

Coupling parameter and particle dynamics for adaptive sampling in Neural Galerkin schemes

Yuxiao Wen* Eric Vanden-Eijnden* Benjamin Peherstorfer*

June 28, 2023

Training nonlinear parametrizations such as deep neural networks to numerically approximate solutions of partial differential equations is often based on minimizing a loss that includes the residual, which is analytically available in limited settings only. At the same time, empirically estimating the training loss is challenging because residuals and related quantities can have high variance, especially for transport-dominated and high-dimensional problems that exhibit local features such as waves and coherent structures. Thus, estimators based on data samples from un-informed, uniform distributions are inefficient. This work introduces Neural Galerkin schemes that estimate the training loss with data from adaptive distributions, which are empirically represented via ensembles of particles. The ensembles are actively adapted by evolving the particles with dynamics coupled to the nonlinear parametrizations of the solution fields so that the ensembles remain informative for estimating the training loss. Numerical experiments indicate that few dynamic particles are sufficient for obtaining accurate empirical estimates of the training loss, even for problems with local features and with high-dimensional spatial domains.

1 Introduction

Nonlinear parametrizations (discretizations) that approximate solution fields of partial differential equations (PDEs) on manifolds can achieve faster error decays than traditional, linear parametrizations that are restricted to approximations in vector spaces. A widely used approach for numerically fitting nonlinear parametrizations is minimizing the norm of the PDE residual, oftentimes even globally over the time interval of interest; see [7, 29, 41, 49, 63, 70] and the early works [20, 64, 65]. However, obtaining an estimate of the norm of the residual—the empirical risk—to find parameters that minimize it can be challenging [11, 24, 67]. For example, consider a PDE that describes the transport of localized features such as phase transitions, waves, or other coherent structures across the spatial domain. For such transport-dominated problems it has been shown that the faster error decay of nonlinear parametrizations can be beneficial compared to linear approximations from an approximation-theoretic perspective [15, 16, 36, 54, 58, 60]. However, precisely for such problems, an un-informed, passive sampling with a uniform distribution over the spatial domain leads to estimators of the residual norm with high variance because the local features of the solution fields typically lead to localized residuals so that few samples fall in regions of the spatial domain where the magnitude of this residual is large and, thereby,

*Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012,
USA

informative. Equations formulated over high-dimensional spatial domains lead to similar challenges of estimating residual norms and training losses in general. The described sampling issue is a special case of the pervasive challenge in machine learning of estimating the population loss with the empirical loss from data [73].

The challenge of accurately estimating the training loss when fitting nonlinear parametrizations has been recognized in a range of works. For certain limited classes of PDEs and corresponding parametrizations, quantities that are needed for computing the training loss can be derived analytically [2, 53] and special structure can be exploited [3]. In model reduction [6, 62, 68], sampling via empirical interpolation [4, 12, 22, 30, 56, 61] is an important component of reduced models when they are formulated over nonlinear parametrizations that represent latent manifolds rather than subspaces; see [60] for a survey. Various sampling methods for empirical interpolation and related techniques are proposed in [10, 17, 18, 59] and the work [55] investigates the sample complexity in the context of nonlinear model reduction. There also is work on using deep autoencoders for dimensionality reduction, where sparse sampling methods are proposed for efficient computations [42, 66]. However, these sampling schemes aim to find an accurate set of samples in a pre-processing step, instead of actively adapting samples. In the context of high-dimensional problems, many different active learning and active data acquisition approaches to adaptively sample the temporal and spatial domains have been developed. There are works that generate a batch of uniform samples and sub-select and use the ones with highest absolute residual values for training; see, e.g., [34, 44, 52] and the survey [74]. However, the initial batch of samples has to capture already the high-residual regions of the spatial domain, which can be challenging for problems with local features in high-dimensional spatial domains. The work [72] trains a separate normalizing flow for generating samples with high residual values, which in turn requires informative training data. In [37], the time domain is split into subdomains and samples in each subdomain are drawn proportionally to its residual value, which is closely related to domain decomposition techniques. Other techniques leverage certain structures in the class of PDEs and setups of interest, such as parameterizing the gradient of the PDE solution rather than directly the solution field [26, 38], approximating committor functions [40, 47, 67], and connections to control problems [27, 46, 48]. Another line of work builds on Gaussian process regression for approximating solution fields and offers strong error analyses, which can be informative for data collection as well [5, 14].

In this work, we estimate the training loss with an ensemble of particles that we actively adapt by imposing dynamics on the particles so that they stay informative for estimating the training loss even as the PDE solution field evolves in time. The adaptation of the particles leads to an active data acquisition in the sense that the training loss is measured in a dynamic way so that few measurements and thus samples are sufficient for an accurate estimation. The starting point for us is the Dirac-Frenkel variational principle [19, 33, 45, 53] that we build on to formulate a Galerkin problem [11] that is solved sequentially in time; see also [2, 13, 23, 31, 39, 43, 69, 75]. Our sequential-in-time approach leads to a dynamical system for the parameters and this dynamical system is integrated forward in time with standard time-integration schemes to compute the parameters that represent the approximate solution field over time. Fitting the parameters sequentially in time with a dynamical system is key for deriving a dynamical system for the ensemble of particles so that it can be moved along in time and lead to an efficient training-loss estimation. It is important to note that with our approach the particle dynamics can be derived for a wide range of nonlinear parametrizations, which is in stark contrast to the previous work [11] that also proposes an active learning scheme but one that is only applicable with a limited class of nonlinear parametrizations, namely shallow neural networks with exponential units. Numerical experiments indicate that integrating forward in time the coupled parameter and particle dynamics circumvent the sampling issue even for problems with local features and with high-dimensional spatial domains.

The paper is organized as follows. In Section 2 we briefly review nonlinear approximations

methods based on the Dirac-Frenkel variational principle such as Neural Galerkin schemes and mathematically formulate the challenge of sampling to compute residual-based quantities for training. In Section 3, we derive Neural Galerkin schemes with coupled parameter and particle dynamics. Numerical results are reported in Section 4 and conclusions are drawn in Section 5.

2 Preliminaries

We briefly describe the need for nonlinear parametrizations for approximating efficiently solutions of certain classes of PDEs in Section 2.1 and discuss Neural Galerkin schemes based on the Dirac-Frenkel variational principle in Section 2.2 for numerically solving PDEs with nonlinear parametrizations. The problem formulation is given in Section 2.3, where we describe that training nonlinear parametrizations is challenging because solutions of precisely those classes of PDEs where nonlinear parametrizations are beneficial in terms of approximation theory often exhibit features that are local in the spatial domain.

2.1 The need for nonlinear parametrizations

Consider the following PDE

$$\begin{aligned} \partial_t u(t, \mathbf{x}) &= f(\mathbf{x}, u), & (t, \mathbf{x}) &\in [0, \infty) \times \mathcal{X}, \\ u(0, \mathbf{x}) &= u_0(\mathbf{x}), & \mathbf{x} &\in \mathcal{X}, \end{aligned} \quad (1)$$

where $u : [0, \infty) \times \mathcal{X} \rightarrow \mathbb{R}$ is a time-dependent field in a space \mathcal{U} over the spatial domain $\mathcal{X} \subseteq \mathbb{R}^d$ and $u_0 : \mathcal{X} \rightarrow \mathbb{R}$ is the initial condition. The right-hand side function $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ can include partial derivatives of u in the spatial coordinate \mathbf{x} . Note that f could depend on time t and it is only for ease of exposition that we skip the time dependence in the following. We consider either Dirichlet or periodic boundary conditions to make the problem (1) well posed, which means that the boundary conditions can be imposed with a suitable choice of the space \mathcal{U} so that all functions in \mathcal{U} satisfy the boundary conditions. As an alternative and for other boundary conditions, they can be imposed via penalty terms, as we demonstrate with numerical experiments in Section 4.

A typical approach in scientific computing to numerically solve PDEs such as (1) is to compute an approximate solution \tilde{u} in a finite-dimensional subspace \mathcal{U}_N of \mathcal{U} . The subspace \mathcal{U}_N is spanned by N basis functions ϕ_1, \dots, ϕ_N and the approximation \tilde{u} is a linear combination of the basis functions with N coefficients $\theta_1(t), \dots, \theta_N(t) \in \mathbb{R}$. Such an approximation is linear in the parameter $\boldsymbol{\theta}(t) = [\theta_1(t), \dots, \theta_N(t)]^T \in \mathbb{R}^N$. There are classes of PDEs where it is known that such linear approximations in subspaces can require high dimensions N . Often such problems are transport-dominated in the sense that a coherent structure such as a wave is traveling through the spatial domain. Notice that adaptive mesh refinement was introduced for efficiently approximating such problems [8, 9]. We refer to the notion of Kolmogorov N -width [15, 16, 32, 36, 54, 57, 58] and nonlinear model reduction [60] for more details about such problems. Another class of PDEs for which approximations in subspaces as described above becomes challenging is given by PDEs with high-dimensional spatial domains \mathcal{X} . Discretizing even moderately high-dimensional spatial domains \mathcal{X} with regular grid-based techniques based on linear approximations as described above becomes intractable quickly because of the curse of dimensionality.

2.2 Nonlinear parametrizations and the Dirac-Frenkel variational principle

In cases where linear approximations in subspaces lead to slow error decay, one can resort to parametrizations $\hat{u} : \mathcal{X} \times \Theta \rightarrow \mathbb{R}$ that are nonlinear in the sense that the parameter $\boldsymbol{\theta}(t) \in \Theta \subseteq \mathbb{R}^N$ enters nonlinearly in the second argument of \hat{u} . One example of such a nonlinear

parametrization is given by deep neural networks that have time-dependent weights $\boldsymbol{\theta}(t)$. In the following, we restrict ourselves to parametrizations \hat{u} that also impose the boundary conditions that accompany the PDE in (1).

