113442

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

SDYN-GANs: Adversarial learning methods for multistep generative models for general order stochastic dynamics [☆]

P. Stinis [a], C. Daskalakis [b], P.J. Atzberger [c,d,*]

[a] Advanced Computing, Mathematics and Data Division, Pacific Northwest National Laboratory (PNNL), Richland, WA, United States of America
[b] Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA, United States of America
[c] Department of Mathematics, University of California Santa Barbara (UCSB), United States of America
[d] Department of Mechanical Engineering, University of California Santa Barbara (UCSB), United States of America

ABSTRACT

We introduce adversarial learning methods for data-driven generative modeling of dynamics of $\mathbf{n}^{th}$-order stochastic systems. Our approach builds on Generative Adversarial Networks (GANs) with generative model classes based on stable m-step stochastic numerical integrators. From observations of trajectory samples, we introduce methods for learning long-time predictors and stable representations of the dynamics. Our approaches use discriminators based on Maximum Mean Discrepancy (MMD), training protocols using both conditional and marginal distributions, and methods for learning dynamic responses over different time-scales. We show how our approaches can be used for modeling physical systems to learn force-laws, damping coefficients, and noise-related parameters. Our adversarial learning approaches provide methods for obtaining stable generative models for dynamic tasks including long-time prediction and developing simulations for stochastic systems.

1. Introduction

Many learning and modeling tasks arising in statistical inference, machine learning, engineering, and the sciences involve inferring representations for systems exhibiting non-linear stochastic dynamics. This requires estimating models that not only account for the observed data but also allow for performing simulations or making predictions of behaviors over longer time-scales. Obtaining stable long-time simulations and predictions requires data-driven approaches capable of learning structured dynamical models from observations. Estimating parameters and representations for stochastic dynamics has a long history in the statistics and engineering community [66,103,92]. This includes methods for inferring models and parameters in robotics, orbital mechanics, geology, political science, finance, and economics [25,106,92]. One of the most prominent strategies for developing estimators has been to use Maximum Likelihood Estimation (MLE) or Bayesian Inference (BI). For systems where likelihoods can be specified readily and computed tractably, estimators can be developed with asymptotically near optimal data efficiency [150]. As we shall discuss, for non-linear stochastic dynamics this is often challenging in practice and we show how alternatives can be developed based on adversarial learning approaches.

Early methods include Wiener and Kalman Filtering [151,72], which was originally designed to estimate underlying states from knowledge of the underlying linear dynamics and a sequence of observations from noisy sensor measurements [22,56]. To further handle non-linear dynamics and propagation of sufficient statistics for the conditional distributions, extended methods also were

[☆] Work supported by DOE Grant ASCR PhILMS DE-SC0019246 and NSF Grant DMS-1616353.

[*] Corresponding author.
E-mail address: atzberg@gmail.com (P.J. Atzberger).
URL: http://atzberger.org/ (P.J. Atzberger).

https://doi.org/10.1016/j.jcp.2024.113442
Received 4 May 2023; Received in revised form 17 August 2024; Accepted 13 September 2024

Available online 24 September 2024

introduced using linearizations or other approximations, such as in Extended Kalman Filters [81,24,149,161] or use of specialized sampling points as in Unscented Kalman Filters [70,37,149]. For some tasks, these were further applied to model estimation by appending system parameters as pseudo-state variables to be estimated as part of the state filtering. This included Joint Methods [149,22] or Dual Methods [65,149,148], with the latter using alternating iterations between filtering and parameter estimation using Expectation-Maximization (EM) procedures [26,63]. These approaches attempt to estimate parameters of the stochastic processes using loss functions based on the probability densities associated with likelihoods in MLE [66] or related Bayesian formulations [66,150]. In practice, the variants of Kalman Filtering have primarily been designed for estimation based on L^2-errors and propagation of variances which tacitly works best when there is Gaussian or unimodal statistics. The methods have been most effective when the underlying dynamics is close to deterministic or there is extensive prior knowledge about system behaviors constraining the model parameterizations [37,22,56,81].

As an alternative to such approaches and MLE methods, we develop sample-based adversarial learning methods using discriminators/critics based on a generalization of the method-of-moments building on [62,55,41]. Our sample-based approaches do not require specification of likelihood functions facilitating use of more general classes of generative models. We also develop methods for learning dynamic models of $\mathfrak{n}^{th}$-order stochastic processes using a class of generative models based on stable m-step stochastic numerical integrators. For gradient-based learning with loss functions based on higher-order statistics, and the high dimensional samples arising from trajectory observations, we develop specialized adjoint methods and custom backpropagation methods. Our approaches provide Generative Adversarial Networks (GANs) for learning generative models for the non-linear dynamics of stochastic systems. We refer to the methods as Stochastic Dynamic Generative Adversarial Neural Networks (SDYN-GANs).

In the literature there have been many system identification approaches and data-driven methods developed for identifying underlying governing equations [121,140,14]. These have primarily been developed for deterministic dynamical systems. A widely-used strategy is to perform an L^2-error analysis and to handle observational noise by performing variants of least-squares-like methods to construct models for the underlying deterministic dynamics [121,140,82,79]. Prominent linear methods include Dynamic Mode Decompositions (DMD) [121,140], Koopman Methods [82,79,97], operator learning approaches [60,107], and Kalman Filtering with Expectation Maximization (K-EM) [47,119]. Further extensions were developed to obtain non-linear models or learn latent space representations in [73,46,43,123,93]. This includes SINDy [40], PINNs [20], extended-DMD [18], and Extended Kalman Filters [65,81,149]. In the case of stochastic dynamical systems, a new set of challenges arises since additional sources of noise drive the underlying dynamics. This can be especially challenging when systems have significant diffusive contributions. As we shall discuss, SDYN-GANs provides methods to address these and other aspects of stochastic systems by using a more general error analysis that makes more direct comparisons between the underlying probability measures of the learned models and ensembles of observed trajectory data. This allows for learning generative models for the diverse probabilistic behaviors that can be exhibited by stochastic systems.

There have also been many neural network methods proposed for learning predictions for time-series based on linear and non-linear approaches [85,91,109,124]. Early methods have been based on feedfoward neural networks [154,163,31] and discrete and continuous-time recurrent neural networks (RNNs) [23,64,31,124]. For RNNs backpropagation methods where extended over time to compute gradients for training [111,53]. For trajectories this can lead to well-known issues with vanishing and exploding gradients [53,133,91,98,110,9]. For RNNs this led to the introduction of Long-Short-Term-Memory (LSTM) [64], Gated Recurrent Units (GRU) units [21,28], and more recently as alternatives, the Transformer architectures [144,32]. The early methods were primarily developed for deterministic systems subjected to noise perturbations [31] and were aimed at performing forecasting [23,31,154]. For long-time prediction or simulation using RNNs, and other neural network methods, there are persistent issues with instabilities in the generated dynamics over long-time horizons or within the internal transformations [139,7,98,162,38,45,33,9]. While some of this can be mitigated by training procedures [98,114,87,110], this often requires inputs remain close to the training data sequences. This can pose challenges especially for stochastic systems that involve diffusive contributions covering large parts of the state-space.

More recent methods for stochastic systems using GANs-like approaches to model deterministic and stochastic dynamics, include Neural-ODEs/SDEs [19,74] and methods motivated by problems in physics [156,6,130,158,157,159,89]. Recent GANs methods are closely related to prior adjoint and system identification methods developed in the engineering communities [90,49,92,17] and finance communities [71,48]. In these GANs methods, Maximum Likelihood Estimation (MLE) is replaced with either the original GANs approximations to Jensen-Shannon (JS) metric by using neural network classifier discriminators [54] or by approximations to the Wasserstein metric [3] based on neural networks with Gradient Penalties (WGAN-GP) [59]. Recent work also has introduced other methods, including energy-based methods [164], Wasserstein metric approaches based on marginal slices [27,78], and other sample comparison approaches [30,8,116,11,36]. For time-series analysis, probabilistic graphical models also have been combined with GANs to enhance training and characterized by establishing subadditivity bounds in [29,30]. The use of general neural network discriminators is known to be challenging to train for GANs. This manifests as oscillations during training with well-known issues with convergence and stability [77,136]. While Neural ODEs/SDEs have been introduced for gradient training [19,141], they make approximations to simplify the gradient calculations, but these can result in inconsistencies with the specific discretizations chosen to represent the stochastic dynamics. These methods also treat dynamics primarily of first-order systems.

In our SDYN-GANs approach, we formulate training methods by developing a set of adjoint methods for computing gradients for general $\mathfrak{n}^{th}$-order dynamics and classes of m-step methods. We take into account the roles of the discretization errors, which we find is important for stability and consistency during training. In our methods, this ensures our training gradients align with our specified loss functions for our structured m-step generative model classes. While our adversarial learning framework can be used more generally, we further mitigate oscillations and other issues in training by focusing on using discriminators/critics D which are analytically tractable. We develop discriminators/critics D for the dynamic setting using a collection of feature maps which can

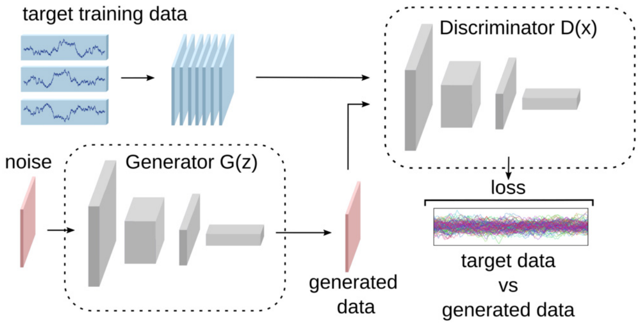

Fig. 1. Adversarial learning. A generative model $G(\cdot;\theta_G)$ and discriminator/critic $D(\cdot;\theta_D)$ are used to learn models from samples of the trajectory data. The D is used in computing a loss comparing samples $\{\tilde{\mathbf{x}}_i\}_{i=1}^{\tilde{N}}$ generated by G with the samples $\{\mathbf{x}_i\}_{i=1}^{N}$ of the training data.

be viewed as employing kernel methods to encode and embed probability distributions into a Reproducing Kernel Hilbert Space (RKHS) [55,127,120]. The comparisons between the samples of the candidate models and the observation data correspond to a set of losses related to Generalized Moments Methods (GMMs) [62] and Maximum Mean Discrepancy (MMD) [55,41].

Our methods with this choice for losses are related to GANs-MMD approaches that previously were applied to problems in image generation [86,11,36]. Our work addresses generative models for $\mathfrak{n}^{th}$-order stochastic dynamics posing new challenges arising from the temporal aspects, stochasticity, high dimensional trajectory samples, and general order dynamics. Related work using MMD to learn SDE models was developed in [1]. However, in this work the focus was on scalar-valued first-order SDEs and learning primarily the deterministic drift terms by using a splitting of the dynamics and fitting with Gaussian Process Regression (GPR) [153]. The GPR was used as a smoother to develop estimators for the drift contributions. For their scalar covariance, they learned a proportionality constant by treating it as part of the hyperparameters for the prior distribution for the GPRs optimized by MLE [1].

Our SDYN-GANs approach provides alternatives to MLE and a broader class of methods for treating vector-valued $\mathfrak{n}^{th}$-order stochastic dynamics and allowing for learning simultaneously the contributions of both the drift and diffusion. We introduce structured classes of implicit generative models by building on m-step stochastic numerical integrators. For training models, we develop custom adjoint methods to obtain gradients consistent with the specific choices of the m-step stochastic discretizations. We develop custom backpropagation procedures allowing for use of second-order statistics in loss functions. We develop different protocols for training by using variants of statistical criteria of the stochastic dynamics based on the (i) full trajectory, (ii) conditional distributions, or (iii) marginal distributions. Given the intrinsic high dimensionality of trajectory data, we also develop training methods for reducing the dimension by using probabilistic dependencies and facilitating sub-sampling of trajectories over specified time-scales τ. Motivated by problems in microscale mechanics, we show how our SDYN-GANs methods can be used to learn generative models of vector-valued second-order inertial stochastic dynamics of physical systems and unknown force-laws. Our SDYN-GANs methods provide flexible sample-based approaches for training over general classes of generative models to learn structured representations of the $\mathfrak{n}^{th}$-order dynamics of stochastic systems.

We organize the paper as follows. In Section 2, we discuss adversarial learning approaches using different formulations of GANs and our approaches for training dynamic models for stochastic systems. In Section 3, we discuss our SDYN-GANs approaches for learning discrete m-step stochastic models. This includes discussion of practical training methods for learning models from observations over ensembles of the stochastic trajectories and methods to handle higher-order statistics. In Section 4, we present results for learning generative models for the stochastic dynamics of physical systems. We first discuss training SDYN-GANs for physical modeling of the mechanics in an inertial stochastic system to learn parameters of both the drift and diffusive contributions in Section 4.1 and 4.2. We then discuss how SDYN-GANs can be used to learn non-linear force laws from an ensemble of trajectory observations in Section 4.3. The introduced adversarial learning approaches of SDYN-GANs provide methods for obtaining stable generative models for dynamic tasks including long-time prediction and developing simulation methods for stochastic systems.

2. Adversarial learning

We develop sample-based training methods for learning implicit generative models $G(\mathbf{Z};\theta_G)$, where $\mathbf{Z}$ is a random variable for a noise source and θ_G are the learned parameters. In adversarial training, a discriminator/critic model $D(\mathbf{X};\theta_D)$ is learned along with G, where $\mathbf{X}$ is a sample and θ_D are the learned parameters. The discriminator/critic D aims to distinguish a collection of samples generated by G from samples of the training data. The generator aims to produce samples of such good quality that any discriminative model D would be unable to distinguish them from the training data, see Fig. 1. This can be viewed as a two-player non-cooperative game [108]. Depending on the choices made for the loss functions, and the model classes of the discriminative and generative models, different types of divergences and metrics can be obtained for comparing the empirical distributions of the samples [128,53,3].

2.1. Generative adversarial networks (GANs)

In Generative Adversarial Networks (GANs), G is a neural network representing an implicit generative model denoted by $G(\mathbf{Z};\theta_G)$. The θ_G are the parameters and the samples are generated from a random source $\mathbf{Z}$ as $\tilde{\mathbf{x}}^{[i]} = G(\mathbf{Z}^{[i]};\theta_G)$. The discriminator/critic

$D(\mathbf{X};\theta_D)$ serves to distinguish how well a set of samples $\{\tilde{\mathbf{x}}^{[i]}\}_{i=1}^{\tilde{N}}$ generated by G matches the samples $\{\mathbf{x}^{[i]}\}_{i=1}^{N}$ of the training data set. Let the empirical distributions be denoted for the generator by $\tilde{\mu}_G$ and for the training data by $\tilde{\mu}_D$. The choices of the form of D and for the loss functions ℓ determine how the comparisons are made between the empirical probability distributions of the two sets of samples, $\tilde{\mu}_G, \tilde{\mu}_D$. There are many ways to formulate GANs and related statistical estimators [128,53].

2.1.1. *Original formulation of GANs*

In the original formulation of GANs [54,53], the objective for D can be expressed as

$$\theta_D^* = \arg\min_{\theta_D} J_D(\theta_G,\theta_D), \text{ with } J_D = \mathbb{E}_{(x,y)\sim p_{\text{synth}}}\left[-\log p_{\text{D}}(y|x;\theta_D)\right], \tag{1}$$

$$p_{\text{synth}}(x,y) = \frac{1}{2}p_{\text{data}}(x,y) + \frac{1}{2}p_{\text{model}}(x,y;\theta_G).$$

The p_{synth} is the probability of a synthetic data distribution obtained by mixing the samples of the generator $\tilde{x} \sim p_{\text{model}} \sim \tilde{\mu}_G$ with the training data $x \sim p_{\text{data}} \sim \tilde{\mu}_D$, [53]. The $\tilde{\mu}_G$, $\tilde{\mu}_D$ denote the empirical measures of the samples. Labels are assigned with $y=-1$ for the generated data samples and $y=+1$ for the training data. For a given θ_D, we let $D(x) = p_{\text{D}}(y=+1|x;\theta^D)$ for the probability that the discriminator classifier assigns the sample x to have the label $y=+1$. We then have $1-D(x) = p_{\text{D}}(y=-1|x;\theta_D)$. With these choices and notation, the objective now can be expressed as

$$J_D = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[\log(D(x))\right] - \frac{1}{2}\mathbb{E}_{x\sim p_{\text{model}}(;\theta_G)}\left[\log(1-D(x))\right]. \tag{2}$$

The aim is for the generator to produce high quality samples so that even the optimal discriminator θ_D^* is confused and gives the value $D(x) = p_{\text{D}}(\cdot;\theta_D^*) = 1/2$.

