scientific reports

OPEN

Symplectic encoders for physics-constrained variational dynamics inference

Kiran Bacsa[1,2]✉, Zhilu Lai[1,2,4,5], Wei Liu[1,3], Michael Todd[6] & Eleni Chatzi[2]

We propose a new variational autoencoder (VAE) with physical constraints capable of learning the dynamics of Multiple Degree of Freedom (MDOF) dynamic systems. Standard variational autoencoders place greater emphasis on compression than interpretability regarding the learned latent space. We propose a new type of encoder, based on the recently developed Hamiltonian Neural Networks, to impose symplectic constraints on the inferred a posteriori distribution. In addition to delivering robust trajectory predictions under noisy conditions, our model is capable of learning an energy-preserving latent representation of the system. This offers new perspectives for the application of physics-informed neural networks on engineering problems linked to dynamics.

List of symbols

t:	time index variable associated to a data sample.
T:	time horizon, i.e., maximum value of the time index variable.
τ:	arbitrary time index variable.
h:	discrete integration time step constant.
n:	number of integration steps for a forward integration scheme.
$\mathbf{x}$:	observable random variable.
F:	dimension of observed random variable.
$\mathbf{z}$:	latent random variable.
G:	dimension of latent random variable.
V:	generative model for the mean latent variable.
W:	generative model for the mean observed variable.
S:	generative model for the standard deviation latent variable.
R:	generative model for the standard deviation observed variable.
$\mathbf{h}$:	hidden layer variable of a neural network.
θ:	parameters of VAE decoder.
ϕ:	parameters of VAE encoder.
ψ:	parameters for arbitrary model.
Ψ:	number of parameters for arbitrary model.
β:	parameters for model of the mean latent variable.
δ:	parameters for model of the standard deviation latent variable.
η:	parameters for model of the mean observed variable.
κ:	parameters for model of the standard deviation observed variable.
p:	true probabilistic distribution.
q:	approximate probabilistic distribution.
D_{KL}:	Kullback-Leibner divergence.
$\mathfrak{J}$:	loss function.
Φ:	function approximating the flow of an ODE.
$\mathcal{H}$:	Hamiltonian of a dynamical system.
$\mathcal{L}$:	Lagrangian for Lagrangian multiplier method.

[1]Singapore-ETH Centre, Future Resilient Systems, 138602 Singapore, Singapore. [2]ETH Zurich, Department of Civil, Environmental and Geomatic Engineering, 8093 Zurich, Switzerland. [3]Department of Industrial Systems and Management, NUS, 117576 Singapore, Singapore. [4]HKUST (GZ), Internet of Things Thrust, Guangzhou 511453, People's Republic of China. [5]Department of Civil and Environmental Engineering, HKUST, Hong Kong, People's Republic of China. [6] Department of Structural Engineering, UC San Diego, San Diego 92093, USA. ✉email: kiran.bacsa@sec.ethz.ch

$\mathfrak{L}$:	continuous Lagrangian of a dynamical system.
$\mathfrak{L}^d$:	discrete Lagrangian of a dynamical system.
$\mathbf{q}$:	standard position vector of a dynamical system.
$\mathbf{p}$:	standard momentum vector of a dynamical system.
α:	standard angle vector of a pendulum system.
$\mathbf{l}$:	standard length vector of a pendulum system.
$\mathbf{u}$:	continuous state-space function a dynamical system.
g:	arbitrary quadrature.
λ:	Lagrange multiplier.
$\mathfrak{T}$:	kinetic energy of a dynamic system.
$\mathfrak{U}$:	potential energy of a dynamical system.
$\mathbf{M}$:	mass matrix of a dynamical system.
$\mathbf{C}$:	dissipation matrix of a dynamical system.
K:	potential function of a dynamical system.
$\mathbf{K}_l$:	linear spring potential matrix of a dynamical system.
$\mathbf{K}_d$:	nonlinear spring potential matrix of a dynamical system.
$\mathbf{I}$:	identity matrix.
$\mathbf{0}$:	zero matrix.
D:	forced excitation function of a dynamical system.
$\mathbf{D}$:	forcing matrix of a dynamical system.
$\omega_{\max}$:	function that returns the maximum eigenvalue of the input.
γ:	amplitude of forced excitation of a dynamical system.
f:	frequency of forced excitation of a dynamical system.
$\mathcal{G}$:	gravitational constant.
$\mathcal{N}$:	normal probability distribution.
$\mathcal{U}$:	uniform probability distribution.
μ:	mean of an arbitrary distribution.
σ:	standard deviation of an arbitrary distribution.
L_E:	number of layers for the emission neural network.
N_E:	width of layers for the emission neural network.
N_T:	width of layers for the transmission neural network.
L_{RNN}:	number of layers for the recurrent neural network in the encoder.
L_P:	number of layers for the energy neural network.
N_P:	width of layers for the energy neural network.

A popular class of methods that find application in system identification and control theory are Gaussian State-Space Models (GSSM), which include the broadly established Bayesian Filters[1,2]. This approach consists in modelling the relationship between the inputs, outputs and inner variables of a dynamic system by a set of differential equations acting on a latent state space; which comprises the so called process model. Noise contamination is assumed both on the process equations, which essentially represent the equations of motion, as well as the measurement equation, which pertains to the observed dynamic outputs. The assumed noise processes are typically modelled as zero-mean Gaussian distributions. GSSMs, such as the Kalman filter[3], are particularly effective in decomposing a time series into trends and cycles. However, these models require known structures of the differential (process) and algebraic (observation) equations that govern the dynamical system. Such information is often not available a-priori for most complex problems, particularly for those tied to the domain of structural dynamics, where the precise properties (stiffness, restoring force) of the underlying system are often uncertain, or of unknown form[4]. Applying GSSMs for such problems thus requires approximations at the cost of accuracy, often relaxed by resorting to joint state-parameter identification problems, where the model structure is assumed to be defined a priori[5]. Over the years, various data-driven system-identifications techniques have been implemented to correct this lack of prior knowledge so as to extend GSSMs to more complex systems[6–8].

On a parallel front, new machine learning algorithms, such as the Evidence Lower Bound Optimization (ELBO)[9] through SGVB (Stochastic Gradient Variational Bayes)[10], have allowed generative models, such as Variational Autoencoders (VAE), to combine deep learning with stochastic modelling. In recent years, VAEs have been extended to process sequential data for Gaussian processes by introducing a temporal constraint on the dynamics of the latent space[11]. This approach to sequential modelling offers greater flexibility than more conventional state-space models, primarily since VAEs are more apt to learn non-linear dynamics. Models such as Stochastic Recurrent Networks (STORN)[11] or Deep Markov Models (DMMs)[12], which are further referred to as DVAEs (Dynamic Variational Autoencoders)[13], have achieved promising results in speech analysis, music synthesis and medical diagnosis prediction. We note that DVAEs are an unsupervised learning scheme, since the learning of the latent variable is conditioned without directly being connected to the data, in stark contrast with supervised learning methods for dynamical systems such as Nonlinear Autoregressive with Exogenous input models (NARX).

Similar to other deep learning methods, DVAEs do not learn interpretable latent spaces, since they are biased towards learning compressed representations. This poses a problem for multiple objective tasks, common in Robotics, Control, Structural Health Monitoring (SHM) and Prognostic Health Management (PHM) applications, where in addition to adequate response predictions, we seek to also extract meaningful information on the dynamics of the system[14]. To address this issue, we propose to add extra assumptions on the inference network. Our proposed approach adopts a Neural ODE (NODE)[15] as the a posteriori model for a DVAE; we essentially add the prior information on the distributions being generated via underlying differential equations. Furthermore, we parametrize the integration of our ODE using a symplectic integrator. For a symplectic integrator, the forward integration step splits the phase-space into its displacement and momentum subspaces and updates these subspaces based on Hamiltonian constraints. This imposes an area-preserving property in the phase space on the trajectories of the observed MDOF system. For our systems, we will assume that phase-preservation and energy-conservation properties are equivalent[16]. We thus postulate that our model is able to performed unsupervised statistical learning with energy constraints.

Other recent papers[17,18] have also explored the idea of symplecticity in time-series analysis. We outline our own contribution as follows:

- We propose a novel framework that combines DMMs with physics-informed machine learning in the form of a NODE. This allows us to explore the use of physics-informed neural networks for statistical unsupervised learning.
- We introduce a symplectic constraint on the learned latent space by using Hamiltonian Neural Networks (HNN) for a posteriori distribution learning.
- We show that our model is able to learn latent quantities, such as energy, in addition to being a state-space observer.
- We study the use of a modified symplectic integrator for cases where the dynamical system's energy is not constant, such as dissipative and forced systems. We note that we only deal with systems whose Hamiltonian is separable when expressed in Cartesian coordinates.

We note that our the present method does not aim to replace current SHM system identification methods. Rather we focus on demonstrating that deep learning methods can be biased with apriori physical knowledge for an improved performance. Using small linear systems as a starting point, we incrementally prove the efficacy of our method on more and more complex, i.e. higher nonlinearity and DOF, systems.

Related work

Introduced by Kingma and Welling[9] and Rezende et al.[10], the VAE is one of the most popular approaches in stochastic unsupervised learning. The main function of the VAE is to learn a low-rank latent space that encodes noisy observations from a dataset. During the learning phase, the inference of the latent space from the data is referred to as the encoding, in contrast to decoding, which is the generation of future samples from the latent space. Hence the term autoencoder, since the decoder's distribution should reproduce that of the encoder's. A summary of the inner workings of the VAE is provided in Fig. 1. VAEs have been extended to noisy time-series autoencoding in a novel group of models referred to as Dynamic Variational Autoencoders (DVAE)[13]. This time-dependency can be modelled in different ways. Bayer and Osendorfer[11] introduced STORN, where the decoding takes into account the previous states of the model in addition to the current state. Krishnan et al.[19] included the idea of a Markov property on the latent space in their Deep Kalman filter (DKF). Similarly to a Kalman filter, the latent space, updated during the encoding, is the result of a combination of the VAE inference and its state at the previous step. Chung et al.[20] combined both these ideas (temporal relationship for both encoding and decoding) in their Variational Recurrent Neural Networks (VRNN). In our work, we will mainly refer to the Deep Markov Model[12]. This model is similar to the DKF, with added gating functions on the latent space update.

Recurrent Neural Networks and their variants, such as Long Short-Term Memory (LSTM) Networks[21] form the most widespread deep learning algorithm for time-series analysis. Therefore, many of the previously cited DVAEs rely on RNNs as their basic feature extractors. While efficient at finding time-based correlations in the data, it is not possible to add any prior knowledge of the dynamics of the data to an RNN. Chen et al.[15] introduced the Neural ODE (NODE), a Residual Neural Network (ResNet)[22] repurposed to learn the flow of the dynamics of an observed system. Each layer of the neural network parametrizes the forward step integration of the underlying differential equation.

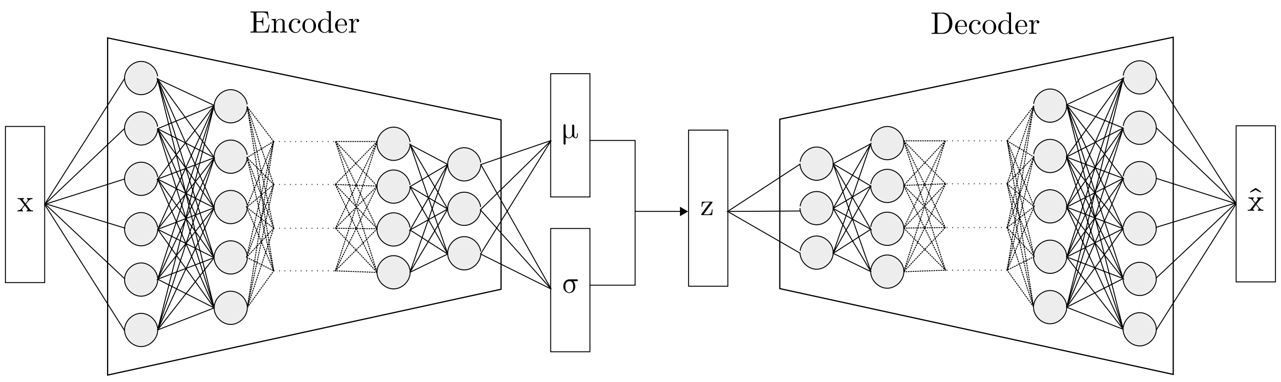

Figure 1. VAE: The data is first passed through an encoder network to output the mean and variance of the latent distribution of the data. Then we sample from this distribution to obtain a candidate latent space. This candidate latent space is then mapped back to an approximation of the original data using a decoder network.

Further works in NODEs by Rubanova et al.[23] have shown that such learning schemes can also be applied to latent variables, i.e., in lifted spaces that have a different dimension from the original observed dataset.

Starting with Raissi et al.[24], deep learning has also found success in physical modelling. Physics-informed Neural Networks (PINN) have achieved state-of-the-art results in fluid mechanics[25] and aerodynamics[26]. In the case of structural dynamics, several classical architectures such as CNNs[27], LSTMs[28,29], as well as Neural ODEs[30], have achieved state-of-the-art performance for structure identification and modelling. Wei et al.[31] have shown that the stochastic nature of VAEs and their subvariants such as the Deep Markov Model[12] can be used to model uncertain dynamics. Further works on generative models have shown their efficacy when applied to fluid dynamic problems. Zabaras and Geneva[32] trained a physics-constrained variational model to predict turbulent flows governed by Navier-Stockes equations. Rasul et al.[33] designed an autoregressive deep learning model, where the data distribution is represented by a conditioned normalizing flow. Their model demonstrate state-of-the-art performance when predicting the dynamics of a flow through a system of pipes.

Hamiltonian Neural Networks (HNN)[34] are a subset of PINNs that parametrize an explicit formulation of the Hamiltonian of the system being observed. This introduces the notion of energy conservation of the latent space. Saemundsson et al.[35] opted instead to parametrize the integration of the dynamic system. Such integrators for Hamiltonian systems are known as symplectic integrators. They allow the update of the latent space to be phase-preserving, a notion close to energy conservation[16].

We note that we are not the first to make the connection between symplectic integration and NODEs. Other works[17,18,36,37] have already explored the idea of learning symplectic flows with modified NODEs. We note in particular the UniCORNN proposed by Rusch and Mishra, which show that the introduction of Hamiltonian constraints in RNNS increases the learning stability by mitigating the exploding/vanishing gradient problem. This is achieved by computing the second order derivative of each hidden layer of the RNN with symplectic constraints. Our main contributions lies in the incorporation of symplectic flows to the encoder of DVAEs, mainly the DMM. Furthermore, we extend our model to learn the dynamics of non-autonomous systems.