We now follow [11] and build on the Dirac-Frenkel variational principle [19, 33, 53] to derive a dynamical system for $\boldsymbol{\theta}(t)$ such that the corresponding parametrization $\hat{u}(\cdot; \boldsymbol{\theta}(t))$ numerically solves the PDE in (1) over time; see also [2, 23, 43, 45, 69] for related work. The process is analogous to the classical method of lines approach for numerically solving PDEs: The PDE (1) is first discretized in space, which results in a dynamical system—system of ordinary differential equations—that is then discretized and integrated forward in time to compute a numerical solution of the PDE. Let us start with the residual of the PDE (1) at time t

$$r_t(\mathbf{x}; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) = \nabla_{\boldsymbol{\theta}} \hat{u}(\mathbf{x}; \boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t) - f(\mathbf{x}, \hat{u}(\cdot; \boldsymbol{\theta}(t))),$$

where $\dot{\boldsymbol{\theta}}(t)$ is the result of applying the chain rule to the time derivative of $\hat{u}(\cdot; \boldsymbol{\theta}(t))$. Define $\mathcal{T}_{\boldsymbol{\theta}(t)}$ to be the space spanned by the component functions of $\nabla_{\boldsymbol{\theta}} \hat{u}(\cdot; \boldsymbol{\theta}(t))$,

$$\mathcal{T}_{\boldsymbol{\theta}(t)} = \text{span}\{\partial_{\theta_1(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), \dots, \partial_{\theta_N(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t))\}.$$

The space $\mathcal{T}_{\boldsymbol{\theta}(t)}$ is the tangent space of the manifold induced by the parameterization \hat{u} at parameter $\boldsymbol{\theta}(t)$; we refer to [53] for details. We then seek $\dot{\boldsymbol{\theta}}(t)$ such that it leads to a residual that is orthogonal in the following sense

$$\langle \partial_{\theta_i(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) \rangle_{\nu} = 0, \quad i = 1, \dots, N, \quad (2)$$

where ν is a measure that is fully supported on \mathcal{X} and $\langle \cdot, \cdot \rangle_{\nu}$ denotes the corresponding L^2 inner product. After transformations, the system of equations (2) can be represented as

$$\mathbf{M}(\boldsymbol{\theta}(t)) \dot{\boldsymbol{\theta}}(t) = \mathbf{F}(\boldsymbol{\theta}(t)), \quad (3)$$

with the operators defined as

$$[\mathbf{M}(\boldsymbol{\theta}(t))]_{ij} = \langle \nabla_{\theta_i(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), \nabla_{\theta_j(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)) \rangle_{\nu}, \quad i, j = 1, \dots, N,$$

and

$$[\mathbf{F}(\boldsymbol{\theta}(t))]_i = \langle \nabla_{\theta_i(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), f(\cdot; \hat{u}(\cdot; \boldsymbol{\theta}(t))) \rangle_{\nu}, \quad i = 1, \dots, N.$$

Thus, to obtain a numerical solution $\hat{u}(\cdot; \boldsymbol{\theta}(t))$ of the equation (1), the dynamical system (3) is integrated forward in time with a numerical time-integration scheme.

2.3 Problem formulation

For a limited class of PDEs and corresponding specific parametrizations, the operators \mathbf{M} and \mathbf{F} of the dynamical system shown in (3) can be derived analytically, which is demonstrated in [2, 45, 53] in the context of time-dependent nonlinear parametrizations. In many cases, however, the operators \mathbf{M} and \mathbf{F} need to be estimated numerically. A common approach, especially if the spatial domain \mathcal{X} is higher dimensional, is to resort to Monte Carlo estimation from m samples $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathcal{X}$ of the distribution ν . However, Monte Carlo estimators based on un-informed sampling can have a high variance, for example if the solution u has local features in the spatial domain and these local features are transported over time. Notice that this is a problem where nonlinear parametrizations can achieve a drastically faster error decay than linear parametrizations as discussed in the survey in [60] and the introduction in Section 1. To see the sampling issue for such a problem, consider Figure 1a that shows an advecting wave governed by the linear advection equation in one spatial dimension. By plotting the wave dynamics in the time-space domain in Figure 1b, it can be seen that an un-informed sampling such as a uniform sampling in the time-space domain is inefficient in the sense that only few samples will be in

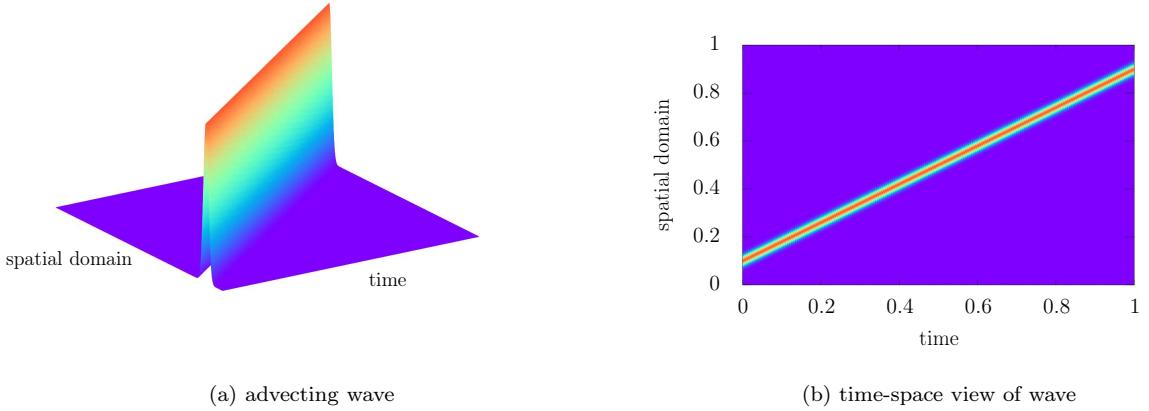


Figure 1: Features of solutions of transport-dominated problems are often local in the spatial domain. The spatial locality makes un-informed sampling such as uniform sampling inefficient for estimating the training loss of fitting nonlinear parametrizations, which is analogous to the challenge of estimating the operators \mathbf{M} and \mathbf{F} of the system (3) of our sequential-in-time approach.

regions where the solution is non-zero. While this is a toy example that is meant to demonstrate the challenge of estimating the operators of the system (3), it is representative for a wide range of transport-dominated problems that have local features in the spatial domain. We stress that similar observations hold for global time-space collocation approaches, where an un-informed sampling can require a large number of samples to accurately estimate the residual norm over the time-space domain.

3 Neural Galerkin schemes with coupled parameter and particle dynamics

We introduce Neural Galerkin schemes with dynamic particles. The particles empirically represent time-dependent measures that are coupled to the parameters $\boldsymbol{\theta}(t)$ that represent numerical approximations of the PDE solution fields. A coupled dynamical system integrates forward in time particles and parameters together such that few particles are sufficient for accurately estimating the inner products with the residual in the Dirac-Frenkel variational problem (2) and the parameters provide an approximation of the solution field in the Dirac-Frenkel variational sense; see Figure 2.

This section is structured as follows. Section 3.1 and Section 3.2 introduce Neural Galerkin schemes with projections that rely on time-dependent measures. The dynamics for the particles that approximate the time-dependent measure are derived in Section 3.3, which then leads to the computational procedure of Neural Galerkin schemes with dynamic particles in Section 3.4. The computational procedure is summarized in algorithmic form in Section 3.5.

3.1 Projections with time-dependent inner products

Instead of projecting with respect to the inner product corresponding to a fixed measure ν as in (2), we introduce a time-dependent inner product $\langle \cdot, \cdot \rangle_{\mu_t}$ that is adapted over time with the time-dependent measure μ_t . The projection with respect to μ_t is

$$\langle \partial_{\theta_i(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) \rangle_{\mu_t} = 0, \quad i = 1, \dots, N, \quad (4)$$

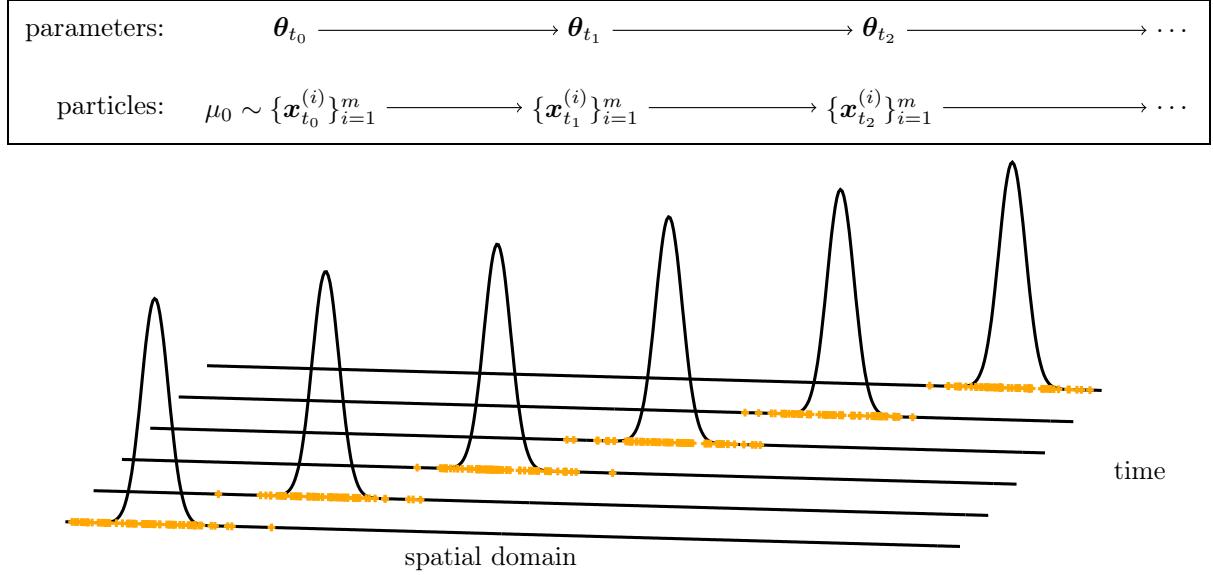


Figure 2: The proposed Neural Galerkin schemes with dynamic particles couple the parameters $\boldsymbol{\theta}(t)$ given by the Dirac-Frenkel variational principle with the ensemble of particles $\{\mathbf{x}_t^{(i)}\}_{i=1}^m$ so that few particles are sufficient for accurately estimating the operators \mathbf{M}_t and \mathbf{F}_t of the system (5), even for problems with high-dimensional spatial domains and when there are local features that travel over time such as waves and other coherent structures.

with the corresponding $\mathbf{M}_t(\boldsymbol{\theta}(t))$ and $\mathbf{F}_t(\boldsymbol{\theta}(t))$ at time t given by

$$[\mathbf{M}_t(\boldsymbol{\theta}(t))]_{i,j} = \langle \nabla_{\theta_i(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), \nabla_{\theta_j(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)) \rangle_{\mu_t}, \quad i, j = 1, \dots, N,$$

and

$$[\mathbf{F}_t(\boldsymbol{\theta}(t))]_i = \langle \nabla_{\theta_i(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), f(\cdot; \hat{u}(\cdot; \boldsymbol{\theta}(t))) \rangle_{\mu_t}, \quad i = 1, \dots, N.$$

Notice that \mathbf{M}_t and \mathbf{F}_t now depend on time and lead to the Neural Galerkin system

$$\mathbf{M}_t(\boldsymbol{\theta}(t)) \dot{\boldsymbol{\theta}}(t) = \mathbf{F}_t(\boldsymbol{\theta}(t)). \quad (5)$$

The aim is to adapt the measure μ_t over time t so that the inner product (4) can be numerically estimated efficiently. If the nonlinear parametrization $\hat{u}(\cdot; \boldsymbol{\theta}(t))$ is so expressive that the component functions $\partial_{\theta_1(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t)), \dots, \partial_{\theta_N(t)} \hat{u}(\cdot; \boldsymbol{\theta}(t))$ of the gradient of $\hat{u}(\cdot; \boldsymbol{\theta}(t))$ span an N -dimensional space and there exists $\boldsymbol{\theta}(t)$ and $\dot{\boldsymbol{\theta}}(t)$ with norm $|\langle r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)), r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) \rangle_{\mu_t}| = \|r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))\|_{\mu_t}^2 = 0$, then the corresponding $\boldsymbol{\theta}(t)$ and $\dot{\boldsymbol{\theta}}(t)$ will lead to a zero residual norm with respect to any other measure that has full support over \mathcal{X} . Furthermore, under the expressiveness condition stated above, solving (4) for any measure μ_t implies that the residual norm is zero for any other measure. Thus, under these idealized conditions, we are free to choose a measure μ_t that is computationally convenient. In the following, we will select μ_t such that the inner product (4) can be accurately estimated with few samples. Notice that even computing the inner product in (4) with an approximation of μ_t is sufficient to obtain a zero residual norm with respect to the actual μ_t . In numerical practice, the residual norm is typically not exactly zero; however, as long as the residual norm is kept small, similar arguments hold at least heuristically and provide a motivation for the following.