The optimal generative model in this case has objective $\theta_G^* = \arg\min_{\theta_G} J_G(\theta_G,\theta_D^*(\theta_G))$. When $J_G = -J_D$, this can be viewed as a two-player zero-sum game with cost function $\ell = J_D$ with θ_D^* and θ_G^* corresponding to the optimal strategies. If the objectives are fully optimized to the equilibrium, this would give the Jensen-Shannon (JS) Divergence $JS(\tilde{\mu}_{G(;\theta_G)},\tilde{\mu}_D)$ [88]. For the empirical distributions of the generator $\tilde{\mu}_G$ and training data $\tilde{\mu}_D$, we have

$$JS(\tilde{\mu}_G,\tilde{\mu}_D) = \frac{1}{2}D_{KL}\left(\tilde{\mu}_D\|\frac{1}{2}\left(\tilde{\mu}_G+\tilde{\mu}_D\right)\right) + \frac{1}{2}D_{KL}\left(\tilde{\mu}_G\|\frac{1}{2}\left(\tilde{\mu}_G+\tilde{\mu}_D\right)\right), \tag{3}$$

with D_{KL} the Kullback-Leibler Divergence [80,88]. The optimal generator would be

$$\theta_G^* = \arg\min_{\theta_G} JS(\tilde{\mu}_{G(\cdot;\theta_G)},\tilde{\mu}_D). \tag{4}$$

In practice, to mitigate issues with gradient-learning, the zero-sum condition can be relaxed by using different loss functions in the objectives with $J_G \neq -J_D$. For example, $J_G = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{model}}(;\theta_G)}\left[\log(D(x))\right]$ has been used for the generator G to improve the gradient [53]. Using this formulation for J_G can result in non-zero sum games with more general solutions characterized as Nash Equilibria or other stationary states [104,108,102,118].

2.1.2. *Alternative formulations for GANs*

To help mitigate some of the issues that can arise with the initial formulation of GANs, we consider alternative statistical estimators based on Integral Probability Metrics (IPMs) [128,100]. In this formulation, we use discriminators/critics with the IPM objective based on

$$J_D(\theta_G,\theta_D) = \left|\mathbb{E}_{\tilde{\mu}_{G(;\theta_G)}}\left[F(X';\theta_D)\right] - \mathbb{E}_{\tilde{\mu}_D}\left[F(X;\theta_D)\right]\right| \tag{5}$$

and $J_G = -J_D$ with $\theta_G^* = \arg\min_{\theta_G} J_G(\theta_G,\theta_D^*(\theta_G))$ and $\theta_D^* = \arg\min_{\theta_D} J_D(\theta_G,\theta_D)$. The IPM can be expressed as

$$\begin{aligned}\theta_G^* &= \arg\min_{\theta_G}\max_{f\in\mathcal{F}}\left(\mathbb{E}_{\tilde{\mu}_D}\left[f(X)\right] - \mathbb{E}_{\tilde{\mu}_{G(;\theta_G)}}\left[f(X')\right]\right) \\ &= \arg\min_{\theta_G}\max_{f\in\mathcal{F}}\ell(\theta_G;f) = \arg\min_{\theta_G}\ell^*(\theta_G),\end{aligned} \tag{6}$$

where $\mathcal{F} = \{f \mid \exists\theta_D \text{ s.t. } f = F(\cdot;\theta_D)\}$ and $\ell^*(\theta_G) = \max_{f\in\mathcal{F}}\ell(\theta_G;f)$. Here, we treat X generally, but intuitively this can be thought of as a random variable representing either the entire dynamical trajectory sample, or for practical computations, as a fragment of a trajectory sample at discrete times $t_i \in [0,\tau]$ over a limited time duration τ. In the case of $\mathcal{F} = \text{Lip-1}$, consisting of Lipschitz functions with $L \leq 1$, this results in $\ell^*(\theta_G) = \max_{f\in\text{Lip-1}}\ell(\theta_G;f) = \mathcal{W}^1\left(\tilde{\mu}_G,\tilde{\mu}_D\right)$. This follows by Kantorovich-Rubinstein Duality, where $\mathcal{W}^1$ is the Wasserstein-1 distance for the empirical measures $\tilde{\mu}_G$ and $\tilde{\mu}_D$ [146]. While Wasserstein distance provides a widely used norm for analyzing measures, in practice for training, it can be computationally expensive to approximate and can result in challenging gradients for learning [95].

As an alternative to these approaches, we use for IPMs a smoother class of discriminators, with $\mathcal{F} \subset \mathcal{H}$ with $\mathcal{H}$ a Reproducing Kernel Hilbert Space (RKHS) [4,10]. By choice of the kernel $k(\cdot,\cdot)$ we also can influence the features of the distributions that are emphasized during comparisons. We take $\mathcal{F} = \{f \in \mathcal{H} \mid \|f\|_{\mathcal{H}} \leq 1\}$ for the objective,

$$\theta_G^* = \arg\min_{\theta_G}\max_{f\in\mathcal{F}}\ell(\theta_G;f), \text{where } \ell(\theta_G;f) = \mathbb{E}_{\tilde{\mu}_D}\left[f(X)\right] - \mathbb{E}_{\tilde{\mu}_{G(;\theta_G)}}\left[f(X')\right]. \tag{7}$$

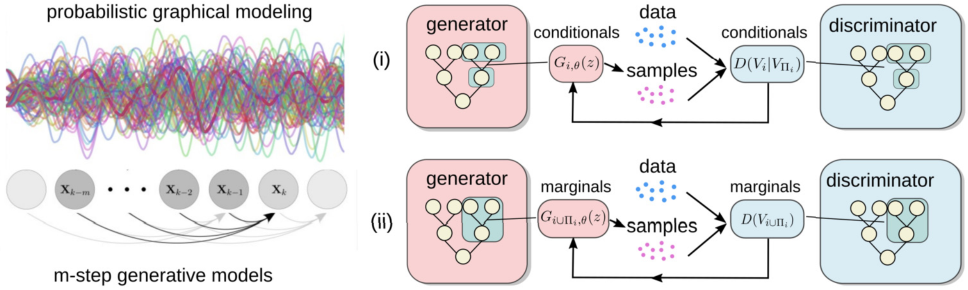

Fig. 2. Probabilistic dependencies in the stochastic m-step models. We utilize for a trajectory sample $\mathbf{X}_0,\ldots,\mathbf{X}_N$ the probabilistic dependencies in time to help facilitate learning models *(left)*. We develop different training protocols using (i) conditional statistics and (ii) marginal statistics. We also develop training methods for different trajectory durations $[0,\mathcal{T}]$ and sub-sampling over time-scales τ *(right)*.

The objective can be partially solved analytically to yield $\ell^*(\theta_G) = \max_{f\in\mathcal{F}} \ell(\theta_G; f) = \|\eta_{\tilde{\mu}_D} - \eta_{\tilde{\mu}_G}\|_{\mathcal{H}}$, where $\eta_{\tilde{\mu}_D} = E_{X\sim\tilde{\mu}_D}[k(\cdot,X)]$, $\eta_{\tilde{\mu}_G} = E_{X'\sim\tilde{\mu}_G}\left[k(\cdot,X')\right]$ and $\|\cdot\|_{\mathcal{H}}$ is the norm of the RKHS [127]. Here, we take $\tilde{\mu}_D$ to be the empirical distribution of the training data and $\tilde{\mu}_G$ the empirical distribution of the generated samples. Characterizing the differences between the probability distributions in this way corresponds to the statistical approaches of Generalized Method of Moments (GMM) [62] and Maximum Mean Discrepancy (MMD) [41,126].

2.2. SDYN-GANs: stochastic dynamic generative adversarial networks

Our objective is to learn generative models for the stochastic dynamics that are capable not only in reproducing observations but also of making predictions or performing simulations on longer time-scales. This presents challenges since the learned models need to have stable behaviors with respect to accumulated errors and grapple with high dimensional trajectory samples. To address these issues, we develop learning methods providing ways make use of properties of stochastic systems, including Markovian conditions, probabilistic dependencies, physical principles when available, and other attributes. While our approaches can be used more broadly, we focus here on learning generative models related to vector-valued Ito Stochastic Processes [105] which can be expressed as solutions of Stochastic Differential Equations (SDEs) [105,44].

In the case of Ito SDEs, the high dimensional probability distributions associated with trajectory observations can be factored. We make use of the probabilistic graphical modeling perspective to express statistical dependencies [147,12,132,101], see Fig. 2. We will use this approach to factorize the distributions to leverage the coupling of components over time to reduce the effective dimensionality of the inference problems and enhance the statistical power of the observation data. We remark that we also could use this approach to factorize further the probability distributions for the states of the system at a given time. For obtaining generative models G that are useful for prediction and long-time simulations, we also develop generative model classes based on stable m-step stochastic numerical integrators with learnable components [76]. For stochastic systems arising in physics, we can further refine the class of m-step models to incorporate physical principles, such as in mechanics using approximate time-reversibility for the conservative contributions to the dynamics, fluctuation-dissipation balance, or other properties [145,57,68,135,138,93].

We develop Stochastic Dynamic Generative Adversarial Networks (SDYN-GANs), for learning generative models G for sampling discretized trajectories, $\mathbf{X}^{[i]} = G(\mathbf{Z}^{[i]},\theta_G)$. The $\mathbf{Z}^{[i]}$ denotes the sample of the noise source mapped under G to generate the sample $\mathbf{X}^{[i]}$. The data samples can be expressed as $\mathbf{X}^{[i]} = \left(\mathbf{X}_1^{[i]},\ldots,\mathbf{X}_N^{[i]}\right)$ where $\mathbf{X}_j^{[i]} = \mathbf{X}^{[i]}(t_j)$ are the temporal components. We use a discretization first approach where our model class consists of the discretized representations of the stochastic system allowing during training for optimization using loss functions that incorporate the contributions of potential numerical artifacts when learning representations of the underlying training data. These methods would differ from alternative approaches using discretizations at later stages, such as first computing variational derivatives for gradients which are then approximated by discretization. Relative to such approaches, we can avoid potential issues with misalignment between the model loss and the gradients used in training that can arise from the error from discretization approximations. In our methods, we instead incorporate discretization artifacts of models more directly into the loss which are optimized as part of the learning process.

2.2.1. Generators G based on stochastic m-step methods

We learn generators G over a class of models corresponding to m-step stochastic numerical integrators of the form

$$\mathbf{X}_j = \mathbf{\Psi}_{j-1}(\mathbf{X}_{j-1},\ldots,\mathbf{X}_{j-m},\boldsymbol{\omega}_{j-1};\mathbf{p}) \tag{8}$$
$$\mathbf{X}_0 = \mathbf{x}_0(\mathbf{p}),\ldots,\mathbf{X}_{m-1} = \mathbf{x}_{m-1}(\mathbf{p}).$$

The trainable term $\mathbf{\Psi}_{j-1}$ approximates the evolution of the dynamics from time-step t_{j-1} to t_j using information at times $t_{j-1},\ldots,t_{j-m}$. Since the dynamics is stochastic, the numerical update is randomized and $\mathbf{\Psi}_j(\cdot,\boldsymbol{\omega};\mathbf{p})$ can be viewed as random variables on a probability space $(\Omega,\mathcal{B},\mu)$. Here, for the discretized system we use a collection of standard i.i.d. Gaussian samples $\boldsymbol{\omega} = \{\boldsymbol{\omega}_j\}_j \in \Omega$. The $\mathbf{p}$ gives the subset of parameters $\mathbf{p}\in\theta_G$ used by the m-step model for generating trajectories and training. We remark that we distinguish $\mathbf{p}$ with θ_G, since θ_G can include additional parameters not impacting trajectory generation which are related to the form of the loss function, observation noise, known terms, or of regularizations. The parameters $\mathbf{p}$ may include the time-step Δt, form of the update, and other properties of the underlying stochastic system. The $\mathbf{p}$ can also be used to parameterize the initial conditions $\mathbf{X}_k = \mathbf{x}_k(\mathbf{p})$ at times t_k with $k \in [0, m-1]$.

In terms of $G(\mathbf{Z}; \theta_G)$, we will consider here primarily the case with $\mathbf{Z} = \{\boldsymbol{\omega}_j\}_{j=1}^N$. In the general case, $\mathbf{Z}$ can also include additional sources of noise. The use of m-step stochastic methods for our generative modeling allows for leveraging probabilistic dependencies to factor high dimensional trajectory data. Our m-step methods can be viewed as a variant of structural equations for probabilistic graphical modeling [12,101,147,29], see Fig. 2.

We will develop our methods for classes of generative models G based on stochastic numerical integration methods [76,68,5]. Methods used widely for first-order stochastic dynamical systems include the $(m = 1)$-step Euler-Maruyuma Methods [96], Milstien Methods [99], and Runge-Kutta Methods [76]. For higher-order accuracy or for $(\mathfrak{n} > 1)$-order dynamics, m-step multistep methods can be developed. We emphasize that m-steps refers here to the integrator updates and not the number of steps n_t over the duration of sample trajectories. For approximating solutions of stochastic and ordinary differential equations, it is well-known the updates of numerical methods must be chosen with care to ensure convergence [76]. A common strategy to obtain convergence of numerical methods is to impose a combination of local accuracy in the approximation (consistency) along with conditions more globally on how errors can accumulate (stability) [67,16,76]. We develop our generative model classes to achieve related properties by allowing for learning models $\boldsymbol{\Psi}$ based on stable and accurate m-step stochastic integrators. This allows for using m-step multistep methods, such as stochastic variants of Adams-Bashforth and Adams-Moulten [76,15,39] or Velocity-Verlet [145]. For physical systems this also allows for developing methods that consider imposing additional properties such as approximate time-reversibility [145,57], fluctuation-dissipation balance [5,135], geometric and topological conditions [93,58], or using exponential time-step integration [68,5]. Our methods allow for learning over generative model classes which can leverage these considerations of stability, accuracy, physical principles, and other properties of stochastic systems and numerical approximations.

2.2.2. Loss function $\ell^*(\theta_G)$ for training

We use adversarial learning to train the generative model G by using discriminators/critics D to distinguish the samples $\{\tilde{\mathbf{x}}_i\}_{i=1}^{\tilde{N}}$ generated by G from those of the training data $\{\mathbf{x}_i\}_{i=1}^N$. We obtain generators by optimizing the loss $\theta_G^* = \arg\min_{\theta_G} \ell^*(\theta_G)$, with $\ell^*(\theta_G) = \ell(\theta_G, \theta_D^*(\theta_G))$. This can be formulated many different ways corresponding to different divergences and metrics of the empirical distributions. As we discussed in Section 2.1, we will use losses $\ell(\cdot)$ corresponding to Integral Probability Metrics (IPMs) of the form

$$\ell_{\text{IPM}}(\theta_G) = \ell_{\text{IPM}}(\tilde{\mu}_D, \tilde{\mu}_G) = \max_{f \in \mathcal{F}} \left(\mathbb{E}_{\tilde{\mu}_D}[f(\mathbf{X})] - \mathbb{E}_{\tilde{\mu}_G}\left[f(\mathbf{X}')\right] \right), \tag{9}$$

where $\tilde{\mu}_D$, $\tilde{\mu}_G$ denote the measures being compared. We take $\tilde{\mu}_D = \tilde{\mu}_D, \tilde{\mu}_G = \tilde{\mu}_G$ and use $\mathcal{F} = \{f \in \mathcal{H} \mid \|f\|_{\mathcal{H}} \leq 1\}$, where $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with kernel $k(\cdot,\cdot)$. With these choices, we can express the GANs and resulting IPM loss as $\ell_{\text{IPM}}(\theta_G) = \|\eta_{\tilde{\mu}_G} - \eta_{\tilde{\mu}_D}\|_{\mathcal{H}}$, where $\eta_{\tilde{\mu}_D} = \mathbb{E}_{\tilde{\mu}_D}[k(\cdot,\mathbf{X})]$ and $\eta_{\tilde{\mu}_G} = \mathbb{E}_{\tilde{\mu}_G}[k(\cdot,\mathbf{X})]$. This can be viewed as embedding each of the measures as functions in the RKHS $\mathcal{H}$ [120,127]. The characterization of their differences can be expressed as the $\mathcal{H}$-norm of their embedded representations. Since $k(\cdot,\mathbf{X})$ gives a collection of features, which under expectation generate generalized moments, this approach corresponds to Generalized Moments Methods (GMMs) [62] and Maximum Mean Discrepancy (MMD) [55,41,126].

