Journal of Scientific Computing (2023) 94:11
https://doi.org/10.1007/s10915-022-02059-4

Generalised Latent Assimilation in Heterogeneous Reduced Spaces with Machine Learning Surrogate Models

Sibo Cheng[1] · Jianhua Chen[2,3] · Charitos Anastasiou[4] · Panagiota Angeli[4] · Omar K. Matar[2] · Yi-Ke Guo[1] · Christopher C. Pain[5] · Rossella Arcucci[1,5]

Received: 8 April 2022 / Revised: 18 August 2022 / Accepted: 30 October 2022 /
Published online: 1 December 2022
© The Author(s) 2022

Abstract
Reduced-order modelling and low-dimensional surrogate models generated using machine learning algorithms have been widely applied in high-dimensional dynamical systems to improve the algorithmic efficiency. In this paper, we develop a system which combines reduced-order surrogate models with a novel data assimilation (DA) technique used to incorporate real-time observations from different physical spaces. We make use of local smooth surrogate functions which link the space of encoded system variables and the one of current observations to perform variational DA with a low computational cost. The new system, named generalised latent assimilation can benefit both the efficiency provided by the reduced-order modelling and the accuracy of data assimilation. A theoretical analysis of the difference between surrogate and original assimilation cost function is also provided in this paper where an upper bound, depending on the size of the local training set, is given. The new approach is tested on a high-dimensional (CFD) application of a two-phase liquid flow with non-linear observation operators that current Latent Assimilation methods can not handle. Numerical results demonstrate that the proposed assimilation approach can significantly improve the reconstruction and prediction accuracy of the deep learning surrogate model which is nearly 1000 times faster than the CFD simulation.

Keywords Deep learning · Data assimilation · Reduced-order-modelling · Explainable AI · Recurrent neural networks

✉ Rossella Arcucci
r.arcucci@imperial.ac.uk

1 Department of Computing, Data Science Institute, Imperial College London, London SW7 2AZ, UK

2 Department of Chemical Engineering, Imperial College London, London SW7 2AZ, UK

3 State Key Laboratory of Multiphase Complex Systems, Institute of Process Engineering, Chinese Academy of Sciences, Beijing 100190, China

4 Department of Chemical Engineering, University College London, London WC1E 6BT, UK

5 Department of Earth Science and Engineering, Imperial College London, London SW7 2AZ, UK

Abbreviations

NN	Neural network
ML	Machine learning
LA	Latent assimilation
DA	Data assimilation
PR	Polynomial regression
AE	Autoencoder
CAE	Convolutional autoencoder
RNN	Recurrent neural network
CNN	Convolutional neural network
LSTM	Long short-term memory
POD	Proper orthogonal decomposition
SVD	Singular value decomposition
ROM	Reduced-order modelling
CFD	Computational fluid dynamics
2D	Two-dimensional
MSE	Mean square error
LHS	Latin hypercube sampling
DL	Deep learning
GLA	Generalised latent assimilation

List of Symbols

$\mathbf{x}_t$	State vector in the full space at time t
$\tilde{\mathbf{x}}_t$	Encoded state in the latent space at time t
$\tilde{\mathbf{x}}_{b,t}$	Background (predicted) state in the latent space at time t
$\tilde{\mathbf{x}}_{a,t}$	Analysis (assimilated) state in the latent space at time t
$\mathbf{x}_{\text{true},t}/\tilde{\mathbf{x}}_{\text{true},t}$	True state vector in the full/latent space at time t
$\mathbf{x}^r_{\text{POD}}$, $\mathbf{x}^r_{\text{CAE}}$, $\mathbf{x}^r_{\text{POD AE}}$	Reconstructed state in the full space
$\mathbf{y}_t$	Observation vector in the full space at time t
$\tilde{\mathbf{y}}_t$	Encoded observation vector in the latent space at time t
$\mathcal{E}_x$, $\mathcal{E}_y$	Encoder for state/observation vectors
$\mathcal{D}_x$, $\mathcal{D}_y$	Decoder for state/observation vectors
$\tilde{\mathbf{L}}_{\mathbf{X},q}$	POD projection operator with truncation parameter q
$\mathcal{H}_t$	Transformation operator in the full physical space
$\tilde{\mathcal{H}}_t$	Transformation operator linking different latent spaces
$\tilde{\mathcal{H}}^p_t$	Approximated transformation operator in GLA
$\tilde{\mathbf{B}}_t$, $\tilde{\mathbf{R}}_t$	Error covariance matrices in the latent space

1 Introduction

Spatial field prediction and reconstruction are crucial in the control of high-dimensional physical systems for applications in CFD, geoscience or medical science. Running physics-informed simulations is often computationally expensive, especially for high resolution and multivariate systems. Over the past years, numerous studies have been devoted to speed up the simulation/prediction of dynamical systems by constructing surrogate models via reduced-order modelling (ROM) and machine learning (ML) techniques [1–4]. More precisely, the simulation/experimental data are first compressed to a low-dimensional latent space through an Autoencoder (AE). A recurrent neural network (RNN) is then used to train a reduced-

order surrogate model for predicting the dynamics in the latent space using compressed data. Once the ML surrogate model is computed, monitoring the model prediction with limited sensor information constitutes another major challenge. Making use of a weighted combination of simulation (also known as 'background') and observation data [5], data assimilation (DA) methods are widely used in engineering applications for field prediction or parameter identification [5, 6].

To incorporate real-time observations for correcting the prediction of the surrogate model, the idea of Latent Assimilation (LA) was introduced [7–9] where DA is performed directly in the reduced-order latent space. It has been shown in [7] that LA has a significant advantage in terms of computational efficiency compared to classical full-space DA methods. However, current approaches of LA require the compression of the observation data into the same latent space of the state variables, which is cumbersome for some applications where the states and the observations are either compressed using different AEs or different physical quantities. The latter is common practice in geoscience and CFD applications. For example, the observation of wind speed/direction can be used to improve the quality of the initial conditions of weather forecasts [10] and precipitation data can be used to correct the river flow prediction in hydrology [11, 12].

The DA is performed through a transformation operator (usually denoted by $\mathcal{H}$) which links the state variables to real-time observations. In real applications, $\mathcal{H}$ is often highly non-linear [13]. In the case of LA, since the assimilation is carried out in the latent space, the $\mathcal{H}$ function also includes several encoder, decoder functions, leading to extra difficulties in solving the assimilation problem. Furthermore, if the state vector and the observation vector are not in the same physical space, the latent spaces where the data are reduced might be different too. In this case, the operator of the data assimilation inverse problem includes the two ML-based functions used to compress the data (state vector and observations) in two different latent spaces. Also, ML functions often involve many parameters and are difficult to train in real-time. This means that performing variational LA, when the background simulation and the observation vector are not in the same physical space, is cumbersome.

The idea of applying ML algorithms, namely recurrent neural networks in a low-dimensional latent space for learning complex dynamical systems has been recently adapted in a wide range of applications including CFD [2, 14], hydrology [12], nuclear science [15] and air pollution quantification [3]. Both proper orthogonal decomposition (POD)-type (e.g., [2, 3, 12, 16]) and neural networks (NNs)-based autoencoding methods [1, 14] have been used to construct the reduced-order latent spaces. The work of [3] is extended in [17] which relies on an Adversarial RNN when the training dataset is insufficient. In terms of compression accuracy, much effort has been devoted to compare the performance of different auto-encoding approaches. The study of [18] shows a significant advantage of NNs-based methods compared to classical POD-type approaches when dealing with highly non-linear CFD applications. A novel ROM method, combining POD and NNs AE has been introduced in the very recent work of [19]. The authors have demonstrated that one of the advantages of this approach, for projection-based ROMs, is that it does not matter whether the high-fidelity solution is on a structured or unstructured mesh. Other approaches applying convolutional autoencoders to data on unstructured meshes include space-filling curves [20], spatially varying kernels [21] or graph-based networks [22].

Performing DA in the latent space in order to monitor surrogate models with real-time observations has led to an increase in research interest recently. The approaches used in the work of [3, 23] consist of learning assimilated results directly via a RNN to reduce forecasting errors. With a similar idea, [24] proposes an iterative process of (DL) and DA, i.e., a NN is retrained after each DA step (based on NN predictions and real observations) until conver-

gence has been achieved. Collectively, the methods in [3, 23, 24] aim to enhance the system prediction by including assimilated dynamics in the training data. However, the requirement to retrain the NN when new observation data become available leads to considerable computational cost for online application of these methods.

In order to incorporate unseen real-time observation data efficiently, the recent works of [7, 8, 25] introduce the concept of LA where an AE network is used to compress the state variables and pre-processed observation data. The DA updating is performed in the reduced-order latent space subsequently. Similarly, in [9], a Generative Adversarial Network (GAN) was trained to produce time series data of POD coefficients, and this algorithm was extended to assimilate data by modifying the loss function and using the back-propagation algorithm of the GAN. Again, this produces an efficient method as no additional simulations of the high-fidelity model are required during the data assimilation process. Also, [26] proposes the use of a recurrent Kalman network in the latent space to make locally linear predictions. However, as mentioned in the Introduction, an important bottleneck of the current LA techniques is that the state and observation variables often can not be encoded into the same latent space for complex physical systems. Performing online LA thus requires a smooth, explainable and efficient-to-train local surrogate transformation function, leading to our idea of implementing polynomial regression.

Local polynomial regression has been widely used for the prediction and calibration of chaotic systems by providing smooth and easily interpretable surrogate functions. The work of [27] uses multivariate local polynomial fitting (M-MLP) which takes previous time steps in a multivariate dynamical systems as input and forecasts the evolution of the state variables. It is demonstrated numerically that the M-MLP outperforms a standard NN in the Lorenz twin experiment. Recently this work has been developed by the same authors to a local polynomial autoregressive model [28] which shows a good performance in one-step prediction. A detailed numerical comparison between (PR) and NN has also been given in [29, 30]. Their results show that PR with a polynomial degree lower than five, can achieve similar results to NNs when fitting a variety of multivariate real functions. Using a similar idea, [31] applies the local polynomial regression to provide not only the single mean forecast but an ensemble of future time steps, which provides better forecasts with noisy data as proved in their paper with geological applications.

Polynomial regression, or more generally, interpretable surrogate models such as Lasso or a Decision Tree (DT), have been widely used to approximate sophisticated deep learning algorithms to improve interpretability [32]. For example, [33] developed the model of Local Interpretable Model-agnostic Explanations (LIME) for improving the interpretability of ML classifiers. More precisely, they make use of a linear regression model to approximate a NNs classifier where the loss function is defined as a fidelity-interpretability tradeoff. The training set of the linear surrogate model is generated via samplings for local exploration of each ML input. It is pointed out by both [32] and [33] that both the distribution and the range of local samplings are crucial to the robustness of the local surrogate model. A small range may lead to overfitting while the efficiency and the local fidelity can decrease when the sampling range is too large.

A graph-based sampling strategy is proposed in the recent work of [34] to improve the performance of LIME. The principle of LIME can be easily extended by using a polynomial regression since our prime concern is not the interpretability but the smoothness of the local surrogate model. On the other hand, some effort has been given to replace the computational expensive ML models by polynomial functions which are much more efficient to evaluate. The use of a data-driven polynomial chaos expansion (PCE) has been proposed recently by [35] to perform ML regression tasks with a similar performance compared to DL and

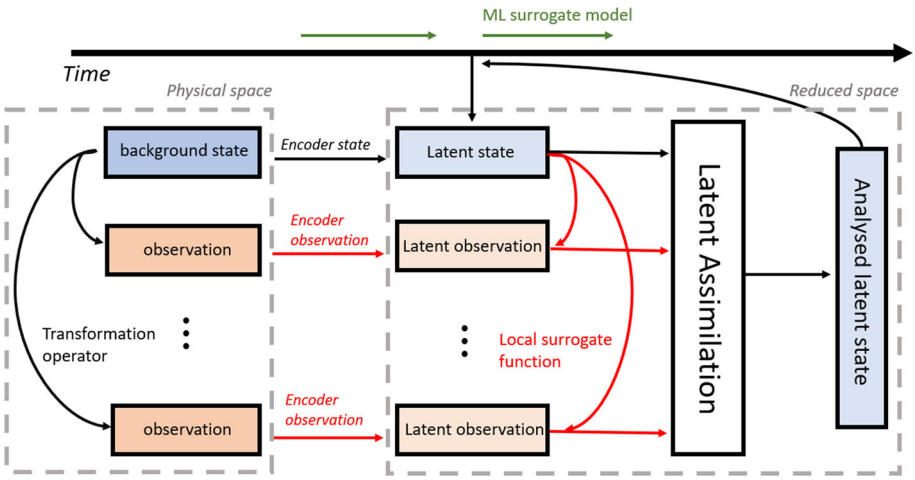


Fig. 1 Flowchart of the generalised latent assimilation with machine learning surrogate models

Support vector machine. Furthermore, PCE is able to deliver a probability density function instead of a single mean prediction for the model output. A similar idea can be found in [36] where the authors compare PCE- and NNs-based surrogate models for sensitivity analysis in a real-world geophysical problem. The study of [37] aims to reduce the over-parametrization of neural networks by using polynomial functions to fit a trained NN of the same inputs. Their study includes sophisticated NNs structures such as twodimensional (2D) convolutional neural network (CNN), in the global space. Despite the fact that the classification accuracy of the surrogate polynomial regression is slightly lower than the state-of-the-art DL approaches, the former exhibits a significantly higher noise robustness on real datasets. In addition, the theoretical study in [37] provides an upper bound of the PR learning error with respect to the number of samplings. Another important advantage of PR compared to other ML models, namely deep learning approaches, is the good performance for small training sets thanks to the small number of tuning parameters required [35]. Moreover, unlike DL methods, polynomial regression requires much less fine tuning of hyper-parameters which makes it more appropriate for online training tasks.

In this study, we develop a novel LA algorithm scheme which generalises the current LA framework [7] to heterogeneous latent spaces and non-linear transformation operators while keeping the important advantage of LA in terms of low computational cost. We use local surrogate functions to approximate the transformation operator from the latent space of the state vector to the observation one. This approach can incorporate observation data from different sources in one assimilation window as shown in Fig. 1. The latent transformation operator, which combines different encoder/decoder networks, and the state-observation transformation mapping, $\mathcal{H}$ in the full physical space, is then used to solve the LA inverse problem. A crucial requirement is ensuring both the approximation accuracy (for unseen data) and the smoothness and interpretability of the surrogate function. For these reasons, we used local PR which is sufficiently accurate and infinitely differentiable [38]. We provide both a theoretical and numerical analysis (based on a high-dimensional CFD application) of the proposed method. The surrogate models we build are based on AE and long short-term memory (LSTM) technologies which have been shown to provide stable and accurate solutions for ROMs [17].

In summary, we make the following contributions in this study:

- We propose a novel Generalised Latent Assimilation (GLA) algorithm. Making use of a local PR to open the blackbox of DL functions addresses one of the major bottlenecks of current LA approaches for combining information sources (namely state vector and observations) issued from different latent spaces. The main differences of the proposed novel Generalised LA compared to the existing LA approaches are underlined in red in Fig. 1.
- We provide a theoretical error upper-bound for the expectation of the cost function in LA when using the local surrogate polynomial function instead of the original DL function. This upper-bound, depending on the polynomial degree and the input dimension, is obtained based on the general results of learning NNs functions via PR [37].
- The new approach proposed in this work can be easily applied/extended to other dynamical systems. The repository of python code scripts, including ROM (POD, Convolutional autoencoder (CAE) and POD AE), latent LSTM and Generalised LA can be found at c

The rest of this paper is organised as follows. In Sect. 2.1, several dimension reduction methods, including POD, ML-based AE and POD AE are introduced. We then address the RNN latent surrogate model in Sect. 2.2. The novel Generalised LA approach with a theoretical analysis is described in Sect. 3 after the introduction of classical variational DA. The CFD application, as a test case in this paper, is briefly explained in Sect. 4.1. The numerical results of this study are split into two parts: Sect. 4.2 for latent surrogate modelling (including ROM reconstruction and LSTM prediction), and Sect. 5 for Generalised LA with heterogeneous latent spaces. Finally, concluding remarks are provided in Sect. 6.

2 Methodology: ROM and RNN

2.1 Reduced-Order-Modelling

Different ROM approaches are introduced in this section with the objective to build an efficient rank reduction model with a low dimensional latent space and high accuracy of reconstruction. Their performance is later compared in the oil-water flow application in Sect. 4.2.1.

