Journal of Scientific Computing (2023) 94:74
https://doi.org/10.1007/s10915-023-02128-2

Non-linear Manifold Reduced-Order Models with Convolutional Autoencoders and Reduced Over-Collocation Method

Francesco Romor[1] · **Giovanni Stabile**[2] · **Gianluigi Rozza**[1]

Received: 28 February 2022 / Revised: 24 October 2022 / Accepted: 20 January 2023 /
Published online: 14 February 2023

Abstract
Non-affine parametric dependencies, nonlinearities and advection-dominated regimes of the model of interest can result in a slow Kolmogorov n-width decay, which precludes the realization of efficient reduced-order models based on linear subspace approximations. Among the possible solutions, there are purely data-driven methods that leverage autoencoders and their variants to learn a latent representation of the dynamical system, and then evolve it in time with another architecture. Despite their success in many applications where standard linear techniques fail, more has to be done to increase the interpretability of the results, especially outside the training range and not in regimes characterized by an abundance of data. Not to mention that none of the knowledge on the physics of the model is exploited during the predictive phase. In order to overcome these weaknesses, we implement the non-linear manifold method introduced by Lee and Carlberg (J Comput Phys 404:108973, 2020) with hyper-reduction achieved through reduced over-collocation and teacher–student training of a reduced decoder. We test the methodology on a 2d non-linear conservation law and a 2d shallow water models, and compare the results obtained with a purely data-driven method for which the dynamics is evolved in time with a long-short term memory network.

Keywords Non-linear model order reduction · Manifold learning · Data-driven methods · Computational fluid dynamics

Mathematics Subject Classification 68T07 · 65M22 · 65D15

✉ Gianluigi Rozza
gianluigi.rozza@sissa.it

Francesco Romor
francesco.romor@sissa.it

Giovanni Stabile
giovanni.stabile@uniurb.it

[1] Mathematics Area, mathLab, SISSA, via Bonomea 265, 34136 Trieste, Italy

[2] Department of Pure and Applied Sciences, Informatics and Mathematics Section, University of Urbino Carlo Bo, Piazza della Repubblica, 13, 61029 Urbino, Italy

1 Introduction

In real world engineering scenarios, when performing outer loop applications such as optimization, uncertainty quantification, sensitivity analysis, parametric partial differential equations (PDEs) often need to be solved numerically numerous times. However, relying on the mathematical properties of some parametric PDEs, the computational cost for many query problems can be drastically reduced taking into account previous results on a set of training parameters: the procedure for the design of reduced-order models (MORs) is divided in an offline (training) stage, during which a set of training solutions is collected, and an online (testing or predictive) stage, which employs the compressed information from the previous step to predict the solutions of the PDE of interest for unseen parameters. This reduction is performed numerically defining a low-dimensional global basis devised in the offline stage, and can be carried out independently of the class of numerical methods chosen: finite element (FEM), spectral element (SEM), discontinuous Galerkin (DGM), and finite volumes method (FVM). One of the most employed model-order reduction method (MOR) is the reduced basis method [1, 2].

Depending on the parametric dependency and mathematical nature of some PDEs, various issues may occur: the Kolmogorov n-width (KnW) is used to characterize the approximability of the solution manifold, that is the set of parameter-dependent solutions of the PDE, by a linear trial subspace. A slow decaying KnW is a symptom of the difficulties in the design of efficient ROMs: this results in the necessity of using a high number of reduced basis or proper orthogonal decomposition (POD) method's modes, corrupting the efficiency of the ROMs till the point that the gain into the computational cost becomes irrelevant. One class of PDEs where this behaviour is evident are time-dependent advection-dominated PDEs. Moreover, non-linear PDEs require hyper-reduction procedures to make the reduced equations independent of the number of degrees of freedom of the full-order model (FOM).

Recently, leveraging machine learning's advances in manifold learning, a class of ROMs that employ a non-linear trial manifold built with convolutional autoencoders (CAEs) [3] was developed by Carlberg et al. [4]. One of the benefits of this approach is the possibility to employ a small latent dimension of the ROMs, thus overcoming the slow decay of the KnW for some parametric PDEs, at the expense of introducing additional nonlinearities from the neural networks (NNs) and sometimes more substantial training costs in the offline stage. The properties of the non-linear manifold methods include the need of less stabilization mechanisms, the less intrusiveness on the FOM solvers—they are in fact equations-based rather than fully-intrusive—and the possibility to apply them for a much broader class of parametric PDEs, differently from ROMs devised specifically for advection-dominated problems.

A hyper-reduction scheme for non-linear manifold Least-Squares Petrov–Galerkin (NM-LSPG) and non-linear manifold Galerkin (NM-G), is introduced in [5]: it relies on Gauss–Newton tensor approximation (GNAT) [6] hyper-reduction method and shallow masked autoencoders to select only the degrees of freedom that explain the dynamics and therefore restrict efficiently the decoder and the discretized residuals. As we will see in our test cases, the reconstruction error of the autoencoder employed empirically bounds from below the errors of all the other non-linear manifold ROMs built upon. Therefore, we devise a method that is independent on the choice of the architecture: a sparse shallow autoencoder is not necessary anymore, and any NN architecture, like CAEs, could be in principle employed. This frees the way to imposing additional inductive biases that help to speed up the offline stage and to achieve accurate approximations of the discrete solution manifolds, a crucial requirement. Moreover, in some cases, reconstructing the residuals with GNAT is

not efficient, still because of a slow decaying KnW, so we choose to employ the reduced over-collocation hyper-reduction method (ROC) [7]: in this case the equation's numerical residual is not reconstructed on a global basis, but collocated on some nodes of the mesh called collocation points.

Once the CAE reaches a satisfactory approximation of the discrete solution manifold, purely data-driven NN PDE can be trained to predict the latent dynamics: the gold standard that is being established for this task are long-short term memory networks (LSTM). Their online computational cost is low even w.r.t linear ROMs, but some new issues appear: their accuracy depends on the regularity of the latent dynamics, especially when predicting the solutions for parameters outside the training range, in the extrapolation regimes; they require hyperparameters tuning, and all the connections to the PDEs model are completely lost, resulting in a loss of interpretability of the results. The non-linear manifold hyper-reduced ROMs we develop solve these issues, at the expense of a higher computational cost in the online stage, since at each time step a physics-based residual is minimized. Moreover, a posteriori error estimates are available [4, 5]. In our test cases, we compare these two approaches to enlight their differences, weak and strong points.

The structure of this paper is as follows. In Sect. 2 we delve into the topic of manifold learning which has many connections with reduced-order modelling, especially since the recent entry of machine learning in the design of ROMs. We will proceed introducing the Kolmogorov n-with (KnW), and we will show that some classes of parametric PDEs suffer from the so called slow decaying Kolmogorov n-width. In Sect. 3, the non-linear manifold (NM) reduced-order models based on the work of Carlberg et al. [4] are introduced. We will focus on the non-linear manifold least-squares Petrov–Galerkin method (NM-LSPG). Afterwords, we describe our new hyper-reduced ROMs: NM-LSPG with reduced over-collocation (NM-LSPG-ROC) and NM-LSPG with reduced over-collocation and teacher–student training of a compressed decoder (NM-LSPG-ROC-TS). These two approaches, to the best of the authors' knowledge, are introduced here for the first time. In Sect. 4, the new model order reduction (MOR) methods are tested on a 2d parametric non-linear time-dependent conservation law model and a 2d parametric non-linear time-dependent shallow water equations model. In Sect. 5, a discussion on the results obtained follows, and in the conclusive Sect. 6 possible future directions of research are explored.

2 Manifold Learning

The subject of manifold learning, classified as a topic of machine learning, had its unique flavour in model order reduction even before nowadays breakout of scientific machine learning [8]. The workhorse of the model order reduction community is POD or SVD. A lot of real applications though, required new methods to approximate the solution manifold in a non-linear fashion. The symptom of this behaviour is a slow decaying Kolmogorov n-width. Some approaches rely on the locality of the validity region of a linear approximation with POD, both in the parameter space and in the spatial and temporal domains, others implement non-linear dimensionality reduction methods from machine learning [9]: kernel principal component analysis (KPCA), Isomap, clustering algorithms. Non-linear MORs include approximations by rational functions, splines or other non-linear functions collected in a dictionary [10].

Interpolatory approaches of the solution manifold with respect to the parameters have been developed, sometimes combined with non-linear dimension reduction techniques like KPCA and its variants: interpolation with geodesics on the Grassmann manifold [11], inter-

polation on the latent space obtained with Isomap dimensionality reduction method [12, 13], interpolation with optimal transport [14], dictionary-based ROM that make use of clustering in the Grassmannian manifold and classification with neural networks (NN) [15], local kernel principal component analysis [16]. At the same time, domain decomposition approaches tackled locality in space [17, 18].

One particular class of dimension reduction techniques is represented by autoencoders, and more generally by other architectures that rely on NNs. In the recent literature many achievements are brought by CAEs, and by extension by Generative Adversarial Networks (GANS), Variational Autoencoders (VAEs), Bayesian convolutional autoencoders [3]: in [19] convolutional autoencoders are utilized for dimensionality reduction and long-short Term Memory (LSTM) NNs or causal convolutional neural networks are used for time-stepping; in [20] the evolution of the dynamics and the parameter dependency is learned at the same time of the latent space with a forward NN and a CNNs on randomized SVD compressed snapshots, respectively; and in [21] spatial and temporal features are separately learned with a multi-level convolutional autoencoder.

In order to extend these architectures to datasets not structured in orthogonal grids, geometric deep learning [22] is called to the task. There are not many works on geometric deep learning applied to model order reduction for mesh-based simulations that achieve the same accuracy of CNNs, yet. Promising results are reached by an architecture that employs graph neural networks (GNNs) and a physics-informed loss [23]. In the future, the potential of GNNs will probably be leveraged extending the range of applicability of nowadays frameworks.

The setting we will base our studies on, does not depend directly on the numerical method used to discretize the parametric PDEs at hand (FVM, FEM, SEM, DGM), so the mathematical formulation will generically be founded on models represented by a parametric system of time-dependent (but also time-independent) PDEs, consisting of a non-linear parametric differential operator $\mathcal{G}$ and of the boundary differential operators $\mathcal{B}, \mathcal{B}_0$ that represent the boundary and initial conditions respectively,

$$\forall \boldsymbol{\mu} \in \mathcal{P} \qquad \begin{cases} \mathcal{G}(\boldsymbol{\mu}, U(\boldsymbol{\mu}, t, \mathbf{x})) & = 0 \quad (\mathbf{x}, t) \in \Omega \times [0, T], \\ \mathcal{B}(\boldsymbol{\mu}, U(\boldsymbol{\mu}, t, \mathbf{y})) & = 0 \quad (\mathbf{y}, t) \in \partial\Omega \times [0, T], \\ \mathcal{B}_0(\boldsymbol{\mu}, U(\boldsymbol{\mu}, 0, \mathbf{x})) & = 0 \quad \mathbf{x} \in \Omega, \end{cases} \tag{1}$$

where $\mathcal{P}$ is the parameter space, U are the state variables and Ω is the 2D or 3D spatial domain. This formulation includes also coupled systems of PDEs. We assume that the solutions belong to a certain Banach or Hilbert space Y, varying $(\boldsymbol{\mu}, t) \in \mathcal{P} \times [0, T]$. The solution manifold $\mathcal{M}$ is represented by the set

$$\mathcal{M} = \{U(\boldsymbol{\mu}, t) \in Y \mid \boldsymbol{\mu} \in \mathcal{P}, t \in [0, T]\}. \tag{2}$$

2.1 Approximability by n-Dimensional Subspaces and Kolmogorov n-Width

We want to remark some results available in the literature, in order to state and comment, for our needs, the problem of solution manifold approximability. In particular, our benchmarks belong to a class of parametric PDEs for which the Kolmogorov n-width decays slowly. Thus, classical Petrov–Galerkin projection with POD needs to be overcome with non-linear methods in place of POD to achieve efficient ROMs.

Let $(X, \|\cdot\|)_X$ be a complex Banach space, and $K \subset X$ a compact subspace, the Kolmogorov n-width (KnW) of K in X is defined as

$$d_n(K)_X = \inf_{\dim(W)=n} \max_{v\in K} \min_{w\in W} \|v - w\|_X. \tag{3}$$

Let $(Y, \|\cdot\|_Y)$ be a complex Banach space and $L : K \subset\subset X \to Y$. In our framework K is the parameter space, possibly infinite dimensional and L is the solution map of the system of parametric PDEs at hand, from the parameter space to the solution manifold. In order to define L we have to suppose that for each parameter in K there is a unique solution in Y.

Following [24], it can be proved that for holomorphic L, thus not necessarily linear, the Kolmogorov n-width decay is one polynomial order below the Kolmogorov n-width of the parameter space K.

Theorem 1 [24, Theorem 1] *Suppose u is a holomorphic mapping from an open set $O \subset X$ into Y and u is uniformly bounded on O,*

$$\exists B > 0, \quad \sup_{x\in O} \|u(x)\|_Y \leq B. \tag{4}$$

If $K \subset O$ is a compact subset of X, then for any $s > 1$ and $t < s - 1$,

$$\sup_{n\geq 1} n^s d_n(K)_X < \infty \Rightarrow \sup_{n\geq 1} n^t d_n(u(K))_K < \infty. \tag{5}$$

In particular, if the hypothesis of the previous theorem are satisfied and if K is a finite dimensional linear subspace, the Kolmogorov n-width decay is exponential. In general, elliptic PDEs, affinely decomposable with respect to the parameters, satisfy the hypothesis of the previous theorem [25, 26]. Not always nonlinearities cause a slow decaying KnW: using Theorem 1, in [24] they prove that the parametric PDE on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^3$

$$u^3 - \nabla \cdot (\exp a)\nabla u = f, \tag{6}$$

$$K = \{a \in L^\infty(\Omega) : \|a\|_{C^\alpha} \leq M\}, \quad f \in H^{-1}(\Omega), \tag{7}$$

with homogeneous Dirichlet boundary conditions, where C^α is the space of Hölder functions, satisfies the hypothesis of the previous theorem. Actually, for Hölder functions the KnW is bounded above by $n^{-\alpha/3}$, which is not a fast convergence, but if instead a belongs to the Sobolev space $W^{s,\infty}(\Omega)$ then the upper bound is $n^{s/3}$ [27]. We will consider a good KnW decay if it has a higher infinitesimal order than n^{-1}.

The same is not valid in the case of the simplest linear advection problem. We briefly report some results from the literature on classical hyperbolic PDEs, for $(t, \mu) \in K = [0, 1]^2$ with the standard norm and $Y = L^2([0, 1])$,

$$L : (t, \mu) \mapsto u(t, x, \mu) \quad \text{s.t.} \quad \partial_t u - \mu \partial_x u = 0 \qquad d_n(\mathcal{M})_{L^2} > n^{-\frac{1}{2}}, \tag{8}$$

$$L : (t, \mu) \mapsto u(t, x, \mu) \quad \text{s.t.} \quad \partial^2_{tt} u - \mu \partial^2_{xx} u = 0 \qquad d_n(\mathcal{M})_{L^2} > n^{-\frac{1}{2}}, \tag{9}$$

where, the results are respectively proven in [28] and [29].

This behaviour is not restricted only to advection-dominated problems. Intuitively also solution manifolds that are characterized by a parametrized locality in space suffer from slow decaying KnW, like elliptic problems with singular sources parametrically moving in the domain [30].

Our newly developed ROMs, should solve the issue of slow decaying KnW in the applications, guaranteeing a low latent or reduced dimension of the approximate solution manifold.

This, because the linear trial manifold, frequently generated by the leading POD modes, is substituted with a non-linear trial manifold, parametrized by the decoder of an autoencoder. The test cases we present in Sect. 4, were chosen in order to be advection-dominated and particularly not suited for linear ROMs, as shown in [5] for the 2d Burgers' equation, that has a close relationship with the NCL problem.

Remark 1 *(Extensions of the Kolmogorov n-width to non-linear approximations)* The autoencoders we implement in our test cases are at least continuous as composition of continuous activations and linear functions. In the literature there exist possible non-linear extensions of the KnW such as the manifold width [31],

$$\delta_n(L(K), X) = \inf_{\substack{\psi \in \mathcal{C}(X, \mathbb{R}^n) \\ \phi \in \mathcal{C}(\mathbb{R}^n, X)}} \sup_{x \in L(K)} \|x - (\phi \circ \psi)(x)\|_X, \tag{10}$$

library widths [32], and entropy numbers, for more insights see [10].

2.2 Singular Values Decomposition and Discrete Spectral Decay

From the discrete point of view the same problematic in tackling the reduction of parametric PDEs with slow KnW decay is encountered: in this case the discrete solution manifold is actually the set of discrete solutions of the full-order model for a selected finite set of parameters. Singular Value Decomposition (SVD) or eigenvalue decomposition (for symmetric positive definite matrices), usually employed on the snapshots matrix for the evaluation of the reduced modes, are characterized by the fact that modes are linear combinations of the snapshots. This is not enough to approximate snapshots that are orthogonal with respect to the collection of the training set.

