Parametric Non-Intrusive Reduced Order Modelling via Taylor Series Neural Networks and Dynamic Identification

Xinlei Lin[a], Andrew.G.Buchan[b], Jiaying Zhang[h], Sibo Cheng[e], Erwu Liu[c], Haibao Wen[d], Fan Xiao[f], Zepeng Yang[g], Dunhui Xiao[a,*]

[a]*School of Mathematical Sciences, Key Laboratory of Intelligent Computing and Applications(Ministry of Education), Tongji University, Shanghai 200092, CHINA*
[b]*School of Engineering and Materials Science, Queen Mary University of London, UK, E1 4NS*
[c]*College of Electronic and Information Engineering, Tongji University, Shanghai, China*
[d]*University of Bath, Bath, United Kingdom, BA2 7AY.*
[e]*CEREA, ENPC, EDF R&D, Institut Polytechnique de Paris, Île-de-France, France*
[f]*School of Earth Sciences and Engineering, Sun Yat-sen University, Zhuhai 519000, China*
[g]*School of Chemical Science and Engineering, Tongji University, Shanghai, China, 200092*
[h]*School of Aeronautic Science and Engineering, Beihang University, 37 Xueyuan Rd, Beijing, 100191, CHINA*

Abstract

Numerically solving parameterised partial differential equations (P-PDEs) is both valuable and computationally intensive, driving the development of reduced-order models (ROMs). Recently, methods that combine latent space identification techniques with deep learning algorithms (e.g., autoencoders) have shown great potential in describing the dynamical system in the lower-dimensional latent space. In this paper, a new parametric latent identification of nonlinear dynamics neural networks, P-TLDINets, is introduced, which relies on a novel neural network structure based on Taylor series expansion and ResNets to learn the parametric differential equations that govern the dynamics in the latent space. During the training process, Taylor series-based Latent Dynamic Neural Network (TLDNet) and identified equations are trained simultaneously to generate a smoother latent space. In order to facilitate the parameterised study, a k-nearest neighbours (KNN) method based on an inverse distance weighting (IDW) interpolation scheme is introduced to predict the identified ODE coefficients using local information. Compared to other latent dynamics identification methods based on autoencoders, P-TLDINet eliminates the encoder-decoder's dependence on spatial coordinates, enabling it to be a parameterised ROM. Additionally, its lightweight network structure allows high generalization capabilities with less training time. P-TLDINet was tested on the 2D Burgers' equation and the lock-exchange case, maintaining an L_2 error of less than 2% compared to high-fidelity models.

Keywords: reduced-order model, latent-space identification, sparse regression, Taylor series expansion, deep learning, equation discovery, ResNet

1. Introduction

In the fields of engineering and physics, the behaviours of many complex systems are governed by partial differential equations (PDEs)[1, 2, 3, 4, 5]. To describe and understand these systems through PDEs, physical simulations, and computational fluid dynamics (CFD) provide essential guidance for advanced design and practical industrial applications[6]. During the past few decades, numerical methods such as the finite element method (FEM)[7] and the finite volume method (FVM)[8] have been continuously advancing, while significant progress has been made in computer hardware, particularly GPUs. However, as the demand for superior accuracy in discrete solutions increases and grids become highly refined, the computational requirements of these traditional methods become exceedingly substantial. This is especially evident for problems involving repeated simulations, such as parametric problems[2, 9], where the expansion of modelling

*Corresponding author
Email address: xiaodunhui@tongji.edu.cn (Dunhui Xiao)

iterations leads to significant resource consumption. *Consequently, these challenges may ultimately limit the practical applicability of such methods.*

To overcome these limitations, reduced-order modelling (ROM) has emerged as a promising approach to alleviate the computational burden across many areas, such as turbulent flows[10, 11], aerodynamics[12], molecular dynamics[13], biomedical sciences[14], meteorology[15], fluids[16, 17], structural dynamics problems[18], aerospace[19], and plasma dynamics[20]. The primary objective of ROM is to reduce the dimensions of full-order models by projecting the system onto a lower-dimensional latent linear or non-linear subspace. Typically, ROM achieves remarkable speed enhancements compared to high-fidelity models, albeit with a negligible decrease in accuracy, rendering it incredibly attractive.

