Hamiltonian reduction using a convolutional auto-encoder coupled to an Hamiltonian neural network

Raphaël Côte, Emmanuel Franck, Laurent Navoret, Guillaume Steimer, Vincent Vigon

Hamiltonian reduction using a convolutional auto-encoder coupled to an Hamiltonian neural network

Raphaël Côte[1], Emmanuel Franck[1, 2], Laurent Navoret[1, 2], Guillaume Steimer[1, 2], and Vincent Vigon[1, 2]

[1]Institut de Recherche Mathématique Avancée, UMR 7501 Université de Strasbourg et CNRS, 7 rue René Descartes 67000 Strasbourg, France
[2]INRIA Nancy-Grand Est, TONUS Project, Strasbourg, France

Abstract

The reduction of Hamiltonian systems aims to build smaller reduced models, valid over a certain range of time and parameters, in order to reduce computing time. By maintaining the Hamiltonian structure in the reduced model, certain long-term stability properties can be preserved. In this paper, we propose a non-linear reduction method for models coming from the spatial discretization of partial differential equations: it is based on convolutional auto-encoders and Hamiltonian neural networks. Their training is coupled in order to simultaneously learn the encoder-decoder operators and the reduced dynamics. Several test cases on non-linear wave dynamics show that the method has better reduction properties than standard linear Hamiltonian reduction methods.

Keywords: Hamiltonian dynamics, model order reduction, convolutional auto-encoder, Hamiltonian neural network, non-linear wave equations, shallow water equation

AMS subject classifications: 65P10, 34C20, 68T07

1 Introduction

Hamiltonian reduced order modeling techniques have been successfully developed in order to perform accelerated numerical simulations of some parameterized Hamiltonian models of large dimension [20, 12, 25]. The spatial discretization of some wave-like partial differential equations gives rise to very large such Hamiltonian systems. Reduced order models can be essential for real-time simulations or when a large number of simulation instances are required as part of a control, optimisation or uncertainty quantifications algorithm. Starting from the initial model, a large differential system, the methods consist into constructing a differential system of a smaller size that can produce valid approximate solutions for a predefined range of times and parameters. Many physical models have a Hamiltonian structure and this gives the system a certain number geometrical properties like the conservation

of energy. In particular, the preservation of this structure at the discrete level enables to ensure large-time stability of the numerical simulations [10]. In order to build consistent and robust reduced models, it is therefore interesting to preserve this Hamiltonian structure through the reduction.

The construction of reduced models can be divided in two steps: (i) find a so-called pair of encoder and decoder operators that goes from the full to the reduced variables and inversely; (ii) identify the dynamics followed by the reduced variables. The construction of the encoder and decoder operators relies on a large number of data produced by numerical simulations in the range of time and parameters of interest.

The first approach to reduce a large Hamiltonian system relies on a linear approximation: the solutions manifold is approximated with a hyperplane of small dimension [20]. The encoder is here a linear mapping, which is also assumed to be symplectic so that the Hamiltonian structure can preserved for the reduced model. Such symplectic mapping can be constructed from data through greedy algorithms [17] or through a Singular Value Decomposition (SVD) methodology: this is the Proper Symplectic Decomposition (PSD) proposed in [20]. In this work, several algorithms have been proposed to define approximated optimal symplectic mapping: for instance, the cotangent-lift algorithm devise a symplectic mapping which is also orthogonal and have a block diagonal structure. Then the reduced model is obtained using the Galerkin projection method: the model is constructed by supposing that a symplectic projection of the residual vanishes, where the residual stands for the error obtained after replacing the original variables by the decoded reduced variables.

Such linear reductions however can hardly handle non-linear dynamics: this is the case for convected-dominated or non-linear wave like problems for which the solution manifold is badly approximated by hyperplanes. In order to build more expressive reduced models, one possibility is to consider time adaptative reduced methods [19, 11]. Another widely investigated possibility is to consider non-linear reduction methods.

Regarding the construction of non-linear encoder-decoder operators, a first class of methods rely on manifold learning techniques [22, 3]. Such methods are based on the geometrical analysis of the neighbors graph of the data thanks to the computation of geodesic distances (ISOMAP method, [23]), of eigenfunctions of the graph Laplacian (EigenMaps method [1]) or of diffusion processes (DiffusionMaps method [6]). This provides reduced variables for each data that can be further interpolated using the Nyström formula [2].

Since the explosion of deep learning in the early 2010s, new dimension reduction methods grounded on neural networks have been developed. The convolutional auto-encoder architecture [8] seems particularly appropriate since its very purpose is to determine latent variables: the neural network is indeed divided into an encoder part and decoder part and they are trained simultaneously so that the sequence of encoder and decoder is close to the identity map. This was originally developed for image generation, but has been also used for reduced order modeling for models coming from the spatial discretization of partial differential equations on a grid [15, 13, 21]. In [4], the authors use auto-encoder neural networks for Hamiltonian reduction: the encoder and the decoder are weakly constrained to be symplectic thanks to a penalization term in the cost functional.

Once non-linear encoders and decoders have been devised from data, the dynamics of the reduced variables still has to be determined. Two strategies can be considered. The first one relies on a Galerkin projection of the Hamiltonian system as in the linear case [4]. Note that the reduced model is indeed Hamiltonian provided the decoder is a symplectic map. However, the reduced model still requires the evaluation of the vector field in the original large dimension space of size $2N$: this is a well-known difficulty in non-linear reduction.

To overcome this difficulty, hyper-reduction methods have been proposed like the discrete empirical interpolation method (DEIM) [5] but they have to be adapted to not destroy the symplectic structure of the equations [11].

Another approach is to learn the dynamics of the reduced variables using a neural network: given the initial state, the neural network provides the full trajectory. As the learning is done directly in the reduced dimension, the obtained reduced model does not require an evaluation of non-linear terms in the original variables: this is a clear advantage of the method compared with projection-based ones. The reduced dynamics can be captured for instance by Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM) neural networks [18], or by fully connected networks [7]. This has also been considered as correction of the Galerkin-type reduced models [26].

Here we consider another strategy which consists in learning the vector field that generates the observed reduced dynamics. The neural network is trained so as to minimize its deviation from the finite difference time derivative of the reduced data obtained after encoding. As we aim at conserving the Hamiltonian structure at the reduced level, the vector field is further supposed to be associated to a reduced Hamiltonian function. Therefore, we can learn directly the Hamiltonian function instead of the vector field. This is a so-called Hamiltonian Neural Network (HNN) strategy proposed in [9] where a symplectic time integrator is used.

The present paper proposes to combine an auto-encoder strategy for the encoding-decoding part and a HNN method to learn the reduced dynamics: this will be referred to as the AE+HNN method. Note that there is a priori no reason for the auto-encoder neural networks to spontaneously provide reduced variables compatible with Hamiltonian dynamics. Therefore, some constraints on the auto-encoder have to be added. This can be done by imposing symplecticity as in [4]. Here, we propose instead to train it simultaneously with the HNN. With this joint training, the auto-encoder will gradually converge to a set of reduced variables compatible with a Hamiltonian system. Of course, this means that the losses associated with each neural network must be weighted judiciously during training. Note that such a joint training of the encoding-decoding operators and the reduce dynamics have been explored in [7], but without considering Hamiltonian structures.

The outline of the article is as follows. In Section 2, we introduce parameterized Hamiltonian systems as well as the main steps for the construction of reduced order models. Section 3 then presents the non-linear AE+HNN reduction method. In particular we describe the architectures and the loss functions used for the trainings. Finally Section 4 is devoted to the numerical results: we apply our reduction method on the Hamiltonian systems obtained after spatial discretization of linear, non-linear wave equations and a shallow water system and compare it with the linear PSD reduction technique.

2 Parameterized Hamiltonian systems and reduction

In this section, we introduce the notations used for the parameterized Hamiltonian systems and the main steps for the construction of a reduced model.

In the following, we often write vectors of interest with bold script letters, operators with capital italic letters, with their parameters as indices, and overline quantities when related to the reduced model.

2.1 Parameterized Hamiltonian dynamics

We consider a parameterized autonomous Hamiltonian system, whose solution, $\mathbf{y}(t;\mu) \in \mathbb{R}^{2N}$ with $N \in \mathbb{N}^*$, depends on time $t \in [0,T]$, with $T > 0$, and on a parameter $\mu \in \Xi \subset \mathbb{R}^d$, with $d \in \mathbb{N}$. The dynamics derives from a given Hamiltonian function $\mathcal{H} : \mathbb{R}^{2N} \times \Xi \to \mathbb{R}$ and writes

$$\begin{cases} \dfrac{d}{dt}\mathbf{y}(t;\mu) = J_{2N}\,\nabla_{\mathbf{y}}\mathcal{H}\left(\mathbf{y}(t;\mu);\mu\right), \quad \forall t \in (0,T], \\ \mathbf{y}(0;\mu) = \mathbf{y}^0(\mu), \end{cases} \tag{1}$$

where $\mathbf{y}^0(\mu) \in \mathbb{R}^{2N}$ is a given initial condition and J_{2N} refers to the canonical symplectic matrix

$$J_{2N} = \begin{pmatrix} 0_N & I_N \\ -I_N & 0_N \end{pmatrix},$$

with I_N the identity matrix of dimension N. Introducing the canonical coordinates $\mathbf{y} = (\mathbf{q},\mathbf{p})^T$, the system becomes:

$$\begin{cases} \dfrac{d}{dt}\mathbf{q}(t;\mu) = \nabla_{\mathbf{p}}\mathcal{H}\left(\mathbf{q},\mathbf{p};\mu\right), \quad \forall t \in (0,T], \\ \dfrac{d}{dt}\mathbf{p}(t;\mu) = -\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q},\mathbf{p};\mu\right), \quad \forall t \in (0,T], \\ \mathbf{q}(0;\mu) = \mathbf{q}^0(\mu), \\ \mathbf{p}(0;\mu) = \mathbf{p}^0(\mu). \end{cases} \tag{2}$$

with $\mathbf{q}^0, \mathbf{p}^0 \in \mathbb{R}^N$ such that $\mathbf{y}^0 = (\mathbf{q}^0,\mathbf{p}^0)^T$. A key property of such systems is that the associated flow is symplectic, meaning that $\phi_t\left(\mathbf{y}^0(\mu);\mu\right) = \mathbf{y}(t;\mu)$ satisfies the relation

$$\left(\nabla_{\mathbf{y}}\phi_t\left(\mathbf{y}^0(\mu);\mu\right)\right)^T J_{2N}\left(\nabla_{\mathbf{y}}\phi_t\left(\mathbf{y}^0(\mu);\mu\right)\right) = J_{2N}.$$

One consequence is that the Hamiltonian $\mathcal{H}$ is preserved along the flow

$$\forall t > 0, \mu \in \Xi, \quad \mathcal{H}\left(\mathbf{y}(t;\mu);\mu\right) = \mathcal{H}\left(\mathbf{y}^0(\mu);\mu\right),$$

which is of particular importance when considering physical systems.

In this work, we are specifically interested in Hamiltonian systems resulting from the space discretization of one-dimensional wave-type equations. In such systems, $\mathbf{q} \in \mathbb{R}^N$ refers to the height of the wave at grid points and $\mathbf{p} \in \mathbb{R}^N$ to the velocity of the wave also at grid points. Examples will be detailed in the numerical section.