In practice, looking at the residual energy retained by the discarded modes, is an indicator of approximability with linear subspaces. Let us assume that d is the total number of degrees of freedom of the discrete problem, N_{train} is the number of training snapshots, and, r, the reduced dimension such that, $r \leq N_{\text{train}} \ll d$. Then, $X \in \mathbb{R}^d \times \mathbb{R}^{N_{\text{train}}}$ is the matrix that has the training snapshots $\{U_i\}_{i=1}^{N_{\text{train}}}$ as columns, $V \in \mathbb{R}^d \times \mathbb{R}^r$ are the reduced modes from the SVD of X, and $\{\sigma_i\}_{i=1}^d$ are the increasingly ordered singular values. It is valid the following relationship of the residual energy (to the left) with the KnW (to the right),

$$\sqrt{\sum_{i=r+1}^{d} \sigma_i^2} = \|(I - VV^T)X\|_F \geq \max_{i=1,\ldots,N_{\text{train}}} \|(I - VV^T)U_i\|_2 \geq d_n(\{U_i\}_{i=1}^{N_{\text{train}}})_{\mathbb{R}^d},$$

where $\|\cdot\|_F$ is the Frobenious norm.

Even though some problems have a slow decaying KnW, this affects only the asymptotic convergence of the ROMs w.r.t. the reduced dimension, while for some applications a satisfying accuracy of the projection error is reached with less than 100 modes, as shown in our benchmarks. So what is actually lost in MOR for problems with a slow decaying KnW is the fast convergence of the projection error w.r.t. the reduced dimension, not the possibility to perform a MOR with enough modes. Moreover, the discrete solution manifold's KnW depends on the time step and spatial discretization size, so that, especially when a coarse mesh is employed, the Knw decays faster w.r.t. the KnW of the continuous solution manifold.

2.3 Convolutional Autoencoders

We have chosen to overcome the slowly decaying KnW problem employing autoencoders [3] as non-linear dimension reduction method substituting POD. Some ROMs predict the latent dynamics on a linear trial manifold with artificial neural networks [33], so are still classified as linear ROMs and, in fact, they are still affected by the slow KnW decay. We remark that while some non-linear approaches to model order reduction are specifically tailored for advection-dominated problems [34, 35], autoencoders are a more general approach. On the other hand, they are also particularly suited to advection-dominated problems with respect to local and/or partitioned ROMs that implement domain decomposition, even when non-linear dimension reduction techniques are employed locally [16]: this is because considering a non-discrete parametric space, like the time interval of a simple linear advection problem, an infinite number of local linear and/or non-linear ROMs would be needed to counter the slow decaying KnW. As anticipated, we implement convolutional autoencoders [3] in libtorch the C++ frontend of PyTorch [36]. We remark that the procedure we developed, considering also teacher–student training of a reduced decoder in Sect. 3.4, can be generally extended to any architecture that can approximate with a sufficiently good accuracy the solution manifold through a low-dimensional latent representation. The choice of a CAE is particularly beneficial when the solution snapshots are associated to a structured mesh and when the components of the vectorial solution fields to be approximated have similar features so that the convolutional filters can be shared among the channels of the CAE.

Let us define $X_h \subset \mathbb{R}^d$ as the state discretization space with d the number of degrees of freedom and h the discretization step. The snapshots are divided in a training set $\mathcal{U}_{\text{train}} = \{\mathbf{U}_i\}_{i=1,\dots,N_{\text{train}}} \subset X_h$ and a test set $\mathcal{U}_{\text{test}} = \{\mathbf{U}_i\}_{i=1,\dots,N_{\text{test}}} \subset X_h$. If the problem has different states and/or vectorial states the training and test set are split in channels for each state and/or component. For example in the 2d non-linear conservation law, we consider two channels, one for each velocity component. In the shallow water test case, we consider three channels, one for each velocity component and one for the free surface height. So, in general, we reshape the snapshots such that $\mathcal{U}_{\text{train}}, \mathcal{U}_{\text{test}} \subset (\mathbb{R}^{d/c})^c$ where c is the number of channels.

As preprocessing step the snapshots are centered and normalized to assume values in the interval $[-1, 1]$

$$\frac{2}{\mathcal{U}_{\max} - \mathcal{U}_{\min}} \left(\mathcal{U}_{\text{train}} - \mathcal{U}_{\text{mean}} - \frac{\mathcal{U}_{\min} + \mathcal{U}_{\max}}{2} \right), \tag{11}$$

where $\mathcal{U}_{\text{mean}}, \mathcal{U}_{\max}, \mathcal{U}_{\min} \in (\mathbb{R}^{d/c})^c$ are evaluated channel wise. The same values obtained from the training snapshots $\mathcal{U}_{\text{train}}$, are employed to center and normalize the test set $\mathcal{U}_{\text{test}}$.

We define the encoder $\psi : (\mathbb{R}^{d/c})^c \to \mathbb{R}^r$ and the decoder $\phi : \mathbb{R}^r \to (\mathbb{R}^{d/c})^c$, where $r \ll d$ is the reduced or latent dimension, as neural networks made by subsequent convolutional layers and linear layers at the end and at the beginning, respectively. For the particular architecture used in the applications we defer it to the "Appendix A". In Fig. 1 is represented the convolutional autoencoder applied for the 2d non-linear conservation law test case, with an approximate size of the filters and the actual number of layers.[1]

Remark 2 *(Regularity)* Regarding the regularity of the CAE, related to the choice of activation functions, it is proved in Theorem 4.2 from [4] that NM-LSTM and non-linear manifold Galerkin methods are asymptotically equivalent provided that the decoder is twice differentiable. Since we are only employing the NM-LSTM method, our only concern in the choice

[1] Figures 1 and 3 were made with the open-source package from GitHub https://github.com/HarisIqbal88/PlotNeuralNet.

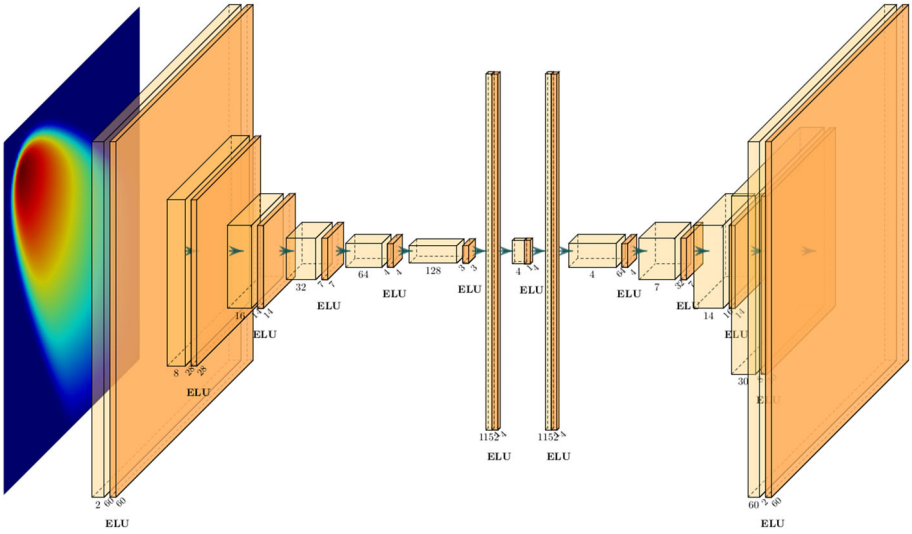

Fig. 1 The convolutional autoencoder architecture employed for the 2d non-linear conservation law test case: the same number of convolutional layers are shown, while the sizes of each layer are rescaled for a better graphical representation of the architecture (Color figure online)

of activation functions is that the reconstruction error is sufficiently low, so that the accuracy of the whole procedure is not undermined.

For each batch $\{\mathbf{U}_i\}_{i=1}^{b} \subset \mathcal{U}_{\text{train}}$, the loss employed is the sum of the reconstruction error, and a regularizing term for the weights:

$$\mathcal{L}(\{\mathbf{U}_i^{t_i}\}_{i=1}^{b}; \Theta) = \frac{1}{b}\sum_{i=1}^{b} \frac{\|\mathbf{U}_i^{t_i} - (\phi \circ \psi)(\mathbf{U}_i^{t_i})\|_2^2}{\|\mathbf{U}_i^{t_i}\|_2^2} + \lambda_1 \|\Theta\|_2^2, \tag{12}$$

where Θ represents the weights of the convolutional autoencoder. The choice of the relative mean squared reconstruction error is important when, varying the parameter $\{\boldsymbol{\mu}_i\}_{i=1}^{N_{\text{train}}}$, the snapshots $\{\mathbf{U}_i\}_{i=1}^{N_{\text{train}}}$ have different orders of magnitude: for example this is the case of flows propagating from a local source on the whole domain with a constant zero state as initial condition.

The training is performed with Adam stochastic optimization method [37]. After the training the whole evolution of the dynamics is carried out with a non-linear optimization algorithm minimizing the residual on the latent domain, as described in Sect. 3. For each new parametric instance and associated initial state $\mathbf{U}_0 \in X_h$, the latent initial condition is found with a single forward of the encoder $z_0 = \psi(\mathbf{U}_0)$ after centering and normalizing $\mathbf{U}_0$. Then, for each time instant t, the decoder

$$\phi(z(t)) = \mathbf{U}(t) \in (\phi \circ \psi)(X_h) \subset \mathbb{R}^d, \tag{13}$$

is used as parametrization of an approximate solution manifold as will be explained in Sect. 3, including in ϕ the renormalization of the output.

Remark 3 *(Initial condition)* With respect to the initial implementation of Carlberg et al. [4] our approach for the definition of the latent parametrization of the solution manifold is

different in the management of the initial condition. Instead of using directly the decoder ϕ, they define the map from the latent to the state space $f : \mathbb{R}^r \to \mathbb{R}^d$, including the renormalization in the ϕ map for brevity, as

$$f(z) = \phi(z) + \mathbf{U}_0(\mu) - \phi(\psi(\mathbf{0})), \quad \text{s.t.} \quad f(\psi(\mathbf{0})) = \mathbf{U}_0(\mu), \tag{14}$$

so that the reconstruction is exact at the initial parametric time instances. So, supposing that the initial condition is parametrically dependent, all initial conditions coincide to $\psi(\mathbf{0})$ in the latent space, and the decoder has to learn variations from the initial latent condition. We prefer instead to split the initial conditions in the latent space in order to aid the non-linear optimization algorithm, and leave to the training of the CAE the accurate approximation of the initial condition. This splitting of the initial conditions can be seen in Figs. 6 and 13. The additional cost of our implementation is the forward of the initial parametric condition through the encoder as first step.

Remark 4 *(Inductive biases)* Imposing inductive biases to increase the convergence speed of a deep learning model to the desired solution has always been a winning strategy in machine learning. This is translated in the context of physical models with the possibility to include, among others, the following inductive biases: first principles (conservation laws [38], equations governing the physical phenomenon), geometrical simmetries (group invariant filters [39, 40]), numerical schemes/residuals (discrete residuals, latent time advancement with Runge-Kutta schemes), latent regularity (minimize the curvature of latent trajectories), latent dynamics (linear or quadratic latent dynamics [41]). As inductive bias, we will impose the positivity of the state variables that are known to be positive throughout their trajectory with a final ReLU activation.

3 Evolution of the Latent Dynamics with NM-LSPG-ROC

The non-linear manifold method introduced by Carlberg et al. [4] does not perform a complete dimension reduction since at each time step the decoder reconstructs the state from the latent coordinates to the whole domain, still depending on the number of degrees of freedom of the FOM. We revisit the non-linear manifold least-squares Petrov–Galerkin method (NM-LSPG) with small modifications and introduce two novel hyper-reduction procedures: one combines teacher–student training of a compressed decoder with the reduced over-collocation method (NM-LSPG-ROC-TS), the other implements only the hyper-reduction of the residual with reduced over-collocation (NM-LSPG-ROC).

3.1 Non-linear Manifold Least-Squares Petrov–Galerkin

We assume that the numerical method of preference discretizes the system (1) in space and in time with an implicit scheme, $G_{h,\delta t} : \mathcal{P} \times X_h \times X_h^{|I_t|} \to X_h$

$$G_{h,\delta t}(\boldsymbol{\mu}, \mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s \in I_t}) = \mathbf{0}, \tag{15}$$

where h, δt are the spatial and temporal discretization steps chosen, $X_h \subset \mathbb{R}^d$ is the state discretization space (d is the number of degrees of freedom), and I_t is the set of past state indexes employed in the temporal numerical scheme to solve for $\mathbf{U}_h^t$. We remark that the numerical discretization employed can differ from the one used to solve for the full-order training snapshots. The method is thus equations-based rather than fully intrusive. This will be the case for the 2d shallow water equations model in Sect. 4.2.

For each discrete time instant t the following non-linear least-squares problem is solved for the latent state $\mathbf{z}^t \in Z$, with the Levenberg–Marquardt algorithm [42]

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \|G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})\|^2_{X_h}. \tag{16}$$

That is for each time instant the following intermediate solutions $\{\mathbf{z}^{t,k}\}_{k\in\{0,\dots,N(t)\}}$, $\mathbf{z}^{t,0} = \mathbf{z}^{t-1,N(t-1)}$ of the linear system in $\mathbb{R}^r$ are computed,

$$\left((dG^{t,k-1}d\phi^{t,k-1})^T dG^{t,k-1}d\phi^{t,k-1} + \lambda\, I_d\right)\delta\mathbf{z}^{t,k} = -(dG^{t,k-1}d\phi^{t,k-1})^T G^{t,k}, \tag{17}$$

$$\mathbf{z}^{t,k} = \mathbf{z}^{t,k-1} + \alpha^k\, \delta\mathbf{z}^{t,k}, \tag{18}$$

where

$$d\phi^{t,k-1} := \frac{d\phi(\mathbf{z}^{t,k-1})}{d\mathbf{z}^{t,k-1}} \in \mathbb{R}^{d\times r}, \tag{19}$$

$$dG^{t,k-1} := \left.\frac{dG_{h,\delta t}(\boldsymbol{\mu}, \mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s\in I_t})}{d\mathbf{U}_h^t}\right|_{(\mathbf{U}_h^t,\{\mathbf{U}_h^s\}_{s\in I_t})=(\phi(\mathbf{z}^{t,k-1}),\{\phi(\mathbf{z}^s)\}_{s\in I_t})} \in \mathbb{R}^{d\times d}, \tag{20}$$

λ is a factor that evolves during the non-linear optimization and balances between a Gauss–Newton and a steepest descent method, and finally α^k is a parameter found with a trust-region method. In the implementation in Eigen [43], $\lambda\, I_d$ is scaled with respect to the diagonal elements of $(dG^{t,k-1}d\phi^{t,k-1})^T dG^{t,k-1}d\phi^{t,k-1}$. All the tolerances for convergence are set to machine precision, and the maximum number of residual evaluations is set to 7 unless explicitly stated differently in the numerical results Sect. 4.

Remark 5 *(Least-squares Petrov–Galerkin)* The method is called manifold LSPG because it refers to the LSPG method usually applied when the manifold is linear. It consists in multiplying the residual to the left with a different matrix Ψ with respect to the linear embedding Φ of the reduced coordinates into the state space,

$$\Psi^T G_{h,\delta t}(\boldsymbol{\mu}, \Phi\mathbf{z}) = 0, \tag{21}$$

where, $\Phi \in \mathbb{R}^{d\times r}$ is the basis of the linear reduced manifold contained in $X_h \subset \mathbb{R}^d$, $\mathbf{z} \in \mathbb{R}^r$, and Ψ is to be defined: the left subspace $\Psi \in \mathbb{R}^{d\times r}$ is used to enforce the orthogonality of the non-linear residual to a left subspace $\mathcal{L} \subset \mathbb{R}^d$. Applying Newton's method because of the nonlinearities, the problem is translated into the iterations for $k = 1, \dots, K$:

$$\Psi^T dG\Phi\, \delta\mathbf{z}^k = -\Psi^T G^k, \tag{22}$$

$$\mathbf{z}^k = \mathbf{z}^{k-1} + \alpha^k\, \delta\mathbf{z}^k. \tag{23}$$

The step length α^k is computed after a line search along the direction p^k. Usually Φ is chosen from a POD basis of the state variable $\mathbf{z} \in X_h$. For the left subspace, that imposes orthogonality constraints, different choices can be applied. In general, given the system

$$dG\Phi\, \delta\mathbf{z}^k = -G^k, \tag{24}$$

the least squares solution is the one orthogonal to the range of $dG\Phi$. In the case of dG symmetric positive definite we have that $\Psi \subset < dG\Phi >=< \Phi >$ i.e. $\Psi = \Phi$ and the method is called Galerkin projection, but in general if this is not true then the optimal left subspace remains $\Psi = dG\Phi$. For examples where the Galerkin projection is not optimal see [44, Sect. 3.4], *Numerical comparison of left subspaces*. This is often the case for advection-dominated discretized systems of PDEs: in these cases LSPG is preferred to Galerkin projection.