2.1.1 Proper Orthogonal Decomposition

The principle of proper orthogonal decomposition was introduced in the work of [39]. In general, a set of n_{state} state snapshots, issued from one or several simulated or observed dynamics, is represented by a matrix $\mathbf{X} \in \mathbb{R}^{[\dim(\mathbf{x}) \times n_{\text{state}}]}$ where each column of $\mathbf{X}$ represents an individual state vector at a given time instant (also known as snapshots), i.e.

$$\mathbf{X}[:, i] = \mathbf{x}_{t=t_i}, \quad \forall i \in \{0, 1, \ldots, n_{\text{state}} - 1\}. \tag{1}$$

Thus the ensemble $\mathbf{X}$ describes the evolution of the state vectors. Its empirical covariance $\mathbf{C_x}$ can be written and decomposed as

$$\mathbf{C_x} = \frac{1}{n_{\text{state}} - 1} \mathbf{X}\mathbf{X}^T = \mathbf{L_X}\mathbf{D_X}\mathbf{L_X}^T \tag{2}$$

where the columns of $\mathbf{L}_{\mathbf{X}}$ are the principal components of $\mathbf{X}$ and $\mathbf{D}_{\mathbf{X}}$ is a diagonal matrix collecting the associated eigenvalues $\{\lambda_{\mathbf{X},i}, i = 0, \ldots, n_{\text{state}} - 1\}$ in a decreasing order, i.e.,

$$\mathbf{D}_{\mathbf{X}} = \begin{bmatrix} \lambda_{\mathbf{X},0} & & \\ & \ddots & \\ & & \lambda_{\mathbf{X},n_{\text{state}}-1} \end{bmatrix}. \tag{3}$$

For a truncation parameter $q \leq n_{\text{state}}$, one can construct a projection operator $\mathbf{L}_{\mathbf{X},q}$ with minimum loss of information by keeping the first q columns of $\mathbf{L}_{\mathbf{X}}$. This projection operator can also be obtained by a singular value decomposition (SVD) [40] which does not require computing the full covariance matrix $\mathbf{C}_{\mathbf{x}}$. More precisely,

$$\mathbf{X} = \mathbf{L}_{\mathbf{X},q} \mathbf{\Sigma} \mathbf{V}_{\mathbf{X},q} \tag{4}$$

where $\mathbf{L}_{\mathbf{X},q}$ and $\mathbf{V}_{\mathbf{X},q}$ are by definition with orthonormal columns. , i.e.,

$$\mathbf{L}_{\mathbf{X},q}{}^T \mathbf{L}_{\mathbf{X},q} = \mathbf{V}_{\mathbf{X},q}{}^T \mathbf{V}_{\mathbf{X},q} = \mathbf{I} \quad \text{and} \quad \mathbf{\Sigma}\mathbf{\Sigma}^T = \mathbf{D}_{q,X}, \tag{5}$$

where $\mathbf{D}_{q,X}$ is a diagonal matrix containing the first q eigenvalues of $\mathbf{D}_X$. For a single state vector $\mathbf{x}$, the compressed latent vector $\tilde{\mathbf{x}}$ can be written as

$$\tilde{\mathbf{x}} = \mathbf{L}_{\mathbf{X},q}{}^T \mathbf{x}, \tag{6}$$

which is a reduced rank approximation to the full state vector $\mathbf{x}$. The POD reconstruction then reads,

$$\mathbf{x}^r_{\text{POD}} = \mathbf{L}_{\mathbf{X},q} \tilde{\mathbf{x}} = \mathbf{L}_{\mathbf{X},q} \mathbf{L}_{\mathbf{X},q}{}^T \mathbf{x}. \tag{7}$$

The compression rate $\rho_{\mathbf{x}}$ and the compression accuracy $\gamma_{\mathbf{x}}$ are defined respectively as:

$$\gamma_{\mathbf{x}} = \sum_{i=0}^{q-1} \lambda^2_{\mathbf{X},i} \Big/ \sum_{i=0}^{n_{\text{state}}-1} \lambda^2_{\mathbf{X},i} \quad \text{and} \quad \rho_{\mathbf{x}} = q \big/ n_{\text{state}}. \tag{8}$$

2.1.2 Convolutional Auto-encoder

An auto-encoder is a special type of artificial NNs used to perform data compression via an unsupervised learning of the identity map. The network structure of an AE can be split into two parts: an encoder which maps the input vector to the latent space, and a decoder which connects the latent space and the output. More precisely, the encoder $\mathcal{E}_x$ first encodes the inputs $\mathbf{x}$ to latent vector $\tilde{\mathbf{x}} = \mathcal{E}_{\mathbf{x}}(\mathbf{x})$, which is often of a much lower dimension (i.e., $\dim(\tilde{\mathbf{x}}) \ll \dim(\mathbf{x})$). A decoder $\mathcal{D}_{\mathbf{x}}$ is then added to approximate the input vector $\mathbf{x}$ by computing a reconstructed vector $\mathbf{x}^r_{\text{AE}} = \mathcal{D}_{\mathbf{x}}\big(\mathcal{E}_{\mathbf{x}}(\mathbf{x})\big)$. The encoder and the decoder are then trained jointly with, for instance, the mean square error (MSE) as the loss function

$$J\left(\boldsymbol{\theta}_{\mathcal{E}}, \boldsymbol{\theta}_{\mathcal{D}}\right) = \frac{1}{N^{\text{AE}}_{\text{train}}} \sum_{j=1}^{N^{\text{AE}}_{\text{train}}} ||\mathbf{x}_j - \mathbf{x}^{\text{r}}_{\text{AE},j}||^2 \tag{9}$$

where $\boldsymbol{\theta}_{\mathcal{E}}, \boldsymbol{\theta}_{\mathcal{D}}$ denote the parameters in the encoder and the decoder respectively, and $N^{\text{AE}}_{\text{train}}$ represents the size of the AE training dataset.

Neural networks with additional layers or more sophisticated structures (e.g.,CNN or RNN) can better recognise underlying spatial or temporal patterns, resulting in a more effective representation of complex data. Since we aim to obtain a static encoding (i.e., a single

latent vector will not contain temporal information) at this stage, we make use of a CNN to build our first AE. In general, a convolutional layer makes use of a local filter to compute the values in the next layer. By shifting the input tensor by a convolutional window of fixed size, we obtain the output of a convolutional layer [41]. Compared to standard AE with dense layers, the advantage of CAE is mainly two-folds: the reduction of the number of parameters in the AE and the capability of capturing local information. Standard 2D CNNs are widely applied in image processing problems while for unsqaured meshes, 1D CNN and Graph NNs [21] are often prioritised due to the irregular structure. For more details about CNN and CAE, interested readers are referred to [41].

2.1.3 POD AE

The combination of POD and AE (also known as POD AE or SVD AE) was first introduced in the recent work of [19] for applications in nuclear engineering. The accuracy and efficiency of this approach has also been assessed in urban pollution applications (e.g., [17]), especially for problems with unstructured meshes. This method consists of two steps of dimension reduction. We first apply the POD to obtain the full set of principle components of the associated dynamical system. Using a part of the principle components as input, a dense autoencoder with fully connected neural networks is then employed to further reduce the problem dimension [17]. As an important note, including all of the PCs can involve some redundancy and noise which affects the performance of the AE. To avoid such effect, a prior POD truncation can be performed. In other words, both the input and ouput of this AE (with Encoder $\mathcal{E}'_{\mathbf{x}}$ and Decoder $\mathcal{D}'_{\mathbf{x}}$) are the compressed latent vectors $\tilde{\mathbf{x}}_\lambda$ associated with the POD coefficients, i.e.,

$$\tilde{\mathbf{x}}_\lambda = \mathbf{L}_{q',\mathbf{X}}{}^T \mathbf{x}, \quad \tilde{\mathbf{x}} = \mathcal{E}'_{\mathbf{x}}(\tilde{\mathbf{x}}_\lambda) \quad \text{while} \quad \tilde{\mathbf{x}}^r_\lambda = \mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}), \quad \mathbf{x}^r_{\text{POD AE}} = \mathbf{L}_{q',\mathbf{X}}\, \tilde{\mathbf{x}}^r_\lambda \tag{10}$$

where $\tilde{\mathbf{x}}^r_\lambda$ and $\mathbf{x}^r_{\text{POD AE}}$ denote the reconstruction of the POD coefficients and the reconstruction of the full physical field respectively. The prior POD truncation parameter is denoted as q'. Since the POD considerably reduce the size of the input vectors in AE, applying fully connected NNs layers is computationally affordable without the concern of over-parameterization as pointed out by [19]. Furthermore, the training time will be reduced in comparison to a full CNN AE applied directly to the high-fidelity solutions. It is important to point out that convolutional layers can also be used in the POD AE approach.

2.2 Surrogate Model Construction and Monitoring

Now that the ROM is performed, we aim to construct a lower-dimensional surrogate model by understanding the evolution of the latent variables. For this purpose, we build a ML surrogate model in the latent space, which is trained by encoded simulation data. With the development of ML techniques, there is an increasing interest in using RNNs to learn the dynamics of CFD or geoscience applications. Addressing temporal sequences as directed graphs, RNNs manage to handle complex dynamical systems because of their ability of capturing historical dependencies through feedback loops [42]. However, training standard RNNs to solve problems with long-term temporal dependencies can be computationally difficult because the gradient of the loss function may decrease exponentially with time. This is also known as the vanishing gradient problem [43]. A specific type of RNN, the long-short-term-memory (LSTM) network is developed to deal with long-term temporal dependencies. In brief, different from standard RNN units, LSTM units C_t^{LSTM} (here t denotes the time) are

capable of maintaining information in memory of long periods with the help of a memory cell. Three gates, each composed of a Sigmoid activation function $\sigma(x) = (1/(1+e^{-x}))$, are used to decide when information is memorised or forgotten. The different gates and their transition functions are listed herebelow:

- *Forget gate* decides whether the information is going to be forgotten for the current cell unit. Here the recurrent variable $\mathbf{h}_{t-1}$ summarises all historical information and $\mathbf{x}_t$ is the current layer input,

$$f_t^{LSTM} = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f) \tag{11}$$

- *Input gate* determines the new information which is going to be added with

$$\tilde{C}_t^{LSTM} = \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_C), \tag{12}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i), \tag{13}$$

  while $\tilde{C}_t^{LSTM}$ is multiplied by weight coefficients, leading to an update of C_t^{LSTM},

$$C_t^{LSTM} = f_t^{LSTM} \odot C_{t-1}^{LSTM} + \mathbf{i}_t \odot \tilde{C}_t^{LSTM}, \tag{14}$$

  where $\odot$ denotes the Hadamard product of vectors and matrices.
- *Output gate* decides the recurrent state $\mathbf{h}_t$ as a function of previous recurrent output $\mathbf{h}_{t-1}$ and the current layer input $\mathbf{x}_t$ through a Sigmoid activation function, i.e.,

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_o) \tag{15}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(C_t^{LSTM}) \tag{16}$$

Here $\mathbf{W}$ and $\mathbf{b}$ denote the weight and the bias coefficients for different gates respectively. Once the LSTM NN is trained in the latent space, a low dimensional surrogate model can then be established for predicting the evolution of the dynamical system with a low computational cost.

3 Methodology: Generalised Latent Assimilation

Latent Assimilation techniques [7, 8] have been developed for the real-time monitoring of latent surrogate models. Here we have developed a new generalised LA approach which can incorporate observation data encoded in a latent space different from the one of state variables. Since we aim to assimilate a dynamical system, the dependence on time t is introduced for all state/observation variables in the rest of this paper.

3.1 Variational Assimilation Principle

Data assimilation algorithms aim to improve the prediction of some physical fields (or a set of parameters) $\mathbf{x}_t$ based on two sources of information: a prior forecast $\mathbf{x}_{b,t}$ (also known as the background state) and an observation vector $\mathbf{y}_t$. The true state which represents the theoretical value of the current state is denoted by $\mathbf{x}_{\text{true},t}$. In brief, Variational DA searches for an optimal weight between $\mathbf{x}_{b,t}$ and $\mathbf{y}_t$ by minimising the cost function J defined as

$$\begin{aligned} J_t(\mathbf{x}) &= \frac{1}{2}(\mathbf{x}-\mathbf{x}_{b,t})^T\mathbf{B}_t^{-1}(\mathbf{x}-\mathbf{x}_{b,t}) + \frac{1}{2}(\mathbf{y}_t-\mathcal{H}_t(\mathbf{x}))^T\mathbf{R}_t^{-1}(\mathbf{y}_t-\mathcal{H}_t(\mathbf{x}_t)) \\ &= \frac{1}{2}||\mathbf{x}-\mathbf{x}_{b,t}||^2_{\mathbf{B}_t^{-1}} + \frac{1}{2}||\mathbf{y}_t-\mathcal{H}_t(\mathbf{x})||^2_{\mathbf{R}_t^{-1}} \end{aligned} \tag{17}$$

where $\mathcal{H}_t$ denotes the state-observation mapping function, and $\mathbf{B}_t$ and $\mathbf{R}_t$ are the error covariance matrices related to $\mathbf{x}_{b,t}$ and $\mathbf{y}_t$, i.e.,

$$\mathbf{B}_t = \mathrm{Cov}(\epsilon_{b,t}, \epsilon_{b,t}), \quad \mathbf{R}_t = \mathrm{Cov}(\epsilon_{y,t}, \epsilon_{y,t}), \tag{18}$$

where

$$\epsilon_{b,t} = \mathbf{x}_{b,t} - \mathbf{x}_{\mathrm{true},t}, \quad \epsilon_{y,t} = \mathcal{H}_t(\mathbf{x}_{\mathrm{true},t}) - \mathbf{y}_t. \tag{19}$$

Since DA algorithms often deal with problems of large dimension, for the sake of simplicity, prior errors ϵ_b, ϵ_y are often supposed to be centered Gaussian, i.e.,

$$\epsilon_{b,t} \sim \mathcal{N}(0, \mathbf{B}_t), \quad \epsilon_{y,t} \sim \mathcal{N}(0, \mathbf{R}_t). \tag{20}$$

Equation (17), also known as the three-dimensional variational (3D-Var) formulation, represents the general objective function of variational assimilation. Time-dependent variational assimilation (so called 4D-Var) formulation can also be reformulated into Eq. (17) as long as the error of the forward model is not considered. The minimisation point of Eq. (17) is denoted as $\mathbf{x}_{a,t}$,

$$\mathbf{x}_{a,t} = \underset{\mathbf{x}}{\mathrm{argmin}}\Big(J_t(\mathbf{x})\Big), \tag{21}$$

known as the analysis state. When $\mathcal{H}_t$ is non-linear, approximate iterative methods [44] have been widely used to solve variational data assimilation. To do so, one has to compute the gradient $\nabla J(\mathbf{x})$, which can be approximated by

$$\nabla J(\mathbf{x}) \approx 2\mathbf{B}_t^{-1}(\mathbf{x} - \mathbf{x}_{b,t}) - 2\mathbf{H}^T\mathbf{R}_t^{-1}(\mathbf{y}_t - \mathcal{H}_t(\mathbf{x})). \tag{22}$$

In Eq. (22), $\mathbf{H}$ is obtained via a local linearization in the neighbourhood of the current vector $\mathbf{x}$. The minimization of 3D-Var is often performed via quasi-Newton methods, including for instance BFGS approaches [45], where each iteration can be written as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - L_{\text{3D-Var}}\big[\mathrm{Hess}(J)(\mathbf{x}_k)\big]^{-1}\nabla J(\mathbf{x}_k) \tag{23}$$

Here k is the current iteration, and $L_{\text{3D-Var}} > 0$ is the learning rate of the descent algorithm, and

$$\mathrm{Hess}\Big(J(\mathbf{x} = [x_0, \ldots, x_{n-1}])\Big)_{i,j} = \frac{\partial^2 J}{\partial x_i \partial x_j} \tag{24}$$

is the Hessian matrix related to the cost function J. The process of the iterative minimization algorithm is summarised in Algorithm 1.

Variational assimilation algorithms could be applied to dynamical systems for improving future prediction by using a transition operator $\mathcal{M}_{t^k \to t^{k+1}}$ (from time t^k to t^{k+1}), thus

$$\mathbf{x}_{t^{k+1}} = \mathcal{M}_{t^k \to t^{k+1}}(\mathbf{x}_{t^k}). \tag{25}$$

In our study, the $\mathcal{M}_{t^k \to t^{k+1}}$ operator is defined by a latent LSTM surrogate model. Typically in DA, the current background state is often provided by the forecasting from the previous time step, i.e.

$$\mathbf{x}_{b,t^k} = \mathcal{M}_{t^{k-1} \to t^k}(\mathbf{x}_{a,t^{k-1}}). \tag{26}$$

A more accurate reanalysis $\mathbf{x}_{a,t^{k-1}}$ leads to a more reliable forecasting $\mathbf{x}_{b,t^k}$. However, in practice, the perfect knowledge of $\mathcal{M}$ is often out of reach. Recent work of [24] makes use of deep learning algorithms to improve the estimation of $\mathcal{M}_{t^{k-1} \to t^k}$. From Algorithm 1,

Algorithm 1 Iterative minization of 3D-Var cost function via quasi-Newton methods

Inputs: $\mathbf{x}_{b,t}, \mathbf{y}_t, \mathbf{B}_t, \mathbf{R}_t, \mathcal{H}_t$
parameters: $k_{\max}$, ϵ
$\mathbf{x}_0 = \mathbf{x}_b$, $k = 0$

while $k < k_{\max}$ and $||\nabla J_t(\mathbf{x}_k)|| > \epsilon$ **do**
  $J_t(\mathbf{x}_k) = \frac{1}{2}||\mathbf{x}_k - \mathbf{x}_{b,t}||^2_{\mathbf{B}_t^{-1}} + \frac{1}{2}||\mathbf{y}_t - \mathcal{H}_t(\mathbf{x}_k)||^2_{\mathbf{R}_t^{-1}}$
  linearize the $\mathcal{H}_t$ operator in the neighbourhood of $\mathbf{x}_k$
  $\nabla J_t(\mathbf{x}_k) \approx 2\mathbf{B}_t^{-1}(\mathbf{x}_k - \mathbf{x}_{b,t}) - 2\mathbf{H}^T\mathbf{R}_t^{-1}(\mathbf{y}_t - \mathcal{H}_t(\mathbf{x}_k))$
  compute $\mathrm{Hess}\big(J_t(\mathbf{x}_k)\big)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k - L_{\text{3D-Var}}\big[\mathrm{Hess}(J)\mathbf{x}_k\big]^{-1}\nabla J_t(\mathbf{x}_k)$
  $k = k+1$
end while

output: $\mathbf{x}_k$

one observes that the linearization of $\mathcal{H}$ and the evaluation of $\mathrm{Hess}\big(J(\mathbf{x}_k)\big)$ is necessary for variational assimilation. Since in this application, the latent variables and observations are linked via NNs functions, the linearization and the partial derivative calculation are almost infeasible due to:

- the huge number of parameters in the NNs combined with non-linear transformation functions;
- the non-differentiability of NNs functions, for instance, when using activation functions such as ReLu or LeakyReLu [46].