In order to provide numerical approximation of the solution, specific numerical schemes have been developed to ensure the symplectic property at the discrete level [10]. These schemes also guarantee large time stability of the numerical solutions. Here we consider the standard second-order Störmer-Verlet scheme. Denoting $\mathbf{y}_\mu^n = \left(\mathbf{q}_\mu^n, \mathbf{p}_\mu^n\right)^T \in \mathbb{R}^{2N}$ the approximate solution at time $t^n = n\Delta t$, with time step $\Delta t > 0$, one iteration of the scheme is defined by:

$$\begin{cases} \mathbf{p}_\mu^{n+\frac{1}{2}} = \mathbf{p}_\mu^n - \frac{1}{2}\Delta t\,\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q}_\mu^n, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right), \\ \mathbf{q}_\mu^{n+1} = \mathbf{q}_\mu^{n+\frac{1}{2}} \Delta t\,\left[\nabla_{\mathbf{p}}\mathcal{H}\left(\mathbf{q}_\mu^n, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right) + \nabla_{\mathbf{p}}\mathcal{H}\left(\mathbf{q}_\mu^{n+1}, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right)\right], \\ \mathbf{p}_\mu^{n+1} = \mathbf{p}_\mu^{n+\frac{1}{2}} - \frac{1}{2}\Delta t\,\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q}_\mu^{n+1}, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right). \end{cases} \tag{3}$$

Under the further assumption that the Hamiltonian $\mathcal{H}$ is separable, i.e. is the sum of a function depending only $\mathbf{q}$ and another depending only $\mathbf{p}$:

$$\mathcal{H}(\mathbf{y};\mu) = \mathcal{H}^1(\mathbf{q};\mu) + \mathcal{H}^2(\mathbf{p};\mu),$$

the implicit first two steps of (3) become explicit and the scheme simplifies into:

$$\begin{cases} \mathbf{p}_\mu^{n+\frac{1}{2}} = \mathbf{p}_\mu^n - \frac{1}{2}\Delta t\, \nabla_{\mathbf{q}} \mathcal{H}^1\left(\mathbf{q}_\mu^n;\mu\right), \\ \mathbf{q}_\mu^{n+1} = \mathbf{q}_\mu^n + \Delta t\, \nabla_{\mathbf{p}} \mathcal{H}^2\left(\mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right), \\ \mathbf{p}_\mu^{n+1} = \mathbf{p}_\mu^{n+\frac{1}{2}} - \frac{1}{2}\Delta t\, \nabla_{\mathbf{q}} \mathcal{H}^1\left(\mathbf{q}_\mu^{n+1};\mu\right). \end{cases} \tag{4}$$

2.2 Hamiltonian reduced order modeling

Solving Hamiltonian systems with large dimension $2N \gg 1$ numerically can be relatively costly, and this is especially true when we want to solve a large number of them for a parametric study, for example. Therefore, methods have been developed in order to construct reduced Hamiltonian systems of smaller size $2K \ll 2N$, which capture the main dynamics for a range of times t and reduction parameters μ.

We first have to define an appropriate change of variable. To do that, we search for a $2K$-dimensional manifold $\widehat{\mathcal{M}}$ that approximates well the manifold

$$\mathcal{M} = \{\mathbf{y}(t;\mu) \text{ with } t \in [0,T], \mu \in \Xi\} \subset \mathbb{R}^{2N}$$

formed by the solutions. The manifold $\widehat{\mathcal{M}}$ is defined thanks to a so-called decoding operator $\mathcal{D}_{\theta_d} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$:

$$\widehat{\mathcal{M}} = \left\{\mathcal{D}_{\theta_d}(\overline{\mathbf{y}}) \text{ with } \overline{\mathbf{y}} \in \mathbb{R}^{2K}\right\} \subset \mathbb{R}^{2N}.$$

We also consider a pseudo-inverse operator $\mathcal{E}_{\theta_e} : \mathbb{R}^{2N} \to \mathbb{R}^{2K}$, called the encoder, which satisfies the relation

$$\mathcal{E}_{\theta_e} \circ \mathcal{D}_{\theta_d} = \mathrm{Id}_{\mathbb{R}^{2K}}.$$

To determine $\mathcal{D}_{\theta_d}$ and $\mathcal{E}_{\theta_e}$, we therefore ask for the projection operator $\mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}$ onto $\widehat{\mathcal{M}}$ to be close to the identity on a data set $U \subset \mathcal{M}$:

$$\forall u \in U, \quad \mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}(u) \approx u. \tag{5}$$

The data set U is composed of snapshots of the solutions at different times and various parameters, obtained with the symplectic algorithm defined above in (3); it writes

$$U = \left\{\mathbf{y}_{\mu_1}^0, \ldots, \mathbf{y}_{\mu_1}^M, \ldots, \mathbf{y}_{\mu_P}^0, \ldots, \mathbf{y}_{\mu_P}^M\right\},$$

where $M \in \mathbb{N}^*$ is the number of time-step chosen and $P \in \mathbb{N}^*$ the number of sampled parameters.

Once the changes of variables are defined, we are interested in the dynamics in the reduced variable:

$$\overline{\mathbf{y}}(t;\mu) = \mathcal{E}_{\theta_e}(\mathbf{y}(t;\mu)) \in \mathbb{R}^{2K}.$$

With specific constraints on the change of variables, we can expect the reduced variables to follow a reduced Hamiltonian dynamics:

$$\begin{cases} \dfrac{d}{dt}\overline{\mathbf{y}}(t;\mu) = J_{2K}\nabla_{\overline{\mathbf{y}}}\overline{\mathcal{H}}_{\theta_h}(\overline{\mathbf{y}}(t;\mu);\mu), \quad \forall t \in (0,T], \\ \overline{\mathbf{y}}(0;\mu) = \mathcal{E}_{\theta_e}(\mathbf{y}^0(\mu)), \end{cases} \tag{6}$$

where $\overline{\mathcal{H}}_{\theta_h} : \mathbb{R}^{2K} \times \Xi \to \mathbb{R}$ is a reduced Hamiltonian to be built.

The most common approach for Hamiltonian reduced order modeling is called the Proper Symplectic Decomposition (PSD) [20]. This method is briefly described in Appendix A. Although efficient for linear dynamics, it fails into reducing non-linear ones. This is why several non-linear Hamiltonian reduction techniques have been developed [4, 11]. In the next section, we present a strategy based on the coupling of an Auto-Encoder (AE) and an Hamiltonian Neural Network (HNN) method.

3 A non-linear Hamiltonian reduction method

The method proposed in this work consists in constructing the Hamiltonian reduced model via neural networks. More precisely, we aim at defining the following three neural networks:

- a decoder $\mathcal{D}_{\theta_d} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$,
- an encoder $\mathcal{E}_{\theta_e} : \mathbb{R}^{2N} \to \mathbb{R}^{2K}$,
- a reduced Hamiltonian $\overline{\mathcal{H}}_{\theta_h} : \mathbb{R}^{2K} \times \Xi \to \mathbb{R}$,

where $(\theta_d, \theta_e, \theta_h)$ stands for their parameters, such that the resulting reduced dynamics provides a good approximation of the initial one. An auto-encoder strategy will be used to define $\mathcal{D}_{\theta_d}$ and $\mathcal{E}_{\theta_e}$ while a Hamiltonian Neural Network will be considered for $\overline{\mathcal{H}}_{\theta_h}$. Note that the encoder and decoder are not enforced to be symplectic but the reduced model is.

Figure 1 illustrates how the reduced model is expected to be used for prediction. The initial condition $\mathbf{y}(t=0;\mu)$ is converted by the encoder $\mathcal{E}_{\theta_e}$ to the reduced initial condition $\overline{\mathbf{y}}(t=0;\mu)$. Then several iterations of the Störmer-Verlet scheme with the reduced Hamiltonian $\overline{\mathcal{H}}_{\theta_h}$ are performed to obtain an approximated reduced solution $\overline{\mathbf{y}}(t=T;\mu)$ at time T. Finally, by using the decoder $\mathcal{D}_{\theta_d}$, the latter is transformed into $\widehat{\mathbf{y}}(t=T;\mu) \simeq \mathbf{y}(t=T;\mu)$. Note parameter μ has to be supplied to the Hamiltonian function.

In order to determine the appropriate parameters of the three neural networks $\mathcal{D}_{\theta_d}$, $\mathcal{E}_{\theta_e}$ and $\overline{\mathcal{H}}_{\theta_h}$, we have to define both their architectures and the loss functions used for their training. This section focuses on the latter.

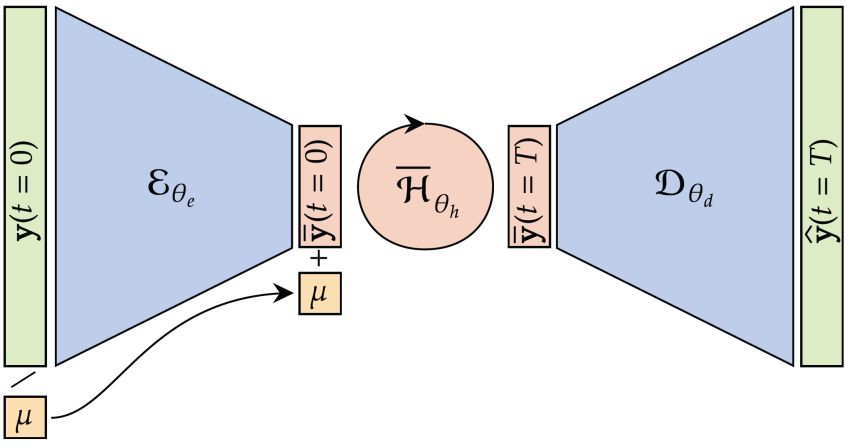


Figure 1: Prediction using the reduced model. The closed loop in the middle refers to the application of several iterations of the Störmer-Verlet scheme.

3.1 Reduction with an Auto Encoder (AE)

An auto-encoder (AE) is a classical architecture of neural networks to find a reduced representation of data [8]. It is composed of two neural networks, $\mathcal{D}_{\theta_d}$ and $\mathcal{E}_{\theta_e}$, which are trained together such as to make the projection operator $\mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}$ the closest to the identity map on the training data set U. Therefore, the AE is trained so as to minimize the following loss

$$\mathcal{L}_{\mathrm{AE}}(\theta_e, \theta_d) = \sum_{\mathbf{y} \in U} \left\| \mathbf{y} - \mathcal{D}_{\theta_d}\left(\mathcal{E}_{\theta_e}\left(\mathbf{y}\right)\right) \right\|_2^2 . \tag{7}$$

To account for the particular structure of $\mathbf{y}$ made of coordinates and momenta, the encoder input is a tensor of size $(N, 2)$. This AE will be referred as the bichannel AE.

Another choice would be to define two separate auto-encoders for coordinates and momenta: the coordinates AE is denoted $(\mathcal{E}^1_{\theta_e,1}, \mathcal{D}^1_{\theta_d,1})$ and the momenta AE is denoted $(\mathcal{E}^2_{\theta_e,2}, \mathcal{D}^2_{\theta_d,2})$; each encoder input has shape $(N, 1)$. The AEs are trained by minimizing the loss:

$$\mathcal{L}_{\mathrm{split,AE}}(\theta_e, \theta_d) = \sum_{(\mathbf{q},\mathbf{p}) \in U} \left\| \mathbf{q} - \mathcal{D}^1_{\theta_d,1}\left(\mathcal{E}^1_{\theta_e,1}\left(\mathbf{q}\right)\right) \right\|_2^2 + \left\| \mathbf{p} - \mathcal{D}^2_{\theta_d,2}\left(\mathcal{E}^2_{\theta_e,2}\left(\mathbf{p}\right)\right) \right\|_2^2 .$$

These AEs will be called the split AE.

The architectures of the neural networks are chosen specifically to the Hamiltonian systems in consideration. In this work, we focus on systems resulting from the spatial discretization of wave-like equations: networks will be more efficient if they take into account the spatial structure of the data. Consequently, the encoder $\mathcal{E}_{\theta_e}$ is first composed of several convolution layers before ending with some dense layers, as depicted in Figure 2. As usual for AE networks, the decoder is constructed in a mirror way, i.e. starting with some dense layers and then ending with convolution layers in reversed size order.