Remark 6 *(Manifold Galerkin)* The manifold Galerkin method proposed in [4] assumes that the columns of the Jacobian of the decoder are good approximations of the state velocity space: if a spatial discretization is applied and the residual has the form $\mathcal{G}_h(\boldsymbol{\mu}, \mathbf{U}) = \dot{\mathbf{U}} - \mathbf{f}(\boldsymbol{\mu}, \mathbf{U})$, where $\mathbf{f}$ is a generic, possibly non-linear, vector field, then

$$\dot{\mathbf{z}} = \underset{\mathbf{v}\in\mathbb{R}^r}{\operatorname{argmin}} \|d\phi(\mathbf{z})\mathbf{v} - \mathbf{f}(\boldsymbol{\mu}, \phi(\mathbf{z}))\|^2, \tag{25}$$

is solved for the latent state velocity, under the hypothesis that $d\phi(\mathbf{z})$ has full-rank and $d\phi$ is a good approximation of the full-order state velocity even if the autoencoder is trained only on the values of the state for different times and parameters, without considering its velocity. If ϕ is linear, we obtain the Galerkin method presented in the Remark 5, after having applied a temporal discretization scheme and multiplied the resulting equation to the left with $d\phi(\mathbf{z}) = \Phi$ as is the case for linear Galerkin projection: the discretize-then-project and project-then-discretize approaches are equivalent in this case [4].

The LSPG performs better as shown in [4], even if they are asymptotically equivalent also in the case of a non-linear manifold, provided ϕ is twice differentiable. So we have chosen to employ only the NM-LSPG method and do not compare it with the non-linear manifold Galerkin (NM-G) method.

For the numerical tests we have performed, the numerical approximation of the Jacobian of the residual $G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})$ is accurate enough. So in the implementation, at each iteration step the Jacobian of the residual with respect to the latent variable, that is $dG^{t,k-1}d\phi^{t,k-1}$ of Eq. (17), is approximated with finite differences. The step size is taken sufficiently lower than the distance between consecutive latent states.

3.2 Reduced Over-Collocation Method

At the point of Eq. (17), the model still depends on the number of degrees of freedom of the full-order model d, since at each time step and optimization step the latent reduced variable $\mathbf{z} \in \mathbb{R}^r$ is forwarded to the reconstructed state $\mathbf{U}_h = \phi(\mathbf{z}) \in \mathbb{R}^d$. A possible solution is represented by the reduced over-collocation method [7], for which the least squares problem (16) is solved only on a limited number of points $r < r_h << d$,

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \| P_{r_h} G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{26}$$

where P_{r_h} is the projection onto r_h standard basis elements in $\mathbb{R}^d$ associated to the over-collocation nodes or magic points and selected as described later. Afterwards the Levenberg–Marquardt algorithm is applied as described in the previous section, to solve the least squares problem (26).

At this point we make the assumption that the method used to discretize the model has a local formulation so that each discrete differential operator can be restricted to the nodes/magic points of the hyper-reduction and consequently, the least-squares problem (26) reduces to

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \| P_{r_h} G_{h,\delta t}(\boldsymbol{\mu}, P_{r_h}(\phi(\mathbf{z})), \{P_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{27}$$

$$= \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \|\tilde{G}_{h,\delta t}(\boldsymbol{\mu}, \tilde{\phi}(\mathbf{z}), \{\tilde{\phi}(\mathbf{z}^s)\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{28}$$

where the projected residual $\tilde{G} = P_{r_h} \circ G$ is introduced and the compressed decoder $\tilde{\phi}$ is defined to substitute $P_{r_h} \circ \phi$ with another embedding from the latent space to the hyper-

reduced space in $\mathbb{R}^{r_h}$, such that the whole structure of the decoder is reduced as described in Sect. 3.4 to further decrease the computational cost.

The Levenberg–Marquardt method is applied also to the hyper-reduced system

$$\left((d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1})^T d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1} + \lambda\ I_d\right)\delta\mathbf{z}^{t,k} = -(d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1})^T\tilde{G}^{t,k}, \quad (29)$$

$$\mathbf{z}^{t,k} = \mathbf{z}^{t,k-1} + \alpha^k\ \delta\mathbf{z}^{t,k}, \quad (30)$$

and as for the manifold LSPG method, the Jacobian matrix $d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1}$ is numerically approximated at each optimization step in the implementations.

Remark 7 *(Submesh needed to define the hyper-reduced differential operators)* To compute $\tilde{G}_{h,\delta t}$ in the nodes/magic points of the reduced over-collocation method, some adjacent degrees of freedom are needed by the discrete differential operators involved. So, actually, at each time step not only the values of the state variables at the magic points are needed, but also at the adjacent degrees of freedom in the mesh with possible overlappings. We represent the restriction to this submesh of magic points and adjacent degrees of freedom with the projector $P^s_{r_h} \in \mathbb{R}^{s_h\times d}$, where s_h is the number of degrees of freedom of the submesh. Equation (28) becomes

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \| P_{r_h} G_{h,\delta t}(\boldsymbol{\mu},\ P^s_{r_h}(\phi(\mathbf{z})),\ \{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}. \quad (31)$$

The stencil around each magic point to consider depends on the type of numerical scheme. Since we are using the finite volume method we have to consider the degrees of freedom of the adjacent cells. For example, for Cartesian grids, the schemes chosen for the 2d non-linear conservation law have a stencil of 1 layer of adjacent cells, 4 additional nodes in total for a cell of the interior of the mesh. The 2d shallow water equations case requires a stencil of 2 layers instead, 12 additional nodes in total for a cell of the interior of the mesh. The two cases are shown in Fig. 2.

3.3 Over-Collocation Nodes Selection

The nodes/magic points of the over-collocation hyper-reduction method should be defined such that

$$P_{r_h} = \underset{P^TP\in\mathcal{S}}{\operatorname{argmin}} \max_{(\mathbf{z}^t,\{\mathbf{z}^s\}_{s\in I_t})\in\mathcal{T}} \|G_{h,\delta t}(\boldsymbol{\mu},(\phi(\mathbf{z})),\{(\phi(\mathbf{z}^s))\}_{s\in I_t}) - P^TPG_{h,\delta t}(\boldsymbol{\mu},P(\phi(\mathbf{z}^t)),\{P(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}} \quad (32)$$

where $\mathcal{S} = \{P\in\mathbb{R}^{r_h\times d} \mid P = (\mathbf{e}_{i_1}|\ldots|\mathbf{e}_{i_{r_h}})^T\}$ is the space of projectors onto r_h coordinates associated to the standard basis $\{\mathbf{e}_i\}_{i\in\{1,\ldots,d\}}$ of $\mathbb{R}^d$ and $\mathcal{T}$ is the space of discrete solution trajectories varying with respect to $\boldsymbol{\mu}$ and the intermediate optimization steps

$$\mathcal{T} = \{(\boldsymbol{\mu},t,k,\mathbf{z}^{t,k},\{\mathbf{z}^{s,k}\}_{s\in I_t})\in\mathcal{P}\times V_{h,\boldsymbol{\mu}}\times V_{h,\boldsymbol{\mu},t}\times\mathbb{R}^d\times\mathbb{R}^{d\times|I_t|}\mid$$
$$\left((dG^{t,k-1}d\phi^{t,k-1})^T dG^{t,k-1}d\phi^{t,k-1} + \lambda\ I_d\right)\delta\mathbf{z}^{t,k} = -(dG^{t,k-1}d\phi^{t,k-1})^T G^{t,k}\},$$

where $V_{h,\boldsymbol{\mu}}$ is the discrete space of time instants, possibly depending on h and the parameter $\boldsymbol{\mu}$, and $V_{h,\boldsymbol{\mu},t}$ is the discrete space of optimization steps at time t. Essentially we want that the nodes/magic points approximate the residuals among all time steps, optimization steps and parameter instances.

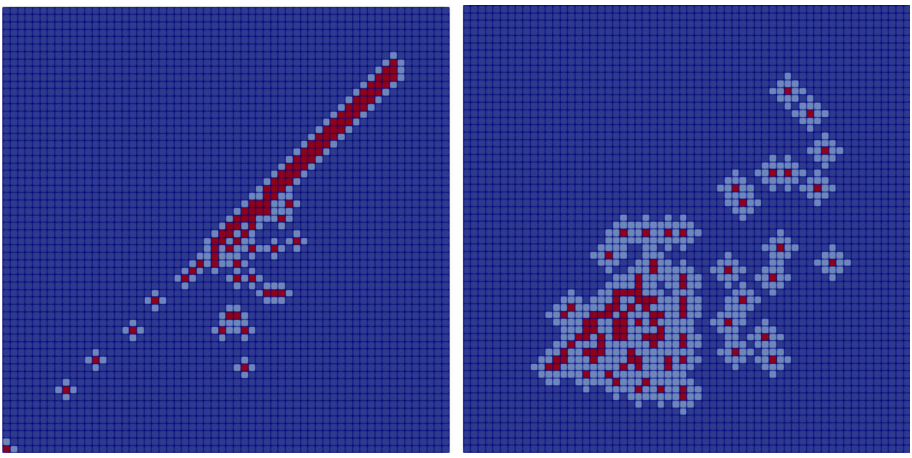

Fig. 2 Left: 100 magic points of the 2d non-linear conservation law test case. Right: 100 magic points of the 2d shallow water test case. The magic points are represented in red, the stencils of the cells and associated degrees of freedom needed for the evaluation of the discrete differential operators are in light-blue. The discarded nodes in the evolution of the dynamics with NM-LSPG-ROC are in blue. The stencil is made by 1 layer of cells in the NCL case and 2 layers in the SW case (Color figure online)

There are many possible algorithms to solve Eq. (32) for P_{r_h}. Usually they are not optimal and compromise between computational cost and accuracy, depending on the problem at hand. Among others, these algorithms are part of the hyper-reduction methods such as empirical interpolation method [45], discrete empirical interpolation method [46], Gauss–Newton tensor approximation (GNAT) [47], space-time GNAT [48] and solution-based non-linear subspace GNAT [49] (SNS-GNAT).

In particular, if $\mathcal{G}_h(\boldsymbol{\mu}, \mathbf{U}) = \dot{\mathbf{U}} - \mathbf{f}(\boldsymbol{\mu}, \mathbf{U})$, then, following some considerations that justify SNS-GNAT [49], the training modes employed to find the nodes/magic points of the over-collocation method are represented by the state snapshots instead of the residual fields. We could in principle use the reduced fields $\boldsymbol{\phi}(\mathbf{z})$ but in practice, for the test case we considered, the full-order state snapshots were enough, without even saving the intermediate optimization states.

The procedure is applied at the same time for all the components of the state field $\mathbf{U} \in X_h$ and follows a greedy approach. We remark that the spatial discretization must not vary, so that the degrees of freedom correspond to the same spatial and physical quantity over time and for every parameter instance. Some approaches tackle also geometry deformations, but keeping the same number of degrees of freedom in a reference system [50].

The Algorithm 1 is an adaptation of GNAT from Algorithm 3 in [6] to the simpler case in which the Jacobian matrix is not considered in the hyper-reduction (since in the LM method the Jacobian matrix is approximated with finite differences from the residual, see Eq. (30)). Also, with respect to Algorithm 3 in [6], the new node/magic point at line 21 in Algorithm 1 is found without computing also the reconstruction error of the degrees of freedom associated to its stencil.

Remark 8 *(Comparison between GNAT and reduced over-collocation)* We must say that the GNAT method, employed for the implementation of NM-LSPG with shallow masked autoencoders in [5], is a generalization of the reduced over-collocation method. In some

Algorithm 1 Greedy nodes/magic points evaluation for ROC method.

Input:
$\mathcal{U}_{\text{train}} := \{\mathbf{U}_{\boldsymbol{\mu},t}\}_{\mu\in\mathcal{P}_{\text{train}},\, t\in V_{\boldsymbol{\mu},h}}$, training state fields,
$n_{r_{\text{init}}}$ nodes/magic points at the boundaries,
n_{r_h}, number of nodes/magic points,
N_{modes} number of training modes.
Output:
r_h nodes/magic points used to define P_{r_h}.

Extract N_{modes} modes $\{\phi^i_{\mathbf{U}}\}_{i\in\{1,\ldots,N_{\text{modes}}\}}$ from the SVD of $\mathcal{U}_{\text{train}}$.
Compute the additional number of nodes to sample $n_a = n_{r_h} - n_{r_{\text{init}}}$.
Initialize the counter for the number of working basis vectors used: $n_b = 0$.
Set the number of greedy iterations to perform: $n_{it} = \min(n_c, n_a)$.
Compute the minimum number of working basis vectors per iteration: $n_{\text{ci,min}} = \text{floor}(n_c/n_{it})$.
Compute the minimum number of sample nodes to add per iteration: $n_{\text{ai,min}} = \text{floor}(n_a/n_c)$.
for $i = 1, \ldots, n_{it}$ (greedy iteration loop)
  Compute the number of basis vectors for this iteration: $n_{ci} = n_{\text{ci,min}}$;
  if $i \le n_c \bmod n_{it}$
    then $n_{ci} = n_{ci} + 1$.
  Compute the number of samples nodes to add: $n_{ai} = n_{\text{ci,min}}$;
  if $i \le n_a \bmod n_c$
    then $n_{ai} = n_{ai} + 1$
  if $i = 1$ then
    $\left[\mathbf{U}^1 \cdots \mathbf{U}^{n_{ci}}\right] = \left[\phi^1_{\mathbf{U}} \cdots \phi^{n_{ci}}_{\mathbf{U}}\right]$,
    Build the intermediate P from the $n_{r_{\text{init}}}$ initial magic points.
  else
    for $q = 1, \ldots, n_{ci}$ (basis vector loop)
      Compute $\alpha = \text{argmin}_{\gamma\in\mathbb{R}^{n_b}} \| \left[P\phi^1_{\mathbf{U}} \cdots P\phi^{n_b}_{\mathbf{U}}\right]\gamma - P\phi^{n_b+q}_{\mathbf{U}} \|_2$,
      $\mathbf{U}^q = \phi^{n_q+q}_{\mathbf{U}} - \left[\phi^1_{\mathbf{U}} \cdots \phi^{n_b}_{\mathbf{U}}\right]\alpha$
  for $j = 1, \ldots, n_{ai}$ (sample node loop)
    Find the node/magic point n that maximizes the state reconstruction error among the nodes not selected yet: $n = \text{argmax} \sum_{q=1}^{n_{ci}} (U^q)^2$.
  Update P with the newly found node/magic point.
  $n_b = n_b + n_{ci}$.

cases though, they may perform similarly. For $P \in \mathcal{S}$, let us define

$$\mathbf{r} = r_{t,\boldsymbol{\mu},k} = G_{h,\delta t}(\boldsymbol{\mu},\, P(\phi(\mathbf{z}^{t,k})),\, \{P(\phi(\mathbf{z}^{s,k-1}))\}_{s\in I}),\ \forall (t,k,\boldsymbol{\mu}) \in V_{h,\boldsymbol{\mu}} \times V_{h,\boldsymbol{\mu},t} \times \mathcal{P},$$

and the GNAT projection operator $\mathbb{P} = \Phi(P\Phi)^\dagger P$, where Φ is the matrix where the columns correspond to a chosen basis (it could be the FOM residual snapshots, ROM residual snapshots, ROM Jacobian and residual snapshots, see [47]). In particular, if $\Phi = P^T_{r_h} = P^T$ we have that

$$\mathbb{P} = \Phi(P_{r_h}\Phi)^\dagger P_{r_h} = \Phi(P_{r_h}\Phi)^{-1} P_{r_h} = P^T_{r_h}(P_{r_h}P^T_{r_h})^{-1} P_{r_h} = P^T_{r_h} P_{r_h}, \tag{33}$$

that is the reduced over-collocation projection in the chosen nodes/magic points and extended to 0 in the remaining degrees of freedom. In this sense, the GNAT method includes the reduced over-collocation one.

