Φ-DVAE: Physics-Informed Dynamical Variational Autoencoders for Unstructured Data Assimilation

Alex Glyn-Davies[a,*], Connor Duffin[a], O. Deniz Akyildiz[b], Mark Girolami[a,c]

[a]*Department of Engineering, University of Cambridge, Trumpington St, Cambridge, CB2 1PZ, United Kingdom*
[b]*Department of Mathematics, Imperial College London, Exhibition Rd, London, SW7 2AZ, United Kingdom*
[c]*The Alan Turing Institute, London, British Library, 96 Euston Rd, NW1 2DB, United Kingdom*

Abstract

Incorporating unstructured data into physical models is a challenging problem that is emerging in data assimilation. Traditional approaches focus on well-defined observation operators whose functional forms are typically assumed to be known. This prevents these methods from achieving a consistent model-data synthesis in configurations where the mapping from data-space to model-space is unknown. To address these shortcomings, in this paper we develop a physics-informed dynamical variational autoencoder (Φ-DVAE) to embed diverse data streams into time-evolving physical systems described by differential equations. Our approach combines a standard, possibly nonlinear, filter for the latent state-space model and a VAE, to assimilate the unstructured data into the latent dynamical system. Unstructured data, in our example systems, comes in the form of video data and velocity field measurements, however the methodology is suitably generic to allow for arbitrary unknown observation operators. A variational Bayesian framework is used for the joint estimation of the encoding, latent states, and unknown system parameters. To demonstrate the method, we provide case studies with the Lorenz-63 ordinary differential equation, and the advection and Korteweg-de Vries partial differential equations. Our results, with synthetic data, show that Φ-DVAE provides a data efficient dynamics encoding methodology which is competitive with standard approaches. Unknown parameters are recovered with uncertainty quantification, and unseen data are accurately predicted.

1. Introduction

Physical models, as represented by differential equations, are ubiquitous throughout engineering and the physical sciences. These equations synthesise scientific knowledge into mathematical form. However, as a description of reality they are imperfect [1], leading to the well-known problem of model misspecification [2]. At least since Kalman [3] physical modellers have been trying to reconcile their inherently misspecified models with observations [4]. Such approaches are usually either solving the inverse problem of attempting to recover model parameters from data, and/or, the data assimilation (DA) problem of conducting state inference based on a time-evolving process.

For the inverse problem, Bayesian methods are common [5, 6]. In this, prior belief in model parameters $\mathbf{\Lambda}$ is updated with data $\mathbf{y}$ to give a posterior distribution, $p(\mathbf{\Lambda}|\mathbf{y})$. This describes uncertainty with parameters given the data and modelling assumptions. DA can also proceed from a Bayesian viewpoint, where inference is cast as a nonlinear state-space model (SSM) [7, 8]. The SSM is typically the combination of a time-discretised differential equation and an observation process: uncertainty enters the model through extrusive, additive errors. For a latent state variable $\mathbf{u}_n$ representing some discretised system at time n, with observations $\mathbf{y}_n$, the object of interest is the filtering distribution $p(\mathbf{u}_n|\mathbf{y}_{1:n})$, where $\mathbf{y}_{1:n} := \{\mathbf{y}_k\}_{k=1}^n$. The joint filtering and estimation problem, which estimates $p(\mathbf{u}_n, \mathbf{\Lambda}|\mathbf{y}_{1:n})$ has received attention in the literature (see, e.g., Kantas et al. [9] and references therein). This has been well studied in, e.g., electrical engineering [10], geophysics [11], neuroscience [12], chemical engineering [13], biochemistry [14], and hydrology [15], to name a few.

---

*Corresponding author. Email: ag933@cam.ac.uk

Typically in data assimilation tasks, while parameters of an observation model may be unknown, the observation model itself is assumed known [9]. This assumption breaks down in settings where data arrives in various modalities, such as videos, images, or audio, hindering the ability to perform inference. However, in such cases often the underlying variation in the data stream is due to a latent physical process, which is typically at least partially known. We refer to this setting, where the observation operator of the physical process is unknown, as the problem of unstructured data assimilation (UDA). In many physics-based applications, simplified or partially known physics models are available, e.g. weather prediction, soil mechanics, fluid mechanics, but require data to be assimilated into the model to adjust for misspecification, and improve predictive capabilities. Our framework is developed with these applications in mind, where we take a physics-informed machine learning approach to allow for the assimilation of *unstructured* data into underlying physical models.

In this work, we develop a variational Bayes (VB) [16] methodology which jointly solves the inverse and filtering problems when this observation operator is unknown. We model this unknown mapping with a variational autoencoder (VAE) [17], which embeds the assumed time-dependent observations $\mathbf{y}_{1:N}$ into encodings $\mathbf{x}_{1:N}$ in a latent space. On this latent space, we stipulate that the latent encodings are observations emitted from a known dynamical system, given by a stochastic differential equation with possibly unknown coefficients. Following [18], we refer to these latent encodings as "pseudo-observations". To be clear, these pseudo-observations are not directly observed, but are treated as observations of the latent SSM. The differential equation is also assumed to have stochastic forcing, which accounts for possible model misspecification. The stipulated system gives a structured prior $p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda})$, which acts as a physics-informed regulariser whilst also enabling inference over the unknown model parameters $\boldsymbol{\Lambda}$. This prior is approximated using classical nonlinear filtering algorithms. Our framework is fully probabilistic: inference proceeds from a derived evidence lower bound (ELBO), enabling joint estimation of unknown network parameters and unknown dynamical coefficients via VB. To set the scene for this work, we now review the relevant literature.

2. Related Work

As introduced above, VAEs [17] are a popular high-dimensional encoder. A VAE defines a generative model that learns low-dimensional representations, $\mathbf{x}$, of high-dimensional data, $\mathbf{y}$, using VB in an unsupervised fashion. To perform efficient inference, a variational approximation $q_\phi(\mathbf{x}|\mathbf{y})$ is made to the intractable posterior $p(\mathbf{x}|\mathbf{y})$, via a neural network (NN). Variational parameters ϕ are estimated via the ELBO. Recent works have extended the VAE to high-dimensional time-series data $\mathbf{y}_{1:N}$, with the aim of jointly learning latent representations $\mathbf{x}_{1:N}$, and a dynamical system that evolves them [19]. Termed dynamical variational autoencoders (DVAEs), they enforce the dynamics with a structured prior $p(\mathbf{x}_{1:N})$ on the latent space.

The Kalman variational autoencoder (KVAE) of [18] is a DVAE methodology, which encodes $\mathbf{y}_{1:N}$ into latent variables $\mathbf{x}_{1:N}$ that are assumed to be observations of a linear Gaussian state-space model driven by latent dynamic states $\mathbf{u}_{1:N}$. Assumed linear dynamics are jointly learnt with the encoder and decoder, via Kalman filtering/smoothing. Another approach is the Gaussian process (GP) VAE [20, 21, 22], which models $\mathbf{x}_{1:N}$ as a temporally correlated GP. The Markovian variant of [23] allows for a similar Kalman procedure as in the KVAE, except, in this instance, the dynamics are known and are given by an stochastic differential equation (SDE) approximation to the GP [24]. Yildiz et al. [25] also propose the so-called ODE^{2}VAE, which encodes the data to an initial condition which is integrated through time using a Bayesian neural ordinary differential equation (ODE) [26]. This trajectory, only, is used to generate the reconstructions via the decoder network.

A related class of methods are deep SSMs [27, 28, 29]. These works assume that the parametric form of the SSM is unknown, and replace the transition and emission distributions with NN models, which are trained based on an ELBO. They harness the representational power of deep NNs to directly model transitions between high-dimensional states. More emphasis is placed on generative modelling and prediction than representation learning, or system identification. We also note the related works which use VAE-type architectures for video prediction tasks [30, 31, 32], and the variational recurrent approach of Chung et al. [33], which is predominantly demonstrated on speech modelling.

Methods to blend physics and autoencoders have also been developed. A popular approach uses SINDy [34] for discovery of low-dimensional latent dynamical systems using autoencoders [35]. A predictive framework is given in Lopez and Atzberger [36], which aims to learn nonlinear dynamics by jointly optimising an ELBO.

Following our notation, this learns a function which maps $\mathbf{u}_n \mapsto \mathbf{u}_{n+k}$, for some k, via a VAE. Lusch et al. [37] use a physics-informed autoencoder to linearise nonlinear dynamical systems via a Koopman approach; inference is regularised through incorporating the Koopman structure in the loss function. Otto and Rowley [38] present a similar method, and an extension of these approaches to PDE systems is given in Gin et al. [39]. Morton et al. [40] use the linear regression methods of Takeishi et al. [41] within a standard autoencoder to similarly compute the Koopman observables. Erichson et al. [42] derive an autoencoder which incorporates a linear Markovian prediction operator, similar to a Koopman operator, which uses physics-informed regulariser to promote Lyapunov stability. Hernández et al. [43] studies VAE methods to encode high-dimensional dynamical systems. Finally, we note the related work which studies the estimation of dynamical parameters within so-called "gray-box" systems, blending NN methods with known physical laws [44, 45, 46, 47]. Methods for solving Bayesian inverse problems using VAEs have been developed for stochastic differential equations [48], and notably the UQ-VAE [49] for when the forward model is known. The UQ-VAE uses observation/latent pairs for training, and the forward model used as the decoder. These methods require the observation operator to be known.

Going beyond such "gray-box" methods, deep learning methods have been increasingly applied to directly approximating partial differential equations (PDEs). Termed the physics-informed neural network (PINN) [50], physical laws corresponding to the PDE are imposed on the loss function of the NN; solutions that do not obey such laws are thus penalised. The framework is flexible, and can be used to solve high-dimensional forward problems [51, 52], or to solve inverse problems where noisy data are observed [50]. A Bayesian variant of the standard PINN has been developed in [53] to provide uncertainty quantification in the forward and inverse problem setting, using Bayesian neural networks.

These PINN models fall under the category of "physics-informed" machine learning. Recent reviews of physics-informed machine learning [54, 55] use this term to refer to integration of prior physics knowledge/mathematical physics models with data-driven machine learning methods. More specifically, PINNs and other methodologies that impose PDE constraints to learn surrogate models *without* data are termed "physics-constrained" learning methods [52, 56], allowing surrogates to be constructed without the need for solving the governing equations. An emerging term in the dynamical systems literature is "physics-enhanced" machine learning [57, 58]. Coined as an umbrella term, this encompasses three types of scientific machine learning: physics-guided, physics-encoded, and physics-informed. Physics-guided having the highest level of physics bias, physics-informed being more data-driven, and physics-encoded providing a combination of both.

Another recent advancement in solving both forward and inverse problems using deep learning is the neural operator [59]. Neural operators are parameterised maps between function spaces, and have been used to learn PDE solution operators. They are therefore discretisation-invariant models, making them robust to mesh-refinement, and able to solve super-resolution problems [60].

These model can be used in place of slower, traditional PDE solvers as efficient surrogates for the forward model, and perform comparably when used for downstream tasks such as Bayesian inversion [61]. Related variational approaches have also been developed for time-varying physical systems by parameterising time as another input to deep NNs that are used to parameterise the variational family [62, 63].

Our contribution

In this paper we propose a physics-informed dynamical variational autoencoder (Φ-DVAE): a DVAE approach which imposes the additional structure of known physics on the latent space. The term physics-informed is used in this paper to describe this integration of the data-driven autoencoder model with the known mathematical physics model. We assume that there is a low-dimensional dynamical system generating the high-dimensional observed time-series. A NN is used to learn the unknown embedding to this lower dimensional space. On the lower-dimensional space, the embedded data are pseudo-observations of a latent dynamical system, which is, in general, derived from a numerical discretisation of a nonlinear PDE. However, the methodology is suitably generic, allowing for ODE latent systems. Inference on this latent system is done with efficient nonlinear stochastic filtering methods, enabling the use of mature DA algorithms within our framework. A probabilistically coherent VB construction allows for joint learning of both the embedding and unknown dynamical parameters.

Instead of learnt dynamics with the KVAE [18], Φ-DVAE assumes a misspecified nonlinear differential equation is driving the variation in the latent space, with possibly unknown parameters. Specifying the dynamics gives a known latent transition density, which is parameterised by physics parameters instead of neural networks

3

as in, e.g., deep SSMs [27, 28, 29]. Whilst we share commonality with latent differential equations, the Φ-DVAE differs with the ODE^{2}VAE [25] as we perform inference with this ODE/PDE, instead of learning it and leveraging it to deterministically evolve the latents. Specifying transitions via a differential equation differs the methodology to incorporating generic physics principles in the latent space [such as 37, 38, 39, 42], or generic temporal structure [such as 20, 21, 22, 23].

Our work supplements the physics-informed machine learning literature through addressing the crucial gap of embedding a physics-driven description of latent state evolution within VAEs. We also provide, to the best of our knowledge, the first contribution within this field which addresses the problem of inversion using physics-informed machine learning [seen in 48, 49, 50, 53, 60, 61] under an unknown observation operator. Results, with synthetic data, show that the method learns sensible encodings, in limited-data settings, and accurately estimates physical model parameters, with uncertainty quantification. We further show that the method performs comparably to current approaches, and, when extrapolating beyond training data, accurately predicts future latent states.

This paper is structured as follows. The probabilistic model is described in Section 3, which shows how the known physics are embedded into the latent space of a VAE (Section 3.1). The pseudo-observation model for the physical system is defined in Section 3.2 and the pseudo-observations $\mathbf{x}_{1:N}$ are related to the unstructured data $\mathbf{y}_{1:N}$ in Section 3.3. In Section 4.1 we derive the variational inference in the setting of the probabilistic model. The learning objective is given by the log-marginal likelihood of the observed data $\mathbf{y}_{1:N}$ as defined by the probabilistic model, and we derive a tractable lower-bound approximation to this. In Sections 4.2 and 4.3 we define the inverse problem we wish to solve, and outline the algorithm for performing the inference. Experiments and results for three different examples are given in Section 5. The paper is concluded in Section 6.

3. The Probabilistic Model

In this section we define our probabilistic model; our presentation roughly follows the structure of the generative model. We first give an overview of the dependencies between variables, as described by conditional probabilities. We then cover the latent differential equation model used to describe the underlying physics. The pseudo-observation model is defined, followed by the decoder and the encoder. We assume a general SSM:

$$\mathbf{\Lambda} \sim p(\mathbf{\Lambda}),$$
$$\mathbf{u}_n|\mathbf{u}_{n-1}, \mathbf{\Lambda} \sim p(\mathbf{u}_n|\mathbf{u}_{n-1}, \mathbf{\Lambda}),$$
$$\mathbf{x}_n|\mathbf{u}_n \sim p(\mathbf{x}_n|\mathbf{u}_n),$$
$$\mathbf{y}_n|\mathbf{x}_n \sim p_\theta(\mathbf{y}_n|\mathbf{x}_n),$$

where $\mathbf{\Lambda}$ describes the parameters of the Markov process $\{\mathbf{u}_n\}_{n=0}^N$. The *dynamic model* $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \mathbf{\Lambda})$ defines the evolution of this process in time. The pseudo-observation likelihood is denoted by $p(\mathbf{x}_n|\mathbf{u}_n)$. Parameters θ are the NN parameters for the *decoder* $p_\theta(\mathbf{y}_n|\mathbf{x}_n)$. The conditional independence structure imposed by the model gives

$$p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N}, \mathbf{u}_{1:N}, \mathbf{\Lambda}) = p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N})p(\mathbf{u}_{1:N}|\mathbf{\Lambda})p(\mathbf{\Lambda}). \tag{1}$$

Intuitively, $\{\mathbf{y}_n\}_{n=1}^N$ is the sequence of high-dimensional observations, $\{\mathbf{x}_n\}_{n=1}^N$ is its embedding/pseudo-data, and $\{\mathbf{u}_n\}_{n=0}^N$ is the *latent physics process*. For each n, we assume that $\mathbf{y}_n \in \mathbb{R}^{n_y}$, $\mathbf{x}_n \in \mathbb{R}^{n_x}$, $\mathbf{u}_n \in \mathbb{R}^{n_u}$, and $\mathbf{\Lambda} \in \mathbb{R}^{n_\lambda}$. In what follows, we describe the components of our probabilistic model in detail.