More precisely, convolutional layers in are built as a succession of blocks. In the encoder, a block is the succession of a stride 1 convolution and a stride 2 convolution . It results in an output that has twice as many channels and half as many rows as the input. In the

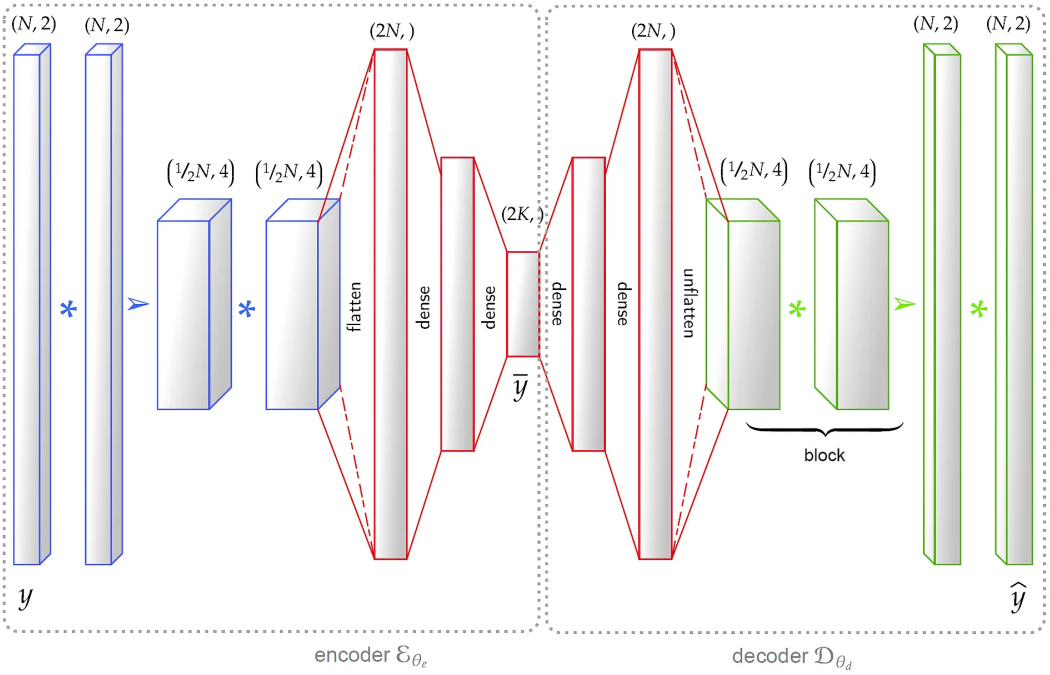


Figure 2: Auto-encoder architecture. Convolution block for the encoder (in blue): a stride 1 convolution $*$ and a stride 2 convolution. Convolution block for the decoder (in red): a stride 1 convolution $*$ and a up-sampling 2 convolution smoothed out with a kernel size 2 convolution.

decoder, a block is the succession of a convolution and an up-sampling of size 2 smoothed out with a convolution with kernel size 2, which symmetrically results in output that has half as many channels and twice as many rows as the input. Convolution layers are connected to the dense layers of the neural networks using flattening and unflattening operations. The architecture of the auto-encoder is thus defined with the number of blocks and the dense layer sizes for both encoder and decoder. Figure 2 illustrates an example of autoencoder architecture with one block for the encoder and one block for the decoder.

3.2 Reduced model with a Hamiltonian Neural Network (HNN)

The AE constructed in the previous section enables us to define the reduced trajectories:

$$\overline{\mathbf{y}}(t;\mu) = \mathcal{E}_{\theta_e}(\mathbf{y}(t;\mu)). \tag{8}$$

To obtain the dynamics of these reduced variables, we propose to use a Hamiltonian Neural Network strategy [9]. We thus look for a neural network function $\overline{\mathcal{H}}_{\theta_h}$, parameterized by θ_h, such that:

$$\frac{d}{dt}\overline{\mathbf{y}}(t;\mu) = J_{2K}\nabla_{\overline{\mathbf{y}}}\overline{\mathcal{H}}_{\theta_h}(\overline{\mathbf{y}}(t;\mu);\mu).$$

Note that the Hamiltonian is supposed to depend on parameter μ. We remind that $\mu \in \Xi$ stands for known parameters of the model, unlike θ_h that is the neural networks parameters

to be learnt. The architecture of the reduced Hamiltonian is a classical MLP neural network. The size of the neural network is chosen to be small so that the reduced model remains competitive. This reduced dynamics is in practice defined through a time discretization. We therefore introduce the prediction operator:

$$\mathcal{P}_s\left(\overline{\mathbf{y}}; \overline{\mathcal{H}}_{\theta_h,\mu}\right),$$

which consists in performing $s \in \mathbb{N}^*$ iterations of the Störmer-Verlet scheme, defined in (3), starting from $\overline{\mathbf{y}}$ and where $\overline{\mathcal{H}}_{\theta_h,\mu}$ stands for the Hamiltonian function $\overline{\mathcal{H}}_{\theta_h}(.;\mu)$. The number of steps s considered in this prediction is called the watch duration. This is a hyper-parameter of the method that has to be set. The parameters of the reduced Hamiltonian are finally obtained by minimizing the following loss:

$$\mathcal{L}^s_{\overline{\text{pred}}}(\theta_e,\theta_h) = \sum_{\mathbf{y}^n_\mu,\mathbf{y}^{n+s}_\mu \in U} \left\|\overline{\mathbf{y}}^{n+s}_\mu - \mathcal{P}_s\left(\overline{\mathbf{y}}^n_\mu; \overline{\mathcal{H}}_{\theta_h,\mu}\right)\right\|_2^2 \tag{9}$$

where $\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu \in U$ denote the sampling of random pairs $(\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu)$ on the data set U. In other words, random time series of size s are sampled and only the data at both ends are considered. This loss thus compares the reduced trajectories (8) with the ones obtained with the reduced Hamiltonian. The name of the loss function, "p$\overline{\text{re}}$d", refers to the prediction in the reduced variables. Note that the encoder neural network is required to obtain the reduced data $\overline{\mathbf{y}} = \mathcal{E}_{\theta_e}(\mathbf{y})$ and this is why the loss depends also on θ_e. This kind of loss function, based on a model, have been widely used in physics based deep learning methods [24].

Then, we constrain the reduced trajectories to preserve the reduced Hamiltonian with the following loss function :

$$\mathcal{L}^s_{\overline{\text{stab}}}(\theta_e,\theta_h) = \sum_{\mathbf{y}^n_\mu,\mathbf{y}^{n+s}_\mu \in U} \left\|\overline{\mathcal{H}}_{\theta_h,\mu}\left(\overline{\mathbf{y}}^{n+s}_\mu\right) - \overline{\mathcal{H}}_{\theta_h,\mu}\left(\overline{\mathbf{y}}^n_\mu\right)\right\|_2^2, \tag{10}$$

where $\overline{\mathbf{y}}^{n+s}_\mu$ and $\overline{\mathbf{y}}^n_\mu$ are still obtained using the encoder $\mathcal{E}_{\theta_e}$. The aim is to ensure some stability of the reduced model, hence its name "st$\overline{\text{ab}}$" . At first sight, this loss seems redundant with the prediction-reduced loss since using Störmer-Verlet schemes in the prediction step ensures that the reduced Hamiltonian is preserved at least approximately. Hence if the prediction-reduced loss becomes small, so does the stability loss. However, this additional loss may help the coupling to converge.

3.3 Strong coupling of the neural networks

The prediction-reduced and the stability-reduced losses (9) already introduce a coupling between the encoder neural networks and the reduced Hamiltonian one. To make the coupling stronger, we could ask for the trajectories in the initial variables to be well recovered. This is why we introduce the following fourth loss function named "pred" which refers to the model prediction in the FOM space:

$$\mathcal{L}^s_{\text{pred}}(\theta_e,\theta_d,\theta_h) = \sum_{\mathbf{y}^n_\mu,\mathbf{y}^{n+s}_\mu \in U} \left\|\mathbf{y}^{n+s}_\mu - \mathcal{D}_{\theta_d}\left(\mathcal{P}_s\left(\overline{\mathbf{y}}^n_\mu; \overline{\mathcal{H}}_{\theta_h,\mu}\right)\right)\right\|_2^2. \tag{11}$$

This is the only loss function that couples the three neural networks. It compares the trajectories in the initial variables with the full process of encoding, predicting in reduced variables over s iterations and then decoding.

To sum up, we use four different loss functions $\mathcal{L}_{\mathrm{AE}}, \mathcal{L}^s_{\overline{\mathrm{pred}}}, \mathcal{L}_{\overline{\mathrm{stab}}}$ and $\mathcal{L}^s_{\mathrm{pred}}$, given by (7)-(9)-(10)-(11) that couple both AE and HNN neural networks. The training aims to find the parameters $(\theta_e, \theta_d, \theta_h)$ that are solution to the minimization problem:

$$\min_{\theta_e,\theta_d,\theta_h} \quad \omega_{\mathrm{AE}}\,\mathcal{L}_{\mathrm{AE}}(\theta_e,\theta_d) + \omega_{\overline{\mathrm{pred}}}\,\mathcal{L}^s_{\overline{\mathrm{pred}}}(\theta_e,\theta_h) + \omega_{\overline{\mathrm{stab}}}\,\mathcal{L}^s_{\overline{\mathrm{stab}}}(\theta_e,\theta_h) + \omega_{\mathrm{pred}}\,\mathcal{L}^s_{\mathrm{pred}}(\theta_e,\theta_d,\theta_h),$$

where ω_{AE}, $\omega_{\overline{\mathrm{pred}}}$, $\omega_{\overline{\mathrm{stab}}}$, ω_{pred} are positive weights: these are hyper-parameters of the method. The four loss functions interact during training and possibly compete with each other. Note that in the end, the only loss value that really quantifies the quality of the reduction and prediction process is the one corresponding to $\mathcal{L}^s_{\mathrm{pred}}$. The other loss functions are only useful in the training process.

3.4 Training hyper-parameters

In addition to the parameters of the neural networks (number of layers, size of the layers), the training of the model depends also on several hyper-parameters.

Watch duration in predictions. One of the hyper-parameter to set is the watch duration s in the loss functions $\mathcal{L}^s_{\overline{\mathrm{stab}}}$, $\mathcal{L}^s_{\overline{\mathrm{pred}}}$ and $\mathcal{L}^s_{\mathrm{pred}}$ that make predictions. This quantity should be not too small to capture the dynamics but also not too large as the computation of the gradients of the associated loss functions may generate vanishing gradient problems. In our numerical setting, the watch duration will be typically set to

$$s = 16.$$

Loss functions weights. Losses weights have been chosen experimentally as follows:

$$\omega_{\mathrm{pred}} = 0.1, \quad \omega_{\mathrm{AE}} = 0.1, \quad \omega_{\overline{\mathrm{pred}}} = 80, \quad \omega_{\overline{\mathrm{stab}}} = 7 \times 10^{-4}.$$

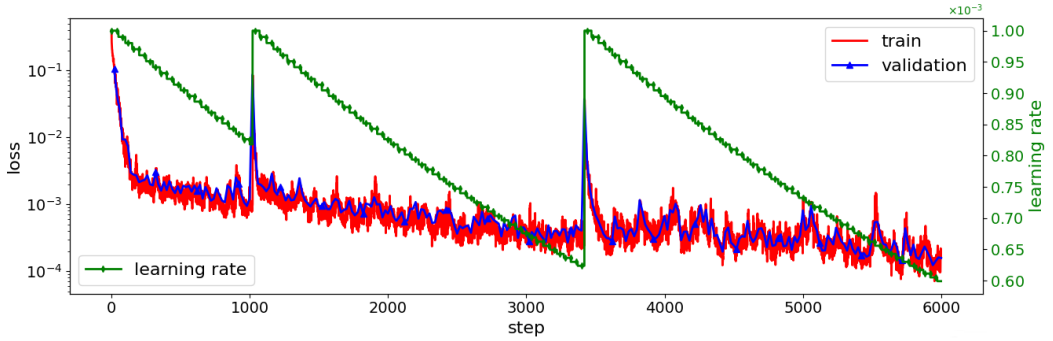
A typical history loss is shown in Figure 3b. The choice of the weights makes prediction loss function $\mathcal{L}_{\mathrm{pred}}$ and AE loss function $\mathcal{L}_{\mathrm{AE}}$ predominant, while the reduced prediction loss $\mathcal{L}_{\overline{\mathrm{pred}}}$ is 10 times smaller. The reduced stability loss function $\mathcal{L}_{\overline{\mathrm{stab}}}$ acts only as a quality control and remains small.