However, for the class of problems we are considering, the GNAT method suffers from the slow decaying KnW. We have the inequalities

$$
\begin{aligned}
d_n^2(\{r_{t,\boldsymbol{\mu},k}\}_{t,\boldsymbol{\mu},k}) &= \inf_{\dim V_n = n} \max_{t,\boldsymbol{\mu},k} \|\mathbf{r} - V_n V_n^T \mathbf{r}\|_2^2 \\
&\leq \inf_{\dim V_n = n} \max_{t,\boldsymbol{\mu},k} \|\mathbf{r} - V_n V_n^T \mathbf{r}\|_2^2 + \|V_n V_n^T \mathbf{r} - \mathbb{P}\mathbf{r}\|_2^2 \\
&= \inf_{\dim V_n = n} \max_{t,\boldsymbol{\mu},k} \|\mathbf{r} - \mathbb{P}\mathbf{r}\|_2^2 = \inf_{\dim V_n = n} \max_{t,\boldsymbol{\mu},k} \|(I - \mathbb{P})(I - V_n V_n^T)\mathbf{r}\|_2^2 \\
&\leq \inf_{\dim V_n = n} \max_{t,\boldsymbol{\mu},k} \|(I - \mathbb{P})\|_2^2 \|(I - V_n V_n^T)\mathbf{r}\|_2^2 \\
&= \inf_{\dim V_n = n} \max_{t,\boldsymbol{\mu},k} \|\mathbb{P}\|_2^2 \|(I - V_n V_n^T)\mathbf{r}\|_2^2,
\end{aligned}
$$

where $(t, k, \boldsymbol{\mu}) \in V_{h,\boldsymbol{\mu}} \times V_{h,\boldsymbol{\mu},t} \times \mathcal{P}$ whenever the maximum is taken. The term $\mathbf{r} - \mathbb{P}\mathbf{r}$ is the GNAT approximation error. The rightmost term is the usual bound on the hyper-reduction error [46], where it was used the fact that $\mathbb{P} = I - \mathbb{P}$. The second equality is valid because $\mathbf{r} - V_n V_n^T \mathbf{r}$ and $V_n V_n^T \mathbf{r} - \mathbb{P}\mathbf{r}$ are orthogonal. The third equality is obtained from the relations

$$
\begin{aligned}
& \mathbf{r} = (\mathbf{r} - \mathbf{r}_*) + \mathbf{r}_* = \mathbf{w} + \mathbf{r}_*, \qquad \text{with } \mathbf{r}_* := V_n V_n^T \mathbf{r}, \\
\Rightarrow \quad & \mathbf{r} - \mathbb{P}\mathbf{r} = \mathbf{w} + \mathbf{r}_* - \mathbb{P}\mathbf{w} - \mathbb{P}\mathbf{r}_* = \mathbf{w} - \mathbb{P}\mathbf{w},
\end{aligned}
$$

where the last equality follows from $\mathbb{P}\mathbf{r}_* = \mathbf{r}_*$. Here we have supposed the nodes/magic points to be independent of V_n. So in the case of slow decaying Kolmogorov n-width, minimizing the hyper-reduced residual $\mathbb{P}\mathbf{r} \in < V_n > \subset \mathbb{R}^d$ is less efficient due to the slow convergence in n of the best approximation error $\|\mathbf{r} - V_n V_n^T \mathbf{r}\|_2^2$. This is one of the reasons why we employed ROC for the SWE test case; for the NCL test case GNAT and ROC performed similarly.

3.4 Compressed Decoder Teacher–Student Training

In order to make the whole methodology independent of the number of degrees of freedom, the decoder has to be substituted with a map $\tilde{\phi} : \mathbb{R}^R \to P_{r_h}(X_h) \subset \mathbb{R}^{r_h}$ from the latent space to the space of discrete full-order solutions evaluated only at the submesh containing the magic points and the needed adjacent degrees of freedom. As architecture, we choose a feedforward neural network (FNN) with one hidden layer, but actually the only requirement is that the computational cost is low enough such that not only a theoretical dimension reduction is achieved, but also a speed-up is reached.

In the literature, the procedure for the training of the compressed decoder $\tilde{\phi}$ from the decoder ϕ is called teacher–student training [51]. In principle, the compressed decoder can be composed of layers inherited by the original decoder, such that the learning process involves only the final new additional layers. In our case, we preferred to train the compressed decoder anew: the latent projections of the training snapshots with the encoder $\psi(\mathcal{U}_{\text{train}}) = \{z_i\}_{i=1}^{N_{\text{train}}}$ are the inputs and the restriction of the snapshots to the submesh $P_{r_h}^s(\mathcal{U}_{\text{train}}) = \{\tilde{U}_i\}_{i=1}^{N_{\text{train}}}$ are the targets, see Eq. (31). A schematic representation of the teacher–student training is represented in Fig. 3. Moreover, to speed up the offline stage, we use the training FOM snapshots restricted to the magic points as training outputs for the teacher–student training, while usually the reconstructed snapshots from the CAE -that would additionally need to be computed- are employed.

Again, for the training, we use a relative mean square loss with an additional regularizing term

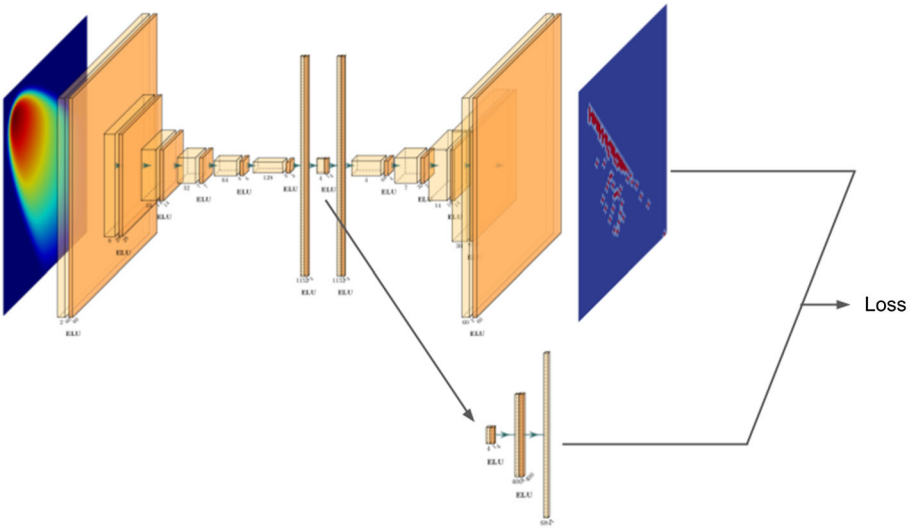

Fig. 3 Teacher–student training of the compressed decoder for the 2d non-linear conservation law test case. The magic points, which the snapshots are restricted to, are shown in red over the domain (Color figure online)

$$\mathcal{L}(\{z_i\}_{i=1}^b; \tilde{\Theta}) = \frac{1}{b}\sum_{i=1}^{b} \frac{\|\tilde{U}_i - \tilde{\phi}(z_i)\|_2^2}{\|\tilde{U}_i\|_2^2} + \lambda_1 \|\tilde{\Theta}\|_2^2, \tag{34}$$

where $\tilde{\Theta}$ are the weights of the compressed decoder.

Remark 9 *(Jacobian evaluation in Levenberg–Marquardt algorithm)* The main reason why finite differences approximations of Jacobians are implemented in the NM-LSPG case, as explained at the end of Sect. 3.1, is that the computational cost of evaluating the Jacobian of the full decoder is too high. In principle Jacobian evaluations of the compressed decoder are cheaper and could be employed, instead of relying again on finite differences approximations.

Remark 10 *(Shallow masked autoencoders)* We are motivated to write this article to extend the results in [5] to a generic architecture composed by neural networks. They performed the hyper-reduction of the non-linear manifold method [4] with a shallow masked autoencoder, so that correctly masking the weights matrices of the decoder, its outputs correspond only to the submesh needed by the GNAT method, thus eliminating the dependence on the FOM's degrees of freedom. We want to reproduce, in some sense, this approach for an arbitrary autoencoder architecture, in this case a CAE, in order to tackle with the latest architectures developed in the literature the problem of solution manifold approximability: we think this is a major concern when trying to apply non-linear MOR to real applications. In fact, as will be clear in the numerical results Sect. 4, the reconstruction error of the autoencoder bounds from below the prediction error of our newly developed ROMs.

It can be seen that the new model order reduction is composed of two distinct procedures to achieve the independence on the number of degrees of freedom: first the residual from NM-LSPG in Eq. (16) is hyper-reduced with ROC in Eq. (26) and secondly the CAE's decoder is compressed with teacher–student training. In principle, we could substitute the use of the compressed decoder with the restriction of the final layer of the CAE's decoder into the

magic points, while keeping the hyper-reduction with ROC of the residual. In this case, the whole methodology would still be dependent on the total number of degrees of freedom, but in practice a CAE's decoder forward is relatively cheap compared to the evaluation of the full residual. So, the hyper-reduction performed with ROC or GNAT only at the equations/residuals level, is already beneficial to reduce the computational cost. We will compare this variant of the NM-LSPG-ROC method with the one that employs the compressed decoder, also to verify the consistency of the teacher–student training that is omitted in the first case.

In the numerical results Sect. 4 we will adopt the acronym NM-LSPG-ROC-TS or NM-LSPG-GNAT-TS for the method that employs the compressed decoder and NM-LSPG-ROC or NM-LSPG-GNAT for the method that performs the hyper-reduction only at the equations/residuals level.

4 Numerical Results

We test the new methodology on two benchmarks with a relatively slow KnW: the first model is governed by a non-linear conservation law (Sect. 4.1) the second by the shallow water equations (Sect. 4.2). Both are parametric, non-linear and time-dependent, and the only other (non-temporal) parameter is a multiplicative constant of the initial condition. The mesh employed is the same: a 60×60 structured orthogonal grid.

All the CFD simulations are obtained by the use of an in-house open source library ITHACA-FV (In real Time Highly Advanced Computational Applications for Finite Volumes) [52], developed in a finite volume environment based on the open-source library OpenFOAM [53]. Regarding the implementation of the convolutional autoencoders and compressed decoders (CAE) we used libtorch, PyTorch C++ frontend, while for the training of the long-short term memory network (LSTM) we used PyTorch [36]. All the CFD simulations were performed on a Intel(R) Core(TM) i7-8750H CPU with 2.20GHz and all the neural networks trainings on a GeForce GTX 1060 GPU. Further reductions in the computational costs could be achieved exploiting the parallel implementation of the training procedures in PyTorch. The details of the architectures of the neural networks that will be employed are reported in the Appendix A.

The following notations are introduced: N^{μ}_{train}, N^{μ}_{test} are the numbers of train and test parameters, respectively; N^{t}_{train}, N^{t}_{test} are the number of time instances associated to the train and test parameters, respectively. The total number of training and test snapshots is thus $N_{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}}$, and $N_{\text{test}} = N^{\mu}_{\text{test}} \cdot N^{t}_{\text{test}}$, respectively.

The accuracy of the reduced-order models devised is measured with the mean realtive L^2-error and the maximum relative L^2-error, where the mean and max are taken with respect to the time scale: since the test cases depend on a non-temporal parameter, for each instance of these parameters a time-series corresponding to the discrete dynamics is associated; the mean and maximum are evaluated w.r.t the elements of these time-series. Let $\{u_{\mu}^{t_i}\}_{i=1,\dots N^t}$ and $\{U_{\mu}^{t_i}\}_{i=1,\dots N^t}$ be the predicted and true time-series N^t elements long, associated to the train or test parameter μ, the mean relative L^2-errors and maximum relative L^2-errors are then defined as

$$\epsilon_{mean}(u_\mu, U_\mu) = \frac{1}{N^t} \sum_{i=1}^{N^t} \frac{\| u_{\mu}^{t_i} - U_{\mu}^{t_i} \|_{L^2}}{\| U_{\mu}^{t_i} \|_{L^2}}, \quad \epsilon_{max}(u_\mu, U_\mu) = \max_{i=1,\dots,N^t} \frac{\| u_{\mu}^{t_i} - U_{\mu}^{t_i} \|_{L^2}}{\| U_{\mu}^{t_i} \|_{L^2}}. \tag{35}$$

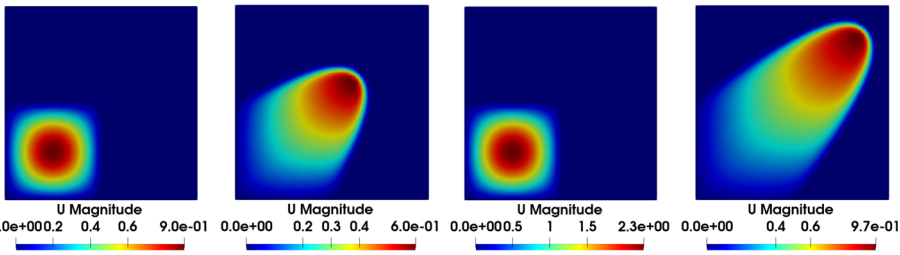

Fig. 4 From left to right: FOM solution of Eq. (36) at $(t, \mu) \in \{(0, 0.8), (2, 0.8), (0, 2), (2, 2)\}$ (Color figure online)

Remark 11 *(Levenberg–Marquardt parameters)* Regarding the Levenberg–Marqurdt non-linear optimization algorithm, we remark that we approximate the Jacobians with forward finite differences, and the optimization process, for each time step, is stopped when the maximum number of residual evaluations is reached. This number is set to 7, including the evaluations related to the Jacobian computations. When 7 residual evaluations are not enough for the method to converge, it is explicitly reported.

4.1 Non-linear Conservation Law (NCL)

We test our procedure for non-linear model order reduction on a 2d non-linear conservation law model (NCL). Two main reasons are behind this choice: the slow Kolmogorov n-width decay of the continuous solution manifold, and the possibility to compare our results with a similar test case realized with an implementation of non-linear manifold based on shallow masked autoencoders and GNAT [5].

The parametrization affects the initial velocity as a scalar multiplicative constant $\mu \in [0.8, 2]$:

$$\begin{cases} \partial_t \mathbf{u} + \frac{1}{2}\nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \nu \Delta \mathbf{u} & (\mathbf{x}, t) \in [0, 1]^2 \times [0, 2], \\ \mathbf{u}(\mathbf{x}, 0) = 0.8 \cdot \mu \cdot \sin(2\pi x) \sin(2\pi y) \chi_{[0,0.5]^2} & \mathbf{x} \in [0, 1]^2, \\ \mathbf{u}(\mathbf{x}, t) = 0 & (\mathbf{x}, t) \in \partial[0, 1]^2 \times [0, 2], \end{cases} \tag{36}$$

where the viscosity $\nu = 0.0001$. We will collect $N^{\mu}_{\text{train}} = 12$ equispaced training parameters from the range $\mu \in [0.8, 2]$ and $N^{\mu}_{\text{test}} = 16$ equispaced test parameters from the range $\mu \in [0.6, 2.2]$. The first two and the last two parameters will account for the extrapolation error. The time step is equal to $\Delta t = 1\text{e}{-3}$ s, but the training snapshots are collected every 4 time steps and the test snapshots every 20, thus $N^{t}_{\text{train}} = 501$, and $N^{t}_{\text{test}} = 101$. In the predictive online phase, the dynamics will be evolved with the same time step $\Delta t = 1\text{e}{-3}$. For easiness of representation, the train parameters are labelled from 1 to 12, and the test parameters are labelled from 1 to 16.

To have a qualitative view on the range of the solution manifold, we report the initial and final time snapshots for the extremal training parameters of the range $\mu \in [0.8, 2]$, in Fig. 4.

In this test case the GNAT method performed slightly better than the ROC method for hyper-reduction, so we employed the former to obtain the results shown.

4.1.1 Full-Order Model

We solve the 2d non-linear conservation law for different values of the parameter μ with OpenFoam [53] open-source software for CFD. We employ the finite volumes method (FVM) in a structured orthogonal grid of 60×60 cells. If we represent with M the mass matrix, with D the diffusive matrix term, and with $C(U^{t-1})$ the advection matrix, then, at every time instant t, the discrete equation

$$\frac{M}{\Delta t}U^t + C(U^{t-1})U^t - \nu DU^t = \frac{M}{\Delta t}U^{t-1}, \tag{37}$$

is solved for the state U^t with a semi-implicit Euler method. The time step is 1e−3, the initial and final time instants are 0 and 2 s. The linear system is solved with the iterative method BiCGStab preconditioned with DILU, until a tolerance of 1e−17 on the FVM residual is reached.

The stencil of the numerical scheme at each cell involves the adjacent cells that share an interface (4 for an interior cell, 3 for a boundary cell and 2 for a corner cell): the value of the state at the interfaces is obtained with the bounded upwind method for the advection term and the surface normal gradient is obtained with central finite differences of two adjacent cell centers. So, in order to implement the reduced over-collocation method, for each node/magic point we have to consider an additional number of maximum 4 cells, that is 8 degrees of freedom to keep track of during the evolution of the latent dynamics; of course in practice they may overlap reducing the computational cost further.

The residual of the NM-LPSG methods is evaluated with the same numerical scheme of the FOM. In the SWE test case the FOM and the ROMs employ different numerical schemes (Sect. 4.2).