Therefore, we propose the use of a smooth local surrogate function to overcome these difficulties.

3.2 Assimilation with Heterogeneous Latent Spaces

Latent Assimilation techniques are introduced in the very recent work of [7, 8] where the DA is performed after having compressed the state and the observation data into the same latent space. In other words, it is mandatory to have the transformation operator $\tilde{\mathcal{H}}_t = \mathbf{I}$ in the latent space. To fulfil this condition, [7] preprocesses the observation data via a linear interpolation to the full space of the state variables. However, as mentioned in their work, this preprocessing will introduce additional errors, which may impact the assimilation accuracy. More importantly, it is almost infeasible to compress $\mathbf{x}$ and $\mathbf{y}$ into a same latent space in a wide range of DA applications, due to, for instance:

- partial observation: only a part of the state variables are observable, usually in certain regions of the full state space;
- a complex $\mathcal{H}$ function in the full space: $\mathbf{x}$ and $\mathbf{y}$ are different physical quantities (e.g., temperature vs. wind in weather prediction, river flow vs. precipitation in hydrology).

A general latent transformation operator $\tilde{\mathcal{H}}_t$ which links the state and the observation latent spaces can be formulated as

$$\begin{aligned} &\tilde{\mathcal{H}}_t = \mathcal{E}_{\mathbf{y}} \circ \mathcal{H}_t \circ \mathcal{D}_{\mathbf{x}}, \quad \text{i.e.,} \quad \tilde{\mathbf{y}} = \mathcal{E}_{\mathbf{y}} \circ \mathcal{H}_t \circ \mathcal{D}_{\mathbf{x}}(\tilde{\mathbf{x}}) = \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}), \\ &\text{with} \quad \tilde{\mathbf{y}}_t = \mathcal{E}_{\mathbf{y}}(\mathbf{y}_t), \quad \mathbf{x}_t = \mathcal{D}_{\mathbf{x}}(\tilde{\mathbf{x}}_t), \end{aligned} \tag{27}$$

where $\mathcal{E}_{\mathbf{y}}, \mathcal{D}_{\mathbf{x}}$ denote the encoder of the observation vectors and the decoder of the state variables respectively. A flowchart of the generalised LA is illustrated in Fig. 2. The cost

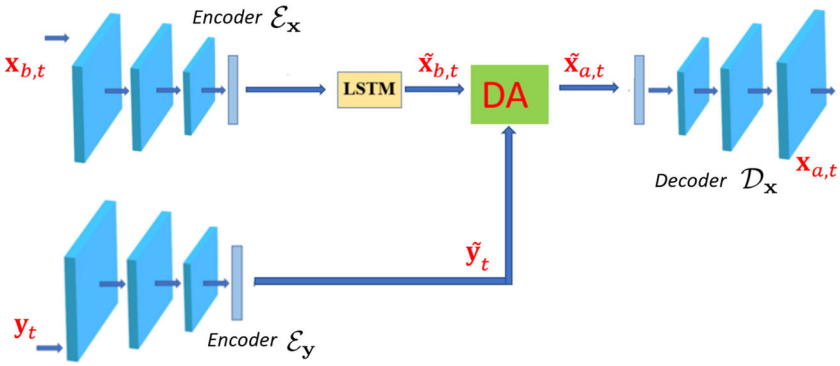


Fig. 2 Flowchart of the LA with heterogeneous latent spaces

function $\tilde{J}_t$ of general LA problems reads

$$\tilde{J}_t(\tilde{\mathbf{x}}) = \frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t})^T \tilde{\mathbf{B}}_t^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t}) + \frac{1}{2}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})^T \tilde{\mathbf{R}}_t^{-1}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})). \tag{28}$$

$$\tilde{\mathbf{x}}_{a,t} = \underset{\tilde{\mathbf{x}}}{\text{argmin}}\Big(\tilde{J}_t(\tilde{\mathbf{x}})\Big). \tag{29}$$

The latent covariance matrices $\tilde{\mathbf{B}}_t$ and $\tilde{\mathbf{R}}_t$ which represent the error covariances in the latent spaces, are defined as

$$\tilde{\mathbf{B}}_t = \text{Cov}(\tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t}, \tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t}), \tag{30}$$

$$\tilde{\mathbf{R}}_t = \text{Cov}(\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathbf{y}}_t, \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathbf{y}}_t). \tag{31}$$

In the rest of this paper, it is supposed that the latent error covariances $\tilde{\mathbf{B}}_t = \tilde{\mathbf{B}}, \tilde{\mathbf{R}}_t = \tilde{\mathbf{R}}$ are time invariant.

3.3 Polynomial Regression for Surrogate Transformation Function

Despite the fact that traditional variational DA approaches can deal with complex $\mathcal{H}$ functions, it is almost impossible to perform descent methods for Algorithm 1 because of the drawbacks described at the end of Sect. 3.1. Our idea consists of building a local smooth and differentiable surrogate function $\tilde{\mathcal{H}}_t^p$ such that

$$\tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}_t^s) \approx \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_t^s) \quad \text{for} \quad \tilde{\mathbf{x}}_t^s \text{ in a neighbourhood of } \tilde{\mathbf{x}}_{b,t}. \tag{32}$$

It is important to note that the computation of $\tilde{\mathcal{H}}^p$ will also depend on the value of the latent variable $\tilde{\mathbf{x}}$. The approximate cost function can then be written as

$$\tilde{J}_t^p(\tilde{\mathbf{x}}) = \frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t})^T \tilde{\mathbf{B}}^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_b) + \frac{1}{2}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}))^T \tilde{\mathbf{R}}^{-1}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})). \tag{33}$$

The way of computing the surrogate function makes crucial impact on both the accuracy and the computational cost of DA since the $\tilde{\mathcal{H}}$ function may vary a lot with time for chaotic dynamical systems. From now, we denote $\tilde{\mathcal{H}}_t$ and $\tilde{\mathcal{H}}_t^p$, the latent transformation function at time t and the associated surrogate function. For time variant $\tilde{\mathcal{H}}_t$ and $\mathbf{x}_t$, the computation of $\tilde{\mathcal{H}}_t^p$ must be performed online. Thus the choice of local surrogate modelling approach

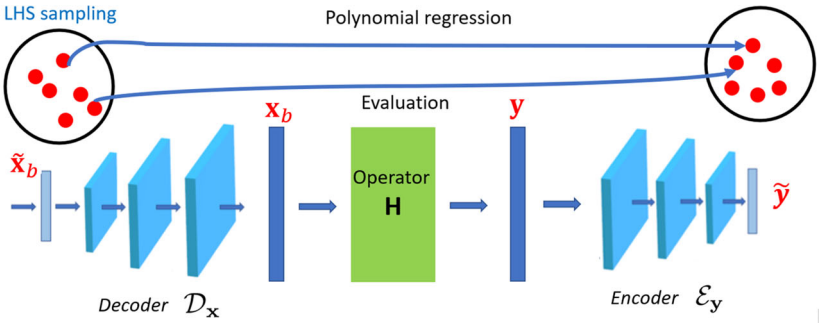


Fig. 3 Flowchart of the polynomial-based local surrogate model in latent assimilation

should be a tradeoff of approximation accuracy and computational time. As mentioned in the Introduction of this paper, the idea of computing local surrogate model has been developed in the field of interpretable AI. Linear regression (including Lasso, Ridge) and simple ML models such as DT are prioritised for the sake of interpretability (e.g., [33]). In this study, the local surrogate function is built via polynomial regression since our main criteria are smoothness and differentiability. Compared to other approaches, employing PR in LA has several advantages in terms of smoothness and computing efficiency.

To perform the local PR, we rely on local training datasets $\{\tilde{\mathbf{x}}^q_{b,t}\}_{q=1..n_s}$ generated randomly around the current background state $\tilde{\mathbf{x}}_{b,t}$ since the true state is out of reach. The sampling is performed using Latin Hypercube Sampling (LHS) to efficiently cover the local neighbourhood homogeneously [47]. Other sampling techniques, such as Gaussian perturbation, can also be considered regarding the prior knowledge of the dynamical system. We then fit the output of the transformation operator by a local polynomial function,

$$\tilde{\mathcal{H}}^p_t = \underset{p\in P(d_p)}{\operatorname{argmin}} \left(\sum_{q=1}^{n_s} ||p(\mathbf{x}^q_{b,t}) - \mathcal{H}_t(\mathbf{x}^q_{b,t})||_2^2 \right)^{1/2}, \tag{34}$$

where $P(d_p)$ represents the set of polynomial functions of degree d_p. We then evaluate the $\tilde{\mathcal{H}}_t$ function to generate the learning targets of local PR as shown in Fig. 3. The pipeline of the LA algorithms for dynamical models is summerised in Algorithm 2, where $\tilde{\mathcal{M}}$ denotes the forward operator in the latent space. In the context of this paper, $\tilde{\mathcal{M}}$ is the latent LSTM surrogate model. When using a sequence-to-sequence prediction, the forecasting model can be accelerated in the sense that a sequence of future background states can be predicted by one evaluation of LSTM. The PR degree, the sampling range and the sampling size are denoted as d_p, r_s and n_s respectively. These parameters affect considerably the performance of Generalised LA. Their values should be chosen carefully as shown later in Sect. 5.2.

3.4 Theoretical Analysis of the Loss Function

The accuracy of the surrogate model with LA depends on a variety of different uncertainties, including the ROM error, the RNN error, the observation error, the minimization error of DA, and the approximation error of GLA. In this section, we focus on the latter which is induced by the approximation using local polynomial functions in our new model.

Algorithm 2 Generalised LA with local polynomial surrogate function

Inputs: $\tilde{\mathbf{x}}_{b,0}$, $\{\mathbf{y}_t\}$, $\mathcal{E}_y$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{R}}$, $\tilde{\mathcal{H}}$, $\tilde{\mathcal{M}}$
paramters: d_p, r_s, n_s, T
$\tilde{\mathbf{x}}_0 = \tilde{\mathbf{x}}_b$, $k = 0$
for $t = 0$ to T **do**
  $\tilde{\mathbf{x}}_{b,t} = \tilde{\mathcal{M}}\big(\tilde{\mathbf{x}}_{t-1}, \tilde{\mathbf{x}}_{t-2}, \tilde{\mathbf{x}}_{t-3}, \dots\big)$
  if $\mathbf{y}_t$ is available **then**
    $\tilde{\mathbf{y}}_t = \mathcal{E}_y(\mathbf{y}_t)$
    $\{\tilde{\mathbf{x}}^q_{b,t}\}_{q=1..n_s} = \text{LHS}_{\{d^p, r_s, n_s\}}(\tilde{\mathbf{x}}_{b,t})$
    for $q = 0$ to n_s **do**
      $\tilde{\mathbf{y}}^q_t = \tilde{\mathcal{H}}(\tilde{\mathbf{x}}^q_{b,t})$
    end for
    $\tilde{\mathcal{H}}^p_t = \text{PR}_{\text{train}}\big(\text{input:}\{\tilde{\mathbf{x}}^q_{b,t}\}, \text{output:}\{\tilde{\mathbf{y}}^q_t\}, q = 1..n_s\big)$
    optional: $\{\tilde{\mathbf{x}}^q_{\text{test}}\}_{q=1..n_s} = \text{LHS Sampling}_{\{d^p, r_s, n_s\}}(\tilde{\mathbf{x}}_{b,t})$

    optional: $\epsilon^p_{r-rmse} = \sqrt{\frac{1}{n_s}\sum_q^{n_s}\left(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}^q_{test}) - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}}^q_{\text{test}})||^2 / ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}^q_{test})||^2\right)}$
    $\tilde{\mathbf{x}}_{a,t} = \underset{\tilde{\mathbf{x}}}{\text{argmin}}\Big(\frac{1}{2}||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t}||^2_{\tilde{\mathbf{B}}^{-1}} + \frac{1}{2}||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}})||^2_{\tilde{\mathbf{R}}^{-1}}\Big)$
    $\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{a,t}$
  else
    $\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{b,t}$
  end if
end for

Objective We aim to provide a theoretical upper bound for the expected absolute and relative approximation error evaluated on the true state, i.e.,

$$\mathbb{E}\Big(J^p_t(\tilde{\mathbf{x}}_{\text{true},t}) - J_t(\tilde{\mathbf{x}}_{\text{true},t})\Big) \quad \text{and} \quad \frac{\mathbb{E}\Big(J^p_t(\tilde{\mathbf{x}}_{\text{true},t}) - J_t(\tilde{\mathbf{x}}_{\text{true},t})\Big)}{\mathbb{E}\Big(J_t(\tilde{\mathbf{x}}_{\text{true},t})\Big)}. \tag{35}$$

Assumptions Following assumptions are made in this section,

1. Both background and observation prior errors follow a centred Gaussian distribution [5] and the prior error covariances are perfectly specified, i.e.,

$$\tilde{\epsilon}_{b,t} = \tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t} \sim \mathcal{N}(0, \tilde{\mathbf{B}}), \quad \tilde{\epsilon}_{y,t} = \tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) \sim (0, \tilde{\mathbf{R}}). \tag{36}$$

2. For simplicity, all the activation functions in the NNs are supposed to be Rectified Linear Unit (ReLu).

Analysis In fact, the difference between $J_t(\tilde{\mathbf{x}})$ and $J^p_t(\tilde{\mathbf{x}})$ for any point $\tilde{\mathbf{x}}$ in the space can be bounded as

$$J^p_t(\tilde{\mathbf{x}}) = \frac{1}{2}\Big(||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t}||^2_{\tilde{\mathbf{B}}^{-1}} + ||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) + \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}})||^2_{\tilde{\mathbf{R}}^{-1}}\Big) \tag{37}$$

$$\leq \frac{1}{2}\Big(||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t}||^2_{\tilde{\mathbf{B}}^{-1}} + ||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||^2_{\tilde{\mathbf{R}}^{-1}} + ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}})||^2_{\tilde{\mathbf{R}}^{-1}} + 2||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}} \cdot ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}}\Big) \tag{38}$$

$$\leq \frac{1}{2}\Big(J_t(\tilde{\mathbf{x}}) + ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}})||^2_{\tilde{\mathbf{R}}^{-1}}\Big) + ||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}} \cdot ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}^p_t(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}}. \tag{39}$$

We are interested in the expectation value of the loss function evaluated on the true state, i.e., $\mathbb{E}(J_t^p(\tilde{\mathbf{x}}_{\text{true},t}))$. Following Eq. (39),

$$\mathbb{E}\big(J_t^p(\tilde{\mathbf{x}}_{\text{true},t})\big) \leq \mathbb{E}\big(J_t(\tilde{\mathbf{x}}_{\text{true},t})\big) + \frac{1}{2}\mathbb{E}\big(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}_{\text{true},t})||^2_{\tilde{\mathbf{R}}^{-1}}\big) + \mathbb{E}\big(||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}} \cdot ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}}\big). \tag{40}$$

Following Eq. (36) ,

$$\sqrt{\tilde{\mathbf{B}}^{-1}}(\tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t}) \sim \mathcal{N}(0, \mathbf{I}_{\dim(\tilde{\mathbf{x}})}), \quad \sqrt{\tilde{\mathbf{R}}^{-1}}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})) \sim \mathcal{N}(0, \mathbf{I}_{\dim(\tilde{\mathbf{y}})}). \tag{41}$$

Here we remind that by definition, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{R}}$ are real constant symmetric positive definite matrices thus $\sqrt{\tilde{\mathbf{B}}^{-1}}$ and $\sqrt{\tilde{\mathbf{R}}^{-1}}$ are well-defined.

$$\mathbb{E}(||\tilde{\mathbf{x}}_{\text{true},t} - \tilde{\mathbf{x}}_{b,t}||^2_{\tilde{\mathbf{B}}^{-1}}) = \mathbb{E}\left((\tilde{\mathbf{x}}_{\text{true},t} - \tilde{\mathbf{x}}_{b,t})^T \tilde{\mathbf{B}}^{-1}(\tilde{\mathbf{x}}_{\text{true},t} - \tilde{\mathbf{x}}_{b,t})\right) \tag{42}$$

$$= \mathbb{E}\left(\left(\sqrt{\tilde{\mathbf{B}}^{-1}}(\tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t})\right)^T \cdot \left(\sqrt{\tilde{\mathbf{B}}^{-1}}(\tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t})\right)\right) \tag{43}$$

$$= \mathbb{E}\left(||\sqrt{\tilde{\mathbf{B}}^{-1}}(\tilde{\mathbf{x}}_{b,t} - \tilde{\mathbf{x}}_{\text{true},t})||_2^2\right) \tag{44}$$

$$= \dim(\tilde{\mathbf{x}}) \tag{45}$$

For the same reason, $\mathbb{E}(||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||^2_{\tilde{\mathbf{R}}^{-1}}) = \dim(\tilde{\mathbf{y}}_t)$. One can then deduce

$$\mathbb{E}\big(J_t(\tilde{\mathbf{x}}_{\text{true},t})\big) = \dim(\tilde{\mathbf{x}}_t) + \dim(\tilde{\mathbf{y}}_t). \tag{46}$$

A similar reasoning via Mahalanobis norm can be found in the work of [48].