3.1. Dynamic Model

The first component of the generative model is the latent dynamical system $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \mathbf{\Lambda})$. We model this latent physics process $\{\mathbf{u}_n\}_{n=0}^N$ as a discretised stochastic PDE, however ODE latent physics is admissible within our framework (see Section 5.2). We discretise this process with the statistical finite element method (statFEM) [64, 65, 66, 67], forming the basis of the physics-informed prior on the latent space. Stochastic additive forcing inside the PDE represents additive model error, which results from possibly misspecified physics. Full details, including the ODE case, are given in Appendix C.
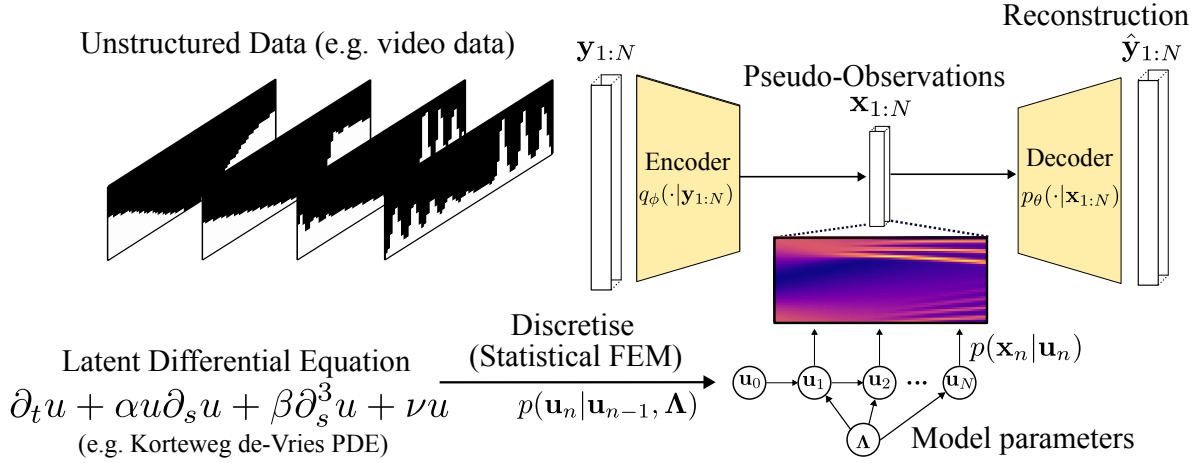
Figure 1: An illustration of the Φ-DVAE model. On the left, the video frames are seen, denoted $\mathbf{y}_{1:N}$. These are converted into physically interpretable low-dimensional encodings $\mathbf{x}_{1:N}$ using an encoder. The learning is informed by the physics-driven state-space model, which treats $\mathbf{x}_{1:N}$ as *pseudo-observations* (bottom right). These pseudo-observations are used to infer the latent states $\mathbf{u}_{1:N}$.

We assume that the model has possibly unknown coefficients $\mathbf{\Lambda}$ (henceforth referred to as 'model parameters') and that these parameters are constant in time. On these we place the Bayesian prior $\mathbf{\Lambda} \sim p(\mathbf{\Lambda})$ [6], describing our *a priori* knowledge on the model parameters before observing any data. We also assume that $\mathbf{u}_0$, the initial condition, is known up to measurement noise, with prior $p(\mathbf{u}_0)$ set accordingly. For pedagogical purposes, we derive the discrete-time dynamic model using the Korteweg-de Vries (KDV) equation as a running example, which is used in later sections as an example PDE. KDV is given by:

$$\partial_t u + \alpha u \partial_s u + \beta \partial_s^3 u + \nu u = \dot{\xi}, \quad \dot{\xi} \sim \mathcal{GP}(0, \delta(t - t') \cdot k(s, s')), \tag{2}$$

where $u := u(s, t) \in \mathbb{R}$, $\xi := \xi(s, t)$, $s \in [0, L]$, $t \in [0, T]$, and $\mathbf{\Lambda} \subseteq \{\alpha, \beta, \nu\}$. Informally $\dot{\xi}$ is a GP, with delta correlations in time, and spatial correlations given by the covariance kernel $k(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ [68]. This is an uncertain term in the PDE, representing possible model misspecification. We assume all GP hyperparameters are known in this work but note that this methodology can be extended to estimate them via the marginal likelihood [68] or in a fully Bayesian manner [69]. The KDV equation is used to model nonlinear internal waves [see, e.g., 70], and describes the balance between nonlinear advection and dispersion. Note that although the KDV equation defines a scalar field, $u(s, t)$, the approach similarly holds for vector fields.

Following STATFEM, the equations are first spatially discretised with the finite element method (FEM) [71], then discretised in time. Equation 2 is multiplied with a smooth test function $w(s) \in W$, where W is an appropriate function space, and integrated over the domain Ω to give the weak form [72]

$$\langle \partial_t u, w \rangle + \alpha \langle u \partial_s u, w \rangle + \beta \langle \partial_s^3 u, w \rangle + \nu \langle u, w \rangle = \langle \dot{\xi}, w \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the $L^2(\Omega)$ inner product.

The domain is discretised to give the mesh Ω_h with vertices $\{s_j\}_{j=0}^{n_h}$. In this case, we take the s_j to be a uniformly spaced set of points, so that $s_j = jh$ (h gives the spacing between grid-points). On the mesh a set of polynomial basis functions $\{\phi_j(s)\}_{i=1}^{n_u}$ is defined, such that approximation to the PDE. Letting $u_h(s, t) = \sum_{i=1}^{n_u} u_i(t) \phi_i(s)$, the weak form is now rewritten with these basis functions

$$\langle \partial_t u_h, \phi_j \rangle + \alpha \langle u_h \partial_s u_h, \phi_j \rangle + \beta \langle \partial_s^3 u_h, \phi_j \rangle + \nu \langle u_h, \phi_j \rangle = \langle \dot{\xi}, \phi_j \rangle, \quad j = 1, \dots, n_u.$$

This gives a finite-dimensional SDE over the FEM coefficients $\mathbf{u}(t) = (u_1(t), \dots, u_{n_u}(t))^\top$:

$$\mathbf{M} \frac{d\mathbf{u}}{dt} + \alpha \mathcal{F}(\mathbf{u}) + \beta \mathbf{A}\mathbf{u} + \nu \mathbf{M}\mathbf{u} = \dot{\xi}, \quad \dot{\xi}(t) \sim \mathcal{N}(\mathbf{0}, \delta(t - t') \cdot \mathbf{G}),$$

where $\mathbf{M}_{ij} = \langle \phi_i, \phi_j \rangle$, $\mathbf{A}_{ji} = \langle \partial_s^3 \phi_i, \phi_j \rangle$, $\mathcal{F}(\mathbf{u})_j = \langle u_h \partial_s u_h, \phi_j \rangle$, and $\mathbf{G}_{ij} = \langle \phi_i, \langle k(\cdot, \cdot), \phi_j \rangle \rangle$. Time discretisation eventually gives the transition density $p(\mathbf{u}_n | \mathbf{u}_{n-1}, \boldsymbol{\Lambda})$, for $\mathbf{u}_n = \mathbf{u}(n\Delta_t)$, whose form is dependent on the discretisation used (see Appendix C).

3.2. Likelihood

The second component of the generative model is the *likelihood $p(\mathbf{x}_n | \mathbf{u}_n)$, for pseudo-data $\{\mathbf{x}_n\}_{n=1}^N$*. The introduction of the likelihood in the model is usually necessary, as high-dimensional observations $\{\mathbf{y}_n\}_{n=1}^N$ may only be generated by some observed dimensions of $\{\mathbf{u}_n\}_{n=0}^N$. For example, perhaps it is known *a priori* that only a single component of a latent coupled differential equation generates the observations (see also Section 5.2). This explicit likelihood is introduced to separate the encoding process from the state-space inference; in practice we sample the pseudo-data from the variational encoding distribution $\mathbf{x}_{1:N} \sim q_\phi(\cdot | \mathbf{y}_{1:N})$, then condition on it with standard nonlinear filtering algorithms [18].

The latent states $\mathbf{u}_n$ are mapped at discrete times to pseudo-observations via $\mathbf{x}_n = \mathbf{H}\mathbf{u}_n + \mathbf{r}_n$, with $\mathbf{r}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. This gives the observation density $p(\mathbf{x}_n | \mathbf{u}_n) = \mathcal{N}(\mathbf{H}\mathbf{u}_n, \mathbf{R})$. Both the pseudo-observation operator $\mathbf{H} \in \mathbb{R}^{n_x \times n_u}$ and the noise covariance $\mathbf{R} \in \mathbb{R}^{n_x \times n_x}$ are assumed to be known in this work. See Appendix C for further details on the construction of the observation model. The combination of the transition and observation densities provides the nonlinear Gaussian SSM:

$$\text{Transition:} \quad \mathbf{u}_n = \mathcal{M}(\mathbf{u}_{n-1}) + \mathbf{e}_{n-1}, \quad \mathbf{e}_{n-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}),$$
$$\text{Pseudo-observation:} \quad \mathbf{x}_n = \mathbf{H}\mathbf{u}_n + \mathbf{r}_n, \quad \mathbf{r}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}).$$

The extended Kalman filter (ExKF) [73, 7] is used to infer the latent states given the pseudo-observations by maintaining a Gaussian approximation to the filtering posterior $p(\mathbf{u}_n | \mathbf{x}_{1:n}, \boldsymbol{\Lambda}) \approx \mathcal{N}(\mathbf{m}_n, \mathbf{C}_n)$, that is recursively updated. The filtering posterior $p(\mathbf{u}_n | \mathbf{x}_{1:n}, \boldsymbol{\Lambda})$ is the distribution of the current latent state, $\mathbf{u}_n$, given the observations up to and including the current time, $\mathbf{x}_{1:n}$. Here we can see the recursive prediction and update steps

$$\text{Prediction:} \quad p(\mathbf{u}_{n+1} | \mathbf{x}_{1:n}, \boldsymbol{\Lambda}) = \int p(\mathbf{u}_n | \mathbf{x}_{1:n}, \boldsymbol{\Lambda}) p(\mathbf{u}_{n+1} | \mathbf{u}_n, \boldsymbol{\Lambda}) d\mathbf{u}_n$$
$$\text{Update:} \quad p(\mathbf{u}_{n+1} | \mathbf{x}_{1:n+1}, \boldsymbol{\Lambda}) \propto p(\mathbf{x}_{n+1} | \mathbf{u}_{n+1}) p(\mathbf{u}_{n+1} | \mathbf{x}_{1:n+1}, \boldsymbol{\Lambda})$$

Firstly the next latent state is predicted using the transition density $p(\mathbf{u}_{n+1} | \mathbf{u}_n, \boldsymbol{\Lambda})$, then the filtering posterior is updated based on the next observation using the observation density $p(\mathbf{x}_{n+1} | \mathbf{u}_{n+1})$. For a linear Gaussian SSM, the Kalman filter computes these steps exactly. However when the SSM is nonlinear these steps can be computed approximately using an appropriate nonlinear filter, e.g. the ExKF. Details on the prediction and update steps of the ExKF are provided in Appendix C.

3.3. Decoder

The last component of our generative model is the decoder $p_\theta(\mathbf{y}_n | \mathbf{x}_n)$, which describes the unknown mapping between the pseudo-observations $\mathbf{x}_n$, and the observed data $\mathbf{y}_n$. The decoding of latents to data should model as closely as possible the true data generation process. Prior knowledge about this process can be used to select an appropriate $p_\theta(\mathbf{y}_n | \mathbf{x}_n)$. No temporal structure is assumed on θ, so the decoder is shared across all times $p_\theta(\mathbf{y}_{1:N} | \mathbf{x}_{1:N}) = \prod_{n=1}^N p_\theta(\mathbf{y}_n | \mathbf{x}_n)$. The function $f_\theta \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ maps the latent encoding to the data reconstruction $\hat{\mathbf{y}}_n = f_\theta(\mathbf{x}_n)$, and for a simple Gaussian probabilistic decoder this reconstruction is the mean, and a Gaussian reconstruction error is assumed, i.e. $\mathbf{y}_n \sim \mathcal{N}(f_\theta(\mathbf{x}), \sigma^2 \mathbf{I})$. For more details on specific architectures defining f_θ see Appendix A.

4. Inference

In the previous section we introduced the probabilistic model, describing the joint distribution of the data $\mathbf{y}$, pseudo-observations $\mathbf{x}$ and latent states $\mathbf{u}$. This section develops the methodology for performing inference on the latent states based on this probabilistic model, i.e. conditional on the data, what is the posterior distribution of the pseudo-observations and latent states? The inference can be broken down into two parts, the VAE which
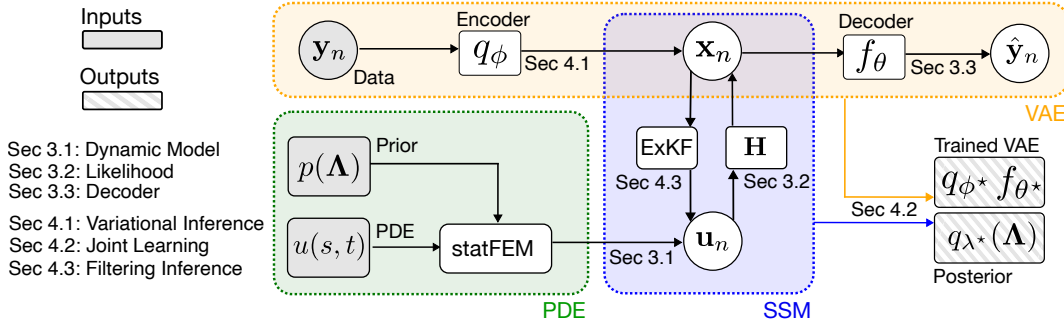
Figure 2: Flow diagram describing connections between the specified partial differential equation (PDE), latent state-space model (SSM) and variational autoencoder (VAE). Inputs to the method include the unstructured data $\mathbf{y}_{1:N}$, the specified PDE $u(x, s)$ and the model parameter prior (solid grey fill). Outputs are the trained autoencoder parameters $\{\phi^\star, \theta^\star\}$, and the model parameter posterior distribution parameters $\lambda^\star$ (striped grey fill). Labels are included referencing the relevant sections.

encodes high-dimensional data into the space of pseudo-observations, approximating $p(\mathbf{x}_n|\mathbf{y}_n)$, and the ExKF for inferring the latent states given the pseudo-observations, approximating $p(\mathbf{u}_n|\mathbf{x}_{1:n}, \mathbf{\Lambda})$. Figure 2 visualises how the VAE and the latent SSM interact, and includes references to the relevant subsections where the parts of the model are described.

Section 4.1 describes the inference of the pseudo-observations given the data; this posterior is estimated using a variational approximation, constructed via a VAE. Section 4.2 describes how we infer the model parameters using a VB approximation to the model parameter posterior. Finally we detail the filtering inference in Section 4.3, where the ExKF is used to approximate the filtering posterior and make predictions of the latent states at future times.

4.1. Variational Inference

In this section, we introduce the variational family and the ELBO. Denote by $q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N})$ the variational posterior, which, similar to Fraccaro et al. [18], factorises as

$$q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N}) = q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \mathbf{\Lambda})q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})q_\lambda(\mathbf{\Lambda}). \tag{3}$$

Note here that we do not make a variational approximation $q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \mathbf{\Lambda})$; this is taken to be the exact posterior $p(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \mathbf{\Lambda})$. We derive the ELBO to be (see also Appendix B)

$$\log p(\mathbf{y}_{1:N}) \geq \int \log \left[\frac{p(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}, \mathbf{y}_{1:N})}{q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N})} \right] q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N})\mathrm{d}\mathbf{x}_{1:N}\mathrm{d}\mathbf{u}_{1:N}\mathrm{d}\mathbf{\Lambda}$$

$$= \mathbb{E}_{q_\phi} \left[\log \frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})}{q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})} + \mathbb{E}_{q_\lambda} \left[\log p(\mathbf{x}_{1:N}|\mathbf{\Lambda}) + \log \frac{p(\mathbf{\Lambda})}{q_\lambda(\mathbf{\Lambda})} \right] \right]. \tag{4}$$

Typically this expectation is not analytically tractable and Monte Carlo (MC) is used to compute an approximation. Estimating $\log p(\mathbf{x}_{1:N}|\mathbf{\Lambda})$ requires marginalising over $\mathbf{u}_{1:N}$, *the latent physics process*. As detailed above, the ExKF recursively computes a Gaussian approximation to the filtering posterior $p(\mathbf{u}_n|\mathbf{x}_{1:n}, \mathbf{\Lambda}) \approx \mathcal{N}(\mathbf{m}_n, \mathbf{C}_n)$. We note however that this can also be realised with other nonlinear filters, e.g., ensemble Kalman filters [74] or particle filters [75]. The factorisation of the pseudo-observation marginal likelihood, $p(\mathbf{x}_{1:N}|\mathbf{\Lambda}) = p(\mathbf{x}_1|\mathbf{\Lambda}) \prod_{n=2}^{N} p(\mathbf{x}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda})$, enables computation, as the filter can compute $p(\mathbf{x}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda})$, at each prediction step, via $p(\mathbf{x}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda}) = \int p(\mathbf{x}_n|\mathbf{u}_n, \mathbf{\Lambda})p(\mathbf{u}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda}) \, \mathrm{d}\mathbf{u}_n$.