Popular linear reduced-order methods, which are typically based on projection techniques, include proper orthogonal decomposition (POD)[21], dynamic mode decomposition (DMD)[22], and reduced basis methods (RB)[23]. Nonlinear projection-based methods, such as autoencoders, also demonstrate impressive capabilities in dimensionality reduction[11, 24]. They have already been widely applied in various fields such as aerodynamic wing design[25], fluid dynamics[26, 27, 28], magnetohydrodynamics (MHD)[29], PDE solving [30], and long-span bridge building[31]. The non-intrusive reduced order model (NIROM) is a recently proposed popular method to avoid the instability and non-linear inefficiency of ROM, which uses interpolation and artificial intelligence methods to construct functions in the reduced space to represent the dynamics. One of the advantages of non-intrusive methods is that they only require input-output data from the high-fidelity solver since they do not need access to its internal structure or source code. A number of NIROM have been presented, such as a NIROM based on POD combined with radial basis interpolation [32], POD combined with the Long Short-Term Memory (LSTM) deep learning method[3], POD combined with the neural network[33], and soft clustering combined with the classification method[34]. It also includes a recently presented nonlinear model reduction via Probabilistic Manifold Decomposition (PMD) [35]. However, the dynamics in the reduced space in the NIROMs remain a black box; these data-driven models are inherently difficult to interpret and extrapolate. Unlike the traditional NIROM, this work uses explicit differential equations to represent reduced dynamics, enabling it to be more easily understood and explainable. In addition, it is difficult for the traditional NIROM to deal with problems with different meshes. This is achieved via equation discovery methods in latent spaces.

The discovery of governing equations in latent spaces is a particularly promising approach, as the dynamics in the latent space can be identified as a set of differential equations. These identified governing equations can be easily solved and mapped back to the high-dimensional flow field solution space[36]. In other words, these equations provide a better representation of the underlying physical or mathematical laws of model learning, leading to improved model interpretability and enhanced generalisation. Symbolic identification and genetic algorithms are two milestone methods for the automatic inference of equations[37, 38]. The sparse identification of nonlinear dynamics (SINDy)[39], through the use of a function library and sparse linear regression, has seen widespread applications and led to the development of several related parametric PDE algorithms[40, 41, 42, 43], such as PDE-FIND[44],LaSDI[42], Greedy Identification of Latent Dynamics (GILD)[45], gLaSDI[41] and GPLaSDI[43]. By extracting coefficients from latent governing equations and combining them with interpolation or fitting methods, such as radial basis function (RBF) interpolation[32], Gaussian process regression (GPR)[43], or linear interpolation techniques, one can achieve efficient and effective evaluation and prediction across the entire parameter space. Recently, we presented a method (REFIND) to discover lower dimensional equations in the latent space[46]. However, this method works only for problems with fixed parameters and cannot handle cases where parameters vary, such as different boundary conditions, initial conditions, or equation coefficients.

This paper extends our previous work[46] to address parametric problems. The method is called the Parametric Taylor series-based Latent Dynamics Identification Neural Network (P-TLDINet). The P-TLDINet is constructed via Taylor series expansions, ResNets, and the k -nearest neighbours (KNN) algorithm combined with an inverse distance weighting (IDW) interpolation scheme. In the context of time discretization for neural ODE-based models, various strategies have been explored to improve accuracy and robustness. Taylor-series expansion has been shown to enable larger time steps while maintaining accuracy, offering a computationally efficient alternative to standard ODE solvers[47]. Meanwhile, perturbation and denoising techniques have been introduced to enhance robustness against error accumulation during long-term predictions[48]. In this work, we adopt the second-order Taylor-series expansion for time advancement. Apart from identifying latent equations, P-TLDINet can be applied to more complex parametric PDE problems; for example, problems in

which the initial conditions vary with the parameters.