3.2 Selecting the time-dependent measure

The Neural Galerkin schemes proposed in this work adapt the space $\mathcal{T}_{\boldsymbol{\theta}(t)}$ over time following the Dirac-Frenkel variational principle as discussed in Section 2.2 and additionally adapt the

inner product $\langle \cdot, \cdot \rangle_{\mu_t}$ with respect to which the residual is set to zero.

To this end, we will consider schemes that use an inner-product with respect to evolving measures μ_t which track Gibbs measures of the form

$$\mu_t^G(dx) = Z_{\theta(t), \dot{\theta}(t)}^{-1} \exp(-V_{\theta(t), \dot{\theta}(t)}(x)) dx, \quad Z_{\theta(t), \dot{\theta}(t)} = \int_{\mathcal{X}} \exp(-V_{\theta(t), \dot{\theta}(t)}(x)) dx, \quad (6)$$

where the potential $V_{\theta(t), \dot{\theta}(t)}$ depends on the parameter $\theta(t)$ and its time derivative $\dot{\theta}(t)$. These Gibbs measures are general enough for our purpose.

There are multiple options for choosing μ_t^G . One option is setting

$$\mu_t^G \propto |r_t(\cdot; \theta(t), \dot{\theta}(t))|^{2\gamma} \nu(\cdot)^\gamma, \quad (7)$$

where $\gamma > 0$ is a tempering parameter. By construction, the samples from the measure μ_t^G defined in (7) are concentrated where the squared residual is high and thus typically few samples are sufficient to accurately estimate the inner products for the projections in (4).

Another option is

$$\mu_t^G \propto |\hat{u}(\cdot; \theta(t))|^\gamma \nu(\cdot)^\gamma, \quad (8)$$

with tempering parameter $\gamma > 0$, which means that samples are drawn in regions of the spatial domain where the numerical solution \hat{u} has large values. We will later consider the Fokker-Planck equation where the solution \hat{u} is approximating a probability density function and thus sampling proportional to it leads to efficient estimators of the Neural Galerkin operators M_t and F_t of (5).

3.3 Coupled system for the concurrent evolution of the parameters and the time-dependent measure

There are several possibilities to concurrently evolve an approximation μ_t of the measure μ_t^G and parameters $\theta(t)$ corresponding to the solution field evolve.

3.3.1 Langevin dynamics

A first option is to evolve the parameters $\theta(t)$ and measure μ_t concurrently via the system of equations

$$\begin{aligned} M_t(\theta(t))\dot{\theta}(t) &= F_t(\theta(t)), \\ \partial_t \mu_t &= \alpha \nabla \cdot (\nabla \mu_t + \mu_t \nabla V_{\theta(t), \dot{\theta}(t)}), \end{aligned} \quad (9)$$

where $\alpha > 0$ is a parameter that controls the separation of timescale between the evolution of the parameters $\theta(t)$ and the measure μ_t . In the limit as $\alpha \rightarrow \infty$, the measure evolves infinitely faster than the parameters, and since the equation for μ_t in (9) is the gradient flow in Wasserstein-2 metric over the Kullback-Leibler divergence of μ_t from the Gibbs measure associated with $V_{\theta(t), \dot{\theta}(t)}$, it realizes (6) at all times. However, it is important to note that (9) leads to zero-residual dynamics for any $\alpha > 0$, as long as the nonlinear parametrization of the solution is expressive enough, as discussed at the end of Section 3.1. In practice, (9) can be implemented by approximating the distribution μ_t by its empirical distribution over m particles $x^{(1)}(t), \dots, x^{(m)}(t)$ as

$$\hat{\mu}_t = \frac{1}{m} \sum_{i=1}^m \delta_{x^{(i)}(t)}. \quad (10)$$

This leads to the coupled system

$$\begin{aligned} \hat{M}_t(\theta(t))\dot{\theta}(t) &= \hat{F}_t(\theta(t)), \\ dx^{(i)}(t) &= -\alpha \nabla V_{\theta(t), \dot{\theta}(t)}(x^{(i)}(t)) dt + \sqrt{2\alpha} dW^{(i)}(t), \quad i = 1, \dots, m, \end{aligned} \quad (11)$$

where $\{W^{(i)}(t)\}_{i=1}^m$ are m independent Wiener processes in \mathbb{R}^d and the particles $\mathbf{x}^{(1)}(t), \dots, \mathbf{x}^{(m)}(t)$ are used to estimate \mathbf{M}_t and \mathbf{F}_t as

$$\begin{aligned} [\hat{\mathbf{M}}_t(\boldsymbol{\theta}(t))]_{l,j} &= \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_l(t)} \hat{u}(\mathbf{x}^{(i)}(t); \boldsymbol{\theta}(t)) \nabla_{\theta_j(t)} \hat{u}(\mathbf{x}^{(i)}(t); \boldsymbol{\theta}(t)), \quad l, j = 1, \dots, N, \\ [\hat{\mathbf{F}}_t(\boldsymbol{\theta}(t))]_j &= \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_j(t)} \hat{u}(\mathbf{x}^{(i)}(t); \boldsymbol{\theta}(t)) f(\mathbf{x}^{(i)}(t); \hat{u}(\cdot; \boldsymbol{\theta}(t))), \quad j = 1, \dots, N. \end{aligned} \quad (12)$$

3.3.2 Stein variational gradient descent

We can also use Stein variational gradient descent (SVGD) [50] to derive dynamical systems for the particles. The SVGD method deterministically approximates the gradient flow (9) in a reproducing kernel Hilbert space with kernel $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. Approximating the gradient flow in an RKHS can be beneficial when the gradients of the potential V are noisy, which is typically the case when taking the gradient of the potential involves differentiating a neural-network approximation.

For using SVGD, we replace the equation for the measure μ_t in (9) with

$$\partial_t \mu_t = \alpha \nabla \cdot \left(\mu_t \mathbb{E}_{\mathbf{x}' \sim \mu_t} [\mathcal{K}(\mathbf{x}', \mathbf{x}) \nabla V_{\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)}(\mathbf{x}') - \nabla_1 \mathcal{K}(\mathbf{x}', \mathbf{x})] \right), \quad (13)$$

where we denote with $\nabla_1 \mathcal{K}$ the gradient of \mathcal{K} in its first argument. In practice, we approximate μ_t with its empirical distribution as in (10) and obtain from (13) the particle dynamics

$$\frac{d}{dt} \mathbf{x}^{(i)}(t) = \frac{\alpha}{m} \sum_{l=1}^m [\nabla_1 \mathcal{K}(\mathbf{x}^{(l)}(t), \mathbf{x}^{(i)}(t)) - \mathcal{K}(\mathbf{x}^{(l)}(t), \mathbf{x}^{(i)}(t)) \nabla V_{\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)}(\mathbf{x}^{(l)}(t))], \quad i = 1, \dots, m. \quad (14)$$

The choice of the kernel \mathcal{K} plays an important role in SVGD in theory and practice. In particular, a poor choice of the kernel can prevent convergence of (13) to the Gibbs measure (6) with potential V [51] and even a failure of detecting non-convergence in practical settings [35]. However, as stated in Section 3.1, if the nonlinear parametrization is expressive enough, it is sufficient in our case to compute the projections in (4) with respect to an approximation of the measure μ_t^G .

3.4 Scheme for concurrently evolving parameter and particle dynamics

We now propose a numerical scheme to integrate the coupled system of parameter dynamics (5) and particle dynamics given by, e.g., SVGD as in (14). Motivated by the heterogeneous multiscale method [1, 25, 28], we propose to move forward the ensemble of particles $\{\mathbf{x}^{(i)}(t)\}_{i=1}^m$ with a faster time scale than the parameters $\boldsymbol{\theta}(t)$ over time t given by the PDE (1).

3.4.1 Discretization in time

We discretize the dynamical system (5) of the parameters $\boldsymbol{\theta}(t)$ in time. Let $\delta t_k > 0$ be the time-step size at time step $k = 0, 1, 2, \dots$ so that at time step $k+1$ the time is $t_{k+1} = t_k + \delta t_k$. Correspondingly, let $\Delta \boldsymbol{\theta}_k$ be the update that is applied at time step k to obtain $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \delta t_k \Delta \boldsymbol{\theta}_k$. The initial parameter is $\boldsymbol{\theta}_0$, which is obtained by fitting $\hat{u}(\cdot; \boldsymbol{\theta}_0)$ to the initial condition u_0 .

The update $\Delta \boldsymbol{\theta}_k$ at time step k is obtained by solving the time-discrete system

$$\hat{\mathbf{M}}_k(\boldsymbol{\theta}_k, \Delta \boldsymbol{\theta}_k) \Delta \boldsymbol{\theta}_k = \hat{\mathbf{F}}_k(\boldsymbol{\theta}_k, \Delta \boldsymbol{\theta}_k), \quad (15)$$

which is obtain after discretizing the time-continuous system (5) and estimating the operators with Monte Carlo over the particles $\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(m)}$ at time step k analogous to (12). Notice

that (15) describes potentially nonlinear equations in $\Delta\boldsymbol{\theta}_k$. For example, if the time-continuous system (5) is discretized with implicit time integration schemes such as the backward Euler method, then $\hat{\mathbf{M}}_k$ and $\hat{\mathbf{F}}_k$ can depend nonlinearly on $\Delta\boldsymbol{\theta}_k$. In contrast, for explicit schemes, system (15) is linear in $\Delta\boldsymbol{\theta}_k$; we refer to [11] for more details.