In practice, to obtain more amenable losses for optimization and for computing gradients, we use a reformulation which squares the objective to obtain

$$\begin{aligned} \ell^*(\theta_G) &= \left(\ell_{\text{IPM}}(\theta_G)\right)^2 = \|\eta_{\tilde{\mu}_D} - \eta_{\tilde{\mu}_G}\|_{\mathcal{H}}^2 \\ &= \mathbb{E}_{\mathbf{X},\tilde{\mathbf{X}}\sim\tilde{\mu}_D}\left[k(\mathbf{X},\tilde{\mathbf{X}})\right] - 2\mathbb{E}_{\mathbf{X}\sim\tilde{\mu}_D,\mathbf{X}'\sim\tilde{\mu}_G}\left[k(\mathbf{X},\mathbf{X}')\right] + \mathbb{E}_{\mathbf{X}',\tilde{\mathbf{X}}'\sim\tilde{\mu}_G}\left[k(\mathbf{X}',\tilde{\mathbf{X}}')\right]. \end{aligned} \tag{10}$$

The $\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{X}', \tilde{\mathbf{X}}'$ are taken to be independent random variables for the specified distributions. This is related to energy statistics [134] and the sample-based estimator developed in [55]. To find approximate minimizers for training G, we use Stochastic Gradient Descent (SGD) methods

$$\theta_G^{n+1} = \theta_G^n - \eta_n \hat{q} \;\text{ with }\; \hat{q} \approx \nabla_{\theta_G} \ell^*(\theta_G^n). \tag{11}$$

The η_n denotes the learning rate and the gradient estimators $\hat{q}$ are based on mini-batches [117,13,12]. We use variants incorporating momentum, such as ADAM and others [75,137,94]. To compute $\hat{q}$, we use the unbiased estimator of [55] as part of the gradient approximation. We estimate gradients for the empirical loss using $\hat{q} = \nabla_{\theta_G} \tilde{\ell}^*(\theta_G)$ where

$$\tilde{\ell}^*(\theta_G) = \frac{1}{N(N-1)} \sum_{i \neq j}^{N} k\left(\mathbf{X}^{[i]}, \mathbf{X}^{[j]}\right) - \frac{2}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} k\left(\mathbf{X}^{[i]}, \mathbf{Y}^{[j]}\right) + \frac{1}{M(M-1)} \sum_{i \neq j}^{M} k\left(\mathbf{Y}^{[i]}, \mathbf{Y}^{[j]}\right). \tag{12}$$

The samples of mini-batches have distributions $\mathbf{X}^{[i]} \sim \tilde{\mu}_G$ and $\mathbf{Y}^{[i]} \sim \tilde{\mu}_D$.

The trajectory samples $\mathbf{X}^{[i]} = G(\mathbf{Z}^{[i]}; \theta_G)$ depend on θ_G, which further requires methods for computing the contributions of the gradients $\nabla_{\theta_G}\mathbf{X}^{[i]}$. This can pose challenges since these gradients involve the $m-$step stochastic integrators when $\mathbf{p} \subset \theta_G$. This can become particularly expensive if using direct backpropagation based on book-keeping for forward evaluations of the stochastic integrators and the second-order statistics of $\tilde{\ell}^*$. This would result in prohibitively large computational call graphs. This arises from the high dimension of the trajectories $\mathbf{X}$ and the multiple dependent time-steps during generation. This is further compounded by the mini-batch sampling $\{\mathbf{X}^{[i]}\}_{i=1}^{n_b}$ and second-order statistics. We address these challenges by developing adapted adjoint methods and custom backpropagation approaches specialized to utilize the structure of the loss functions. This allows us to compute efficiently the contributions of $\nabla_{\theta_G}\mathbf{X}^{[i]}$ to $\nabla_{\theta_G}\tilde{\ell}^*$ to obtain the gradient estimates needed for training. We discuss more details of how we address these issues in Sections 3.1 and 3.2.

2.2.3. Training protocols using conditional and marginal distributions

We also introduce a few different training protocols for SDYN-GANs. These each treat the trajectory data in different ways to compare the underlying probability distributions using the samples. This includes comparisons using (i) distributions over the entire trajectory, (ii) marginal distributions over trajectory fragments, and (iii) conditional distributions over trajectory fragments, see Fig. 2. In more detail, our training approaches for $\mathfrak{n}^{th}$-order stochastic systems are

Full-Trajectory Method (full-traj): Samples full trajectory $\mathbf{R} = (\mathbf{X}(t_0), \mathbf{X}(t_0+\delta t), \ldots, \mathbf{X}(t_0+(m-1)\delta t), \mathbf{X}(t_1), \mathbf{X}(t_1+\tau), \ldots, \mathbf{X}(t_1 + (N_{\mathcal{T}}-1)\tau))$ corresponding to distribution $p_\theta(\mathbf{R})$ and trajectory with $N_{\mathcal{T}}$ steps.

Marginals Method (marginals): Samples marginals over trajectory fragments $\mathbf{R}_k$ of the form $\mathbf{R}_k = (\mathbf{X}(t_k), \mathbf{X}(t_k+\delta t), \ldots, \mathbf{X}(t_k + (m-1)\delta t), \mathbf{X}(s_k), \mathbf{X}(s_k+\tau), \ldots, \mathbf{X}(s_k + (\ell_\tau - 1)\tau))$, $k \in [1, N_m]$, with marginal distributions $\{p_{\theta,k}(\mathbf{R}_k)\}_{k=1}^{N_m}$.

Conditional Method (conditionals): Samples conditionals $\mathbf{R}_k|\mathbf{Q}_k$ over trajectory fragments $\mathbf{R}_k$ conditioned on fragments $\mathbf{Q}_k$ of the form $\mathbf{R}_k = (\mathbf{X}(s_k), \mathbf{X}(s_k+\tau), \ldots, \mathbf{X}(s_k + (\ell_\tau - 1)\tau))$ and $\mathbf{Q}_k = (\mathbf{X}(t_k), \mathbf{X}(t_k+\delta t), \ldots, \mathbf{X}(t_k+(m-1)\delta t)$, $k \in [1, N_m]$, with conditional distributions $\{p_{\theta,k}(\mathbf{R}_k|\mathbf{Q}_k)\}_{k=1}^{N_m}$.

The marginal and conditional training approaches make use of fragments of the trajectories in different ways. In the conditional case when sampling, the initial starting fragments $\mathbf{Q}_k$ of the trajectories always agree between the training and generated samples. Only the response part $\mathbf{R}_k$ of the trajectories can differ during the empirical sampling of the conditional distributions. In the marginal case there is more flexibility and both the initial and response parts of the trajectories can differ in the training and generated samples. The training in this case is based on empirical sampling of trajectories from the marginal distributions.

In each of these approaches we have two adjustable time-scales δt and τ. The τ is associated with s_k and δt is associated with t_k. In the notation t_k gives times for the initial state sampling and s_k for subsequent sample times. This decoupling allows in our methods for flexibility and controlling independently the accuracy and stability of the stochastic numerical methods used to generate trajectories and the time-scales on which we probe the system dynamics when learning model features. This also allows for s_k to be adjusted to help ensure accurate estimation of the contributions of the initial velocity or other conditions needed by the m-step multistep methods for generating trajectories of the $\mathfrak{n}^{th}$-order dynamical system. The τ is the time-scale on which the dynamic evolution is sampled. The δt corresponds to the time-scale for sampling the initial dynamical state information. In general we allow for $\tau \neq \delta t$ and for $m \neq \mathfrak{n}$. For $\tau > \delta t$, this would correspond to using trajectories down-sampled in time useful when making comparisons. We also mention the methods allow for a distinction during training between the number of numerical time-steps n_t used to generate the trajectories and the number of time-slices N_τ on time-scale τ used in the loss function. This allows for further control over the time-scales probed and efficiency trade-offs in the computational methods. In the conditional and marginal cases, the $\ell_\tau = (t_N - s_k)/\tau + 1$.

The methods allow for $m \neq \mathfrak{n}$, where $\mathfrak{n}$ is referring to $\mathfrak{n}^{th}$-order dynamical systems and m the number of steps in the multistep stochastic numerical methods. Choices with $m > 1$ can arise and be useful even for $\mathfrak{n} = 1$ first-order dynamical systems to obtain enhanced accuracy, stability, efficiency, or from other considerations [76]. Examples include $m > 1$ stochastic multistep numerical method, such as Adams-Bashforth [76,15,39]. This provides flexibility for applying our methods in different settings and application domains. In practice, it is natural to use choices with $\Delta t = \alpha\delta t$ and $\tau = a\Delta t$ for $\alpha, a \in \mathbb{N}$ to obtain aligned temporal sampling between the stochastic trajectories of the generative models and the observation data. While our methods could be used more generally with interpolation in time, we will take as our default choice $t_k = k\Delta t$ and $s_k = t_k + \tau$.

We also remark that our sample-based methods result in methods that need only save the states of the individual trajectories for our forward-pass to compute losses and for our backward-pass to compute gradients. Concerning potential applications to high dimensional systems, our methods have a scaling proportional to the number of time-steps and the number of samples saved. This results in a linear scaling in the number of dimensions. The different protocols and adjustable parameters allow for making further trade-offs in the time-scales and duration of the generated trajectories used in the forward and backward computational steps. We discuss how practical methods can be developed for computing gradients and learning with these approaches in Section 3. We show results for SDYN-GANs using these different training protocols (i)–(iii) for learning physical model parameters and non-linear force-laws in Section 4.

3. Methods for learning generative models with SDYN-GANs

Learning models in SDYN-GANs requires being able to compute the gradients of the sample-based loss functions. This requires methods for computing during training the gradients of the second-order statistics for the loss in equation (10) with high dimensional trajectory samples $\mathbf{X}, \mathbf{X}'$. We develop practical computational methods leveraging the structure of the loss $\tilde{\ell}^*$ in equation (12), the probabilistic dependencies, and by developing specialized adjoint methods. We remark that while direct backpropagation through the generated time-steps of trajectories is sometimes used, this can lead to large computational call graphs and other numerical issues resulting in expensive computational methods in calculations of the gradients needed for training. As we shall discuss, our methods avoid this by mathematically deriving a set of adjoint equations for gradients of our class of generative models, which can be reduced to solving a set of recurrence equations. These approaches provide practical methods for computing the gradients needed for training SDYN-GANs.

3.1. Gradient methods for *m*-step generative models

The training of the m-step generative models requires methods for computing the gradients of the loss $\nabla_{\theta_G}\tilde{\ell}^*$ of equation (12). This poses challenges given the contributions of $\nabla_{\theta_G}\mathbf{X}^{[i]}(;\theta_G)$ where $\theta_G = \mathbf{p}$, as discussed in Section 2.2. Training requires computing contributions of the gradients of the discretized sample paths $\{\mathbf{X}(t;\mathbf{p})\}_{t\in[0,\mathcal{T}]}$. For this purpose, we will utilize the Implicit Function Theorem to develop a set of adjoint methods for our m-step generative models [69,17,131,125].

Consider the implicit formulation in terms of the vector-valued function $\mathbf{f} = \mathbf{f}(\mathbf{X}_0, \ldots, \mathbf{X}_N; \mathbf{p}) = 0$ with components

$$[\mathbf{f}]^{[i]}_{(j,d)} = \left[\mathbf{X}^{[i]}_j - \mathbf{\Psi}_{j-1}(\mathbf{X}^{[i]}_{j-1}, \ldots, \mathbf{X}^{[i]}_{j-m}, \boldsymbol{\omega}_{j-1}; \mathbf{p})\right]^{(d)} = 0, \tag{13}$$

where $\mathbf{X}^{[i]}_k = \mathbf{X}^{[i]}(t_k)$ is the i^{th} sample. The superscript index d denotes the dimension component for $\mathbf{X}_k \in \mathbb{R}^n$. To simplify the notation let $\mathbf{x} = \{\mathbf{x}^{[i]}\}_{i=1}^M = \{(\mathbf{X}_0, \ldots, \mathbf{X}_{N_\mathcal{T}})^{[i]}\}_{i=1}^M$, where there are M samples indexed by i. For each realization of $\boldsymbol{\omega}^{[i]}$, the trajectory samples are determined implicitly by $\mathbf{f}^{[i]}(\mathbf{x}; p) = 0$.

Now consider a function $g(\mathbf{x}, p)$ which depends on $\mathbf{x}$ and where p denotes the subset of parameters on which we seek to compute gradients for training. For example, we can set g to be the loss function $g(\mathbf{x}, p) = \tilde{\ell}^*$ with $p = \theta_G$ or set g to other quantities of interest of the stochastic trajectories. The total derivative of g is then given by the chain-rule

$$[\nabla_p g]_j = \frac{dg}{dp_j} = \frac{\partial g}{\partial p_j} + \sum_k \frac{\partial g}{\partial x_k}\frac{\partial x_k}{\partial p_j} = [\mathbf{g}_p + \mathbf{g}_x \mathbf{x}_p]_j. \tag{14}$$

We collect derivatives into vectors $\mathbf{g}_x \in \mathbb{R}^{1\times n_x}, \mathbf{g}_p \in \mathbb{R}^{1\times n_p}, \mathbf{x}_p \in \mathbb{R}^{n_x \times n_p}$, where $\mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{p} \in \mathbb{R}^{n_p}$. The derivatives of the loss g in $\mathbf{x}$ and $\mathbf{p}$ are typically easier to compute relative to the derivatives in $\mathbf{p}$ for $\mathbf{x}_p = \nabla_p \mathbf{x}(p)$.

We can obtain expressions for the derivatives $\mathbf{x_p}$ using the Implicit Function Theorem for $\mathbf{f}^{[i]}(\mathbf{x}, \mathbf{p}) = 0$ with $\mathbf{f}^{[i]} \in \mathbb{R}^{n_f^{[i]}}$ and under the assumption $\mathbf{f}^{[i]}_{\mathbf{x}}$ is invertible [69,125]. This gives

$$\frac{df}{dp} = \mathbf{f}_x \mathbf{x}_p + \mathbf{f}_p = 0 \;\Rightarrow\; \mathbf{x}_p = -\mathbf{f}_x^{-1}\mathbf{f}_p = -J^{-1}\mathbf{f}_p. \tag{15}$$

We assume throughout $n_f = n_x$, but distinguish these in the notation to make more clear the roles of the conditions. The Jacobian in x is given by $J = \mathbf{f}_x = \nabla_x \mathbf{f} \in \mathbb{R}^{n_f \times n_x}$ for the map $\mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{p})$ defined by fixed p for mapping $\mathbf{x} \in \mathbb{R}^{n_x}$ to $\mathbf{y} \in \mathbb{R}^{n_f}$. The $\mathbf{x}_p = \nabla_p \mathbf{x}(p) \in \mathbb{R}^{n_x \times n_p}$ and $\mathbf{f}_p \in \mathbb{R}^{n_f \times n_p}$. We can substitute this above to obtain

$$\nabla_p g = \mathbf{g}_p + \mathbf{g}_x(-\mathbf{f}_x^{-1}\mathbf{f}_p) = \mathbf{g}_p + \left(-\mathbf{f}_x^{-T}\mathbf{g}_x^T\right)^T \mathbf{f}_p = \mathbf{g}_p - \left(J^{-T}\mathbf{g}_x^T\right)^T \mathbf{f}_p = \mathbf{g}_p - \mathbf{r}^T \mathbf{f}_p. \tag{16}$$

We let $\mathbf{r} = J^{-T}\mathbf{g}_x^T \in \mathbb{R}^{n_f \times 1}$. To compute the gradient $\nabla_p g$ we proceed in the following steps: (i) compute the derivatives $\mathbf{g}_p, \mathbf{g}_x, \mathbf{f}_p$, (ii) solve the linear system $J^T\mathbf{r} = \mathbf{g}_x^T$, (iii) evaluate the gradient using $\nabla_p g = \mathbf{g}_p - \mathbf{r}^T\mathbf{f}_p$.