Gradient descent. An Adam optimizer [14] is used for the training. The learning rate follows the following rule:

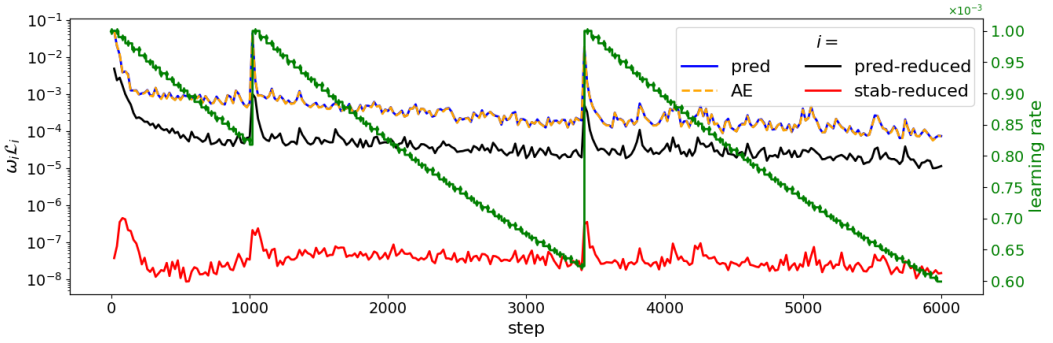
$$\rho_k = (0.99)^{k/150}\, 0.001.$$

where the division operator stands here for the integer division, and k is the train step. Thus, the learning rate is constant over 150 iterations, then decreases. It has an exponential decay with the shape of a staircase. In addition, we can reset this decay i.e. set $k = 0$ at any time if we notice that the loss is reaching a plateau. The main goal of this reset strategy is to escape poor local minima of the minimization problem with a sudden large learning rate. On Figure 3a is shown a typical training and validation loss history as functions of the training step k as well as the learning rate at each step. The resets enable to make the training loss go from 1×10^{-3} to 5×10^{-4} and then to 1×10^{-4}. With the loss functions weights above-mentioned, we consider 1×10^{-5} to be a correct plateau value to stop the training process.

Pre-processing Data pre-processing are required to optimize the learning process. Here we use usual standardization techniques.



(a) Training loss function (blue) and validation loss function (red) as functions of the training step.



(b) All the loss functions as functions of the training step.

Figure 3: Example of loss functions history during a training, overlayed with the evolution of the learning rate (green).

4 Numerical results

This section is devoted to the numerical results obtained with the proposed Hamiltonian reduction method. We consider one-dimensional discretizations of three wave-type equations: the linear wave equation, the non-linear wave equation and the shallow water equation.

4.1 Wave equations

We introduce a parameterized one-dimensional wave equation:

$$\begin{cases} \partial_{tt} u(t,x) - \mu_a \, \partial_x \Big(w'\big(\partial_x u(t,x), \mu_b\big)\Big) + g'(u(t,x), \mu_c) = 0, \quad (t,x) \in (0,T] \times [0,1], \\ u(0,x) = u_{\text{init}}(x), \quad x \in [0,1], \\ \partial_t u(0,x) = v_{\text{init}}(x), \quad x \in [0,1], \end{cases} \tag{12}$$

complemented with periodic boundary conditions. The solution $u(t,x)$ represents the vertical displacement of a string over the interval $[0,1]$. The model depends on two given functions $w, g : \mathbb{R} \to \mathbb{R}$ and three parameters: $\mu = (\mu_a, \mu_b, \mu_c)^T \in \Xi \subset \mathbb{R}^3_+$.

Defining the vertical displacement velocity $v(x,t) = \partial_t u(t,x)$, Equation (12) can be

reformulated as a first order in time system:

$$\begin{cases} \partial_t u(x,t) - v(x,t) = 0, & (t,x) \in (0,T] \times [0,1], \\ \partial_t v(x,t) + \mu_a \partial_x \Big(w'\big(\partial_x u(t,x), \mu_b\big)\Big) + g'(u(x,t), \mu_c) = 0, & (t,x) \in (0,T] \times [0,1] \\ u(0,x) = u_{\text{init}}(x), & x \in [0,1], \\ v(0,x) = v_{\text{init}}(x), & x \in [0,1]. \end{cases}$$

Then, we consider a spatial finite difference discretization of this system. Introducing a uniform mesh of the interval $[0,1]$, with N cells, space step $\Delta x = 1/(N-1)$ and nodes $x_i = i\Delta x$, for $i \in [\![0, N-1]\!]$, the approximate solution $(\mathbf{u}, \mathbf{v}) = (u_i(t), v_i(t))_{i \in [\![0,N-1]\!]} \in \mathbb{R}^N \times \mathbb{R}^N$ satisfies the following system:

$$\begin{cases} \dfrac{d}{dt} u_i(t) = v_i(t), & t \in (0,T] \\ \dfrac{d}{dt} v_i(t) = -\mu_a \dfrac{1}{\Delta x} \Big(w'\big(\dfrac{u_{i+1}(t) - u_i(t)}{\Delta x}, \mu_b\big) - w'\big(\dfrac{u_i(t) - u_{i-1}(t)}{\Delta x}, \mu_b\big)\Big) + g'(u_i(t), \mu_c) \\ u_i(0) = u_{\text{init}}(x_i), \\ v_i(0) = v_{\text{init}}(x_i). \end{cases} \tag{13}$$

These equations actually form a Hamiltonian system of size $2N$, with a separable Hamiltonian function ($(\mathbf{u}, \mathbf{v})$ stands for variables $(\mathbf{q}, \mathbf{p})$) given by

$$\mathcal{H}(\mathbf{u}, \mathbf{v}; \mu) = \mathcal{H}^1(\mathbf{u}; \mu) + \mathcal{H}^2(\mathbf{v}; \mu), \tag{14}$$

with

$$\mathcal{H}^1(\mathbf{u}; \mu) = \Delta x \sum_{i=0}^{N-1} \left(\mu_a w \left(\frac{u_{i+1} - u_i}{\Delta x}, \mu_b \right) + \mu_a w \left(\frac{u_i - u_{i-1}}{\Delta x}, \mu_b \right) + g(u_i, \mu_c) \right),$$

$$\mathcal{H}^2(\mathbf{v}; \mu) = \frac{1}{2} \Delta x \sum_{i=0}^{N-1} v_i^2.$$

In the following, we test the AE+HNN method for two choices of functions w and g. The same hyper-parameters are used in both test-cases. They are summarized in Table 1.

4.1.1 Linear wave equation

Here we consider the linear wave equations:

$$\begin{cases} \partial_{tt} u(t,x) - \mu_a \, \partial_{xx} u(t,x) = 0, & (t,x) \in (0,T] \times [0,1], \\ u(0,x) = u_{\text{init}}(x), & x \in [0,1]. \end{cases}$$

corresponding to $w(x, \mu_b) = \frac{1}{2} x^2/2$ and $g(x, \mu_c) = 0$. In particular, we conserve only one parameter $\mu_a \in [0.2, 0.6]$, which corresponds to the square of the wave velocity. The initial condition is taken equal to:

$$u_{\text{init}}(x) = h(10|x - \tfrac{1}{2}|), \quad v_{\text{init}}(x) = 0. \tag{15}$$

where h is the compactly supported (single bump) function:

$$h(r) = \big(1 - \tfrac{3}{2} r^2 + \tfrac{3}{4} r^3\big) \, 1_{[0,1]}(r) + \tfrac{1}{4} (2 - r)^3 1_{(1,2]}(r). \tag{16}$$

		Wave equations	Shallow water system
AE	type	bichannel	split
	nb of convolution blocks (encoder)	4	4
	dense layers (encoder)	$[256, 128, 64, 32]$	$[256, 128, 64, 32]$
	activation functions	ELU	swich
HNN	dense layers	$[24, 12, 12, 12, 6]$	$[24, 12, 12, 12, 6]$
	activation functions	tanh	swich
watch duration	s	16	48

Table 1: Hyper-parameters. Activation functions are used except for the last layer of the neural networks. ELU refers to the the function $\text{elu}(x) = x1_{x>0} + (e^x - 1)1_{x<0}$ and swish to the function $\text{swish}(x) = x/(1+e^{-x})$. For the auto-encoder (AE), the number of convolution blocks and the sizes of the hidden of layers are those of the encoder. The decoder is constructed in a mirror way.

The initial condition can be observed in Figure 5a. We consider $N = 1024$ discretization points, set $T = 0.4$ and $\Delta t = 1 \times 10^{-4}$. In Figure 4, we observe the solution at final time $T = 0.4$ as a function of μ_a. Each color corresponds to a different value of μ_a. As we can expect, large values of μ_a generate waves farther from the initial condition.

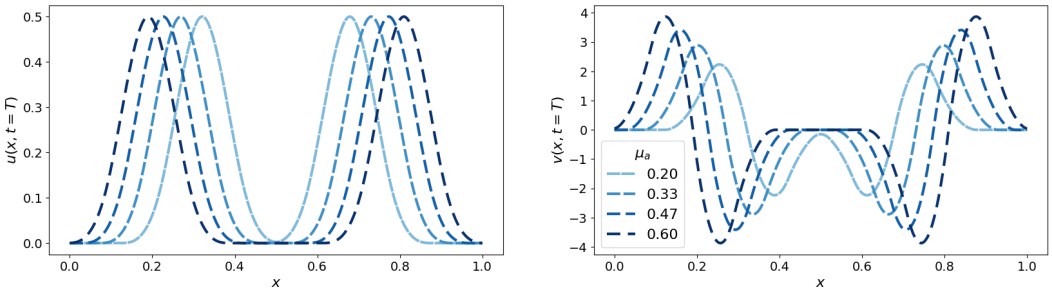


Figure 4: (Linear wave) Solution $(\mathbf{u}, \mathbf{v})$ at final time $T = 0.4$ for various parameters $\mu = \mu_a$

To train the model, we consider numerical solutions obtained with $P = 20$ different values μ_a taken regularly spaced in the interval $[0.2, 0.6]$. The validation set is obtained considering numerical solutions with 6 different parameter values in the same interval and 128 random pairs $(\mathbf{y}^n, \mathbf{y}^{n+s})$ for each validation parameter. The final value of the validation loss funcitons are given in Table 2.

Then we test the obtained model on 3 test values:

$$\mu_a = 0.2385 \text{ (test 1)}, \quad \mu_a = 0.3798 \text{ (test 2)}, \quad \mu_a = 0.5428 \text{ (test 3)}.$$

To evaluate our model, we compute the relative L^2 discrete error between the reference

solution u_{ref} and the reduced model prediction u_{pred}:

$$\text{err} = \frac{\sum_{n=1}^{M} \Delta t \left(\sum_{i=1}^{N} \Delta x (u^n_{\text{ref},i} - u^n_{\text{pred},i})^2 \right)}{\sum_{n=1}^{M} \Delta t \left(\sum_{i=1}^{N} \Delta x (u^n_{\text{pred},i})^2 \right)},$$

where M is the total number of time steps. Table 3 compares the errors of the AE+HNN method with the ones obtained with the PSD and the POD for different reduced dimension K. We have colored in blue the cells of the PSD and POD methods with an error lower than the AE+HNN for $K = 1$ (in yellow). As shown in Table 3, we succeed in capturing the dynamics with a AE+HNN model of dimension $K = 1$ with a relative error equal to 1×10^{-2}. To obtain comparable performance, $K = 6$ modes are needed for the PSD and $K = 10$ modes for the POD. The reduction capability of the PSD is nevertheless quite good and this is due to the linearity of the problem. Also, the POD results show that taking into account the symplecticity of the problem enables us to improve significantly the reduction.

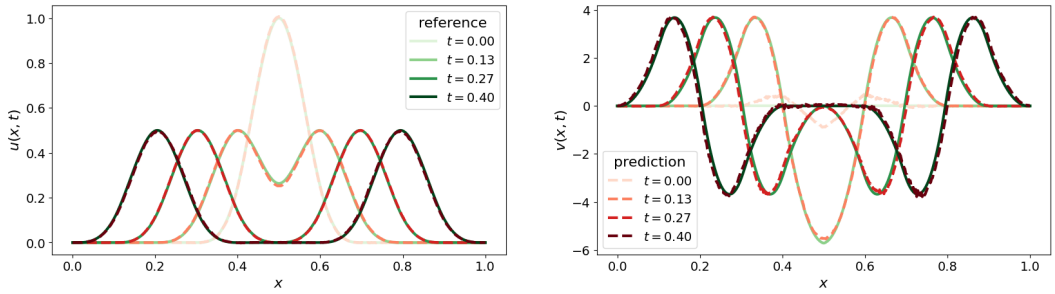
Figure 5 shows the simulations of the reduced models for both the AE+HNN ($K = 1$) and the PSD ($K = 6$) methods at different times for test 3. As expected, both methods provide good results. On Figure 6 is shown the time evolution of the reduced Hamiltonian $\overline{\mathcal{H}}_{\theta_h}$ for both the reference solution after encoding, in black, and for the reduced dynamics, in red. While the reduced solution preserves the reduced Hamiltonian by construction, the reduced Hamiltonian on the encoded reference solution is not preserved and strongly deviates from its initial value for time larger than 0.2. This is because the encoder is not exactly symplectic. Larger reduced dimension may help increasing the precision on this conservation.