Encoder parameters ϕ are shared between variational distributions $\left\{q_\phi(\mathbf{x}_n|\mathbf{y}_n)\right\}_{n=1}^{N}$ to give an amortized approach [76]. Unless otherwise specified, for each n the encoding has the form $q_\phi(\mathbf{x}_n|\mathbf{y}_n) = \mathcal{N}(\mu_\phi(\mathbf{y}_n), \sigma_\phi(\mathbf{y}_n))$. Functions $\mu_\phi(\mathbf{y}_n)\colon \mathbb{R}^{n_y} \to \mathbb{R}^{n_x}$ and $\log \sigma_\phi(\mathbf{y}_n)\colon \mathbb{R}^{n_y} \to \mathbb{R}^{n_x}$ are NNs, with parameters ϕ to be learnt. Parameter updates are performed using the Adam optimiser [77]. Practical implementation details and specific encoding architectures are given in Appendix A. As for q_λ, the form of the variational approximation is set based on the parameters of interest, see Table 1 for a few candidate distributions.

4.2. Joint Learning

The inverse problem we aim to solve is the joint estimation of the filtering posterior, $p(\mathbf{u}_n|\mathbf{y}_{1:n})$, and the model parameters $\boldsymbol{\Lambda}$. Here, we assume the latent transition distribution, $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \boldsymbol{\Lambda})$ is known and can be constructed from the PDE via statFEM, and treat $\boldsymbol{\Lambda}$ as a parameter we wish to estimate via the variational approximation in Eq. 3. The prior distribution $p(\boldsymbol{\Lambda})$ in effect places a prior over a family of partial differential equations, whose posterior is targeted by learning the parameters of the variational approximation $q_\lambda(\boldsymbol{\Lambda})$. These distribution parameters λ are learned via gradient ascent of the ELBO, using MC samples from the variational posterior $\boldsymbol{\Lambda}^{(i)} \sim q_\lambda(\cdot)$.

Algorithm 1: Φ-DVAE: joint learning of autoencoder and model parameter distribution

Input: training data $\mathbf{y}_{1:N}$, SSM $\{\mathcal{M}, \mathbf{H}, \mathbf{Q}, \mathbf{R}\}$, model parameter prior $p(\boldsymbol{\Lambda})$, initial parameters $\{\phi_0, \theta_0, \lambda_0\}$, step number K

for $k = 0, \ldots, K-1$ **do**

    $\mathbf{x}_{1:N} \sim q_{\phi_k}(\cdot|\mathbf{y}_{1:N})$ ▷ Sample pseudo-observations

    $\boldsymbol{\Lambda} \sim q_{\lambda_k}(\cdot)$ ▷ Sample model parameters

    $\mathcal{L}_k = \log p_{\theta_k}(\mathbf{y}_{1:N}|\mathbf{x}_{1:N}) - \log q_{\phi_k}(\mathbf{x}_{1:N}|\mathbf{y}_{1:N}) + \log p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda}) - \mathrm{KL}(q_{\lambda_k} \| p)$ ▷ Calculate ELBO

    $(\phi_{k+1}, \theta_{k+1}, \lambda_{k+1}) = \mathrm{Adam}(\nabla \mathcal{L}_k, \phi_k, \theta_k, \lambda_k)$ ▷ Adam update step

end

return $(\phi_K, \theta_K, \lambda_K)$

This relies on the KL-divergence between the prior and variational posterior being available analytically, which is the case for some choices of prior and variational posterior. We also require a closed form reparameterisation for sampling from the posterior to allow for gradient computation. Table 1 details the relevant formulae for a few popular univariate distributions. Note that results are available for multivariate cases, e.g. see [68] for multivariate Gaussian reparametrisation via the Cholesky decomposition, and analytic KL-divergence between two multivariate Gaussians. Pseudo-code for the joint learning is provided in Algorithm 1.

Distribution	λ	Support	PDF	KL-Divergence	Reparameterise
Gaussian	$\{\mu, \sigma\}$	$x \in \mathbb{R}$	$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\frac{\mu_2}{2} - \frac{\mu_1}{2} - \frac{1}{2} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2}$	$\mu + \sigma * Z$
Exponential	$\{\lambda\}$	$x \in \mathbb{R}^+$	$\lambda\exp(-\lambda x)$	$\log\left(\frac{\lambda_1}{\lambda_2}\right) + \frac{\lambda_2}{\lambda_1} - 1$	$-\log(U)/\lambda$
Log-normal	$\{\mu, \sigma\}$	$x \in \mathbb{R}^+$	$\frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{(\log(x)-\mu)^2}{2\sigma^2}\right)$	$\log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{(\mu_1-\mu_2)^2+\sigma_1^2-\sigma_2^2}{2\sigma_2^2}$	$\exp(\mu + \sigma * Z)$

Table 1: Variational distributions for parameters, $Z \sim \mathcal{N}(0,1)$, $U \sim \mathcal{U}[0,1]$.

4.3. Filtering Inference

Once the autoencoder parameters $\{\phi, \theta\}$ have been learned jointly with the variational parameters λ, we can assess the performance of the Φ-DVAE algorithm by comparing the filtering posterior $p(\mathbf{u}_n|\mathbf{y}_{1:n})$, to the true latent solution $\mathbf{u}(n\Delta_t)$ that generated the data. This assesses the ability of both the model to provide meaningful encodings via the autoencoder, and the extended Kalman filter to give latent solutions that match the true data generating solution. This distribution can be approximated via MC samples from the variational autoencoder, marginalising over the latent encodings $\mathbf{x}_{1:N}$ and the model parameters $\boldsymbol{\Lambda}$. We can approximate the filtering marginal distribution at a time n:

$$p(\mathbf{u}_n|\mathbf{y}_{1:n}) \approx \int p(\mathbf{u}_n|\mathbf{x}_{1:n}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:n}|\mathbf{y}_{1:n})\, d\boldsymbol{\Lambda}d\mathbf{x}_{1:n}$$

$$\approx \frac{1}{M}\sum_{i=1}^{M} p(\mathbf{u}_n|\mathbf{x}_{1:n}^{(i)}, \boldsymbol{\Lambda}^{(i)}), \quad \mathbf{x}_{1:n}^{(i)} \sim q_\phi(\cdot|\mathbf{y}_{1:n}), \; \boldsymbol{\Lambda}^{(i)} \sim q_\lambda(\cdot). \tag{5}$$

8

The resulting distribution is a mixture of Gaussians over the latent state, with each $p(\mathbf{u}_n|\mathbf{x}_{1:n}^{(i)}, \boldsymbol{\Lambda}^{(i)}) = \mathcal{N}(\mathbf{m}_n^{(i)}, \mathbf{C}_n^{(i)})$.

The model parameter inference can be assessed by comparing the variational posterior $q_\lambda(\boldsymbol{\Lambda})$ to the prior distribution $p(\boldsymbol{\Lambda})$ and the true coefficient value that generated the data.

Finally we can compare the performance of the joint estimation by marginalising over the model parameters $\boldsymbol{\Lambda}$, and comparing this to a model that learned the autoencoder parameters conditioning only on samples from the prior $p(\boldsymbol{\Lambda})$. We denote this trained prior model $\{\phi, \theta\}$, and denote the jointly trained model $\{\phi^\star, \theta^\star, \lambda^\star\}$. This should isolate the impact of learning a variational approximation to the model parameters $\boldsymbol{\Lambda}$ on the resulting latent inference and predictive power. Ideally, the learned posterior should be more concentrated around the ground truth value of the model parameter, giving predictions of latent states that are closer to the true latent process. For prediction, the aim will be to infer the values of the latent process past the final time of the training dataset; these predictions are made by iterating only the prediction step of the Kalman filter ahead in time from the final mean and covariance. We will refer to these as *rollout predictions*, as in essence we are 'rolling out' the filter to provide future predictions without incorporating observations.

For the rollout, the target distribution in the latent state a number of time steps τ ahead of step n, i.e. $p(\mathbf{u}_{n+\tau}|\mathbf{y}_{1:n})$. Consider the distribution over the latents a single time step ahead,

$$
\begin{aligned}
p(\mathbf{u}_{n+1}|\mathbf{y}_{1:n}) &\approx \int p(\mathbf{u}_{n+1}|\mathbf{u}_n, \boldsymbol{\Lambda})p(\mathbf{u}_n|\mathbf{x}_{1:n}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:n}|\mathbf{y}_{1:n})\mathrm{d}\mathbf{u}_n\mathrm{d}\mathbf{x}_{1:n}\mathrm{d}\boldsymbol{\Lambda} \\
&= \int \underbrace{\left[\int p(\mathbf{u}_{n+1}|\mathbf{u}_n, \boldsymbol{\Lambda})p(\mathbf{u}_n|\mathbf{x}_{1:n}, \boldsymbol{\Lambda})\mathrm{d}\mathbf{u}_n\right]}_{\text{Kalman Filter Prediction Step}} q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:n}|\mathbf{y}_{1:n})\mathrm{d}\mathbf{x}_{1:n}\mathrm{d}\boldsymbol{\Lambda} \\
&= \int p(\mathbf{u}_{n+1}|\mathbf{x}_{1:n}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:n}|\mathbf{y}_{1:n})\mathrm{d}\mathbf{x}_{1:n}\mathrm{d}\boldsymbol{\Lambda} \\
&\approx \frac{1}{M}\sum_{i=1}^{M} p(\mathbf{u}_{n+1}|\mathbf{x}_{1:n}^{(i)}, \boldsymbol{\Lambda}^{(i)}), \quad \mathbf{x}_{1:n}^{(i)} \sim q_\phi(\cdot|\mathbf{y}_{1:n}), \ \boldsymbol{\Lambda}^{(i)} \sim q_\lambda(\cdot).
\end{aligned} \tag{6}
$$

Here, the mixture of Gaussians distribution is constructed with the one-step-ahead predictive distributions $p(\mathbf{u}_{n+1}|\mathbf{x}_{1:n}^{(i)}, \boldsymbol{\Lambda}^{(i)}) = \mathcal{N}(\hat{\mathbf{m}}_{n+1}^{(i)}, \hat{\mathbf{C}}_{n+1}^{(i)})$. This step can be iterated on the mean and covariance to give predictions τ steps ahead, i.e. the distribution $p(\mathbf{u}_{n+\tau}|\mathbf{x}_{1:n}^{(i)}, \boldsymbol{\Lambda}^{(i)})$.

5. Experiments

In this section, we aim to simulate the following scenario where a practitioner with an experimental setup recording unstructured data generated by a physical process. There is a mechanistic model the practitioner believes to be the underlying physical process, with some unknown model parameters described through a prior distribution. Given this setup, our model can be used to embed this unstructured data into the latent space described by the physical model. The resulting output is a posterior distribution of the latent states given the unstructured data (given by embedding the data with the trained VAE and filtering with the ExKF, giving $p(\mathbf{u}_n|\mathbf{y}_{1:n})$), and an updated estimate of the model parameters (in the form of a variational posterior $q_\lambda(\boldsymbol{\Lambda})$). We provide three experimental examples, each with different latent differential equations:

- **Advection equation:** First, we introduce a *linear* PDE example, in the form of the advection equation with video data as unstructured observations. Due to the linearity of the latent space, a vanilla Kalman filter can be used for filtering pseudo-observations. Therefore, this example provides a valid comparison to to another DVAE method, the KVAE, which uses a vanilla Kalman filter and learns a general linear Gaussian SSM.

- **Lorenz-63:** The second example extends this to the Lorenz-63 system of *nonlinear* ordinary differential equations, where we demonstrate the ExKF for nonlinear filtering. In this example, we use velocity field measurements as the unstructured data. Experiments consider the joint estimation of a model parameter and inference of unobserved latent states.

- **KᴅV equation:** Finally, our most complex example is the *nonlinear* and *partial* differential equation: the KᴅV equation, with video data observations. Here, both sᴛᴀᴛFEM and ExKF are employed. We demonstrate the ability of Φ-DVAE to adjust for model parameter misspecification and improve future latent prediction, purely using information extracted from video data.

We choose to affect our data with noise, to reflect the manner in which this data would be collected. For velocity field data, additive Gaussian noise is applied to simulate measurement errors, and for video data we include salt-and-pepper noise [78] to simulate noise for image frames.

5.1. Advection PDE

As our first example, we consider the advection equation with periodic boundary conditions. In this example, we derive the transition density $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \Lambda)$ from a sᴛᴀᴛFEM discretisation of a stochastic advection equation:

$$\partial_t u + c\, \partial_s u = \dot{\xi}, \quad \dot{\xi} \sim \mathcal{GP}(0, \delta(t-t') \cdot k(s, s')), \tag{7}$$

where $u := u(s,t)$, $\xi := \xi(s,t)$, $s \in [0,1]$, $t \in [0,40]$, and $u(s,t) = u(s+1,t)$. Recall that, as in Section 3.2, the FEM coefficients $\mathbf{u}_n = (u_1(n\Delta_t), \dots, u_{n_u}(n\Delta_t))$ are the latent variables. These are related to the discretised solution via $u_h(s, n\Delta_t) = \sum_{i=1}^{n_u} u_i(n\Delta_t)\phi_i(s)$ (see Appendix A and Appendix C).

Video data $\mathbf{y}_{1:N}$ is generated from the deterministic version of (7) (i.e. (7) with $\xi \equiv 0$). Since the solution of this equation is available analytically, we use this for data generation. Denoting the initial condition $f(s) = u(s,0)$, the solution to (7) at a time t is given by $u(s,t) = f(s-ct)$. This trajectory is evaluated at discrete time points $\{n\Delta_t\}_{n=1}^N$ and the solutions $u(s, n\Delta_t)$ are imposed onto a 2D grid. On the grid, pixels below $u(s, n\Delta_t)$ are lit-up in a binary fashion, along with salt-and-pepper noise. In this experiment we use fixed parameters, setting the wavespeed $c = 0.5$. We set the decoder as $p_\theta(\mathbf{y}_n|\mathbf{x}_n) = \mathrm{Bern}(f_\theta(\mathbf{x}_n))$ and the encoder as $q_\phi(\mathbf{x}_n|\mathbf{y}_n) = \mathcal{N}(\mu_\phi(\mathbf{y}_n), \sigma_\phi(\mathbf{y}_n))$. As previous, see Appendix A for full details.

Due to linearity of the underlying dynamical system, we compare the Φ-DVAE to the KVAE for a set of video data generated from the advection equation, for various dimensions of the KVAE latent space. Specifying a particular form of PDE dynamics on the latent states increases the inductive bias imposed on the latent space, and should provide better reconstructions and higher likelihood of the data. We choose two metrics to evaluate this: the ELBO given by $\log p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N}) - \log q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N}) + \log p(\mathbf{x}_{1:N})$, and the normalised mean-squared-error (MSE) of reconstruction compared to the noise-free video frames, $\mathbf{y}_{1:N}^{\text{true}}$. The normalised MSE is given by $\|\mathbf{y}_{1:N}^{\text{true}} - \hat{\mathbf{y}}_{1:N}\|_2^2 / \|\mathbf{y}_{1:N}^{\text{true}}\|_2^2$. We compare our method to KVAE for the linear advection example for varying latent dimension of the KVAE, and the results are plotted in Fig. 3. We also include the results for our method but with deterministic latent transitions ($\mathbf{Q} = \mathbf{0}$) to assess the impact of accounting for misspecification through the use of sᴛᴀᴛFEM.
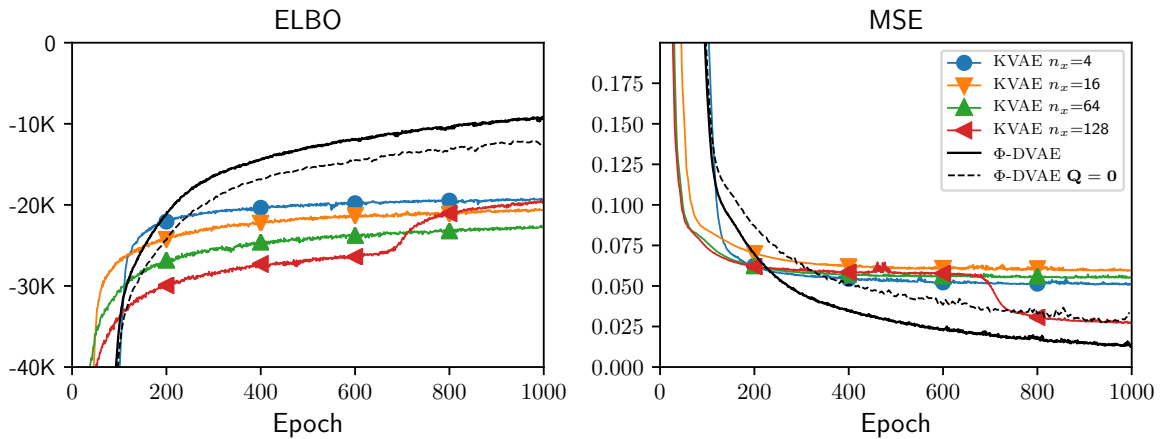

Figure 3: Comparison of KVAE and Φ-DVAE for the advection equation.