This provides methods for computing the gradients of g using the solutions $\mathbf{r} \in \mathbb{R}^{n_f \times 1}$, which in general for the trajectory samples will be more efficient than solving directly for $\mathbf{x}_p \in \mathbb{R}^{n_x \times n_p}$. We emphasize that the construction of a dense $n_x \times n_x$ Jacobian and its inversion would be prohibitive, so instead we will solve the linear equations by approaches that use sparsity and only require computing the action of the adjoint J^T. We show how this can be done for our m-step stochastic methods below and give more technical details in Appendix A.

Our adjoint approaches are used to compute the gradient of the losses based on expectations of the following form and their gradients

$$g^*(\mathbf{X}(\cdot), p) = \mathbb{E}_\omega[\phi(\mathbf{X}(\omega; p); p)] = \int \phi(\mathbf{X}(\omega; p); p)\, d\mu_\omega, \quad \frac{dg^*}{dp} = \mathbb{E}[\nabla_p \phi] = \mathbb{E}[\nabla_p \tilde{g}]. \tag{17}$$

For generality and to further make clear the parameter dependencies, we will perform our derivations using the notation above for general expectations for the function $\phi = \phi(x; p)$. The ϕ can be any random variable that depends on the stochastic trajectories. As we shall discuss in more detail, ϕ will be related to our MMD-loss by setting it to the probing function $f(X')$ in equation (7). We emphasize that g^* has all of its parameter dependence explicitly on the random variable $\mathbf{X}(\omega, p)$ through the random variable $\tilde{g} = \phi$. This corresponds to the reference measure μ_ω having no p-dependence. In practice, the expectations will be approximated statistically by sample averages $\bar{g} = \frac{1}{M}\sum_{i=1}^M \tilde{g}^{[i]}$, where $\tilde{g}^{[i]}$ denotes the i^{th} sample of $\tilde{g}$. In this case our adjoint approaches compute the gradients using

$$\nabla_p \bar{g}(\mathbf{x}, \mathbf{p}) = \frac{1}{M}\sum_{i=1}^M \nabla_p \tilde{g}^{[i]}, \;\; \nabla_p \tilde{g}^{[i]} = \tilde{\mathbf{g}}_p^{[i]} - \mathbf{r}^{[i],T}\mathbf{f}_p^{[i]}, \;\; \mathbf{r}^{[i]} = J^{[i],-T}\tilde{\mathbf{g}}_x^{[i],T}, \tag{18}$$

where $J^{[i],T} = \mathbf{f}_x^{[i],T}$. We provide efficient direct methods for solving these linear systems for $\mathbf{r}^{[i]}$ for the m-step stochastic methods in Appendix A.

3.2. Gradient methods for the second-order statistics in the loss ℓ^*

The loss function $\tilde{\ell}^*$ in equation (12) involves computing second-order statistics of the samples. This is used to compare the samples $\mathbf{X}^{[i]} \sim \tilde{\mu}_G$ of the candidate generative model with samples $\mathbf{Y}^{[i]} \sim \tilde{\mu}_D$ of the training data. In computing gradients, this poses challenges in using directly backpropagation on the forward computations, which can result in prohibitively large computational call graphs.

We develop alternative methods utilizing special structure of the loss function $\tilde{\ell}^*$ to obtain more efficient methods for estimating the needed gradients for training. From equation (12), we see $\tilde{\ell}^*$ involves kernel calculations involving terms of the form $k(\mathbf{W}_1^{[i]}, \mathbf{W}_2^{[j]}; p)$, where $\mathbf{W}_1, \mathbf{W}_2 \in \{\mathbf{X}, \mathbf{Y}\}$. We remark the notation allows for expressing succinctly the three cases for sample comparisons that arise, and for decoupling the inputs for differentiation. We utilize this to obtain the gradients $\nabla_p \tilde{\ell}^*$. This requires us to compute the contributions of three terms, each of which have the general form

$$\frac{d}{dp} k(\mathbf{W}_1^{[i]}, \mathbf{W}_2^{[j]}; p) = \partial_1 k \cdot \mathbf{W}_{1,p}^{[i]} + \partial_2 k \cdot \mathbf{W}_{2,p}^{[i]} + \partial_3 k. \tag{19}$$

The $\partial_q k$ denote taking the partials in the q^{th} input to $k(\cdot, \cdot; p)$. The most challenging terms to compute are the partials in the trajectory samples $\mathbf{W}_{a,p} = \partial \mathbf{W}_a / \partial p$ when $\mathbf{W}_a = \mathbf{X}$. In this case, we have $\mathbf{W}_{a,p} = \mathbf{X}_p$. For $\mathbf{Y}$, we have $\mathbf{Y}_p = 0$, since the training data does not depend on p. We use that the gradients ultimately contract against the vector-valued terms $\partial_1 k$, $\partial_2 k$ in equation (19).

As a consequence, we can leverage our adjoint methods by computing the terms using $h(\mathbf{X}^{[i]}) = \mathbf{c} \cdot \mathbf{X}^{[i]}$. We treat $\mathbf{c}$ as a fixed constant to obtain $\nabla_p h(\mathbf{X}^{[i]}) = \mathbf{c} \cdot \mathbf{X}_p^{[i]}$. After differentiation, we use for the numerical values $\mathbf{c} = \partial_1 k$ or $\mathbf{c} = \partial_2 k$ for the final evaluation. This is done since there are efficiencies in computing the derivatives of the scalar quantity h, which avoids the need to construct and store the full gradient which would later be contracted anyway. We can then compute efficiently the derivatives $\nabla_p h(\mathbf{X}^{[i]})$ by leveraging our adjoint methods in Section 3.1. To compute the third term in equation (19) involving partial $\partial_3 k = \partial k / \partial p$, we use backpropagation while by holding the other inputs fixed.

To compute the contributions to the gradient $\tilde{\ell}^*$, we consider the first term $k(\mathbf{X}^{[i]}, \mathbf{X}^{[j]}; p)$ in equation (12) and treat the sums over the samples using

$$\frac{d}{dp} \sum_{i \neq j} k(\mathbf{X}^{[i]}, \mathbf{X}^{[j]}; p) = \sum_{i_0} \mathbf{c}_1(i_0)^T \mathbf{X}_p^{[i_0]} + \sum_{j_0} \mathbf{c}_2(j_0)^T \mathbf{X}_p^{[j_0]} + \mathbf{c}_3, \tag{20}$$

where

$$\mathbf{c}_1(i_0) = \sum_{j \neq i_0} \partial_1 k(\mathbf{X}^{[i_0]}, \mathbf{X}^{[j]}; p), \quad \mathbf{c}_2(j_0) = \sum_{i \neq j_0} \partial_2 k(\mathbf{X}^{[i]}, \mathbf{X}^{[j_0]}; p), \quad \mathbf{c}_3 = \sum_{i \neq j} \partial_3 k(\mathbf{X}^{[i]}, \mathbf{X}^{[j]}; p). \tag{21}$$

To compute the contributions of the terms $\mathbf{X}_p^{[i_0]} = \frac{d}{dp} \mathbf{X}^{[i_0]}$ and $\mathbf{X}_p^{[j_0]} = \frac{d}{dp} \mathbf{X}^{[j_0]}$, we use the adjoint methods discussed in Section 3.1 and Appendix A. For the second term contributing to $\tilde{\ell}^*$ in equation (12), we use

$$\frac{d}{dp} \sum_{i,j} k(\mathbf{X}^{[i]}, \mathbf{Y}^{[j]}; p) = \sum_{i_0} \tilde{\mathbf{c}}_1(i_0)^T \mathbf{X}_p^{[i_0]} + \tilde{\mathbf{c}}_3, \tag{22}$$

$$\tilde{\mathbf{c}}_1(i_0) = \sum_{j} \partial_1 k(\mathbf{X}^{[i_0]}, \mathbf{Y}^{[j]}; p), \;\; \tilde{\mathbf{c}}_3 = \sum_{i,j} \partial_3 k(\mathbf{X}^{[i]}, \mathbf{X}^{[j]}; p). \tag{23}$$

For the third term in equation (12), the only dependence on p is through the direct contributions to the kernel, since $\mathbf{Y}^{[i]}$ does not depend on p. We compute the gradient using

$$\frac{d}{dp} \sum_{i \neq j} k(\mathbf{Y}^{[i]}, \mathbf{Y}^{[j]}; p) = \hat{\mathbf{c}}_3, \;\; \hat{\mathbf{c}}_3 = \sum_{i \neq j} \partial_3 k(\mathbf{Y}^{[i]}, \mathbf{Y}^{[j]}; p). \tag{24}$$

For each term, the $\partial_j k = \partial_j k(w_1, w_2; p)$ can be computed using evaluations of k and local backpropagation methods. We use this to compute $\mathbf{c}_j, \tilde{\mathbf{c}}_j, \hat{\mathbf{c}}_j$. While the full gradient sums over all the samples, in practice we use stochastic gradient descent. This allows for using smaller mini-batches of samples to further speed up calculations. For SDYN-GANs, we use these approaches to implement custom backpropagation methods for gradient-based training of our m-step generative models. We also provide further details in Appendix A.

4. Results

We present results for SDYN-GANs in learning generative models for second-order stochastic dynamics. This is motivated by the physics of microscale mechanical systems and their inertial stochastic dynamics. We develop methods for learning force-laws, damping coefficients, and noise-related parameters. We present results for learning physical models for a micro-mechanical system corresponding to an Inertial Ornstein-Uhlenbeck (I-OU) Process in Section 4.2. We then consider learning non-linear force-laws for a particle system having inertial Langevin Dynamics in Section 4.3. We investigate in these studies the role of the time-scale τ for the trajectory sampling and the different training protocols based on sampling from the distributions of the (i) full-trajectory, (ii) conditionals, and (iii) marginals. The results show a few strategies for training SDYN-GANs to obtain generative models for stochastic systems.

Table 1
Parameters. The default values used in training.

Parameter	Value	Parameter	Value	Parameter	Value
Δt	10^{-3}	δt	10^{-3}	t_N	1.8×10^{-2}
n_t	18	τ	10^{-3}	N_τ	20
ℓ_τ	3	m	2	n	2

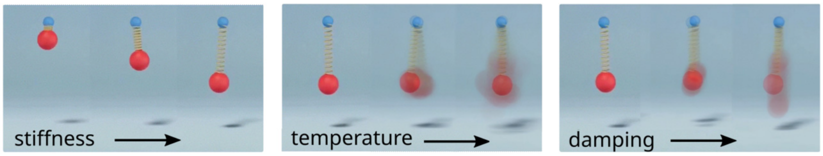

Fig. 3. Bead-spring mechanics: inertial Ornstein-Uhlenbeck process. We consider the stochastic dynamics of the three dimensional I-OU process under different physical conditions including when (i) stiffness decreases *(left)*, (ii) temperature increases *(middle)*, or (iii) damping decreases *(right)*.

4.1. *Discretization of inertial second-order stochastic dynamics*

We consider throughout second-order inertial stochastic dynamics of the form

$$m d\mathbf{V}(t) = -\gamma \mathbf{V}(t)dt + \mathbf{F}(\mathbf{X}(t))dt + \sigma d\mathbf{W}(t), \quad d\mathbf{X}(t) = \mathbf{V}(t)dt. \tag{25}$$

For SDYN-GANs, our generative models use the following $(m = 2)$-step stochastic numerical integrator [57]. The integrator can be expressed as a Velocity-Verlet-like scheme [145] of the form

$$\begin{aligned} \mathbf{X}_{n+1} &= \mathbf{X}_n + b\Delta t \mathbf{V}_n + \frac{b\Delta t^2}{2m}\mathbf{F}_n + \frac{b\Delta t}{2m}\boldsymbol{\eta}_{n+1}, \qquad a = \left(1 - \frac{\gamma\Delta t}{2m}\right)\left(1 + \frac{\gamma\Delta t}{2m}\right)^{-1}, \\ \mathbf{V}_{n+1} &= a\mathbf{V}_n + \frac{\Delta t}{2m}\left(a\mathbf{F}_n + \mathbf{F}_{n+1}\right) + \frac{b}{m}\boldsymbol{\eta}_{n+1}, \qquad b = \left(1 + \frac{\gamma\Delta t}{2m}\right)^{-1}. \end{aligned} \tag{26}$$

We discretize with $\mathbf{X}_n = \mathbf{X}(t_n)$, $\mathbf{V}_n = \mathbf{V}(t_n)$, $\mathbf{F}_n = \mathbf{F}(\mathbf{X}(t_n))$. We refer to this discretization of the stochastic dynamics as the Farago Method, which has been shown to have good stability and statistical properties in [57]. The thermal forcing is approximated by Gaussian noise $\boldsymbol{\eta}_n$ with mean zero and variance $\langle \boldsymbol{\eta}_n \boldsymbol{\eta}_\ell^T \rangle = \sigma^2 I \delta_{n,\ell} \Delta t$ for time-step Δt. This can be expressed as the 2-stage multi-step method as $\mathbf{X}_{n+1} = \boldsymbol{\Psi}_n(\mathbf{X}_n, \mathbf{X}_{n-1}, \boldsymbol{\omega}_n; \mathbf{p})$ with $\boldsymbol{\Psi}_n = 2b\mathbf{X}_n - a\mathbf{X}_{n-1} + \frac{b\Delta t^2}{m}\mathbf{F}_n + \frac{b\Delta t}{2m}\left(\boldsymbol{\eta}_{n+1} + \boldsymbol{\eta}_n\right)$. The constants also satisfy the identity $1 + a = 2b$. For our generative models, we use throughout this stochastic numerical discretization for approximating the inertial second-order stochastic dynamics of equation (25).

4.2. *Learning physical models by generative modeling: inertial Ornstein-Uhlenbeck processes*

Consider the Ornstein-Uhlenbeck (I-OU) process [142] which have the stochastic dynamics

$$m d\mathbf{V}(t) = -\gamma \mathbf{V}(t)dt - K_0 \mathbf{X}dt + \mathbf{F}_0 dt + \sigma d\mathbf{W}(t), \quad d\mathbf{X}(t) = \mathbf{V}(t)dt. \tag{27}$$

The $\mathbf{X}, \mathbf{V} \in \mathbb{R}^n$. Related dynamics arise when studying molecular systems [142,113,42,160], microscopic devices [52,51,34], problems in finance and economics [129,143,122,2], and many other applications [83,152,115,50].

From the perspective of mechanics, these dynamics can be thought of as modeling a microscopic bead-spring system within a viscous solvent fluid. In this case, $\mathbf{X}$ is the position and $\mathbf{V}$ is the velocity with $\mathbf{X}, \mathbf{V} \in \mathbb{R}^3$. The m is the bead's mass, γ is the level of viscous damping, $\sigma = \sqrt{2k_B T \gamma}$ gives the strength of the fluctuations for temperature T, with k_B denoting the Boltzmann factor [113]. The conservative forces include a harmonic spring with stiffness K_0 and a constant force $\mathbf{F}_0 = -mg\mathbf{e}_z$, such as gravity with coefficient g and $\mathbf{e}_z = (0, 0, 1)$. This gives the potential energy $U(\mathbf{X}) = \frac{1}{2}K_0\mathbf{X}^2 - \mathbf{F}_0 \cdot \mathbf{X}$ yielding the conservative forces as $\mathbf{F}(\mathbf{X}) = -\nabla_{\mathbf{X}} U$. We use for our generative models the dynamics of equation (27) approximated by the $(m = 2)$-step stochastic numerical discretization discussed in Section 4.1. We use the default values in training given in Table 1. We denote by θ the collection of physical parameters to be adjusted as $\theta = (K_0, \gamma, k_B T)$. See also Fig. 3.

The data-driven task is to learn a generative stochastic model $G(\cdot; \theta_G)$ from observations of trajectories of the system configurations $\mathbf{X}_{[0,\mathcal{T}]} = \{\mathbf{X}(t)\}_{t \in [0,\mathcal{T}]}$, so that $G(\mathbf{Z}; \theta_G) \sim \mathbf{X}_{[0,\mathcal{T}]}$. The $\mathbf{Z}$ is the noise source, here taken to be a high dimensional vector-valued Gaussian with i.i.d. components. The $\mathbf{V}(t)$ is not observed directly. To help illustrate more directly the behaviors of the methods, we have not included here the observation noise. This also could be included in the generative model with additional parameters optimized using the adversarial training. We show samples of the training data in Fig. 4. Training uses discretized trajectories and trajectory fragments of the form $(\mathbf{X}(t_1), \ldots, \mathbf{X}(t_k))$. Without using further structure of the data, this task can become challenging given the high dimensional distribution of trajectories $\mathbf{X}(t)$.