Moreover, we remark that we are not the first to introduce the idea of Hamiltonian dynamics for VAEs. Wolf et al.[38] showed that Hamiltonian Markov Chain Monte-Carlo (MCMC) can be introduced for the sampling of the posterior. Hamiltonian MCMC is an alternative sampling method to the Metropolis-Hastings algorithm used in the original work on VAEs. In the case of Metropolis-Hastings, samples are accepted or rejected based on a threshold calculated using the target distributions density function (or a proxy function proportional to said density, as it is the case in Variational Inference). In the case of Hamiltonian MCMC, the target distribution is interpreted as a position vector, and an auxiliary momentum distribution is added. A new sample is generated by integrating the position and momentum distributions forward in time along energy-preserving trajectories with a random momentum value. The new sample is accepted or rejected based on the same thresholding scheme of the Metropolis-Hastings algorithm. The added benefit of the Hamiltonian MCMC is that it enables the sampling of distant states that are still highly probable due to the energy-preserving properties of the Hamiltonian integration scheme. This reduces the correlation between two subsequent samples and allows for a better global convergence of the sampling algorithm. Wolf et al.[38] showed that using Hamiltonian MCMC improved the global convergence for the approximation of the posterior distribution. Caterini et al.[39] mention that Wolf et al.'s estimation method is not amenable to the reparameterization trick, an essential step towards training neural networks for variational inference. Instead, they use Hamiltonian Importance Sampling[40], an annealed version of Hamiltonian MCMC to enforce compatibility with the training of a Hamiltonian VAE. Wang and Delinguette[41] introduced Quasi-symplectic Langevin VAEs, a similar model to the Hamiltonian VAE that replaces the Hamiltonian dynamics with Langevin dynamics according to Langevin's stochastic differential equation which describes the motion of a particle in a fluid.

Methodology

Deep Markov model. The variational autoencoder (VAE). As previously mentioned, the VAE is a stochastic version of an autoencoder. An autoencoder is a DNN trained to match the predicted output to its original input ($\mathbf{x} \approx \hat{\mathbf{x}}$) for $\mathbf{x} \in \mathbb{R}^F$. The autoencoder is parametrized by two neural networks. First, the encoder DNN maps the input vector $\mathbf{x}$ to its latent representation $\mathbf{z}$ with $\mathbf{z} \in \mathbb{R}^G$ and $G \leq F$. The decoder DNN then inverts the previous transformation by mapping the latent variable $\mathbf{z}$ back to the original input $\mathbf{x}$. The VAE[9] extends the concept of an autoencoder to stochastic latent variables. The encoder is used to infer the Probability Density Function (PDF) of the latent variable $\mathbf{z}$ for the data vector $\mathbf{x}$. Within the context of Bayesian inference, the PDF of the decoder for $\mathbf{x}$ is:

$$p(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) \tag{1}$$

with θ the parameters of the generative model, and where $p_\theta(\mathbf{z})$ is the prior distribution of the latent variable. In most cases it is assumed to be a unit Gaussian, i.e. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_L, \mathbf{I}_L)$.

This latent variable is sampled from its inferred PDF and then passed through the decoder to generate a new datapoint $\hat{\mathbf{x}}$. For the VAE, the PDF of $\mathbf{z}$ is assumed to be Gaussian. Th assumption hold for most applications of the VAE. However, for datasets with non-gaussian distributions, the latent space can be deformed a non-gaussian distribution using Neural Autoregressive Flows[42], i.e., a series of invertable, smooth and trainable transformations.

DVAE and ELBO. Our DVAE model reuses the DMM structure from Krishnan et al.[12]. The choice of this model is motivated by the unsupervised training for the discovery of the latent space that encodes the temporal dynamics of the data. In addition, the latent space of the DMM has an enforced Markov property, i.e. the model assumes that the current state of the latent space can be inferred from its previous state; an assumptions that holds true for linear mechanical systems. In the remainder of the paper, we will adhere to the formalism from Girin et al.[13] so that our proposed model can be easily compared to other DVAEs.

We consider a sequence of T observed random vectors of dimension F such as $\mathbf{x}_{1:T} = \{\mathbf{x}_t \in \mathbb{R}^F\}_{t=1}^T$ to which we associate a sequence of latent random vectors of dimension G such that $\mathbf{z}_{1:T} = \{\mathbf{z}_t \in \mathbb{R}^G\}_{t=1}^T$. Both $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ are stochastic variables. The generative model of the DMM models the probability distributions of both these stochastic variables as follows:

$$\mathbf{z}_t \sim \mathcal{N}\big(V_\beta(\mathbf{z}_{t-1}), S_\delta(\mathbf{z}_{t-1})\big) \tag{2}$$

$$\mathbf{x}_t \sim \mathcal{N}\big(W_\eta(\mathbf{z}_t), R_\kappa(\mathbf{z}_t)\big) \tag{3}$$

where V and W are the respective mean models for the latent and observed variables, S and R are the respective variance models, and $\{\beta,\ \delta\}$, $\{\eta,\ \kappa\}$ are the parameter sets for each model, respectively. From now on, we will group $\beta, \delta, \eta, \kappa$ as the parameter set θ, describing the latent distribution model. Unlike conventional VAEs, we introduce a "Markov property", i.e. time dependency on the latent variable $\mathbf{z}_t$. Furthermore, our model differs from that adopted by Krishnan et al.[19] in two ways. First we consider the PDF of the observables to be a Gaussian instead of Bernoulli distribution. We also remove the input dependency and only consider the systems from an output perspective. Our model is mainly applicable where the input of the system is unknown or difficult to measure. The generative model is shown on Fig. 2.

As per Kingma and Welling[9], the true posterior $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})$ is intractable. Therefore, common optimization methods, such as expectation maximization cannot be used. The workaround by Kingma and Welling[9] can be summarized by approximating the posterior distribution with an auxiliary distribution q_ϕ and optimizing the lower bound for the marginal likelihood, such as:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \;||\; p_\theta(\mathbf{z})) \tag{4}$$

Where D_{KL} is the Kullback-Leibner divergence. Both p_θ and q_ϕ can be parametrized by neural networks that can be optimized using Monte-Carlo estimates to compute the unbiased gradients of $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$. This process is known as Evidence Lower Bound Optimization (ELBO).

Training loss. Krishan et al.[12] point out that, in the case of the inference model, the Markov property implies that all past information at time t is contained within $\mathbf{z}_{t-1}$. The posterior can thus be factorized as:

$$p_\theta = p_\theta(\mathbf{z}_1|\mathbf{x}_{1:T}) \prod_{t=2}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T}) \tag{5}$$

This allows us modify the inference model as:

$$q_\phi(\mathbf{z}, \mathbf{x}_{1:T}) = q_\phi(\mathbf{z}_1|\mathbf{x}_{t:T}) \prod_{t=2}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T}) \tag{6}$$

Finally, the loss used to optimize both the generative and inference model during the training phase follows from (4) and is given by:

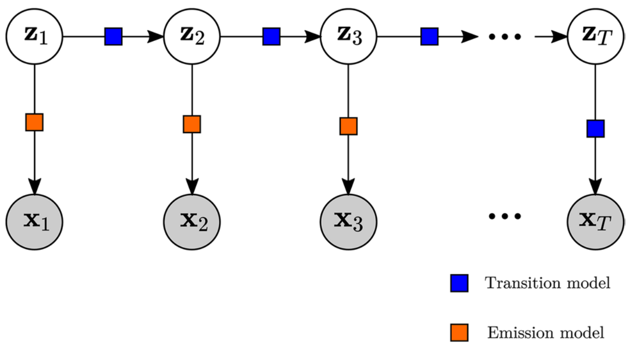

Figure 2. Illustration of the DMM generative process. The latent random variable $\mathbf{z}_t$ a function of its previous instance $\mathbf{z}_{t-1}$ passed through the transmission model, enforcing a Markov property on the latent variable. At each time step, observations $\mathbf{x}_t$ are generated by the emission model.

$$\mathfrak{J}(\theta,\phi,\mathbf{x}_{1:T}) = \sum_{t=1}^{T}\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T})}\left[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t)\right] - \sum_{t=1}^{T}\mathbb{E}_{q_\phi(\mathbf{z}_{t-1}|\mathbf{x}_{1:T})}\left[D_{\mathrm{KL}}(q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T})\;||\;p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}))\right] \tag{7}$$

The first term of the loss maximizes the likelihood $p_\theta(\mathbf{x}_t|\mathbf{z}_t)$, i.e., it enforces reconstruction accuracy by maximizing the likelyhood that the training data can be generated by the latent variable of the model. The second term is referred to as the information gain[43]. This is because the Kullback-Leibner divergence allows us to minimize the difference between the approximate posterior distribution $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T})$ and the Markov transmission prior distribution $p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1})$ and acts as a regularizer[9]. Kingma and Welling[9] have shown that the joint learning of these two terms is equivalent to minizing the Kullback-Leibner divergence between the true and the approximate posterior distributions. The overall computation flow of the DMM is summarized in Fig. 3.

The computation of the ELBO of the DMM requires the direct sampling of $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T})$ to build the expectation. However, the posterior factorization given by Krishnan et al.[12] in (5) and (6) allows us to employ the following "cascade trick" detailed in Girin et al.[13] to approximate the intractable expectations in the loss function. The first expectation in this Variational Lower Bound expression can be developed as follows:

$$\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T})}\left[f(\mathbf{z}_t)\right] = \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})}\left[f(\mathbf{z}_t)\right] = \mathbb{E}_{q_\phi(\mathbf{z}_1|\mathbf{x}_{1:T})}\left[\mathbb{E}_{q_\phi(\mathbf{z}_2|\mathbf{z}_1,\mathbf{x}_{2:T})}[\ldots[\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T})}[f(\mathbf{z}_t)]]]\right] \tag{8}$$

where $f(\mathbf{z}_t)$ denotes an arbitrary function of $\mathbf{z}_t$. The second expectation of the Variational Lower bound can be developed by a similar procedure. Then each intractable expectation can be approximated using Monte Carlo estimates. This requires the sampling of $q_\phi(\mathbf{z}_\tau|\mathbf{z}_{\tau-1},\mathbf{x}_{\tau:T})$, iteratively for $\tau = 1$ to t using the same reparametrization trick as in standard VAEs. Thus, the Variational Lower Bound becomes differentiable and can be optimized with gradient-descent-based techniques.

Neural ODEs.

Neural ODEs are a new type of DNN architecture designed for time-series analysis. Under the assumption that the dynamics of a system follows a set of differential ordinary equations, the model attempts to approximate the flow of the system.[15] reuse the existing ResNet[22] architecture, where each neural network layer parametrizes a forward flow step.

This is equivalent to approximating a forward integration of the governing ODE using a single layer perceptron. For an arbitrary hidden step $\mathbf{h}_t$ at time t of the encoding process, for a nondescript forward time step discretization τ, the forward time step is updated with:

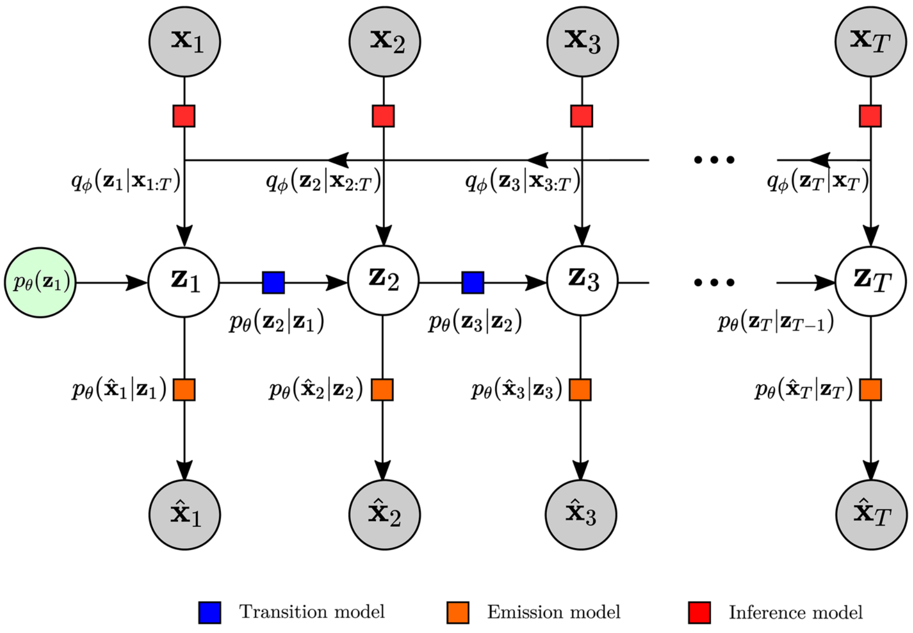
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_T$
$q_\phi(\mathbf{z}_1|\mathbf{x}_{1:T})$
$q_\phi(\mathbf{z}_2|\mathbf{x}_{2:T})$
$q_\phi(\mathbf{z}_3|\mathbf{x}_{3:T})$
$q_\phi(\mathbf{z}_T|\mathbf{x}_T)$
$p_\theta(\mathbf{z}_1)$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_T$
$p_\theta(\mathbf{z}_2|\mathbf{z}_1)$
$p_\theta(\mathbf{z}_3|\mathbf{z}_2)$
$p_\theta(\mathbf{z}_T|\mathbf{z}_{T-1})$
$p_\theta(\hat{\mathbf{x}}_1|\mathbf{z}_1)$
$p_\theta(\hat{\mathbf{x}}_2|\mathbf{z}_2)$
$p_\theta(\hat{\mathbf{x}}_3|\mathbf{z}_3)$
$p_\theta(\hat{\mathbf{x}}_T|\mathbf{z}_T)$
$\hat{\mathbf{x}}_1$
$\hat{\mathbf{x}}_2$
$\hat{\mathbf{x}}_3$
$\hat{\mathbf{x}}_T$
🟦 Transition model
🟧 Emission model
🟥 Inference model

Figure 3. Deep Markov Model: Similarly to the VAE, the data is mapped to a latent representation of the data via the encoder. However, the final latent state is estimated as a weighted average between it's previous and inferred state. For the first estimated state, the weighted average is computed with a learned prior distribution. This resulting latent state $\mathbf{z}_t$ is the one that is passed on to the decoder transition and emission models for an estimation of the data and the subsequent latent state.

$$\mathbf{h}_{t+\tau} = \mathbf{h}_t + \frac{\mathrm{d}\mathbf{h}_t}{\mathrm{d}\tau} \tag{9}$$

With the approximation using the parameter set ψ as the weights of our neural network :

$$\frac{\mathrm{d}\mathbf{h}_t}{\mathrm{d}\tau} = \Phi(\mathbf{h}_t, \tau, \phi) \tag{10}$$

The optimization for NODEs relies on the *adjoint sensitivity method* by Pontryagin et al.[44]. For a dataset generated by the underlying state-space function $u(t)$ and a model Φ of parameters ψ of size Ψ replicating the flow of $u(t)$, the initial value problem (IVP) can be formulated as:

$$\text{IVP:}\begin{cases} \frac{du(t)}{dt} = \Phi(u, t, \psi) \\ u(t=0) = u_0 \end{cases} \tag{11}$$

The model Φ is optimized by minimizing the loss $\mathfrak{J}$ over the time horizon T. Here $\mathfrak{J}$ is defined as:

$$\mathfrak{J}(u, \psi) = \int_0^T g(u, \psi)\mathrm{d}t \tag{12}$$

Where g is an arbitrary quadrature computing the fitness between $\frac{du(t)}{dt}$ and $\Phi(u, t, \psi)$. In the case of NODEs, the model Φ is a neural network optimized through backpropagation. Backpropagation requires the computation of the gradients of the loss with respect to the parameters, i.e. :

$$\frac{\mathrm{d}\mathfrak{J}}{\mathrm{d}\psi}(u, \psi) = \int_0^T \frac{\mathrm{d}}{\mathrm{d}\psi} g(u, \psi)\mathrm{d}t = \int_0^T \frac{\partial g}{\partial \psi}(u, \psi) + \frac{\partial g}{\partial u}(u, \psi)\frac{\mathrm{d}u}{\mathrm{d}\psi}(t)\mathrm{d}t \tag{13}$$

The $\frac{\mathrm{d}u}{\mathrm{d}\psi}(t)$ term scales linearly with the number of parameters Ψ. For a single parameter $\psi_i, i \leq \Psi$, the following IVP must be satisfied:

$$\text{IVP:}\begin{cases} \frac{\mathrm{d}}{\mathrm{d}\psi_i}\frac{\mathrm{d}u(t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}\psi_i}\Phi(u, t, \psi) \\ \frac{\mathrm{d}}{\mathrm{d}\psi_i}u(t=0) = \frac{\mathrm{d}}{\mathrm{d}\psi_i}u_0 \end{cases} \tag{14}$$

Thus every single additional parameter will result in an additional IVP that will need to be solved jointly with the initial IVP. This makes the implicit solving (Euler, Runge-Kutta, etc...) of the IVP unfeasible with models such as neural networks where $\Psi \gg 1$.