Now we focus on the other terms of Eq. (40). In fact, the observation error $||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}}$ is only related to instrument noises or representation error if the encoder error can be neglected. On the other hand, the approximation error $||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}}$ is only related to polynomial regression where the real observation vector $\mathbf{y}$ is not involved. Therefore, we can suppose that $||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}}$ is uncorrelated to $||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}}$. This assumption will be proved numerically in experiments. One can further deduce that,

$$\mathbb{E}\big(||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}} \cdot ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}}\big) = \mathbb{E}\big(||\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}}\big) \cdot \mathbb{E}\big(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}_{\text{true},t})||_{\tilde{\mathbf{R}}^{-1}}\big) = 0. \tag{47}$$

Now we only need to bound the polynomial regression error. For this, we rely on the recent theoretical results in the work of [37], which proves that for learning a teacher NNs via polynomial regression,

$$N^* = d^{O(L/\epsilon^*)^L} \quad \text{for the ReLU activation function,} \tag{48}$$

where N^* is the required number of samples in the training dataset, d is the input dimension, L is the number of NNs layers and ϵ^* is the relative target prediction error (i.e., in our case $\epsilon = \left(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_2 / ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_2\right) \leq \epsilon^*$). Since we are looking for a bound of the

regression error ϵ,

$$N^* = d^{\left(c(L/\epsilon^*)^L\right)} \quad \text{where} \quad c \text{ is a real constant} \tag{49}$$

$$\Leftrightarrow \quad \log_d N^* = c(L/\epsilon^*)^L \tag{50}$$

$$\Leftrightarrow \quad \left(\frac{\log_d N^*}{c}\right)^{1/L} = L/\epsilon^* \tag{51}$$

$$\Leftrightarrow \quad \epsilon \leq \epsilon^* = L\left(\frac{c}{\log_d N^*}\right)^{1/L} \tag{52}$$

$$\Leftrightarrow \quad ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_2 \leq L\left(\frac{c}{\log_d N^*}\right)^{1/L} ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_2. \tag{53}$$

Now that we have a relative bound of the polynomial prediction error in the L^2 norm, we want to extend this boundary to the matrix norm $||.||_{\tilde{\mathbf{R}}^{-1}}$. For this we use a general algebraic result:

$$\forall a \in \mathbb{R}^{\dim(a)}, \quad \mathbf{C}_{p,d} \in \mathbb{R}^{\dim(a)\times\dim(a)} \quad \text{is a symmetric positive definite matrix then}$$
$$\sqrt{\lambda_{\min}}||a||_2 \leq ||a||_{\mathbf{C}_{p,d}} \leq \sqrt{\lambda_{\max}}||a||_2 \tag{54}$$

where $\lambda_{\min}, \lambda_{\max}$ represent the smallest and the largest eigenvalues of $\mathbf{C}_{p,d}$ respectively. Since $\mathbf{C}_{p,d}$ is positive definite, $0 < \lambda_{\min} \leq \lambda_{\max}$. We denote $0 < \lambda^{\tilde{\mathbf{R}}}_{\dim(\tilde{\mathbf{y}})} \leq \cdots \leq \lambda^{\tilde{\mathbf{R}}}_1$ the eigenvalues of $\tilde{\mathbf{R}}$. Thus the eigenvalues of $\tilde{\mathbf{R}}^{-1}$ are $0 < 1/\lambda^{\tilde{\mathbf{R}}}_1 \leq \cdots \leq 1/\lambda^{\tilde{\mathbf{R}}}_{\dim(\tilde{\mathbf{y}})}$. Following the result of Eq. (54),

$$||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_2 \leq \sqrt{\lambda^{\tilde{\mathbf{R}}}_1}||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}} \quad \text{and}$$
$$||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_2 \geq \sqrt{\lambda^{\tilde{\mathbf{R}}}_{\dim(\tilde{\mathbf{y}})}}||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}}. \tag{55}$$

Therefore, we can deduce from Eq. (53) that

$$||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}} \leq \sqrt{\lambda^{\tilde{\mathbf{R}}}_1/\lambda^{\tilde{\mathbf{R}}}_{\dim(\tilde{\mathbf{y}})}}\, L\left(\frac{c}{\log_d N^*}\right)^{1/L} ||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})||_{\tilde{\mathbf{R}}^{-1}}. \tag{56}$$

Thus,

$$\mathbb{E}\left(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t}) - \tilde{\mathcal{H}}_t^p(\tilde{\mathbf{x}}_{\text{true},t})||^2_{\tilde{\mathbf{R}}^{-1}}\right)$$
$$= \text{cond}(\mathbf{R})L^2\left(\frac{c}{\log_d N^*}\right)^{2/L}\mathbb{E}\left(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||^2_{\tilde{\mathbf{R}}^{-1}}\right), \tag{57}$$

where $\text{cond}(\mathbf{R}) = \lambda^{\tilde{\mathbf{R}}}_1/\lambda^{\tilde{\mathbf{R}}}_{\dim(\tilde{\mathbf{y}})}$ is the condition number of the $\mathbf{R}$ matrix. Combining Eqs. (40), (47) and (57),

$$\mathbb{E}\left(J_t^p(\tilde{\mathbf{x}}_{\text{true},t})\right)$$
$$\leq \mathbb{E}\left(J_t(\tilde{\mathbf{x}}_{\text{true},t})\right) + \frac{1}{2}\text{cond}(\mathbf{R})L^2\left(\frac{c}{\log_d N^*}\right)^{2/L}\mathbb{E}\left(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||^2_{\tilde{\mathbf{R}}^{-1}}\right)$$
$$= \dim(\tilde{\mathbf{x}}_t) + \dim(\tilde{\mathbf{y}}_t) + \frac{1}{2}\text{cond}(\mathbf{R})L^2\left(\frac{c}{\log_d N^*}\right)^{2/L}\mathbb{E}\left(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||^2_{\tilde{\mathbf{R}}^{-1}}\right). \tag{58}$$

Therefore we have an upper bound of $\mathbb{E}\left(J_t^p(\tilde{\mathbf{x}}_{\text{true},t})\right)$ and $\mathbb{E}\left(J_t^p(\tilde{\mathbf{x}}_{\text{true},t})\right) - \mathbb{E}\left(J_t(\tilde{\mathbf{x}}_{\text{true},t})\right)$ which doesn't depend on the local polynomial surrogate model $\tilde{\mathcal{H}}_t^p$. An upper bound for the relative

error can also be found, i.e.,

$$\frac{\mathbb{E}\big(J_t^p(\tilde{\mathbf{x}}_{\text{true},t}) - J_t(\tilde{\mathbf{x}}_{\text{true},t})\big)}{\mathbb{E}\big(J_t(\tilde{\mathbf{x}}_{\text{true},t})\big)} \leq \frac{\text{cond}(\mathbf{R})L^2\Big(\frac{c}{\log_d N^*}\Big)^{2/L}\mathbb{E}\big(||\tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}_{\text{true},t})||^2_{\tilde{\mathbf{R}}^{-1}}\big)}{2(\dim(\tilde{\mathbf{x}}) + \dim(\tilde{\mathbf{y}}))}. \tag{59}$$

Furthermore, in the case where the target NNs is fixed and we have infinite local training data for the polynomial surrogate model,

$$\mathbb{E}\big(J_t^p(\tilde{\mathbf{x}}_{\text{true},t}) - J_t(\tilde{\mathbf{x}}_{\text{true},t})\big) \xrightarrow{N^* \to +\infty} 0. \tag{60}$$

This result obtained is consistent with the Stone-Weierstrass theorem which reveals the fact that every continuous function defined on a closed interval can be approximated as closely as desired by a polynomial function [49]. The proof in this section is made by assuming all the activation functions are ReLu in the NNs. This analysis can be extended to other activation functions, such as sigmoid, based on the recent results in [37].

4 Results: ROM and RNN Approaches

In this section, we describe the test case of an oil-water two-phase flow CFD simulation, used for numerical comparison of different ML surrogate models and LA approaches.

4.1 CFD Modelling

Liquid-liquid two-phase flows are widely encountered in many industrial sectors, including petroleum, chemical and biochemical engineering, food technology, pharmaceutics, and so on. In crude-oil pipelines or oil recovery equipment, both dispersed and separated oil-water flows can be observed, and the transition between these flow patterns can impact the operating cost and safety. Therefore, fundamental understanding of the oil-water flow behavior in pipelines has been tackled for a long term with various efforts from theoretical, experimental, and simulating perspectives. However, it is not fully solved yet due to the complexity of the multiphase flow characteristics. Even for a very simple case of the separating process of oil droplets in water in a horizontal pipeline, it is still challenging to predict the separation length and layer height distribution. Although there are a lot of experimental data, the prediction of such kind of flow regime transition is still poor due to the limited understanding of the underlying physics.

The experiment in this study is conducted in the flow rig developed by [50]. The flow pipe consists of a front and a back leg with equal length of 4 m and a uniform diameter of 26 mm as shown in Fig. 4. The two legs are connected by a U-bend. Measurements are conducted in the front leg only, and High-speed imaging, combined with Particle Image Velocimetry and Laser Induced Fluorescence experiments are carried out to study the drop evolution, velocity profiles and flow patterns. As shown in Table 2, the two test cases explored in this work have initial mixture velocity of 0.52 m/s and 1.04 m/s respectively. The average oil inlet volume fraction of both simulations is set to 30%. The first simulation (i.e., the one with $U_m = 0.52$ m/s) is used to train the surrogate model while the second one is used latter to test the performance of ROMs. The simulations are validated against experimental data of the concentration profiles and layer heights. The simulations adopt the same physical properties and operating parameters as those in the experiment. The related parameters are shown in Tables 1 and 2.

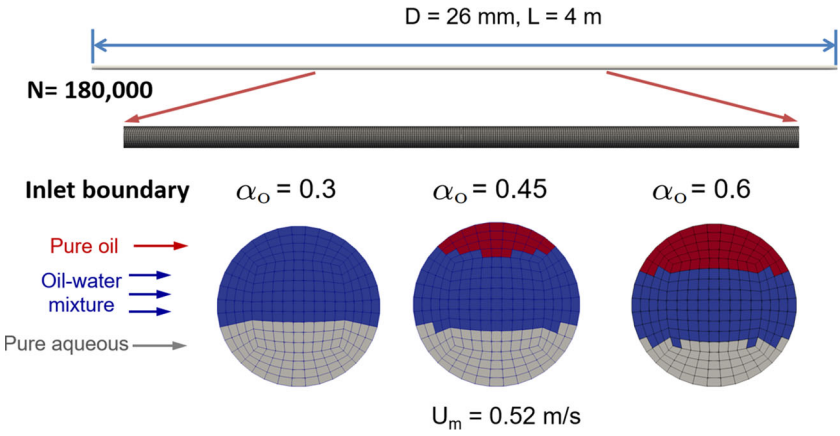


Fig. 4 Dimension and parameters of the pipe and the two-phase flow

Table 1 Physical properties of the experimental system

Liquid	Phase	ρ (kg m^{-3})	μ (Pa s)	σ (N m^{-1})
Water	Aqueous	998	0.89×10^{-3}	~ 0.0329
Exxsol D140	Organic	828	5.5×10^{-3}	

Table 2 Operating parameters of the experiment

α_o	$h_{C0}^{+} = h_{C0}/D$	$h_{O0}{}^{+} = h_{O0}/D$	$h_{P0}{}^{+} = h_{P0}/D$	d_{320} (mm)	U_m(m s^{-1})
0.3	0.405	0.997	0.76	3.41	0.52
0.3	0.189	0.997	0.92	1.27	1.04

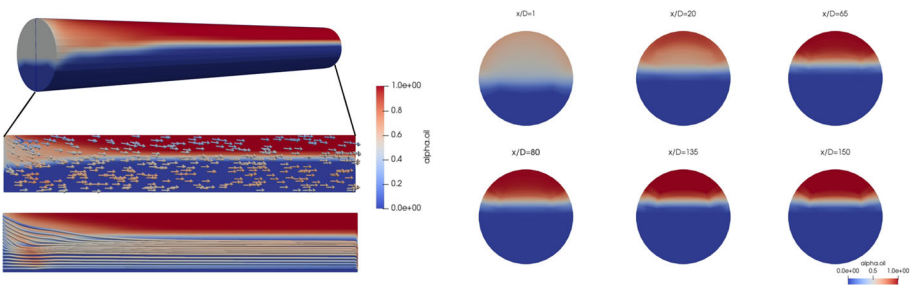


Fig. 5 CFD modelling of the two-phase flow

The CFD simulation (as illustrated in Fig. 5) aims to study the flow separation characteristics. The two-phase flow of silicone oil and water in a pipe with a length of 4m and a diameter of 26 mm is studied. Eulerian–Eulerian simulations are performed through the opensource CFD platform of OpenFOAM (version 8.0), and population balance models [51] are used to model the droplet size and coalescence behaviour. The governing equations of the Eulerian

framework are given as below:

$$\frac{\partial}{\partial t}\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\right)+\nabla\cdot\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\right)=0, \tag{61}$$

$$\frac{\partial}{\partial t}\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\right)+\nabla\cdot\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\right)=-\alpha_{\mathrm{k}}\nabla p+\nabla\cdot\left(\alpha_{\mathrm{k}}\boldsymbol{\tau}_{\mathrm{k}}\right)+\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{g}+\boldsymbol{M}_{\mathrm{k}}, \tag{62}$$

where the subscript of k represents the phases of water and oil respectively, and $\boldsymbol{\tau}$ is the stress tensor expressed as

$$\boldsymbol{\tau}_{\mathrm{k}}=\mu_{\mathrm{eff}}\left[\nabla\boldsymbol{U}_{\mathrm{k}}+\left(\nabla\boldsymbol{U}_{\mathrm{k}}\right)^{\mathrm{T}}-\frac{2}{3}\left(\nabla\cdot\boldsymbol{U}_{\mathrm{k}}\right)I\right]. \tag{63}$$

A structured mesh with 180000 nodes is generated by the utility of blockMesh, and the volume concentration at the inlet boundary is prescribed by the patch manipulation (the utility of *createPatch* in OpenFOAM.). In all cases, the mixture $k-\epsilon$ model and wall functions are used to model turbulence equations. In order to obtain a steady flow pattern, the flow time is set to 10 s. The time step is 0.005 s for all the cases, which ensures the convergence at the current mesh resolution. The running time is 40 h on a four-nodes parallel computing mode. The computing nodes harness an Intel® Xeon(R) CPU E5-2620 (2.00 GHz, RAM 64 GB). Finally, snapshots of oil concentration $\boldsymbol{\alpha}_t$ and velocities $\mathbf{V}_{x,t}, \mathbf{V}_{y,t}, \mathbf{V}_{z,t}$ in the x, y, z axes respectively (i.e., $\boldsymbol{U}_{k,t} = [V_{x,t}, V_{y,t}, V_{z,t}]$) can be generated from the CFD model to describe the two-phase flow dynamics. In this study, we are interested in building a machine learning surrogate model for predicting the evolution of $\boldsymbol{\alpha}_t$ along the test section. The training of autoencoders and LSTM is based on 1000 snapshots (i.e., every 0.01 s) as described in Sect. 4.2.

4.2 Numerical Results of Latent Surrogate Modelling

In this section, we compare different latent surrogate modelling techniques, including both ROM and RNN approaches in the CFD application described in Sect. 4.1.

4.2.1 ROM Reconstruction

We first compare the performance of the different autoencoding approaches introduced in Sect. 2.1. The single-trajectory simulation data of 1000 snapshots in total are split into a training (including validation) dataset with 80% of snapshots and a test dataset with the remaining 20% snapshots. Following the setup in [7], the data split is performed homogeneously where the four snapshots between two consecutive test snapshots are used for training. In other words, the test dataset contains the snapshots $\{\boldsymbol{\alpha}_4, \boldsymbol{\alpha}_9, \boldsymbol{\alpha}_{14}, \ldots, \boldsymbol{\alpha}_{999}\}$. Since we are dealing with cylindrical meshes and the length of the pipe (4 m) is much larger than its diameter (26 mm), we decide to first flatten the snapshots to 1D vectors before auto-encoding as shown in Fig. 6.