4.1.2 Manifold Learning

As first step of the procedure the discrete solution manifold is learned through the training of a convolutional autoencoder (CAE) whose specific architecture is reported in Table 7. The CAE is trained with the ADAM [37] stochastic optimization algoritm for 2000 epochs, halving the learning rate by a factor of 2 if after 200 epochs the loss does not decrease. The initial learning rate is 1e−3, its lower bound is 1e−6. The number of training snapshots is $N_{\text{train}} = 12 \times 501 = 6012$, the batch size 20. It could be further refined in order the increase the efficiency of the whole procedure.

We choose as latent dimension 4, 2 dimensions greater than the number of parameters (the scalar multiplying the initial condition and time). We don't perform a convergence study of the accuracy with respect to the latent dimension since our focus is on the implementation of the NM-LSPG-ROC and NM-LSPG-ROC-TS model-order reduction methods: we are satisfied as long as the accuracy is relatively high, while the reduced dimension corresponds to an inaccurate linear approximating manifold spanned by the same number of POD modes.

In Fig. 5 is shown the reconstruction error of the CAE and its decay with respect to the number of POD modes chosen [4, 10, 25, 50, 100]. To reach the same accuracy of the CAE with latent dimension 4, around 50 POD modes are needed. In order to state that the slow KnW decay problem is overcome by the CAE, the asymptotic convergence of the reconstruction error w.r.t. the latent dimension should be studied as was done for similar problems in [4, 5]. Instead, we will empirically prove that we can devise an hyper-reduced ROM with latent dimension 4 and accuracy lower than the 2% for the mean relative $L2$-error, a task that

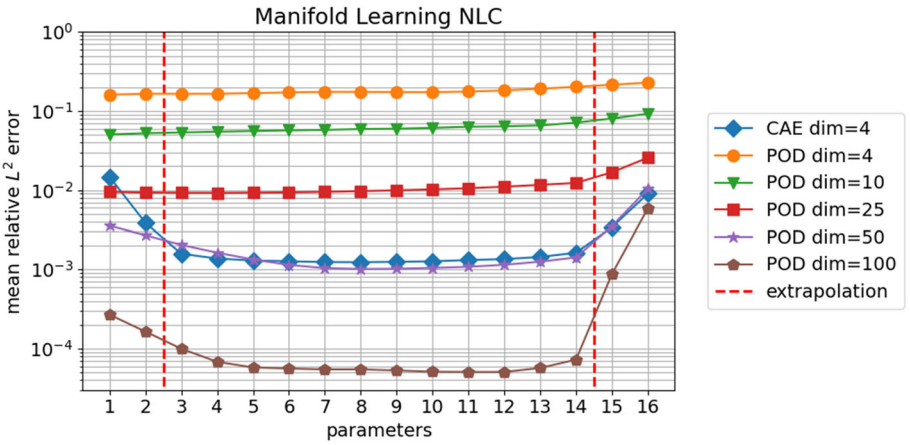

Fig. 5 Comparison between the CAE's and POD's projection errors of the discrete solution manifold represented by the $N_{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}} = 12 \cdot 501$ training snapshots. The error is evaluated on the 16 test parameters, each associated with a time series of $N^{t}_{\text{test}} = 101$, for a total of $N_{\text{test}} = N^{\mu}_{\text{test}} \cdot N^{t}_{\text{test}} = 16 \cdot 101 = 1616$ test snapshots. The mean is performed over the time scale (Color figure online)

would be impossible for a POD based ROM with the same reduced dimension, since the reconstruction error is near 20% for all the test parameters.

Without imposing any additional inductive bias a part from the regularization term in the loss from Eq. (12) and the positiveness of the velocity components, the latent trajectories reported in Fig. 6 for the odd parameters of the test set, are qualitatively smooth. An important detail to observe is that the initial conditions are well separated one from another in the latent space, see Remark 3, and that the dynamics is non-linear. It also can be noticed that the 2 extremal parameters, corresponding to the extrapolation regime and represented in the plot by the two most outer trajectories that enclose the other 6, have smooth latent dynamics analogously to the others even though the reconstruction error starts degrading, as can be seen from Fig. 5.

4.1.3 Hyper-Reduction and Teacher–Student Training

The selection of the magic points is carried out with the greedy Algorithm 1. The FOM snapshots employed correspond to the training parameters 1 and 12, but are sampled every 10 time step instead of every 4, as for the training snapshots of the CAE.

We perform a convergence study increasing the number of magic points from 50 to 100 and 150. The corresponding submesh sizes, i.e. the number of cells involved in the discretization of the residuals, are reported in the Table 1. The submesh size is bounded above with the total number of the cells in the mesh, that is 3600.

After the computation of the magic points, the FOM snapshots are restricted to those cells and employed as training outputs of the compressed decoder, as described in Sect. 3.4. The actual dimension of the outputs is twice the submesh size, since for each cell there are 2 degrees of freedom corresponding to the velocity components. The inputs are the 4-dimensional latent coordinates of the encoded FOM training snapshots, for a total of 6012 training input–output pairs. The architecture of the compressed decoder is a feedforward neural network (FNN) with one hidden layer, whose number of nodes is reported in Table 1,

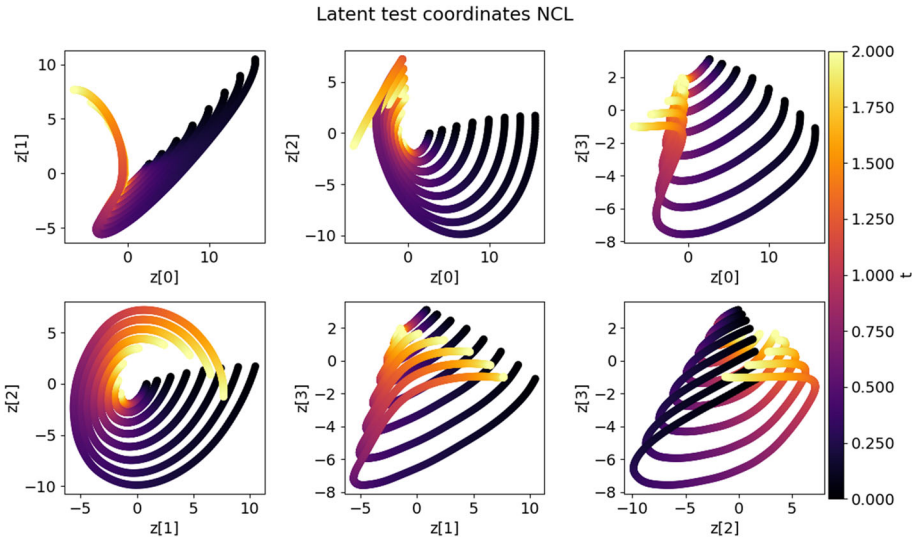

Fig. 6 Correlations among the 4 latent coordinates of the odd parameters of the test set, 8 in total. They are obtained projecting the test snapshots into the latent space of the CAE with the encoder. The coloring corresponds to the time instants from 0 to 2 (Color figure online)

Table 1 In this table are reported for the NCL test case: the submesh size and the hidden layer (HL) number of nodes of the compressed decoder; the Teacher–Student training (TS) duration in seconds (TS total epochs), the Teacher–Student training average epoch duration in seconds (TS avg epoch); the average time step for NM-LSPG-GNAT with compressed decoder (avg GNAT-TS) in milliseconds, and the average time step for NM-LSPG-GNAT with the hyper-reduced residuals but full CAE decoder (avg GNAT-no-TS) in milliseconds

MP	Submesh size	HL size	TS total epochs (s)
50	139	300	1682
100	246	350	2482
150	335	400	4245
MP	TS avg epoch (s)	Avg GNAT-TS (ms)	Avg GNAT-no-TS (ms)
50	0.560	1.88	4.496
100	0.827	**4.599**	4.499
150	1.415	2.318	4.208

In bold, the results are obtained with 13 maximum residual evaluations of the Levenberg–Marquardt algorithm, instead of the fixed 7, see Remark 11

under ‘HL size’. The compressed decoders architecture’s specifics are also summarized in Table 7.

Each compressed decoder is trained for 3000 epochs, with an initial learning rate of 1e−4, that halves if the loss from Eq. (34) does not decrease after 200 epochs. The batch size is 20. The duration of the training is reported in Table 1 under ‘TS total epochs’, that stands for Teacher–Student training total epochs, along with the average cost for an epoch, under ‘TS avg epoch’. The accuracy of the predictions on the test snapshots restricted to the magic points is assessed in Fig. 7.

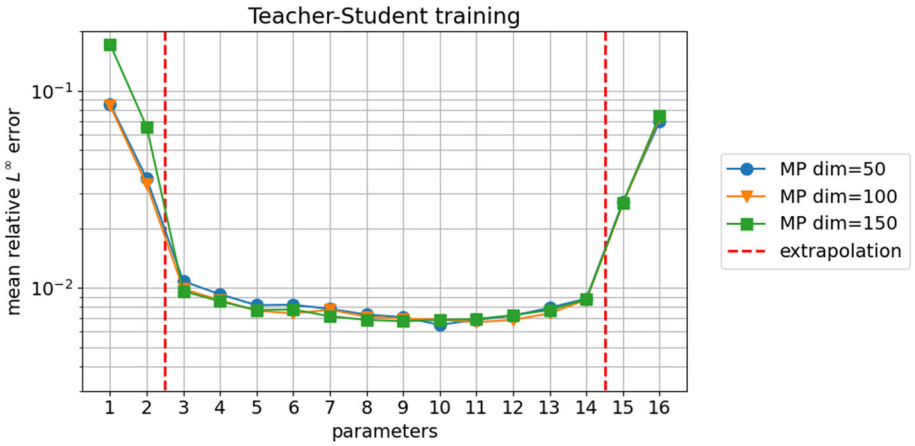

Fig. 7 Prediction accuracy on the test snapshots restricted to the magic points. The accuracy is measured with the relative L^∞-error averaged over the time trajectories associated to each one of the $N^\mu_{\text{test}} = 16$ test samples (Color figure online)

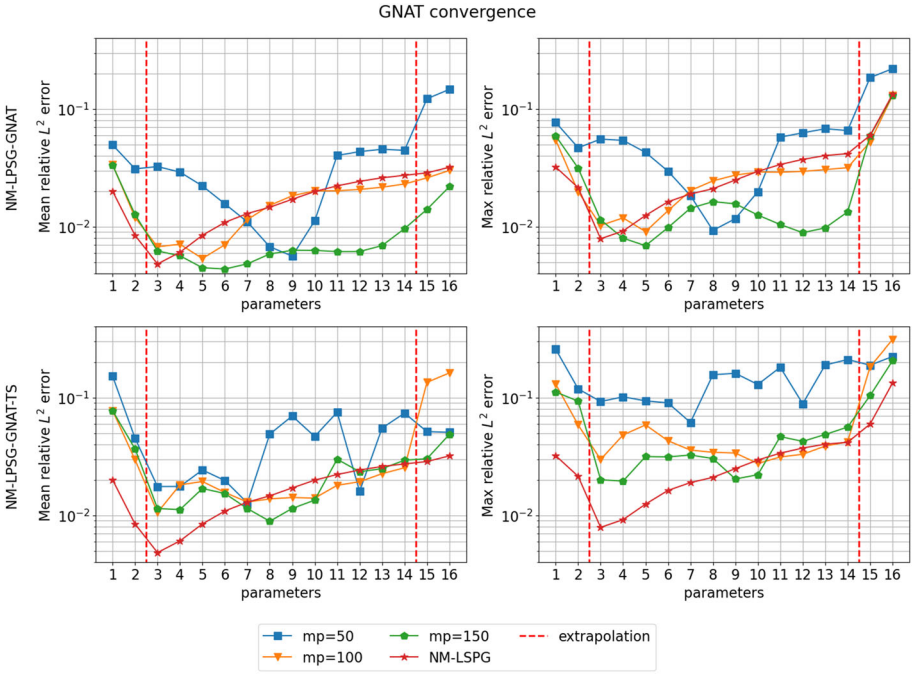

Fig. 8 First row: NM-LSPG-GNAT mean and max relative L^2-error. Second row: NM-LSPG-GNAT-TS mean and max relative L^2-error. In red also the NM-LSPG accuracy is reported. The mean and max values are evaluated with respect to the time series of intermediate solutions associated to each one of the $N^\mu_{\text{test}} = 16$ test parameters (Color figure online)

The convergence with respect to the number of magic points is shown in Fig. 8. Since, especially for the NM-LSPG-GNAT-TS reduced-order model, the relative L^2-error is not uniform along the time scale, we report both the mean and max relative L^2-errors over the time series associated to each one of the 16 test parameters.

From Fig. 8 and Table 1 it can be seen that NM-LPSG-GNAT is more accurate than NM-LSPG-GNAT-TS even tough computationally more costly in the online stage. We underline that in the offline stage NM-LSPG-GNAT-TS requires the training of the compressed decoder. However, this could be performed at the same time of the CAE training, see the discussion Sect. 5. The NM-LSPG-GNAT reduced-order model achieves better results also in the extrapolation error, sometimes even lower than the NM-LSPG method: this remains true even when increasing the maximum residual evaluations of the LM algorithm, and it may be related to the nonlinearity of the decoder that introduces difficult to interpret correlations of the latent dynamics with the output solutions restricted to the magic points.

All the simulations of NM-LSPG, NM-LSPG-GNAT-TS and NM-LSPG-GNAT methods converge with a maximum of 7 residual evaluations of the LM algorithm, for all magic points reported, that is 50, 100, and 150 and for all the 16 test parameters. However, 3 test parameters could not converge for the method NM-LSPG-GNAT-TS with 100 magic points, so we increased the maximum function evaluations to 13 for all the test points. The higher computational cost per time step is shown in Table 1. Apart from those 3 test points not converging, the accuracy remains the same for the other 13 test parameters, so we have chosen to report the results in the case of 13 residual evaluations for all the 16 test parameters in Fig. 8.

4.1.4 Comparison with Data-Driven Predictions Based on a LSTM

To assess the quality of the reduced-order models devised, we compare the accuracy in the training and extrapolation regimes, and the computational cost of the offline and online stages with a purely data-driven ROM in which the solutions manifold is approximated by the same CAE, but the dynamics is evolved in time with a LSTM neural network. The architecture of the LSTM employed is reported in Table 9. The results are summarized in Fig. 9, the computational costs in Table 2.

The LSTM is trained for 10,000 epochs with the ADAM stochastic optimization algorithm and an initial learning rate of 0.001, halved if after 500 epochs the loss does not decrease. The time series used for the training are the same $N_{\text{train}} = 6012$ training snapshots employed for the CAE. We remark that the LSTM cannot approximate the dynamics for an arbitrary time step, but it is fixed, depending on the training time step used, in this case 0.004 s.

The offline stage's computational cost is determined by the heavy CAE training for both the procedures, see the Discussion Sect. 5 for possible remedies. The LSTM-NN achives a speed-up close to 3 with respect to the FOM, differently from the NM-LSPG-GNAT and NM-LSPG-GNAT-TS methods. However, since the models are hyper-reduced, increasing the degrees of freedom refining the mesh should increase the computational cost of the FOM and NM-LSPG methods only, with due precautions. The average cost of the evaluation of the dynamics for the LSTM model with a time step of 0.004 s is associated to the label 'avg LSTM-NN full-dynamics'; thanks to vectorization, the dynamics for all the $N^{\mu}_{\text{test}} = 16$ parameters is evaluated with a single forward of the LSTM, thus the low computational cost reported.