Pontryagin et al.'s[44] *adjoint sensitivity method* proposes to solve the IVP using the Lagrange multiplier method over time with the model fitness as a constraint. The optimization Lagrangian $\mathcal{L}$ with multipliers λ is thus given as:

$$\mathcal{L}(u, \lambda, \psi) = \mathfrak{J}(u, \psi) + \int_0^T \lambda^T(t)(\Phi(u, t, \psi) - \frac{\mathrm{d}u(t)}{\mathrm{d}t})\mathrm{d}t \tag{15}$$

The multipliers λ are constrained such that the $\frac{\mathrm{d}u}{\mathrm{d}\psi}(t)$ are negated from the gradient of the Lagrangian. The computation of λ for all t is itself the result of a terminal value problem (TVP) given as:

$$\text{TVP:}\begin{cases} \frac{\mathrm{d}\lambda^T(t)}{\mathrm{d}t} = -\frac{\partial\Phi}{\partial u}(u, t, \psi)\lambda^T(t) \\ \lambda^T(t=T) = -\frac{\partial\mathfrak{J}}{\partial u}(u, \psi) \end{cases} \tag{16}$$

Therefore during the computation of the Lagrangian, we only need to call our implicit solvers twice: once for the original IVP, and once for the multipliers' TVP solving backwards in time. The remaining gradient terms $\frac{\partial\Phi}{\partial u}$, $\frac{\partial\Phi}{\partial\psi}$, $\frac{\partial g}{\partial u}$, $\frac{\partial g}{\partial\psi}$ and $\frac{\partial u_0}{\partial\psi}$ can be computed numerically using reverse automatic differentiation. The NODE's optimization can be summarized by the following steps:

- Solve the IVP for the initial $u(t)$.
- Compute $\lambda^T(t=T)$.
- Solve the TVP for $\lambda(t)$.
- Compute gradients and perform backpropagration.

The overall computation flow of the NODE is summarized in Fig. 4. Chen et al.[15] and Rubanova et al.[23] have shown that such a paradigm can further be applied with a generative approach; from a set of samples $\mathbf{x}$, a latent variable space $\mathbf{z}$ can be learned. This variable can then be evolved into future time steps to generate future predictions. This is equivalent to the standard VAE setting where the latent space can be evolved forward in time to generate future data samples. Using a decoder with parameters θ and using the encoder parameters ϕ to replace ψ, the NODE in the VAE context is given by:

$$\mathbf{z}_{\tau_0} \sim p(\mathbf{z}_{\tau_0}) \tag{17}$$

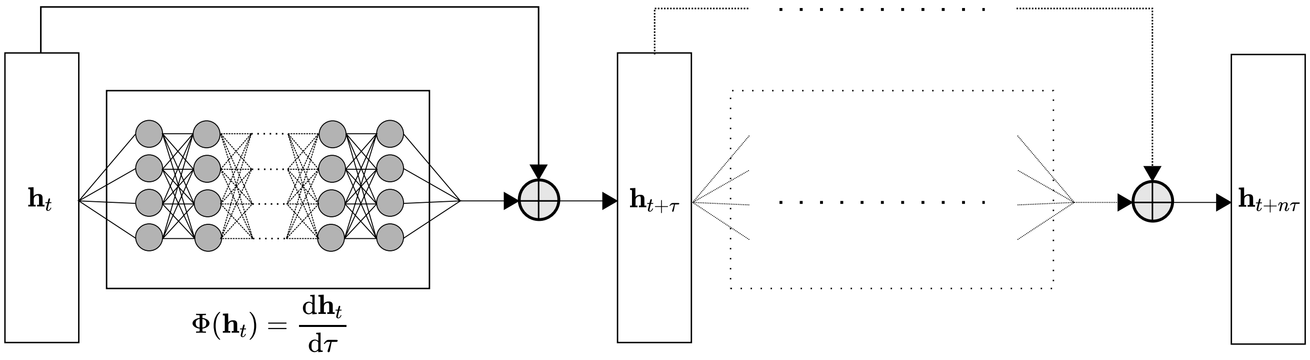
$\mathbf{h}_t$
$\Phi(\mathbf{h}_t) = \frac{\mathrm{d}\mathbf{h}_t}{\mathrm{d}\tau}$
$\mathbf{h}_{t+\tau}$
$\mathbf{h}_{t+n\tau}$

Figure 4. Neural ODE: For a dynamical system, we discretize a single step into n intervals of time τ. For each interval, we use a Multi Layer Perceptron (MLP) to approximate the forward derivative. Using a residual connection, we can approximate a forward integration for a time interval τ. We repeat this process n times for the full NODE.

$$\mathbf{z}_{\tau_1}, \mathbf{z}_{\tau_2}, ..., \mathbf{z}_{\tau_N} = \text{ODESolve}(\mathbf{z}_{\tau_0}, \Phi, \phi, \tau_0, ..., \tau_n) \tag{18}$$

$$\text{each } \mathbf{x}_{\tau_i} \sim p(\mathbf{x}|\mathbf{z}_{\tau_i}, \theta) \tag{19}$$

In their approach, the NODE was used to obtain a decoder capable of generating observations for an irregularly sampled time-series. In contrast we make use of NODEs as en encoder for a better approximation of the a posteriori distribution.

Hamiltonian neural networks. An emerging field in Machine Learning is that of Physics-Informed Neural Networks[24]. The intuition behind such networks is to add additional biases to Machine Learning models, such as conservation and invariance, to incorporate the physical laws that govern the observed system in the first place. Greynanus et al.[34] made the observation that such laws can be described within the framework of Hamiltonian mechanics. At any arbitrary time step, the update of the position $\mathbf{q}$ and the momentum $\mathbf{p}$ for a Hamiltonian system described by the Hamiltonian function $\mathcal{H}$ can be given as:

$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \tag{20}$$

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{21}$$

Based on the Hamiltonian, we can derive a symplectic forward map for the position and momentum. A mapping in a 3D space is symplectic[45] if it induces volume preservation. The idea behind the method by Greydanus et al.[34] is to parametrize the Hamiltonian by a neural network so as to embed the properties of symplecticity on the updates of the system. A similar method is that of Saemundsson et al.[35]. In this approach, the idea of symplecticity is introduced through the Lagrangian perspective. Assuming the mass to be retrievable, we only need to parametrize the potential energy for the forward step, known as the velocity-verlet integrator, as explained later. We justify our assumptions from the fact that in practical applications, one can usually estimate the mass with much more accuracy than the spring constants of the dynamical system being studied. For an autonomous non-dissipative system, the Euler-Lagrange equations are given as follows:

$$\mathfrak{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathfrak{T}(\dot{\mathbf{q}}) - \mathfrak{U}(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{M}\dot{\mathbf{q}} - \mathfrak{U}(\mathbf{q}) \tag{22}$$

with $\mathbf{q}$ denoting the vector of generalized coordinates, $\mathbf{M}$ the diagonal mass matrix and $\mathfrak{T}$, $\mathfrak{U}$ denoting the kinetic and potential energy, respectively. The discrete time Lagrangian can be approximated, similarly to the Euler method, by integrating the equations of motion over an arbitrarily small time step h, i.e.:

$$\mathfrak{L}^d(\mathbf{q}_t, \mathbf{q}_{t+1}, h) \approx \int_t^{t+h} \mathfrak{L}\big(\mathbf{q}(\tau), \dot{\mathbf{q}}(\tau)\big)\, d\tau \tag{23}$$

Note that the above equation can be derived on the basis of the principle of least action (any real path is a sum of infinitesimal Lagrangian steps).

From this quadrature, we derive what are known as Variational Integrators (VI). VIs are an alternative to Euler-based methods for the numerical integration of Hamiltonian systems. VIs are known as symplectic integrators, since they conserve the continuous time energy of the system in the discrete domain with a third order error. VIs have been shown to be more stable, even for larger time steps which are not feasible for implicit integration methods[45].

We derive our integrator as follows. We start by parametrizing the continuous Lagrangian with the parameter vector ϕ.

$$\mathfrak{L}_\phi(\mathbf{q},\dot{\mathbf{q}}) = \mathfrak{T}_\phi(\dot{\mathbf{q}}) - \mathfrak{U}_\phi(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{M}_\phi\dot{\mathbf{q}} - \mathfrak{U}_\phi(\mathbf{q}) \tag{24}$$

A quadrature rule is applied to derive the discretized equivalent; here the trapezoidal rule is adopted:

$$\mathfrak{L}^d_\phi(\mathbf{q}_t,\mathbf{q}_{t+1},h) = \frac{h}{2}\left(\mathfrak{L}_\phi\left(\mathbf{q}_t,\frac{(\mathbf{q}_{t+1}-\mathbf{q}_t)}{h}\right) + \mathfrak{L}_\phi\left(\mathbf{q}_{t+1},\frac{(\mathbf{q}_{t+1}-\mathbf{q}_t)}{h}\right)\right) \tag{25}$$

We can then derive the velocity-Verlet integrator, as per Saemundsson et al.[35] :

$$\mathbf{q}_{t+1} = \mathbf{q}_t + h\mathbf{M}_\phi^{-1}\dot{\mathbf{q}}_t - \frac{h^2}{2}\mathbf{M}_\phi^{-1}\frac{\partial\mathfrak{U}_\phi(\mathbf{q}_t)}{\partial\mathbf{q}_t} \tag{26}$$

$$\mathbf{p}_{t+1} = \mathbf{p}_t - \frac{h}{2}\left(\frac{\partial\mathfrak{U}_\phi(\mathbf{q}_t)}{\partial\mathbf{q}_t} + \frac{\partial\mathfrak{U}_\phi(\mathbf{q}_{t+1})}{\partial\mathbf{q}_{t+1}}\right) \tag{27}$$

with $\mathbf{p}_t = \mathbf{M}_\phi^{-1}\dot{\mathbf{q}}_t$.

Saemundsson et al.[35] then combine these symplectic steps into a single RNN to create a symplectic version of the NODE described above. This method was shown capable of retrieving the latent space for MDOF dynamical systems.

Symplectic DVAE. We combine all the models mentioned above into a symplectic DVAE. We do this by replacing the DMM's RNN encoder with a NODE as per Fig. 5. It is noted here that we still use an RNN to augment the observations to the latent space's dimension. Then, we pass these samples through the Neural ODE to predict the previous latent distribution; we pass the samples in reverse, since we wish to infer the posterior distribution. This process is summarized in Fig. 5. Furthermore, each layer of the NODE follows the Velocity-Verlet integration step. The full model architecture comprises the following neural networks :

- The emission neural network, an MLP with L_E layers of N_E width estimating the emission PDF $p(\mathbf{x}_t|\mathbf{z}_t)$
- The transmission neural network, a two layer MLP with a width of N_T estimating the transmission PDF $p(\mathbf{z}_t|\mathbf{z}_{t-1})$
- The RNN with L_{RNN} layers for the computation of the approximate posterior $q(\mathbf{z}_t|\mathbf{x}_{t:T})$
- The energy neural network; an MLP with L_P layers of N_P width estimating $\frac{\partial\mathfrak{U}_\phi(\mathbf{q}_t)}{\partial\mathbf{q}_t}$

Our methodology can be summarized as follows: we postulate that our data is a random variable generated by a latent physical process which motivates our use of Variational Inference to learn a generative model. We choose the DMM as our generative model to enforce a Markov property which holds true globally for linear systems and locally for nonlinear systems. We use a NODE as our inference network to which biases the model to learn the flow of a dynamical system governed by an ODE. We can enforce additional conditions on the type of ODE that is learned. In our work, we choose to enforce symplectic constraints on the forward integration step to imbue our model with energy-preserving properties.

Dissipative systems. The use of symplectic integration restricts us to the study of systems with constant total energy. However, the vast majority of systems with applications in industry are either non-autonomous, or dissipative, or both. In the context of this paper, we will focus on the latter. To incorporate the idea of dissipation

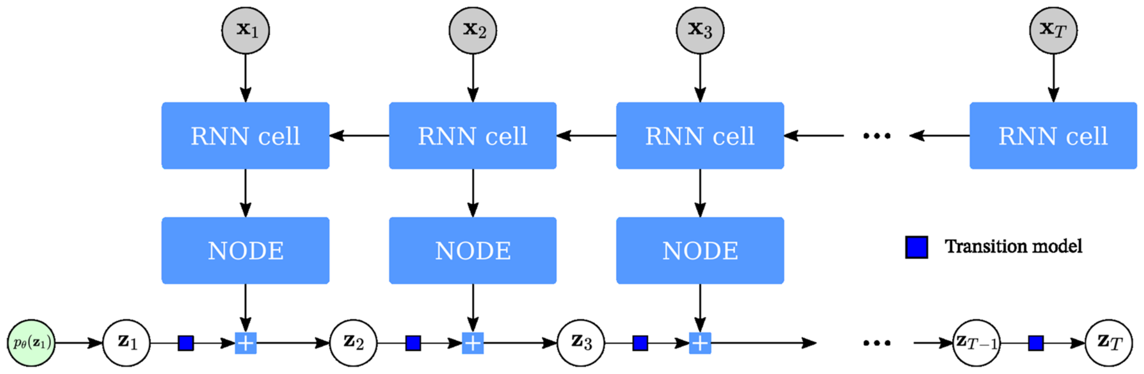

Figure 5. RNN-NODE encoder: Observations at time t are augmented to the latent space dimension using a RNN network. We use an RNN instead of a regular neural network so as to pass long-term dependencies to the next time step. The lifted data is then passed through a NODE to simulate a forward integration to estimate the next latent step. The NODE is modified such that its estimated flow is symplectic.