POD The distribution of the eigenvalues respectively for $\boldsymbol{\alpha}$, normalised $\mathbf{V}_x$, normalised $\mathbf{V}_y$ and normalised $\mathbf{V}_z$ is shown in Fig. 7 while the compression accuracy γ and rate ρ, as defined in Eq. (8), are displayed in Table 3 for the truncation paramater $q = 30$. In this application, POD exhibits a high compression accuracy with an extremely low compression rate on the training data set issued from one CFD simulation. The performance on the test dataset will be further examined in Sect. 4.2.1.

1D CAE Since the meshes have an unsquared structure and the pipe's length is much larger than the diameter, we decide to proceed with 1D CAE. As pointed out by [52], the ordering

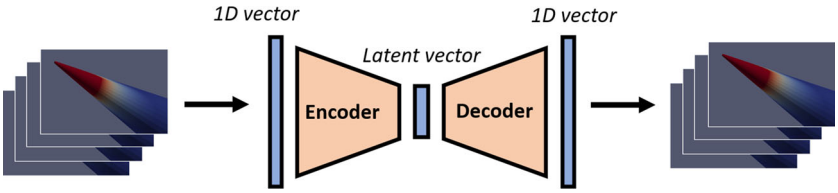


Fig. 6 Encoder–decoder modelling for the two-phase flow in the pipe

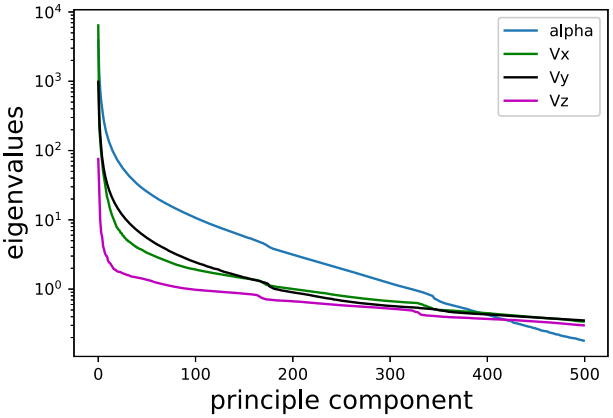


Fig. 7 Eigenvalues for $\boldsymbol{\alpha}$, $\mathbf{V}_x$, $\mathbf{V}_y$ and $\mathbf{V}_z$ on the training set, issued from one simulation

Table 3 Compression accuracy γ and rate ρ with truncation parameter $q = 30$ for $\boldsymbol{\alpha}$, $\mathbf{V}_x$, $\mathbf{V}_y$ and $\mathbf{V}_z$

Field	$\boldsymbol{\alpha}$	$\mathbf{V}_x$	$\mathbf{V}_y$	$\mathbf{V}_z$
γ	99.76%	99.99%	99.81%	96.40%
ρ	1.66×10^{-5}	1.66×10^{-5}	1.66×10^{-5}	1.66×10^{-5}

of points is crucial in CNN algorithms especially for problems with non-square meshes. Denoting $\mathcal{Z} = \{z_1, z_2, ..z_{n_z}\}$ the ensemble of nodes in the mesh structure, their links can be represented by the Adjacency matrix $\mathbf{A}^z$ defined as

$$\mathbf{A}^z_{i,j} = \begin{cases} 1 & \text{if } z_i \text{ is connected to } z_j \\ 0 & \text{otherwise.} \end{cases} \tag{64}$$

In this study, when we flatten the 3D meshes to a 1D vector, the corresponding adjacency matrix contains many non-zero values outside the diagonal band as shown in Fig. 8a. In other words, when applying 1D CNN, the edges $\mathbf{A}^z_{i,j}$ represented by the non-zero values in the adjacency matrix can not be included in the same convolutional window thus the information of these links will be lost during the encoding. This is a common problem when dealing with unstructured or non-square meshes [17, 19]. Much effort has been devoted to finding the optimum ordering of sparse matrices for reducing the matrix band [53, 54]. In this work, we make use of the Cuthill-McKee algorithm [55] based on ideas from graph theory, which is proved to be efficient for dealing with symmetric sparse matrices. The adjacency matrix for the reordered nodes is shown in Fig. 8b where all non-zero elements are included in the diagonal band of width 10. We then perform the 1D CNN based on these reordered nodes. The exact NNs structure of this 1D CAE can be found in Table 4.

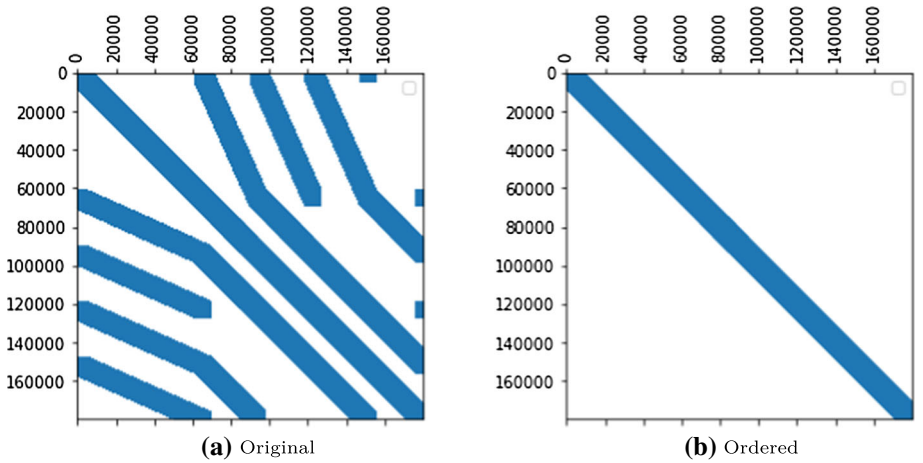


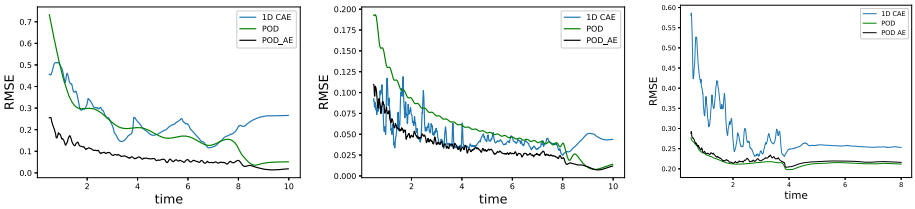
Fig. 8 Adjacency matrices before (**a**) and after (**b**) mesh reordering

Table 4 NN structure of the CAE with ordered meshes

Layer (type)	Output shape	Activation
Encoder		
Input	(180000, 1)	
Conv 1D (8)	(180000, 4)	ReLu
Droppout (0.2)	(180000, 4)	
MaxPooling 1D (5)	(36000, 4)	
Conv 1D (8)	(36000, 4)	ReLu
Droppout (0.2)	(36000, 4)	
MaxPooling 1D (5)	(7200, 4)	
Conv 1D (8)	(7200, 1)	LeakyReLu (0.2)
AveragePooling 1D (5)	(1440, 1)	
Flatten	720	
Dense (30)	30	ReLu
Decoder		
Input	30	
Flatten (720)	720	
Conv 1D (8)	(720, 1)	ReLu
Upsampling (10)	(7200, 1)	
Conv 1D (8)	(7200, 4)	ReLu
Upsampling (5)	(36000, 4)	
Conv 1D (8)	(36000, 4)	LeakyReLu (0.2)
Upsampling (5)	(180000, 1)	

Table 5 NN structure of the POD AE

Layer (type)	Output shape	Activation
Encoder		
Input	799	
Dense (128)	128	LeakyReLu(0.3)
Dense (30)	30	LeakyReLu(0.3)
Decoder		
Input	30	
Dense 128	128	LeakyReLu(0.3)
Dense 799	799	LeakyReLu(0.3)



(a) latent dimension 5 **(b)** latent dimension 30 **(c)** latent dimension 30

Fig. 9 Comparison of reconstruction errors of oil concentration α using different auto-encoder approaches. Figures **a**, **b** are evaluated on the simulation data of $U_m = 0.52$ (i.e., the first row of Table 2) while figure **c** is evaluated on the simulation data of $U_m = 0.52$ (i.e., the second row of Table 2)

POD AE We first apply the POD operators to obtain the full set of PCs of $\boldsymbol{\alpha}$, $\mathbf{V}_x$, $\mathbf{V}_y$ and $\mathbf{V}_z$ respectively as described in Sect. 2.1.1. Since 20% of the snapshots are used for testing, we obtain 799 PCs for each variable. Then the auto-encoding of $\boldsymbol{\alpha}$, $\mathbf{V}_x$, $\mathbf{V}_y$ and $\mathbf{V}_z$ to compressed latent variables $\tilde{\boldsymbol{\alpha}}$, $\tilde{\mathbf{V}}_x$, $\tilde{\mathbf{V}}_y$ and $\tilde{\mathbf{V}}_z$ is performed individually with the same NNs structure as displayed in Table 5. The training is very efficient for POD AE so much that it can be easily performed on a laptop CPU in less than 15 min. On the other hand, 1D CAE training takes several hours if training with the full set of snapshots.

Numerical comparison The relative mean square error (RMSE) for the oil concentration α of different ROM reconstructions is illustrated in Fig. 9 on the CFD simulations. The first simulation (Fig. 9a, b) includes both training (80%) and test (20%) data while the second simulation (Fig. 9c) consists of purely unseen test data. In order to further inspect the ROM accuracy against the dimension of the latent space (i.e., the truncation parameter), we show in Fig. 9 the performance for both $q = 5$ (a) and $q = 30$ (b,c). It can be clearly observed that the POD and 1D CAE (with reordered nodes) are out-performed by POD AE in terms of both average accuracy and temporal robustness for the first CFD simulation data. For all ROM approaches, a higher dimension of the latent space ($5 \longrightarrow 30$) can significantly enhance the reconstruction. In the case of POD AE, the RMSE has been reduced from around 10% to around 3%. We thus choose to use the POD AE approach for computing the latent surrogate model in this work. As expected, the RMSE evaluated on the second simulation dataset is larger than the first one. In Fig. 9c, the POD and POD AE show a better generalizability compared to the 1D CAE, which confirms our choice of POD AE in this application.

4.2.2 LSTM Surrogate Model

In this study, instead of classical many-to-one LSTM setting (e.g., [1, 7]), we make use of a sequence-to-sequence LSTM structure to speed up the evaluation of the surrogate model. More precisely, in lieu of a single time output, the LSTM predicts a time series of latent variables with an internal connection according to the time steps. For more details about sequence-to-sequence LSTM, interested readers are referred to the work of [56]. The recent work of [57] shows that incremental LSTM which forecasts the difference between output and input variables can significantly improve the accuracy and efficiency of the learning procedure, especially for multiscale and multivariate systems. Therefore, we have adapted the incremental LSTM in the sequence-to-sequence learning with

- LSTM input: $\mathbf{u}_{\text{input}} = [\tilde{\mathbf{x}}_t, \tilde{\mathbf{x}}_{t+1}, \ldots, \tilde{\mathbf{x}}_{t+l_{\text{input}}-1}]$,
- LSTM output: $\mathbf{u}_{\text{output}} = [\tilde{\mathbf{x}}_{t+l_{\text{input}}} - \tilde{\mathbf{x}}_{t+l_{\text{input}}-1}, \tilde{\mathbf{x}}_{t+l_{\text{input}}+1} - \tilde{\mathbf{x}}_{t+l_{\text{input}}}, \ldots, \tilde{\mathbf{x}}_{t+l_{\text{input}}+l_{\text{output}}-1} - \tilde{\mathbf{x}}_{t+l_{\text{input}}+l_{\text{output}}-2}]$,

where l_{input} and l_{output} denote the length of the input and the output sequences respectively. $\tilde{\mathbf{x}}_t$ represents the latent vector encoded via the POD AE approach at time step t. The training data is generated from the simulation snapshots by shifting the beginning of the input sequence as shown in Fig. 10. Similar to the setup of AEs, 80% of input and output sequences are used as training data while the remaining 20% are divided into the test dataset. In this work, we implement two LSTM models where the first one includes only the encoded concentration (i.e., $\tilde{\boldsymbol{\alpha}}$) and the second one uses both concentration and velocity variables (i.e., $\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{V}}_x, \tilde{\mathbf{V}}_y, \tilde{\mathbf{V}}_z$) as illustrated in Fig. 10. We set $l_{\text{intput}} = l_{\text{output}} = 30$ for the joint LSTM model (i.e., the one including the velocity data), meaning that 33 iterative applications of LSTM are required to predict the whole CFD model. On the other hand, the single concentration model is trained using a LSTM 10to10 (i.e., $l_{\text{intput}} = l_{\text{output}} = 10$) since the instability of the single model doesn't support long range predictions, which will be demonstrated later in this section. For clarity, in the rest of this paper, single and joint models refer to

- Single model: LSTM 10to10 predictive model based on encoded concentration $\tilde{\boldsymbol{\alpha}}$
- Joint model: LSTM 30to30 predictive model based on encoded concentration and velocity variables $\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{V}}_x, \tilde{\mathbf{V}}_y, \tilde{\mathbf{V}}_z$

The exact NNs structure of the joint LSTM model is shown in table 7 where the sequence-to-sequence learning is performed. On the other hand, the single conceration model is implemented thanks to the *RepeatVector* layer. The reconstructed principle components via LSTM prediction (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$ following the notation in Sect. 2.1.3) against compressed ground truth (i.e., $\mathbf{L}_{\mathbf{x}}^T(\mathbf{x})$) are shown in Figs. 11 and 12. As observed in Fig. 12, the latent prediction is accurate until around 200 time steps (2 s) for all eigenvalues. However, a significant divergence can be observed just after $t = 2$ s for most principal components due to the accumulation of prediction error. On the other hand, the joint LSTM model with similar NNs structures exhibits a much more robust prediction performance despite that some temporal gap can still be observed. The reconstructed prediction of oil concentration $\boldsymbol{\alpha}$ at $t = 7$ s (i.e. $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_{t=700}^{\text{predict}})$), together with the CFD simulation of $\boldsymbol{\alpha}_{t=700}$ are illustrated in Fig. 13. The joint LSTM model predicts reasonably well the CFD simulation with a slight delay of the oil dynamic while the prediction of the single LSTM model diverges at $t = 7$ s. These results are coherent with our analysis of Figs. 11 and 12. In summary, although the objective here is to build a surrogate model for simulating the oil concentration, it is demonstrated numerically that more physics information can improve the prediction performance. The computational time of both LSTM surrogate models (on a Laptop CPU) and CFD (with parallel computing

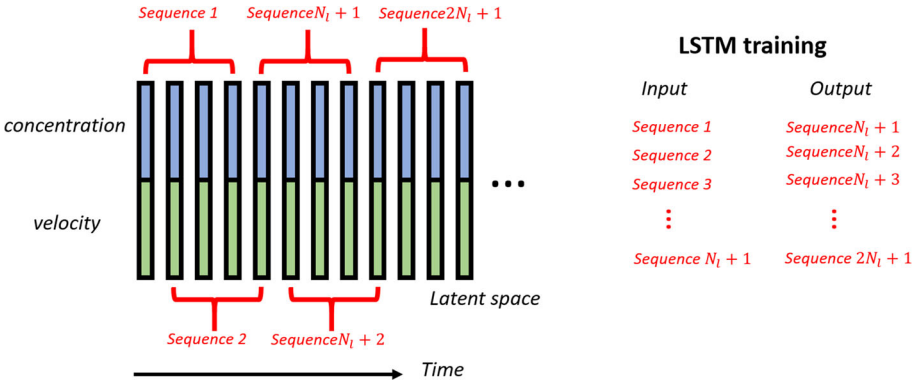
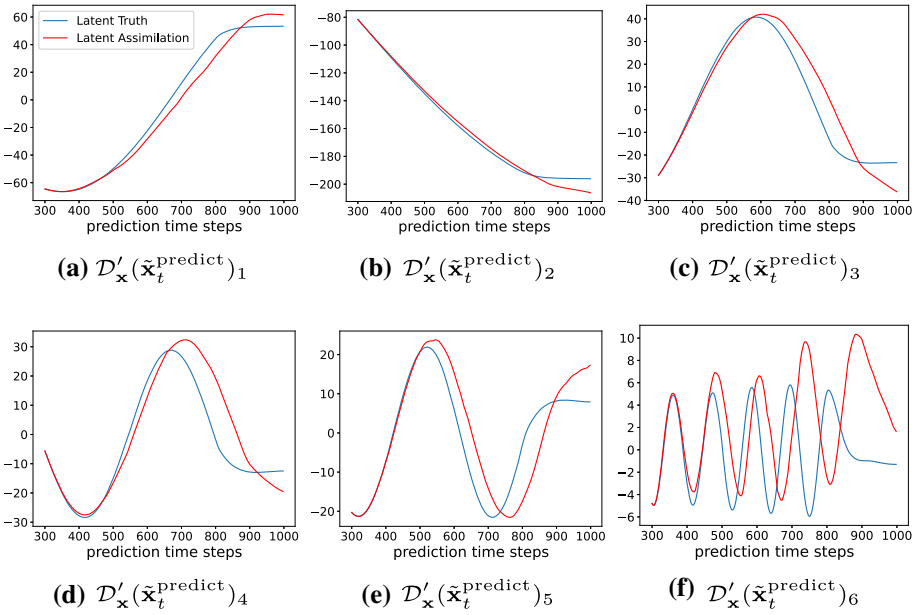


Fig. 10 LSTM training in the latent space for a joint model of concentration and velocity



(a) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_1$ **(b)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_2$ **(c)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_3$

(d) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_4$ **(e)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_5$ **(f)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_6$

Fig. 11 The LSTM prediction of reconstructed POD coefficients (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) with joint LSTM 30to30 surrogate model

mode) approaches for the entire simulation is illustrated in table 6. For both LSTM models the online prediction takes place from t=1s (100th time step) until t = 10 s (1000th time step) where the first 100 time steps of exact encoded latent variables are provided to 'warm up' the prediction system. From table 6, one observes that the online computational time of LSTM surrogate models is around 1000 times shorter compared to the CFD. Table 6 also reveals the fact that a longer prediction sequence in sequence-to-sequence LSTM can significantly reduce the online prediction complexity. As shown in Table 6, the offline computation of both approaches is also very fast thanks to the training efficiency of POD AE (Table 7).