	$\mathcal{L}^s_{\text{pred}}$	$\mathcal{L}_{\text{AE}}$	$\mathcal{L}^s_{\overline{\text{pred}}}$	$\mathcal{L}^s_{\overline{\text{stab}}}$
Linear wave	8.25×10^{-4}	8.19×10^{-4}	2.52×10^{-7}	1.54×10^{-5}
Non-linear wave	3.47×10^{-5}	3.53×10^{-5}	1.84×10^{-8}	1.98×10^{-6}
Shallow water	6.49×10^{-5}	6.51×10^{-5}	7.70×10^{-9}	1.19×10^{-7}

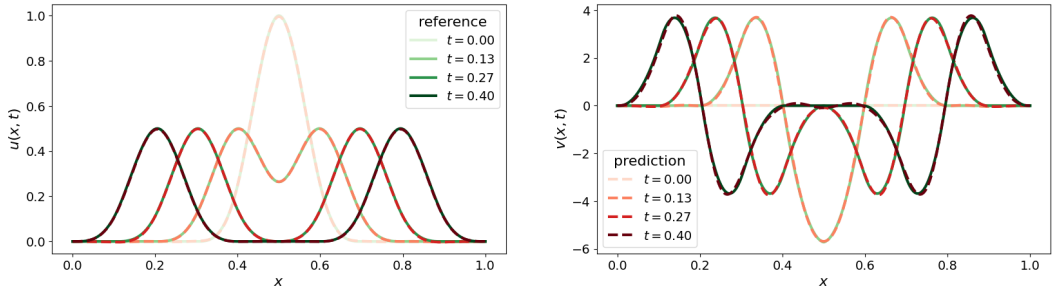
Table 2: Values of the loss functions on validation data for the different test-cases.

		test 1		test 2		test 3	
		error u	error v	error u	error v	error u	error v
AE+HNN	$K = 1$	1.20×10^{-2}	6.05×10^{-2}	7.07×10^{-3}	5.84×10^{-2}	1.43×10^{-2}	8.09×10^{-2}
	$K = 2$	6.71×10^{-3}	2.24×10^{-2}	1.48×10^{-2}	3.99×10^{-2}	9.71×10^{-3}	2.30×10^{-2}
	$K = 5$	1.67×10^{-2}	2.54×10^{-2}	1.48×10^{-2}	2.90×10^{-2}	1.79×10^{-2}	3.55×10^{-2}
PSD	$K = 4$	6.70×10^{-1}	3.67×10^{-1}	7.01×10^{-1}	8.69×10^{-1}	7.30×10^{-1}	3.65×10^{-1}
	$K = 5$	1.28×10^{-1}	1.34×10^{-1}	1.60×10^{-1}	1.43×10^{-1}	1.91×10^{-1}	1.52×10^{-1}
	$K = 6$	4.80×10^{-3}	2.11×10^{-2}	5.52×10^{-2}	2.14×10^{-2}	5.89×10^{-3}	2.23×10^{-2}
POD	$K = 6$	3.07×10^{-2}	1.08×10^{-1}	1.30×10^{-2}	2.67×10^{-2}	6.00×10^{-2}	1.54×10^{-1}
	$K = 8$	1.28×10^{-2}	4.89×10^{-2}	1.12×10^{-2}	2.66×10^{-2}	3.96×10^{-2}	7.05×10^{-2}
	$K = 10$	9.78×10^{-3}	4.36×10^{-2}	2.30×10^{-3}	9.07×10^{-3}	1.58×10^{-2}	5.93×10^{-2}

Table 3: (Linear wave) Relative L^2 errors for different reduced dimensions K. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE+HNN simulation in yellow.



(a) AE+HNN, $K = 1$



(b) PSD, $K = 6$

Figure 5: (Linear wave) Solution $(\mathbf{u}, \mathbf{v})$ at different times on test 3, reference solution (full lines) and prediction (dashed lines).

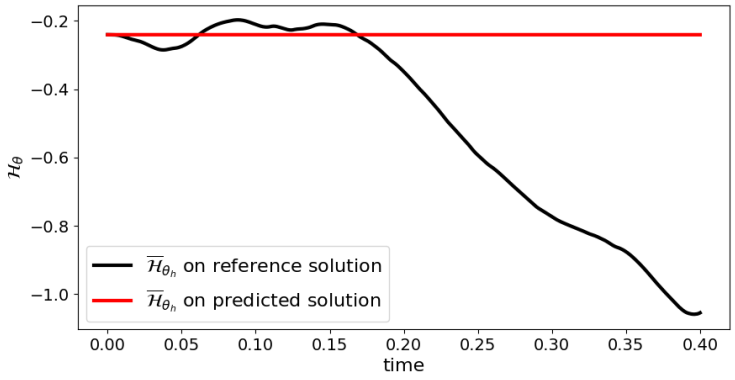


Figure 6: (Linear wave) Time evolution of $\overline{\mathcal{H}}_{\theta_h}$ as a function of time on test 3, on encoded reference solution (black) and predicted solution with the AE+HNN and $K = 1$ (red).

4.1.2 Non-linear wave equation

Now, we consider the following non-linear wave equations:

$$\begin{cases} \partial_{tt} u(t,x) - \mu_a \, \partial_{xx} u(t,x) - \mu_b \, \partial_x \left(\cos(\mu_b \partial_x u(t,x)) \right) + 30 \mu_c x^2 = 0, \quad (t,x) \in (0,T] \times [0,1], \\ u(0,x) = u_{\text{init}}(x), \; x \in [0,1]. \end{cases}$$

corresponding to $w(x,\mu_b) = x^2/2 + \sin(\mu_b x)$ and $g(x,\mu_c) = 10\mu_c x^3$. There are three parameters $\mu_a \in [0.2, 0.6], \mu_b \in [0.025, 0.5]$ and $\mu_c \in [0.4, 2.4]$. The initial condition is given by Eq. (15)-(16).

The number of discretization points is still $N = 1024$ but the final time is taken equal to $T = 0.3$ and the time step $\Delta t = 1 \times 10^{-4}$. These parameters have been chosen so that the numerical simulations remain stable despite the strong non-linearities. In Figure 7, we observe the numerical solutions at final time for several sets of parameters.

Training data are obtained using $P = 20$ different values (μ_a, μ_b, μ_c) regularly spaced in the segment $[(0.2, 0.025, 0.4), (0.6, 0.5, 2.4)]$. The validation set is also made of 6 different triplets define on the same domain. At the end of the training, the validation loss takes the values given in Table 2.

The obtained model is tested on 3 sets of parameters:

$$\begin{aligned} (\mu_a, \mu_b, \mu_c) &= (0.2385, 0.088, 0.5485) \qquad \text{(test 1)}, \\ (\mu_a, \mu_b, \mu_c) &= (0.3785, 0.281, 1.354) \qquad \text{(test 2)}, \\ (\mu_a, \mu_b, \mu_c) &= (0.5528, 0.437, 2.128) \qquad \text{(test 3)}, \end{aligned}$$

and Table 4 presents the relative errors. A relative error of order 1×10^{-2} can be reached with the AE+HNN method with a reduced dimension of $K = 3$ only. In comparison, the PSD and the POD requires $K = 15$ resp. $K = 30$ reduced dimensions to obtain similar results.

Figure 8 shows numerical solutions obtained for parameters corresponding to test 3. While the AE+HNN method with $K = 3$ remains close to the reference solution (Fig. 8a), the PSD with the same reduced dimension does not provide satisfactory results (Fig. 8b). Increasing the reduced dimension to $K = 15$ for the PSD allows us to recover comparable results (Fig. 8c). Figure 9 shows the time evolution of the reduced Hamiltonian. With a reduced dimension $K = 3$, we observe that the encoded reference solution does not exactly conserve the reduced Hamiltonian value but the relative error is less than 0.05.

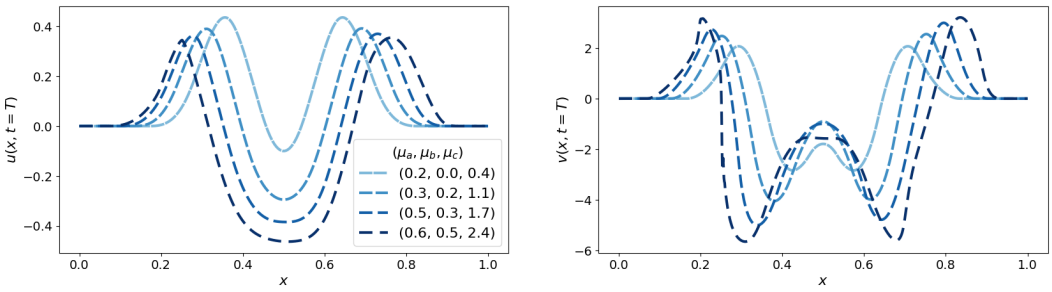
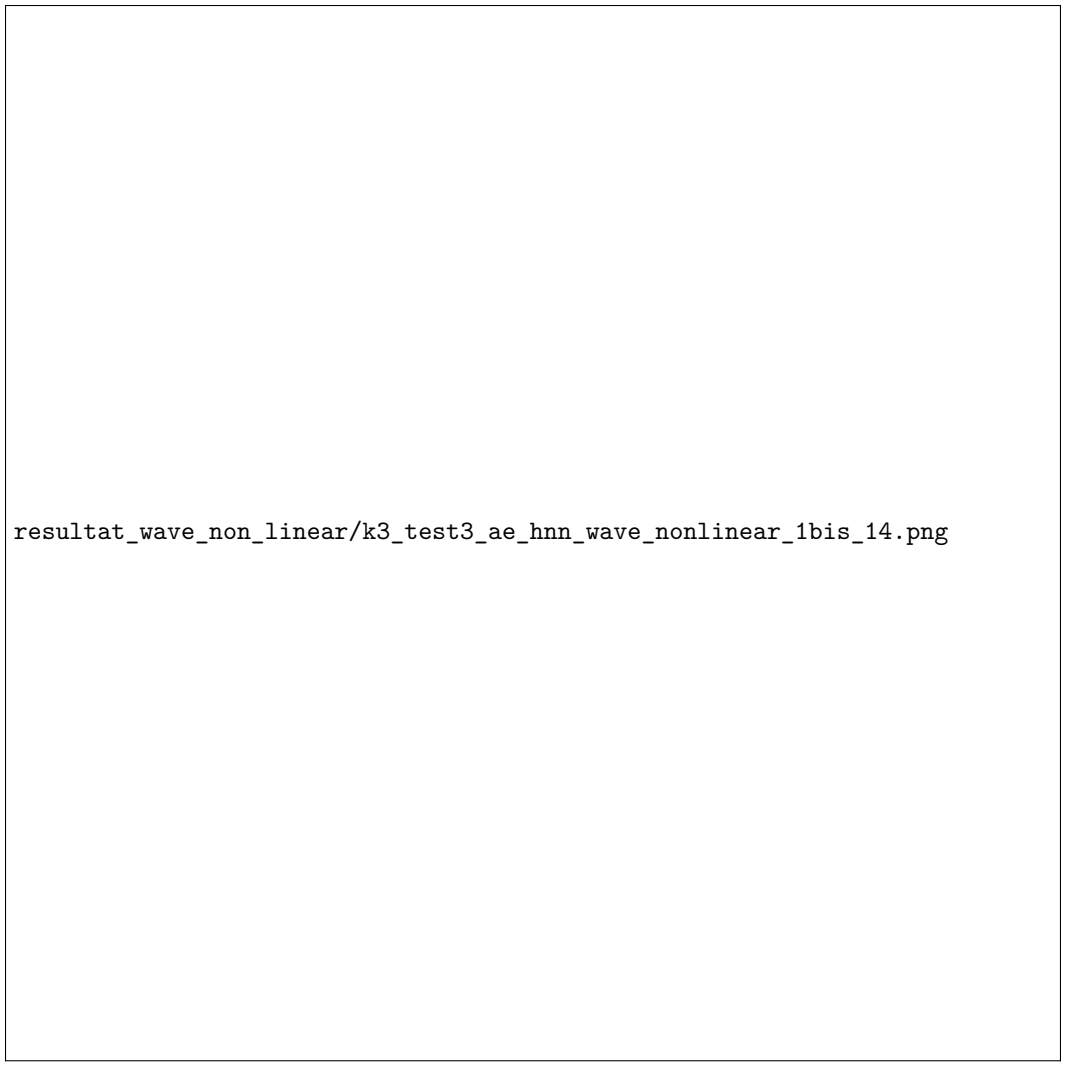