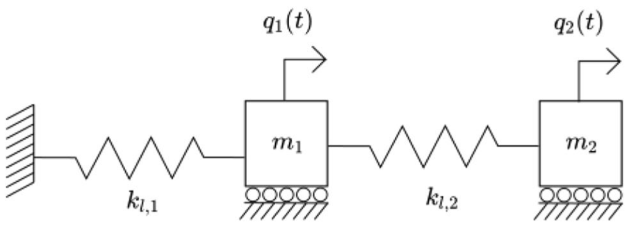

Figure 6. Autonomous 2DOF system.

into our system, we implemented a presymplectic Verlet integrator following the recommendations of Franca et al.[46]. By augmenting the integrator state with an auxiliary time variable τ_t, we obtain the following updates for a single step:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + h\mathbf{M}_\phi^{-1}\dot{\mathbf{q}}_t - \frac{h^2}{2}\mathbf{M}_\phi^{-1}\frac{\partial\mathfrak{U}_\phi(\tau_t,\mathbf{q}_t)}{\partial\mathbf{q}_t} \tag{28}$$

$$\tau_{t+1} = \tau_t + h \tag{29}$$

$$\mathbf{p}_{t+1} = \mathbf{p}_t - \frac{h}{2}\left(\frac{\partial\mathfrak{U}_\phi(\tau_t,\mathbf{q}_t)}{\partial\mathbf{q}_t} + \frac{\partial\mathfrak{U}_\phi(\tau_{t+1},\mathbf{q}_{t+1})}{\partial\mathbf{q}_{t+1}}\right) \tag{30}$$

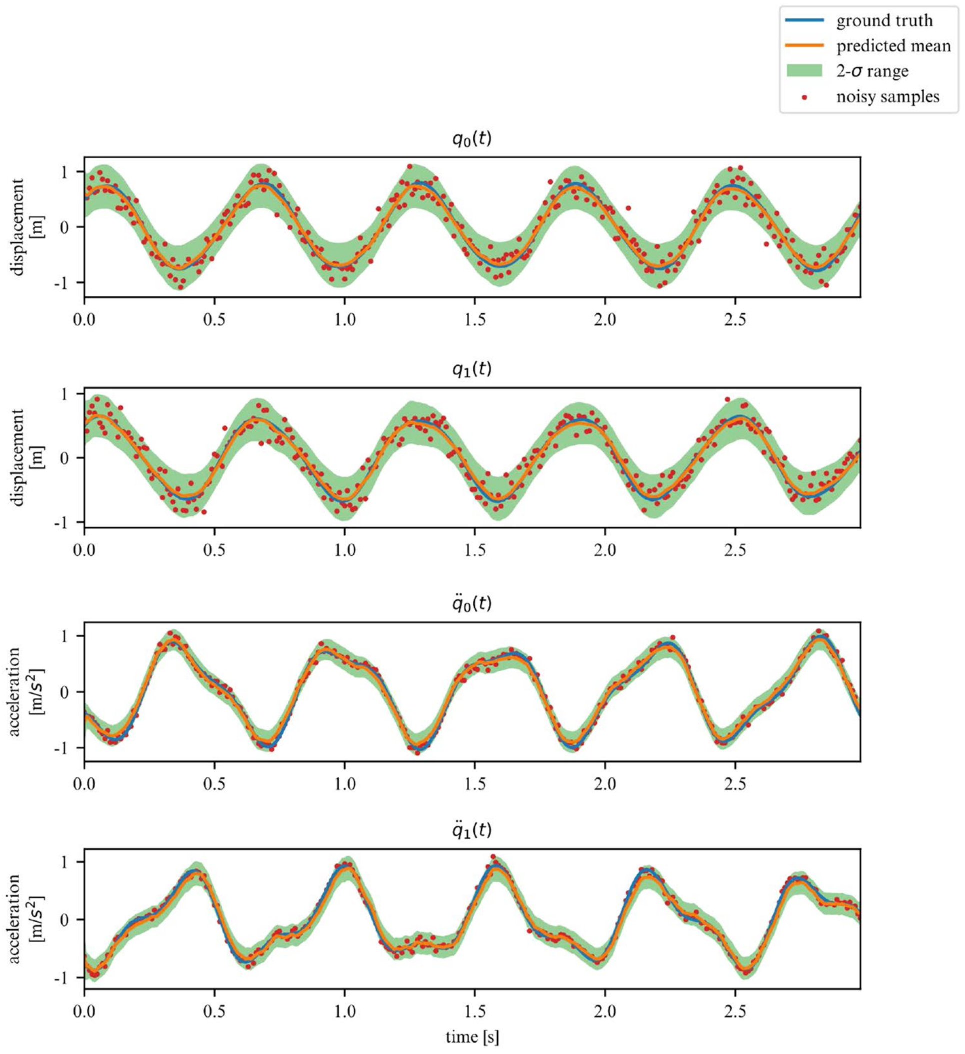

Figure 7. Observations for 2DOF system.

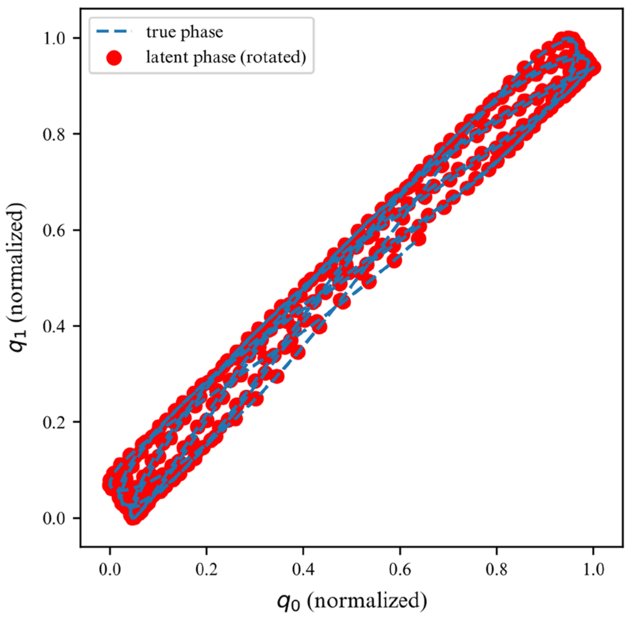

Figure 8. Phase space for first DOF of a linear 2DOF system.

where, as previously, with $\mathbf{p}_t = \mathbf{M}_\phi^{-1}\dot{\mathbf{q}}_t$.

Experiments

We implement both the emitter and transmission MLP using the popular PyTorch library. The encoder's RNN is also implemented using PyTorch[47]. For the neural ODE, we use a modified version of the torchdiffeq library[15] provided by Ishikawa[48]. For the variational inference, we rely heavily on the automated processes provided by the Pyro library[49]. Notably, we let the library handle the calculation of the ELBO as well as all Monte Carlo simulations necessary for stochastic backpropagation.

Dynamical system.

The focus on our experiments will be the study of linear and nonlinear ODE systems subject to noise. The ODE governing the dynamics of each system is defined as:

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{C}\dot{\mathbf{q}}(t) + K(\mathbf{q}(t)) = D(t) \tag{31}$$

Where **M** is the diagonal mass matrix, **C** is the dissipation matrix, K is the potential function, D is the external excitation force function and $\mathbf{q}(t)$ is the position vector. In our default experiments, we assume that displacements and accelerations are measurable. For all systems, we assume that these measurements are available for all DOFs (in a separate section, we explore the case where part of the DOFs are not accessible.). We will also assume that the mass matrix is known ($\mathbf{M} = \mathbf{I}$ the identity matrix) and that the rest of the parameters for **C**, K and **D** are randomly sampled from the uniform distribution $\mathcal{U}[0, 2]$. Furthermore, we make the assumption that displacement measurements are subject to higher uncertainty than that of accelerations, and therefore corrupt the former with 10dB of Gaussian noise and the latter with 20dB of Gaussian noise.

In each experiment, the training dataset is generated as follows. Once we have defined the parameters of the dynamical system, we solve the ODE using RK45[50] for 500 iterations with different initial values. We then select our observed states (positions and accelerations) and corrupt these with additive Gaussian noise. We would like to note that we are aware that this assumption already poses some form of bias on the assumed noise distribution. However, non-Gaussian noise could be easily tackled via use of normalizing flows[51]. Inspired by Saedmunsson et al.[35], we augment our dataset by subdividing each simulation into windows of 50 with a shift of 1.

Coupled spring-mass system.

As a proof of concept, we start off by studying the dynamics two masses coupled with two springs, as shown on Fig. 6. For now, we only consider the free vibration non-dissipative responses of the system. As observables, we choose the position and the acceleration of both masses, as these are the most common quantities measured my sensors in monitoring systems. We define the equations of motion for each degree of freedom and put these in matrix form, i.e. :

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{K}_l\mathbf{q}(t) = \mathbf{0} \tag{32}$$

Here, the potential function $K(\mathbf{q}(t)) = \mathbf{K}_l\mathbf{q}(t)$, i.e. is a linear transformation. The excitation function $\mathbf{D}(t) = \mathbf{0}$ the zero matrix.

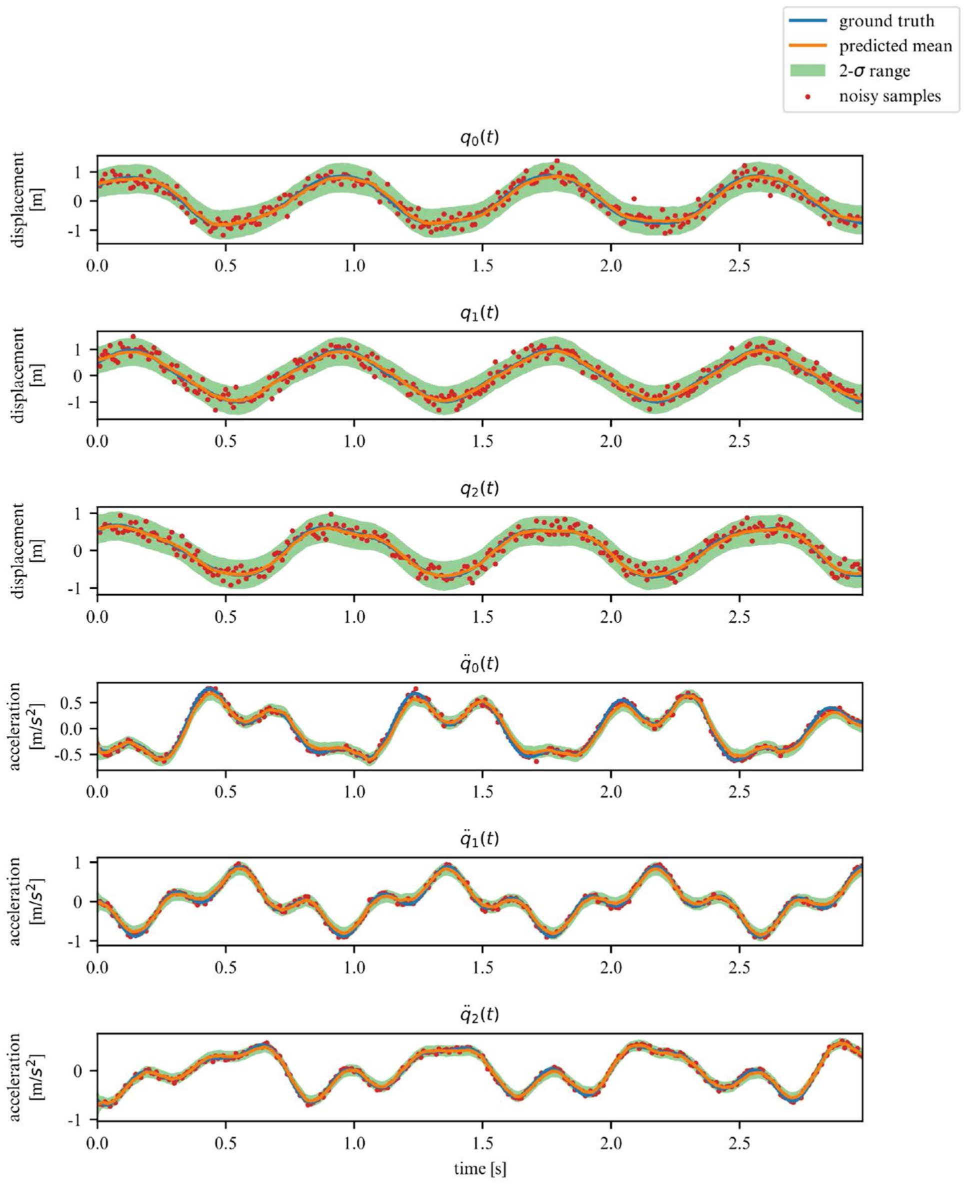

Figure 9. Observations for a linear 3DOF system.

Optimal parameter search. Our model's ability to fit the training data is highly dependant on the hyperparameters defining the dimensions of the different neural networks. However, these networks are relatively small ($N_{\text{parameters}} \sim 10^3$), which renders the involved training costs computationally cheap. Motivated by the low dimensionality of the problem, we perform a hyperparameter search using Bayesian Parameter Search[52] to determine the optimal model configuration, which is found to be described by the following parameters:

Parameter	N_E	L_E	N_T	N_P	L_P	L_{RNN}
Value	18	2	15	85	3	15

While the ELBO allows our DMM to infer latent variables from noisy data, is does not offer an interpretable metric on how well the model fits the data. Once trained, we can use the derived DMM as a one-step ahead predictor to verify how well it has learned the dynamics of the system (Fig. 7). To further verify that the model has learned the underlying dynamics, we can plot the phase-space of the latent representation. In our case, the model learns a rotated version of the phase-space; symplecticity only constrains the area, not the orientation. Therefore, we can correct for the angle by applying a rotation transformation calculated with a least-square estimator. On Fig. 8 we can see that both phase-spaces line up, indicating that our model was able to learn a latent representation corresponding to the phase-space of the linear system.