To keep the notation manageable, we focus on the Runge-Kutta 4 (RK4) scheme [21] here. The corresponding update is given by

$$\Delta\boldsymbol{\theta}_k = \frac{1}{6} \left(\Delta\boldsymbol{\theta}_k^{(1)} + 2\Delta\boldsymbol{\theta}_k^{(2)} + 2\Delta\boldsymbol{\theta}_k^{(3)} + \Delta\boldsymbol{\theta}_k^{(4)} \right),$$

where

$$\begin{aligned} \hat{\mathbf{M}}_k(\boldsymbol{\theta}_k, \Delta\boldsymbol{\theta}_k^{(1)}) \Delta\boldsymbol{\theta}_k^{(1)} &= \hat{\mathbf{F}}_k(\boldsymbol{\theta}_k, \Delta\boldsymbol{\theta}_k^{(1)}), \\ \hat{\mathbf{M}}_k\left(\boldsymbol{\theta}_k + \frac{\delta t_k}{2}\Delta\boldsymbol{\theta}_k^{(1)}, \Delta\boldsymbol{\theta}_k^{(2)}\right) \Delta\boldsymbol{\theta}_k^{(2)} &= \hat{\mathbf{F}}_k\left(\boldsymbol{\theta}_k + \frac{\delta t_k}{2}\Delta\boldsymbol{\theta}_k^{(1)}, \Delta\boldsymbol{\theta}_k^{(2)}\right), \\ \hat{\mathbf{M}}_k\left(\boldsymbol{\theta}_k + \frac{\delta t_k}{2}\Delta\boldsymbol{\theta}_k^{(2)}, \Delta\boldsymbol{\theta}_k^{(3)}\right) \Delta\boldsymbol{\theta}_k^{(3)} &= \hat{\mathbf{F}}_k\left(\boldsymbol{\theta}_k + \frac{\delta t_k}{2}\Delta\boldsymbol{\theta}_k^{(2)}, \Delta\boldsymbol{\theta}_k^{(3)}\right), \\ \hat{\mathbf{M}}_k\left(\boldsymbol{\theta}_k + \delta t_k\Delta\boldsymbol{\theta}_k^{(3)}, \Delta\boldsymbol{\theta}_k^{(4)}\right) \Delta\boldsymbol{\theta}_k^{(4)} &= \hat{\mathbf{F}}_k\left(\boldsymbol{\theta}_k + \delta t_k\Delta\boldsymbol{\theta}_k^{(3)}, \Delta\boldsymbol{\theta}_k^{(4)}\right). \end{aligned} \quad (16)$$

The estimated matrices are

$$\begin{aligned} [\hat{\mathbf{M}}_k(\boldsymbol{\theta}, \Delta\boldsymbol{\theta})]_{l,j} &= \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_l} \hat{u}(\mathbf{x}_k^{(i)}; \boldsymbol{\theta}) \nabla_{\theta_i} \hat{u}(\mathbf{x}_k^{(i)}; \boldsymbol{\theta}), \quad l, j = 1, \dots, N, \\ [\hat{\mathbf{F}}_k(\boldsymbol{\theta}, \Delta\boldsymbol{\theta})]_j &= \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_j} \hat{u}(\mathbf{x}_k^{(i)}; \boldsymbol{\theta}) f(\mathbf{x}_k^{(i)}, \hat{u}(\cdot; \boldsymbol{\theta})), \quad j = 1, \dots, N, \end{aligned} \quad (17)$$

with the particles $\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(m)}$ at time step k . Notice that $\Delta\boldsymbol{\theta}$ only formally enters in (17) but does not change $\hat{\mathbf{M}}_k$ and $\hat{\mathbf{F}}_k$ and thus the four equations in (16) are linear in the updates $\Delta\boldsymbol{\theta}_k^{(1)}, \dots, \Delta\boldsymbol{\theta}_k^{(4)}$. This means that at each time step with RK4, four systems of linear equations are solved.

3.4.2 Predictor-corrector scheme to update particles

To obtain the particles $\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(n)}$ for computing (17) at time step k , we first take a predictor step because the potential $V_{\boldsymbol{\theta}_k, \Delta\boldsymbol{\theta}_k}$ at time step k depends on $\boldsymbol{\theta}_k$ and $\Delta\boldsymbol{\theta}_k$, which is unavailable. With a predictor step we compute $\Delta\boldsymbol{\theta}_k^{(P)}$ as an approximation of $\Delta\boldsymbol{\theta}_k$ to use in adapting the particles. The predictor system is obtained with a forward Euler discretization of (5), so that

$$\begin{aligned} [\hat{\mathbf{M}}_k^{(P)}]_{l,j} &= \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_l} \hat{u}(\mathbf{x}_{k-1}^{(i)}; \boldsymbol{\theta}_k) \nabla_{\theta_j} \hat{u}(\mathbf{x}_{k-1}^{(i)}; \boldsymbol{\theta}_k), \quad l, j = 1, \dots, N, \\ [\hat{\mathbf{F}}_k^{(P)}]_j &= \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_j} \hat{u}(\mathbf{x}_{k-1}^{(i)}; \boldsymbol{\theta}_k) f(\mathbf{x}_{k-1}^{(i)}, \hat{u}(\cdot; \boldsymbol{\theta}_k)), \quad j = 1, \dots, N, \end{aligned} \quad (18)$$

which leads to a linear system of equations in $\Delta\boldsymbol{\theta}_k^{(P)}$. Notice that the estimates $\hat{\mathbf{M}}_k^{(P)}$ and $\hat{\mathbf{F}}_k^{(P)}$ are based on the particles $\mathbf{x}_{k-1}^{(1)}, \dots, \mathbf{x}_{k-1}^{(m)}$ from time step $k-1$.

We use the result of the predictor $\Delta\boldsymbol{\theta}_k^{(P)}$ to define the potential $V_{\boldsymbol{\theta}_k, \Delta\boldsymbol{\theta}_k^{(P)}}$, which is then used to update the particles as described in Section 3.3. The initial distribution is the empirical measure $\hat{\mu}_{k-1}$ from the previous time step $k-1$. Integrating the particle dynamics forward in time with a time-step size $\delta\tau > 0$ until particle end time gives the particles at time step k ,

$$\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(m)}.$$

Algorithm 1 Coupling parameter and particle dynamics in Neural Galerkin schemes

Input: $\{\mathbf{x}_0^{(i)}\}_{i=1}^m, \boldsymbol{\theta}_0, K, \delta t$

for $k \leftarrow 1$ to K **do**

- $\Delta\boldsymbol{\theta}_k^{(P)} \leftarrow$ predictor step (18) with the operators estimated with ensemble $\{\mathbf{x}_{k-1}^{(i)}\}_{i=1}^m$
- $\{\mathbf{x}_k^{(i)}\}_{i=1}^m \leftarrow$ initialize with $\{\mathbf{x}_{k-1}^{(i)}\}_{i=1}^m$ from time step $k-1$ and then update as in Section 3.3
- $\hat{\mathbf{M}}_k, \hat{\mathbf{F}}_k \leftarrow$ Estimate the operators with ensemble $\{\mathbf{x}_k^{(i)}\}_{i=1}^m$
- $\Delta\boldsymbol{\theta}_k \leftarrow$ take a time step of system (15) with the estimated operators $\hat{\mathbf{M}}_k$ and $\hat{\mathbf{F}}_k$
- $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \delta t \Delta\boldsymbol{\theta}_k$ update parameters

end for

The time-step size $\delta\tau = \delta t/\alpha$ with which the particles are integrated in time is controlled by the parameter α of equations (11) and (14), respectively. The particles $\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(m)}$ are then used to compute (17) and solve for $\Delta\boldsymbol{\theta}_k$ to obtain $\boldsymbol{\theta}_{k+1}$. This process is repeated for all time steps $k = 1, 2, 3, \dots$.

3.5 Computational procedure

We summarize the computational procedure of Neural Galerkin schemes with dynamic particles in Algorithm 1. The procedure requires as inputs the parameter $\boldsymbol{\theta}_0$ that is obtained by fitting $\hat{u}(\cdot; \boldsymbol{\theta}_0)$ to the initial condition u_0 of the PDE (1). Other inputs are the number of time steps K and the time-step size δt . For simplicity, we use a fixed time step size $\delta t = \delta t_k$ for $k = 1, \dots, K$. Additionally, an initial ensemble of particles $\{\mathbf{x}_0^{(i)}\}_{i=1}^m$ is required, which can be obtained by sampling from the distribution that is proportional to the absolute initial condition $|u_0|$.

The procedure consists of a loop over the time steps $k = 1, \dots, K$ corresponding to the physical time t and a nested loop to integrate the particle dynamics over time τ as described in Section 3.3. At each time step $k = 1, \dots, K$, the predictor step gives $\Delta\boldsymbol{\theta}_k^{(P)}$, which is then used to compute the potential $V_{\boldsymbol{\theta}_k, \Delta\boldsymbol{\theta}_k^{(P)}}$ and its gradient for updating the particles. The particles are initialized at time step k with the ensemble of particles $\{\mathbf{x}_{k-1}^{(i)}\}_{i=1}^m$ from the previous time step $k-1$ and then updated to the ensemble of particles $\{\mathbf{x}_k^{(i)}\}_{i=1}^m$ at time k . The particles $\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(m)}$ are then used to estimate $\hat{\mathbf{M}}_k$ and $\hat{\mathbf{F}}_k$ to assemble the system (15), which is then integrated forward for one time step to obtain the update $\Delta\boldsymbol{\theta}_k$ and ultimately $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_k$.

4 Numerical experiments

In this section, we demonstrate Neural Galerkin schemes with coupled particles dynamics on three numerical examples. We start with the one-dimensional Korteweg-de Vries (KdV) equation, for which we plot the particle dynamics for demonstration purposes. We then consider transport equations and the Fokker-Planck equation in moderately high dimensions. We will demonstrate the effectiveness of our adaptive sampling scheme with dynamic particles by comparing it to sampling from a measure ν that is fixed over time. In all three examples, ν is chosen to be the uniform measure over the spatial domain \mathcal{X} .

4.1 Korteweg-de Vries equation (KdV)

Consider the KdV equation

$$\partial_t u(t, x) + \partial_x^3 u(t, x) + 6u(t, x)\partial_x u(t, x) = 0, \quad (t, x) \in [0, 6] \times \mathcal{X}, \quad (19)$$

over the spatial domain $\mathcal{X} = [-20, 40]$ and time range $t \in [0, 6]$. We consider the same initial condition u_0 as in [71] for which an analytic solution to (19) is available. The solution consists

of two solitons that approach each other, merge, and then separate again. We approximate the solution by imposing Dirichlet boundary conditions on (19) via a penalty residual on the left $x_l = -20$ and right $x_r = 40$ boundary point. Consider the residual at the boundary

$$r_t^\partial(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) = \nabla_{\boldsymbol{\theta}} \hat{u}(\cdot; \boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t) - g(\cdot; \boldsymbol{\theta}(t)),$$

where the function $g \equiv 0$ is constant zero so that we penalize any deviation over time from the value of initial condition at the boundary points. Then the residual that we project following (4) is

$$\bar{r}_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) = r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) + \zeta(r_t^\partial(x_l; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)) + r_t^\partial(x_r; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)))$$

where $\zeta > 0$ is an adjustable weight we put on the boundary condition penalty. In this experiment, we set it to $\zeta = 10^4$ and note that in later examples we consider a different way of imposing boundary conditions that does not require setting an adjustable weight. For the nonlinear parameterization \hat{u} , we use a fully connected feed-forward network with two hidden layers, five nodes per layer, and sigmoid activation functions. The total number of parameters is $N = 45$. We use the RK4 time integration scheme with a fixed time-step size $\delta t = 10^{-4}$.