The SDYN-GANs uses the probabilistic dependence within the trajectories and the adversarial learning approaches discussed in Section 2.2. This provides flexible methods for learning general models for a wide variety of tasks, even when the generative

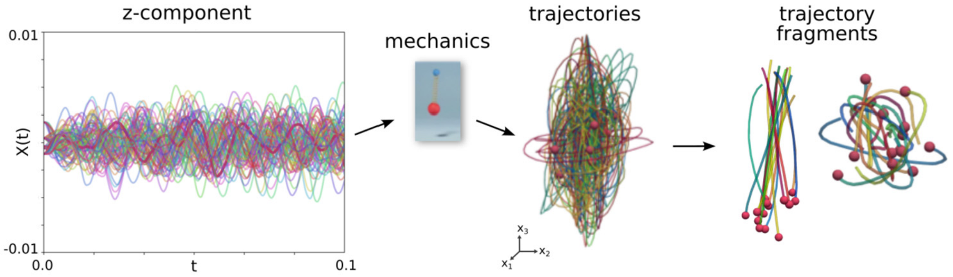

Fig. 4. Trajectory training data: inertial Ornstein-Uhlenbeck process. We show samples of the trajectory data used for training models. The z-component of the trajectory data is shown on the *(left)*. The bead-spring system and traces of the trajectory in three-dimensional space is shown in the *(middle)*. The fragments of the trajectory used during the different training protocols discussed in Section 2.2, on the *(right)*.

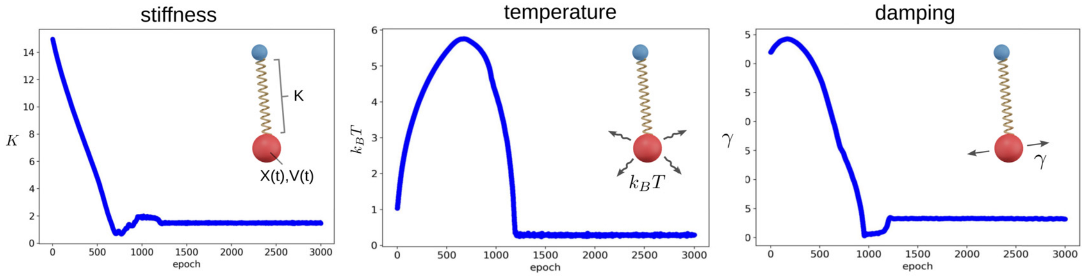

Fig. 5. Inertial Ornstein-Uhlenbeck process: learning the stochastic dynamics. We use SDYN-GANs to learn generative models for the observed trajectories of the stochastic dynamics. We learn simultaneously the drift and diffusive contributions to the dynamics. The target values for this trial were $K_0 = 1.5$, $\gamma = 3.2$ and $k_BT = 0.1$. We see early in training the stochastic parameters become large to minimize the loss by enhancing the overlap of the samples of the candidate generative model with the observed samples. As the stiffness and damping parameters become more accurate during training, the stochastic parameters decrease toward the target values.

models may not have readily expressible probability distributions. By using MMD within our SDYN-GANs methods allows for further inductive biases to be introduced based on the type of kernel employed. This can be used to emphasize key features of the empirical distributions being compared. For the OU process studies, we use a RKHS with Rational-Quadratic Kernel (RQK) [112,35] given by $k(x,y) = \left(1 + \frac{\|x-y\|^2}{2\alpha\ell^2}\right)^{-\alpha}$. The α determines the rate of decay and ℓ provides a characteristic scale for comparisons. The RQK is a characteristic kernel allowing for general comparison of probability distributions [127]. It has the advantage for gradient-based learning of not having an exponentially decaying tail yielding better behaviors in high dimensions [112,35]. For the training data, we generate continuously throughout learning new sample trajectories from the SDEs of the underlying target system when data is requested. For the conditional training case, empirical training samples and generated samples are compared that have the same initial trajectory fragment $\mathbf{Q}_k$. For the conditional part, 20 fragments $\mathbf{R}_k$ are then sampled for each of the initial fragments $\mathbf{Q}_k$.

We show a typical training behavior for learning simultaneously parameters $\theta = (K_0, \gamma, k_BT)$ in Fig. 5. We find that during learning the training initially proceeds by increasing k_BT which broadens the generative model distribution to overlap the training data samples to reduce the loss. However, we see that once the stiffness K_0 and damping γ parameters approach reasonable values, around the epoch ~ 750, the temperature k_BT rapidly decreases to yield models that can more accurately capture the remaining features of the observation data. We see for this training trial that all parameters are close to the correct values around epoch $\sim 1,200$, see Fig. 5.

4.2.1. Comparison to maximum likelihood estimation (MLE) of linear dynamical systems

Many estimators for stationary linear dynamical systems have been developed for dynamics of the general form

$$\mathbf{Z}_n = A\mathbf{Z}_{n-1} + Q\xi_{n-1}, \;\; \xi_{n-1} \sim \eta(0,I), \quad QQ^T = \Gamma. \tag{28}$$

This includes Kalman filters/smoothers Gaussian process approaches, and other likelihood estimators [151,72,149,22,65,148]. The Gaussian assumptions in such dynamics give the log-likelihood for a single trajectory

$$\begin{aligned}\log\left(\Pr\left\{\mathbf{Z}_1,\ldots,\mathbf{Z}_N\right\}\right) = &-\frac{N-1}{2}\log((\det(\Gamma)) \\ &-\frac{1}{2}\sum_{k=2}^{N}\left(\mathbf{Z}_k - A\mathbf{Z}_{k-1}\right)^T\Gamma^{-1}\left(\mathbf{Z}_k - A\mathbf{Z}_{k-1}\right) + \log(\left(\Pr\left\{\mathbf{Z}_1\right\}\right).\end{aligned} \tag{29}$$

This can be averaged further over a sample of trajectories. By maximizing the log-likelihood by computing the gradients and setting them to zero, we obtain the estimators

$$\tilde{\mathbf{A}} = \left(\sum_{n=2}^{N} \mathbb{E}\left[\mathbf{Z}_n \mathbf{Z}_{n-1}^T\right]\right)\left(\sum_{n=2}^{N} \mathbb{E}\left[\mathbf{Z}_{n-1} \mathbf{Z}_{n-1}^T\right]\right)^{-1} \tag{30}$$

$$\tilde{\mathbf{\Gamma}} = \frac{1}{N-1} \sum_{n=2}^{N} \Big[\mathbb{E}\left[\mathbf{Z}_n \mathbf{Z}_n^T\right] - \tilde{\mathbf{A}}\mathbb{E}\left[\mathbf{Z}_{n-1} \mathbf{Z}_n^T\right] \tag{31}$$
$$-\mathbb{E}\left[\mathbf{Z}_n \mathbf{Z}_{n-1}^T\right]\left(\tilde{\mathbf{A}}\right)^T + \tilde{\mathbf{A}}\mathbb{E}\left[\mathbf{Z}_{n-1} \mathbf{Z}_{n-1}^T\right]\left(\tilde{\mathbf{A}}\right)^T\Big].$$

In these expressions, the $\tilde{\mathbf{A}}$ is evaluated then used in estimating the $\tilde{\mathbf{\Gamma}}$.

These expressions yield results similar to Kalman filtering/smoothing when aiming to learn the linear terms in the dynamics using Expectation-Maximization (EM) [47,119]. We can use likelihoods to further obtain estimators for parameters θ, ϕ for the drift contributions when $A = A(\theta) = R + \theta S$ and for the stochastic contributions when $\Gamma = \Gamma(\phi)$ where $\phi = \mathbf{a}^T \Gamma \mathbf{b}$ for some vectors $\mathbf{a}, \mathbf{b}$. Both of these cases include when a parameter contributes linearly to entries of A or Γ. Using a similar approach as in the above expressions, we obtain the estimators

$$\tilde{\theta} = \frac{\sum_{n=2}^{N} \mathbb{E}\left[\mathbf{Z}_n \mathbf{Z}_{n-1}^T\right] : E - R\sum_{n=2}^{N} \mathbb{E}\left[\mathbf{Z}_{n-1} \mathbf{Z}_{n-1}^T\right] : E}{S\sum_{n=2}^{N} \mathbb{E}\left[\mathbf{Z}_{n-1} \mathbf{Z}_{n-1}^T\right] : E}, \qquad \tilde{\phi} = \mathbf{a}^T \tilde{\mathbf{\Gamma}} \mathbf{b}. \tag{32}$$

The $F : E$ denotes tensor contraction (summing the product of matrix entries term-wise). In the case that the parameter appears in component A_{i_0,j_0} we can use $E = \delta_{i,i_0}\delta_{j,j_0}$. More generally, the estimator just needs the matrix E to yield a denominator that is non-zero. The form of the dependence of $\tilde{\mathbf{\Gamma}}$ on $\tilde{\mathbf{A}}$ further suggests that noise-related parameters ϕ could be more challenging to estimate that θ parameters.

Using our Verlet-style integrator above as our linear model for the dynamics we can obtain an MLE approach for estimating the following parameters with the specified target values. We consider trajectories that were generated with the stiffness $K = 1.5$, drag $\gamma = 3.2$, and temperature $k_bT = 0.1$. We take the mass to be $m = 0.1$ and generate trajectories having 18 time-steps to obtain 10^3 samples using the specified time-steps in Δt below. In this study, we use the time step as a proxy for the time-scale on which we probe the physical system. To simplify the estimation problem, we also take the time-steps to be the same when generating the training data as when estimating the physical parameters. We perform estimation using the same linear dynamics as the numerical integrator and obtain the following relative accuracies. For $\Delta t = 5 \times 10^{-4}$, the relative accuracies were $K : 4.6 \times 10^{-1}$, $\gamma : 1.0 \times 10^{-3}$, $k_bT : 2.0 \times 10^0$. We see on this time-scale the stiffness and thermal parameters are not estimated very well. When we increase the time-scale to $\Delta t = 10^{-3}$ we obtain relative accuracies $K : 1.9 \times 10^{-2}$, $\gamma : 1.3 \times 10^{-4}$, $k_bT : 3.9 \times 10^{-2}$. When we further increase the time-scale to $\Delta t = 10^{-2}$ we obtain relative accuracies $K : 2.3 \times 10^{-2}$, $\gamma : 3.4 \times 10^{-3}$, $k_bT : 6.6 \times 10^{-3}$. We find γ is estimated accurately throughout, but the accuracy of K and K_BT are more sensitive and depend on Δt.

We see that such estimators rely on specifying a linear model and approximations that tacitly make assumptions on the time-scale on which $\mathbf{Z}_n$ is represented and on other statistical properties. This is required to ensure the sampling yields matrices and divisors that are non-singular. This further requires sufficient decay in correlations so that the differences appearing in expressions are large enough to be numerically well-conditioned and that the sampling errors are sufficiently small. Among other factors, this can depend sensitively on the choices of Δt and the number of time-steps. For inertial stochastic systems and integrators modeling such dynamics there also can be non-directly measured quantities obscured by fluctuations, such as the velocity or latent variables. While more sophisticated MLE estimators also can be considered, this poses additional analytic complexity to obtain the needed expressions, to obtain problem specific implementations, or to fine tune hyper-parameters to obtain viable estimators [149,22].

While the SDYN-GANs approaches also require probing the dynamics on appropriate time-scales, they require fewer assumptions about the dynamics and probe systems more directly using the marginal and conditional distributions of sample trajectories. This provides for a more general and flexible approach than the estimators based on the stepping dynamics and first and second moment-based estimates. The SDYN-GANs also make fewer assumptions on the parameter dependence and can readily handle cases even when the dependence on the parameters is non-linear in the trajectory observation data.

4.2.2. Results for SDYN-GANs using conditional and marginal distributions over different time-scales

To investigate further the performance of SDYN-GANs, we perform training using a few different approaches. This includes varying the time-scale τ over-which we sample the system configurations $\mathbf{X}(t)$ and the type of distributions implicitly probed. For instance, we use the full-trajectory over a time-duration $\mathcal{T}$, while considering conditionals or marginals statistics over shorter time durations. For this purpose, we use the different training protocols and time-scales τ discussed in Section 2.2. Throughout, we use the default target model parameters $m = 0.1, K_0 = 1.5, \gamma = 3.2, k_BT = 0.1$ and kernel parameters $\alpha = 2, \ell = 0.01$. During training mini-batch sizes of 20 trajectories were used. The duration of training trajectories excluding the initial conditions were $t_N = 1.8 \times 10^{-2}$ over $n_t = 18$ steps with $\Delta t = 10^{-3}$. For the different SDYN-GANs methods and training protocols, we show the accuracy of our learned generative models in Tables 2–4.

We find the stiffness and damping mechanics each can be learned with an accuracy of around 5% or less. The thermal fluctuations of the system were more challenging to learn from the trajectory observations. This appeared to arise from other contributing factors to perceived fluctuations associated with sampling errors and a tendency to over-estimate the thermal parameters to help enhance alignment of the candidate generative model with the trajectory data. We found accuracies of only around 50% for the thermal

Table 2
Accuracy K_0 verses τ-delay: inertial Ornstein-Uhlenbeck process. SDYN-GANs training performed using the protocols over the (i) full trajectory, (ii) marginals, and (iii) conditionals, as discussed in Section 2.2. The accuracy of the stiffness estimate $\tilde{\phi}$ for $\phi = K_0$ given by the relative error $\epsilon_{\text{rel}} = |\tilde{\phi} - \phi|/|\phi|$. The number of epochs was 3000. Standard deviations are reported for 4 runs. The duration of trajectories is $t_N = 1.8 \times 10^{-2}$ over $n_t = 18$ steps with $\Delta t = 10^{-3}$.

Accuracy K_0 vs τ-Delay			
τ-Delay	**1.70e-02**	**1.19e-02**	**5.83e-03**
MMD-full-traj	1.03e-02 ± 1.1e-02	2.62e-02 ± 2.4e-02	1.13e-02 ± 7.8e-03
MMD-conditionals	9.19e-03 ± 1.9e-03	2.22e-02 ± 1.7e-02	2.66e-02 ± 4.5e-02
MMD-marginals	1.54e-02 ± 6.7e-03	8.92e-03 ± 1.1e-02	4.37e-02 ± 4.5e-02
τ-Delay	**4.08e-03**	**2.86e-03**	**2.00e-03**
MMD-full-traj	1.62e-02 ± 1.5e-02	8.02e-03 ± 5.2e-03	1.37e-02 ± 7.0e-03
MMD-conditionals	1.03e-01 ± 1.9e-01	8.52e-01 ± 1.7e+00	5.68e-01 ± 1.1e+00
MMD-marginals	4.44e-02 ± 8.0e-02	1.63e+00 ± 3.2e+00	1.72e+00 ± 3.4e+00

Table 3
Accuracy k_BT verses τ-delay: inertial Ornstein-Uhlenbeck process. SDYN-GANs training performed using the protocols over the (i) full trajectory, (ii) marginals, and (iii) conditionals, as discussed in Section 2.2. The accuracy of the temperature estimate $\tilde{\phi}$ for $\phi = k_BT$ given by the relative error $\epsilon_{\text{rel}} = |\tilde{\phi} - \phi|/|\phi|$. The number of epochs was 3000. Standard deviations are reported for 4 runs. The duration of trajectories is $t_N = 1.8 \times 10^{-2}$ over $n_t = 18$ steps with $\Delta t = 10^{-3}$.

Accuracy k_BT vs τ-Delay			
τ-Delay	**1.70e-02**	**1.19e-02**	**5.83e-03**
MMD-full-traj	5.36e-01 ± 9.5e-01	1.71e+00 ± 3.4e+00	3.62e+00 ± 7.1e+00
MMD-conditionals	6.43e-01 ± 1.1e+00	1.46e+00 ± 2.6e+00	1.92e+01 ± 3.8e+01
MMD-marginals	6.54e-01 ± 1.2e+00	1.51e+00 ± 2.9e+00	1.39e+01 ± 2.7e+01
τ-Delay	**4.08e-03**	**2.86e-03**	**2.00e-03**
MMD-full-traj	3.47e+00 ± 6.7e+00	6.43e+00 ± 1.3e+01	6.90e+00 ± 1.4e+01
MMD-conditionals	2.42e+01 ± 4.7e+01	9.54e+01 ± 1.7e+02	9.38e+01 ± 1.7e+02
MMD-marginals	1.85e+01 ± 3.7e+01	3.95e+01 ± 7.9e+01	3.82e+01 ± 7.6e+01

Table 4
Accuracy γ verses τ-delay: inertial Ornstein-Uhlenbeck process. SDYN-GANs training performed using the protocols over the (i) full trajectory, (ii) marginals, and (iii) conditionals, as discussed in Section 2.2. The accuracy of the damping estimate $\tilde{\phi}$ for $\phi = \gamma$ given by the relative error $\epsilon_{\text{rel}} = |\tilde{\phi} - \phi|/|\phi|$. The number of epochs was 3000. Standard deviations are reported for 4 runs. The duration of trajectories is $t_N = 1.8 \times 10^{-2}$ over $n_t = 18$ steps with $\Delta t = 10^{-3}$.