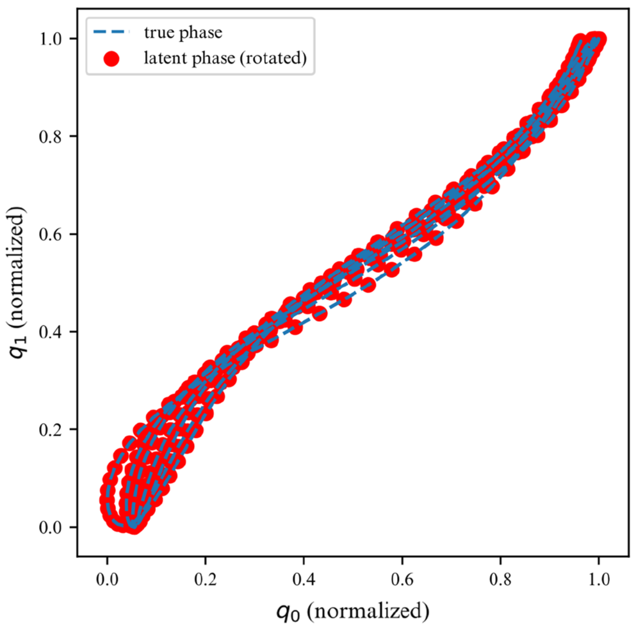

Figure 10. Phase space for the first DOF of a 3DOF system.

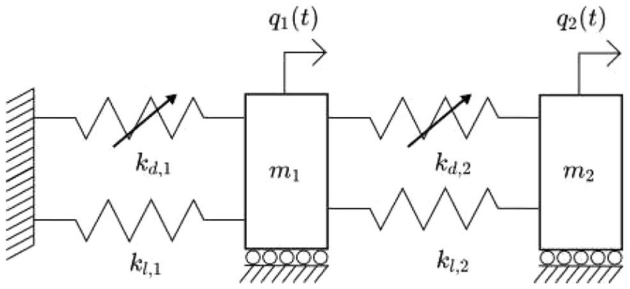

Figure 11. 2DOF duffing oscillator system.

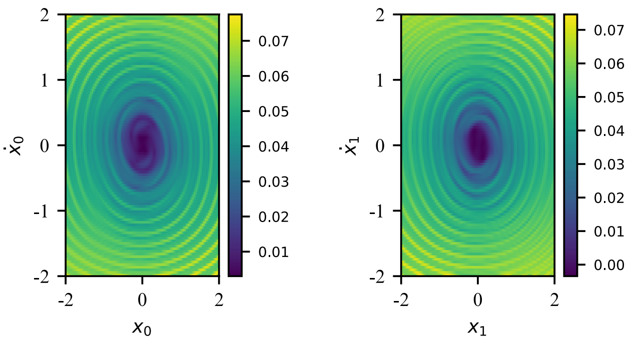

Figure 12. FTLE for the duffing oscillator.

Linear system of higher dimensionality.

Moving on, we investigate the ability of the proposed model to adapt to a higher number of degrees of freedom. We verify that our model can scale up by learning the dynamics of a 3DOF system. The optimal parameter search now returns the following hyperparameters:

Parameter	N_E	L_E	N_T	N_P	L_P	L_{RNN}
Value	29	1	24	11	5	1

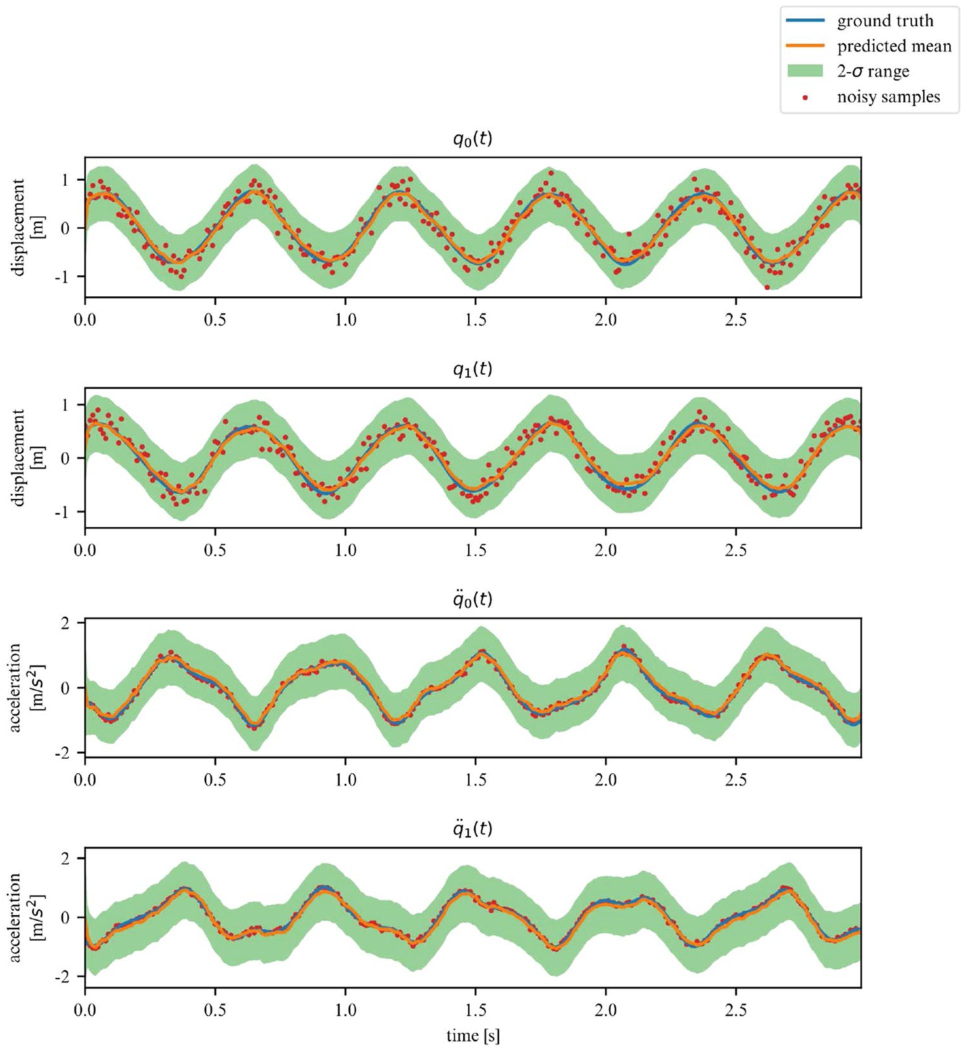

Figure 13. Duffing oscillator observations.

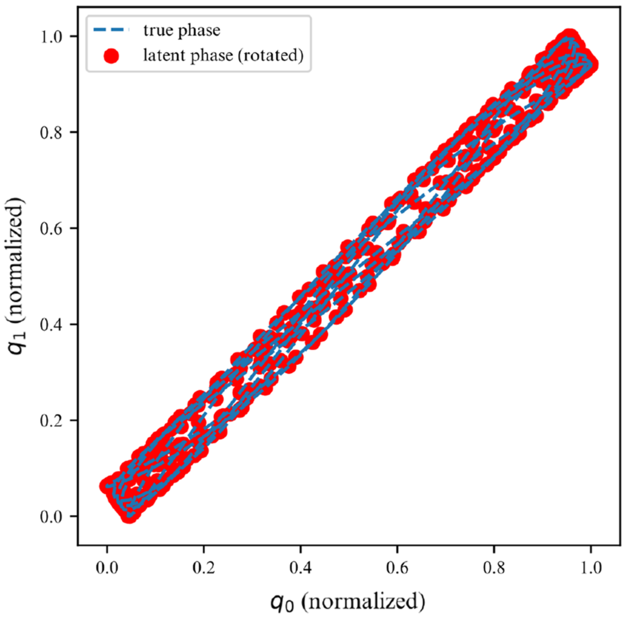

Figure 14. Duffing oscillator phase space.

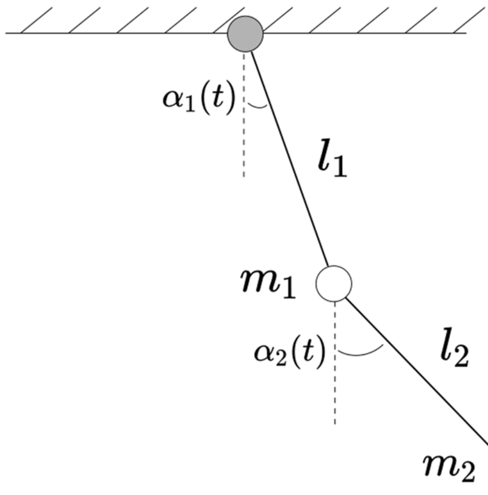

Figure 15. Double pendulum system.

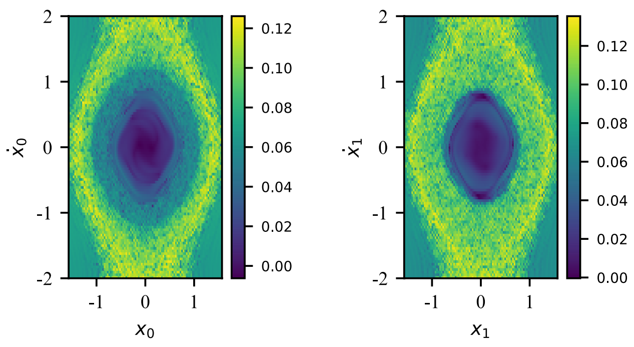

Figure 16. FTLE for the double pendulum.

We corroborate our previous conclusions for the 3DOF system on Figs. 9 and 10.

While we have just demonstrated that the number of DOFs can be increased, our model struggles to scale for much larger numbers of DOFs. This is due to the heavy influence of our hyperparameters on the performance of the model. This makes the Bayesian Parameter Search more and more costly as the number of DOFs increases. We leave this weakness of our model as a topic for further research.

Nonlinear dynamics. *Two degree of freedom duffing oscillator.* We modify the previous 2DOF example via addition of cubic springs, resulting in a two degree of freedom Duffing oscillator, as shown on Fig. 11.

This adds a cubic nonlinear term to the elastic potential. The updated equations of motion are now given by :

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{K}_l\mathbf{q}(t) + \mathbf{K}_d\mathbf{q}^3(t) = \mathbf{0} \tag{33}$$

Here, the potential function $K\left(\mathbf{q}(t)\right) = \mathbf{K}_l\mathbf{q}(t) + \mathbf{K}_d\mathbf{q}^3(t)$, i.e. is a nonlinear transformation. Nonlinear dynamical systems are notorious for exhibiting chaotic dynamics for certain regions of the initial phase-space. To explore the impact of the initial conditions on the dynamics of this system, we calculate the Finite-Time Lyapunov Exponent (FTLE) along the phase-plane. The FTLE is a useful metric to quantify the deviation of the trajectory of the system for an infinitesimal perturbation around a given initial condition. A high FTLE would indicate a high deviation for a small perturbation, which is a characteristic of a chaotic trajectory. The FTLE is given by:

$$\mathrm{FTLE}_{t_0}^{T}(\mathbf{q}_0) = \frac{1}{2|T - t_0|}\log\left(\omega_{\max}(\nabla_{\mathbf{q}}\Phi^{\mathrm{T}}(\mathbf{q}_0)\nabla_{\mathbf{q}}\Phi(\mathbf{q}_0))\right) \tag{34}$$

with $\mathbf{q}(T) = \Phi(\mathbf{q}_0)$ and $\omega_{\max}$ a function that returns the maximum eigenvalue of the input. We can now calculate the FTLE for both DOFs along the [-2, 2] square phase-space with an initial $\Delta_{q_0} = \Delta_{\dot{q}_0} = 0.01$:

An observation of Fig. 12 indicates that as we move away from the origin along the phase-space, the FTLE tends to increase, particularly around "chaotic rings". Running a hyperparameter sweep, we are able to design a model capable of adapting to this more diverse dataset:

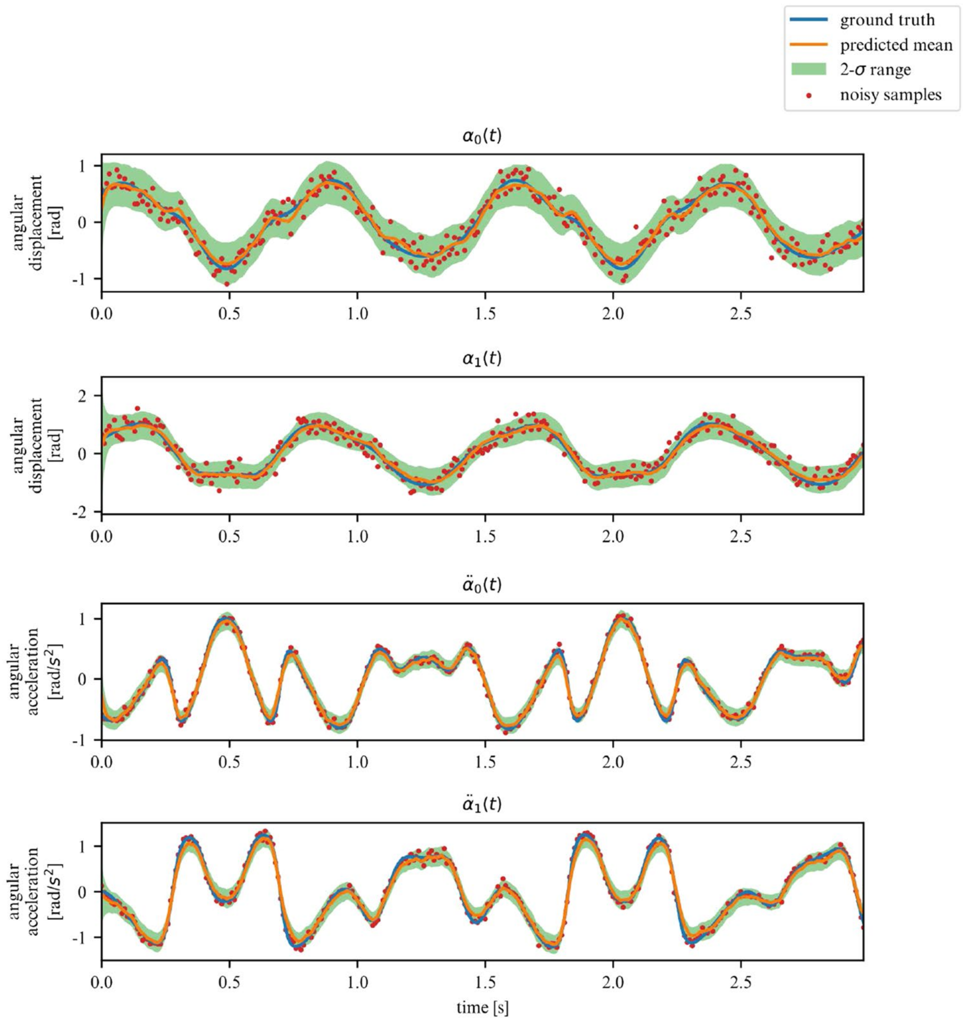

Figure 17. Double pendulum observations.