We compare our adaptive sampling scheme with dynamic particles to a static sampling that keeps the distribution fixed as the uniform distribution over the spatial domain \mathcal{X} . Our target measure is set to $\mu_t \propto |r_t(\cdot; \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))|^2 \nu(\cdot)$ and the number of particles is $m = 100$, which is the same number of samples drawn from the uniform measure at each time t . Particles dynamics are imposed via SVGD as described in Section 3.3.2. The kernel is the Gaussian kernel with bandwidth 0.05, the step size for SVGD is 0.05, and the number of SVGD steps is 500 at each physical time t . The tempering parameter is set to $\gamma = 0.25$. SVGD particles are initialized proportional to the fitted initial solution $\hat{u}(\cdot; \boldsymbol{\theta}_0)$ at $t = 0$.

Figure 3 compares the approximation of the solution obtained with Neural Galerkin schemes with dynamic particles to the approximation obtained with static sampling. Dynamically moving the particles leads to a good approximation of the solution that captures the local features of this problem. In contrast, the accuracy of the approximation based on uniform samples quickly deteriorates. A quantitative comparison between Neural Galerkin schemes with dynamic particles versus static approaches is shown in Figure 4. The relative L^2 error is computed with respect to the analytic solution over the spatial domain \mathcal{X} . As shown in Figure 4, Neural Galerkin schemes with dynamic particles lead to orders of magnitude lower errors than static sampling.

4.2 High-dimensional transport equations

Consider the advection equation

$$\partial_t u(t, \mathbf{x}) + \mathbf{a}(t) \cdot \nabla_{\mathbf{x}} u(t, \mathbf{x}) = 0, \quad u(0, \mathbf{x}) = u_0(\mathbf{x}),$$

over the five-dimensional spatial domain $\mathcal{X} = [0, 10]^5$ for time range $t \in [0, 1.2]$. The time-dependent transport coefficient is

$$\mathbf{a}(t) = \mathbf{c} \odot (\sin(\pi t \mathbf{a}_d) + 5/4),$$

where $\mathbf{c} = [1, 2, \dots, d]^T$, $\mathbf{a}_d = 2 + \frac{2}{d}[0, 1, \dots, d-1]^T$, and \odot denotes element-wise multiplication. The initial condition u_0 is a mixture of two non-isotropic Gaussian waves with means

$$\boldsymbol{\mu}_1 = \frac{11}{10} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \frac{3}{4} \begin{bmatrix} 1.5 - (-1)^1/(d+1) \\ 1.5 - (-1)^2/(d+1) \\ \vdots \\ 1.5 - (-1)^d/(d+1) \end{bmatrix},$$

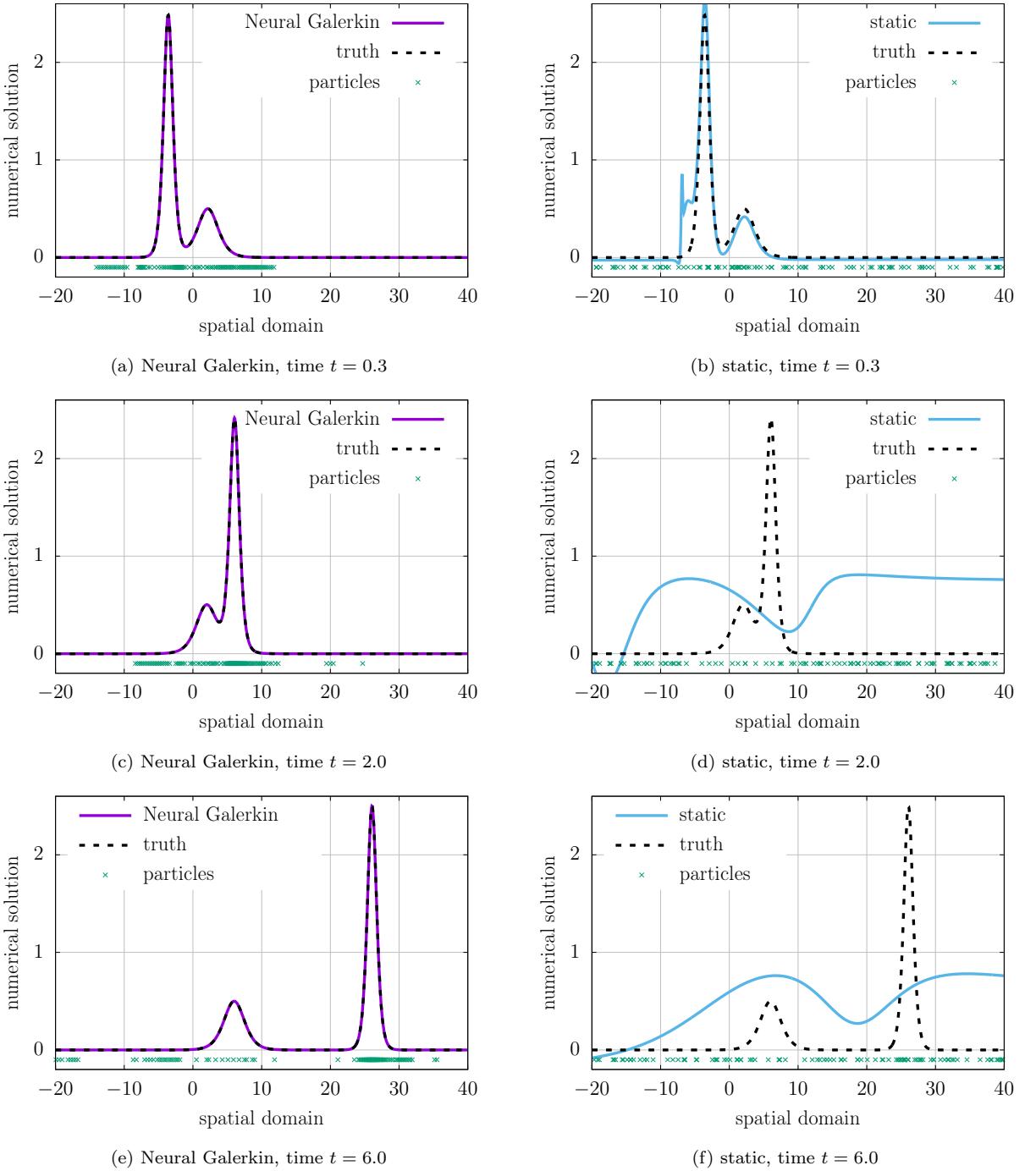


Figure 3: KdV: Neural Galerkin schemes with dynamic particles (left column) achieves an accurate approximation of the solution field with only $m = 100$ particles, whereas static sampling (right column) with $m = 100$ samples leads to inaccurate approximations after only a few time steps.

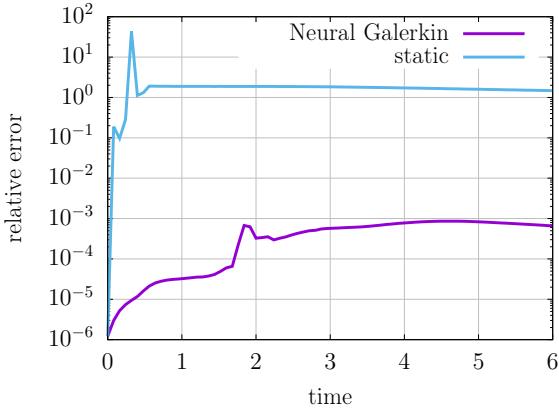


Figure 4: KdV: Neural Galerkin schemes with dynamic particles lead to orders of magnitude lower errors than approximations based on static sampling in this example.

and covariance matrices

$$\Sigma_1 = \frac{1}{200} \begin{bmatrix} 2 & & & \\ & 4 & & \\ & & \ddots & \\ & & & 2d \end{bmatrix}, \quad \Sigma_2 = \frac{1}{200} \begin{bmatrix} d+1 & & & \\ & d & & \\ & & \ddots & \\ & & & 2 \end{bmatrix}.$$

We enforce that the solution is zero at the origin with a penalty term that is applied similarly as described in Section 4.1. The coefficient of the penalty term is $\zeta = 10^2$ in this example. The analytic solution of this equation can be derived via the method of characteristics; see [11] for details. The analytic solution will serve as benchmark in this example. For the nonlinear parameterization, we use a fully connected feed-forward network with two hidden layers, fifteen nodes per layer, and sigmoid activation functions. The number of parameters is $N = 345$. We use the RK4 time integration scheme with a fixed time-step size $\delta t = 10^{-3}$.

We compare Neural Galerkin with dynamic particles to uniform sampling with $m = 2500$ samples over the five-dimensional domain \mathcal{X} . The target measure μ_t is proportional to the squared residual as in Section 4.1. In this example, we take 300 SVGD steps at each time t and set the kernel bandwidth and the SVGD step size to 0.1. The tempering parameter is set to $\gamma = 0.25$.

Figure 5 compares the analytic solution to the approximate solutions obtained with dynamic particles and with static sampling and $m = 2500$ samples. The figure shows the marginals

$$u_i^{\text{marg}}(t, x) = \int_0^{10} \cdots \int_0^{10} u(t, x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_d) dx_1 \dots dx_{i-1} dx_{i+1} \dots dx_d$$

for dimensions $i = 1, \dots, d$ of the analytic solution, the numerical solution obtained with static sampling, and the numerical solution obtained with Neural Galerkin and dynamic particles. The results shown in Figure 5 indicate that 2500 uniform samples over the five-dimensional spatial domain \mathcal{X} are insufficient to capture the local features and so lead to a poor approximation of the analytic solution. In contrast, if we dynamically adapt the particles over time, we obtain an approximation that is in close agreement with the analytic solution in this example.

4.3 Non-Gaussian interaction systems described by the Fokker-Planck equation

We consider a system of d interacting physical particles in a bounded domain $\mathcal{X} = [-3, 11]^d \subset \mathbb{R}^d$ with positions $X_1(t), \dots, X_d(t)$ described by the stochastic differential equation (SDE)

$$dX_i = f_g(t, X_i)dt + \sum_{j=1}^d f_K(X_i, X_j)dt + \sqrt{2D}dW_i, \quad \text{for } i = 1, \dots, d, \quad (20)$$

with one-body force $f_g : [0, \infty) \times \mathbb{R} \rightarrow \mathbb{R}$, a pairwise interaction term $f_K : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, the diffusion coefficient $D > 0$, and independent Wiener processes W_i .