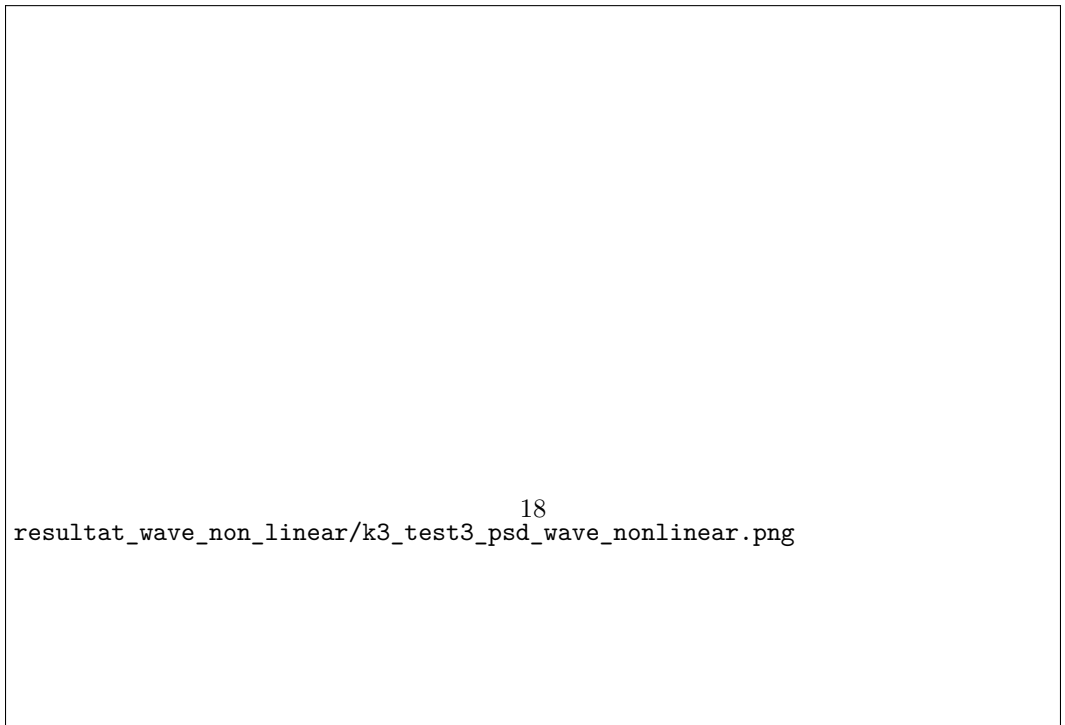
Figure 7: (Non-linear wave) final condition $(u, v)(x, t = T; \mu)$ for various parameters $\mu = (\mu_a, \mu_b, \mu_c)$

		test 1		test 2		test 3	
		error u	error v	error u	error v	error u	error v
AE+HNN	$K=3$	4.34×10^{-3}	1.17×10^{-2}	5.82×10^{-2}	8.65×10^{-2}	1.06×10^{-2}	1.36×10^{-2}
PSD	$K=3$	4.04×10^{-1}	2.55×10^{-1}	4.21×10^{-1}	2.49×10^{-1}	4.33×10^{-1}	2.73×10^{-1}
	$K=10$	3.26×10^{-2}	4.92×10^{-2}	3.84×10^{-2}	4.53×10^{-2}	5.63×10^{-2}	4.94×10^{-2}
	$K=15$	8.48×10^{-3}	1.66×10^{-2}	6.45×10^{-3}	1.66×10^{-2}	5.29×10^{-3}	1.87×10^{-2}
POD	$K=3$	1.86×10^{-1}	3.16×10^{-1}	5.72×10^{-2}	6.66×10^{-2}	2.44×10^{-1}	2.89×10^{-1}
	$K=20$	2.08×10^{-2}	4.00×10^{-2}	3.10×10^{-2}	7.00×10^{-2}	3.66×10^{-3}	1.01×10^{-2}
	$K=30$	5.36×10^{-3}	2.09×10^{-2}	9.54×10^{-3}	2.26×10^{-2}	8.87×10^{-3}	1.93×10^{-2}

Table 4: (Non-linear wave) Relative L^2 errors for different reduced dimensions K and different sets parameters. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE+HNN simulation in yellow.


resultat_wave_non_linear/k3_test3_ae_hnn_wave_nonlinear_1bis_14.png

(a) AE+HNN, $K = 3$


resultat_wave_non_linear/k3_test3_psd_wave_nonlinear.png

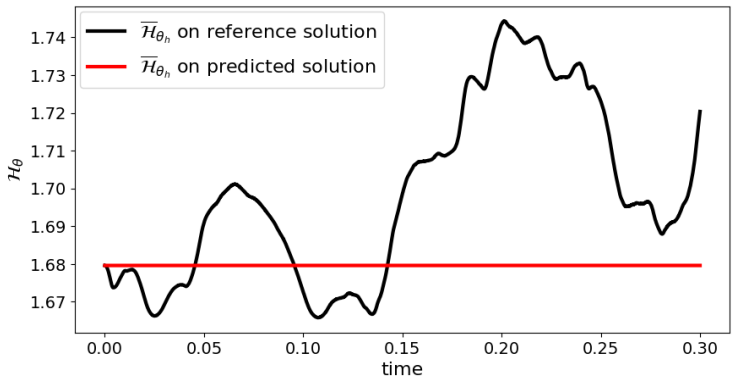


Figure 9: (Non-linear wave) Reduced Hamiltonian as a function of time on test 3 for an encoded reference solution (black) and a predicted solution with the AE+HNN with $K = 3$ (red)

4.1.3 Comparison with a non Hamiltonian reduction

In this section, we want to highlight the importance of the Hamiltonian framework for the reduced dynamics. Indeed, if the Hamiltonian structure is ignored, then the reduced system can be written:

$$\begin{cases} \dfrac{d}{dt}\overline{\mathbf{y}}(t;\mu) = \overline{\mathcal{F}}(\overline{\mathbf{y}}(t;\mu);\mu), \quad \forall t \in (0,T], \\ \overline{\mathbf{y}}(0;\mu) = \mathcal{E}(\mathbf{y}^0(\mu)), \end{cases} \tag{17}$$

where $\overline{\mathcal{F}} : \mathbb{R}^K \times \Xi \to \mathbb{R}^K$ is the reduced vector field. Then the reduced dynamics can be learnt by designing a neural network approximation $\overline{\mathcal{F}}_{\theta_f}$, with a classical MLP architecture. To this aim, we can consider the following loss function:

$$\mathcal{L}^s_{\text{Flow}}(\theta_e, \theta_f) = \sum_{\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu \in U} \left\| \overline{\mathbf{y}}^{n+s}_\mu - \mathcal{P}_s\left(\overline{\mathbf{y}}^n_\mu; \overline{\mathcal{F}}_{\theta_f}\right) \right\|_2^2, \tag{18}$$

where the prediction operator is here a simple RK2 scheme. We discard the stability loss function $\mathcal{L}^s_{\text{stab}}$ and keep all the other hyper-parameters identical to those of the previous section. Coupled with the auto-encoder, this reduction method will be referred to as AE-Flow.

We compare the two resulting method on the non-linear wave test-case considered in Section 4.1.2. The HNN parameters are unchanged. For a fair comparison, the flow neural network $\mathcal{F}_{\theta_f}$ has the same amount of parameters: the hidden layers have $[32, 24, 16, 16]$ units, which results in 1 886 parameters instead of 1 728 for the HNN. We train both models up to reach a loss value of about 1×10^{-5}. The obtained validation loss function are as follows

$$\mathcal{L}^s_{\text{pred}} = 9.46 \times 10^{-5}, \quad \mathcal{L}_{\text{AE}} = 9.88 \times 10^{-5}, \quad \mathcal{L}^s_{\text{Flow}} = 7.92 \times 10^{-7}.$$

They are of comparable magnitude to that of the AE+HNN method in Section 4.1.2.

Relative errors are provided in Table 5. The errors of the AE+HNN method are about 5 times smaller than those of the AE-Flow method. Figure 10 shows the time evolution of the relative error for test case 3 and Figure 11 depicts the solutions at different times.

We observe that the AE+HNN solution remains close to the reference while the AE-Flow solution drifts from it. Furthermore, we compare in Figure 12 the time evolution of the Hamiltonian for the reference solution, the AE+HNN solution, and the AE-Flow solution on test case 2. We observe that the AE+HNN method results in a better preservation of the Hamiltonian than the AE-Flow solution.

	test 1		test 2		test 3	
	error u	error v	error u	error v	error u	error v
AE+HNN	4.35×10^{-3}	1.18×10^{-2}	5.82×10^{-2}	8.66×10^{-2}	1.05×10^{-2}	1.36×10^{-2}
AE+Flow	5.11×10^{-2}	5.56×10^{-2}	1.27×10^{-1}	1.50×10^{-1}	9.99×10^{-2}	1.07×10^{-1}

Table 5: (Non linear wave) Relative L^2 errors for the AE-Flow and the AE+HNN

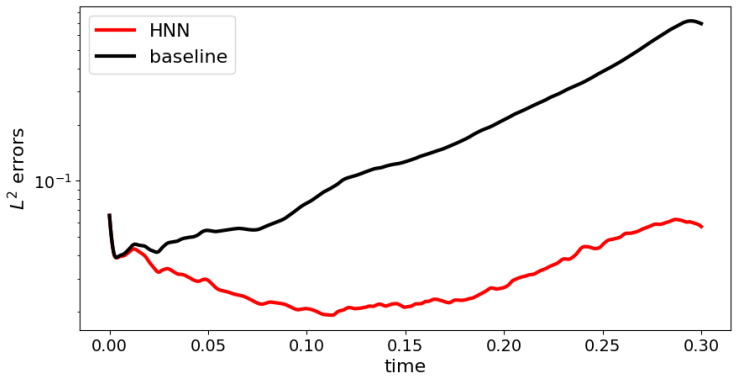


Figure 10: (Non linear wave) L^2 errors as a function of time for the AE-Flow (black) and the AE+HNN (red)

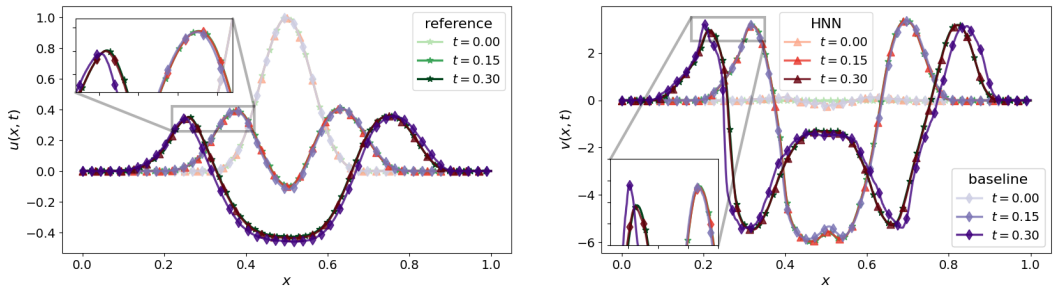


Figure 11: (Non linear wave) $(\overline{u},\overline{v})$ for different times on test 3, reference solution (green), AE+HNN solution (red) and AE-Flow solution (purple)

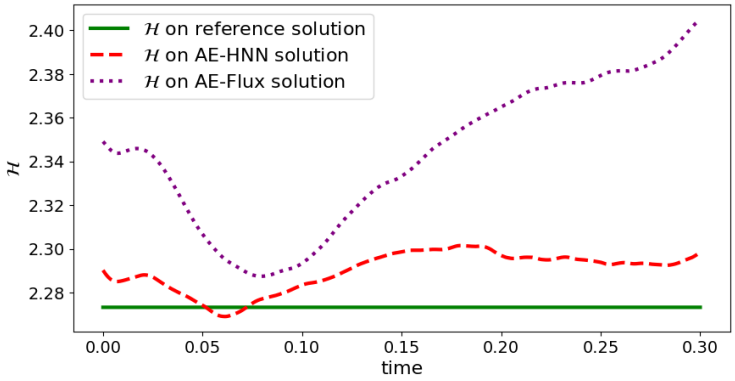


Figure 12: (Non linear wave) Time evolution of the Hamiltonian on test 2 for the reference solution (green), the AE+HNN solution (red) and the AE-Flow solution (purple).