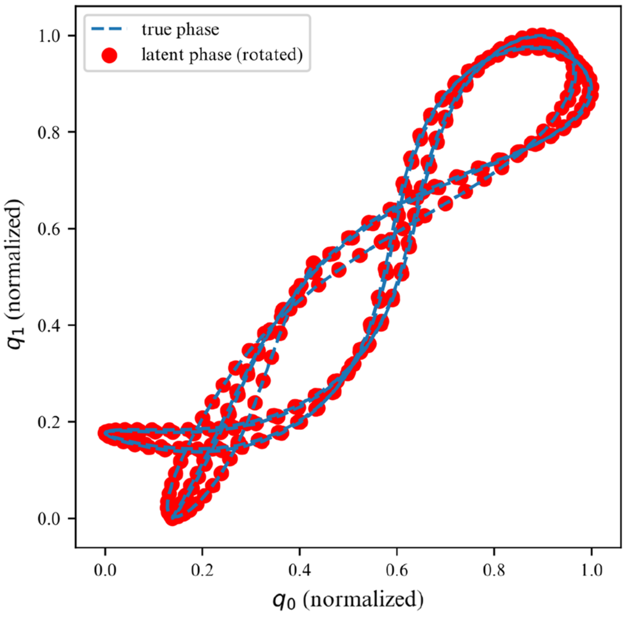

Figure 18. Double pendulum phase space.

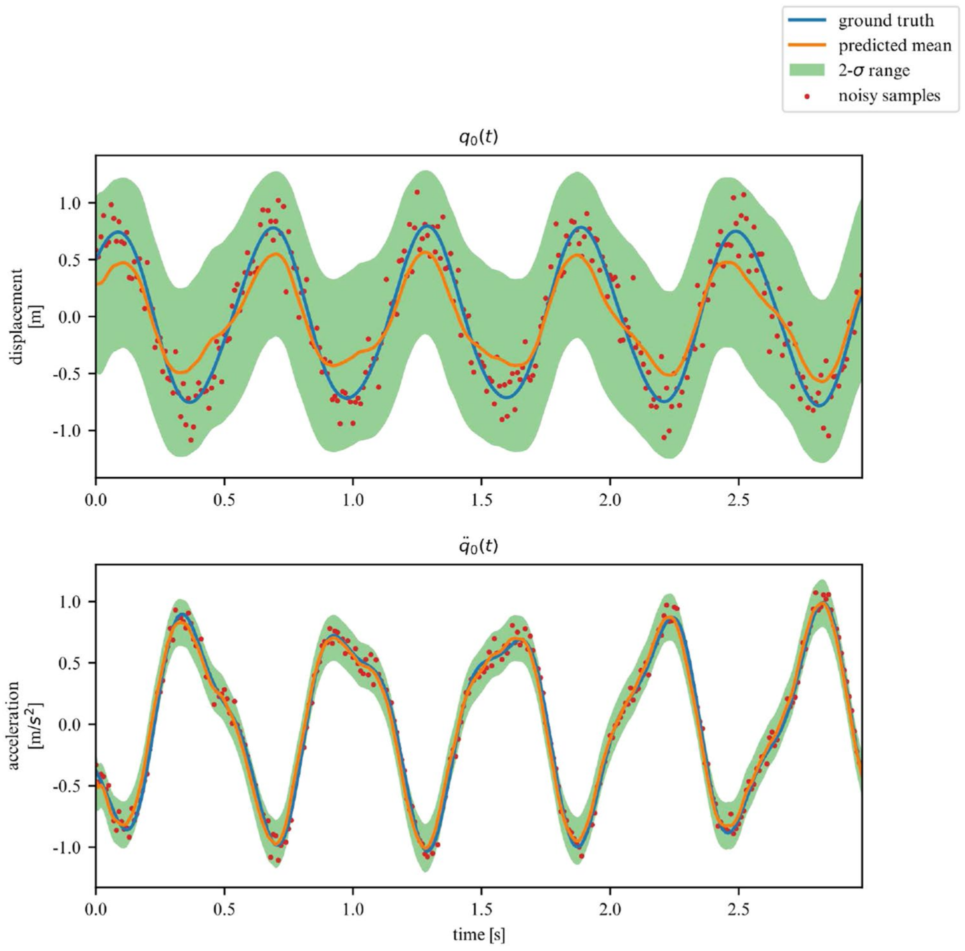

Figure 19. Partially observed linear 2DOF system.

Parameter	N_E	L_E	N_T	N_P	L_P	L_{RNN}
Value	14	1	36	54	1	2

Figures 13 and 14 verify the ability of the proposed model to learn nonlinear dynamics just like in the linear dynamics. In the case of the Duffing oscillator, the chaoticness is low enough for our model to generalize for the entire phase-space.

Double pendulum. The NDOF pendulum is another example of a nonlinear system that exhibits chaotic dynamics. Already for a 2DOF pendulum (Fig. 15), the equations of motion are quite complicated. Using angular coordinates α for the angle and **l** for the length, the dynamics is described as follows:

$$\begin{cases} (m_1+m_2)l_1\ddot{\alpha_1} + m_2l_2\ddot{\alpha_2}\cos(\alpha_1-\alpha_2) + m_2l_2\dot{\alpha_2}^2\sin(\alpha_1-\alpha_2) + (m_1+m_2)\mathcal{G}\sin(\alpha_1) = 0 \\ m_2l_2\ddot{\alpha_2} + m_2l_1\ddot{\alpha_1}\cos(\alpha_1-\alpha_2) - m_2l_1\dot{\alpha_1}^2\sin(\alpha_1-\alpha_2) + m_2\mathcal{G}\sin(\theta_2) = 0 \end{cases} \tag{35}$$

In our study, we will fix $m_1 = m_2 = 1$ and $l_1 = l_2 = 8$ and $\mathcal{G}$ is the gravitational constant. We compute the FTLE for the $[-\pi/2, \pi/2, -2, 2]$ phase-space of the pendulum with an initial $\Delta_{x_0} = \Delta_{\dot{x}_0} = 0.01$ (Fig. 16).

Unlike the duffing oscillator, the double pendulum's phase-space exhibit chaotic dynamics in broader regions, typically when $\theta_2 > \frac{\pi}{2}$. The proposed models is not able to generalize for such as diverse set of dynamics. Focusing on a subs-space of the phase-space, namely [0, 1, 0, 1], we were able to train a model via use of a hyperparameter sweep :

Parameter	N_E	L_E	N_T	N_P	L_P	L_{RNN}
Value	13	0	18	21	4	1

Over a limited non-chaotic phase-space, the dynamics of the double pendulum can be learned, as per Figs. 17 and 18. This demonstrates versatility of the proposed model, in the sense that it can be repurposed for a variety of nonlinear dynamics, as long as the phase-space only contains few chaotic initial configurations.

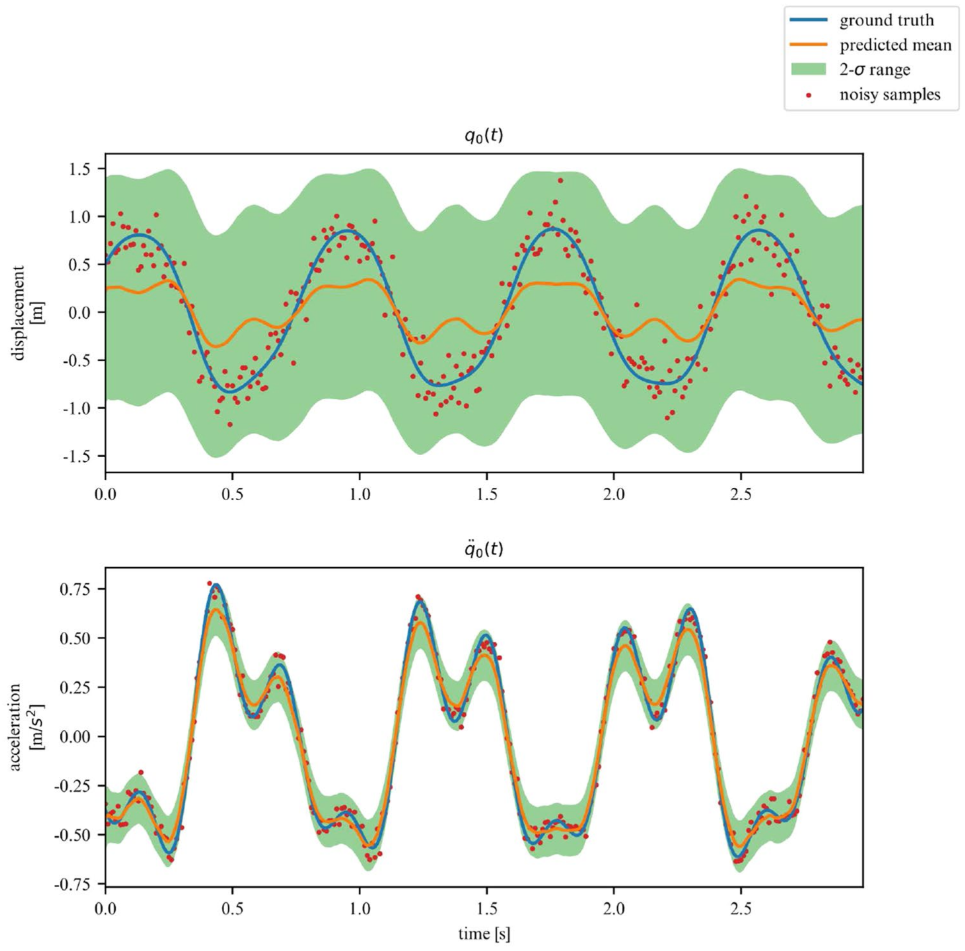

Figure 20. Partially observed linear 3DOF system.

Systems with partial observations.

While we deemed it useful for demonstration purposes in what was shown in the previous examples, the proposed framework does not require the observation of all DOFs. In this section, we demonstrate its performance on learning the dynamics of the previously examined systems, but this time for a subset of observations. More specifically, we now chose to only observe the acceleration and displacement of the first degree of freedom. We apply this approach to the 2DOF linear, 3DOF linear, 2DOF duffing and 2DOF pendulum systems. Samples of the observed time series are showcased in Figs. 19, 20, 21 and 22.

In these scenarios, the model has to deal with a greater level of uncertainty, and this is reflected in the widening of the variance band, particularly for displacements. However, overall, our model is able to generalize for linear and nonlinear systems, even when only a subset of the system's DOFs are observed (measured).

Nonautonomous systems.

Dissipation. We return to the coupled spring-mass system. However, we now add damping to both degrees of freedom. The corresponding equations of motion are now given by:

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{C}\dot{\mathbf{q}}(t) + \mathbf{K}_l\mathbf{q}(t) = \mathbf{0} \tag{36}$$

We retrain the same model as in section 4.2, but now with implementation of the dissipative integrator described in section 3.5.

Figure 23 illustrates that the herein proposed model delivers stable predictions on the dynamics of the model. However, we would further like to verify whether the model's latent space takes into account the loss of energy of the system. The results obtained in Fig. 24 are mixed. The model reflects the fact that the energy variations are diminishing over time and the model is converging towards a constant state. However, we are left with a large residual energy even when the system is at rest. This is because of the gauge invariance of the potential energy of this system; a trajectory can give us information on the gradient of the energy, but there are infinitely many possible starting points. Since we do not offer any prior information on the initial potential energy to the model, it will not be able to recover this information by simply observing the system's trajectories.

Forced dissipative nonlinear dynamics. We now additionally subject the dissipative version of a nonlinear system to external forces. In particular, we will study the effects of a sinusoidal excitation on a dissipative 2DOF duffing oscillator. The corresponding equations of motion are now given by:

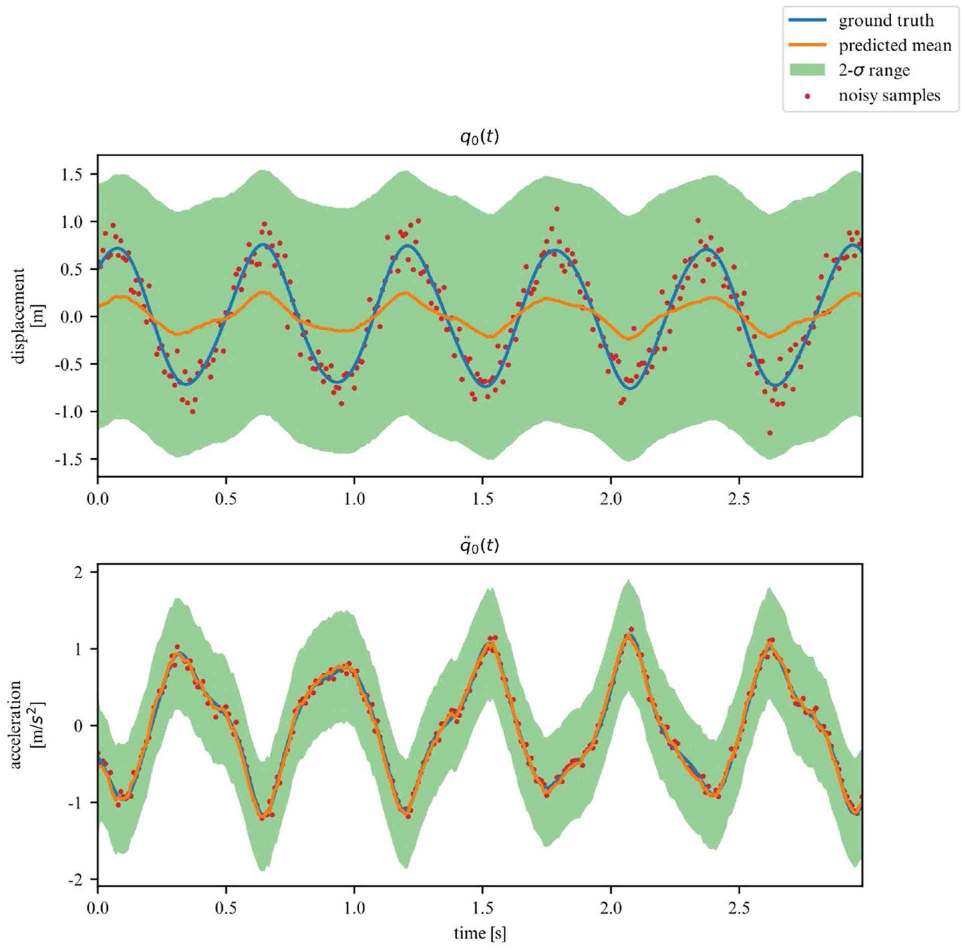

Figure 21. Partially observed duffing 2DOF system.

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{C}\dot{\mathbf{q}}(t) + \mathbf{K}\mathbf{q}(t) + \mathbf{K}_d\mathbf{q}^3(t) = \mathbf{D}\boldsymbol{\gamma}\sin(2\pi f t) \tag{37}$$

We retrain the 2DOF duffing oscillator model with values of $\boldsymbol{\gamma} = [0, 2]$ and $f = [0.5, 3]$. Here, the matrix $\mathbf{D}$ is used to project the excitation of a subset of DOFs. For our experiments, $\mathbf{D}_{1,1} = 1$ whereas the rest of $\mathbf{D}$ is left to be equal to 0 so as to apply the forcing on the first DOF. We only consider the first DOF to be directly excited. As examples, we display below the predictions for a fixed $f = 1.5$ and $\gamma = 0.3, 0.9, 2.0$ on Figs. 25, 26 and 27 respectively.