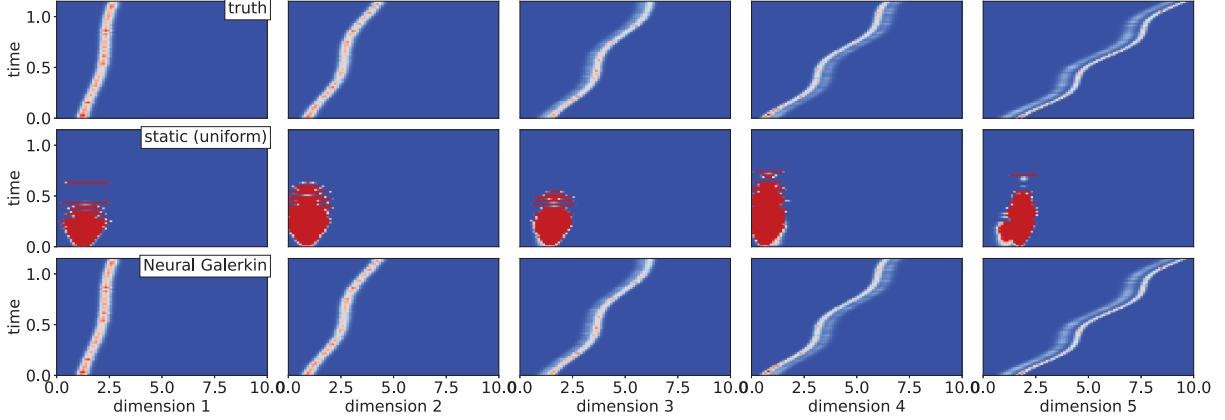


Figure 5: High-dimensional advection: This figure shows the marginals of the analytic solution (truth) and of the numerical solutions obtained with static sampling (uniform) and dynamic particles (Neural Galerkin). With only $m = 2500$ samples, Neural Galerkin with dynamic particles can predict well the local features of the solution in this moderately high-dimensional example. In contrast, static sampling based on the uniform distribution fails to provide meaningful predictions.

Fokker-Planck equation The joint density of the positions $X_1(t), \dots, X_d(t)$ is governed by the Fokker-Planck equation with homogeneous Dirichlet boundary conditions,

$$\partial_t u(t, \mathbf{x}) = \sum_{i=1}^d -\partial_{x_i}(u(t, \mathbf{x}) h_i(t, \mathbf{x})) + D \partial_{x_i}^2 u(t, \mathbf{x}),$$

with $h_i(t, \mathbf{x}) = h_i(t, x_1, \dots, x_d) = f_g(t, x_i) + \sum_{j=1}^d f_K(x_i, x_j)$. We consider this problem in $d = 8$ dimensions, the one-body force $f_g(t, \mathbf{x}) = \frac{5 \times 10^{1/3}}{4}(\sin(\pi t) + \frac{3}{2}) - \mathbf{x}$, interacting term $f_K(\mathbf{x}, \mathbf{y}) = \frac{1}{2d}(\mathbf{y} - \mathbf{x})$, and diffusion coefficient $D = 0.5$. The initial condition u_0 is an isotropic Gaussian density with mean $\frac{29}{10} + \frac{21}{10(d-1)}[0, 1, \dots, d-1]^T$ and variance $\sigma^2 = 0.1$.

Parametrization and enforcing boundary conditions Since the solution u represents a probability density function, we parameterize the potential $\log u$ of u , instead of u directly. Additionally, we enforce homogeneous Dirichlet boundary conditions via a product structure so that we obtain the parametrization

$$\hat{u}(\mathbf{x}; \boldsymbol{\theta}(t)) = \hat{u}_{\text{BC}}(\mathbf{x}) \exp(\hat{u}_{\text{potential}}(\mathbf{x}; \boldsymbol{\theta}(t))),$$

where $\hat{u}_{\text{potential}}$ is the parametrized potential and

$$\hat{u}_{\text{BC}}(\mathbf{x}) = \prod_{i=1}^d \tanh\left(\frac{1}{2}x_i\right) \tanh\left(\frac{1}{2}(7-x_i)\right), \quad \mathbf{x} = [x_1, \dots, x_d]^T,$$

such that $\hat{u}_{\text{BC}}(\mathbf{x}) = 0$ and $\hat{u}(\mathbf{x}; \boldsymbol{\theta}(t)) = 0$ at the boundary of the domain \mathcal{X} . The parametrization $\hat{u}_{\text{potential}}$ of the potential is a two-layer fully connected network with sigmoid activation and 30 nodes per layer, with a total number of parameters $N = 1230$. As in previous examples, we use the RK4 scheme with a fixed time step size $\delta t = 10^{-3}$.

Monte Carlo mean and covariance Because no analytic solution is available but the mean and covariance can be estimated with Monte Carlo sampling, we evaluate the quality of numerical approximations by comparing their mean and covariance with those obtained with Monte Carlo sampling. The benchmark mean and covariance are estimated with Monte Carlo with 10^5 paths

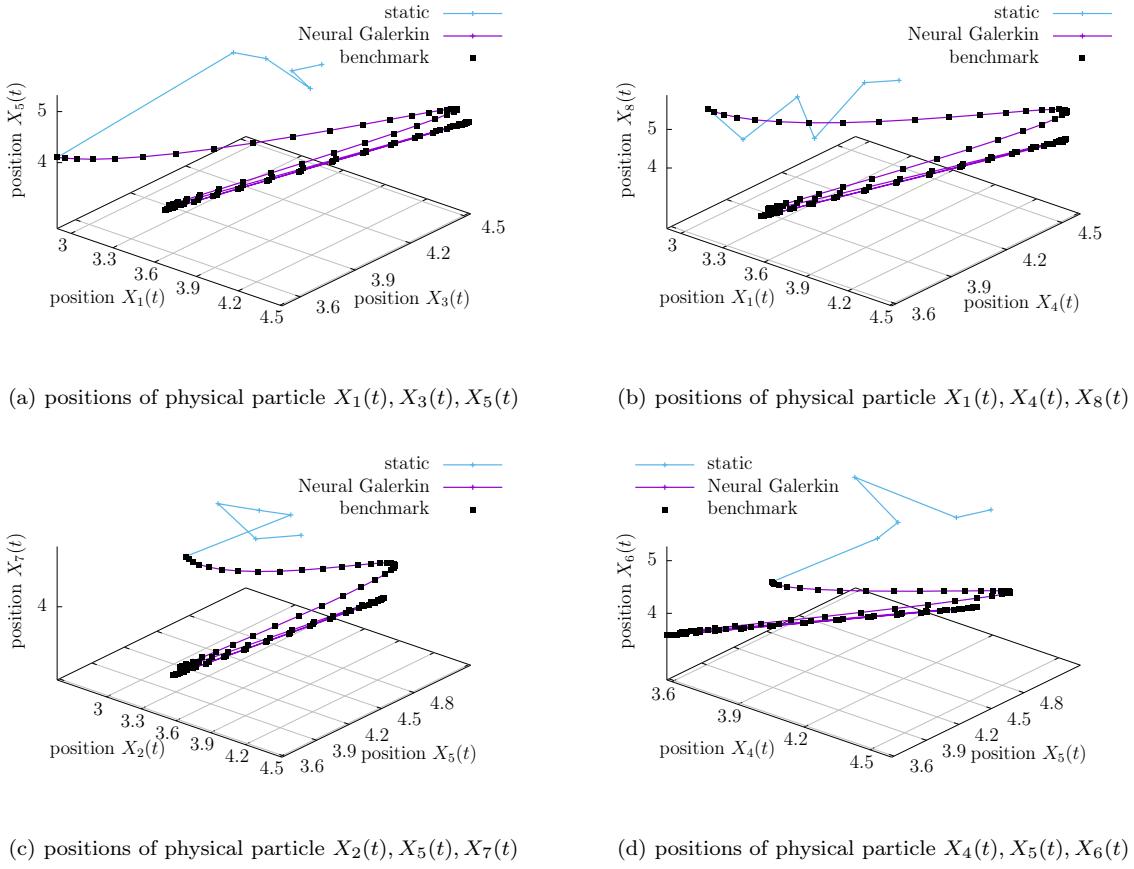


Figure 6: The physical particles concentrate in the eight-dimensional domain over time, which means that the density function u that is approximate by Neural Galerkin becomes local in the spatial domain. However, the dynamic particles allow Neural Galerkin schemes to adaptively keep track of the local behavior over time and give accurate approximations, whereas uniformly sampling the spatial domain leads to large errors in the predictions.

from the SDE given in (20). The mean and covariance of the Neural Galerkin approximations \hat{u} are estimated with self-normalized importance sampling, where the biasing density is the Gaussian with the benchmark mean and covariance.

Quality of predicted mean and covariance We compare the proposed Neural Galerkin scheme with dynamic particles to uniform sampling with $m = 2500$ samples over the eight-dimensional domain \mathcal{X} . We first consider the case where the target measure μ_t is proportional to the squared residual. For the SVGD setup, we take 250 SVGD steps at each time step, with kernel bandwidth 0.05 and SVGD step size 0.5. The tempering parameter is set to $\gamma = 0.5$. As shown in Figure 6, the mean of the positions of the physical particles described by (20) concentrate over time so that the joint density u becomes local in the eight-dimensional spatial domain. With $m = 2500$ samples from the static, uniform measure, mean positions cannot be well approximated. In contrast, for the same number of $m = 2500$ samples, the proposed Neural Galerkin scheme provides accurate predictions of the positions. We quantify the accuracy in Figure 7 that shows the relative error in the mean and covariance with respect to the benchmark Monte Carlo results. Plot (a) of Figure 7 shows the relative error of the mean, which is orders of magnitude lower for Neural Galerkin approximations with dynamic particles than with uniform sampling. Similarly, the relative error of the covariance estimate obtained with Neural Galerkin