In addition, the P-TLDINet model uses only a small amount of high-fidelity training data to achieve accurate, robust, and efficient data-driven reduced-order modelling for physical systems. Besides, the P-TLDINet is a parametric ROM that is capable of running simulations with variable meshes. This allows P-TLDINet to predict high-dimensional approximation data at any query point $\boldsymbol{x}$ in the computational domain Ω, even if some of them may not be part of the training grids. This is another novelty of the newly presented P-TLDINet.

The subsequent part of the paper is organised as follows: Section 2 describes the related governing equations for parametric systems. The framework of P-TLDINet includes several parts, such as a family of networks called the Taylor series-based Latent Dynamics Network (TLDNet) in Section 3 and Identification of Dynamics (ID models) in latent space in Section 4. Section 5 shows the KNN method and the interpolation process. Section 6 is data preprocessing, including data sampling and normalisation of P-TLDINet. The whole framework of P-TLDINet is summarised in Section 7. The capabilities of P-TLDINet are demonstrated using two numerical examples: the 2D-Burgers equations in Section 8.1 and lock exchange in Section8.2. Furthermore, we evaluate the performance of P-TLDINet on multiple mesh resolutions during training, demonstrating its discretisation independence. Through these examples, P-TLDINet demonstrates its distinct differences and superiority over autoencoder-based latent dynamics identification methods. Finally, we summarise the main innovations and application prospects of the proposed model, providing a comprehensive overview of this research in Section 9.

2. Governing equations of parameterised dynamic systems

In practical simulations, the PDEs that govern the systems are typically characterised by the parameter vector $\boldsymbol{\mu}$, which contains initial or boundary conditions as well as the dynamics of the system. Suppose $\boldsymbol{\mu}$ is the input variable in a parameter domain $\mathcal{D} \subset \mathbb{R}^{N_{\mathcal{D}}}$, in which $N_{\mathcal{D}}$ is the number of parameters in $\boldsymbol{\mu}$. The governing PDEs of a parameterised dynamical system are given as

$$\begin{cases} \boldsymbol{F}(\boldsymbol{u}(t,\boldsymbol{x};\boldsymbol{\mu}),t,\boldsymbol{x};\boldsymbol{\mu}) = s(t,\boldsymbol{x},\boldsymbol{\mu}) & (t,\boldsymbol{x}) \in [0,T]\times\Omega \\ \boldsymbol{u}(t=0,\boldsymbol{x};\boldsymbol{\mu}) = \boldsymbol{u}_0(\boldsymbol{\mu}), \end{cases} \tag{1}$$

where $\boldsymbol{u} : [0,T]\times\Omega \mapsto \mathcal{U} \subset \mathbb{R}^{N_f}$ stands for either a scalar field or vector fields, such as velocity, pressure, or temperature. The term $T \in \mathbb{R}^+$ denotes the simulation time, and $\Omega \in \mathbb{R}^d$ defines the spatial domain of the problem. The operator $\boldsymbol{F} : \mathbb{R}^{N_f}\times[0,T]\times\Omega\times\mathcal{D} \mapsto \mathbb{R}^{N_f}$ denotes the dynamic evolution of $\boldsymbol{u}$ and its conservation laws, while s is the source term. Finally, the term $\boldsymbol{u}_0$ denotes the initial value of $\boldsymbol{u}$.