From Fig. 3, Φ-DVAE outperforms each KVAE model, with a larger ELBO and smaller normalised MSE during training. The MSE for the Φ-DVAE model is 1.26%, compared to the best KVAE model ($n_x = 128$) with 2.71%, where roughly a 5% normalised MSE is considered a good reconstruction. The KVAE models performed similarly in terms of ELBO, with dimension 4 and 128 the best with -19273 and -19659 respectively, but are outperformed by both Φ-DVAE models. We also note that including the process noise matrix $\mathbf{Q}$ for the sᴛᴀᴛFEM model which accounts for misspecification outperforms the model that evolves the latent states deterministically.

5.2. Lorenz-63

In our next example, the latent dynamical model $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \mathbf{\Lambda})$ is given by an Euler-Maruyama discretisation [79] of the stochastic Lorenz-63 system,

$$
\begin{aligned}
du_1 &= (-\sigma u_1 + \sigma u_2)dt + dw_1, \\
du_2 &= (-u_1 u_3 + r u_1 - u_2)dt + dw_2, \\
du_3 &= (u_1 u_2 - b u_3)dt + dw_3,
\end{aligned}
\tag{8}
$$

where $\mathbf{u}(t) := [u_1(t), u_2(t), u_3(t)]^\top$, $t \in [0, 6]$, $\mathbf{u}_n = [u_1(n\Delta_t), u_2(n\Delta_t), u_3(n\Delta_t)]$, $\mathbf{\Lambda} \subseteq \{\sigma, r, b\}$, and w_1, w_2, and w_3 are independent Brownian motion processes [80].

Latent Dynamics: $\mathbf{u}_{1:N}$ **Pseudo-Observations: $\mathbf{x}_{1:N}$** **Velocity Field: $\mathbf{y}_N$**
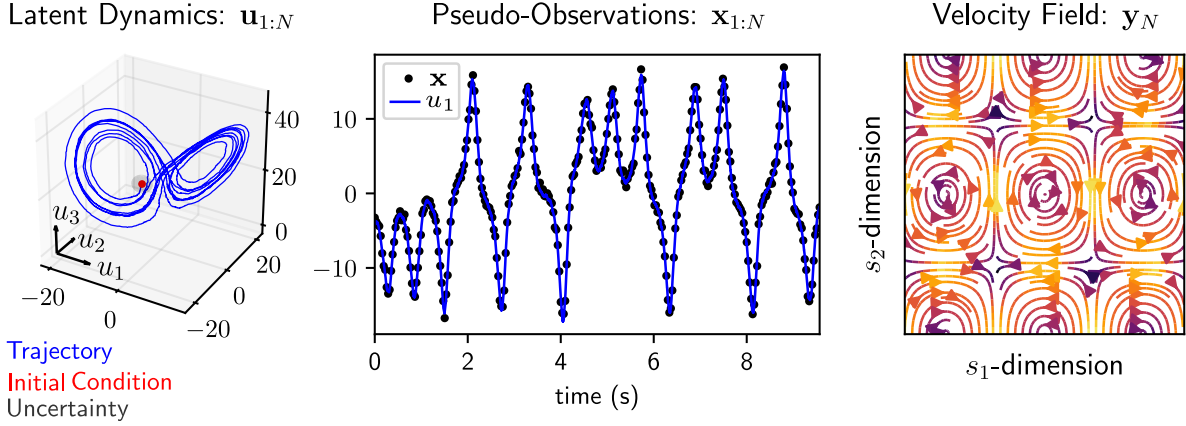
Figure 4: Lorenz-63: latent states $\mathbf{u}_{1:N}$, pseudo-observations $\mathbf{x}_{1:N}$, and velocity field $\mathbf{y}_N$.

We observe synthetic 2D velocity fields, $\mathbf{y}_{1:N}$ of convective fluid flow and we use our method to embed these synthetic data into the stochastic Lorenz-63 system. The Lorenz-63 system is related to the velocity fields through a truncated spectral expansion. In brief, it is assumed that the velocity fields have no vertical velocity, so the 3D velocity field is realised in 2D. The velocity field can be described by the stream function $\psi := \psi(s_1, s_2, t)$, where s_1 and s_2 are the spatial coordinates of variation, respectively, and thus $\mathbf{y}(t) = (-\partial_{s_2}\psi, 0, \partial_{s_1}\psi)$. A truncated spectral approximation and a transform to non-dimensional equations yields $\psi(s_1, s_2, t) \propto u_1(t) \sin(\pi s_1/l) \sin(\pi s_2/d)$, where $u_1(t)$ is governed by the Lorenz-63 ODE (i.e., (8) with $w_i \equiv 0$). For full details we refer to Appendix A and Appendix C. The Lorenz-63 system is classical system, popularised in Lorenz [81] through its characteristic "deterministic nonperiodic flow": chaotic dynamics.

To generate the synthetic data $\mathbf{y}_{1:N}$, we generate a trajectory $\mathbf{u}^{\text{true}}_{1:N}$ from the deterministic version of 8, at discrete timepoints $n\Delta_t$, and use the generated $u^{\text{true}}_{1,n}$ to compute the two-dimensional velocity field, $\mathbf{y}_n$, via $\psi(s_1, s_2, t)$. This corresponds to having a middle layer $x_n = u_{1,n} + w_n$ where $w_n \sim \mathcal{N}(0, R^2)$, with likelihood $p(x_n|\mathbf{u}_n) = \mathcal{N}(\mathbf{h}^\top\mathbf{u}_n, R^2)$ where $\mathbf{h} = [1, 0, 0]^\top$. The synthetic data of $\mathbf{u}^{\text{true}}(t)$, $\mathbf{x}_{1:N}$, and $\mathbf{y}_N$ are visualised in Figure 4; the full trajectory $\mathbf{u}^{\text{true}}(t)$ is shown in 3D, and the velocity field $\mathbf{y}_n$ is shown in 2D, for a single n. The decoding is assumed to be of the form $p_{\mathbf{w}}(\mathbf{y}_n|x_n) = \mathcal{N}(\mathbf{w}x_n, \eta^2\mathbf{I})$, with unknown coefficients $\mathbf{w} \in \mathbb{R}^{n_y}$ which are to be learned from the data. The variational encoding $q_{\mathbf{w}}(x_n|\mathbf{y}_n)$ is determined via a pseudo-inverse, as detailed in Appendix A, along with the relevant hyperparameters and numerical details.

Lorenz-63 Results

The coefficients used to generate the Lorenz-63 data are $\{\sigma, b, r\} = \{10, 8/3, 28\}$, and it is assumed both r and b are known, whereas σ is not. A poorly specified prior $p(\Lambda) = \mathcal{N}(30, 5^2)$ is placed over this unknown parameter $\Lambda = \sigma$, in order to test the model's ability to correct this misspecification through the adjustment of the variational posterior.

For this experiment, we first train the linear autoencoder parameters while sampling the model parameter from the prior distribution. This pre-training step is to allow sensible VAE embeddings to be learned before adjusting the variational posterior over model parameters [18]. The parameters of the variational distribution $\lambda = \{\mu_\lambda, \sigma_\lambda\}$ are then jointly estimated with the autoencoder parameters to train the complete model, $\{\mathbf{w}^\star, \lambda^\star\}$. See Algorithm 1.

The training data consists of the first 50 velocity field measurements. The latent states that generated these measurements come from only one 'wing' of the characteristic butterfly-shaped Lorenz-63 attractor. This assesses the model's ability to extrapolate, and encode velocity measurements unseen during training that are generated from the other 'wing' of the attractor. The full dataset $\mathbf{y}_{1:N}$ is encoded using the trained variational autoencoder, and a sample from the variational posterior is compared to the ground truth pseudo-observations $\mathbf{x}_{1:N}^{\text{true}}$ in Fig. 5. The sampled encoding is close to both the training and test ground truth pseudo-observations. Results for the
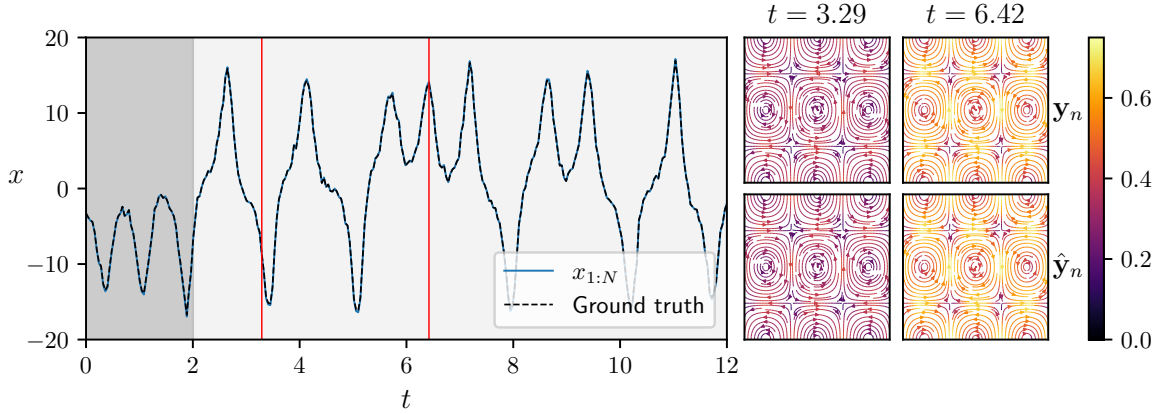

Figure 5: *Left:* Ground truth pseudo-observations (dashed) compared to encoding $\mathbf{x}_{1:N} \sim q_{\phi^\star}(\cdot|\mathbf{y}_{1:N})$ (solid). Training data indicated by grey-fill ($t = [0, 2)$), red lines indicate reconstruction times. *Right:* Observed data $\mathbf{y}_n$ plotted alongside reconstructions $\hat{\mathbf{y}}_n$, with the velocity field shown as a streamplot where the color indicates the speed of the flow.

estimation of parameter σ during the joint training are given in Fig. 6a. The variational distribution is initialised at the prior $q_{\lambda_0}(\Lambda) = \mathcal{N}(30, 5^2)$, and over the joint training period the mean converges on the true value $\sigma = 10$, and the variance reduces and appears to level off. Fig. 6b shows the latent inference for the full dataset, the first column shows the distribution $p(\mathbf{u}_n|\mathbf{x}_{1:n}, \Lambda)$, with sampled $\mathbf{x}_{1:n} \sim q_{\phi^\star}(\cdot|\mathbf{y}_{1:N})$ and $\Lambda \sim q_{\lambda^\star}(\cdot)$ which shows good correspondence to the true latent values. The third column shows the mixture of Gaussian approximation to $p(\mathbf{u}_n|\mathbf{y}_{1:n})$ (see Eq. 5), with the second column visualising the individual mixture components. As expected, the observed latent dimension u_1 has a smaller variance than the unobserved components u_2, u_3, with low bias seen for all three dimensions. Fig. 7 shows the results of the predictive rollout. In this scenario, the training data is encoded from $t = 0$ to $t = 2$, and the latent states are predicted ahead in time by iterating Eq. 6 until $t = 4.5$. The predictive variance for each sample is displayed as a ± 2 standard deviation fill for each sample. For reference, the Lyapunov time for the deterministic latent system with the ground truth parameters is shown as a red vertical bar. This indicates the time for which prediction should be possible for the chaotic system [82, 83]. The latent predictions from the misspecified prior model rapidly diverge from the true latent state. In comparison, the posterior predictions successfully track the latent state up until the Lyapunov time, where predictions become uncertain and begin to diverge as expected. It should be noted the prior model is misspecified in the sense that the prior mean produces a non-chaotic system, with most of the probability mass concentrated on non-chaotic systems.
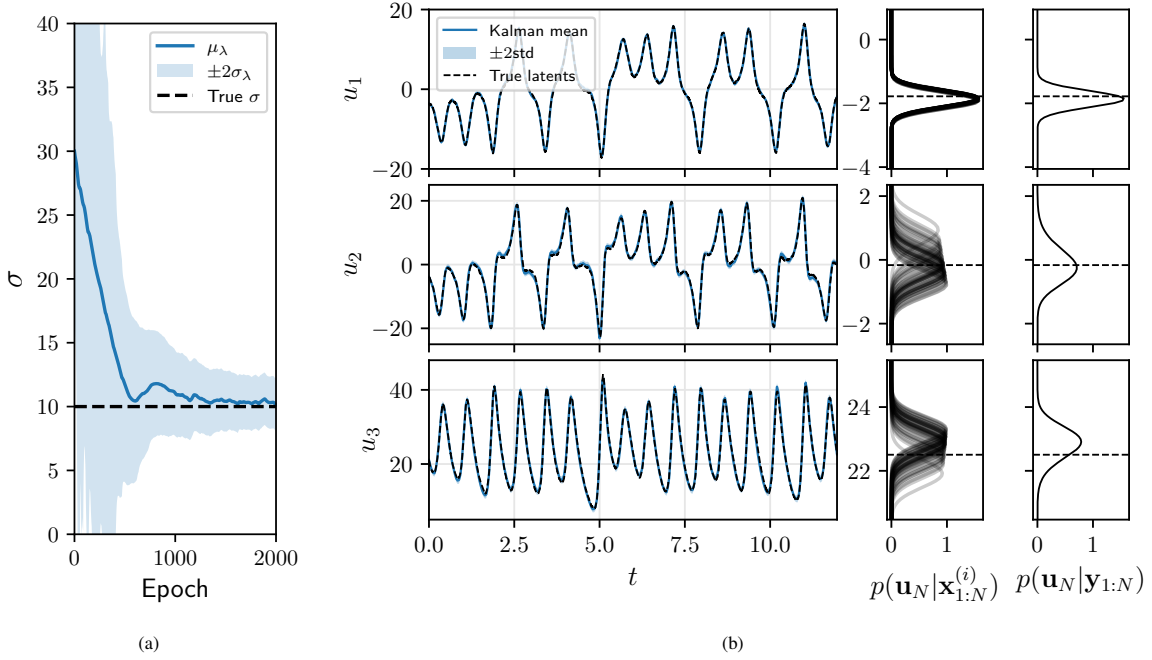
Figure 6: Inference results for Φ-DVAE jointly trained on Lorenz-63 system. (a) Trace plot of parameter variational distribution $q_\lambda(\Lambda) = \mathcal{N}(\mu_\lambda, \sigma_\lambda^2)$, with mean (blue) and ± 2 standard deviations (blue fill). Initialised at poorly specified prior, $p(\Lambda) = \mathcal{N}(30, 5^2)$. True value $\sigma = 10$ (dashed). (b) *First column:* Filtering inference for latent states $\mathbf{u}_{1:N}$, where the filtering distribution $p(\mathbf{u}_n|\mathbf{x}_{1:n})$ is plotted with the ground truth $\mathbf{u}_n^{\text{true}}$. *Second column:* Final-time filtering distribution $p(\mathbf{u}_N|\mathbf{x}_{1:N}^{(i)})$, for multiple samples $\mathbf{x}_{1:N}^{(i)}, i = 1, \ldots, 50$. *Third column:* Mixture of Gaussians distribution, $p(\mathbf{u}_N|\mathbf{y}_{1:N})$, calculated via Eq. 5. True latent values $\mathbf{u}_N^{\text{true}}$ shown as horizontal dashed lines.

5.3. Korteweg–de Vries PDE

As previously, the latent transition density $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \Lambda)$ defines the evolution of the FEM coefficients, as given by a statFEM discretisation of a stochastic KdV equation:

$$\partial_t u + \alpha u \partial_s u + \beta \partial_s^3 u + \nu u = \dot{\xi}, \quad \dot{\xi} \sim \mathcal{GP}(0, \delta(t - t') \cdot k(s, s')),$$

where $u := u(s,t), \xi := \xi(s,t), s \in [0,2], t \in [0,1]$, and $u(s,t) = u(s+2,t)$. We set the coefficients $\alpha = 1, \beta = 0.022^2$, and we investigate two cases, one with and one without drag ($\nu = 0, \nu = 1$). The initial condition is set to $u_h^{\text{true}}(s,0) = \cos(\pi s)$ as in the classical work of Zabusky and Kruskal [84]. This regime is characterised by the steepening of the initial condition and the generation of solitons; nonlinear waves which have particle-like interactions [70].