The CAE reconstruction error in blue in Fig. 9, lower bounds all the other models' errors. This is the reason why having a good accuracy of the CAE's solution manifold approximation is mandatory to build up non-linear manifold methods. In this sense NM-LSPG-ROC-TS and

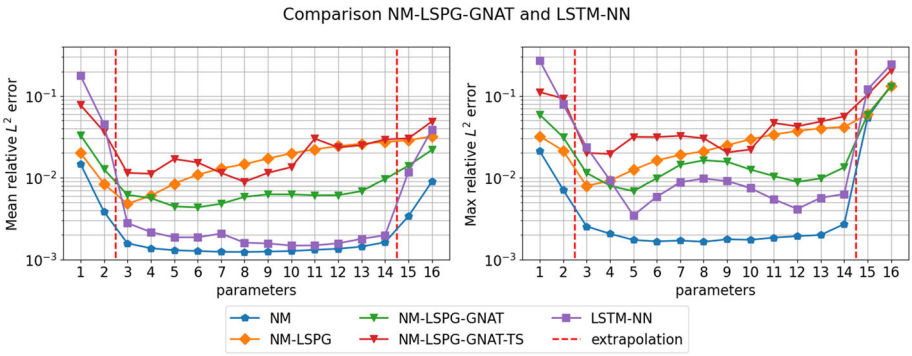

Fig. 9 Comparison of the accuracies between all the ROMs presented on the test set of $N^{\mu}_{\text{test}} = 16$ parameters, each associated to a time series of $N^{t}_{\text{test}} = 101$ intermediate solutions. The number of magic points employed for NM-LSPG-GNAT and NM-LSPG-GNAT-TS is 150. The mean and maximum relative L^2-errors are taken with respect to the time scale (Color figure online)

Table 2 Offline stage: full-order model (FOM) computation of the $N_{\text{train}} = 6012$ snapshots (FOM snapshots evaluation), average cost of a single epoch for the training of the CAE with a batch size of 20 (CAE training single epoch avg), and total cost for 3000 epochs (CAE training); average epoch's cost for the LSTM training with a batch size of 100, and total cost for 10,000 epochs

Offline stage	Time
FOM snapshots evaluation	29.04 [s]
CAE training single epoch avg	10.1 [s]
CAE training	20213.3 [s]
LSTM-NN training single epoch avg	0.139 [s]
LSTM-NN training	1365 [s]
Online stage	**Time**
Avg FOM time step	1.210 [ms]
Avg FOM full dynamics	2.42 [s]
Avg NM-LSPG time step	22.126 [ms]
Avg NM-LSPG full-dynamics	133.2 [s]
Avg LSTM-NN time step	0.432 [ms]
LSTM-NN full-dynamics	6.982 [ms]

Online stage: for the FOM, NM-LSPG and LSTM-NN models it is reported the average of a single time step cost over all the $N_{\text{test}} = 1616$ test parameters and time series, and the average cost of the full dynamics over the $N^{\mu}_{\text{test}} = 16$ test parameters. The LSTM-NN full-dynamics is evaluated with a single forward

NM-LSPG-GNAT-TS with respect to NM-LSPG-GNAT with shallow autoencoders [5] offer the possibility to choose an arbitrary architecture for the autoencoder, thus allowing a more accurate solution manifold approximation.

While in the training range from test parameter 3 to 14, the accuracy of the LSTM-NN is significantly better than NM-LSPG-GNAT and NM-LSPG-GNAT-TS models', in the extrapolation regime we observe that the predictions of the fully data-driven model degrades. The extrapolation error of the LSTM-NN model depends on the architecture chosen, regularization applied, training procedure, and hyperparameters tuning. What can be assessed from the results is that, outside the training range, the LSTM-NN's accuracy is dependent on all these factors, with sometimes a difficult interpretation of the results, while NM-LSPG-

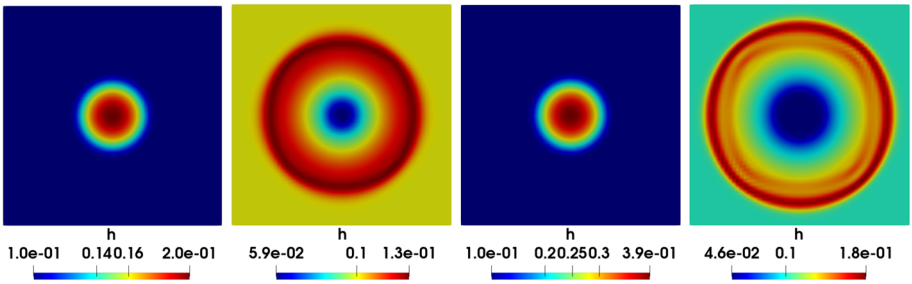

Fig. 10 From left to right: FOM solution of Eq. (38) at $(t, \mu) \in \{(0.01, 0.1), (0.2, 0.1), (0.01, 0.3), (0.2, 0.3)\}$ (Color figure online)

GNAT relies only on the number of magic points employed and the dynamics is evolved in time minimizing a physical residual directly related to the NCL model's equations.

4.2 Shallow Water Equations (SWE)

The second test case we present is a 2d non-linear, time-dependent, parametric model based on the shallow water equations (SWE). Also in this case, the non-temporal parameter affects the initial conditions, $\mu \in [0.1, 0.3]$, $t \in [0, 0.2] = I$:

$$\begin{cases} \partial_t(h\mathbf{u}) + \nabla \cdot (h\mathbf{u} \otimes \mathbf{u}) + \frac{1}{2} \mid \mathbf{g} \mid h\nabla h = 0 & (\mathbf{x}, t) \in [0, 1]^2 \times I, \\ \partial_t h + \nabla \cdot (\mathbf{u}h) = 0 & (\mathbf{x}, t) \in [0, 1]^2 \times I, \\ \mathbf{u}(\mathbf{x}, 0) = 0 & \mathbf{x} \in [0, 1]^2, \\ h(\mathbf{x}, 0) = \mu \left(\frac{1}{e^1} \cdot e^{-\frac{1}{0.04 - \|\mathbf{x} - \mathcal{O}\|_2^2}} \chi_{\|\mathbf{x}\|_2^2 < 0.2} + \chi_{\|\mathbf{x}\|_2^2 \geq 0.2} \right) & \mathbf{x} \in [0, 1]^2, \\ \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{n} = 0 & (\mathbf{x}, t) \in \partial[0, 1]^2 \times I, \\ \nabla h(\mathbf{x}, t) \cdot \mathbf{n} = 0 & (\mathbf{x}, t) \in \partial[0, 1]^2 \times I, \end{cases} \tag{38}$$

where h is the water depth, $\mathbf{u}$ is the velocity vector, $\mathbf{g}$ is the gravitational acceleration, and $\mathcal{O}$ is the point $(0.5, 0.5) \in \Omega$. We consider a constant bathymetry $h_0 = 0$, so that the free surface height $h_{\text{total}} = h + h_0$ is equal to the water depth h.

The time step that will be employed for the evolution of the dynamics of the FOM is 1e−4 s. The training and test snapshots are sampled every 4 time steps. The training non-temporal parameters are $N^{\mu}_{\text{train}} = 10$ in number, and they are sampled equispacedly in the training interval $\mu \in [0.1, 0.3]$, for a total of $N_{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}} = 10 \cdot 501 = 5010$ training snapshots.

Due to an inaccurate reconstruction error of the CAE for the first time instants, the predictions of the dynamics of the reduced model are evaluated from the time instant $t_0 = 0.01$ s. The test non-temporal parameters are $N^{\mu}_{\text{test}} = 8$ in number and sampled equispacedly in the test interval $\mu \in [0.05, 0.35]$, for a total of $N_{\text{test}} = N^{\mu}_{\text{test}} \cdot N^{t}_{\text{test}} = 8 \cdot 475 = 3800$ test snapshots, since the first 26 are cut from the time series, $N^{t}_{\text{train}} = 501$ snapshots long. Again the first 2 and the last 2 parameters correspond to the extrapolation regime. The initial latent variables are obtained projecting with the encoder into the latent space the test snapshots corresponding to the time instants $t_0 = 0.01$ instead of $t = 0$. The training and test time series are labelled from 1 to 10 and from 1 to 8 with an increasing order (Figs. 10 and 11).

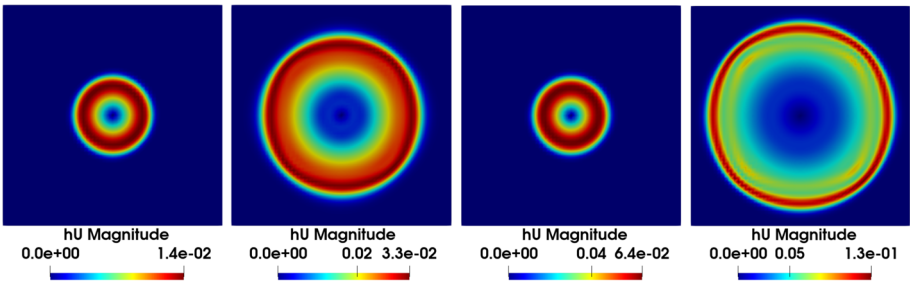

Fig. 11 From left to right: FOM solution of Eq. (38) at $(t, \mu) \in \{(.010, 0.1), (0.2, 0.1), (0.01, 0.3), (0.2, 0.3)\}$ (Color figure online)

In this test case the ROC hyper-reduction is more accurate with respect to the GNAT one, so the results are reported w.r.t. this hyper-reduction method.

4.2.1 Full-Order Model

One detail that we didn't stress in the previous test case is that the FOM and the NM-LSPG ROM can discretize the residuals of the SWE differently: only the consistency of the discretization is required, characterizing the NM-LPSG family as equations-based rather than fully intrusive.

The FOM solutions are computed with the OpenFoam solver *shallowWaterFoam* [53], while the ROMs discretize the residual with a much simpler numerical scheme.

The FOM numerical scheme is the PIMPLE algorithm, a combination of PISO [54] (Pressure Implicit with Splitting of Operator) and SIMPLE [55] (Semi-Implicit Method for Pressure-Linked Equations). For the shallow water equations the free surface height h plays the role of the pressure in the Navier–Stokes equations, regarding the PIMPLE algorithm implementation. The number of outer PISO corrections is 3.

The time discretization is performed with the semi-implicit Euler method. The non-linear advection terms are discretized with the Linear-Upwind Stabilised Transport (LUST) scheme that requires a stencil with 2 layers of adjacent cells for the hyper-reduction. The gradients are linearly interpolated through Gauss formula. The solutions for hU are obtained with Gauss–Seidel iterative method, and for h with the conjugate gradient method preconditioned by the Diagonal-based Incomplete Cholesky (DIC) preconditioner. For both of them the absolute tolerance on the residual is 1e−6 and the relative tolerance of the residual w.r.t. the initial condition is 0.1.

The residual of ROMs is instead discretized as follows. If we represent with M_{hU}, M_h the mass matrices, with $G(h)$ the discrete gradient vector of h, and with $C_{hU}((hU)^{t-1})$, $C_h(U^{t-1})$ the advection matrices, then, at every time instant t, the discrete equations

$$\frac{M_{hU}}{\Delta t}(hU)^t + C_{hU}((hU)^{t-1})U^t + ghG(h) = \frac{M_{hU}}{\Delta t}(hU)^{t-1}, \tag{39}$$

$$\frac{M_h}{\Delta t}h^t + C_h(U^{t-1})h^t = \frac{M_h}{\Delta t}h^{t-1}, \tag{40}$$

are solved for the state $((hU)^t, h^t)$ with a semi-implicit Euler method. The same numerical schemes and linear systems iterative solvers of the FOM are employed. In principle, they could be changed.

Since now the stencil of a single cell needs two layers of adjacent cells for the discretizations, for each internal magic point, 12 additional cells need to be considered for the hyper-reduction.

4.2.2 Manifold Learning

The CAE architecture for the SWE model is reported in Table 6. This time one encoder and two decoders, one for the velocity U and one for the height h are trained. Moreover, to increase the generalization capabilities we converted 2 layers of the decoder for U in recurrent convolutional layers as shown in the Appendix. Even with this modification the initial time steps, from 0 to 0.01 are associated to a high reconstruction error: the relative L^2-error is around 0.1 for every test parameter at the initial time instants, slowly decreasing towards the accuracy shown in Fig. 12 after $t = 0.01$, chosen as initial instant from here onward.

The CAE is trained for 500 epochs with a batch size of 20 and an initial learning rate of 1e−4, that halves if the loss from Eq. (12) does not decrease after 50 epochs. In this case the state is (U, h) so the encoder has three channels, two for the velocity components, U_1, U_2, and one for the height, h. The high computational cost is shown in Table 4. We have to observe that nor the architecture is parsimonious for a good approximation of the discrete solution manifold in the time interval $[0.01, 0.2]$, neither the number of training snapshots 5010 is optimized to reach the highest efficiency with the lowest computational cost. Our focus is obtaining a satisfactory reconstruction error in order to build up our ROMs.

The solution manifold parametrized by the decoder achieves the reconstruction error of a linear manifold spanned by around 20 POD modes. In fact, the decay of the reconstruction error associated to the POD approximations is faster than the previous test case in the time interval $[0.01, 0.2]$.

It can be seen from the representation of the latent dynamics associated to the train parameters in Fig. 13, that the initial solutions overlap. This and the low accuracy could be explained by the fact that the FOM dynamics has different scales, especially for the velocity U that from the initial constant zero solution reaches a magnitude of 10^{-1} m per seconds. Further observations and possible solutions are presented in the Discussion Sect. 5.

4.2.3 Hyper-Reduction and Teacher–Student Training

Mimicking the structure of the CAE, the compressed decoder is split in two, one for the velocity U and one for the free surface height h. The architecture for both the decoders is a feed-forward NN with a single hidden layer; they are reported in the Table 8. The compressed decoders are trained for 1500 epochs, with a batch size of 20, and an initial learning rate of 1e−4 that halves after 100 epochs if the loss does not decrease, with a minimum value of 1e−6. As for the previous test case, the number of magic points, hidden layer sizes, training times, average training epoch computational cost are reported in Table 3. This time for each magic point correspond 3 degrees of freedom, so the actual output dimension of the compressed decoders is three times the submesh sizes.

The relative L^∞-error of the compressed decoder is shown in Fig. 14. This time the extrapolation error is sensibly higher as already seen in the reconstruction error of the CAE.

As in the previous case, both the NM-LSPG-ROC and NM-LSPG-ROC-TS ROMs are considered. This time it's the NM-LSPG-ROC's dynamics to be less stable with 7 maximum residual evaluations of the LM algorithm. In this respect, for parameter test 4 and mp = 100

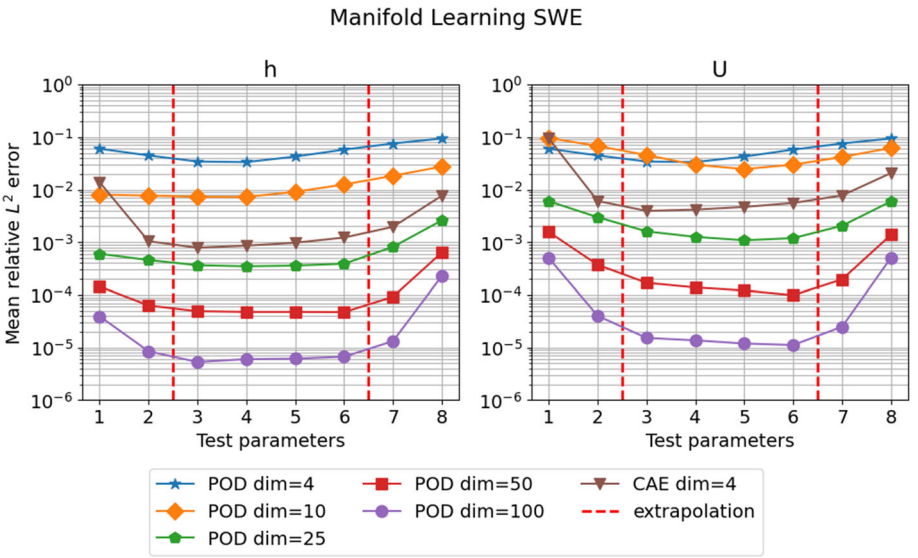

Fig. 12 Comparison between the CAE's and POD's projection errors of the discrete solution manifold represented by the $N^{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}} = 10 \cdot 501 = 5010$ training snapshots. The error is evaluated on the 8 test parameters, each associated with a time series of $N^{t}_{\text{test}} = 475$, for a total of $N_{\text{test}} = 3800$ test snapshots. The mean is performed over the time scale (Color figure online)

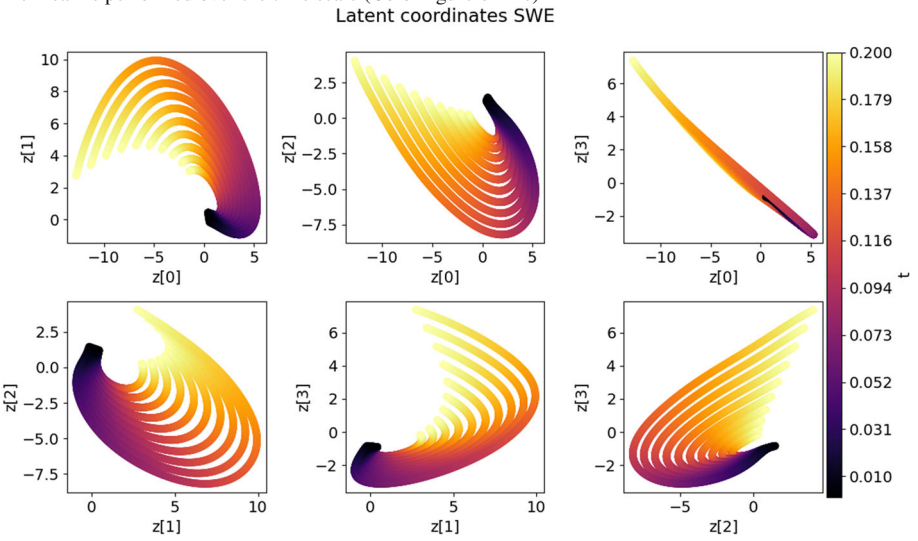

Fig. 13 Correlations among the 4 latent coordinates of the train set, 10 in total. They are obtained projecting the train snapshots into the latent space of the CAE with the encoder. The coloring corresponds to the time instants from 0. to 0.2, with a time step of 4e−4 s. The initial time steps employed in the ROMs is 0.01 (Color figure online) s