For the training of the following networks, Equation 1 is solved using some computational model, and solutions are sampled at $N_u \times N_t$ points within the spatial-temporal domain. These sampling points in space are denoted by $\boldsymbol{x} \in \Omega$, which are typically vectors of length determined by the problem's dimension d. For example, in a problem based on a two-dimensional space, $\boldsymbol{x} = (x_1, x_2)$. The computer model is resolved a total of N_μ times, with varying parameters $\boldsymbol{\mu}$. The discrete set of sampled points, corresponding to a given parameter vector $\boldsymbol{\mu}_i \in \mathcal{D}$, is defined as $\boldsymbol{u}_i = [u_{i,1}(t_l), u_{i,2}(t_l), \ldots, u_{i,N_u}(t_l)] \in \mathbb{R}^{N_u}$, for $l \in \{1, 2 \ldots N_t\}$. A summary of the main notation used for data set sizes are listed in Table 1.

In this work, an efficient model reduction method is developed for fast and accurate solutions to the PDE defined in Equation 1. This model, P-TLDINet, includes two components: A Taylor series-based Latent Dynamic Network (TLDNet) and Identification of Dynamics models (ID models). These are introduced in Section 3 and Section 4 respectively.

3. Taylor series-based Latent Dynamic Neural Network (TLDNet)

The Taylor series-based Latent Dynamic Network (TLDNet) consists of three network structures: the dynamic network NN_{dyn}, the reconstruction network NN_{rec}, and the initial mapping network NN_{z_0}, which correspond to the model parameters W_{dyn}, W_{rec} and W_{z_0} respectively. The first two networks are used to complete the model reduction, and the last is used to process the initial values so that P-TLDINet can adapt to more situations. They are constructed using fully connected neural networks (FCNNs) and residual neural networks. ResNets and Taylor series expansion theory are integrated into TLDNets to enhance the representation ability of P-TLDINets, enabling their application to more complex fluid cases.

Notation	Description
$\boldsymbol{u}$	The state variables of dynamics.
$\mathcal{S}$	The latent state space manifold.
Ω	The whole high-dimensional computational domain. And the coordinates $\boldsymbol{x} \in \Omega$.
N_u	The number of spatial sampling training points on Ω.
N_s	The dimension of state variables, i.e. the dimension of latent space.
$N_{\mathcal{D}}$	The dimension of the entire parameter space $\mathcal{D}$.
$N_{\boldsymbol{\mu}}$	The number of sampling parameter points in the training parameter space $\mathcal{D}_{train}$.
N_h	The number of sampling parameter points in the testing parameter space $\mathcal{D}^h$.
N_t	The number of discrete-time points in the time domain.

Table 1: Notations used for variables and set sizes

3.1. TLDNet model structure

Three networks are first constructed to form the TLDNet framework. These are defined as,

$$\begin{cases} [\frac{d}{dt}\boldsymbol{z}(t), \frac{d^2}{dt^2}\boldsymbol{z}(t)] = NN_{dyn}(\boldsymbol{z}(t), \boldsymbol{\mu}) & t \in (0, T] \\ \boldsymbol{z}(0) = NN_{z_0}(\boldsymbol{\mu}) \\ \tilde{\boldsymbol{u}}(\boldsymbol{x}, t) = NN_{rec}(\boldsymbol{z}(t), \boldsymbol{\mu}, \boldsymbol{x}), & t \in [0, T], \end{cases} \tag{2}$$

which consists of a dynamic model NN_{dyn}, an initial mapping model NN_{z_0}, and a reconstruction model NN_{rec}. The models all involve a set of time dependent variables $\boldsymbol{z}(t)$ which lies in some hypothetical latent space $\mathcal{S} \subset \mathbb{R}^{N_s}$. These are vectors of size N_s which is defined by the user through the setting of the input size of the dynamic network NN_{dyn}. The intention is that the size is relatively small in comparison with the sampled space, i.e. $N_s \ll N_u$, in order that training and executing of all the networks are efficient. The latent space itself is formed as part of the training of the networks for which all three networks in Equation 2 are trained simultaneously.