Data is generated by simulating a trajectory $\mathbf{u}_{1:N}^{\text{true}}$ using a FEM discretisation of the deterministic KdV equation and we impose FEM solutions $u_h^{\text{true}}(s, n\Delta_t)$ on a 2D grid. We light up pixels below the solution, which mimics an experimental setup where the wave-profile is recorded side-on. Independent salt-and-pepper noise is used to corrupt the image frames. We set the decoder as $p_\theta(\mathbf{y}_n|\mathbf{x}_n) = \text{Bern}(\mu_\theta(\mathbf{x}_n))$ and the encoder as $q_\phi(\mathbf{x}_n|\mathbf{y}_n) = \mathcal{N}(\mu_\phi(\mathbf{y}_n), \sigma_\phi(\mathbf{y}_n))$; see Appendix A for details.

KdV *Results*

The aim of this experiment is to learn the mapping from image frames to the latent observations, which we approximate with a neural network as described in Section 4.1. Additionally, we wish to estimate the posterior distribution over the unknown drag coefficient ν, for both with drag ($\nu = 1$) and without ($\nu = 0$). The prior distribution $p(\nu)$ (with $\Lambda = \{\nu\}$) implicitly places a prior over a family of KdV PDEs with varying drag coefficients. Since this physical coefficient is positively constrained, we reflect this in the choice of prior such that $\text{supp}(p(\nu)) = \mathbb{R}^+$. For $\nu = 1$ we use a log-normal prior distribution, and log-normal variational approximation,
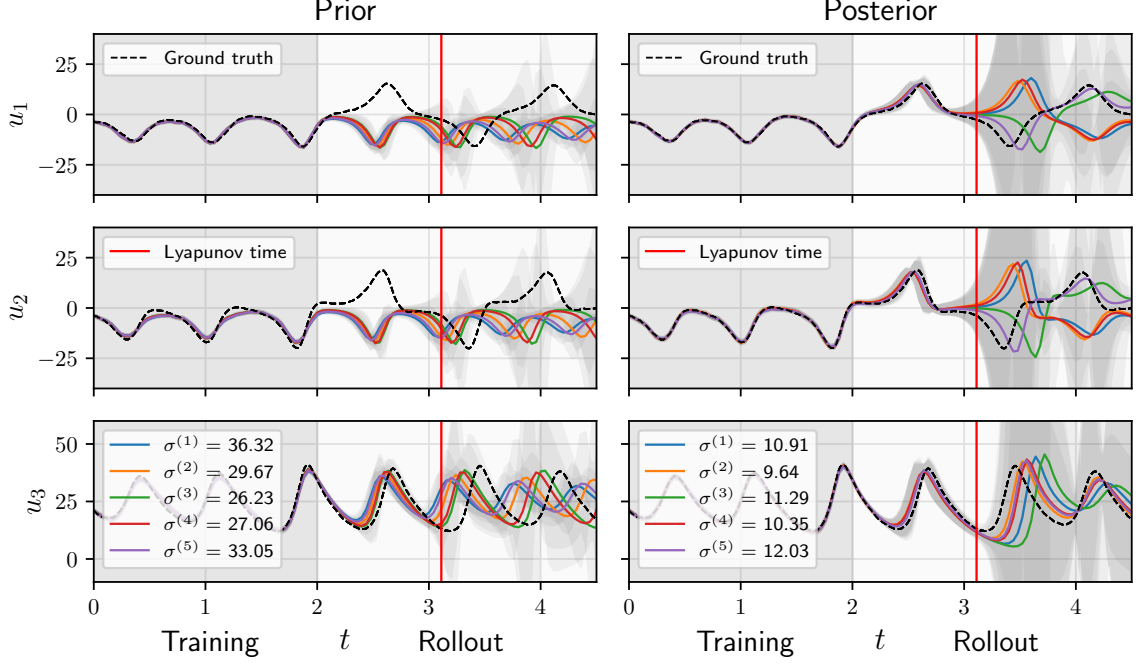
Figure 7: Training time indicated with grey-fill $t = [0, 2)$, and rollout time from $t = [2, 4.5]$. Filtering distributions $p(\mathbf{u}_n | \mathbf{x}_{1:n}^{(i)}, \mathbf{\Lambda}^{(i)}) = \mathcal{N}(\mathbf{m}_n^{(i)}, \mathbf{C}_n^{(i)})$ visualised with $\mathbf{m}_n^{(i)}$ (solid) and ± 2 standard deviations computed from $\mathbf{C}_n^{(i)}$ (grey-bands), alongside the true latents $\mathbf{u}_{1:N}^{\text{true}}$ (dashed). *First column:* samples generated by the prior model, $\mathbf{x}_{1:N}^{(i)} \sim q_\phi(\cdot | \mathbf{y}_{1:N})$, $\mathbf{\Lambda}^{(i)} \sim p(\cdot)$. *Second column:* samples generated from the posterior model, $\mathbf{x}_{1:N}^{(i)} \sim q_{\phi^\star}(\cdot | \mathbf{y}_{1:N})$, $\mathbf{\Lambda}^{(i)} \sim q_{\lambda^\star}(\cdot)$.

which we denote by $p(\nu) = \mathcal{LN}(\mu_0, \sigma_0^2)$, $q_\lambda(\nu) = \mathcal{LN}(\mu_\lambda, \sigma_\lambda^2)$ respectively. For the no-drag case ($\nu = 0$), we use the exponential distribution for both prior and variational approximation, denoted by $p(\nu) = \text{Exp}(\lambda_0)$ and $q_\lambda(\nu) = \text{Exp}(\lambda)$ respectively. For both cases, the choice of prior/posterior pairs provide analytic KL-divergences (see Table 1) for use in the ELBO computation. Following the Lorenz-63 example, the variational approximations are initialised at poorly specified priors to assess the model's ability to correct for this misspecification.

KdV with drag For $\nu = 1$, we set the prior $p(\nu) = \mathcal{LN}(2, 0.5^2)$ and show this alongside the posterior estimate in Fig. 8a. The posterior has contracted around the true value. Fig. 8b shows the latent inference at specific time-points for multiple samples from both the prior and posterior, showing the improvement of the model when adapting the variational posterior jointly with the encoder. Marginalising over the parameter ν using samples from the posterior approximation $q_{\lambda^\star}(\nu)$ allow us to investigate the effect of uncertainty on the parameter estimate. These samples show a spread in mean estimates where the solitons are mixing at time $t = 2.5$; the amplitude here is sensitive to changes in the drag coefficient, which is reflected in the variation of the filtering posterior samples. Fig. 8c shows the resulting inference and rollout for the jointly trained encoder $\phi^\star$ and sample from the posterior $q_{\lambda^\star}(\nu)$. The relative error is computed by normalising the absolute error by the max absolute value of the true solution at each time point, which accounts for the reduction in amplitude of the solution through time so that the growth in relative error can be observed. The latent mean is visualised alongside the ground truth, with the relative error. The relative error is low for the training time, and steadily increasing during rollout. Visualisation of the uncertainty is provided by the standard deviation of each latent dimension through time. The structure of the uncertainty is consistent with the relative error. We note that when initialised with a prior over ν centered on a much less dissipative system, the model struggles to adjust the posterior to a more dissipative system. We believe that since the pre-training on a less dissipative system produces latents that are more separated on the latent
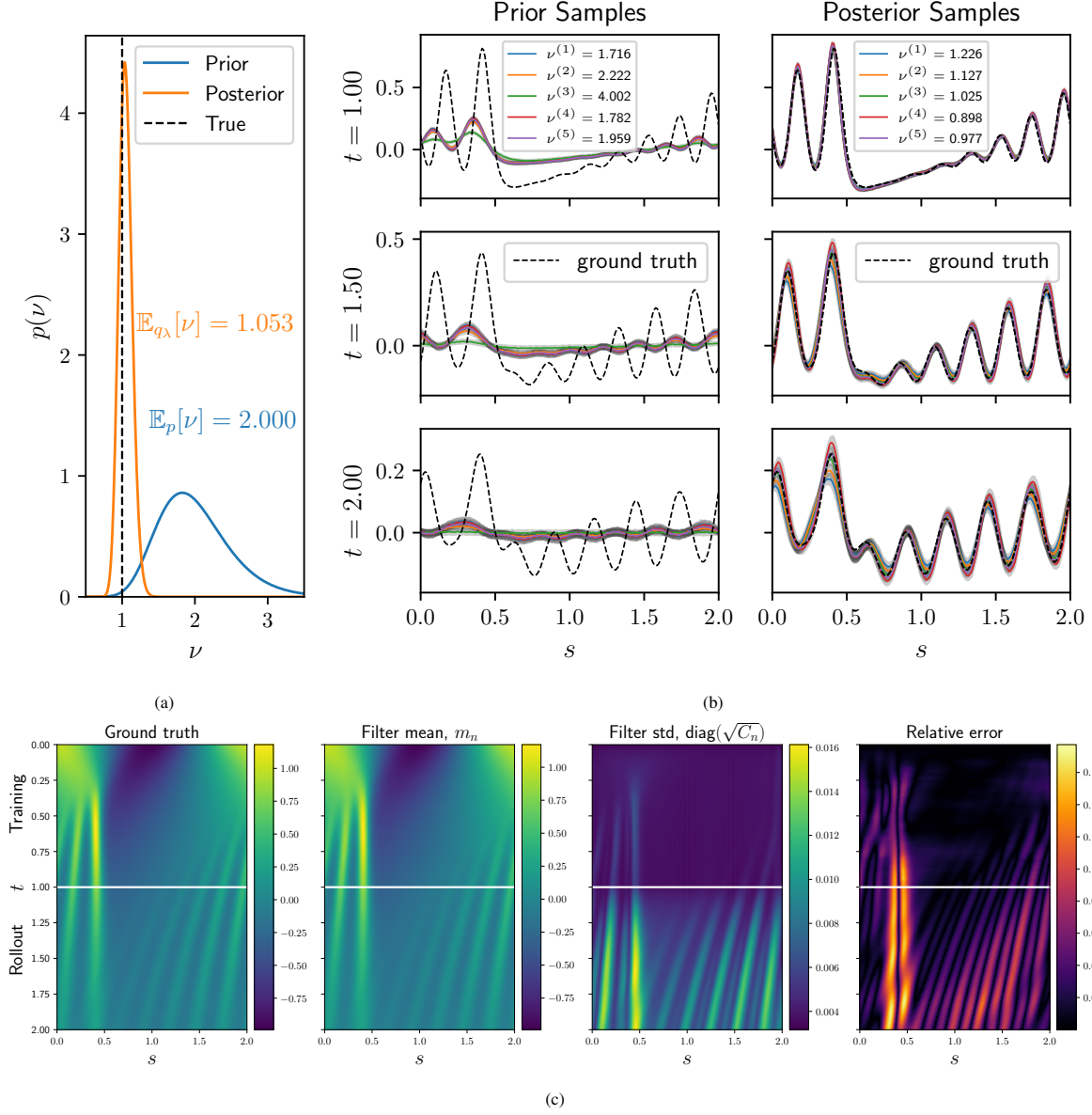
14

Figure 8: Inference results for KDV with drag. (a) Comparison of prior and variational posterior for model parameter $\nu = 1$. (b) Latent filtering distribution shown for prior and posterior model for multiple samples. *Prior model*: $\{\phi, \theta\}$ trained using $\Lambda \sim p(\nu)$. *Posterior model*: $\{\phi^\star, \theta^\star, \lambda^\star\}$ trained using $\Lambda \sim q_\lambda(\nu)$. (c) Rollout prediction. The values of the latent process are shown in space (*horizontal axis*) and time (*vertical axis*). Rollout begins at $t = 1.0$ (*horizontal white line*). Relative error: $|\mathbf{u}_n - \mathbf{u}_n^{\text{true}}| / \max_{1 \le i \le n_u} |u_{i,n}^{\text{true}}|$.

space, the autoencoder is able to still able to reconstruct the video frames due to the flexibility of the NN decoder. For these results, the system is initialised with a large drag coefficient, and early stopping used for regularisation.

No-drag KDV For $\nu = 0$, we set the prior $p(\nu) = \text{Exp}(1.0)$ and show this alongside the posterior estimate in Fig. 9a. The posterior $q_\lambda(\nu) = \text{Exp}(62.2)$ places more density on the true value, $q_{\hat\lambda}(0) > p(0)$, and the resulting latent inference shows the comparison to the prior (see Fig. 9b). Whilst the posterior shows an improved fit to the true latents compared to the prior, there is still a clear mismatch. For $t = 1$ when the rollout begins, the solitons of the latent samples appear out of phase with the true solution, which causes a mismatch when predicted forward
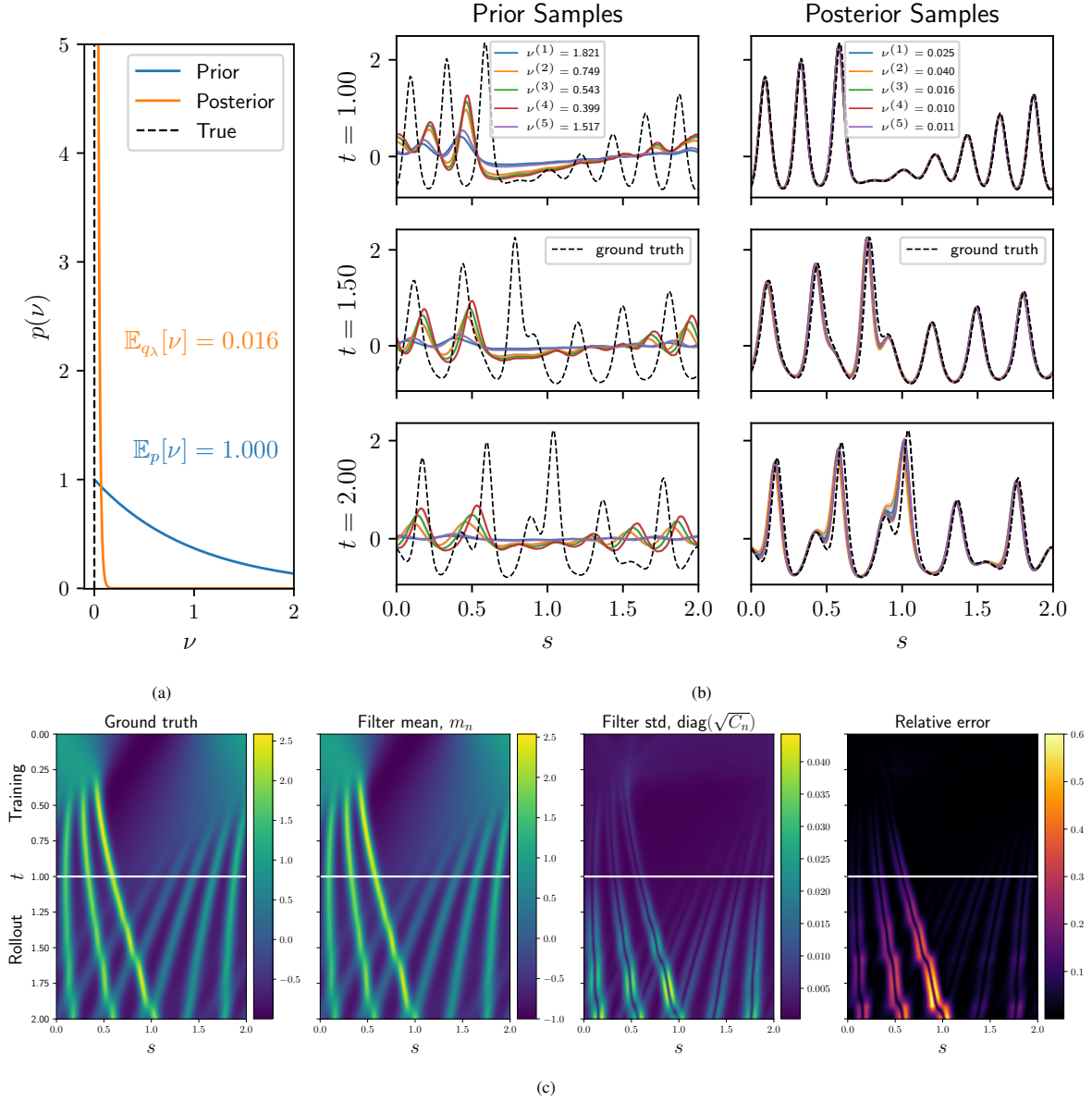
15

Figure 9: Inference results for no-drag KdV. (a) Comparison of prior and variational posterior for model parameter $v = 0$. (b) Latent filtering distribution shown for prior and posterior model for multiple samples. *Prior model*: $\{\phi, \theta\}$ trained using $\mathbf{\Lambda} \sim p(v)$. *Posterior model*: $\{\phi^\star, \theta^\star, \lambda^\star\}$ trained using $\mathbf{\Lambda} \sim q_\lambda(v)$. (c) Rollout prediction. The values of the latent process are shown in space (*horizontal axis*) and time (*vertical axis*). Rollout begins at $t = 1.0$ (*horizontal white line*). Relative error: $|\mathbf{u}_n - \mathbf{u}_n^{\text{true}}|/\max_{1 \le i \le n_u} |u_{i,n}^{\text{true}}|$.

in time. There is also a bias due to the parameter samples being non-zero, and hence predictions begin to diverge from the true solution which does not dissipate. Fig. 9c shows the inference and rollout prediction. The predicted mean has the same structure as the true latents, and the covariance structure reflects the error in the mean.