Comparison with other encoders. To compare our encoders with those of Krishnan et al.[12], we decide to measure the Mean Squared Error (MSE) between the predicted mean and the ground truth trajectory. Furthermore, we also count the percentage of noisy datapoints that are outside the $\mu \pm 2\sigma$ range. Each metric is computed over the test set (10% of the full dataset, i.e. 500 samples) and averaged. We implement the RNN and BiRNN encoders from the original DMM[12], our NODE encoder and our Symplectic NODE encoder. All experiments are performed with the same random seed.

Encoder	**RNN**		**BiRNN**		**ODE (Ours)**		**Symplectic ODE (Ours)**	
	MSE	Outlier	MSE	Outlier	MSE	Outlier	MSE	Outlier
Linear 2DOF	4.18	0.357%	4.12	0.288%	**0.73**	**0.038%**	1.56	0.056%
Linear 3DOF	1.12	0.175%	1.14	0.209%	1.02	0.244%	**0.76**	**0.031%**
Duffing 2DOF	1.69	38.954%	1.73	42.50%	1.88	2.71%	**1.01**	**0.88%**
Pendulum 2DOF	1.64	0.514%	1.59	0.316%	1.43	0.610%	**1.42**	**0.687%**
Linear 2DOF Dissipative	20.95	0.115%	15.04	0.146%	2.82	0.391%	**2.24**	**1.073%**
Duffing 2DOF Dissipative Sinusoidal Forcing	231.10	2.639%	147.13	0.654%	134.27	0.600%	**101.72**	**1.765%**

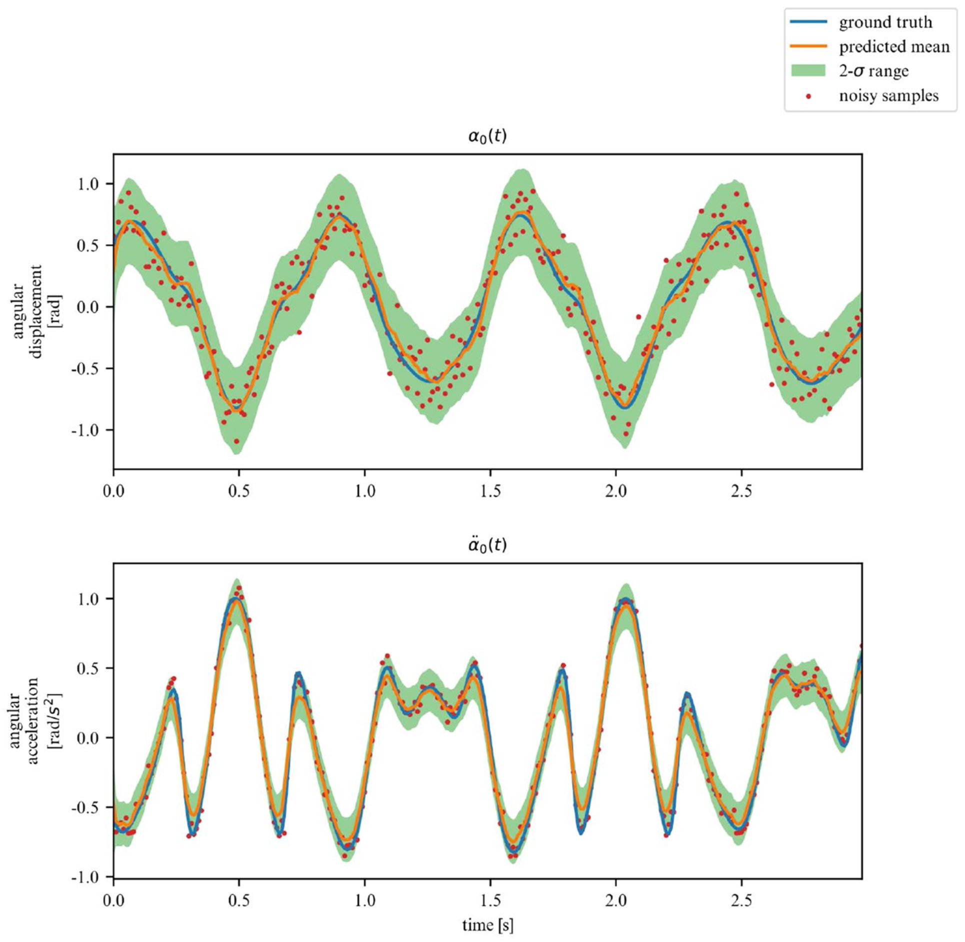

Figure 22. Partially observed double pendulum system.

In the most simple experiment, the base NODE encoder is able to outperform its symplectic counterpart. However, for more complex dynamics, the symplectic encoder is able to make more accurate predictions than the other encoders.

Conclusion

We presented a variation of the DMM that is able to learn a physics-informed latent representation from noisy samples of a MDOF system. In particular, the use of symplectic encoders, derived from the Hamiltonian formalism, successfully introduces the property of energy presentation to the latent space. Our model is able to learn the dynamics of a variety of linear and nonlinear system dynamics, namely linear systems, duffing oscillators and double pendulums. Our models can also be applied to the non-autonomous case. In physical terms, we are able to account for both dissipation and external forces.

Our future works in the field of dynamical system will focus on incorporating additional latent biases in our model. For example, the exact measure of the energy is still a problem to be solved. In addition, we will also explore way to scale our solutions to systems with a much larger number of DOFs.

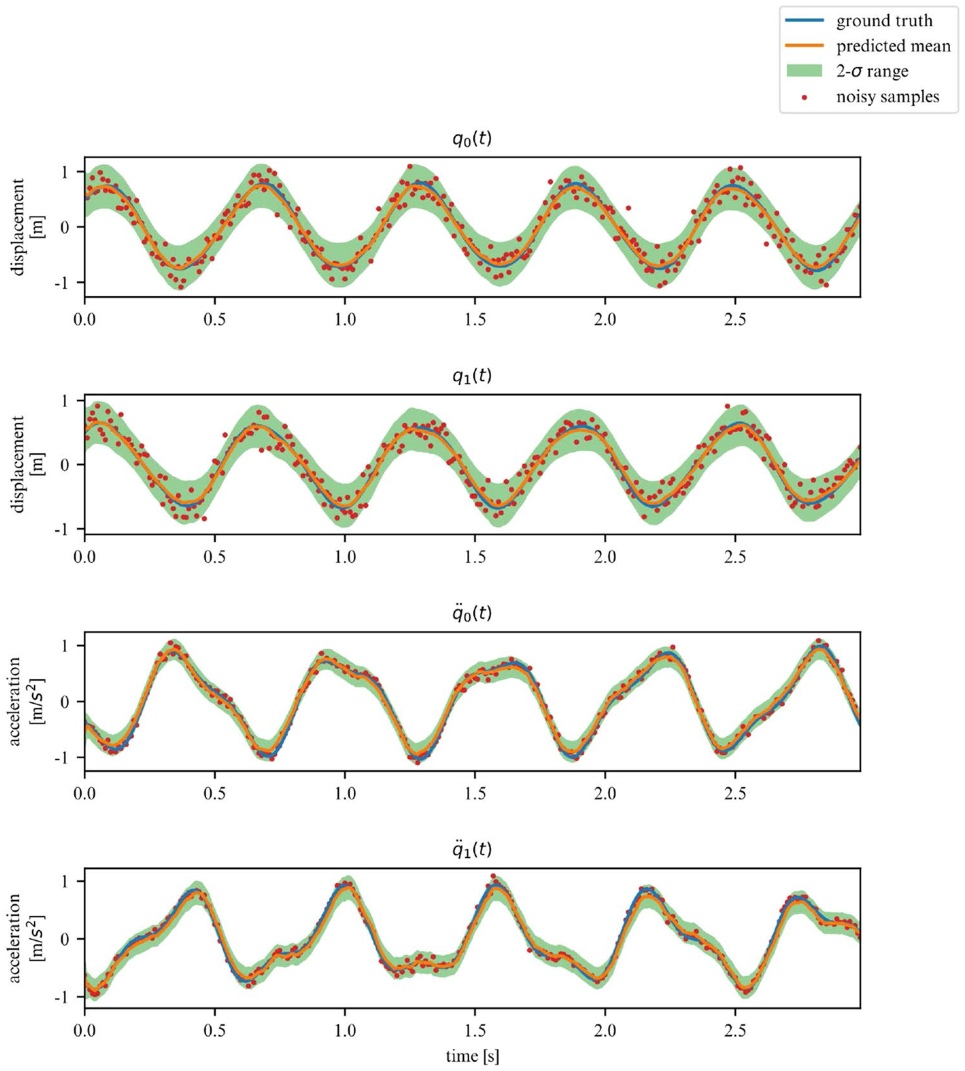

Figure 23. Linear dissipative 2DOF system observations.

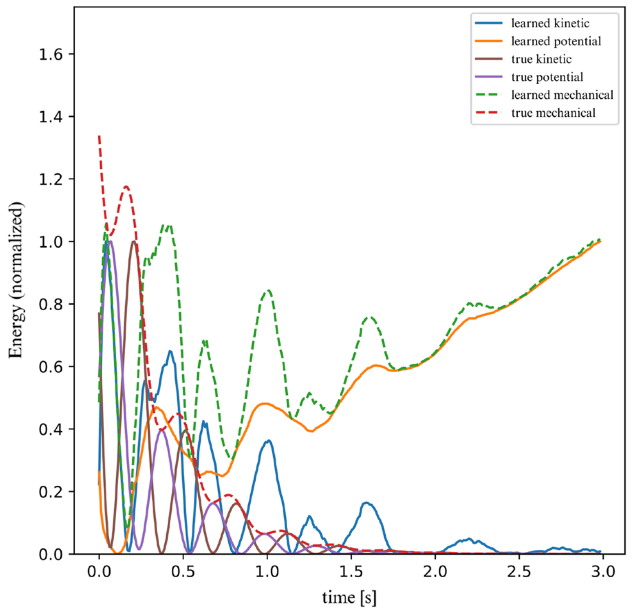

Figure 24. Linear dissipative 2DOF system energy.

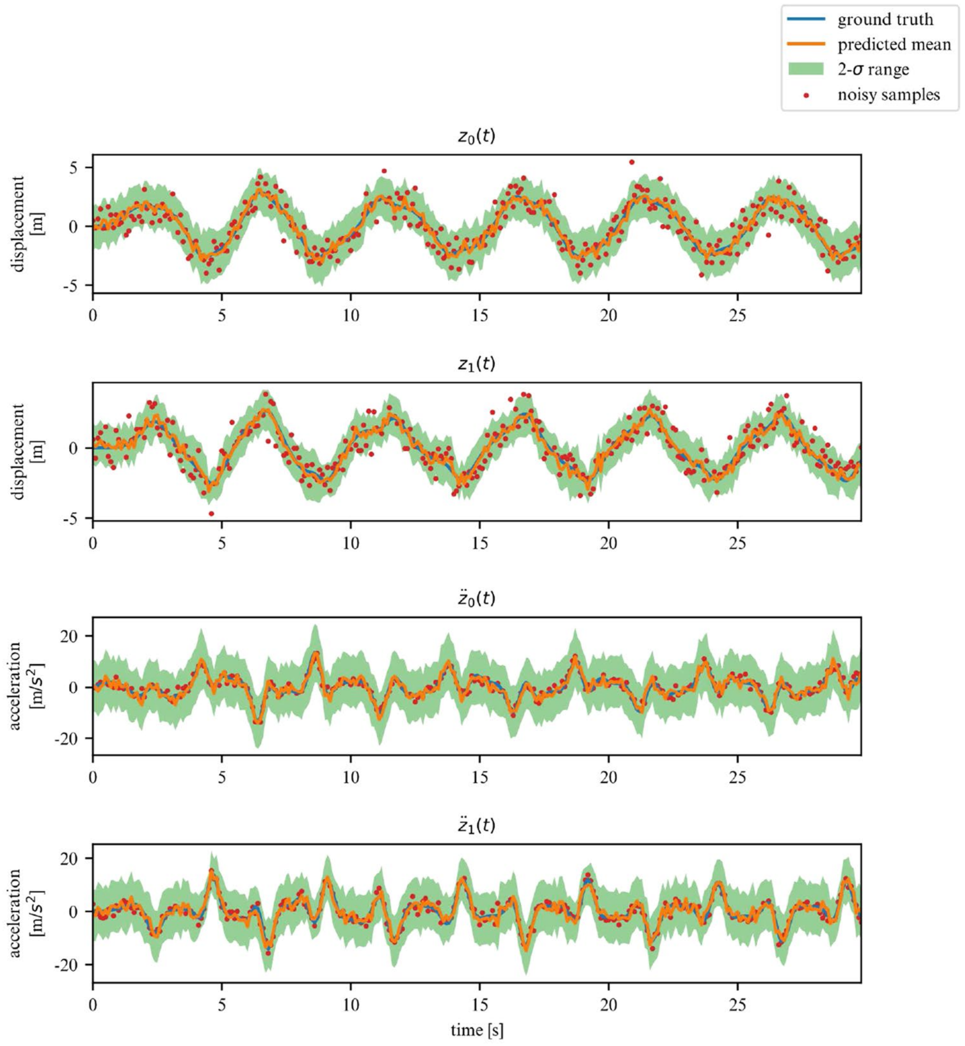

Figure 25. Forced dissipative 2DOF system observations for $\gamma = 0.3$.