The structure of the dynamic model NN_{dyn} takes, as its inputs, the latent vector $\boldsymbol{z}(t)$ (at some time t) and the vector of parameters $\boldsymbol{\mu}$. When combined, the input will be a vector of size $N_s + N_D$. Its output is a vector of size $2N_s$, which comprises the first-order $\frac{d}{dt}\boldsymbol{z}(t)$ and second-order $\frac{d^2}{dt^2}\boldsymbol{z}(t)$ time derivatives of the latent variables $\boldsymbol{z(t)}$. Note that these outputs are not known from any training data, but are formed and learnt as part of the training process. However, once formed, they are used in combination with a Taylor series to form the updates of $\boldsymbol{z(t)}$ at the next time step. That is, from time instance t_m, the values of $\boldsymbol{z}$ at time t_{m+1} are calculated as:

$$\boldsymbol{z}(t_{m+1}) = \boldsymbol{z}(t_m) + \frac{d}{dt}\boldsymbol{z}(t_m)\Delta t + \frac{d^2}{dt^2}\boldsymbol{z}(t_m)\frac{\Delta t^2}{2!} + o((\Delta t)^3), \quad m = \{1, 2, \ldots, N_t\}, \tag{3}$$

where the higher order terms beyond the 3rd time derivative are ignored. The dynamic network NN_{dyn} is combined with the reconstruction network NN_{rec} that takes as input the vector of latent variable $\boldsymbol{z(t)}$, the vector of parameters $\boldsymbol{\mu}$, and a spatial position $\boldsymbol{x}$. The total size of the input is $N_s + N_D + d$. The output of the reconstruction network is the estimated value of $\boldsymbol{u}(\boldsymbol{x}, t)$ (which may be a vector or scalar depending on the nature of the governing equations) which is the state variable evaluated at position $\boldsymbol{x}$ at time t.

Thus far, the dynamic network and the reconstruction network are coupled via the following route. The latent and parameter variables are fed through the dynamic network that predicts the 1st and 2nd time derivatives of the latent variables. These are used to update the latent variables at the next time step via the Taylor series. The reconstruction network is used to predict the values of $\boldsymbol{u}(\boldsymbol{x}, t)$, from $\boldsymbol{z}(t), \boldsymbol{x}$ and $\boldsymbol{\mu}$, at some position $(\boldsymbol{x}, t)$ in space-time. When training the networks, given the training data points, a loss function can be formulated as,

$$Loss_{rec}(W_{rec}, W_{dyn}) = \frac{1}{N_\mu}\sum_{i=1}^{N_\mu}\left[\frac{1}{N_t+1}\sum_{l=0}^{N_t}\left(\frac{1}{N_u}\sum_{m=1}^{N_u}\|NN_{rec}(\boldsymbol{z}(t_l), \boldsymbol{\mu}_i, \boldsymbol{x}^m) - \boldsymbol{u}_{i,m}(t_l)\|_2^2\right)\right], \tag{4}$$

where $\|\cdot\|_2$ denotes the L_2 norm. This calculates the discrepancy between the predicted and estimated values of the state variable over all space-time data points, and for all training parameter sets $\boldsymbol{\mu_i}$.
The final component is to define the initialisation network N_{z_0} that maps the parameter variables $\boldsymbol{\mu}$ to the initial latent space variables at time t_0, i.e. $\boldsymbol{z}(t_0)$. Thus its inputs and outputs are vectors of size N_D and N_s, respectively. The loss function of NN_{z_0} is based on mean square error (MSE) and is summarised as

$$\begin{aligned} Loss_{z_0}(W_{z_0}) &= \sum_i MSE(NN_{rec}(\boldsymbol{z}_0, \boldsymbol{\mu}_i, \boldsymbol{x}), \boldsymbol{u}_i(t_0)) \\ &= \sum_i MSE(NN_{rec}(NN_{z_0}(\boldsymbol{\mu_i}), \boldsymbol{\mu}_i, \boldsymbol{x}), \boldsymbol{u}_i(t_0)) \end{aligned} \tag{5}$$