6. Conclusions

In this work, we introduced the concept of unstructured data assimilation (UDA), where unstructured data is used for state and parameter inference in a known physics model. We proposed a dynamical autoencoder approach to address the UDA problem, and demonstrated the model on three differential equation systems. The model can accurately embed unstructured data into pseudo-observations of the known physical system, and recover the true latent state of the system by filtering the pseudo-observations with the extended Kalman filter (ExKF).

With the inclusion of prior knowledge on the physics, data-efficient learning is achieved. In particular, the extrapolation of the Lorenz-63 system to inferring unseen regions of the attractor based on 50 velocity field observations, and the inversion of the KDV PDE with 100 observed video frames. Accurate parameter estimation has also been demonstrated in both these examples, with the added benefit of parameter uncertainty quantification. We have shown the effect of maginalising out this uncertainty on the latent inference, which results in a mixture of Gaussians filtering distribution that is able to capture variation in the latent solution in areas that are sensitive to model parameters. Comparison of Φ-DVAE was made to a baseline model, the KVAE. This showed our model performing better in terms of Bayesian model evidence and mean-squared-error of reconstruction due to the added inductive bias introduced by the known physical system, compared to an arbitrary learned linear Gaussian state-space model.

Our Φ-DVAE model has demonstrated that state estimation and inversion are possible in the case of UDA, when carefully embedding knowledge about the underlying dynamical system. Further work could extend this methodology to include *direct* measurements, as well as the *indirect*, unstructured observations in order to augment the inference when direct measurements are sparse, or expensive.

References

[1] K. Judd, L. Smith, Indistinguishable states II. The imperfect model scenario, Physica D: Nonlinear Phenomena 196 (2004) 224–242.

[2] G. E. P. Box, Robustness in the Strategy of Scientific Model Building, in: R. L. Launer, G. N. Wilkinson (Eds.), Robustness in Statistics, Academic Press, 1979, pp. 201–236.

[3] R. E. Kalman, A New Approach to Linear Filtering and Prediction Problems, Journal of Basic Engineering 82 (1960) 35–45.

[4] B. D. Anderson, J. B. Moore, Optimal filtering, Englewood Cliffs, N.J. Prentice Hall, 1979.

[5] A. Tarantola, Inverse Problem Theory and Methods for Model Parameter Estimation, Society for Industrial and Applied Mathematics, 2005.

[6] A. M. Stuart, Inverse problems: A Bayesian perspective, Acta Numerica 19 (2010) 451–559.

[7] K. Law, A. Stuart, K. Zygalakis, Data Assimilation: A Mathematical Introduction, volume 62, Springer, Cham, Switerland, 2015.

[8] S. Reich, C. Cotter, Probabilistic forecasting and Bayesian data assimilation, Cambridge University Press, 2015.

[9] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, N. Chopin, On particle methods for parameter estimation in state-space models, Statistical science 30 (2015) 328–351.

[10] G. Storvik, Particle filters for state-space models with the presence of unknown static parameters, IEEE Transactions on signal Processing 50 (2002) 281–289.

[11] M. Bocquet, P. Sakov, Joint state and parameter estimation with an iterative ensemble kalman smoother, Nonlinear Processes in Geophysics 20 (2013) 803–818.

[12] S. Ditlevsen, A. Samson, Estimation in the partially observed stochastic morris–lecar neuronal model with particle filter and stochastic approximation methods, The annals of applied statistics 8 (2014) 674–702.

[13] C. Kravaris, J. Hahn, Y. Chu, Advances and selected recent developments in state and parameter estimation, Computers & chemical engineering 51 (2013) 111–123.

[14] D. Dochain, State and parameter estimation in chemical and biochemical processes: a tutorial, Journal of process control 13 (2003) 801–818.

[15] H. Moradkhani, S. Sorooshian, H. V. Gupta, P. R. Houser, Dual state–parameter estimation of hydrological models using ensemble kalman filter, Advances in water resources 28 (2005) 135–147.

[16] D. M. Blei, A. Kucukelbir, J. D. McAuliffe, Variational Inference: A Review for Statisticians, Journal of the American Statistical Association 112 (2017) 859–877.

[17] D. P. Kingma, M. Welling, Auto-Encoding Variational Bayes, 2014. `arXiv:1312.6114`.

[18] M. Fraccaro, S. Kamronn, U. Paquet, O. Winther, A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning, in: Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

[19] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, X. Alameda-Pineda, Dynamical variational autoencoders: A comprehensive review, Foundations and Trends in Machine Learning 15 (2021) 1–175.

[20] M. Pearce, The gaussian process prior vae for interpretable latent dynamics from pixels, in: Symposium on advances in approximate bayesian inference, 2020, pp. 1–12. Tex.organization: PMLR.

[21] M. Jazbec, M. Ashman, V. Fortuin, M. Pearce, S. Mandt, G. Rätsch, Scalable Gaussian Process Variational Autoencoders, in: Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 3511–3519.

[22] V. Fortuin, D. Baranchuk, G. Rätsch, S. Mandt, Gp-vae: Deep probabilistic time series imputation, in: International conference on artificial intelligence and statistics, PMLR, 2020, pp. 1651–1661.

[23] H. Zhu, C. B. Rodas, Y. Li, Markovian Gaussian Process Variational Autoencoders, 2022. `arXiv:2207.05543`.

[24] J. Hartikainen, S. Sarkka, Kalman filtering and smoothing solutions to temporal Gaussian process regression models, in: 2010 IEEE International Workshop on Machine Learning for Signal Processing, IEEE, Kittila, Finland, 2010, pp. 379–384.

[25] C. Yildiz, M. Heinonen, H. Lahdesmaki, ODE2VAE: Deep generative second order ODEs with Bayesian neural networks, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019.

[26] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural Ordinary Differential Equations, in: Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018.

[27] J. Bayer, C. Osendorfer, Learning stochastic recurrent networks, arXiv preprint arXiv:1411.7610 (2014).

[28] R. G. Krishnan, U. Shalit, D. Sontag, Deep kalman filters, arXiv preprint arXiv:1511.05121 (2015).

[29] M. Karl, M. Soelch, J. Bayer, P. van der Smagt, Deep variational bayes filters: Unsupervised learning of state space models from raw data, in: International Conference on Learning Representations, 2017.

[30] B. Wu, S. Nair, R. Martin-Martin, L. Fei-Fei, C. Finn, Greedy Hierarchical Variational Autoencoders for Large-Scale Video Prediction, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Nashville, TN, USA, 2021, pp. 2318–2328.

[31] J.-Y. Franceschi, E. Delasalles, M. Chen, S. Lamprier, P. Gallinari, Stochastic Latent Residual Video Prediction, in: Proceedings of the 37th International Conference on Machine Learning, PMLR, 2020, pp. 3233–3246.

[32] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, S. Levine, Stochastic Variational Video Prediction, in: International Conference on Learning Representations, 2022.

[33] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, Y. Bengio, A recurrent latent variable model for sequential data, Advances in neural information processing systems 28 (2015).

[34] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (2016) 3932–3937.

[35] K. Champion, B. Lusch, J. N. Kutz, S. L. Brunton, Data-driven discovery of coordinates and governing equations, Proceedings of the National Academy of Sciences 116 (2019) 22445–22451.

[36] R. Lopez, P. J. Atzberger, Variational Autoencoders for Learning Nonlinear Dynamics of Physical Systems, 2021. arXiv:2012.03448.

[37] B. Lusch, J. N. Kutz, S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature communications 9 (2018) 1–10.

[38] S. E. Otto, C. W. Rowley, Linearly Recurrent Autoencoder Networks for Learning Dynamics, SIAM Journal on Applied Dynamical Systems 18 (2019) 558–593.

[39] C. Gin, B. Lusch, S. L. Brunton, J. N. Kutz, Deep learning models for global coordinate transformations that linearise PDEs, European Journal of Applied Mathematics 32 (2021) 515–539.

[40] J. Morton, A. Jameson, M. J. Kochenderfer, F. Witherden, Deep Dynamical Modeling and Control of Unsteady Fluid Flows, in: Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018.

[41] N. Takeishi, Y. Kawahara, T. Yairi, Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition, in: Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

[42] N. B. Erichson, M. Muehlebach, M. W. Mahoney, Physics-informed Autoencoders for Lyapunov-stable Fluid Flow Prediction, 2019. arXiv:1905.10866.

[43] C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, V. S. Pande, Variational encoding of complex dynamics, Physical Review E 97 (2018) 062412.

[44] P. Y. Lu, S. Kim, M. Soljačić, Extracting Interpretable Physical Parameters from Spatiotemporal Systems Using Unsupervised Learning, Physical Review X 10 (2020) 031056.

[45] Y. Yin, V. Le Guen, J. Dona, E. de Bézenac, I. Ayed, N. Thome, P. Gallinari, Augmenting physical models with deep networks for complex dynamics forecasting, Journal of Statistical Mechanics: Theory and Experiment 2021 (2021) 124012.

[46] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-Net: Learning PDEs from Data, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.

[47] E. de Bézenac, A. Pajot, P. Gallinari, Deep learning for physical processes: Incorporating prior scientific knowledge, Journal of Statistical Mechanics: Theory and Experiment 2019 (2019) 124009.

[48] H. Shin, M. Choi, Physics-informed variational inference for uncertainty quantification of stochastic differential equations, Journal of Computational Physics 487 (2023) 112183.

[49] H. Goh, S. Sheriffdeen, J. Wittmer, T. Bui-Thanh, Solving bayesian inverse problems via variational autoencoders, volume 145 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 386–425.

[50] M. Raissi, P. Perdikaris, 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 (2019) 686–707.

[51] N. Wandel, M. Weinmann, M. Neidlin, R. Klein, Spline-pinn: Approaching pdes without data using fast, physics-informed hermite-spline cnns, volume 36, 2022, p. 8529 – 8538.

[52] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81.

[53] L. Yang, X. Meng, G. E. Karniadakis, B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data, Journal of Computational Physics 425 (2021) 109913.

[54] C. Meng, S. Seo, D. Cao, S. Griesemer, Y. Liu, When physics meets machine learning: A survey of physics-informed machine learning, arXiv preprint arXiv:2203.16797 (2022).

[55] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (2021) 422–440.

[56] N. Geneva, N. Zabaras, Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks, Journal of Computational Physics 403 (2020) 109056.

[57] A. Cicirello, Physics-enhanced machine learning: a position paper for dynamical systems investigations, arXiv preprint arXiv:2405.05987 (2024).

[58] C. Lathourakis, A. Cicirello, Physics enhanced sparse identification of dynamical systems with discontinuous nonlinearities, Nonlinear Dynamics (2024) 1–28.

[59] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces with applications to pdes, Journal of Machine Learning Research 24 (2023) 1–97.

[60] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020).

[61] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint arXiv:2111.03794 (2021).

[62] A. Vadeboncoeur, Ö. D. Akyildiz, I. Kazlauskaite, M. Girolami, F. Cirak, Deep probabilistic models for forward and inverse problems in parametric pdes, arXiv preprint arXiv:2208.04856 (2022).

[63] A. Vadeboncoeur, I. Kazlauskaite, Y. Papandreou, F. Cirak, M. Girolami, O. D. Akyildiz, Random grid neural processes for parametric partial differential equations, in: International Conference on Machine Learning, PMLR, 2023, pp. 34759–34778.

[64] M. Girolami, E. Febrianto, G. Yin, F. Cirak, The statistical finite element method (statFEM) for coherent synthesis of observation data and model predictions, Computer Methods in Applied Mechanics and Engineering 375 (2021) 113533.

[65] C. Duffin, E. Cripps, T. Stemler, M. Girolami, Statistical finite elements for misspecified models, Proceedings of the National Academy of Sciences 118 (2021).

[66] C. Duffin, E. Cripps, T. Stemler, M. Girolami, Low-rank statistical finite elements for scalable model-data synthesis, Journal of Computational Physics 463 (2022).

[67] Ö. D. Akyildiz, C. Duffin, S. Sabanis, M. Girolami, Statistical Finite Elements via Langevin Dynamics, SIAM/ASA Journal of Uncertainty Quantification (2022).

[68] C. K. Williams, C. E. Rasmussen, Gaussian Processes for Machine Learning, volume 2, MIT press Cambridge, MA, 2006.

[69] V. Lalchand, C. E. Rasmussen, Approximate inference for fully bayesian gaussian process regression, in: Symposium on Advances in Approximate Bayesian Inference, PMLR, 2020, pp. 1–12.

[70] P. G. Drazin, R. S. Johnson, Solitons: An Introduction, Cambridge Texts in Applied Mathematics, second ed., Cambridge University Press, Cambridge, 1989.

[71] S. C. Brenner, L. R. Scott, The Mathematical Theory of Finite Element Methods, volume 15 of *Texts in Applied Mathematics*, Springer New York, New York, NY, 2008.

[72] V. Thomée, Galerkin Finite Element Methods for Parabolic Problems, number v. 25 in Springer Series in Computational Mathematics, 2nd ed ed., Springer, Berlin ; New York, 2006.

[73] A. H. Jazwinski, Stochastic Processes and Filtering Theory, Academic Press, 1970.

[74] Y. Chen, D. Sanz-Alonso, R. Willett, Autodifferentiable ensemble kalman filters, SIAM Journal on Mathematics of Data Science 4 (2022) 801–833.

[75] A. Corenflos, J. Thornton, G. Deligiannidis, A. Doucet, Differentiable particle filtering via entropy-regularized optimal transport, in: International Conference on Machine Learning, PMLR, 2021, pp. 2100–2111.

[76] D. P. Kingma, M. Welling, et al., An introduction to variational autoencoders, Foundations and Trends® in Machine Learning 12 (2019) 307–392.

[77] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2017. `arXiv:1412.6980`.

[78] C. Boncelet, Chapter 7 - image noise models, in: The Essential Guide to Image Processing, Academic Press, Boston, 2009, pp. 143–167. URL: `https://www.sciencedirect.com/science/article/pii/B978012374457900007X`. doi:`https://doi.org/10.1016/B978-0-12-374457-9.00007-X`.

[79] P. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, Applications of Mathematics, Springer Berlin Heidelberg, 1992.

[80] B. Øksendal, Stochastic differential equations, in: Stochastic Differential Equations, Springer, 2003, pp. 65–84.

[81] E. N. Lorenz, Deterministic nonperiodic flow, Journal of atmospheric sciences 20 (1963) 130–141.

[82] A. Wolf, J. B. Swift, H. L. Swinney, J. A. Vastano, Determining lyapunov exponents from a time series, Physica D: Nonlinear Phenomena 16 (1985) 285–317.

[83] R. Shaw, Strange attractors, chaotic behavior, and information flow, Zeitschrift für Naturforschung A 36 (1981) 80–112.

[84] N. J. Zabusky, M. D. Kruskal, Interaction of "Solitons" in a Collisionless Plasma and the Recurrence of Initial States, Physical Review Letters 15 (1965) 240–243.

[85] A. Debussche, J. Printems, Numerical simulation of the stochastic Korteweg–de Vries equation, Physica D: Nonlinear Phenomena 134 (1999) 200–226.

[86] W. E. Schiesser, The Numerical Method of Lines: Integration of Partial Differential Equations, Academic Press, 1991.

[87] L. Evans, Partial Differential Equations, volume 19 of *Graduate Studies in Mathematics*, second ed., American Mathematical Society, Providence, Rhode Island, 2010.

[88] G. Evensen, The Ensemble Kalman Filter: Theoretical formulation and practical implementation, Ocean Dynamics 53 (2003) 343–367.

[89] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, Statistics and Computing 10 (2000) 197–208.