Accuracy γ vs τ-Delay			
τ-Delay	**1.70e-02**	**1.19e-02**	**5.83e-03**
MMD-full-traj	1.53e-02 ± 1.5e-02	3.31e-02 ± 2.5e-02	2.91e-02 ± 4.3e-02
MMD-conditionals	2.85e-02 ± 2.3e-02	6.57e-02 ± 6.0e-02	2.29e-02 ± 3.3e-02
MMD-marginals	1.67e-02 ± 8.4e-03	3.34e-02 ± 1.9e-02	6.20e-02 ± 8.9e-02
τ-Delay	**4.08e-03**	**2.86e-03**	**2.00e-03**
MMD-full-traj	5.20e-02 ± 4.7e-02	4.70e-02 ± 4.6e-02	2.20e-02 ± 1.8e-02
MMD-conditionals	2.00e-01 ± 3.7e-01	1.91e+00 ± 3.8e+00	1.88e+00 ± 3.7e+00
MMD-marginals	1.90e-01 ± 3.0e-01	5.54e+00 ± 1.1e+01	5.47e+00 ± 1.1e+01

fluctuations of the system. We found the time delay τ used in probing the temporal sub-sampling of the dynamics can play a significant role in the performance of the learning methods.

For the temporal sampling, we found when the τ time-scales are relatively large the models were learned more accurately. This appears to be related to the samples better probing the time-scales in the trajectory data over-which the stiffness and damping make stronger contributions. The longer trajectory durations appear to provide a better signal for learning these properties. As τ became smaller, we found the accuracy of the learned models decreased for this task.

We further found the training protocol chosen could have a significant impact on the accuracy of the learned models. For the current task, we found throughout that either the full-trajectory protocols or marginals protocols tended to perform the best. A notable feature of the marginals protocol is that the trajectory samples include the same fragments of the trajectory arising in the conditionals, particularly those starting near the sampled initial conditions. This appears to have contributed to the marginals out-performing the conditionals, the latter of which contain less information about the longer-time dynamics.

We found overall a better job was done by the full-trajectory training protocol which uses more steps along the trajectory than the marginals and conditionals protocols. The marginals also tended to do better than the conditionals. As an exception, we did find in a few cases for small τ the conditionals did a better job than the marginals in estimating the stiffness. This was perhaps since when working over shorter time durations, the more temporally localized sampling near the same initial conditions in the conditionals

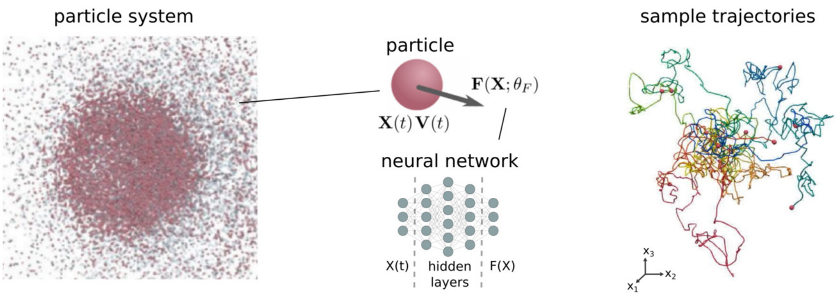

Fig. 6. Learning non-linear force-laws. SDYN-GANs is used to learn an unknown force-law from observation of particle trajectories *(left)*. The learned force-law can be represented using neural networks or other model classes *(middle)*. The generative models use the dependence in the probabilities of the stochastic trajectories using *m*-step stochastic numerical discretizations *(right)*. The models are learned using a few different training protocols using statistics of the (i) full-trajectory, (ii) conditionals, and (iii) marginals, as discussed in Section 2.2.

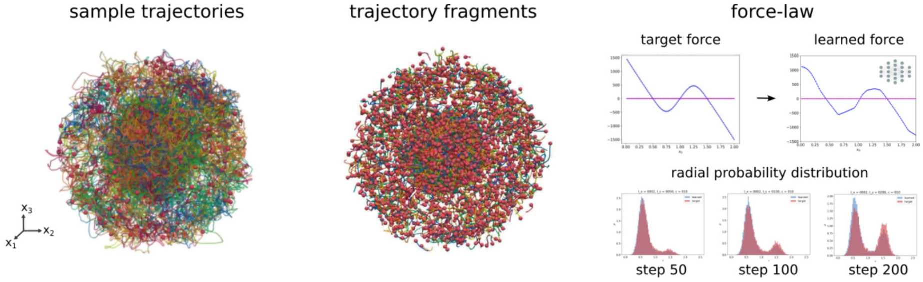

Fig. 7. Learning non-linear force laws: trajectory data. We show a few samples of the particle trajectories *(left)*. The force-law is learned from samples of trajectory fragments using the different training protocols discussed in Section 2.2, *(middle)*. SDYN-GANs is used to learn a neural network representation of the force-law from observations of trajectories of the particle system. We show a few empirical radial probability distributions computed from histograms using the range $[0, L]$ with $L = 2.5$ and bin-width $\Delta x = 0.05$ *(right)*. We evaluate the accuracy of the learned model by considering the relative L^1-norm of marginal time-dependent radial probability distributions of samples of the learned model compared with the target particle system.

provide more consistent and less obscured information on the stiffness responses, relative to the full set of trajectory fragments contributing in the marginals protocols.

For the longer τ cases when comparing with the conditionals, the marginals tended to more accurately estimate simultaneously the stiffness and damping. This likely arose, since unlike the conditionals case, the marginals do not require the initial parts of the trajectory fragments being compared to share the same initial conditions. For the general short-duration fragments, there is additional information on the long-time dynamical responses contributing to the marginals. In summary, the results suggest using the full-trajectory protocols when possible and relatively long τ sampling. When one is only able to probe short-duration trajectory fragments, the results suggest the marginals provide some advantages over the conditionals. Which protocols work best may be dependent on the task and the time-scales associated with the dynamical behaviors being modeled. These results show a few strategies for using SDYN-GANs to learn generative models for stochastic systems.

4.3. Learning non-linear force-laws with SDYN-GANs generative modeling

We show how SDYN-GANs can be used to learn force-laws $\mathbf{F}(\mathbf{X})$ from observations of trajectories of stochastic systems. We consider the inertial Langevin dynamics of particle systems of the form

$$m d\mathbf{V}(t) = -\gamma \mathbf{V}(t)dt + \mathbf{F}(\mathbf{X};\theta_F)dt + \sqrt{2k_BT\gamma}\, d\mathbf{W}(t), \quad d\mathbf{X}(t) = \mathbf{V}(t)dt, \tag{33}$$

where $\mathbf{X}, \mathbf{V} \in \mathbb{R}^3$. For individual particles this dynamics gives stochastic trajectories in a 6 dimensional space. The methods also readily can be generalized with $\mathbf{X}, \mathbf{V} \in \mathbb{R}^n$ giving stochastic trajectories in a $2n$ dimensional space. The force-law $\mathbf{F}(\mathbf{X})$ will be modeled using Deep Neural Networks (DNNs) [84,53]. In this case, θ_F are the DNN weights and biases to be learned. We consider here the case of radial potentials $\mathbf{F}(\mathbf{X}) = \mathbf{F}(\|\mathbf{X}\|) = F(r)\mathbf{e}_r$ with $\mathbf{e}_r = \mathbf{X}/\|\mathbf{X}\|$. For simplicity, we focus on the case where m, γ, T are fixed, but these also could be learned in conjunction with the force-law. The data-driven task is to learn DNN representations $\mathbf{F}(\mathbf{X};\theta_F)$ for the force-law from the configuration trajectories $\mathbf{X}_{[0,\mathcal{T}]} = \{\mathbf{X}(t)\}_{t\in[0,\mathcal{T}]}$ discretized as $(\mathbf{X}(t_1),\ldots,\mathbf{X}(t_N))$, see Fig. 6. The $\mathbf{V}(t)$ is not observed directly. We use for our generative models the Langevin dynamics of equation (33) approximated by the $(m = 2)$-step stochastic numerical discretizations discussed in Section 4.1. We use the default values in training given in Table 1. We show samples of the training data in Fig. 7.

In our generative models, we use DNN architectures of multi-layer perceptrons with input-hidden-output sizes n_{in}-n_1-n_2-n_3-n_{out} and LeakyReLU [155] units with *slope* = -0.01. In the DNNs we use $n_{\text{out}} = n_{\text{in}} = 1$, with default values of $n_1 = n_2 = n_3 = 100$ and biases for all layers except for the last layer. The DNNs serve to model the F in the radial force $\mathbf{F} = F(r;\theta_F)\mathbf{e}_r$. For learning the dynamics we

Table 5
L^1-Accuracy of learned force-law. We investigate how the relative L^1-errors of the time-dependent marginal probability densities at times $I_{step} = 50, 100, 200$, epoch = 5000, compare between the learned generative models and the target stochastic process. The mean and standard deviation are reported over 5 runs. SDYN-GANs training protocols are based on trajectory samples from distributions of the (i) full trajectory, (ii) conditionals, and (iii) marginals, for details see Section 2.2. The L^1-relative-errors of the empirical radial probability distributions were computed from histograms using the range $[0, L]$ with $L = 2.5$ and bin-width $\Delta x = 0.125$. The duration of trajectories is $t_{n_t} = 2.0 \times 10^{-2}$ over $n_t = 20$ steps with $\Delta t = 10^{-3}$.

Accuracy vs τ-Delay			
Method / τ-Delay	**1.90e-02**	**1.16e-02**	**7.12e-03**
MMD-full-traj ($I_s = 50$)	1.20*e*-01 ± 4.9*e*-02	1.02*e*-01 ± 4.1*e*-02	8.67*e*-02 ± 3.2*e*-02
MMD-conditionals	1.33*e*-01 ± 3.5*e*-02	1.42*e*-01 ± 6.1*e*-02	1.42*e*-01 ± 4.2*e*-02
MMD-marginals	8.20*e*-02 ± 4.5*e*-02	1.05*e*-01 ± 3.4*e*-02	9.00*e*-02 ± 2.3*e*-02
MMD-full-traj ($I_s = 100$)	1.56*e*-01 ± 8.8*e*-02	1.27*e*-01 ± 4.3*e*-02	1.06*e*-01 ± 2.8*e*-02
MMD-conditionals	1.39*e*-01 ± 4.0*e*-02	1.68*e*-01 ± 5.7*e*-02	1.33*e*-01 ± 4.7*e*-02
MMD-marginals	9.75*e*-02 ± 4.7*e*-02	9.96*e*-02 ± 3.4*e*-02	9.97*e*-02 ± 2.3*e*-02
MMD-full-traj ($I_s = 200$)	2.22*e*-01 ± 1.6*e*-01	1.69*e*-01 ± 7.7*e*-02	1.51*e*-01 ± 3.8*e*-02
MMD-conditionals	1.67*e*-01 ± 6.2*e*-02	2.28*e*-01 ± 8.3*e*-02	2.28*e*-01 ± 3.7*e*-02
MMD-marginals	1.16*e*-01 ± 3.1*e*-02	1.24*e*-01 ± 2.3*e*-02	1.15*e*-01 ± 2.4*e*-02

consider the case with target forces arising from a double-well radial potential $U(r) = \frac{k}{4}(r - r_1)^2(r - r_2)^2$. This gives the target force $F(r) = -U'(r) = -\frac{k}{2}\left((r - r_1)(r - r_2)^2 + (r - r_1)^2(r - r_2)\right)$, where $k = 500$, $r_1 = 0.5$, and $r_2 = 1.5$. For the other parameters the model had $m = 0.1$, $k_B T = 50$, $\gamma = 32$.

We use SDYN-GANs to learn generative models by probing trajectories on different time-scales τ and training protocols that are based on sampling the (i) full-trajectory, (ii) conditionals, and (iii) marginals, as discussed in Section 2.2. To train the models, we simulate the trajectories of the candidate generative models and compare with the samples of the observation training data. To characterize the final accuracy of the learned models, we use the relative errors under the L^1-norm of the time-dependent radial probability distributions to compare the samples of the generative model with the target particle system. We emphasize the L^1-error is not used during training, but only on the final models to characterize the accuracy of the stochastic trajectories they generate. We train using SDYN-GANs with different time-scales τ and training protocols, with our results reported in Table 5.

We find the generative models learn the force-law with an accuracy of around 20%. We found the training protocols using the full trajectory and the marginals performed best, see Table 5. We remark that in theory the large τ in the infinite time horizon limit would give samples for $\mathbf{X}$ from the equilibrium Gibbs-Boltzmann distribution, $\rho(\mathbf{X}) = 1/Z \exp\left(-U(\mathbf{X})/k_B T\right)$. The equilibrium distribution only depends on the potential U of the force-law. Interestingly, we find in the training there was not a strong dependence in the results on the choice of τ. The results highlight that in contrast to equilibrium estimators, the SDYN-GANs learning methods allow for flexibility in estimating the force-law even in the non-equilibrium setting and from observations over shorter-time trajectories relative to the equilibration time-scale.

We found for this task the sampling from the full trajectory and marginals tended to perform better than the conditionals. For the shorter-time durations, this appears to arise from the marginals including contributions from trajectory fragments that are included in the conditionals and also trajectory fragments from later parts of the trajectories. Again, this provides additional information on the longer-time dynamical responses, here enhancing learning of the force-laws. The matching of the marginals also may be more stringent during training than the conditionals. Obtaining agreement requires more global coordination in the generated trajectories so that statistics of later segments of the trajectory match the observed target system.

We remark that for both the conditionals and marginals, the use of shorter duration trajectory fragments helps reduce the dimensionality of the samples. This appears to improve in the loss function the ability to make comparisons with the observation data. The results indicate the marginals again may provide some advantages for training SDYN-GANs when one is able only to probe trajectory data in fragments over relatively short time durations.

In summary, the results show a few strategies for using SDYN-GANs for learning non-linear force-laws and other physical properties from observations of the system dynamics. The samples-based methods of SDYN-GANs allow for general generative modeling classes to be used, without a need for specification of likelihood functions. This facilitates the use of general generative models for learning generative models of stochastic systems. The SDYN-GANs approaches also can be used for non-neural network modeling classes, and combined with other training protocols, such as data augmentation, further regularizations, and other prior information motivated by the task and knowledge from the application domain.

5. Conclusions

We introduced SDYN-GANs for adversarial learning of generative models of stochastic dynamical systems. The learning approaches only require making comparisons between empirical trajectory samples from the candidate generative model and observation data. This is in contrast to MLE which requires specifying likelihood functions. We developed practical adversarial training methods for generative models based on stable m-step stochastic numerical integrators. We showed results for how this can be used to facilitate learning models for long-time prediction and simulation. We introduced several training methods for utilizing probabilistic dependencies, leveraging different statistical properties of the conditional and marginal distributions, and for probing dynamics on different time-scales. We showed how our methods can be used for physical systems to learn force-laws, mechanical responses, and thermal parameters. The introduced SDYN-GANs methods provide versatile approaches for learning robust stochastic dynamical models for use in diverse tasks arising in machine learning, statistical inference, dynamical systems identification, and scientific computation.

CRediT authorship contribution statement

P. Stinis: Investigation. **C. Daskalakis:** Investigation. **P.J. Atzberger:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Authors PJA and CD were supported by grant DOE Grant ASCR PHILMS DE-SC0019246. The author PJA was also supported by NSF Grant DMS-1616353 and NSF Grant DMS-2306101. Author PJA also acknowledges UCSB Center for Scientific Computing NSF MRSEC (DMR1121053) and UCSB MRL NSF CNS-1725797. The work of PS is supported by the DOE-ASCR DE-AC05-76RL01830-funded "Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems (PhILMs)." Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for DOE under Contract DE-AC05-76RL01830. Author PJA would also like to acknowledge a hardware grant from Nvidia.