which defines the discrepancy between the projected initial condition and that of the training data. The fact that the values $\boldsymbol{z}(0)$ are directly projected from $\boldsymbol{\mu}$ enables TLDNet to address complex parametrised problems, especially those with different initial conditions. This helps expand its applicability without the need to infer from modify the original high-dimensional system. This offers a computational advantage over many typical ROM algorithms, particularly when combined with SINDy of the next section, where $\boldsymbol{z}(0)$ is formed through a more computationally expensive mapping of the high-dimensional initial values to the lower-dimensional space. As previously mentioned, the three networks are trained simultaneously by minimising the loss functions of equations 4 and 5,

$$Loss_{z_0}(W_{z_0}) + Loss_{rec}(W_{rec}, W_{dyn}) \tag{6}$$

which trains all network variables while forming the latent space and, in turn, producing the dynamic network that forms both first and second order derivatives with respect to time.

3.2. TLDNet Network Structures

The networks of equations 2 use feed-forward structured networks that consist of a combination of fully connected neural networks (FCNNs) and residual neural networks (ResNet). The FCNNs are used as they have shown a considerable degree of superiority in function approximation across various fields [49]. However, as the depth of networks increases with problem complexity, deep learning models are prone to suffer from gradient vanishing and overfitting, resulting in weakened performance. Deep residual learning neural networks (ResNet) are architectures proposed to address these challenges to a certain extent, improving the ability of networks to capture more dynamics in data. They were initially based on convolutional neural network (CNN) technology and were used in the field of computer vision and image classification [50]. Here, the residual learning concept is applied to FCNNs. The main difference between residual learning and classical FCNNs lies in the structure of their networks. An FCNN consists of multiple hidden layers, where adjacent neurones are directly connected by weight matrices and biases, and nonlinearity is provided by activation functions. In contrast, for a ResNet, there exists a shortcut path connecting the input and output layers directly, known as a "skip connection." This can be viewed as the mapping $\mathcal{H}$ from the input X when used in combination with the set of fully connected layer mappings $\mathcal{F}$.

$$\mathcal{H}(X) := \mathcal{F}(X) + X. \tag{7}$$

In this article, the fully connected layers and residual layers are used in combination with the Tanh and ReLU activation functions, respectively.

4. Identification of dynamics (ID model) in the latent space

Following the construction of the TLDNet framework, an identification of dynamics (ID) model is formulated for $\boldsymbol{z}(t) \in \mathcal{S}$ to discover the intrinsic characteristics of the latent dynamics. By its construction in section 3.1, the vector $\boldsymbol{z}(t)$ is equivalent to a compressed representation of the high-dimensional physical data $\boldsymbol{u}(\boldsymbol{x}, t)$, defined over both time and space. The temporally dependent $\boldsymbol{z}(t)$ reflects the physical features of the original solution manifold, so that it can be viewed as a dynamical system governed by a set of ODEs. Therefore, consider the reduced-order dynamic identification model as follows:

$$\frac{d}{dt}\boldsymbol{z}(t;\boldsymbol{\mu}) = f(\boldsymbol{z}(t;\boldsymbol{\mu})), \quad t \in [0, T] \tag{8}$$

where $\boldsymbol{z}(t;\boldsymbol{\mu}) \in \mathbb{R}^{N_s}$ is defined by Equation 3. For a single $\boldsymbol{\mu}_i \in \mathcal{D}_{train}$, the associated discrete latent variable matrix $\boldsymbol{z}_i$ and the time derivative matrix $\dot{\boldsymbol{z}}_i$ can be expressed as,

$$\begin{aligned}\boldsymbol{z}_i(t;\boldsymbol{\mu}_i) &= [z_{i,1}(t;\boldsymbol{\mu}_i), z_{i,2}(t;\boldsymbol{\mu}_i), \ldots, z_{i,N_s}(t;\boldsymbol{\mu}_i)] \in \mathbb{R}^{N_t+1} \times \mathbb{R}^{N_s},\\ \dot{\boldsymbol{z}}_i(t;\boldsymbol{\mu}_i) &= [\dot{z}_{i,1}(t;\boldsymbol{\mu}_i), \dot{z}_{i,2}(t;\boldsymbol{\mu}_i), \ldots, \dot{z}_{i,N_s}(t;\boldsymbol{\mu}_i)] \in \mathbb{R}^{N_t+1} \times \mathbb{R}^{N_s}.\end{aligned} \tag{9}$$