Table 3 In this table are reported for the SWE test case: the submesh size and the hidden layer (HL) number of nodes of the compressed decoder; the Teacher–Student training (TS) duration in seconds, the Teacher–Student training average epoch duration in seconds; the average time step for NM-LSPG-GNAT with compressed decoder (GNAT-TS) in milliseconds, and the average time step for NM-LSPG-GNAT with the hyper-reduced residuals but full CAE decoder (GNAT-no-TS) in milliseconds

MP	Submesh size	HL size	TS total epochs
25	238	600	582 [s]
50	345	600	1234 [s]
100	590	900	6428 [s]
150	654	900	6746 [s]
MP	TS avg epoch	Avg GNAT-TS	Avg GNAT-no-TS
25	0.388432 [s]	3.227 [ms]	14.782 [ms]
50	0.823084 [s]	4.062 [ms]	13.742 [ms]
100	4.285968 [s]	4.387 [ms]	15.238/**22.688** [ms]
150	4.497579 [s]	4.307 [ms]	14.307 [ms]

In bold the average computational cost with 13 maximum residual evaluations due to the instability in the evolution of the dynamics of the test parameter 4

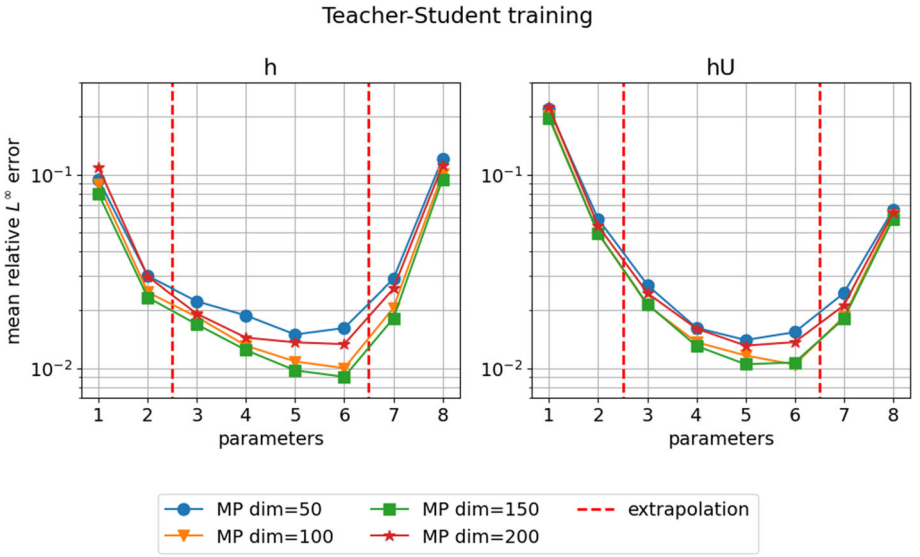

Fig. 14 Prediction accuracy on the test snapshots restricted to the magic points. The accuracy is measured with the relative L^∞-error averaged over the time trajectories associated to each one of the $N^\mu_{\text{test}} = 8$ test samples (Color figure online)

the maximum number of residual evaluations is increased to 13; in the error plots in Figs. 15 and 16, only the value for parameter 4, mp $= 100$ is substituted. It is relevant to notice that the extrapolation error is lower for higher numbers of magic points and for the NM-LSPG-ROC method.

4.2.4 Comparison with Data-Driven Predictions Based on LSTM

The same LSTM architecture of the NCL test case is trained with the same training procedure, only that now there are 5010 training parameters-latent coordinates pairs. For the

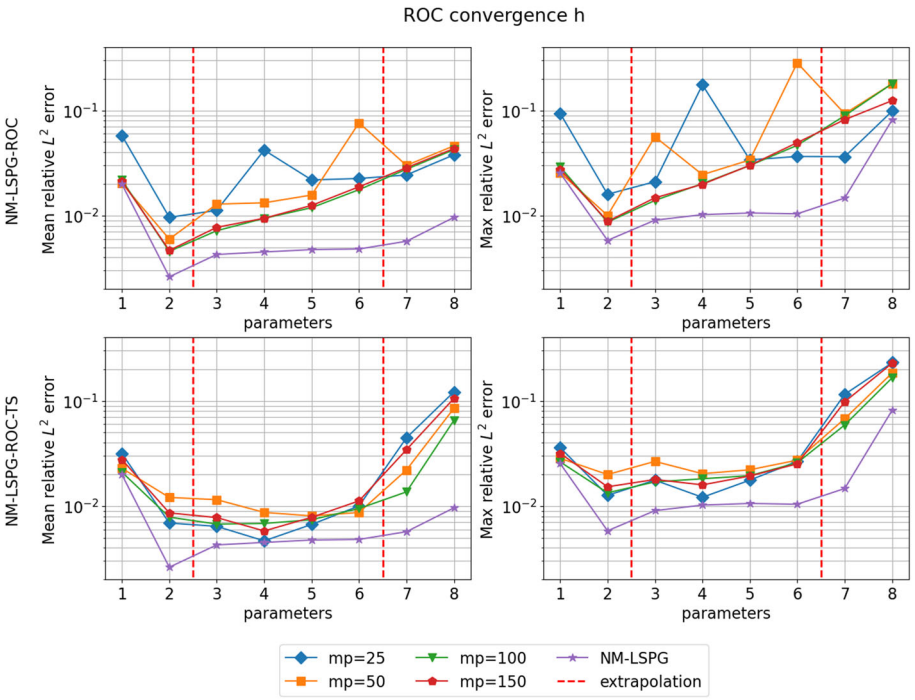

Fig. 15 First row: NM-LSPG-ROC mean and max relative L^2-error. Second row: NM-LSPG-ROC-TS mean and max relative L^2-error. In violet also the NM-LSPG accuracy is reported. The mean and max values are evaluated with respect to the time series of intermediate solutions associated to each one of the 8 test parameters (Color figure online)

architecture's specifics see Table 9. The computational costs introduced in the previous test case are reported also for the SWE model in Table 4.

With the architecture and training procedure employed, the LSTM could not achieve a good accuracy at the initial time instants after $t_0 = 0.01$ s: for this reason in the plot of the errors in Fig. 17 is reported both the mean over the whole test time series of 475 elements and over the time series after the 40-th element of the 475, that corresponds to the time instant 0.017 s. This issue should be ascribed at what we discussed in Sect. 4.2.2, about the latent dynamics overlappings. Further remarks are provided in the Discussion Sect. 5.

A part from this, the LSTM predictions are for almost every test parameter above only the NM-LSPG and CAE's reconstruction errors. Even in the extrapolation regimes, the predictions are more accurate than the NM-LSPG-ROC and NM-LSPG-ROC-TS reduced-order models. Regarding the computational costs, this time not only the LSTM model but also the NM-LSPG-ROC-TS ROMs achieve a little speed-up w.r.t. the FOM. The choice of doubling the decoders has repercussions in the online costs, but it was made only in an effort to reach a good reconstruction error of the CAE; maybe more light architectures could be employed for the restricted time interval [0.01, 0.2].

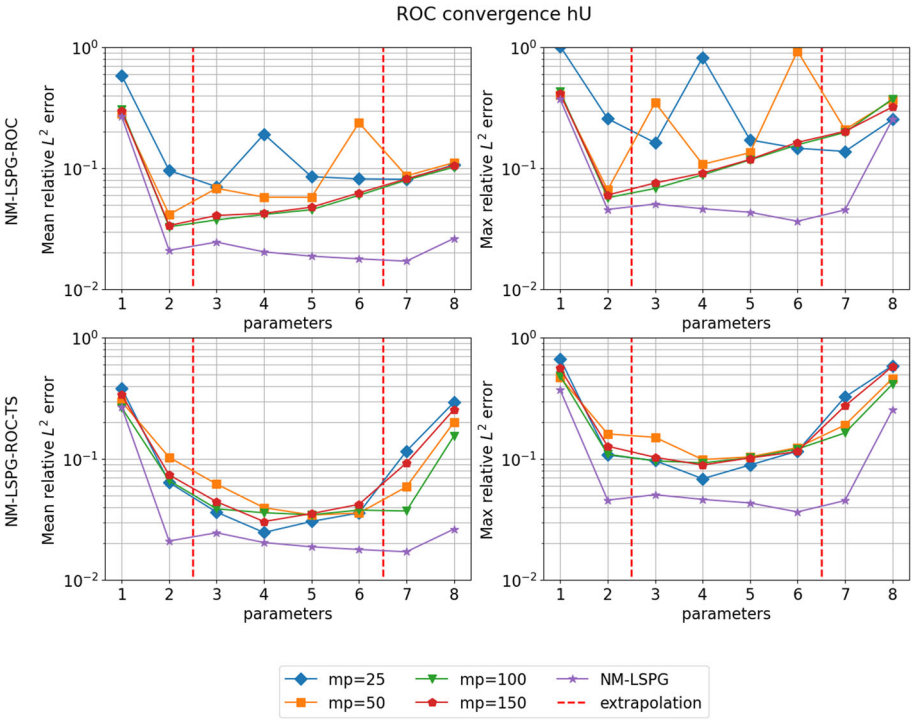

Fig. 16 First row: NM-LSPG-ROC mean and max relative L^2-error. Second row: NM-LSPG-ROC-TS mean and max relative L^2-error. In violet also the NM-LSPG accuracy is reported. The mean and max values are evaluated with respect to the time series of intermediate solutions associated to each one of the 8 test parameters (Color figure online)

5 Discussion

We comment the numerical results obtained:

- Computational cost of the CAEs training. It is evident from Tables 2 and 7 that the offline stage's computational cost is dominated by the CAEs trainings. We have to remark that the architectures were not optimized to be the most parsimonious ones in order to achieve the desired reconstruction error. Moreover, libtorch training took almost twice more time than the same architecture's training in PyTorch, due to implementation inconsistencies. The cost of the forward evaluations of the decoder are comparable instead, not changing much the online costs. The training of the CAEs could be further reduced with transfer learning [3] or preprocessing steps that enlight some features of the dynamics that are more easily learnable, as was done in [20]. Also, the number of training snapshots could be optimized further for the test cases presented. The same observations apply also for the compressed decoders. Moreover, a parallel implementation of the training procedures on more than one GPU is mandatory to achieve competitive computational costs.
- Simultaneous CAE and compressed decoder/LSTM training. The additional costs of the LSTM and compressed decoder training could be cut with a unified training of the CAE and compressed decoder: after some epochs, the training of the LSTM or compressed decoder could be switched on and performed at the same time of the CAE's since the only

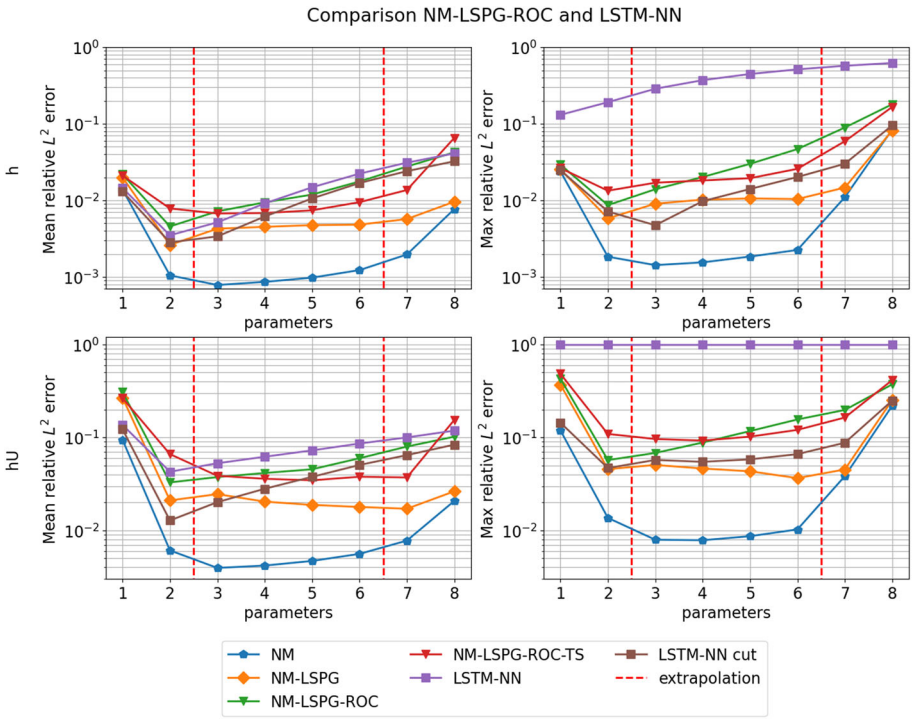

Fig. 17 Comparison of the accuracies between all the ROMs presented on the test set of 8 parameters, each associated to a time series of 475 intermediate solutions. The number of magic points employed for NM-LSPG-ROC and NM-LSPG-ROC-TS is 100. The mean and maximum relative L^2-errors are taken with respect to the time scale. Since the LSTM is inaccurate between the time instants 0.01 and 0.017 s we report also the mean over the time interval [0.017, 0.2] witch label 'LSTM-NN cut', to establish a fair comparison (Color figure online)

additional information, apart from the restriction of the snapshots into the magic points, is the latent dynamics coordinates learned anyway during the CAE's optimization.

- Increasing the speed-up of the non-linear manifold ROMs. The bottleneck for the efficiency of the NM-LSPG-ROC-TS and NM-LSPG-ROC ROMs in the online stage is the cost of the compressed decoder or CAE's decoder forward. Regarding the NLC model, our results for the average time step of the NM-LPSG-GNAT-TS and NM-LSPG-GNAT ROMs from Table 1 are comparable if not lower than the approximate time step of 7–8 ms for the NM-LSPG-GNAT with shallow autoencoders ROM presented in [5]. The difference is that now the FOM implemented with the FVM in OpenFoam [53] takes 2.42 s for 2000 time steps instead of 140.67 s for 1500 time steps in [5] with finite differences for a similar test case, i.e. 2d burgers equation instead of our non-linear conservation law. To show a more consistent speed-up w.r.t. the FOM, as for the SWE test case, the number of degrees of freedom could be increased without influencing the NM-LSPG-ROC-TS ROM since it is not dependent on the FOM's dimension. In a weaker sense also the NM-LSPG-ROC ROM is also independent on the number of degrees of freedom of the FOM, apart from the weights of the CAE's decoder that concur in increasing the cost of a single forward in the online stage.

Table 4 Offline stage: full-order model (FOM) computation of the 5010 snapshots, average cost of a single epoch for the training of the CAE with a batch size of 20, and total cost for 500 epochs; average epoch's cost for the LSTM training with a batch size of 100, and total cost for 10,000 epochs

Offline stage	Time
FOM snapshots evaluation	137.079881 [s]
CAE training single epoch avg	94.700484 [s]
CAE training	51431 [s]
LSTM-NN training single epoch avg	0.107722 [s]
LSTM-NN training	1076.150 [s]
Online stage	**Time**
Avg FOM time step	7.251 [ms]
Avg FOM full dynamics	116.024578 [s]
Avg NM-LSPG time step	22.126 [ms]
Avg NM-LSPG full-dynamics	336.171916 [s]
Avg LSTM-NN time step	1.802 [ms]
Avg LSTM-NN full-dynamics	6.834247 [s]

Online stage: for the FOM, NM-LSPG and LSTM-NN models it is reported the average of a single time step cost over all the 8 test parameters and time series, and the average cost of the full dynamics over the 8 test parameters. The LSTM-NN full-dynamics is evaluated with a single forward

- Generalization error of LSTM and CAE and additional inductive biases. The generalization error of the NN employed depends on a lot of factors, from the number of training samples to the regularization term in the loss, the architectures, etc. It is thus difficult to predict how much the accuracy of the predictions will decay outside the training range. Adding a physics-informed term in the loss of the CAE does not provide an improvement of the reconstruction error if enough training data are employed as in our test cases. Some regularization properties could be imposed in the latent dynamics to facilitate the evolution of the NM-LPSG-ROC ROMs, for example imposing a linear latent dynamics could be beneficial
- Purely data-driven LSTM ROM vs NM-LSPG-ROC and NM-LSPG-ROC-TS ROMs. One crucial difference is interpretability: in the first case, the latent dynamics is obtained training the LSTM to approximate the latent coordinates, in the second case, the latent dynamics is evolved in time minimizing the hyper-reduced residual based on the physical model's equations. Regarding the LSTM predictions, we have seen in the test cases that despite the higher accuracy in the training range and the low computational online cost, there might be other issues in the extrapolation regime: in the NLC test case, the accuracy was sensitively lower in the extrapolation regime, even if it could be improved in theory increasing the layers and nodes of the LSTM in exchange for a higher training cost; in the SWE test case the initial time step from 0.01 to 0.017 s could not be well-approximated due to different scales and overlappings in the latent dynamics. The NM-LSPG-ROC and NM-LSPG-ROC-TS mitigated in some sense these issues, delegating less effort in the tuning of the hyperparameters of the LSTM and increasing the interpretability of the results. Moreover, the LSTM-NN, approximate the dynamics only every time step imposed by the training input–output pairs, while the non-linear manifold ROMs, can in principle approximate the latent dynamics with an arbitrary small time step, since the decoder provides a continuous approximation of the solution manifold and the dynamics is evolved based on a numerical scheme with changeable time step. With respect to purely

data-driven methods though, non-linear manifold methods require a lower reconstruction error of the CAE since every successive ROMs' dynamics evolution depends on how well the intermediate solutions are reconstructed through the decoder.