Appendix A. Gradient equations for *m*-step generative models

For our m-step generative models, the use of direct backpropagation for book-keeping the forward calculations in generating trajectory samples can result in prohibitively large computational call graphs. This is further increased as the time duration of the trajectories grows. We derive alternative adjoint approaches to obtain more efficient methods for loss functions to compute the gradients needed during training.

For our general m-step generative models discussed in Section 3.1, we use the formulation $\mathbf{f}(\mathbf{x},\mathbf{p}) = 0$. For $m \leq j \leq N$, we have

$$[\mathbf{f}]_{(j,d)} = \left[\mathbf{X}_j - \boldsymbol{\Psi}_{j-1}(\mathbf{X}_{j-1},\ldots,\mathbf{X}_{j-m},\boldsymbol{\omega}_{j-1};\mathbf{p})\right]^{(d)} = 0. \tag{34}$$

The index (j,d) refers to the j^{th} time-step and d^{th} dimension component associated with $\mathbf{X}_j^{(d)}$. We use for this the notations $\mathbf{X}_j^{(d)} = \left[\mathbf{X}_j\right]^{(d)} = \mathbf{X}_{(j,d)}$. The update map is denoted as $\boldsymbol{\Psi}_{j-1}$. For $j = 0,\ldots,m-1$, we have

$$[\mathbf{f}]_{(0,d)} = \left[\mathbf{X}_0 - \mathbf{x}_0\right]^{(d)} = 0,\ \ [\mathbf{f}]_{(m-1,d)} = \left[\mathbf{X}_{m-1} - \mathbf{x}_{m-1}\right]^{(d)} = 0. \tag{35}$$

We can then provide the gradients for $\nabla_{X_j}\boldsymbol{\Psi}_j,\ldots,\nabla_{X_j}\boldsymbol{\Psi}_{j+m}$ and other terms.

We derive methods for solving efficiently $J^T\mathbf{r} = -\mathbf{f}_p$ for $\mathbf{r}$ where $J = \mathbf{f}_x$. For details motivating this formulation, see Section 3.1. Throughout, we use the Einstein convention for repeated indices, which contract the two tensors over the shared index by summing the product [61]. We also first show the overall product with $\mathbf{f}_x$ being computed and then give the detailed form of this term which shows the structure of the equations we need to solve. By using the structure of $J = \mathbf{f}_x$ from equations (34)–(35), we can obtain the following set of recurrence equations for the components of $\mathbf{r}$. For $0 \leq k \leq N-m$, we have

$$r_{(j,d)}f_{(j,d),(k,d')} = r_{(k,d)}\delta_{d,d'} - \left[\nabla_{X_{(k,d')}}\Psi_{(k,d)}\right]r_{(k+1,d)}\cdots - \left[\nabla_{X_{(k,d')}}\Psi_{(k+m,d)}\right]r_{(k+m,d)} = \frac{\partial\phi}{\partial X_{(k,d')}}. \tag{36}$$

For $k = N-\ell$, $1 \leq \ell \leq m-1$, we have

$$r_{(j,d)}f_{(j,d),(k,d')} = r_{(k,d)}\delta_{d,d'} - \left[\nabla_{X_{(k,d')}}\Psi_{(k,d)}\right]r_{(k+1,d)}\cdots - \left[\nabla_{X_{(k,d')}}\Psi_{(k+\ell,d)}\right]r_{(k+\ell,d)} = \frac{\partial\phi}{\partial X_{(k,d')}}. \tag{37}$$

For $k = N$, we have

$$r_{(j,d)}f_{(j,d),(k,d')} = r_{(N,d)}\delta_{d,d'} = \frac{\partial\phi}{\partial X_{(N,d')}}. \tag{38}$$

This gives an m^{th}-order recurrence relation we can use to propagate values from $N, N-1,\ldots,0$ to obtain the components of $\mathbf{r}$.

To obtain $\mathbf{f}_p$, we have for $m \leq j \leq N$, that

$$[\mathbf{f}_p]_{(j,d)} = -\nabla_p \boldsymbol{\Psi}_{(j-1,d)}. \tag{39}$$

For $0 \leq j \leq m-1$, we have

$$[\mathbf{f}]_{(j,d)} = \left[\mathbf{X}_j - \mathbf{x}_j\right]^{(d)} = 0, \;\; [\mathbf{f}]_{(j,d),(k,d')} = \delta_{(j,d),(k,d')}, \;\; [\mathbf{f}_p]_{(j,d)} = -\frac{\partial \left[\mathbf{x}_j\right]^{(d)}}{\partial p}. \tag{40}$$

We compute the gradients needed for training by using

$$\nabla_p \tilde{g} = \tilde{\mathbf{g}}_p - \mathbf{r}^T \mathbf{f}_p. \tag{41}$$

To compute $\mathbf{r}$, we solve for each sample $\mathbf{X}^{[i]}$ of the stochastic trajectories the m^{th}-order recurrence relations given in equations (36)–(38). The $\tilde{\mathbf{g}}_p = \partial \tilde{\mathbf{g}}/\partial p$ and $\mathbf{f}_p$ are computed using equations (39)–(40). The final gradient is computed by averaging using $\nabla_p \bar{g} = \frac{1}{M} \sum_{1=1}^{M} \nabla_p \tilde{g}^{[i]}$ over the M samples of the stochastic trajectories indexed by $[i]$. There are also ways to obtain additional efficiencies by using structure of the specific loss functions, as we discuss in Section 3.2. We use equations (34)–(41) to develop efficient methods for computing the gradients of our m-step generative models.

References

[1] Gabriele Abbati, et al., AReS and MaRS adversarial and MMD-minimizing regression for SDEs, in: Kamalika Chaudhuri, Ruslan Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 1–10.

[2] Yacine Aït-Sahalia, Robert Kimmel, Maximum likelihood estimation of stochastic volatility models, J. Financ. Econ. 83 (2) (2007) 413–452.

[3] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein generative adversarial networks, in: Doina Precup, Yee Whye Teh (Eds.), Proceedings of Machine Learning Research, in: Proceedings of Machine Learning Research, vol. 70, PMLR, International Convention Centre, Sydney, Australia, 2017, pp. 214–223.

[4] N. Aronszajn, Theory of reproducing kernels, Trans. Am. Math. Soc. 68 (3) (1950) 337–404, ISSN: 00029947.

[5] Paul J. Atzberger, Peter R. Kramer, Charles S. Peskin, A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales, J. Comput. Phys. 224 (2) (2007) 1255–1292.

[6] Filipe de Avila Belbute-Peres, et al., End-to-end differentiable physics for learning and control, Adv. Neural Inf. Process. Syst. 31 (2018).

[7] Coryn A.L. Bailer-Jones, David J.C. MacKay, Philip J. Withers, A recurrent neural network for modelling dynamical systems, Netw. Comput. Neural Syst. 9 (4) (1998) 531.

[8] Marc G. Bellemare, et al., The Cramer Distance as a Solution to Biased Wasserstein Gradients, 2018.

[9] Yoshua Bengio, Patrice Simard, Paolo Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.

[10] Alain Berlinet, Christine Thomas-Agnan, Reproducing Kernel Hilbert Spaces in Probability and Statistics, Springer Science & Business Media, 2011.

[11] Mikolaj Binkowski, et al., Demystifying MMD GANs, in: International Conference on Learning Representations, 2018.

[12] Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, ISBN 0387310738, 2006.

[13] Léon Bottou, et al., Stochastic gradient learning in neural networks, Proceedings of Neuro-Nımes 91 (8) (1991) 12.

[14] Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. (ISSN 0027-8424) 113 (15) (2016) 3932–3937, https://doi.org/10.1073/pnas.1517384113, eprint, https://www.pnas.org/content/113/15/3932.full.pdf.

[15] Evelyn Buckwar, Renate Winkler, Multistep methods for SDEs and their application to problems with small noise, SIAM J. Numer. Anal. 44 (2) (2006) 779–803.

[16] Richard L. Burden, Douglas Faires, Numerical Analysis, Brooks/Cole Cengage Learning, 2010.

[17] Jean Céa, Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût, ESAIM: Math. Model. Numer. Anal. 20 (3) (1986) 371–402.

[18] Kathleen Champion, et al., Data-driven discovery of coordinates and governing equations, Proc. Natl. Acad. Sci. 116 (45) (2019) 22445–22451.

[19] Tian Qi Chen, et al., Neural ordinary differential equations, in: Bengio Samy, et al. (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada, 3–8 December 2018, 2018, pp. 6572–6583.

[20] Zhao Chen, Yang Liu, Hao Sun, Physics-informed learning of governing equations from scarce data, Nat. Commun. 12 (1) (2021) 6136.

[21] Kyunghyun Cho, et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint, arXiv:1406.1078, 2014.

[22] Charles K. Chui, et al., Extended Kalman filter and system identification, in: Kalman Filtering: with Real-Time Applications, 2017, pp. 115–137.

[23] Jerome T. Connor, R. Douglas Martin, Les E. Atlas, Recurrent neural networks and robust time series prediction, IEEE Trans. Neural Netw. 5 (2) (1994) 240–254.

[24] Henry Cox, On the estimation of state variables and parameters for noisy dynamic systems, IEEE Trans. Autom. Control 9 (1) (1964) 5–12.

[25] Jan Salomon Cramer, Econometric Applications of Maximum Likelihood Methods, CUP Archive, 1989.

[26] Arthur P. Dempster, Nan M. Laird, Donald B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B 39 (1) (1977) 1–22.

[27] Ishan Deshpande, et al., Max-sliced Wasserstein distance and its use for gans, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10648–10656.

[28] Rahul Dey, Fathi M. Salem, Gate-variants of gated recurrent unit (GRU) neural networks, in: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), IEEE, 2017, pp. 1597–1600.

[29] Mucong Ding, Constantinos Daskalakis, Soheil Feizi, GANs with conditional independence graphs: on subadditivity of probability divergences, in: Arindam Banerjee, Kenji Fukumizu (Eds.), Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 130, PMLR, 2021, pp. 3709–3717.

[30] Mucong Ding, Constantinos Daskalakis, Soheil Feizi, Subadditivity of probability divergences on Bayes-nets with applications to time series GANs, preprint, arXiv:2003.00652, 2020.

[31] Georg Dorffner, Neural networks for time series processing, Neural Netw. World 6 (4) (1996) 447–468.

[32] Alexey Dosovitskiy, et al., An image is worth 16x16 words: transformers for image recognition at scale, arXiv preprint, arXiv:2010.11929, 2020.

[33] J.-P.S. Draye, et al., Dynamic recurrent neural networks: a dynamical analysis, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 26 (5) (1996) 692–706.

[34] O.K. Dudko, et al., Dynamic force spectroscopy: a Fokker–Planck approach, Chem. Phys. Lett. 352 (5–6) (2002) 499–504.

[35] David Duvenaud, The kernel cookbook: advice on covariance functions, https://www.cs.toronto.edu/~duvenaud/cookbook, 2014.

[36] Gintare Karolina Dziugaite, Daniel M. Roy, Zoubin Ghahramani, Training generative neural networks via maximum mean discrepancy optimization, in: Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI'15, AUAI Press, Amsterdam, Netherlands, ISBN 9780996643108, 2015, pp. 258–267.
[37] S. Eftekhar Azam, M. Bagherinia, S. Mariani, Stochastic system identification via particle and sigma-point Kalman filtering, Sci. Iran. (ISSN 1026-3098) 19 (4) (2012) 982–991, https://doi.org/10.1016/j.scient.2012.06.007.
[38] N. Benjamin Erichson, et al., Shallow learning for fluid flow reconstruction with limited sensors and limited data, CoRR, arXiv:1902.07358 [abs], 2019, arXiv:1902.07358.
[39] Brian D. Ewald, Roger Témam, Numerical analysis of stochastic schemes in geophysics, SIAM J. Numer. Anal. 42 (6) (2005) 2257–2276.
[40] Urban Fasel, et al., Ensemble-SINDy: robust sparse model discovery in the low-data, high-noise limit, with active learning and control, Proc. R. Soc. A 478 (2260) (2022) 20210904.
[41] Robert Fortet, Edith Mourier, Convergence de la répartition empirique vers la répartition théorique, Ann. Sci. Éc. Norm. Supér. (3) 70 (3) (1953) 267–285, https://doi.org/10.24033/asens.1013.
[42] Daan Frenkel, Berend Smit, Understanding Molecular Simulation: from Algorithms to Applications, vol. 1, Elsevier, 2001.
[43] William D. Fries, Xiaolong He, Youngsoo Choi, Lasdi: parametric latent space dynamics identification, Comput. Methods Appl. Mech. Eng. 399 (2022) 115436.
[44] C.W. Gardiner, Handbook of Stochastic Methods, Series in Synergetics, Springer, 1985.
[45] Zhenglin Geng, Daniel Johnson, Ronald Fedkiw, Coercing machine learning to output physically accurate results, J. Comput. Phys. 406 (2020) 109099.
[46] Amin Ghadami, Bogdan I. Epureanu, Data-driven prediction in dynamical systems: recent developments, Philos. Trans. R. Soc. A 380 (2229) (2022) 20210213.
[47] Zoubin Ghahramani, Geoffrey E. Hinton, Parameter Estimation for Linear Dynamical Systems, 1996.
[48] Michael Giles, Paul Glasserman, Smoking adjoints: fast evaluation of Greeks in Monte Carlo calculations, Risk (2006).
[49] Michael B. Giles, Niles A. Pierce, An introduction to the adjoint approach to design, Flow Turbul. Combust. (ISSN 1573-1987) 65 (3) (2000) 393–415, https://doi.org/10.1023/A:1011430410075.
[50] L.T. Giorgini, W. Moon, J.S. Wettlaufer, Analytical survival analysis of the Ornstein-Uhlenbeck process, J. Stat. Phys. (ISSN 1572-9613) 181 (6) (2020) 2404–2414, https://doi.org/10.1007/s10955-020-02669-y.
[51] Rémi Goerlich, et al., Noise and ergodic properties of Brownian motion in an optical tweezer: looking at regime crossovers in an Ornstein-Uhlenbeck process, Phys. Rev. E 103 (3) (2021) 032132.
[52] J.R. Gomez-Solano, et al., Steady-state fluctuation relations for systems driven by an external random force, Europhys. Lett. 89 (6) (2010) 60003, https://doi.org/10.1209/0295-5075/89/60003.
[53] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, The MIT Press, ISBN 0262035618, 2016.
[54] Ian Goodfellow, et al., Generative adversarial nets, in: Z. Ghahramani, et al. (Eds.), Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 2672–2680.
[55] Arthur Gretton, et al., A kernel two-sample test, J. Mach. Learn. Res. 13 (25) (2012) 723–773.
[56] Mohinder S. Grewal, Angus P. Andrews, Chris G. Bartone, Kalman filtering, 2020.
[57] Niels Gronbech-Jensen, Oded Farago, A simple and effective Verlet-type algorithm for simulating Langevin dynamics, Mol. Phys. (ISSN 1362-3028) 111 (8) (2013) 983–991, https://doi.org/10.1080/00268976.2012.760055.
[58] B.J. Gross, et al., Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: a generalized moving least-squares (GMLS) approach, J. Comput. Phys. (ISSN 0021-9991) 409 (2020) 109340, https://doi.org/10.1016/j.jcp.2020.109340.
[59] Ishaan Gulrajani, et al., Improved training of Wasserstein GANs, in: I. Guyon, et al. (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5767–5777.
[60] Siyuan Guo, et al., Causal de Finetti: on the identification of invariant causal structure in exchangeable data, https://doi.org/10.48550/ARXIV.2203.15756, 2022.
[61] Wolfgang Hackbusch, Tensor Spaces and Numerical Tensor Calculus, vol. 42, Springer, 2012.
[62] Lars Peter Hansen, Large sample properties of generalized method of moments estimators, Econometrica (1982) 1029–1054.
[63] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Elements of Statistical Learning, Springer Series in Statistics, Inc., New York, NY, USA: Springer New York, 2001.
[64] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[65] Masaru Hoshiya, Etsuro Saito, Structural identification by extended Kalman filter, J. Eng. Mech. 110 (12) (1984) 1757–1770, https://doi.org/10.1061/(ASCE)0733-9399(1984)110:12(1757), eprint https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9399%281984%29110%3A12%281757%29.
[66] A. Stan Hurn, J.I. Jeisman, Kenneth A. Lindsay, Seeing the wood for the trees: a critical evaluation of methods to estimate the parameters of stochastic differential equations, J. Financ. Econom. 5 (3) (2007) 390–455.
[67] Arieh Iserles, A First Course in the Numerical Analysis of Differential Equations, Cambridge University Press, 2009.
[68] Dev Jasuja, P.J. Atzberger, Magnus exponential integrators for stiff time-varying stochastic systems, preprint, arXiv:2212.08978, 2022, https://doi.org/10.48550/ARXIV.2212.08978.
[69] K. Jittorntrum, An implicit function theorem, J. Optim. Theory Appl. (ISSN 1573-2878) 25 (4) (1978) 575–577, https://doi.org/10.1007/BF00933522.
[70] Simon J. Julier, Jeffrey K. Uhlmann, New extension of the Kalman filter to nonlinear systems, in: Signal Processing, Sensor Fusion, and Target Recognition VI, vol. 3068, SPIE, 1997, pp. 182–193.
[71] C. Kaebe, J.H. Maruhn, E.W. Sachs, Adjoint-based Monte Carlo calibration of financial market models, Finance Stoch. (ISSN 1432-1122) 13 (3) (2009) 351–379, https://doi.org/10.1007/s00780-009-0097-9.
[72] R.E. Kalman, R.S. Bucy, New results in linear filtering and prediction theory, J. Basic Eng. (ISSN 0021-9223) 83 (1) (Mar. 1961) 95–108, https://doi.org/10.1115/1.3658902, eprint, https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/83/1/95/5503549/95_1.pdf.
[73] Sebastian Kaltenbach, Phaedon-Stelios Koutsourelakis, Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems, J. Comput. Phys. 419 (2020) 109673.
[74] Patrick Kidger, et al., Neural sdes as infinite-dimensional gans, in: International Conference on Machine Learning, PMLR, 2021, pp. 5453–5463.
[75] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, preprint, arXiv:1412.6980, 2014.
[76] P.E. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, Springer-Verlag, 1992.
[77] Naveen Kodali, et al., On convergence and stability of gans, arXiv preprint, arXiv:1705.07215, 2017.
[78] Soheil Kolouri, Yang Zou, Gustavo K. Rohde, Sliced Wasserstein Kernels for Probability Distributions, https://doi.org/10.1109/cvpr.2016.568, 2016.
[79] Milan Korda, Igor Mezic, Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control, Automatica 93 (2018) 149–160, https://doi.org/10.1016/j.automatica.2018.03.046.
[80] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (Mar. 1951) 79–86, https://doi.org/10.1214/aoms/1177729694.
[81] Harold Kushner, Approximations to optimal nonlinear filters, IEEE Trans. Autom. Control 12 (5) (1967) 546–556.
[82] Henning Lange, Steven L. Brunton, J. Nathan Kutz, From Fourier to Koopman: spectral methods for long-term time series prediction, J. Mach. Learn. Res. 22 (1) (2021) 1881–1918.
[83] Steven J. Large, Stochastic control in microscopic nonequilibrium systems, in: Dissipation and Control in Microscopic Nonequilibrium Systems, Springer, 2021, pp. 91–111.