The m^{th} row of these matrices respectively contain the vector $\boldsymbol{z}$ and its time derivative at the m^{th} time step. In this article, the values for both matrices are obtained from the TLDNet's outputs. More specifically, $\boldsymbol{z}_i(t_m)$ can be found following the evaluation of the Taylor's series, Equation 3, at the m^{th} time step, and $\dot{\boldsymbol{z}}_i(t_m)$ is obtained from the output of the network NN_{dyn}, Equation 2, again at the m^{th} time step. This is a distinction from other methods forming $\dot{\boldsymbol{z}}_i$ which has typically been computed via finite difference methods, or from physical priors, during the training procedure in previous studies [51, 39, 43, 41, 46].

Following the path of the SINDy method [39], the form of the governing dynamic function f is determined to be an expansion of non-linear functions, or a library of candidate basis functions, that is selected by the user. Typically the set of basis functions, denoted by $\Theta(z)$, is of a simple form, for example of polynomial, trigonometric, or exponential type. These operate on the elements of the vector of latent variables $\boldsymbol{z}(t)$. By way of example, if the vector was composed of 3 elements, $\boldsymbol{z}(t) = \{z_1(t), z_2(t), z_3(t)\}$, the basis set up to polynomial with quadratic order would take the form:

$$\Theta(z_i) = \begin{pmatrix} 1 & z_1(t_0) & z_2(t_0) & z_3(t_0) & z_1^2(t_0) & z_2^2(t_0) & z_3^2(t_0) & z_1(t_0)z_2(t_0) & z_1(t_0)z_3(t_0) & z_2(t_0)z_3(t_0) \\ 1 & z_1(t_1) & z_2(t_1) & z_3(t_1) & z_1^2(t_1) & z_2^2(t_1) & z_3^2(t_1) & z_1(t_1)z_2(t_1) & z_1(t_1)z_3(t_1) & z_2(t_1)z_3(t_1) \\ \vdots & & & & \vdots & & & & & \vdots \\ 1 & z_1(t_{N_t}) & z_2(t_{N_t}) & z_3(t_{N_t}) & z_1^2(t_{N_t}) & z_2^2(t_{N_t}) & z_3^2(t_{N_t}) & z_1(t_{N_t})z_2(t_{N_t}) & z_1(t_{N_t})z_3(t_{N_t}) & z_2(t_{N_t})z_3(t_{N_t}) \end{pmatrix}. \tag{10}$$

In general, the set $\Theta(z)$ is a matrix of functions of size, $\mathbb{R}^{N_t+1} \times \mathbb{R}^{N_b}$, where $\mathbb{R}^{N_b}$ denotes the total number of basis functions used. The expansion of the basis functions is then formed via the multiplication with a coefficient set, which can be written in matrix form as,

$$\dot{\boldsymbol{z}}_i \approx \boldsymbol{\Theta}(\boldsymbol{z}_i) \cdot \boldsymbol{\Xi} \tag{11}$$

where $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_s}] \in \mathbb{R}^{N_b} \times \mathbb{R}^{N_s}$ is the coefficient matrix.