- Learn the solution manifolds with autoencoders in unstructured meshes. The natural question of how to extend the CAE architecture to 3D or unstructured meshes, is being currently studied. In the literature there are already interesting results that employ graph neural networks and their variants to find latent representations of 3d simulations [23].

6 Conclusions and Perspectives

We have developed two new hyper-reduced non-linear manifold ROMs: NM-LSPG-ROC and NM-LSPG-ROC-TS, that can be converted in NM-LSPG-GNAT and NM-LSPG-GNAT-TS. In NM-LPSG-ROC-TS the residuals of the NM-LSPG ROM are hyper-reduced with over-collocation, while the decoder of the CAE is approximated with teacher–student training into a compressed decoder. In NM-LSPG-ROC only the residuals' hyper-reduction is carried out. The methods perform similarly in accuracy and computational cost w.r.t. the NM-LSPG-GNAT with shallow autoencoders ROM introduced in [5], for a similar test case. The flexibility of our method permits to change the CAE architecture depending on the problem at hand, without imposing too many constraints on its structure, in order to reach the convergence faster and achieve a lower reconstruction error.

With respect to purely data-driven ROMs built on the CAE's solution manifold, NM-LSPG-ROC and NM-LSPG-ROC-TS provide more interpretable and, in the extrapolation regimes, sometimes more accurate predictions, and they need less hyperparameters tuning, once the CAE is trained. Moreover, the methods developed are equations-based rather than fully-intrusive, and exploit the physics of the model to evolve the latent dynamics. It is crucial, though, that the CAE's reconstruction error is sufficiently low, since the latent dynamics needs to be computed with numerical schemes that rely on the accuracy of the reconstructed solutions through the decoder of the CAE or the compressed decoder. We base these observations on the results obtained for two parametric non-linear time-dependent benchmarks presented in the numerical results Sect. 4, that is a 2d non-linear conservation law model (NLC) and a 2d shallow water equations model. Despite the speed-up is not achieved or not significant with respect to the FOMs, we reached satisfactory results in terms of the accuracy and the latent or reduced dimension of the ROMs.

Future directions of research involve the implementation of the developed ROMs in more complex applications with higher computational costs and degrees of freedom, such that a more evident speed-up is reached. This may involve the development of more parsimonious architectures and training procedures to reduce the offline cost. The research in geometric deep learning will be crucial for the possible extensions of the present methodology to mesh-based 2d and 3d simulations. More has to be done also to further improve the interpretability of the results possibly taking into account a probabilistic approach, for example using Bayesian neural networks, and adhering more tightly to the physics of the model with additional inductive biases.

Acknowledgements We acknowledge the PRIN 2017 "Numerical Analysis for Full and Reduced Order Methods for the efficient and accurate solution of complex systems governed by Partial Differential Equations" (NA-FROM-PDEs).

Funding Open access funding provided by Scuola Internazionale Superiore di Studi Avanzati - SISSA within the CRUI-CARE Agreement. This work was partially funded by European Union Funding for Research and

Innovation-Horizon 2020 Program-in the framework of the European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 "Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics" P.I. Professor Gianluigi Rozza.

Data availability The datasets generated during and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

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

A Neural Networks' Architectures

The architectures of the CAEs are shown in Tables 5, 6, of the compressed decoders in Tables 7 and 8, and of the LSTMs in Table 9. We mainly employ the Exponential Linear Unit (ELU) and Rectified Linear Unit (ReLU) activation functions. The padding is symmetric. The labels *Conv2d*, *ConvTr2d* and *ConvTr2dRec*, stand for 2d convolutions, transposed 2d convolutions [36], and 2d transposed convolution associated to a recurrent layer, i.e. they are summed to the previous layer and then passed to an ELU activation function.

Table 5 Nonlinear conservation law model's convolutional autoencoder

Encoder	Activation	Weights	Padding
Conv2d	ELU	[2, 8, 5, 5]	0
Conv2d	ELU	[8, 16, 3, 3]	1
Conv2d	ELU	[16, 32, 3, 3]	1
Conv2d	ELU	[32, 64, 3, 3]	1
Conv2d	ELU	[64, 128, 2, 2]	1
Linear	ELU	[1152, 4]	–
Decoder	**Activation**	**Weights**	**Padding**
Linear	ELU	[4, 1152]	–
ConvTr2d	ELU	[128, 64, 2, 2]	1
ConvTr2d	ELU	[64, 32, 3, 3]	1
ConvTr2d	ELU	[32, 16, 4, 4]	1
ConvTr2d	ELU	[16, 8, 4, 4]	0
ConvTr2d	ReLU	[8, 2, 4, 4]	1

Table 6 Shallow water equations model's convolutional autoencoder

Encoder	Activation	Weights	Padding
Conv2d	ELU	[2, 8, 5, 5]	0
Conv2d	ELU	[8, 16, 3, 3]	1
Conv2d	ELU	[16, 32, 3, 3]	1
Conv2d	ELU	[32, 64, 3, 3]	1
Conv2d	ELU	[64, 128, 2, 2]	1
Linear	ELU	[1152, 4]	–
Decoder h	**Activation**	**Weights**	**Padding**
Linear	ELU	[4, 1152]	–
ConvTr2d	ELU	[240, 120, 2, 2]	1
ConvTr2d	ELU	[120, 60, 3, 3]	1
ConvTr2d	ELU	[60, 30, 4, 4]	1

The decoder for U has two recurrent layers

Table 6 continued

Decoder h	Activation	Weights	Padding
ConvTr2d	ELU	[30, 15, 4, 4]	0
ConvTr2d	ReLU	[15, 1, 4, 4]	1
Decoder U	**Activation**	**Weights**	**Padding**
Linear	ELU	[4, 2700]	–
ConvTr2d	ELU	[300, 75, 2, 2]	1
ConvTr2d	ELU	[150, 75, 3, 3]	1
ConvTr2dRec	–	[150, 75, 3, 3]	1
ConvTr2d	ELU	[75, 35, 4, 4]	1
ConvTr2d	ELU	[35, 20, 4, 4]	0
ConvTr2dRec	–	[35, 20, 4, 4]	0
ConvTr2d	–	[20, 2, 4, 4]	1

The decoder for U has two recurrent layers

Table 7 Nonlinear conservation law model's compressed decoders

Magic points	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
50	[4, 300]	ELU	[300, 278]	ReLU
100	[4, 350]	ELU	[350, 492]	ReLU
150	[4, 400]	ELU	[400, 670]	ReLU

Table 8 Shallow water equations model's compressed decoders: height h and velocity U in order

Magic points	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
25	[4, 200]	ELU	[200, 238]	ReLU
50	[4, 200]	ELU	[200, 345]	ReLU
100	[4, 300]	ELU	[300, 590]	ReLU
150	[4, 300]	ELU	[300, 654]	ReLU
Magic points	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
25	[4, 400]	ELU	[400, 478]	–
50	[4, 400]	ELU	[400, 960]	–
100	[4, 600]	ELU	[600, 1180]	–
150	[4, 600]	ELU	[600, 1308]	–

Table 9 Nonlinear conservation law model's and shallow water equations model's long-shot term memory neural network

	Input dim	Output dim		Number of LSTM layers
LSTM layer	2	100		2
	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
Linear encoding layer	[100, 50]	ELU	[50, 4]	–

References

1. Hesthaven, J.S., Rozza, G., Stamm, B., et al.: Certified Reduced Basis Methods for Parametrized Partial Differential Equations, vol. 590. Springer, Berlin (2016)
2. Quarteroni, A., Manzoni, A., Negri, F.: Reduced Basis Methods for Partial Differential Equations: An Introduction, vol. 92. Springer, Berlin (2015)
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
4. Lee, K., Carlberg, K.T.: Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. J. Comput. Phys. **404**, 108973 (2020)
5. Kim, Y., Choi, Y., Widemann, D., Zohdi, T.: A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. arXiv preprint arXiv:2009.11990 (2020)
6. Carlberg, K., Farhat, C., Cortial, J., Amsallem, D.: The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. J. Comput. Phys. **242**, 623–647 (2013). https://doi.org/10.1016/j.jcp.2013.02.028
7. Chen, Y., Gottlieb, S., Ji, L., Maday, Y.: An EIM-degradation free reduced basis method via over collocation and residual hyper reduction-based error estimation. arXiv preprint arXiv:2101.05902 (2021)
8. Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., Parashar, M., Patra, A., Sethian, J., Wild, S., et al.: Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC, USA (2019)
9. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge (2012)
10. Benner, P., Ohlberger, M., Cohen, A., Willcox, K.: Model Reduction and Approximation: Theory and Algorithms. SIAM, Philadelphia (2017)
11. Amsallem, D., Farhat, C.: Interpolation method for adapting reduced-order models and application to aeroelasticity. AIAA J. **46**(7), 1803–1813 (2008)
12. Franz, T., Zimmermann, R., Görtz, S., Karcher, N.: Interpolation-based reduced-order modelling for steady transonic flows via manifold learning. Int. J. Comput. Fluid Dyn. **28**(3–4), 106–121 (2014)
13. Bhattacharjee, S., Matouš, K.: A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. J. Comput. Phys. **313**, 635–653 (2016)

14. Bernard, F., Iollo, A., Riffaud, S.: Reduced-order model for the BGK equation based on POD and optimal transport. J. Comput. Phys. **373**, 545–570 (2018)
15. Díez, P., Muixí, A., Zlotnik, S., García-González, A.: Nonlinear dimensionality reduction for parametric problems: a kernel Proper Orthogonal Decomposition (kPOD). arXiv preprint arXiv:2104.13765 (2021)
16. Li, W., Zhen, M., Yaolin, J.: Model order reduction based on Galerkin KPOD for partial differential equations with variable coefficients. J. Numer. Methods Comput. Appl. **42**(3), 226 (2021)
17. Lucia, D.J., King, P.I., Beran, P.S.: Reduced order modeling of a two-dimensional flow with moving shocks. Comput. Fluids **32**(7), 917–938 (2003)
18. Buffoni, M., Telib, H., Iollo, A.: Iterative methods for model reduction by domain decomposition. Comput. Fluids **38**(6), 1160–1167 (2009)
19. Mücke, N.T., Bohté, S.M., Oosterlee, C.W.: Reduced order modeling for parameterized time-dependent PDEs using spatially and memory aware deep learning. J. Comput. Sci. **53**, 101408 (2021)
20. Fresca, S., Manzoni, A.: POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Comput. Methods Appl. Mech. Eng. **388**, 114181 (2022)
21. Xu, J., Duraisamy, K.: Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. Comput. Methods Appl. Mech. Eng. **372**, 113379 (2020)
22. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017)
23. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W.: Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409 (2020)
24. Cohen, A., DeVore, R.: Kolmogorov widths under holomorphic mappings. IMA J. Numer. Anal. **36**(1), 1–12 (2016)
25. Babuška, I., Nobile, F., Tempone, R.: A stochastic collocation method for elliptic partial differential equations with random input data. SIAM J. Numer. Anal. **45**(3), 1005–1034 (2007)
26. Lassila, T., Manzoni, A., Quarteroni, A., Rozza, G.: Generalized reduced basis methods and n-width estimates for the approximation of the solution manifold of parametric PDEs. In: Brezzi, F., Colli Franzone, P., Gianazza, U., Gilardi, G. (eds.) Analysis and Numerics of Partial Differential Equations, pp. 307–329. Springer, Berlin (2013)
27. Geller, D., Pesenson, I.Z.: Kolmogorov and linear widths of balls in Sobolev spaces on compact manifolds. Math. Scand. **115**, 96–122 (2014)
28. Ohlberger, M., Rave, S.: Reduced basis methods: success, limitations and future challenges. arXiv preprint arXiv:1511.02021 (2015)
29. Greif, C., Urban, K.: Decay of the Kolmogorov N-width for wave problems. Appl. Math. Lett. **96**, 216–222 (2019)
30. Franco, N.R., Manzoni, A., Zunino, P.: A deep learning approach to reduced order modelling of parameter dependent partial differential equations. arXiv preprint arXiv:2103.06183 (2021)
31. DeVore, R.A., Howard, R., Micchelli, C.: Optimal nonlinear approximation. Manuscr. Math. **63**(4), 469–478 (1989)
32. Temlyakov, V.N.: Nonlinear Kolmogorov widths. Math. Notes **63**(6), 785–795 (1998)
33. Pichi, F., Ballarin, F., Rozza, G., Hesthaven, J.S.: An artificial neural network approach to bifurcating phenomena in computational fluid dynamics (2021)
34. Torlo, D.: Model reduction for advection dominated hyperbolic problems in an ALE framework: offline and online phases. arXiv preprint arXiv:2003.13735 (2020)
35. Papapicco, D., Demo, N., Girfoglio, M., Stabile, G., Rozza, G.: The Neural Network Shifted-Proper Orthogonal Decomposition: A Machine Learning Approach for Non-linear Reduction of Hyperbolic Equations. Elsevier BV, Amsterdam (2022). https://doi.org/10.1016/j.cma.2022.114687
36. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates Inc, Red Hook (2019)
37. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
38. Lee, K., Carlberg, K.: Deep conservation: a latent dynamics model for exact satisfaction of physical conservation laws. arXiv preprint arXiv:1909.09754 (2019)
39. Smets, B., Portegies, J., Bekkers, E., Duits, R.: PDE-based group equivariant convolutional neural networks. arXiv preprint arXiv:2001.09046 (2020)

40. Finzi, M., Stanton, S., Izmailov, P., Wilson, A.G.: Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In: International Conference on Machine Learning, pp. 3165–3176. PMLR (2020)
41. Goyal, P., Benner, P.: LQResNet: a deep neural network architecture for learning dynamic processes. arXiv preprint arXiv:2103.02249 (2021)
42. Quarteroni, A., Sacco, R., Saleri, F.: Matematica Numerica. Springer, Berlin (2010)
43. Guennebaud, G., Jacob, B., et al.: Eigen v3. http://eigen.tuxfamily.org (2010)
44. Carlberg, K., Bou-Mosleh, C., Farhat, C.: Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. Int. J. Numer. Methods Eng. **86**(2), 155–181 (2011)
45. Barrault, M., Maday, Y., Nguyen, N.C., Patera, A.T.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. C. R. Math. **339**(9), 667–672 (2004)
46. Chaturantabut, S., Sorensen, D.C.: Nonlinear model reduction via discrete empirical interpolation. SIAM J. Sci. Comput. **32**(5), 2737–2764 (2010)
47. Carlberg, K., Farhat, C., Cortial, J., Amsallem, D.: The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. J. Comput. Phys. **242**, 623–647 (2013)
48. Choi, Y., Carlberg, K.: Space-time least-squares Petrov–Galerkin projection for nonlinear model reduction. SIAM J. Sci. Comput. **41**(1), 26–58 (2019)
49. Choi, Y., Coombs, D., Anderson, R.: SNS: a solution-based nonlinear subspace method for time-dependent model order reduction. SIAM J. Sci. Comput. **42**(2), 1116–1146 (2020)
50. Stabile, G., Zancanaro, M., Rozza, G.: Efficient geometrical parametrization for finite-volume-based reduced order methods. Int. J. Numer. Methods Eng. **121**(12), 2655–2682 (2020)
51. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: a survey. Int. J. Comput. Vis. **129**(6), 1789–1819 (2021)
52. Stabile, G., Gianluigi, R.: ITHACA-FV in real time highly advanced computational applications for finite volumes. http://www.mathlab.sissa.it/ithaca-fv. Accessed 28 Feb 2022
53. OpenFOAM Documentation Website. https://www.openfoam.com
54. Patankar, S.V., Spalding, D.B.: A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In: Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion, pp. 54–73. Elsevier, Amsterdam (1983)
55. Issa, R., Ahmadi-Befrui, B., Beshay, K., Gosman, A.: Solution of the implicitly discretised reacting flow equations by operator-splitting. J. Comput. Phys. **93**(2), 388–410 (1991)