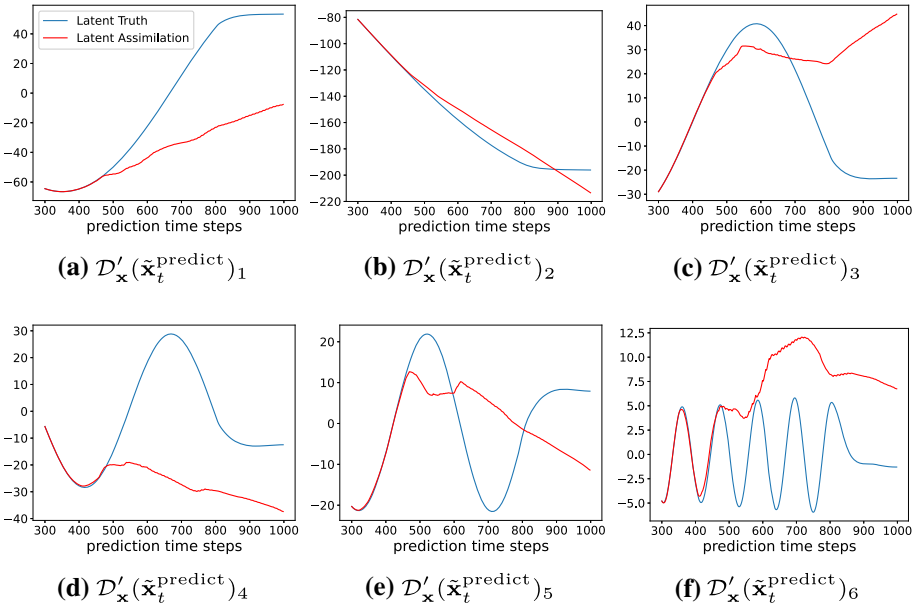


Fig. 12 The LSTM prediction of reconstructed POD coefficients (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) with single LSTM 10to10 surrogate model

Table 6 Computational time of LSTM surrogate models and CFD modelling

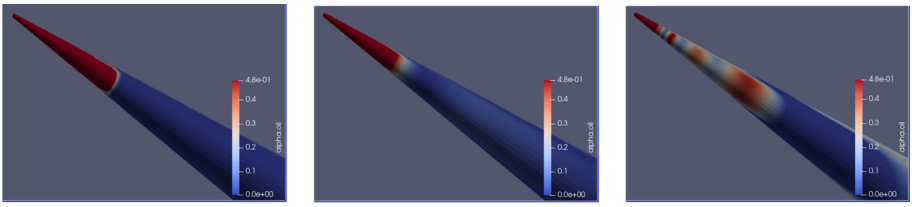
	LSTM 10to10 (s)	LSTM 30to30(s)	CFD
Offline time	1426	1597	
Online time	175	124	$\approx$ 40 h

Table 7 LSTM structure in POD AE latent space for the single model (only concentration) and the joint model (concentration and velocity)

Layer (type)	Output shape single model	Output shape joint model	Activation
Input	(30, 30)	(30, 120)	
LSTM	50	200	Sigmoid
RepeatVector	(30, 50)	(30, 200)	
LSTM	(30, 100)	(30, 200)	ReLu
Dense	(30, 200)	(30, 200)	ReLu
Time distributed	(30, 30)	(30, 120)	LeakyReLu

5 Results: GLA Approach

In this section, we test the performance of the novel generalised latent assimilation algorithm on the CFD test case of oil-water two-phase flow. The strength of the new approach proposed in this paper compared to existing LA methods, is that DA can be performed with heterogeneous latent spaces for state and observation data. In this section, we evaluate the algorithm performance using randomly generated observation function $\mathcal{H}$ in the full space.



(a) CFD **(b)** LSTM joint **(c)** LSTM single

Fig. 13 The original CFD simulation against LSTM predictions at $t = 7$ s

5.1 Non-linear Observation Operators

In order to evaluate the performance of the novel approach, we work with different synthetically generated non-linear observation vectors for LA. Since we would like to remain as general as possible, we prefer not to set a particular form of the observation operator, which could promote some space-filling properties. For this purpose, we decide to model the observation operator with a random matrix $\mathbf{H}$ acting as a binomial selection operator. The full-space transformation operator $\mathcal{H}$ consists of the selection operator $\mathbf{H}$ and a marginal non-linear function $f_{\mathcal{H}}$. Each observation will be constructed as the sum of a few true state variables randomly collected over the subdomain. In order to do so, we introduce the notation for a subset sample $\left\{\mathrm{x}_t^*(i)\right\}_{i=1\ldots n_{\text{sub}}}$ randomly but homogeneously chosen (with replacement) with probability P among the available data set $\{\mathrm{x}_t(k)\}_{k=1\ldots n=180000}$. The evaluations of the $f_{\mathcal{H}}$ function on the subsets (i.e., $f_{\mathcal{H}}(\mathrm{x}_t^*)$) are summed up and the process is re-iterated $m \in \{10000, 30000\}$ times in order to construct the observations:

$$y_t(j) = \sum_{i=1}^{n_j} f_{\mathcal{H}}(\mathrm{x}_t^*(i)), \quad \text{for } j = 1, \ldots, m, \tag{65}$$

where the size n_j (invariant with time) of the collected sample used for each jth observation data point $y_t(j)$ is random and by construction follows a binomial distribution $\mathcal{B}(n, P)$. As for the entire observation vector,

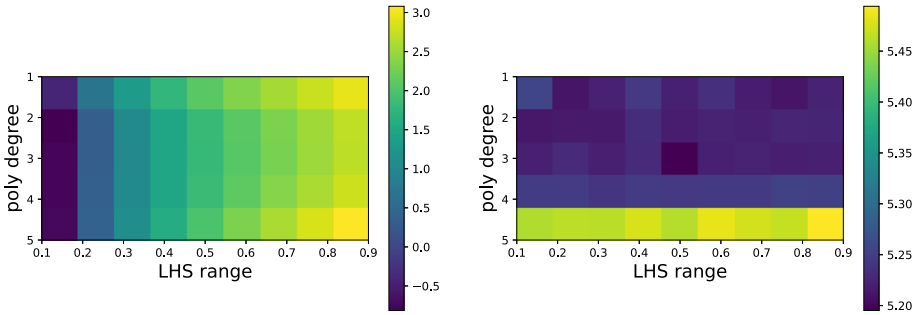
$$\mathbf{y}_t = \begin{bmatrix} y_t(0) \\ y_t(1) \\ \vdots \\ y_t(m-1) \end{bmatrix} = \mathcal{H}(\mathbf{x}_t) = \mathbf{H} f_{\mathcal{H}}(\mathbf{x}_t) = \begin{bmatrix} \mathbf{H}_{0,0}, \ldots \mathbf{H}_{0,n-1} \\ \vdots \\ \mathbf{H}_{m-1,0}, \ldots \mathbf{H}_{m-1,n-1} \end{bmatrix} \begin{bmatrix} f_{\mathcal{H}}(x_t(0)) \\ f_{\mathcal{H}}(x_t(1)) \\ \vdots \\ f_{\mathcal{H}}(x_t(n-1)) \end{bmatrix}$$

$$\text{with} \quad \mathbf{H}_{i,j} = \begin{cases} 0 & \text{with probability} \quad 1-P \\ 1 & \text{with probability} \quad P \end{cases}. \tag{66}$$

Using randomly generated selection operator for generating observation values is commonly used for testing the performance of DA algorithms (e.g., [58, 59]). In this work we choose a sparse representation with $P = 0.1\%$. Once $\mathbf{H}$ is randomly chosen, it is kept fixed for all the numerical experiments in this work. Two marginal non-linear functions $f_{\mathcal{H}}$ are employed in this study:

- quadratic function: $f_{\mathcal{H}}(x) = x^2$
- reciprocal function: $f_{\mathcal{H}}(x) = 1/(x + 0.5)$.

After the observation data is generated based on Eq. (66), we apply the POD AE approach to build an observation latent space of dimension 30 with associated encoder $\mathcal{E}_y$ and decoder $\mathcal{D}_y$.



(a) log(RMSE) **(b)** log(training time(s))

Fig. 14 Logarithm of RMSE in the test dataset (evaluated on 1000 points) and the training time in seconds

In this application, the dimension of the observation latent space is chosen as 30 arbitrarily. In general, there is no need to keep the same dimension of the latent state space and the latent observation space. Following Eqs. (27) and (66), the state variables $\tilde{\mathbf{x}}_t$ and the observations $\tilde{\mathbf{y}}_t$ in LA can be linked as:

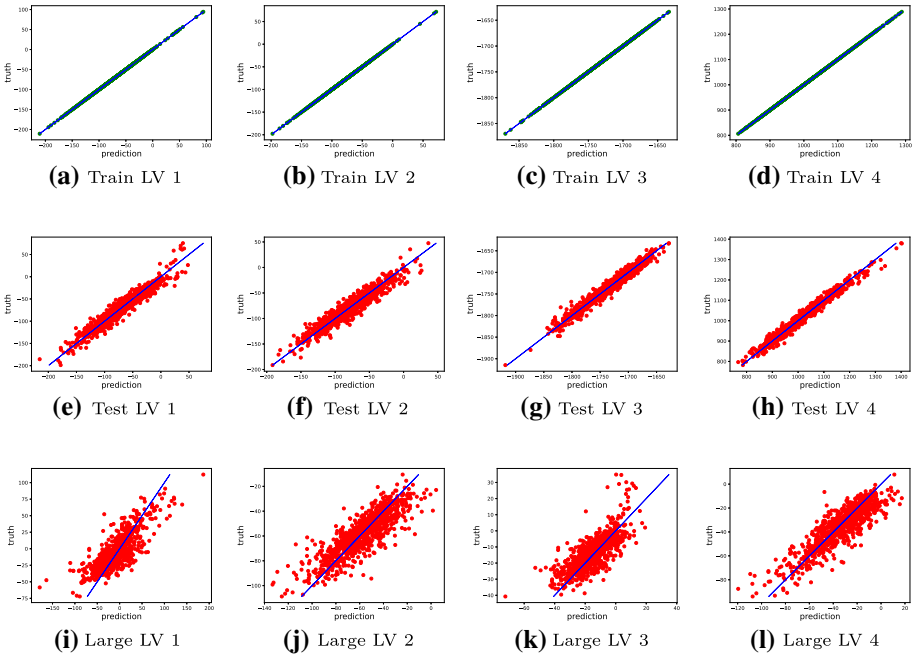
$$\tilde{\mathbf{y}}_t = \tilde{\mathcal{H}}(\tilde{\mathbf{x}}_t) = \mathcal{E}_{\mathbf{y}} \circ \mathbf{H} \circ f_{\mathcal{H}} \circ \mathcal{D}_{\mathbf{x}}(\tilde{\mathbf{x}}_t). \tag{67}$$

5.2 Numerical Validation and Parameter Tuning

Local polynomial surrogate functions are then used to approximate the transformation operator $\tilde{\mathcal{H}} = \mathcal{E}_{\mathbf{y}} \circ \mathbf{H} \circ f_{\mathcal{H}} \circ \mathcal{D}_{\mathbf{x}}$ in Latent Assimilation. In order to investigate the PR accuracy and perform the hyper-parameters tuning, we start by computing the local surrogate function at a fixed time step $t = 3$ s with $(\tilde{\mathbf{x}}_{300}, \tilde{\mathbf{y}}_{300})$. Two LHS ensembles $\{\tilde{\mathbf{x}}^q_{\text{train}}\}_{q=1..1000}$ and $\{\tilde{\mathbf{x}}^q_{\text{test}}\}_{q=1..1000}$, each of 1000 sample vectors, are generated for training and validating local PR respectively. As mentioned previously in Sect. 3.2, the polynomial degree d^p and the LHS range r_s are two important hyper-parameters which impacts the surrogate function accuracy. r_s also determines the expectation of the range of prediction errors in the GLA algorithm. For hyper-parameters tuning, we evaluate the root-mean-square-error (RMSE) (of $\{\tilde{\mathbf{x}}^q_{\text{test}}\}_{q=1..1000}$) and the computational time of local PR with a range of different parameters, i.e.,

$$\begin{aligned} &\{\tilde{\mathbf{x}}^q_{\text{train}}\}_{q=1..1000} / \{\tilde{\mathbf{x}}^q_{\text{test}}\}_{q=1..1000} = \text{LHS Sampling}_{\{d^p, r_s, 1000\}}(\tilde{\mathbf{x}}_{t=300}) \\ &\text{for} \quad d^p \in \{1, \ldots, 5\} \quad \text{and} \quad r_s \in \{10\%, \ldots, 90\%\}. \end{aligned} \tag{68}$$

The results are presented in Fig. 14 with a logarithmic scale for both RMSE and computational time (in seconds). Here the quadratic function is chosen as the transformation operator to perform the tests. Figure 14a reveals that there is a steady rise of RMSE against LHS ranger_s . This fact shows the difficulties of PR predictions when the input vector is far from the LHS center (i.e., $\tilde{\mathbf{x}}_{300}$) due to the high non-linearity of NNs functions. The PR performance for $d^p = 2, 3, 4$ on the test dataset $\{\tilde{\mathbf{x}}^q_{\text{test}}\}_{q=1..1000}$ is more robust compared to linear predictions (i.e., $d^p = 1$), especially when the LHS range grows. However, a phenomenon of overfitting can be noticed when $d^p \geq 5$ where an increase of prediction error is noticed. One has to make a tradeoff between prediction accuracy and application range when choosing the value of r_s. In general, PR presents a good performance with a relative low RMSE (with an upper bound of $e^3 = 20.08$) given that $||\tilde{\mathbf{x}}_{t=300}||_2 = 113.07$. As for the



(a) Train LV 1 **(b)** Train LV 2 **(c)** Train LV 3 **(d)** Train LV 4

(e) Test LV 1 **(f)** Test LV 2 **(g)** Test LV 3 **(h)** Test LV 4

(i) Large LV 1 **(j)** Large LV 2 **(k)** Large LV 3 **(l)** Large LV 4

Fig. 15 Latent variable prediction results in the training (**a**–**d**) and test (**e**–**l**) datasets against the true values with the polynomial degree $d^p = 4$. The LHS sampling range is $r_s = 30\%$ for a–h and $r_s = 60\%$ for i–l

computational time of a local PR, it stays in the same order of magnitude for different set of parameters (from $e^{5.2} \approx 181$ s to $e^{5.5} \approx 244$ s) where the cases of $d^p = 1, 2, 3, 4$ are extremely close. Considering the numerical results shown in Fig. 14 and further experiments in Latent Assimilation, we fix the parameters as $d^p = 4$ and $r_s = 0.3$ in this application. The PR prediction results against the compressed truth in the latent space are shown in Fig. 15 for 4 different latent observations. What can be clearly seen is that the local PR can fit very well the $\tilde{\mathcal{H}}$ function in the training dataset (Fig. 15a–d) while also provides a good prediction of unseen data (Fig. 15e–h), which is consistent with our conclusion in Fig. 14. When the sampling range increases in the test dataset (Fig. 15i–l), it is clear that the prediction start to perform less well. This represents the case where we have under-estimated the prediction error by 100% (i.e., $r_s = 30\%$ for training and $r_s = 60\%$ for testing). The required number of samples (i.e., $n_s = 1000$) is obtained by offline experiments performed at $(\mathbf{x}_{300}, \mathbf{y}_{300})$. For different polynomial degrees $d_p \in \{1, 2, 3, 4, 5\}$, no significant improvement in terms of prediction accuracy on the test dataset can be observed when the number of samples $n_s > 1000$. We have also performed other experiments at different time steps (other than $t = 3$ s) and obtained similar results qualitatively.

5.3 Generalised Latent Assimilation

In this section, we illustrate the experimental results of performing variational Generalised LA with the POD AE reduced-order-modelling and the LSTM surrogate model. The loss functions in the variational methods are computed thanks to the local polynomial surro-

gate functions. The obtained results are compared with CFD simulations both in the low dimensional basis and the full physical space.