4.1.4 Computation time gain

In this section, we compare the actual computation time on an Intel(R) Xeon(R) CPU. To obtain a fair comparison, we use the same implementation of the Störmer-Verlet algorithm for all the methods. The only difference between the PSD and the AE+HNN methods lies in the gradient computations. For the former, we use the explicit expression of the reduced Hamiltonian as defined in Appendix A and for the latter, we use the HNN backpropagation algorithm. We use 32 bits floating-point numbers and include the encoding or decoding of the data in the computation time. We consider the non-linear wave test case with the following parameters: $N = 1024, K = 3, T = 0.4$ and $\Delta t = 1 \times 10^{-4}$.

The reference solution without reduction is computed in $2\,200 \pm 14$ ms. In comparison, the AE+HNN method spends 452 ± 20 ms, which is five times faster. On the contrary, the PSD method spends $2\,382 \pm 12$ ms, which is is a bit slower than the reference solution. This results from the evaluation of the gradient of the Hamiltonian in the non-reduced dimension. Hyper-reduction technics like Hamiltonian DEIM would accelerate the computations [11]. However, this could also deteriorate the precision, and one should then consider larger reduced dimensions.

4.2 Shallow water system

We consider the one-dimensional Shallow Water system under the following formulation:

$$\begin{cases} \partial_t \chi(t,x) + \partial_x\big((1+\chi(t,x))\,\partial_x\phi(t,x)\big) = 0, & \forall (t,x) \in (0,T] \times [-1,1], \\ \partial_t \phi(t,x) + \frac{1}{2}(\partial_x \phi(t,x))^2 + \chi(t,x) = 0, & \\ \chi(x,0) = \chi_{\text{init}}(x), & \forall x \in [-1,1], \\ \phi(x,0) = \phi_{\text{init}}(x), & \end{cases}$$

where $\chi(x,t)$ denotes the perturbation from the equilibrium and $\phi(x,t)$ is the scalar velocity potential. Periodic boundary conditions are considered. This system admits a Hamiltonian function given by:

$$\mathcal{H}_{\text{cont}}(\chi,\phi) = \frac{1}{2}\int_{-1}^{1} \left((1+\chi)(\partial_x\phi)^2 + \chi^2\right) dx.$$

Like for the wave equation, we consider a spatial discretization of this model, still using a uniform mesh grid with N nodes $(x_i)_{i\in[\![0,N-1]\!]}$ on $[-1,1]$. The approximate solution is denoted $\boldsymbol{\chi} = (\chi_i)_{i\in[\![0,N-1]\!]}, \boldsymbol{\phi} = (\phi_i)_{i\in[\![0,N-1]\!]}$. We then introduce the discrete Hamiltonian function:

$$\mathcal{H}(\boldsymbol{\chi},\boldsymbol{\phi}) = \frac{\Delta x}{2}\sum_{i=1}^{N}(1+\chi_i)\left(\frac{\phi_{i+1}-\phi_{i-1}}{\Delta x}\right)^2 + \chi_i^2, \tag{19}$$

and the resulting discrete shallow water equation:

$$\begin{cases} \frac{d}{dt}\boldsymbol{\chi} = -D_x^2\boldsymbol{\phi} - D_x(\boldsymbol{\chi}\odot D_x\boldsymbol{\phi}), \\ \frac{d}{dt}\boldsymbol{\phi} = -\frac{1}{2}(D_x\boldsymbol{\phi})\odot(D_x\boldsymbol{\phi}) - \boldsymbol{\chi}, \\ \chi_i(0) = \chi_{\text{init}}(x_i), \\ \phi_i(0) = \phi_{\text{init}}(x_i), \end{cases} \tag{20}$$

where $\odot$ denotes the element-wise vector multiplication and D_x the centered second-order finite differences matrix with periodic boundary condition

$$D_x = \frac{1}{2\Delta x}\begin{pmatrix} 0 & 1 & & -1 \\ -1 & & \ddots & \\ & \ddots & & 1 \\ 1 & & -1 & 0 \end{pmatrix}.$$

We note that Hamiltonian (19) is not separable. As a consequence, the numerical resolution of (20) with the Störmer-Verlet scheme is implicit.Therefore, applying a reduction is all the more attractive.

The number of discretization points it taken equal to $N = 1024$, the final time equal to $T = 0.5$ and the time step $\Delta t = 2\times10^{-4}$. The initial condition is parameterized with 2 parameters $\mu = (c,\sigma) \in [0, 0.2]\times[0.2, 0.05]$

$$\chi_{\text{init}}(x) = \frac{0.02}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2\right), \quad \phi_{\text{init}}(x) = 0.$$

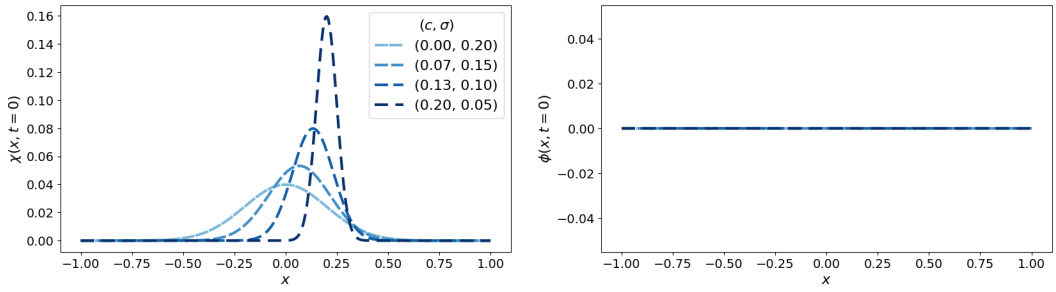
For the training data, we use 20 different solution parameters (c,σ): we take regularly spaced values in the segment $[(0, 0.2), (0.2, 0.05)]$. Let us observe the influence the parameters on the solutions at initial and final time on Figure 13. Non-linear patterns appear for small values of the standard deviation σ.

The set of hyper-parameters is almost identical to the previous test cases except for the activation functions and the watch duration, which here is taken equal to 48. They are gathered in Table 1. As in the previous sections, we inspect the validation loss functions and obtain value given in Table 2.

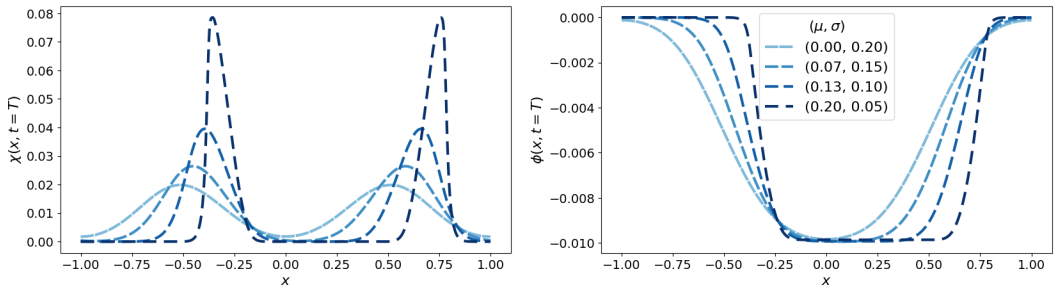
We choose three different sets of parameters to test the AE+HNN method:

$$(c,\sigma) = (0.105, 0.11), \qquad \text{(test 1)}$$
$$(c,\sigma) = (0.195, 0.053), \qquad \text{(test 2)}$$
$$(c,\sigma) = (0.21, 0.045). \qquad \text{(test 3)}$$

As in the previous test cases, we compute the relative errors of the AE+HNN method with respect to the reference solution (see Table 6) and compare them to the results obtained with the PSD and POD methods for different values of K.



(a) $t = 0$

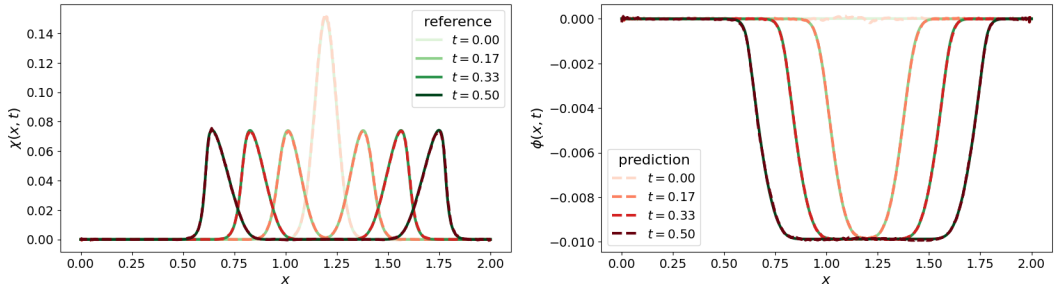


(b) $t = 0.5$

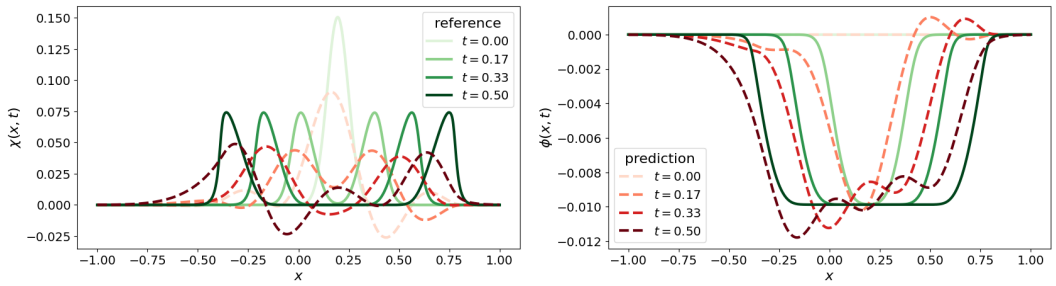
Figure 13: (Shallow Water) Solutions at initial time $t = 0$ and final time $t = 0.5$ for various parameters (μ, σ).

The AE+HNN method achieves a mean error of about $3{\times}10^{-2}$ with a reduced dimension of $K = 4$ only. To achieve a similar performance, the PSD requires a reduced dimension of $K = 24$ and the POD need a value larger than $K = 32$. Table 6 shows also that the AE+HNN method has a different behavior with respect to K than the POD and PSD methods. For the latter, increasing K improves accuracy (at the expense of computation time). With the AE+HNN method, increasing K improves the encoder-decoder performance but makes it more difficult to learn the reduced dynamics for the HNN neural network. Therefore, the HNN performance requires a balance between an adequate compression and a low-dimensional reduced model.