Appendix A. Numerical Details and Network Architectures

Lorenz-63 experiment.

- Time-series length: $N = 300$. Training length: 50, Test length: 250
- Input: $n_y = 200$.
- Pseudo-observations: $n_x = 1$.
- Latent $n_u = 3$.
- Decoder: $p_{\mathbf{w}}(\mathbf{y}_n|x_n) = \mathcal{N}(\mathbf{w}x_n, \eta^2\mathbf{I})$, $\eta = 0.1$.
- Encoder: $q_{\mathbf{w}}(x_n|\mathbf{y}_n) = \mathcal{N}((\mathbf{w}^\top\mathbf{w})^{-1}\mathbf{w}^\top\mathbf{y}_n, \eta^2(\mathbf{w}^\top\mathbf{w})^{-1})$
- Latent initial condition: $\mathbf{u}_0 = [-3.7277, -3.8239, 21.1507]^\top$
- Latent noise processes: $\mathbf{L} = \mathrm{diag}(1.0^2)$, $\mathbf{R} = \mathrm{diag}(0.4^2)$.
- Latent discretisation: Euler-Maruyama, $dt = 0.001$.
- Model parameter prior: $p(\sigma) = \mathcal{N}(30, 5^2)$.
- Optimiser: Adam, learning rate $= 10^{-2}$.
- Training Epochs: 2000.

To generate the data, we simulate the Lorenz SDE with $dt = 0.001$ and take pseudo-observations x_n every 40 time-steps of the latent system, for a total of $N = 300$ with $\Delta_t = 0.04$. Velocity measurements are taken in the s_1 and s_2 directions over a regular 10×10 grid on the domain $s_1, s_2 \in [-4, 4]$, via the streamfunction $\psi(s_1, s_2, t)$. These measurements are flattened to the data vector $\mathbf{y}_n \in \mathbb{R}^{n_y}$, $n_y = 200$. Parameters for data generation are $\Lambda = \{\sigma = 10, r = 28, b = 8/3\}$.

Advection equation experiment.

- Time-series length: $N = 200$.
- Input: $n_y = 784$.
- Pseudo-observations: $n_x = 64$.
- Latent $n_u = 64$.
- Decoder: $p_\theta(\mathbf{y}_n|\mathbf{x}_n) = \mathrm{Bern}(\mu_\theta(\mathbf{x}_n))$
- $\mu_\theta(\cdot)$: MLP, two fully connected hidden layers with dimension 128
- Encoder: $q_\phi(\mathbf{x}_n|\mathbf{y}_n) = \mathcal{N}(\mu_\phi(\mathbf{y}_n), \sigma_\phi(\mathbf{y}_n))$
- $\mu_\phi(\cdot), \log\sigma_\phi(\cdot)$: MLP, two fully connected hidden layers with dimension 128
- Neural Network activations: LeakyReLU, negative slope $= 0.01$
- Latent initial condition: $u(s, 0) = \exp(-2(s - 2.5)^2)$, $s \in [0, 5]$.
- Latent noise processes: $\rho = 0.1$, $\ell = 0.1$, $\mathbf{R} = \mathrm{diag}(0.1^2)$.
- Latent discretisation: FEM, $C^0([0, 1])$ polynomial trial/test functions, Crank-Nicolson time discretisation, $dt = 0.02$.

- Optimiser: Adam, learning rate = 0.001.

- Salt-and-pepper noise, probability of pixel information loss: 0.02

To generate the data, we evaluate the analytical solution at every $dt = 0.02$, taking pseudo-observations every 10 time-steps for $\Delta_t = 0.2$ and $N = 200$. Latent dimensions are $n_u = 64$ and $n_x = 64$, with each image 28×28 pixels. These images are then flattened to vectors $\mathbf{y}_n \in [0, 1]^{n_y}$, with $n_y = 784$.

KDV equation experiment.

- Time-series length: $N = 100$.

- Input: $n_y = 1792$.

- Pseudo-observations: $n_x = 40$.

- Latent $n_u = 600$.

- Decoder: $p_\theta(\mathbf{y}_n|\mathbf{x}_n) = \text{Bern}(S(f_\theta(\mathbf{x}_n)))$

- $f_\theta(\cdot)$: MLP, three fully connected hidden layers with dimension 128

- Encoder: $q_\phi(\mathbf{x}_n|\mathbf{y}_n) = \mathcal{N}(\mu_\phi(\mathbf{y}_n), \sigma_\phi(\mathbf{y}_n))$

- $\mu_\phi(\cdot), \log\sigma_\phi(\cdot)$: MLP, three fully connected hidden layers with dimension 128

- Neural Network activations: tanh

- Fixed parameters: $\alpha = 1.0, \beta = 0.022^2$

- Unknown parameter (with drag): $\nu = 1.0$.

- Unknown parameter (no-drag): $\nu = 0$.

- Latent initial condition: $u(s, 0) = \cos(\pi s), s \in [0, 2]$.

- Latent noise processes: $\rho = 0.01, \ell = 0.2, \mathbf{R} = \text{diag}(0.02^2)$.

- Latent discretisation: Petrov-Galerkin approach of Debussche and Printems [85]: $C^0([0, 2])$ polynomial trial functions, Crank-Nicolson time discretisation, $dt = 0.01$.

- Parameter prior (with drag): $p(\mathbf{\Lambda}) = \mathcal{LN}(2, 0.5^2)$.

- Parameter prior (no-drag): $p(\mathbf{\Lambda}) = \text{Exp}(1.0)$.

- Optimiser: Adam, learning rate = 0.005.

- Salt-and-pepper noise, probability of pixel information loss: 0.02

To generate the data we simulate the KDV equation with a high-fidelity in space and time to ensure accurate latent data, and avoid inverse crimes. For the data used in comparison, we subsample this simulation with $dt = 0.01$, observing every timestep for $\Delta_t = 0.01$, and we take $N = 100$ observations $\mathbf{y}_n$, with $\mathbf{y}_n \in [0, 1]^{n_y}$. Each $\mathbf{y}_n$ is a flattened image of dimension $n_y = 64 \times 28 = 1792$, and we encode to pseudo-observations $\mathbf{x}_n$ of dimension $n_x = 40$. The latent state dimension is $n_u = 600$.

Neural network decoding/encoding. Variational autoencoders use neural networks to parameterise the encoding distribution $q_\phi(\mathbf{x}|\mathbf{y})$, and the probablisitic decoder $p_\theta(\mathbf{y}|\mathbf{x})$. A common choice of variational distribution is the Gaussian, which uses a NN to parameterise the mean $\mu_\phi(\mathbf{y})$, where $\mu_\phi \colon \mathbb{R}^{n_y} \to \mathbb{R}^{n_x}$, and another to parametrise the diagonal of the covariance. Parameterising the log-variance allows for both positive and negative neural network output — the values are exponentiated to calculate the positively constrained variances.

$$q_\phi(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mu_\phi(\mathbf{y}), \text{diag}(\sigma_\phi(\mathbf{y}))\right)$$

The decoder attempts to reconstruct the original data $\mathbf{y}$ from the latents $\mathbf{x}$. A NN decoder maps the latents to a reconstruction $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$, where $f_\theta \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ is the NN mapping parameterised by θ. The probabilistic decoder depends on the form of the data. In this work, we use a Gaussian distribution for the velocity field data, and a Bernoulli distribution for the video data. When reconstructing video frames, an element-wise sigmoid function is used as the final layer of the neural network, in order to output grayscale pixel values, i.e. $S(y) = 1/(1 + \exp(-y))$, where $S \colon \mathbb{R}^{n_y} \to [0, 1]^{n_y}$. The two examples of unstructured data, real-valued velocity field data, and binary pixel data can be reconstructed as follows:

Real-valued data, $\mathbf{y} \in \mathbb{R}^{n_y}$, we reconstruct $\hat{\mathbf{y}} \in \mathbb{R}^{n_y}$, where $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$. Often in practice an independent noise model is used, with standard deviation σ, giving the density and log-density as follows:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}; \hat{\mathbf{y}}, \sigma^2 \mathbf{I}\right)$$

$$\log p_\theta(\mathbf{y}|\mathbf{x}) = -\frac{n_y}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

Binary pixel data, $\mathbf{y} \in \{0, 1\}^{n_y}$ we reconstruct $\hat{\mathbf{y}} \in [0, 1]^{n_y}$, where $\hat{\mathbf{y}} = S(f_\theta(\mathbf{x}))$. The density and log-density are given as follows:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \text{Bern}(\mathbf{y}; \hat{\mathbf{y}})$$

$$\log p_\theta(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{n_y} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Linear decoding/encoding. Unlike the nonlinear NN decoders, if linear data generation is assumed from $\mathbf{x}$ to $\mathbf{y}$, we can invert this in order to encode our data. With a linear decoder of the form:

$$p_{\mathbf{A}}(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{A}\mathbf{x}, \eta^2 I).$$

In this case, we use the "inverted" linear decoder given by:

$$q_{\mathbf{A}}(\mathbf{x}|\mathbf{y}) = \mathcal{N}((\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{y}, \eta^2 (\mathbf{A}^\top \mathbf{A})^{-1}).$$

By selecting the encoder appropriately, the space of parameterised variational distributions can be restricted to align with our beliefs about the data generation process.

Appendix B. Full Variational Framework

We derive the approximate ELBO for joint estimation of model parameters $\mathbf{\Lambda}$, and autoencoder parameters ϕ, θ. We start by writing the evidence

$$p(\mathbf{y}_{1:N}) = \int p(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}, \mathbf{y}_{1:N}) d\mathbf{x}_{1:N} d\mathbf{u}_{1:N} d\mathbf{\Lambda}.$$

We maximise $\log p(\mathbf{y}_{1:N})$ as

$$
\begin{aligned}
\log p(\mathbf{y}_{1:N}) &= \log \int p(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}, \mathbf{y}_{1:N}) d\mathbf{x}_{1:N} d\mathbf{u}_{1:N} d\mathbf{\Lambda} \\
&= \log \int \frac{p(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}, \mathbf{y}_{1:N})}{q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N})} q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N}) d\mathbf{x}_{1:N} d\mathbf{u}_{1:N} d\mathbf{\Lambda} \\
&\geq \int \log \left[\frac{p(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}, \mathbf{y}_{1:N})}{q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N})} \right] q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \mathbf{\Lambda}|\mathbf{y}_{1:N}) d\mathbf{x}_{1:N} d\mathbf{u}_{1:N} d\mathbf{\Lambda} \\
&= \text{ELBO},
\end{aligned}
$$

where the third line follows from the application of Jensen's inequality. Our generative model determines the factorisation of the joint distribution, given in (1):

$$p(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \boldsymbol{\Lambda}, \mathbf{y}_{1:N}) = p(\mathbf{y}_{1:N}|\mathbf{x}_{1:N}, \mathbf{u}_{1:N}, \boldsymbol{\Lambda})p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N}, \boldsymbol{\Lambda})p(\mathbf{u}_{1:N}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})$$
$$= p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N}, \boldsymbol{\Lambda})p(\mathbf{u}_{1:N}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}).$$

Next, we plug this factorised distribution into the ELBO and obtain

$$\text{ELBO} = \int \log\left[\frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N}, \boldsymbol{\Lambda})p(\mathbf{u}_{1:N}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})}{q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \boldsymbol{\Lambda}|\mathbf{y}_{1:N})}\right] \times q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \boldsymbol{\Lambda}|\mathbf{y}_{1:N}) \, d\mathbf{x}_{1:N}d\mathbf{u}_{1:N}d\boldsymbol{\Lambda}.$$

The family of distributions which we use to approximate the posterior is described below. We assume a factorisation based on the model into variational encoding $q_\phi(\cdot)$, a full latent state posterior $q_\nu(\cdot)$, and the variational approximation to the parameter posterior $q_\lambda(\cdot)$:

$$q(\mathbf{u}_{1:N}, \mathbf{x}_{1:N}, \boldsymbol{\Lambda}|\mathbf{y}_{1:N}) = q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})$$
$$= q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})\prod_{n=1}^{N} q_\phi(\mathbf{x}_n|\mathbf{y}_n).$$

The second line demonstrates the amortized structure of the autoencoder, where the same encoding parameters are shared across datapoints. We can then substitute this expression into our ELBO and obtain

$$\text{ELBO} = \int \log\left[\frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N}, \boldsymbol{\Lambda})p(\mathbf{u}_{1:N}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})}{q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})}\right] \times q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})d\mathbf{x}_{1:N}d\mathbf{u}_{1:N}d\boldsymbol{\Lambda}.$$

Assuming the variational posterior is the exact filtering posterior, i.e., $q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda}) = p(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})$ then applying Bayes rule

$$\frac{p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N}, \boldsymbol{\Lambda})p(\mathbf{u}_{1:N}|\boldsymbol{\Lambda})}{q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})} = p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda})$$

leads to a simplification of ELBO in terms of the marginal likelihood $p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda})$ of the state-space model. Substituting this expression leads to

$$\text{ELBO} = \int \log\left[\frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})}{q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})q_\lambda(\boldsymbol{\Lambda})}\right]$$
$$\times q(\mathbf{u}_{1:N}|\mathbf{x}_{1:N}, \boldsymbol{\Lambda})q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})d\mathbf{x}_{1:N}d\mathbf{u}_{1:N}d\boldsymbol{\Lambda}$$
$$= \int \log\left[\frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})}{q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})}\right]q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})d\mathbf{x}_{1:N}$$
$$+ \int \left[\log p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda}) + \log\frac{p(\boldsymbol{\Lambda})}{q_\lambda(\boldsymbol{\Lambda})}\right]q_\lambda(\boldsymbol{\Lambda})q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})d\mathbf{x}_{1:N}d\boldsymbol{\Lambda}$$
$$= \mathbb{E}_{q_\phi}\left[\log\frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})}{q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})}\right] + \mathbb{E}_{q_\phi}\left[\mathbb{E}_{q_\lambda}\left[\log p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda}) + \log\frac{p(\boldsymbol{\Lambda})}{q_\lambda(\boldsymbol{\Lambda})}\right]\right]$$
$$= \mathbb{E}_{q_\phi}\left[\log\frac{p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N})}{q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N})}\right] + \mathbb{E}_{q_\phi}\left[\mathbb{E}_{q_\lambda}\left[\log p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda})\right]\right] - \text{KL}(q_\lambda \parallel p).$$

Using a single MC samples $\mathbf{x}_{1:N} \sim q_\phi(\cdot|\mathbf{y}_{1:N})$, and $\boldsymbol{\Lambda} \sim q_\lambda(\cdot)$ to approximate the expectations, we can write the approximate ELBO, $\mathcal{L}(\phi, \theta, \lambda)$ as

$$\mathcal{L}(\theta, \phi, \lambda) = \log p_\theta(\mathbf{y}_{1:N}|\mathbf{x}_{1:N}) - \log q_\phi(\mathbf{x}_{1:N}|\mathbf{y}_{1:N}) + \log p(\mathbf{x}_{1:N}|\boldsymbol{\Lambda}) - \text{KL}(q_\lambda \parallel p).$$

See Algorithm 1 for pseudo-code, and Table 1 for distributions that provide analytic KL-divergence.

Appendix C. Further Details on the Dynamic Model

In this work we take the latent dynamical model to be a stochastic ODE or PDE. For an ODE this follows from a standard SDE [80], given by

$$d\mathbf{u} = f_\Lambda(\mathbf{u}, t; \Lambda)dt + \mathbf{L}(t)d\mathbf{W}(t),$$

where $\mathbf{u} := \mathbf{u}(t) \in \mathbb{R}^{n_u}$, $t \in [0, T]$, $f_\Lambda \colon \mathbb{R}^{n_u} \times [0, T] \to \mathbb{R}^{n_u}$, $\mathbf{L} \colon [0, T] \to \mathbb{R}^{n_u \times n_u}$. The noise process $\mathbf{W}(t)$ is a standard vector Brownian motion. The diffusion term $\mathbf{L}(t)$ can be used to describe any *a priori* correlation in the error process dimensions. As stated in the main text, this error process is taken to represent possibly misspecified/unknown physics, which may have been omitted when specifying the model. Discretisation with an explicit Euler-Maruyama scheme [79] gives,

$$\mathbf{u}_n = \mathbf{u}_{n-1} + \Delta_t f_{n-1}(\mathbf{u}_{n-1}; \Lambda) + \mathbf{L}_{n-1}\Delta\mathbf{W}_{n-1}, \quad \Delta\mathbf{W}_{n-1} \sim \mathcal{N}(\mathbf{0}, \Delta_t\mathbf{I}),$$

where $\mathbf{u}_n := \mathbf{u}(n\Delta_t)$, $f_n(\cdot; \Lambda) = f_\Lambda(\cdot, n\Delta_t)$, and so on. This gives a transition density

$$p(\mathbf{u}_n|\mathbf{u}_{n-1}, \Lambda) = \mathcal{N}(\mathbf{u}_{n-1} + \Delta_t f_{n-1}(\mathbf{u}_{n-1}; \Lambda), \Delta_t\mathbf{L}_{n-1}\mathbf{L}_{n-1}^\top),$$

defining a Markov model on the now discretised state vector $\mathbf{u}_n$. To align with the notation introduced in the main text, this gives:

$$p(\mathbf{u}_n|\mathbf{u}_{n-1}, \Lambda) = \mathcal{N}(\mathcal{M}(\mathbf{u}_{n-1}), \mathbf{Q}),$$
$$\mathcal{M}(\mathbf{u}_{n-1}) := \mathbf{u}_{n-1} + \Delta_t f_{n-1}(\mathbf{u}_{n-1}; \Lambda), \quad \mathbf{Q} := \Delta_t\mathbf{L}_{n-1}\mathbf{L}_{n-1}^\top.$$

Due to the structure of the sᴛᴀᴛFEM discretisation, the fully-discretised underlying model is of the same mathematical form as this ODE case. The difference lies in the dynamics being defined from either a PDE or ODE system. In common cases, a lower dimensional state vector, $\mathbf{u}_n$, typically results for the ODE case in comparison to the PDE case. For the PDE case, entries of the state vector $\mathbf{u}_n$ are coefficients of the finite element basis functions.