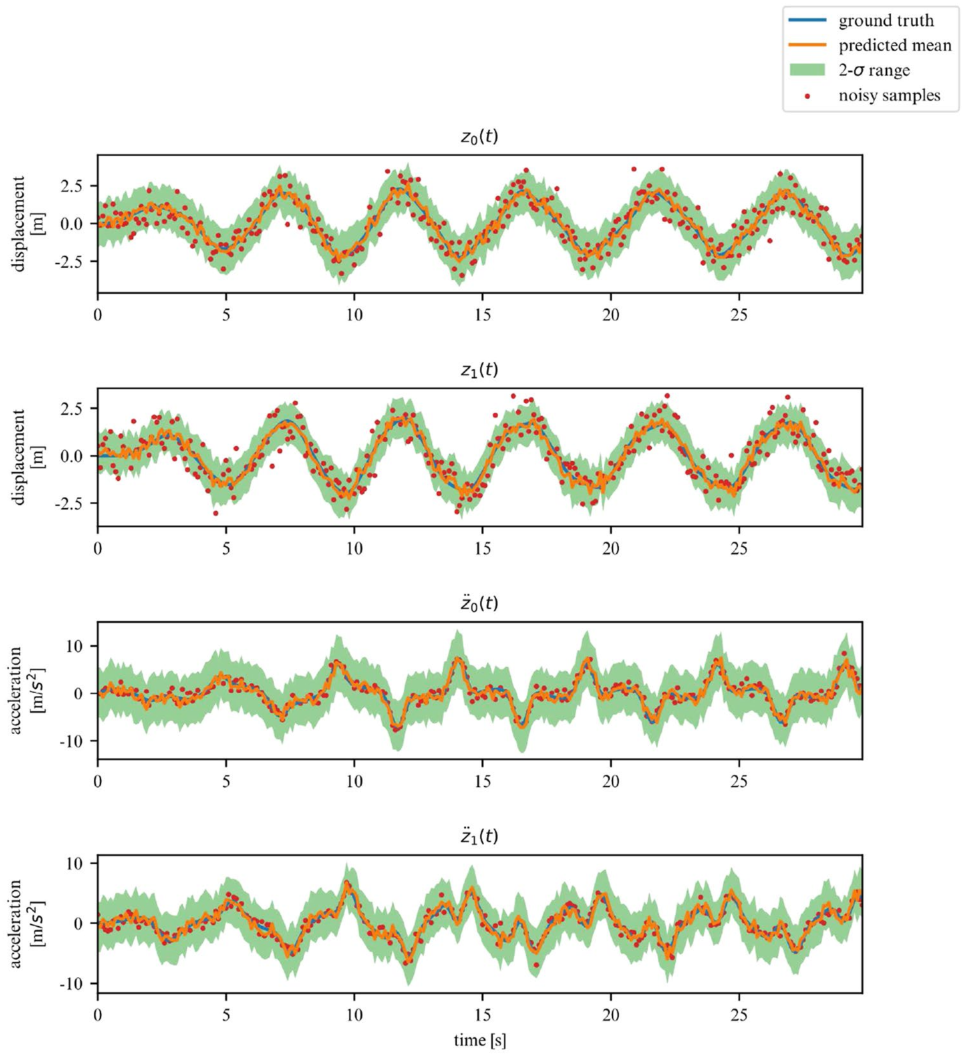

Figure 26. Forced dissipative 2DOF system observations for $\gamma = 0.9$.

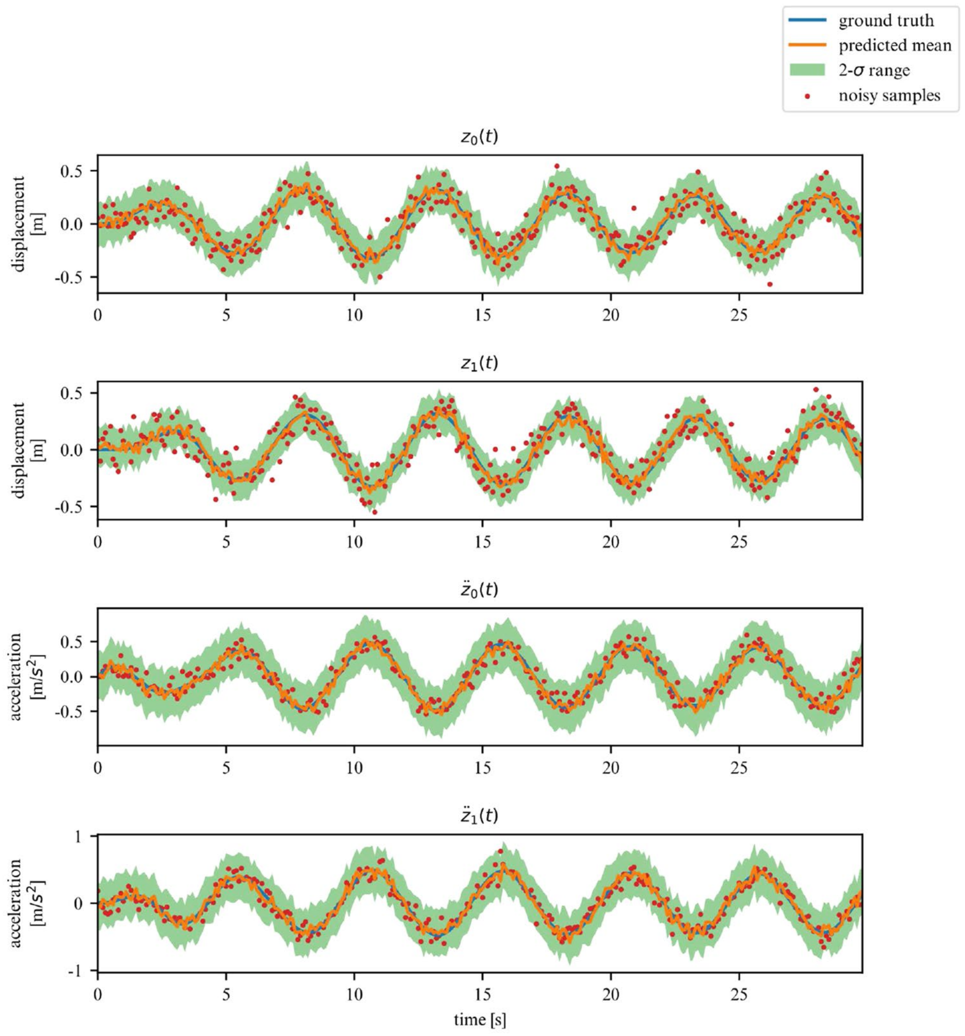

Figure 27. Forced dissipative 2DOF system observations for $\gamma = 2.0$.

Data availability

All datasets used to train and validate the models can be generated using the "simulate.py" script from https://github.com/kbacsa-ethz/phys-stoch with a desktop computer within approximately 10 min.

Received: 29 September 2022; Accepted: 31 January 2023
Published online: 14 February 2023

References

1. Arulampalam, M., Maskell, S., Gordon, N. & Clapp, T. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. Signal Process.* **50**, 174–188. https://doi.org/10.1109/78.978374 (2002).
2. Särkkä, S. *Bayesian Filtering and Smoothing* (Cambridge University Press, Cambridge, 2013).
3. Kalman, R. E. A new approach to linear filtering and prediction problems. *Trans. ASME J. Eng.* **82**, 35–45 (1960).
4. Chatzi, E. N. & Smyth, A. W. The unscented kalman filter and particle filter methods for nonlinear structural system identification with non-collocated heterogeneous sensing. *Struct. Control Health Monit.* **16**, 99–123, https://doi.org/10.1002/stc.290 (2009).
5. Dertimanis, V., Chatzi, E., Eftekhar Azam, S. & Papadimitriou, C. Input-state-parameter estimation of structural systems from limited output information. *Mech. Syst. Signal Process.* **126**, 711–746, https://doi.org/10.1016/j.ymssp.2019.02.040 (2019).
6. Kitagawa, G. & Gersch, W. *Linear Gaussian State Space Modeling* 55–65 (Springer, NY, 1996).
7. Zhang, K. & Hyvärinen, A. A general linear non-Gaussian state-space model: Identifiability, identification, and applications. In Hsu, C.-N. & Lee, W. S. (eds.) *Proc. of the Asian Conf. on Machine Learning*, vol. 20 of *Proc. of Machine Learning Research*, 113–128 (PMLR, South Garden Hotels and Resorts, Taoyuan, Taiwain, 2011).
8. Eleftheriadis, S., Nicholson, T. F. W., Deisenroth, M. P. & Hensman, J. Identification of Gaussian Process State Space Models. *arXiv:1705.10888 [stat]* (2017).
9. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]* (2014).
10. Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv:1401.4082 [cs, stat]* (2014).
11. Bayer, J. & Osendorfer, C. Learning stochastic recurrent networks *arXiv preprint arXiv:1411.7610* (2015).
12. Krishnan, R. G., Shalit, U. & Sontag, D. Structured inference networks for nonlinear state space models. *arXiv:1609.09869 [cs, stat]* (2016).
13. Girin, L. *et al.* Dynamical variational autoencoders: A comprehensive review. *arXiv:2008.12595 [cs, stat]* (2020).
14. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acade. Sci.* **113**, 3932–3937, https://doi.org/10.1073/pnas.1517384113 (2016). https://www.pnas.org/content/113/15/3932.full.pdf.
15. Chen, R. T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. Neural ordinary differential equations. *arXiv:1806.07366[cs, stat]* (2019)
16. Zhong, G. & Marsden, J. E. Lie-poisson hamilton-jacobi theory and lie-poisson integrators. *Phys. Lett. A* **133**, 134–139. https://doi.org/10.1016/0375-9601(88)90773-6 (1988).
17. Chen, Z., Zhang, J., Arjovsky, M. & Bottou, L. Symplectic recurrent neural networks. *arXiv:1909.13334 [cs, stat]* (2020).
18. Zhong, Y. D., Dey, B. & Chakraborty, A. Symplectic ODE-Net: Learning hamiltonian dynamics with control. *arXiv:1909.12077 [physics, stat]* (2020
19. Krishnan, R. G., Shalit, U. & Sontag, D. Deep Kalman filters. *arXiv:1511.05121 [cs, stat]* (2015)
20. Chung, J. *et al. Recurr. Latent Var. Model Seq. Data* **1506**, 02216 (2016).
21. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
22. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* **1512**, 03385 (2015).
23. Rubanova, Y., Chen, R. T. Q. & Duvenaud, D. Latent odes for irregularly-sampled time series. *Adv. Neural Inform. Process. Syst.* **1907**, 03907 (2019).
24. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. **1711**, 10561 (2017).
25. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid mechanics. *Ann. Rev. Fluid Mech.* **52**, 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214 (2020).
26. Mao, Z., Jagtap, A. D. & Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **360**, 112789. https://doi.org/10.1016/j.cma.2019.112789 (2020).
27. Stoffel, M., Gulakala, R., Bamer, F. & Markert, B. Artificial neural networks in structural dynamics: A new modular radial basis function approach vs. convolutional and feedforward topologies. *Comput. Methods Appl. Mech. Eng.* **364**, 112989. https://doi.org/10.1016/j.cma.2020.112989 (2020).
28. Zhang, R., Liu, Y. & Sun, H. Physics-informed multi-lstm networks for metamodeling of nonlinear structures. *Comput. Methods Appl. Mech. Eng.* **369**, 113226. https://doi.org/10.1016/j.cma.2020.113226 (2020).
29. Brink, A. R. & Najera-Flores, D. A. Efficient random vibration analysis of nonlinear systems with long short-term memory networks for uncertainty quantification. *Conference: ISMA 2018 International Conf. on Noise and Vibration Engineering and USD2018 International Conf. on Uncertainty in Structural Dynamics* (2018).
30. Lai, Z., Mylonas, C., Nagarajaiah, S. & Chatzi, E. Structural identification with physics-informed neural ordinary differential equations. *J Sound Vib.* **508**, 116196. https://doi.org/10.1016/j.jsv.2021.116196 (2021).
31. Liu, W., Lai, Z., Bacsa, K. & Chatzi, E. Physics-guided deep markov models for learning nonlinear dynamical systems with uncertainty. *Mech. Syst. Signal Process.* **178**, 109276 (2021).
32. Geneva, N. & Zabaras, N. *Multi-fidelity generative deep learning turbulent flows*https://doi.org/10.3934/fods.2020019 *(2020).*
33. Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U. & Vollgraf, R. Multivariate probabilistic time series forecasting via conditioned normalizing flows. *ARXIV:2002.06103* (2020).
34. Greydanus, S., Dzamba, M. & Yosinski, J. Hamiltonian neural networks. *arXiv:1906.01563 [cs]* (2019).
35. Saemundsson, S., Terenin, A., Hofmann, K. & Deisenroth, M. P. Variational integrator networks for physically structured embeddings. *arXiv:1910.09349 [cs, stat]* (2020).
36. Toth, P. *et al.* Hamiltonian generative networks. *arXiv:1909.13789 [cs, stat]* (2020).
37. Rusch, T. K. & Mishra, S. Unicornn: A recurrent model for learning very long time dependencies In *International Conf. on Machine Learning*. (PLMR, 2021).
38. Wolf, C., Karl, M. & van der Smagt, P. Variational inference with hamiltonian monte carlo. *ArXiv:1609.08203* (2016).
39. Caterini, A. L., Doucet, A. & Sejdinovic, D. Hamiltonian variational auto-encoder. *ArXiv.1805.11328* (2018).
40. Neal., R. M. Hamiltonian importance sampling In *talk presented at the Banff International Research Station (BIRS) workshop on Mathematical Issues in Molecular Dynamics*(BIRF, Bannf, 2005).
41. Wang, Z. & Delingette, H. Quasi-symplectic langevin variational autoencoder. *ArXiv:2009.01675* (2020).

42. Huang, C., Krueger, D., Lacoste, A. & Courville, A. C. Neural autoregressive flows. *ArXiv:1804.00779* (2018).
43. Burnham, K. & Anderson, D. *Model Selection and Multimodel Inference: A Practical Information-theoretic approach* (Springer Verlag, Berlin, 2002).
44. Pontrjagin, L., Boltyanskii, V., Gamkrelidze, R., Mishchenko, E. & Brown, D. *The Mathematical Theory of Optimal Processes*. International series of monographs in pure and applied mathematics (Wiley, 1962).
45. Hairer, E., Lubich, C. & Wanner, G. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. No. 31 in Springer series in computational mathematics (Springer, Berlin ; New York, 2006), 2nd edn. OCLC: ocm69223213.
46. França, G., Jordan, M. I. & Vidal, R. On dissipative symplectic integration with applications to gradient-based optimization. *J. Stat. Mech. Theory Exp.* **2021**, 043402. https://doi.org/10.1088/1742-5468/abf5d4 (2021).
47. Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems 32*, 8024–8035 (Curran Associates, Inc., 2019).
48. Ishikawa, F. Implementation of 4th order symplectic integrator with adjoint method by fishikawa . pull request n127 . rtqichen/torchdiffeq (2020).
49. Bingham, E. *et al.* Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.* **20**, 1–6 (2019).
50. Fehlberg, E. Klassische runge-kutta-formeln vierter und niedrigerer ordnung mit schrittweiten-kontrolle und ihre anwendung auf wärmeleitungsprobleme. *Computing* **6**, 61–71. https://doi.org/10.1007/BF02241732 (1970).
51. Rezende, D. J. & Mohamed, S. *Variational inference with normalizing flows* **1505**, 05770 (2016).
52. Chipman, H. A., George, E. I. & McCulloch, R. E. Bayesian cart model search. *J. Am. Stat. Assoc.* **93**, 935–948. https://doi.org/10.1080/01621459.1998.10473750 (1998).

Acknowledgements
The research was conducted at the Singapore-ETH Centre, which was established collaboratively between ETH Zurich and the National Research Foundation Singapore. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. The project is also supported by the Stavros Niarchos Foundation through the ETH Zurich Foundation and the ETH Zurich Postdoctoral Fellowship scheme.

Author contributions
K.B. conceived the model and the experiments, Z.L. and W.L. contributed to the code implementations, E.C. and M.T. provided extensive feedback on the method and experiments. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to K.B.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.