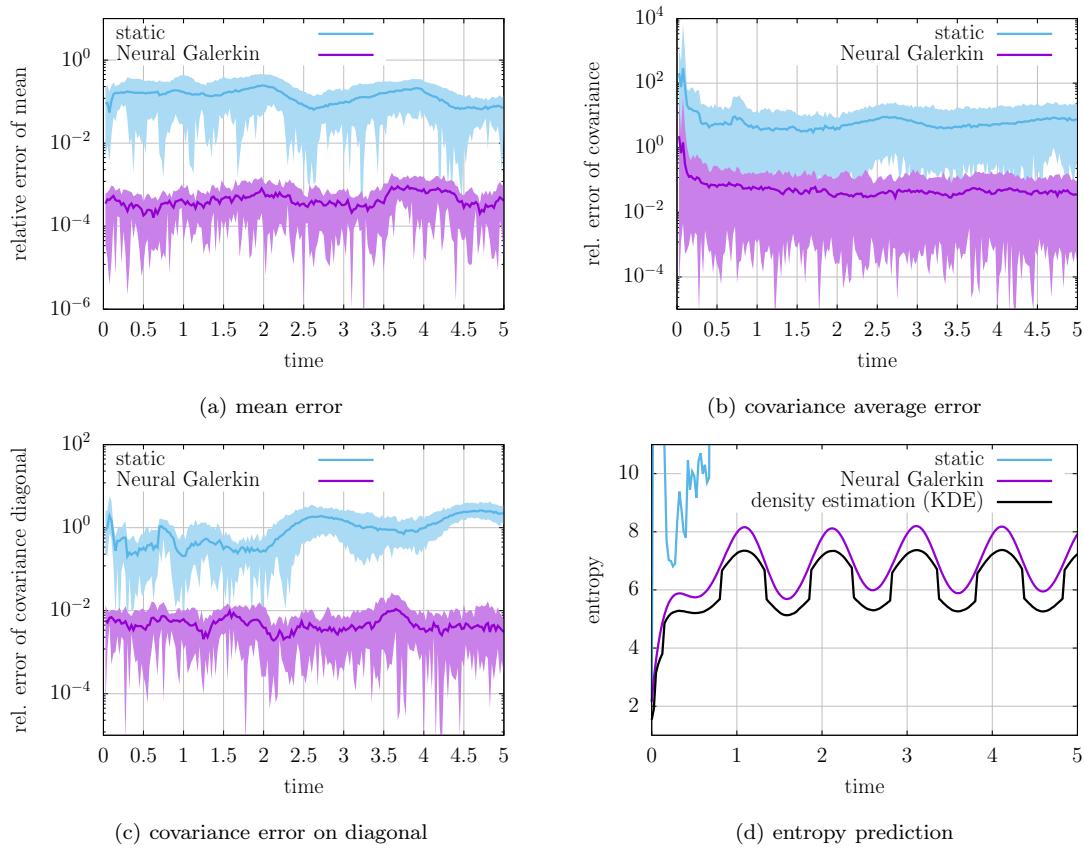


Figure 7: With dynamic particles, Neural Galerkin schemes achieve orders of magnitude higher accuracy in predicting the mean and covariance of the positions $X_1(t), \dots, X_d(t)$. The shaded region shows the minimum and maximum of the relative error over all dimensions, while the bold line shows the average relative error. Because Neural Galerkin schemes approximate the density, rather than providing sample paths as Monte Carlo approaches, quantities of interest that involve the density function such as the entropy can be computed.

and dynamic particles is also orders of magnitude lower than with uniform sampling. In fact, for uniform sampling, the relative error of the covariance is larger than one, which means the corresponding numerical solution is not predictive, which is in agreement with the results shown in Figure 6. If we consider only the diagonal elements of the covariance matrix corresponding to the Neural Galerkin solution with dynamic particles, then the relative error is about one order of magnitude lower than the error averaged over all entries of the covariance matrix.

Computing the entropy with Neural Galerkin schemes In contrast to sampling paths of the SDE (20) with Monte Carlo, we obtain an approximation of the density u of the positions of $X_1(t), \dots, X_d(t)$ and thus can compute downstream quantities such as the entropy. For comparison purposes, we estimate a density from Monte Carlo paths of the SDE (20) with kernel density estimation (KDE) and use it to compute an approximation of the entropy. Figure 7(d) compares the entropy approximation obtained with Neural Galerkin schemes and the entropy obtained with KDE from Monte Carlo paths. For systems as the ones described by the SDE (20), it is known that the entropy oscillates smoothly, which is captured well by the Neural Galerkin approximation. The entropy approximation computed with Monte Carlo and KDE also captures the oscillations but is less smooth and shows spurious jumps.

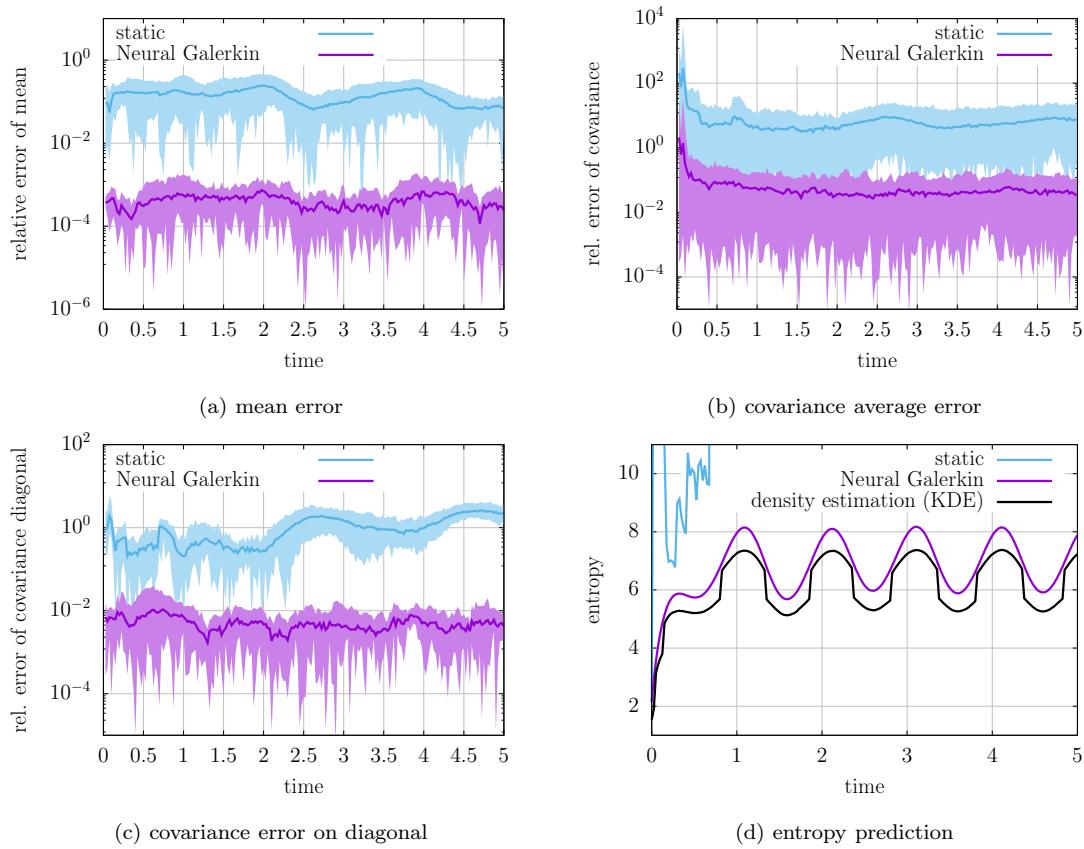


Figure 8: This figure shows analogous results to Figure 7 except that in this figure the particle dynamics are induced by the magnitude of the solution as in (8).

Sampling proportional to the magnitude of the numerical solution Now we impose dynamics on the particles with the target measure being proportional to the magnitude of the solution, as given in (8). In this case, we take 100 SVGD steps at each time step and set the kernel bandwidth to 5.0 and the SVGD step size to 0.01. The rest of the setup is the same as above: The tempering parameter is $\gamma = 0.5$ and the number of particles is $m = 2500$. Figure 8 shows that the Neural Galerkin approximation with dynamic particles achieves a higher accuracy than the approximation obtained via uniform sampling.

Capturing non-Gaussian behavior with Neural Galerkin schemes Because the solution u represents a density, one may be inclined to just approximate it with a Gaussian density, with mean and covariance that can be either estimated or computed via the solution of a system of ordinary differential equations; see [11]. However, as we demonstrate in Figure 9, we consider a configuration of the Fokker-Planck equation that leads to non-Gaussian distributions that are captured well by the Neural Galerkin approximation. The results in Figure 9 demonstrate the importance of approximating the density function rather than just the mean and covariance: The Neural Galerkin solution approximates well the density in general, rather than only the mean and covariance that would also be captured by a Gaussian approximation.

5 Conclusions

Nonlinear parametrizations such as deep neural networks can achieve a faster error decay than traditional linear approximations from an approximation-theoretic perspective. However, numerically fitting the parameters to realize the fast error decay is challenging and critically depends

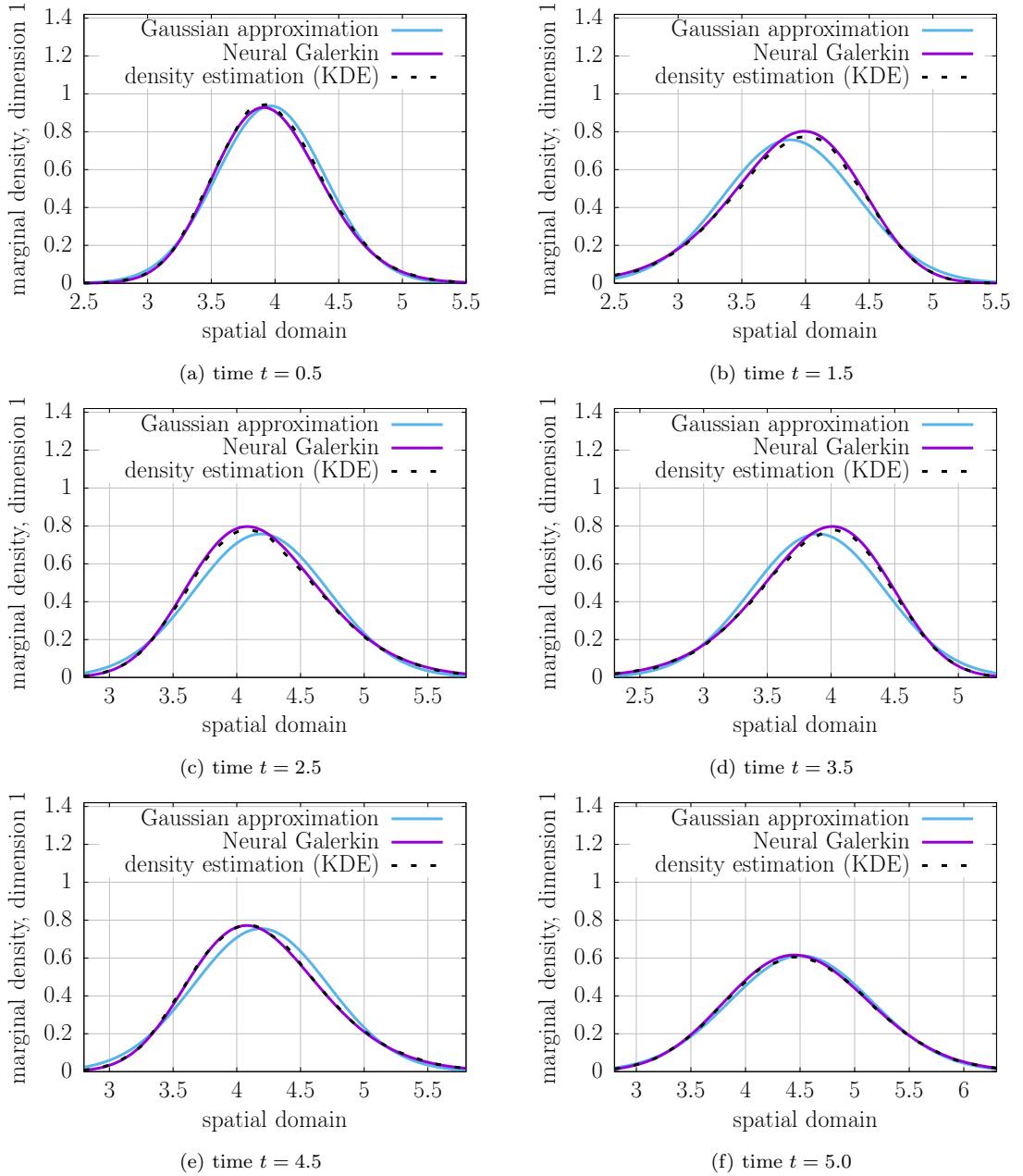


Figure 9: The density u given by the configuration of the Fokker-Planck equation that we consider is slightly different from a Gaussian density. This plot shows that Neural Galerkin approximations capture this difference well, which demonstrates the importance of approximating the density function rather than just the mean and covariance of the distribution.

on the available training data for estimating the population risk with the empirical risk. The results of this work indicate that actively adapting the measure that specifies where in the spatial domain to collect data for training is essential for numerically approximating well solution fields governed by evolution equations. The adaptive data collection is especially important when solution fields have local features that move through the spatial domain over time, such as wave fronts, phase transitions, and other coherent structures.