5.3.1 GLA with a Quadratic Operator Function

Following the setup in Sect. 5.1, the full-space observation operator is computed with a binomial random selection matrix $\mathbf{H}$ and quadratic marginal equation $f_{\mathcal{H}}(x) = x^2$ as shown in Eq. (66). Separate POD AEs (i.e., $\mathcal{E}_{\mathbf{y}}$ and $\mathcal{D}_{\mathbf{y}}$) are trained for encoding the observation data. The prediction of the LSTM surrogate model start at $t = 3$ s, i.e., the 300th time step. Since the prediction of the joint model is made using a 30 to 30 LSTM, the LA takes place every 1.5 s starting from 5.7 s for 30 consecutive time steps each time. In other words, the LA takes place at time steps 570 to 599, 720 to 749 and 870 to 899, resulting in 90 steps of assimilations among 700 prediction steps. As for the 10to10 single concentration LSTM model, since the prediction accuracy is relatively mediocre as shown in Fig. 12, more assimilation steps are required. In this case the LA takes place every 0.6 s starting from 5 s for 10 consecutive time steps each time, leading to 180 in total. For the minimization of the cost function in the variational LA (Eq. (33)), Algorithm 2 is performed with the maximum number of iterations $k_{max} = 50$ and the tolerance $\epsilon = 0.05$ in each assimilation window. Identifying the error covariances in the latent space is challenging. Current LA methods either make use of diagonal matrices [7, 60] or perform ensemble-type [8, 25, 61] data assimilation to estimate $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{R}}$. The latter can be computationally difficult for complex and highly non-linear transformation functions. Thus in this test case, identity matrices are chosen as latent covariance matrices. To increase the importance of observation data, the error covariance matrices in Algorithm 1 are fixed as:

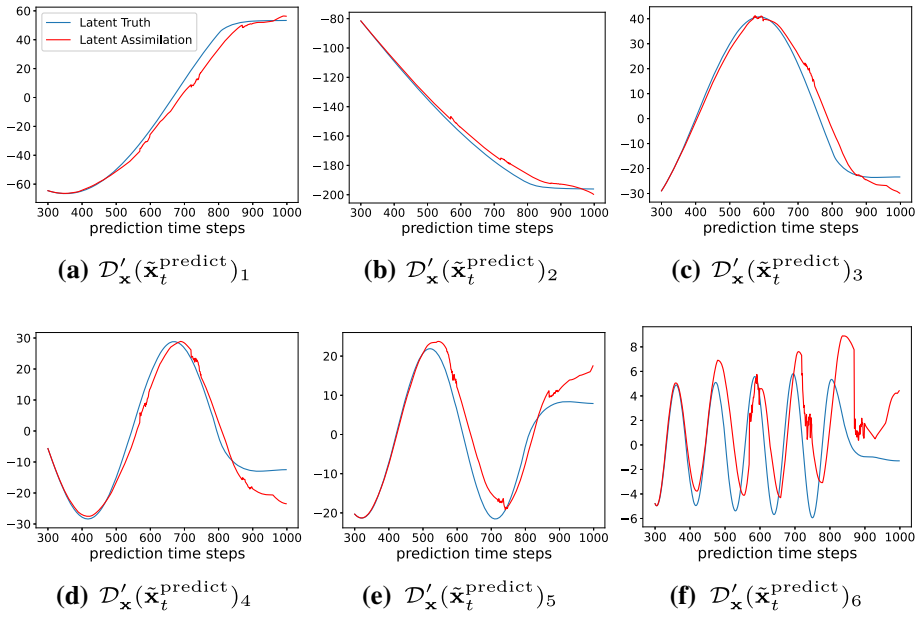
$$\tilde{\mathbf{B}} = \mathbf{I}_{30} \quad \text{and} \quad \tilde{\mathbf{R}} = 0.1 \times \mathbf{I}_{30}, \tag{69}$$

where $\mathbf{I}_{30}$ denotes the identity matrix of dimension 30. In this particular test, no artificial error has been added to synthetic observations. However, encoding observation data from the full space to the latent space will inevitably create compression errors. These noises can be included in the DA process through the modelling of the $\mathbf{R}$ matrix [62, 63].

The Latent assimilation of reconstructed principle components (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) against the compressed ground truth is illustrated in Figs. 16 and 17 for the joint and single LSTM surrogate model respectively. The red curves include both prediction and assimilation results starting at $t = 3$ s (i.e., 300th time step). What can be clearly observed is that, compared to pure LSTM predictions shown in Figs. 11 and 12, the mismatch between predicted curves and the ground truth (CFD simulation) can be considerably reduced by the novel generalised LA technique, especially for the single LSTM model. As for the joint LSTM surrogate model (Fig. 16), the improvement is significant for $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_4$, $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_5$, and $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_6$. These results show that the novel approach can well incorporate real-time observation data with partial and non-linear transformation operators that the state-of-the-art LA can not handle. Prediction/assimilation mismatch in the full physical space will be discussed later in Sect. 5.3.3.

5.3.2 GLA with a Reciprocal Operator Function

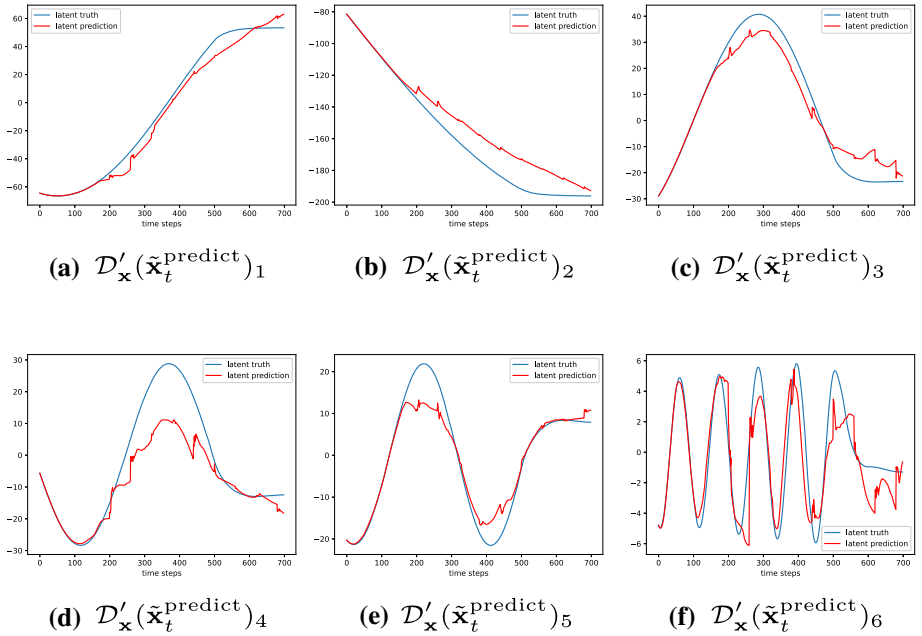
Here we keep the same assimilation setup as in Sect. 5.3.1 in terms of assimilation accuracy and error covariances specification. Instead of a quadratic function, the observation data are generated using the reciprocal function $f_{\mathcal{H}}(x) = 1/(x + 0.5)$ in the full space as described



(a) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_1$ **(b)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_2$ **(c)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_3$

(d) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_4$ **(e)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_5$ **(f)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_6$

Fig. 16 The LA of reconstructed POD coefficients (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) with joint LSTM 30to30 surrogate model and quadratic observation function. Results of the same experiment without GLA is shown in Fig. 11



(a) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_1$ **(b)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_2$ **(c)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_3$

(d) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_4$ **(e)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_5$ **(f)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_6$

Fig. 17 The LA of reconstructed POD coefficients (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) with single LSTM 10to10 surrogate model and quadratic observation function. Results of the same experiment without GLA is shown in Fig. 12

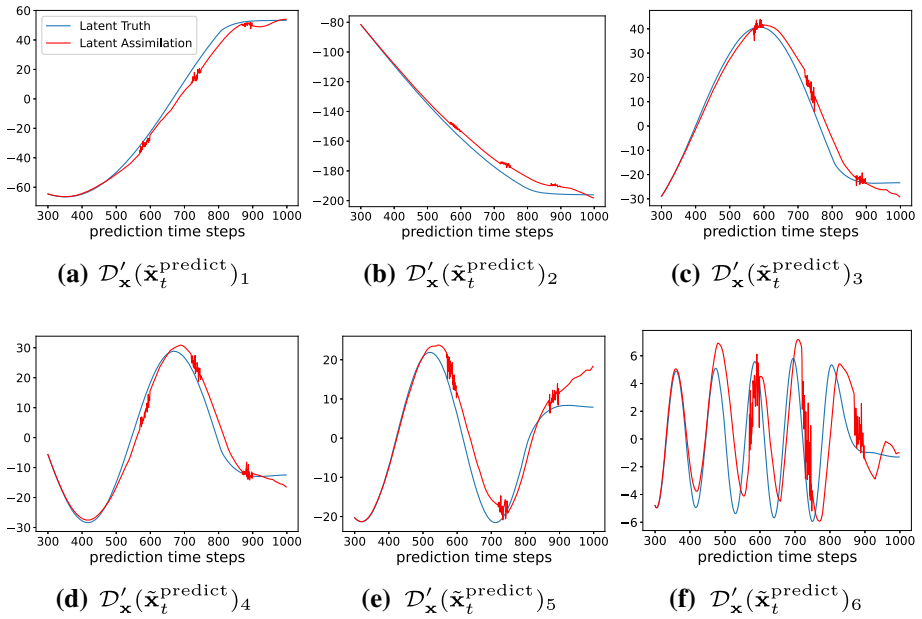


Fig. 18 The LA of reconstructed POD coefficients (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) with joint LSTM 30to30 surrogate model and reciprocal observation function

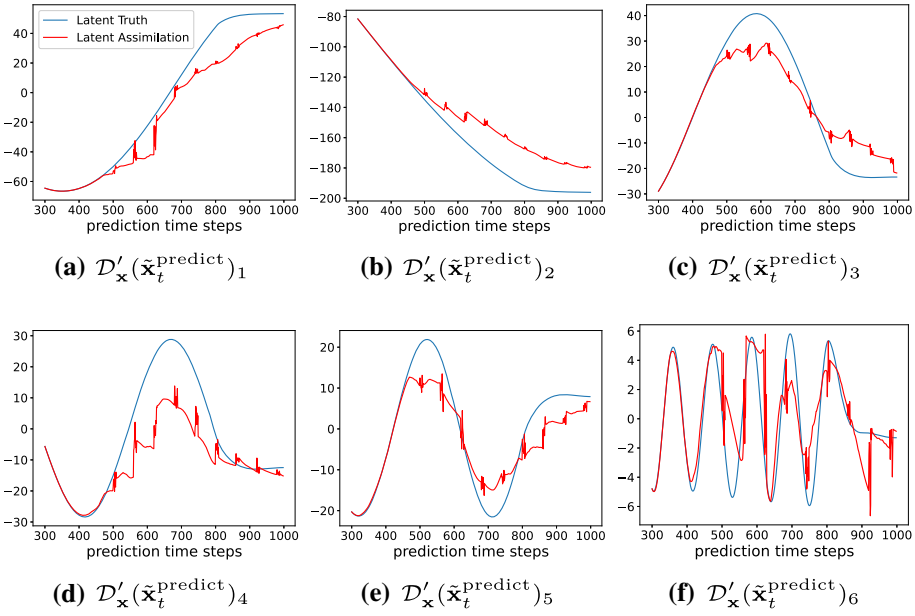
in Sect. 5.1. Therefore, new autoencoders are trained to compress the observation data for $\boldsymbol{\alpha}_t, \mathbf{V}_{x,t}, \mathbf{V}_{y,t}, \mathbf{V}_{z,t}$ to latent spaces of dimension 30. The results of predicted/assimilated POD coefficients $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$ are shown in Figs. 18 and 19. Similar conclusion can be drawn as in Sect. 5.3.1, that is, the generalised LA approach manages to correctly update the LSTM predictions (for both joint and single models) on a consistent basis. Some non-physical oscillatory behaviours can be observed in Figs. 16, 17, 18 and 19. This is due to the application of LA which modified the dynamics in the latent space. Comparing the assimilated curves using quadratic and reciprocal observation functions, the latter is slightly more chaotic due to the fact that reciprocal functions, when combined with DL encoder–decoders (as shown in Fig. 3) can be more difficult to learn for local polynomial surrogate functions.

5.3.3 Prediction Error in the Latent and the Full Space

In this section, we illustrate the evolution of the global prediction/assimilation errors and the forecasting of the global physical field based on the results obtained in Sects. 5.3.1 and 5.3.2. The relative L_2 error in the space of the principle components and the full space of the concentration, i.e.,

$$\frac{||\mathbf{L}_{\mathbf{x}}^T\boldsymbol{\alpha}_t - \mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})||_2}{||\mathbf{L}_{\mathbf{x}}^T\boldsymbol{\alpha}_t||_2} \quad \text{and} \quad \frac{||\boldsymbol{\alpha}_t - \mathbf{L}_{\mathbf{x}}\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})||_2}{||\boldsymbol{\alpha}_t||_2}, \tag{70}$$

for both joint and single models is shown in Fig. 20. The evolution of the relative error in the global space is consistent with our analysis in Figs. 16, 17, 18 and 19 for decoded POD coefficients. The LA with quadratic (in red) and reciprocal (in green) observation operators can significantly reduce the relative error as compared to the original LSTM model (in blue). More



(a) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_1$ **(b)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_2$ **(c)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_3$

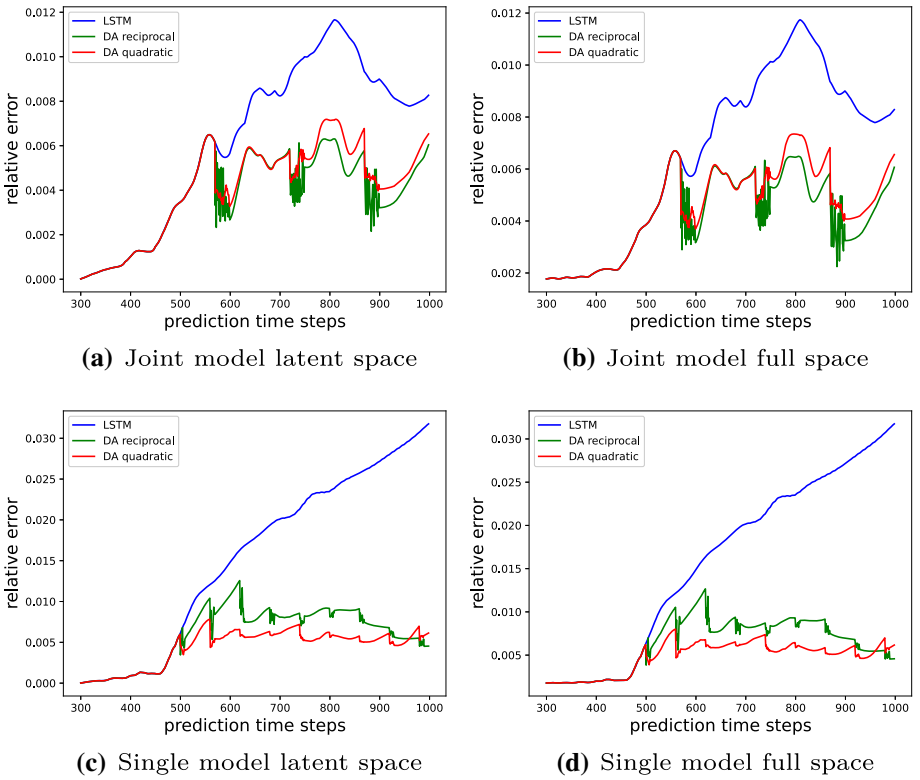
(d) $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_4$ **(e)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_5$ **(f)** $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})_6$

Fig. 19 The LA of reconstructed POD coefficients (i.e., $\mathcal{D}'_{\mathbf{x}}(\tilde{\mathbf{x}}_t^{\text{predict}})$) with single LSTM 10to10 surrogate model and reciprocal observation function

importantly, the DA does not only impact the estimation of current time steps, it improves also future predictions after assimilation, thus demonstrating the stability of the proposed approach. The prediction error in the latent space and the full physical space share very similar shapes for both single and joint models, showing that the ROM reconstruction errors are dominated by the LSTM prediction error. The reconstructed model prediction/assimilation in the full space at $t = 7$ s is shown in Fig. 21. Compared to Fig. 13, the prediction of the single LSTM model (Fig. 21a, b) can be greatly improved with an output much more realistic and closer to the CFD simulation (Fig. 13a). As for the joint model, the initial delay of the oil dynamic can also be well corrected thanks to the variational LA approach despite some noises can still be observed. In summary, the novel LA technique with local polynomial surrogate function manages to improve the current assimilation reconstruction, and more importantly future predictions of latent LSTM. The averaged computational time of GLA on a laptop CPU for one time step is presented in Table 8. Little difference can be found between quadratic and reciprocal marginal functions. The optimization of Eq. (33) is implemented using the ADAO [64] package where the maximum number of iterations and the stopping tolerance of the BFGS algorithm are fixed as 50 and 0.01, respectively.