For the PDE case, the derivation is similar, with an additional step pre-time-discretisation to spatially discretise the system. This yields a method-of-lines approach [86]. As in the main text, we consider a generic nonlinear PDE system of the form

$$\partial_t u + L_\Lambda u + F_\Lambda(u) = f + \dot{\xi}, \quad \dot{\xi} \sim \mathcal{GP}(0, \delta(t - t') \cdot k(\mathbf{s}, \mathbf{s}')), \tag{C.1}$$

where $u := u(\mathbf{s}, t)$, $\xi := \xi(\mathbf{s}, t)$, $f := f(\mathbf{s})$, $\mathbf{s} \in \Omega \subset \mathbb{R}^d$, and $t \in [0, T]$. The operators L_Λ and $F_\Lambda(\cdot)$ are linear and nonlinear differential operators, respectively. The process $\dot{\xi}$ is the derivative of a function-valued Wiener process, whose increments are given by a Gaussian process with the covariance kernel $k(\cdot, \cdot)$. In our examples, we use the squared-exponential covariance function [68]

$$k(\mathbf{s}, \mathbf{s}') = \rho^2 \exp\left(-\frac{\|\mathbf{s} - \mathbf{s}'\|_2^2}{2\ell^2}\right).$$

Hyperparameters $\{\rho, \ell\}$ are always assumed to be known, being set *a priori*. Further work investigating inference of these hyperparameters is of interest.

As stated in the main text we discretise the linear time-evolving PDE following the sᴛᴀᴛFEM as in Duffin et al. [65], for which we refer to for the full details of this approach. In brief, we discretise spatially with finite elements (see, e.g., Brenner and Scott [71], Thomée [72], for standard references) then temporally via finite differences. We first multiply (C.1) with a sufficiently smooth test function $v \in V$, where V is an appropriate function space (e.g. the $H_0^1(\Omega)$ Sobolev space [87]) and integrate over the domain Ω to give the weak form [71]

$$\langle \partial_t u, v \rangle + \mathcal{A}_\Lambda(u, v) + \langle F_\Lambda(u), v \rangle = \langle f, v \rangle + \langle \dot{\xi}, v \rangle, \quad \forall v \in V.$$

Recall that $\mathcal{A}_\Lambda(\cdot, \cdot)$ is the bilinear form generated from the linear operator L_Λ, and

$$\langle f, g \rangle = \int_\Omega f(\mathbf{s}) g(\mathbf{s}) \, d\mathbf{s},$$

the $L^2(\Omega)$ inner product.

Next we introduce a discrete approximation to the domain, $\Omega_h \subseteq \Omega$, having vertices $\{\mathbf{s}_j\}_{j=1}^{n_h}$. This is parameterised by h which indicates the degree of mesh-refinement. We now introduce a finite-dimensional set of polynomial basis functions $\{\phi_j(\mathbf{s})\}_{j=1}^{n_u}$, such that $\phi_i(\mathbf{s}_j) = \delta_{ij}$. In this work these are exclusively the $C^0(\Omega)$ linear polynomial "hat" basis functions. This gives the finite-dimensional function space $V_h = \text{span}\{\phi_j(\mathbf{s})\}_{j=1}^{n_u}$, which is the space we look for solutions in. Next, we rewrite u and v in terms of these basis functions: $u_h(\mathbf{s}, t) = \sum_{j=1}^{n_u} u_j(t) \phi_j(\mathbf{s})$ and $v_h(\mathbf{s}, t) = \sum_{j=1}^{n_u} v_j(t) \phi_j(\mathbf{s})$. As the weak form must hold for all $v_h \in V_h$, this is equivalent to holding for all ϕ_j. Thus, the weak form can now be rewritten in terms of this set of basis functions

$$\langle \partial_t u_h, \phi_j \rangle + \mathcal{A}_\Lambda(u_h, \phi_j) + \langle F_\Lambda(u_h), \phi_j \rangle = \langle f, \phi_j \rangle + \langle \dot{\xi}, \phi_j \rangle, \quad j = 1, \ldots, n_u.$$

Note that, in general, u_h and v_h do not necessarily have to be defined on the same function space, but as we use the linear basis functions in this work we stick with this here.

As stated in the main text, this is an SDE over the FEM coefficients $\mathbf{u}(t) = (u_1(t), \ldots, u_{n_u}(t))^\top$, given by

$$\mathbf{M} \frac{d\mathbf{u}}{dt} + \mathbf{A}\mathbf{u} + \mathcal{F}(\mathbf{u}) = \mathbf{b} + \dot{\xi}, \quad \dot{\xi}(t) \sim \mathcal{N}(\mathbf{0}, \delta(t - t') \cdot \mathbf{G})$$

where $\mathbf{M}_{ij} = \langle \phi_i, \phi_j \rangle$, $\mathbf{A}_{ij} = \mathcal{A}_\Lambda(\phi_i, \phi_j)$, $\mathcal{F}(\mathbf{u})_j = \langle F_\Lambda(u_h), \phi_j \rangle$, $\mathbf{b}_j = \langle f, \phi_j \rangle$, and $\mathbf{G}_{ij} = \langle \phi_i, \langle k(\cdot, \cdot), \phi_j \rangle \rangle$. Letting $\mathbf{G} = \mathbf{L}\mathbf{L}^\top$ we can then write this in the familiar notation as above

$$\mathbf{M}\,d\mathbf{u} + \mathbf{A}\mathbf{u}\,dt + \mathcal{F}(\mathbf{u})\,dt = \mathbf{b}\,dt + \mathbf{L}\,d\mathbf{W}(t),$$

from which an Euler-Maruyama time discretisation gives

$$\mathbf{u}_n = (\mathbf{I} - \Delta_t \mathbf{M}^{-1} \mathbf{A})\mathbf{u}_{n-1} - \Delta_t \mathbf{M}^{-1} \mathcal{F}(\mathbf{u}_{n-1}) + \Delta_t \mathbf{M}^{-1}\mathbf{b} + \mathbf{M}^{-1}\mathbf{L}\Delta\mathbf{W}_{n-1},$$

where $\Delta\mathbf{W}_{n-1} \sim \mathcal{N}(0, \Delta_t \mathbf{I})$, eventually defining a transition model of the form

$$p_\Lambda(\mathbf{u}_n | \mathbf{u}_{n-1}) = \mathcal{N}\left((\mathbf{I} - \Delta_t \mathbf{M}^{-1}\mathbf{A})\mathbf{u}_{n-1} - \Delta_t \mathbf{M}^{-1}\mathcal{F}(\mathbf{u}^{n-1}) + \Delta_t \mathbf{M}^{-1}\mathbf{b}, \Delta_t \mathbf{M}^{-1}\mathbf{G}\mathbf{M}^{-\top}\right). \quad \text{(C.2)}$$

Note that also that the STATFEM methodology also allows for implicit discretisations which may be desirable for time-integrator stability. The transition equations for these approaches can be written out in closed form, yet although they give Markovian transition models, the resultant transition densities $p(\mathbf{u}_n | \mathbf{u}_{n-1}, \Lambda)$ are not necessarily Gaussian due to the nonlinear dynamics being applied to the current state $\mathbf{u}_n$. Letting $\mathbf{e}_{n-1} = \mathbf{L}\Delta\mathbf{W}_{n-1} \sim \mathcal{N}(\mathbf{0}, \Delta_t \mathbf{G})$, then the implicit Euler is

$$\mathbf{M}(\mathbf{u}_n - \mathbf{u}_{n-1}) + \Delta_t \mathbf{A}\mathbf{u}_n + \Delta_t \mathcal{F}(\mathbf{u}_n) + \Delta_t \mathbf{b} = \mathbf{e}_{n-1}, \quad \text{(C.3)}$$

and the Crank-Nicolson is

$$\mathbf{M}(\mathbf{u}_n - \mathbf{u}_{n-1}) + \Delta_t \mathbf{A}\mathbf{u}_{n-1/2} + \Delta_t \mathcal{F}(\mathbf{u}_{n-1/2}) + \Delta_t \mathbf{b} = \mathbf{e}_{n-1}, \quad \text{(C.4)}$$

where $\mathbf{u}_{n-1/2} = (\mathbf{u}_n + \mathbf{u}_{n-1})/2$. Furthermore, to compute the marginal measure $p(\mathbf{u}_n | \Lambda)$ this also requires integrating over the previous solution $\mathbf{u}_{n-1}$; again due to nonlinear dynamics this will not necessarily be Gaussian.

In each of these cases, therefore, the transition equation is

$$\mathcal{M}(\mathbf{u}_n, \mathbf{u}_{n-1}) = \mathbf{e}_{n-1},$$

where we take, for the implicit Euler,

$$\mathcal{M}(\mathbf{u}_n, \mathbf{u}_{n-1}) = \mathbf{M}(\mathbf{u}_n - \mathbf{u}_{n-1}) + \Delta_t \mathbf{A}\mathbf{u}_n + \Delta_t \mathcal{F}(\mathbf{u}_n) + \Delta_t \mathbf{b}$$

and, for the Crank-Nicolson,

$$\mathcal{M}(\mathbf{u}_n, \mathbf{u}_{n-1}) = \mathbf{M}(\mathbf{u}_n - \mathbf{u}_{n-1}) + \Delta_t \mathbf{A}\mathbf{u}_{n-1/2} + \Delta_t \mathcal{F}(\mathbf{u}_{n-1/2}) + \Delta_t \mathbf{b}.$$

In practice due to conservative properties of the Crank-Nicolson discretisation we use this for all our models in this work.

Discretised solutions $\mathbf{u}_n$ are mapped at time n to "pseudo-observations" via the observation process

$$\mathbf{x}_n = \mathbf{H}\mathbf{u}_n + \mathbf{r}_n, \quad \mathbf{r}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}).$$

This observation process has the density $p_\nu(\mathbf{x}_n|\mathbf{u}_n)$ where $\nu = \{\mathbf{H}, \mathbf{R}\}$. As stated in the main text, the pseudo-observation operator $\mathbf{H}$ and observational covariance $\mathbf{R}$ are assumed known in this work. We typically use a diagonal covariance, setting $\mathbf{R} = \sigma^2\mathbf{I}$. In the PDE case, for a given statFEM discretisation as above, these pseudo-observations are assumed to be taken on a user-specified grid, given by $\mathbf{x}_{\mathrm{obs}} \in \mathbb{R}^{n_x}$. The pseudo-observation operator thus acts as an interpolant, such that

$$\mathbf{H}\mathbf{u}_n = \left[u_h(\mathbf{x}_{\mathrm{obs}}^1, n\Delta_t), u_h(\mathbf{x}_{\mathrm{obs}}^2, n\Delta_t), \ldots, u_h(\mathbf{x}_{\mathrm{obs}}^{n_x}, n\Delta_t) \right]^\top.$$

For the ODE case, we have worked with observation operators that extract relevant entries from the state vector. The pseudo-observations are mapped to high-dimensional observed data through a possibly nonlinear observation model which has the probability density $p_\theta(\mathbf{y}_n|\mathbf{x}_n)$. Recall that in this, θ are neural network parameters. This defines the decoding component of our model (see Figure 1).

Nonlinear Filtering for Latent State Estimation. To perform state inference given a set of pseudo-observations we use the extended Kalman filter (ExKF). The ExKF constructs an approximate Gaussian posterior distribution via linearising about the nonlinear model $\mathcal{M}(\cdot)$. The action of the nonlinear $\mathcal{M}(\cdot)$ is approximated via tangent linear approximation. We will derive our filter in the general context of a nonlinear Gaussian SSM given by

$$\begin{aligned} \text{Transition:} \quad & \mathcal{M}(\mathbf{u}_n, \mathbf{u}_{n-1}) = \mathbf{e}_{n-1}, \quad \mathbf{e}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \\ \text{Observation:} \quad & \mathbf{x}_n = \mathbf{H}\mathbf{u}_n + \mathbf{r}_n, \quad \mathbf{r}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \end{aligned}$$

This allows for the use of implicit time-integrators and subsumes the derivation for the explicit case. We assume that at the previous timestep an approximate Gaussian posterior has been obtained, $p(\mathbf{u}_{n-1}|\mathbf{x}_{1:n-1}, \mathbf{\Lambda}) = \mathcal{N}(\mathbf{m}_{n-1}, \mathbf{C}_{n-1})$. For each n the ExKF thus proceeds as:

1. Prediction step. Solve $\mathcal{M}(\hat{\mathbf{m}}_n, \mathbf{m}_{n-1}) = 0$ for $\hat{\mathbf{m}}_n$. Calculate the tangent linear covariance update:

$$\hat{\mathbf{C}}_n = \mathbf{J}_n^{-1} \left(\mathbf{J}_{n-1} \mathbf{C}_{n-1} \mathbf{J}_{n-1}^\top + \mathbf{Q} \right) \mathbf{J}_n^{-\top},$$

where $\mathbf{J}_n = \partial\mathcal{M}/\partial\mathbf{u}_n|_{\hat{\mathbf{m}}_n, \mathbf{m}_{n-1}}$ and $\mathbf{J}_{n-1} = \partial\mathcal{M}/\partial\mathbf{u}_{n-1}|_{\hat{\mathbf{m}}_n, \mathbf{m}_{n-1}}$.
This gives $p(\mathbf{u}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda}) = \mathcal{N}(\hat{\mathbf{m}}_n, \hat{\mathbf{C}}_n)$.

2. Update step. Compute the posterior mean $\mathbf{m}_n$ and covariance $\mathbf{C}_n$:

$$\begin{aligned} \mathbf{m}_n &= \hat{\mathbf{m}}_n + \hat{\mathbf{C}}_n \mathbf{H}^T (\mathbf{H}\hat{\mathbf{C}}_n \mathbf{H}^T + \mathbf{R})^{-1} (\mathbf{y}_n - \mathbf{H}\hat{\mathbf{m}}_n), \\ \mathbf{C}_n &= \hat{\mathbf{C}}_n - \hat{\mathbf{C}}_n \mathbf{H}^T (\mathbf{H}\hat{\mathbf{C}}_n \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H}\hat{\mathbf{C}}_n. \end{aligned}$$

This gives $p(\mathbf{u}_n|\mathbf{x}_{1:n}, \mathbf{\Lambda}) = \mathcal{N}(\mathbf{m}_n, \mathbf{C}_n)$.

The log-marginal likelihood can be calculated recursively, with each term of the log-likelihood computed after each prediction step:

$$\log p(\mathbf{x}_{1:N}|\mathbf{\Lambda}) = \log p(\mathbf{x}_1|\mathbf{\Lambda}) + \sum_{n=2}^{N} \log p(\mathbf{x}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda}),$$

$$p(\mathbf{x}_n|\mathbf{x}_{1:n-1}, \mathbf{\Lambda}) = \mathcal{N}(\mathbf{H}\hat{\mathbf{m}}_n, \mathbf{H}\hat{\mathbf{C}}_n \mathbf{H}^T + \mathbf{R}).$$

Note that although we focus on the ExKF other nonlinear filters could be used; two popular alternatives are the ensemble Kalman filter [88], or, the particle filter [89]. For a linear dynamical model, such as the advection equation considered in Section 5.1, the ExKF reduces to the standard Kalman filter [3].