Acknowledgements

The authors Wen and Peherstorfer were partially supported by the National Science Foundation under Grant No. 2046521 and the Office of Naval Research under award N00014-22-1-2728.

References

- [1] A. Abdulle, E. Weinan, B. Engquist, and E. Vanden-Eijnden. The heterogeneous multiscale method. *Acta Numerica*, 21:1–87, 2012.
- [2] W. Anderson and M. Farazmand. Evolution of nonlinear reduced-order solutions for PDEs with conserved quantities. *SIAM Journal on Scientific Computing*, 44(1):A176–A197, 2022.
- [3] W. Anderson and M. Farazmand. Fast and scalable computation of shape-morphing nonlinear solutions with application to evolutional neural networks. *arXiv*, 2207.13828, 2023.
- [4] M. Barrault, Y. Maday, N. Nguyen, and A. Patera. An “empirical interpolation” method: Application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathématique. Académie des Sciences. Paris*, I:339–667, 2004.
- [5] P. Batlle, Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Error analysis of kernel/GP methods for nonlinear and parametric PDEs. *arXiv*, 2305.04962, 2023.
- [6] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.*, 57(4):483–531, 2015.
- [7] J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, nov 2018.
- [8] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.
- [9] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35:2298–2316, 1998.
- [10] Black, Felix, Schulze, Philipp, and Unger, Benjamin. Projection-based model reduction with dynamically transformed modes. *ESAIM: M2AN*, 54(6):2011–2043, 2020.
- [11] J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden. Neural Galerkin scheme with active learning for high-dimensional evolution equations. *arXiv preprint arXiv:2203.01360*, 2022.
- [12] S. Chaturantabut and D. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.
- [13] P. Y. Chen, J. Xiang, D. H. Cho, Y. Chang, G. A. Pershing, H. T. Maia, M. M. Chiaramonte, K. T. Carlberg, and E. Grinspun. CROM: Continuous reduced-order modeling of PDEs using implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023.

- [14] Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Solving and learning nonlinear PDEs with Gaussian processes. *Journal of Computational Physics*, 447:110668, 2021.
- [15] A. Cohen and R. DeVore. Kolmogorov widths under holomorphic mappings. *IMA J. Numer. Anal.*, 36(1):1–12, 2016.
- [16] A. Cohen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Optimal stable nonlinear approximation. *Foundations of Computational Mathematics*, 2021:1–42, 2021.
- [17] A. Cohen, C. Farhat, A. Somacal, and Y. Maday. Nonlinear compressive reduced basis approximation for PDE’s. *HAL preprint*, 04031976: , 2023.
- [18] A. Cortinovis, D. Kressner, S. Massei, and B. Peherstorfer. Quasi-optimal sampling to learn basis updates for online adaptive model reduction with adaptive empirical interpolation. In *2020 American Control Conference (ACC)*, pages 2472–2477, 2020.
- [19] P. A. M. Dirac. Note on exchange phenomena in the Thomas atom. *Mathematical Proceedings of the Cambridge Philosophical Society*, 26(3):376–385, 1930.
- [20] M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.
- [21] J. Dormand and P. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.
- [22] Z. Drmac and S. Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.
- [23] Y. Du and T. A. Zaki. Evolutional deep neural network. *Physical Review E*, 104(4), oct 2021.
- [24] W. E. The dawning of a new era in applied mathematics. *Notices of the AMS*, 2021.
- [25] W. E and B. Engquist. The heterognous multiscale methods. *Communications in Mathematical Sciences*, 1(1):87 – 132, 2003.
- [26] W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, nov 2017.
- [27] W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, Dec 2017.
- [28] W. E, W. Ren, and E. Vanden-Eijnden. A general strategy for designing seamless multiscale methods. *Journal of Computational Physics*, 228(15):5437–5453, 2009.
- [29] W. E and T. Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2017.
- [30] R. Everson and L. Sirovich. The Karhunen-Loeve Procedure for Gappy Data. *Journal of the Optical Society of America*, 12:1657–1664, 1995.
- [31] M. A. Finzi, A. Potapczynski, M. Choptuik, and A. G. Wilson. A stable and scalable method for solving initial value PDEs with neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

- [32] N. R. Franco, A. Manzoni, and P. Zunino. A deep learning approach to reduced order modelling of parameter dependent partial differential equations. *Math. Comp.*, 92:483–524, 2023.
- [33] J. Frenkel. *Wave Mechanics, Advanced General Theory*. Clarendon Press, Oxford, 1934.
- [34] W. Gao and C. Wang. Active learning based sampling for high-dimensional nonlinear partial differential equations. *Journal of Computational Physics*, 475:111848, 2023.
- [35] J. Gorham and L. Mackey. Measuring sample quality with kernels. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1292–1301. PMLR, 06–11 Aug 2017.
- [36] C. Greif and K. Urban. Decay of the Kolmogorov N -width for wave problems. *Appl. Math. Lett.*, 96:216–222, 2019.
- [37] J. Guo, H. Wang, and C. Hou. A novel adaptive causal sampling method for physics-informed neural networks. *arXiv*, 2210.12914, 2022.
- [38] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, aug 2018.
- [39] J. S. Hesthaven, C. Pagliantini, and G. Rozza. Reduced basis methods for time-dependent problems. *Acta Numerica*, 31:265–345, 2022.
- [40] Y. Khoo, J. Lu, and L. Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6(1):1, Oct 2018.
- [41] Y. Khoo, J. Lu, and L. Ying. Solving parametric PDE problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.
- [42] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.
- [43] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.
- [44] C. L. Wight and J. Zhao. Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *Communications in Computational Physics*, 29(3):930–954, 2021.
- [45] C. Lasser and C. Lubich. Computing quantum dynamics in the semiclassical regime. *Acta Numerica*, 29:229–401, 2020.
- [46] Q. Li, L. Chen, C. Tai, and W. E. Maximum principle based algorithms for deep learning. *Journal of Machine Learning Research*, 18(165):1–29, 2018.
- [47] Q. Li, B. Lin, and W. Ren. Computing committor functions for the study of rare events using deep learning. *The Journal of Chemical Physics*, 151(5):054112, 2019.
- [48] Q. Li, T. Lin, and Z. Shen. Deep learning via dynamical systems: An approximation perspective. *J. Eur. Math. Soc.*, 25:1671–1709, 2023.
- [49] Y. Li, J. Lu, and A. Mao. Variational training of neural network approximations of solution maps for physical models. *Journal of Computational Physics*, 409:109338, 2020.

- [50] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 2378–2386. Curran Associates, Inc., 2016.
- [51] J. Lu, Y. Lu, and J. Nolen. Scaling limit of the Stein variational gradient descent: The mean field regime. *SIAM Journal on Mathematical Analysis*, 51(2):648–671, 2019.
- [52] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, jan 2021.
- [53] C. Lubich. *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*. EMS Press, 2008.
- [54] Y. Maday, A. T. Patera, and G. Turinici. Global a priori convergence theory for reduced-basis approximations of single-parameter symmetric coercive elliptic partial differential equations. *Comptes Rendus Mathematique*, 335(3):289–294, 2002.
- [55] R. Mojgani, M. Balajewicz, and P. Hassanzadeh. Kolmogorov n-width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs. *Computer Methods in Applied Mechanics and Engineering*, 404:115810, 2023.
- [56] F. Negri, A. Manzoni, and D. Amsallem. Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *Journal of Computational Physics*, 303:431–454, 2015.
- [57] M. Nonino, F. Ballarin, G. Rozza, and Y. Maday. A reduced basis method by means of transport maps for a fluid–structure interaction problem with slowly decaying Kolmogorov n-width. *Advances in Computational Science and Engineering*, 1(1):36–58, 2023.
- [58] M. Ohlberger and S. Rave. Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing. *C. R. Math. Acad. Sci. Paris*, 351(23-24):901–906, 2013.
- [59] B. Peherstorfer. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *SIAM Journal on Scientific Computing*, 42(5):A2803–A2836, 2020.
- [60] B. Peherstorfer. Breaking the Kolmogorov barrier with nonlinear model reduction. *Notices of the American Mathematical Society*, 69:725–733, 2022.
- [61] B. Peherstorfer, Z. Drmac, and S. Gugercin. Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points. *SIAM Journal on Scientific Computing*, 42:A2837–A2864, 2020.
- [62] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations*. Springer, 2016.
- [63] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [64] R. Rico-Martinez, J. S. Anderson, and I. G. Kevrekidis. Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 596–605, 1994.

- [65] R. Rico-Martinez, K. Krischer, I. G. Kevrekidis, M. C. Kube, and J. L. Hudson. Discrete- vs. continuous-time nonlinear signal processing of Cu electrodissolution data. *Chemical Engineering Communications*, 118(1):25–48, 1992.
- [66] F. Romor, G. Stabile, and G. Rozza. Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method. *Journal of Scientific Computing*, 94(3):74, Feb 2023.
- [67] G. M. Rotskoff, A. R. Mitchell, and E. Vanden-Eijnden. Active importance sampling for variational objectives dominated by rare events: Consequences for optimization and generalization. In J. Bruna, J. Hesthaven, and L. Zdeborova, editors, *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pages 757–780. PMLR, 16–19 Aug 2022.
- [68] G. Rozza, D. Huynh, and A. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):1–47, 2007.
- [69] T. P. Sapsis and P. F. Lermusiaux. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Physica D: Nonlinear Phenomena*, 238(23):2347–2360, 2009.
- [70] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, dec 2018.
- [71] T. R. Taha and M. J. Ablowitz. Analytical and numerical aspects of certain nonlinear evolution equations. III. Numerical, Korteweg-de Vries equation. *J. Comput. Phys.*, 55:231–253, 1984.
- [72] K. Tang, X. Wan, and C. Yang. DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 2023.
- [73] V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.
- [74] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.
- [75] R. Zimmermann, B. Peherstorfer, and K. Willcox. Geometric subspace updates with applications to online adaptive nonlinear model reduction. *SIAM Journal on Matrix Analysis and Applications*, 39(1):234–261, 2018.