[84] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, Nature (ISSN 1476-4687) 521 (7553) (May 2015) 436–444.
[85] Yann LeCun, Yoshua Bengio, et al., Convolutional networks for images, speech, and time series, in: The Handbook of Brain Theory and Neural Networks, 1995.
[86] Yujia Li, Kevin Swersky, Richard S. Zemel, Generative moment matching networks, in: Francis R. Bach, David M. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, in: JMLR Workshop and Conference Proceedings, vol. 37, JMLR.org, 2015, pp. 1718–1727.
[87] Soon Hoe Lim, et al., Noisy recurrent neural networks, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 5124–5137.
[88] J. Lin, Divergence measures based on the Shannon entropy, IEEE Trans. Inf. Theory 37 (1) (1991) 145–151, https://doi.org/10.1109/18.61115.
[89] Kevin K. Lin, Fei Lu, Data-driven model reduction, Wiener projections, and the Koopman-Mori-Zwanzig formalism, J. Comput. Phys. 424 (2021) 109864.
[90] Jacques Louis Lions, Optimal Control of Systems Governed by Partial Differential Equations, Springer-Verlag, Berlin, 1971.
[91] Zachary C. Lipton, John Berkowitz, Charles Elkan, A critical review of recurrent neural networks for sequence learning, arXiv preprint, arXiv:1506.00019, 2015.
[92] Ljung Lennart, System Identification, Prentice Hall, 1987, ISBN 10: 0136566952 ISBN 13: 9780136566953.
[93] Ryan Lopez, Paul J. Atzberger, GD-VAEs: geometric dynamic variational autoencoders for learning nonlinear dynamics and dimension reductions, preprint, arXiv:2206.05183, 2022.
[94] Agnes Lydia, Sagayaraj Francis, Adagrad-an optimizer for stochastic gradient descent, Int. J. Inf. Comput. Sci. 6 (5) (2019) 566–568.
[95] Anton Mallasto, Guido Montùfar, Augusto Gerolin, How well do WGANs estimate the Wasserstein metric?, preprint, arXiv:1910.03875, 2019, https://doi.org/10.48550/ARXIV.1910.03875.
[96] Gisiro Maruyama, Continuous Markov processes and stochastic equations, in: Rendiconti del Circolo Matematico di Palermo, vol. 4, 1955, pp. 48–90.
[97] Igor Mezìc, Analysis of fluid flows via spectral properties of the Koopman operator, Annu. Rev. Fluid Mech. 45 (1) (2013) 357–378, https://doi.org/10.1146/annurev-fluid-011212-140652.
[98] John Miller, Moritz Hardt, Stable recurrent models, arXiv preprint, arXiv:1805.10369, 2018.
[99] G.N. Milstein, Approximate integration of stochastic differential equations, Theory Probab. Appl. 19 (3) (1975) 557–562, https://doi.org/10.1137/1119062.
[100] Alfred Muller, Integral probability metrics and their generating classes of functions, Adv. Appl. Probab. 29 (2) (Sept. 1997) 429–443, ISSN: 00018678.
[101] Kevin P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, ISBN 0262018020, 2012.
[102] John Nash, Non-cooperative games, Ann. Math. 54 (2) (1951) 286–295, ISSN: 0003486X.
[103] Jan Nygaard Nielsen, Henrik Madsen, Peter C. Young, Parameter estimation in stochastic differential equations: an overview, Annu. Rev. Control 24 (2000) 83–94.
[104] Noam Nisan, et al., Algorithmic Game Theory, Cambridge University Press, 2007.
[105] B. Oksendal, Stochastic Differential Equations: An Introduction, Springer, 2000.
[106] Martin M. Olsen, Jan Swevers, Walter Verdonck, Maximum likelihood identification of a dynamic robot model: implementation issues, Int. J. Robot. Res. 21 (2) (2002) 89–96, https://doi.org/10.1177/027836402760475379.
[107] Samuel E. Otto, Clarence W. Rowley, Koopman operators for estimation and control of dynamical systems, Annu. Rev. Control Robot. Auton. Syst. 4 (2021) 59–87.
[108] Guillermo Owen, Game Theory, Emerald Group Publishing, 2013.
[109] Shirui Pan, et al., Adversarially regularized graph autoencoder for graph embedding, in: Jérôme Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 2609–2615.
[110] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, PMLR, 2013, pp. 1310–1318.
[111] Fernando Pineda, Generalization of back propagation to recurrent and higher order neural networks, in: Neural Information Processing Systems, 1987.
[112] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, USA, Jan. 2006, p. 248.
[113] L.E. Reichl, A Modern Course in Statistical Physics, Jon Wiley and Sons Inc., 1997.
[114] Max Revay, Ian Manchester, Contracting implicit recurrent neural networks: stable models with improved trainability, in: Learning for Dynamics and Control, PMLR, 2020, pp. 393–403.
[115] Luigi M. Ricciardi, Laura Sacerdote, The Ornstein-Uhlenbeck process as a model for neuronal activity, Biol. Cybern. 35 (1) (1979) 1–9.
[116] Eitan Richardson, Yair Weiss, On GANs and GMMs, in: Bengio Samy, et al. (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada, 3–8 December 2018, 2018, pp. 5852–5863.
[117] Herbert Robbins, Sutton Monro, A stochastic approximation method, Ann. Math. Stat. 22 (3) (Sept. 1951) 400–407, https://doi.org/10.1214/aoms/1177729586.
[118] Tim Roughgarden, Algorithmic game theory, Commun. ACM 53 (7) (2010) 78–86.
[119] Sam Roweis, Zoubin Ghahramani, A unifying review of linear Gaussian models, Neural Comput. 11 (2) (1999) 305–345.
[120] Saburou Saitoh, Yoshihiro Sawano, Fundamental Properties of RKHS, Singapore, 2016.
[121] Peter J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28, https://doi.org/10.1017/S0022112010001217.
[122] Rainer Schöbel, Jianwei Zhu, Stochastic volatility with an Ornstein–Uhlenbeck process: an extension, Rev. Finance 3 (1) (1999) 23–46.
[123] Jacob Seidman, et al., NOMAD: nonlinear manifold decoders for operator learning, Adv. Neural Inf. Process. Syst. 35 (2022) 5601–5613.
[124] Jonas Sjöberg, et al., Nonlinear black-box modeling in system identification: a unified overview, Automatica 31 (12) (1995) 1691–1724.
[125] Kennan T. Smith, Primer of Modern Analysis, Springer Science & Business Media, 2012.
[126] Alexander J. Smola, A. Gretton, K. Borgwardt, Maximum mean discrepancy, in: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006: Proceedings, 2006.
[127] Bharath K. Sriperumbudur, et al., Hilbert space embeddings and metrics on probability measures, J. Mach. Learn. Res. (ISSN 1532-4435) 11 (Aug. 2010) 1517–1561.
[128] Bharath K. Sriperumbudur, et al., On integral probability metrics, ϕ-divergences and binary classification, arXiv arXiv:0901.2698 [cs.IT], 2009.
[129] J. Michael Steele, Stochastic Calculus and Financial Applications, vol. 1, Springer, 2001.
[130] Panos Stinis, et al., Enforcing constraints for interpolation and extrapolation in generative adversarial networks, J. Comput. Phys. (ISSN 0021-9991) 397 (2019) 108844, https://doi.org/10.1016/j.jcp.2019.07.042.
[131] Gilbert Strang, Computational Science and Engineering, vol. 791, Wellesley-Cambridge Press, Wellesley, 2007.
[132] Luis Enrique Sucar, Probabilistic Graphical Models, Advances in Computer Vision and Pattern Recognition, Springer London, London, 2015.
[133] Ilya Sutskever, Training Recurrent Neural Networks, University of Toronto, Toronto, ON, Canada, 2013.
[134] Gábor J. Székely, Maria L. Rizzo, Energy statistics: a class of statistics based on distances, J. Stat. Plan. Inference (ISSN 0378-3758) 143 (8) (2013) 1249–1272, https://doi.org/10.1016/j.jspi.2013.03.018.
[135] Gil Tabak, Paul J. Atzberger, Stochastic reductions for inertial fluid-structure interactions subject to thermal fluctuations, SIAM J. Appl. Math. 75 (4) (2015) 1884–1914.
[136] Daniel Soukup, Thomas Pinetz, Thomas Pock, What is optimized in Wasserstein GANs?, in: Zuzana Kukelov, Julia Skovierov (Eds.), 23rd Computer Vision Winter Workshop, 2018.
[137] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: divide the gradient by a running average of its recent magnitude, in: Slides, 2012.

[138] Nathaniel Trask, et al., GMLS-Nets: a machine learning framework for unstructured data, in: Proceedings of AAAI-MLPS, 2020.
[139] Adam P. Trischler, Gabriele M.T. D'Eleuterio, Synthesis of recurrent neural networks for dynamical system simulation, in: Neural Networks, vol. 80, 2016, pp. 67–78.
[140] Jonathan H. Tu, et al., On dynamic mode decomposition: theory and applications, J. Comput. Dyn. (2014).
[141] Belinda Tzen, Maxim Raginsky, Neural stochastic differential equations: deep latent gaussian models in the diffusion limit, arXiv preprint, arXiv:1905.09883, 2019.
[142] George E. Uhlenbeck, Leonard S. Ornstein, On the theory of the Brownian motion, Phys. Rev. 36 (5) (1930) 823.
[143] Oldrich Vasicek, An equilibrium characterization of the term structure, J. Financ. Econ. 5 (2) (1977) 177–188.
[144] Ashish Vaswani, et al., Attention is all you need, in: Isabelle Guyon, et al. (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, 2017, pp. 5998–6008.
[145] Verlet Loup, Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1 1967) 98–103, https://doi.org/10.1103/PhysRev.159.98.
[146] C. Villani, Optimal Transport: Old and New, Grundlehren der mathematischen Wissenschaften, Springer, Berlin, Heidelberg, ISBN 9783540710509, 2008.
[147] Martin J. Wainwright, Michael I. Jordan, Graphical models, exponential families, and variational inference, Found. Trends Mach. Learn. 1 (1–2) (2008) 1–305, https://doi.org/10.1561/2200000001.
[148] Eric A. Wan, Alex T. Nelson, Dual extended Kalman filter methods, in: Kalman Filtering and Neural Networks, 2001, pp. 123–173.
[149] Eric A. Wan, Rudolph Van Der Merwe, The unscented Kalman filter for nonlinear estimation, in: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373), IEEE, 2000, pp. 153–158.
[150] Larry Wasserman, All of Statistics: A Concise Course in Statistical Inference, Springer Publishing Company, Incorporated, ISBN 1441923225, 2010.
[151] Norbert Wiener, Extrapolation, Interpolation, and Smoothing of Stationary Time Series: with Engineering Applications, The MIT Press, 1949.
[152] M. Wilkinson, H.R. Kennard, A model for alignment between microscopic rods and vorticity, J. Phys. A, Math. Theor. 45 (45) (2012) 455502, https://doi.org/10.1088/1751-8113/45/45/455502.
[153] Christopher K.I. Williams, Carl Edward Rasmussen, Gaussian processes for regression, in: D.S. Touretzky, M.C. Mozer, M.E. Hasselmo (Eds.), Advances in Neural Information Processing Systems 8, MIT Press, 1996, pp. 514–520.
[154] Francis S. Wong, Time series forecasting using backpropagation neural networks, Neurocomputing 2 (4) (1991) 147–159.
[155] Bing Xu, et al., Empirical evaluation of rectified activations in convolutional network, arXiv preprint, arXiv:1505.00853, 2015.
[156] Kailai Xu, Weiqiang Zhu, Eric Darve, Learning generative neural networks with physics knowledge, Res. Math. Sci. 9 (2) (2022) 1–19.
[157] Liu Yang, Constantinos Daskalakis, George E. Karniadakis, Generative ensemble regression: learning particle dynamics from observations of ensembles with physics-informed deep generative models, SIAM J. Sci. Comput. 44 (1) (2022) B80–B99.
[158] Liu Yang, Dongkun Zhang, George Em Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, SIAM J. Sci. Comput. 42 (1) (2020) A292–A317, https://doi.org/10.1137/18M1225409.
[159] Kyongmin Yeo, Igor Melnyk, Deep learning algorithm for data-driven simulation of noisy dynamical system, J. Comput. Phys. 376 (2019) 1212–1231.
[160] O.V. Yushchenko, A. Yu Badalyan, Statistical description of the collective motion of nanoparticles, Phys. Rev. E 85 (5) (2012) 051127.
[161] Paul Zarchan, Progress in Astronautics and Aeronautics: Fundamentals of Kalman Filtering: a Practical Approach, vol. 208, AIAA, 2005.
[162] Huaguang Zhang, Zhanshan Wang, Derong Liu, A comprehensive review of stability analysis of continuous-time recurrent neural networks, IEEE Trans. Neural Netw. Learn. Syst. 25 (7) (2014) 1229–1262.
[163] Xiru Zhang, Time series analysis and prediction by neural networks, Optim. Methods Softw. 4 (2) (1994) 151–170.
[164] Junbo Jake Zhao, Michaël Mathieu, Yann LeCun, Energy-based generative adversarial networks, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.