6 Conclusion and Future Work

Performing DA with simulation and observation data encoded in heterogeneous latent spaces is an important challenge since background and observation quantities are often different in real DA scenarios. On the other hand, it is extremely difficult, if not infeasible, to apply directly classical variational DA approaches due to the complexity and non-smoothness of the NNs function which links different latent spaces. In this paper, we introduce a novel algorithm,

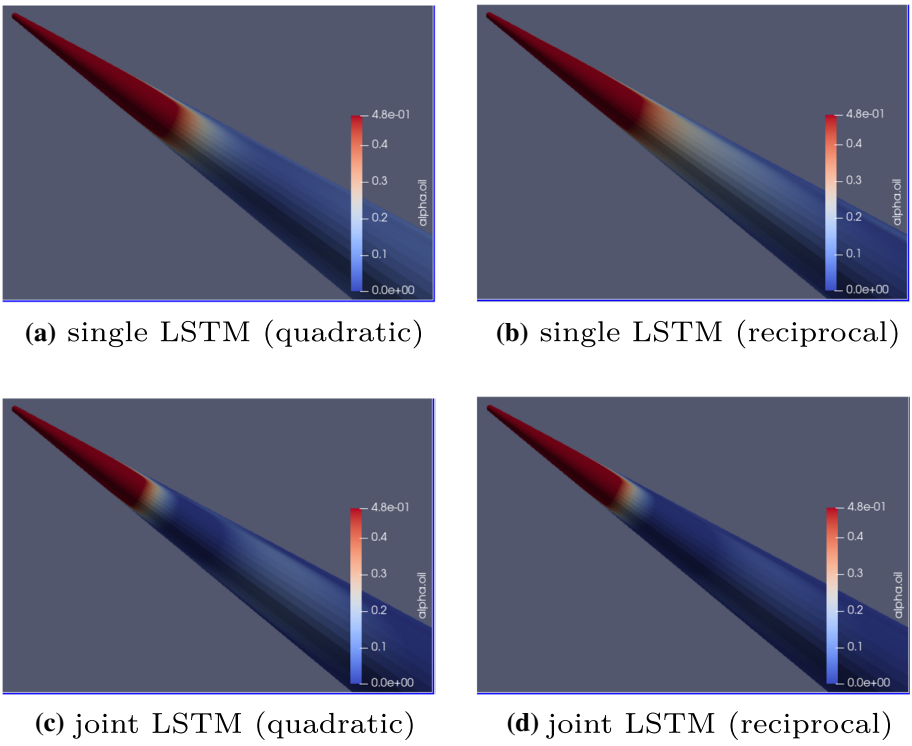


(a) Joint model latent space **(b)** Joint model full space

(c) Single model latent space **(d)** Single model full space

Fig. 20 Prediction relative error in the space of the principle component and the full space

named generalised Latent Assimilation, which makes use of a polynomial surrogate function to approximate the NNs transformation operator in a neighbourhood of the background state. Variational DA can then be performed by computing an observation loss using this local polynomial function. This new method promotes a much more flexible use of LA with machine learning surrogate models. A theoretical analysis is also given in the present study, where an upper bound of the approximation error of the DA cost function (evaluated on the true state) is specified. Future work can further focus on the minimization error related to the surrogate loss function in GLA. The numerical tests in the high-dimensional CFD application show that the proposed approach can ensure both the efficiency of the ROMs and the accuracy of the assimilation/prediction. In this study, the training and the validation for both ROM and LSTM are performed using a single CFD simulation with well separated training and testing datasets. Future work will investigate to build robust models for both autoencding and machine learning prediction using multiple CFD simulations as training data. However, building such training dataset can be time-consuming due to the complexity of the CFD code. The local polynomial surroagate function is computed relying on LHS samplings in this work. Other sampling strategies, such as Gaussian perturbations, can also be considered. Representing model or observation error (originally in the full space) in the latent space is challenging due to the non-linearity of ROMs. Future work can also be considered to enhance the error covariance specification in the latent space by investigating, for instance, uncertainty



(a) single LSTM (quadratic) **(b)** single LSTM (reciprocal)

(c) joint LSTM (quadratic) **(d)** joint LSTM (reciprocal)

Fig. 21 Prediction in the full CFD space after LA with quadratic (**a**, **c**) and reciprocal (**b**, **d**) observation functions

Table 8 Averaged computational time of GLA for one time step

Single model		Joint model	
Quadratic	Reciprocal	Quadratic	Reciprocal
57.62s	61.42s	51.33s	56.10s

propagation from the full physical space to the latent space, posterior error covariance tuning (e.g., [58, 65, 66]) or Ensemble-type [67] DA approaches.

Author Contributions SC and RA conceived the presented idea. SC developed the theory and performed the ML and DA computations. JC implemented the CFD computation and CA performed the physics experiments for initial conditions. OM, RA, Y-KG, PA and CP supervised the findings of this work. SC took the lead in writing the manuscript. All authors discussed the results and contributed to the final manuscript.

Funding This research is funded by the EP/T000414/1 PREdictive Modelling with QuantIfication of UncERtainty for MultiphasE Systems (PREMIERE). This work is partially supported by the Leverhulme Centre for Wildfires, Environment and Society through the Leverhulme Trust, Grant Number RC-2018-023. This work is partially supported by the CAS scholarship. This work is partially supported by RELIANT (EP/V036777/1), INHALE (EP/T003189/1), Wave-Suite (EP/V040235/1) and MUFFINS (EP/P033180/1).

Data Availability The code (python scripts) of the current study (including ROM, LSTM and GLA) is available at https://github.com/DL-WG/GLA_with_ROSM. The data generated in this study are available upon reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Nakamura, T., Fukami, K., Hasegawa, K., Nabae, Y., Fukagata, K.: Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. Phys. Fluids **33**(2), 025116 (2021)
2. Mohan, A.T., Gaitonde, D.V.: A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. arXiv preprint arXiv:1804.09269 (2018)
3. Casas, C.Q., Arcucci, R., Wu, P., Pain, C., Guo, Y.-K.: A reduced order deep data assimilation model. Physica D **412**, 132615 (2020)
4. Fu, R., Xiao, D., Navon, I., Wang, C.: A data driven reduced order model of fluid flow by auto-encoder and self-attention deep learning methods. arXiv preprint arXiv:2109.02126 (2021)
5. Carrassi, A., Bocquet, M., Bertino, L., Evensen, G.: Data assimilation in the geosciences: an overview of methods, issues, and perspectives. Wiley Interdiscip. Rev. Clim. Change **9**(5), 535 (2018)
6. Gong, H., Yu, Y., Li, Q., Quan, C.: An inverse-distance-based fitting term for 3D-Var data assimilation in nuclear core simulation. Ann. Nucl. Energy **141**, 107346 (2020). https://doi.org/10.1016/j.anucene.2020.107346
7. Amendola, M., Arcucci, R., Mottet, L., Casas, C.Q., Fan, S., Pain, C., Linden, P., Guo, Y.-K.: Data assimilation in the latent space of a neural network (2020)
8. Peyron, M., Fillion, A., Gürol, S., Marchais, V., Gratton, S., Boudier, P., Goret, G.: Latent space data assimilation by using deep learning. arXiv preprint arXiv:2104.00430 (2021)
9. Silva, V.L., Heaney, C.E., Li, Y., Pain, C.C.: Data assimilation predictive GAN (DA-PredGAN): applied to determine the spread of covid-19. arXiv preprint arXiv:2105.07729 (2021)
10. Fowler, A., Dance, S., Waller, J.: On the interaction of observation and prior error correlations in data assimilation. Q. J. R. Meteorol. Soc. **144**(710), 48–62 (2018)
11. Cheng, S., Argaud, J.-P., Iooss, B., Lucor, D., Ponçot, A.: Error covariance tuning in variational data assimilation: application to an operating hydrological model. Stoch. Environ. Res. Risk Assess. (2020) (**accepted for publication**)
12. Cheng, S., Lucor, D., Argaud, J.-P.: Observation data compression for variational assimilation of dynamical systems. J. Comput. Sci. **53**, 101405 (2021)
13. Nichols, N.K.: Mathematical concepts of data assimilation. In: Data Assimilation, pp. 13–39. Springer, Berlin (2010)
14. San, O., Maulik, R., Ahmed, M.: An artificial neural network framework for reduced order modeling of transient flows. Commun. Nonlinear Sci. Numer. Simul. **77**, 271–287 (2019)
15. Gong, H., Cheng, S., Chen, Z., Li, Q.: Data-enabled physics-informed machine learning for reduced-order modeling digital twin: application to nuclear reactor physics. Nucl. Sci. Eng. **196**, 1–26 (2022)
16. Arcucci, R., Mottet, L., Pain, C., Guo, Y.-K.: Optimal reduced space for variational data assimilation. J. Comput. Phys. **379**, 51–69 (2018)
17. Quilodrán-Casas, C., Arcucci, R., Mottet, L., Guo, Y., Pain, C.: Adversarial autoencoders and adversarial LSTM for improved forecasts of urban air pollution simulations. arXiv preprint arXiv:2104.06297 (2021)
18. Murata, T., Fukami, K., Fukagata, K.: Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. J. Fluid Mech. **882** (2020)

19. Phillips, T.R., Heaney, C.E., Smith, P.N., Pain, C.C.: An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. Int. J. Numer. Methods Eng. **122**(15), 3780–3811 (2021)
20. Heaney, C.E., Li, Y., Matar, O.K., Pain, C.C.: Applying convolutional neural networks to data on unstructured meshes with space-filling curves. arXiv preprint arXiv:2011.14820 (2020)
21. Zhou, Y., Wu, C., Li, Z., Cao, C., Ye, Y., Saragih, J., Li, H., Sheikh, Y.: Fully convolutional mesh autoencoder using efficient spatially varying kernels. arXiv preprint arXiv:2006.04325 (2020)
22. Xu, M., Song, S., Sun, X., Zhang, W.: Ucnn: A convolutional strategy on unstructured mesh. arXiv preprint arXiv:2101.05207 (2021)
23. Arcucci, R., Zhu, J., Hu, S., Guo, Y.-K.: Deep data assimilation: integrating deep learning with data assimilation. Appl. Sci. **11**(3), 1114 (2021)
24. Brajard, J., Carrassi, A., Bocquet, M., Bertino, L.: Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. J. Comput. Sci. **44**, 101171 (2020)
25. Liu, C., Fu, R., Xiao, D., Stefanescu, R., Sharma, P., Zhu, C., Sun, S., Wang, C.: Enkf data-driven reduced order assimilation system. Eng. Anal. Bound. Elem. **139**, 46–55 (2022). https://doi.org/10.1016/j.enganabound.2022.02.016
26. Becker, P., Pandya, H., Gebhardt, G., Zhao, C., Taylor, C.J., Neumann, G.: Recurrent kalman networks: factorized inference in high-dimensional deep feature spaces. In: International Conference on Machine Learning, pp. 544–552. PMLR (2019)
27. Su, L.: Prediction of multivariate chaotic time series with local polynomial fitting. Comput. Math. Appl. **59**(2), 737–744 (2010)
28. Su, L., Li, C.: Local prediction of chaotic time series based on polynomial coefficient autoregressive model. Math. Probl. Eng. **2015** (2015)
29. Choon, O.H., Hoong, L.C., Huey, T.S.: A functional approximation comparison between neural networks and polynomial regression. WSEAS Trans. Math **7**(6), 353–363 (2008)
30. Hwang, J., Lee, J., Lee, K.-S.: A deep learning-based method for grip strength prediction: comparison of multilayer perceptron and polynomial regression approaches. PLoS ONE **16**(2), 0246870 (2021)
31. Regonda, S., Rajagopalan, B., Lall, U., Clark, M., Moon, Y.-I.: Local polynomial method for ensemble forecast of time series. Nonlinear Process. Geophys. **12**(3), 397–406 (2005)
32. Molnar, C.: Interpretable Machine Learning. Lulu.com, New York (2020)
33. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)
34. Shi, S., Zhang, X., Fan, W.: A modified perturbed sampling method for local interpretable model-agnostic explanation. arXiv preprint arXiv:2002.07434 (2020)
35. Torre, E., Marelli, S., Embrechts, P., Sudret, B.: Data-driven polynomial chaos expansion for machine learning regression. J. Comput. Phys. **388**, 601–623 (2019)
36. Shahzadi, G., Soulaïmani, A.: Deep neural network and polynomial chaos expansion-based surrogate models for sensitivity and uncertainty propagation: an application to a rockfill dam. Water **13**(13), 1830 (2021)
37. Emschwiller, M., Gamarnik, D., Kızıldağ, E.C., Zadik, I.: Neural networks and polynomial regression. demystifying the overparametrization phenomena. arXiv preprint arXiv:2003.10523 (2020)
38. Ostertagová, E.: Modelling using polynomial regression. Procedia Eng. **48**, 500–506 (2012)
39. Lumley, J.L.: The structure of inhomogeneous turbulent flows. In: Atmospheric Turbulence and Radio Wave Propagation (1967)
40. Stewart, G.W.: On the early history of the singular value decomposition. SIAM Rev. **35**(4), 551–566 (1993)
41. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput. **29**(9), 2352–2449 (2017)
42. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech, vol. 2, pp. 1045–1048. Makuhari (2010)
43. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **6**(02), 107–116 (1998)
44. Lawless, A., Gratton, S., Nichols, N.: Approximate iterative methods for variational data assimilation. Int. J. Numer. Methods Fluids **47**(10–11), 1129–1135 (2005)
45. Fulton, W.: Eigenvalues, invariant factors, highest weights, and Schubert calculus. Bull. Am. Math. Soc. **37**, 209–250 (2000)
46. Wang, G., Giannakis, G.B., Chen, J.: Learning relu networks on linearly separable data: algorithm, optimality, and generalization. IEEE Trans. Signal Process. **67**(9), 2357–2370 (2019)

47. Tang, B.: Orthogonal array-based Latin hypercubes. J. Am. Stat. Assoc. **88**(424), 1392–1397 (1993)
48. Talagrand, O.: A posteriori evaluation and verification of analysis and assimilation algorithms. In: Workshop on Diagnosis of Data Assimilation Systems, 2–4 November 1998, pp. 17–28. ECMWF, Shinfield Park, Reading. ECMWF (1999)
49. Holladay, J.: A note on the Stone–Weierstrass theorem for quaternions. In: Proceedings of the American Mathematical Society, vol. 8, pp. 656–657 (1957)
50. Voulgaropoulos, V., Angeli, P.: Optical measurements in evolving dispersed pipe flows. Exp. Fluids **58**(12), 170 (2017)
51. Kumar, S., Ramkrishna, D.: On the solution of population balance equations by discretization-II. A moving pivot technique. Chem. Eng. Sci. **51**(8), 1333–1342 (1996)
52. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: convolution on x-transformed points. Adv. Neural Inf. Process. Syst. **31**, 820–830 (2018)
53. Reid, J.K., Scott, J.A.: Reducing the total bandwidth of a sparse unsymmetric matrix. SIAM J. Matrix Anal. Appl. **28**(3), 805–821 (2006)
54. Oliker, L., Li, X., Heber, G., Biswas, R.: Ordering unstructured meshes for sparse matrix computations on leading parallel systems. In: International Parallel and Distributed Processing Symposium, pp. 497–503. Springer (2000)
55. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proceedings of the 1969 24th National Conference, pp. 157–172 (1969)
56. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
57. Carta, A., Sperduti, A., Bacciu, D.: Incremental training of a recurrent neural network exploiting a multi-scale dynamic memory. arXiv preprint arXiv:2006.16800 (2020)
58. Cheng, S., Argaud, J.-P., Iooss, B., Lucor, D., Ponçot, A.: Background error covariance iterative updating with invariant observation measures for data assimilation. Stoch. Environ. Res. Risk Assess. **33**(11), 2033–2051 (2019)
59. Farchi, A., Laloyaux, P., Bonavita, M., Bocquet, M.: Using machine learning to correct model error in data assimilation and forecast applications. Q. J. R. Meteorol. Soc. **147**(739), 3067–3084 (2021)
60. Cheng, S., Prentice, I.C., Huang, Y., Jin, Y., Guo, Y.-K., Arcucci, R.: Data-driven surrogate model with latent data assimilation: application to wildfire forecasting. J. Comput. Phys. **464**, 111302 (2022)
61. Zhuang, Y., Cheng, S., Kovalchuk, N., Simmons, M., Matar, O., Guo, Y., Arcucci, R.: Ensemble latent assimilation with deep learning surrogate model: application to drop interaction in microfluidics device. Lab on a Chip **22**, 3187–3202 (2022)
62. Arcucci, R., D'Amore, L., Pistoia, J., Toumi, R., Murli, A.: On the variational data assimilation problem solving and sensitivity analysis. J. Comput. Phys. **335**, 311–326 (2017)
63. Cacuci, D.G., Ionescu-Bujor, M.: Sensitivity and uncertainty analysis, data assimilation, and predictive best-estimate model calibration. In: Handbook of Nuclear Engineering (2010)
64. Argaud, J.-P.: User documentation, in the SALOME 9.3 platform, of the ADAO module for "Data Assimilation and Optimization". Technical report 6125-1106-2019-01935-EN, EDF / R&D (2019)
65. Desroziers, G., Berre, L., Chapnik, B., Poli, P.: Diagnosis of observation, background and analysis-error statistics in observation space. Q. J. R. Meteorol. Soc. **131**(613), 3385–3396 (2005)
66. Cheng, S., Qiu, M.: Observation error covariance specification in dynamical systems for data assimilation using recurrent neural networks. Neural Comput. Appl. **34**, 1–19 (2021)
67. Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. J. Geophys. Res. Oceans **99**(C5), 10143–10162 (1994)

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