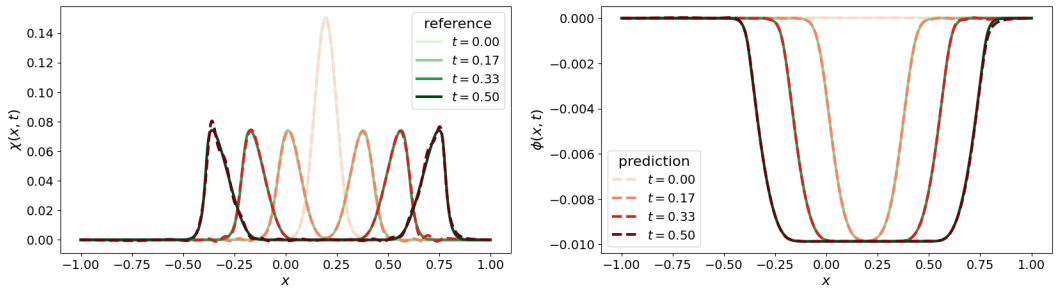
Then, we compare the solutions obtained with the AE+HNN method and the PSD in Figure 14. For a given reduced dimension $K = 4$, the AE+HNN method solution remains close to the reference as expected (Fig. 14a) while the PSD solution oscillates and stays far from the reference solution, even for the initial condition (Fig. 14b). When increasing the dimension of the reduced model to $K = 24$ for the PSD method (Fig. 14c), we recover similar results as the ones obtained with the AE+HNN.



(a) AE+HNN, $K = 4$



(b) PSD, $K = 4$



(c) PSD, $K = 24$

Figure 14: (Shallow Water) (χ, ϕ) as a at different times on test 2 and $K = 8$, reference solution (full lines) and prediction (dashed lines)

		test 1		test 2		test 3	
		error χ	error ϕ	error χ	error ϕ	error χ	error ϕ
AE+HNN	$K=4$	5.86×10^{-2}	2.89×10^{-2}	1.34×10^{-2}	5.00×10^{-3}	8.12×10^{-2}	2.27×10^{-2}
	$K=6$	8.46×10^{-2}	5.17×10^{-2}	2.07×10^{-2}	7.77×10^{-3}	9.70×10^{-2}	2.80×10^{-2}
	$K=8$	6.26×10^{-2}	3.54×10^{-2}	2.21×10^{-2}	1.20×10^{-2}	1.45×10^{-1}	4.99×10^{-2}
PSD	$K=10$	7.11×10^{-2}	1.71×10^{-2}	1.64×10^{-1}	2.74×10^{-2}	3.08×10^{-1}	5.89×10^{-2}
	$K=14$	2.67×10^{-2}	5.36×10^{-3}	7.98×10^{-2}	1.00×10^{-2}	2.08×10^{-1}	3.17×10^{-2}
	$K=24$	1.19×10^{-2}	3.43×10^{-3}	2.69×10^{-2}	4.20×10^{-3}	9.56×10^{-2}	1.06×10^{-2}
POD	$K=14$	1.13×10^{-1}	4.14×10^{-2}	1.22×10^{-1}	4.10×10^{-2}	6.98×10^{-1}	2.99×10^{-1}
	$K=24$	4.22×10^{-2}	9.91×10^{-3}	3.31×10^{-2}	7.81×10^{-3}	3.10×10^{-1}	7.96×10^{-2}
	$K=32$	8.70×10^{-3}	1.67×10^{-3}	1.32×10^{-2}	2.31×10^{-3}	1.35×10^{-1}	2.43×10^{-2}

Table 6: (Shallow Water) Relative L^2 errors for different reduced dimension K. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE+HNN simulation in yellow.

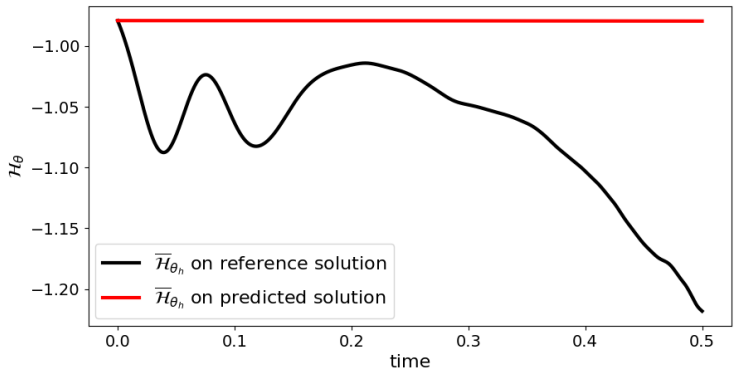


Figure 15: (Shallow Water) Evolution of $\overline{\mathcal{H}}_{\theta_h}$ as a function of time on test 3, on compressed reference solution (black) and predicted solution (red)

5 Conclusion

We have developed a new Hamiltonian reduction method. It is based on an auto-encoder (AE) to transform initial variables into reduced variables and vice-versa, and on an Hamiltonian Neural Network (HNN) to learn the Hamiltonian reduced dynamics. Using a set of coupled loss functions, we are able to learn a reduced model (AE+HNN), which has an Hamiltonian structure. It already has better reduction properties than the PSD in linear test-cases, but the gain is much larger in non-linear test-cases, as expected. Due to its Hamiltonian structure, the reduced model also shows good stability properties.

Obviously, the results will have to be extended to partial differential equations in two or three spatial dimension. As convolution layers are used in the auto-encoder, the increase of dimension should not require too large an increase in the size of the neural networks. In consequence, we expect that the computational gain will be even larger for such dynamics. Other extensions would be to consider time-dependent reductions as in [11] and adapt the method for the reduction of large Hamiltonian differential equations that do not have spatial structures [11].

A Linear reduction

A.1 Proper Symplectic Decomposition

Here we briefly present the Proper Symplectic Decomposition (PSD) [20]. In this approach, we assume that the decoder is a linear operator

$$\mathcal{D}_{\theta_d}(\overline{\mathbf{y}}) = A\overline{\mathbf{y}},$$

with $A \in M_{2N,2K}(\mathbb{R})$. To be consistent with the notations, we keep the subscript θ_d to refer to the decoder parameters, which here are the coefficients of the matrix A. The trial manifold $\widehat{\mathcal{M}}$ is a hyperplane. We further assume that A is symplectic, i.e. satisfies $A^T J_{2N} A = J_{2K}$, and we denote by $\mathrm{Sp}_{2N,2K}(\mathbb{R})$ the set of symplectic matrices. The encoder is thus defined as the symplectic inverse of A:

$$\mathcal{E}_{\theta_e}(\mathbf{y}) = A^+\mathbf{y},$$

with $A^+ = J_{2K}^T A^T J_{2N} \in \mathrm{Sp}_{2K,2N}(\mathbb{R})$ and where $\theta_e = \theta_d$ also denotes the coefficients of the matrix A. Then the projection operator onto $\widehat{\mathcal{M}}$ writes AA^+ and A is determined by solving the minimization problem

$$\min_{A \in \mathrm{Sp}_{2N,2K}(\mathbb{R})} \|Y - AA^+Y\|_F^2,$$

where Y refers to the snapshot matrix:

$$Y = \left[\mathbf{y}_1, \ldots, \mathbf{y}_p\right] \in M_{2N,p}(\mathbb{R}), \tag{21}$$

where $(\mathbf{y}_i)_{i \in [\![1,p]\!]}$ are p values obtained from numerical simulations of the original problem at different times and for different parameters, and $\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}$ is the Frobenius norm of $X \in M_{2N,p}(\mathbb{R})$.

An approximate solution to this problem can be obtained with the cotangent lift algorithm [20]. First, the snapshot matrix Y is reshaped into

$$\tilde{Y} = \left[\mathbf{q}_1, \ldots, \mathbf{q}_p, \mathbf{p}_1, \ldots, \mathbf{p}_p\right] \in \mathcal{M}_{N,2p}(\mathbb{R}).$$

Its Singular Value Decomposition (SVD) is computed leading to $\tilde{Y} = U\Sigma V^T$, where Σ is the diagonal matrix of the singular values and U (resp. V) the matrix of the left (resp. right) singular vectors. Then, we define $\Phi = U[:, :K]$ composed of the K-th left eigenvectors associated with the K-th largest singular values and we consider the following orthogonal symplectic matrix:

$$A = \begin{pmatrix} \Phi & 0 \\ 0 & \Phi \end{pmatrix}.$$

The reduced model satisfied by the reduced variable $\overline{\mathbf{y}} = A^+\mathbf{y}$ can be obtained by multiplying (1) by A^+ so that

$$\frac{d}{dt}\overline{\mathbf{y}} = A^+ J_{2N}\, \nabla_{\mathbf{y}}\mathcal{H}(\mathbf{y}) = J_{2K} A^T\, \nabla_{\mathbf{y}}\mathcal{H}(\mathbf{y}) \approx J_{2K}\, \nabla_{\overline{\mathbf{y}}}(\mathcal{H} \circ A)\,(\overline{\mathbf{y}}),$$

where we use the expression of A^+, the identity $J_{2N}^2 = -\mathrm{Id}$ and the approximation $\mathbf{y} \approx A\overline{\mathbf{y}} = AA^+\mathbf{y}$. The reduced model involves the following reduced Hamiltonian

$$\overline{\mathcal{H}}_{\theta_h} = \mathcal{H} \circ A,$$

where $\theta_h = \theta_e = \theta_d$ still denotes the coefficients of the matrix A, and whose evaluation requires to compute the non reduced quantity $A\overline{\mathbf{y}} \in \mathbb{R}^N$. To avoid this evaluation in large dimension, approximations of the reduced Hamiltonian can be constructed using the discrete empirical interpolation method (DEIM) [27, 20].

The non symplectic version of the algorithm is called the Proper Orthogonal Decomposition (POD) [16], where the matrix A is obtained using the K first singular vectors of the SVD of the original snapshot matrix Y defined in (21).

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[2] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural computation*, 16(10):2197–2219, 2004.

[3] S. Bhattacharjee and K. Matous. A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. *Journal of Computational Physics*, 313, 2016.

[4] P. Buchfink, S. Glas, and B. Haasdonk. Symplectic model reduction of Hamiltonian systems on nonlinear manifolds and approximation with weakly symplectic autoencoder. *SIAM Journal on Scientific Computing*, 45(2):A289–A311, 2023.

[5] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[6] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006. Special Issue: Diffusion Maps and Wavelets.

[7] S. Fresca, L. Dede', and A. Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *J. Sci. Comput.*, 87:1–36, 2021.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[9] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[10] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration, Structure-preserving algorithms for ordinary differential equations*, volume 31. Springer-Verlag Berlin, 2006.

[11] J. Hesthaven, C. Pagliantini, and N. Ripamonti. Adaptive symplectic model order reduction of parametric particle-based Vlasov–Poisson equation. *Mathematics of Computation*, 2023.

[12] J.S. Hesthaven, C. Pagliantini, and G. Rozza. Reduced basis methods for time-dependent problems. *Acta Numerica*, 31:265–345, 2022.

[13] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

[14] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

[15] K. Lee and K.T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[16] Y.C. Liang, H.P. Lee, S.P. Lim, W.Z. Lin, K.H. Lee, and C.G. Wu. Proper orthogonal decomposition and its applications - part i: Theory. *Journal of Sound and Vibration*, 252:527–544, 05 2002.

[17] B. Maboudi Afkham and J.S. Hesthaven. Structure-preserving model-reduction of dissipative hamiltonian systems. *Journal of Scientific Computing*, 81(1):3–21, 2019.

[18] R. Maulik, B. Lusch, and P. Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, 2021.

[19] C. Pagliantini. Dynamical reduced basis methods for hamiltonian systems. *Numerische Mathematik*, 148(2):409–448, 2021.

[20] L. Peng and K. Mohseni. Symplectic model reduction of Hamiltonian systems. *SIAM Journal on Scientific Computing*, 38(1):A1–A27, 2016.

[21] F. Romor, G. Stabile, and G. Rozza. Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method. *J. Sci. Comput.*, 94(3):74, 2023.

[22] B.E. Sonday, A. Singer, C.W. Gear, and I.G. Kevrekidis. Manifold learning techniques and model reduction applied to dissipative pdes. *arXiv e-prints*, pages arXiv–1011, 2010.

[23] J. B. Tenenbaum, V. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[24] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

[25] T.M. Tyranowski and M. Kraus. Symplectic model reduction methods for the Vlasov equation. *Contributions to Plasma Physics*, 63(5-6):e202200046, 2023.

[26] Q. Wang, N. Ripamonti, and J.S. Hesthaven. Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism. *Journal of Computational Physics*, 410:109402, 2020.

[27] Z. Wang. Structure-Preserving Galerkin POD-DEIM Reduced-Order Modeling of Hamiltonian Systems, 2021.