In this paper, polynomials up to the first order terms are used to construct $\boldsymbol{\Theta}(\boldsymbol{z}(\boldsymbol{t}))$, this would related to the first 4 columns of Equation 10. For each $\boldsymbol{\mu}_i$, there is a specific system of ODEs to identify the latent dynamics. In fact, during the physical training and discovery of ID models, evaluating the error between $\boldsymbol{z}_i$ and the estimated value $\tilde{\boldsymbol{z}}_i$ is inefficient. This is because the right side of Equation 11 approximates the time derivative of $\boldsymbol{z}_i$ (i.e. $\dot{\boldsymbol{z}}_i$), rather than $\boldsymbol{z}_i$ itself, which must be obtained by solving ODEs. Consequently, the loss function in the ID model must be adjusted to account for the computational cost, and it can be summarised as

$$Loss_{\mu_i} = ||\frac{d\boldsymbol{z_i}}{dt} - \boldsymbol{\Theta}(\boldsymbol{z}_i) \cdot \boldsymbol{\Xi}_i||_2^2 \tag{12}$$

for each ID model with parameter $\boldsymbol{\mu}_i$. This idea was first utilised by Brunton et al. in [40], in which the errors in the time derivatives of both the original space (i.e. $\frac{d\boldsymbol{u}}{dt}$) and the latent space (i.e. $\frac{d\boldsymbol{z}}{dt}$) are factored into the loss function. However, considering that incorporating terms related to $\frac{d\boldsymbol{u}}{dt}$ would significantly increase computational costs and the risk of overfitting, only the loss in $\frac{d\boldsymbol{z}}{dt}$ is included in the loss function here. Therefore, the low-dimensional data related to all training parameters in $\mathcal{D}_{train}$, a joint training approach is employed in TLDNets by the loss function, simultaneously minimising the sum of MSE as follows:

$$Loss_{dz/dt}(\boldsymbol{\Xi}) = \frac{1}{N_{\boldsymbol{\mu}}} \sum_{i=1}^{N_{\boldsymbol{\mu}}} \Big(\frac{1}{N_s} \sum_{j=1}^{N_s} ||\frac{d\boldsymbol{z}_i^{(j)}}{dt} - \boldsymbol{\Theta}(\boldsymbol{z}_i^{(j)}) \cdot {\xi_i^{(j)}}^T ||_2^2 \Big), \tag{13}$$

where $N_{\boldsymbol{\mu}} = |\mathcal{D}_{train}|$. Furthermore, given the sparsity of the SINDy coefficients, a penalty (or regularisation) term for $\boldsymbol{\Xi}$ is incorporated into the loss function of the ID model,

$$Loss_{ID}(\boldsymbol{\Xi}) = \omega_{ID} Loss_{dz/dt}(\boldsymbol{\Xi}) + \omega_{coef} \frac{1}{N_{\boldsymbol{\mu}}} \sum_{i=1}^{N_{\boldsymbol{\mu}}} ||\boldsymbol{\Xi}_i||_2^2. \tag{14}$$

In addition, both the loss term, $Loss_{dz/dt}$, and the penalty term have loss weights ω_{ID} and ω_{coef} applied.

5. k-Nearest Neighbour interpolation with inverse distance weighting

Based on latent ID models discovered from the training parameter set $\mathcal{D}_{train}$ and the corresponding high-dimensional data $\{\boldsymbol{u}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$, low-dimensional dynamics can be explored at any unseen or trained parameter point $\boldsymbol{\mu}$ throughout the parameter space $\mathcal{D}$. To improve the efficiency of parametric study, the k-nearest neighbours (KNN) convex interpolation method is used to fit the learned ID model coefficient set $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$. As an interpolation method with broad application prospects [52], the KNN approach captures the ID model coefficient values of each $\boldsymbol{\mu}$ from local features rather than globally, and offers greater accuracy compared to linear interpolation. In contrast to radial basis functions (RBF) and Gaussian process regression (GPR) methods, which ask optimisation to acquire interpolation weights, KNN-based interpolation results in lower computational costs and higher efficiency.

In this paper, we combine the inverse distance weighting (IDW) function with the KNN method as an interpolation technique. This integration ensures the convexity of the fitting interpolation problem and enhances robustness. For any testing parameter vector $\boldsymbol{\mu}^*\in\mathcal{D}$, denote the set of k nearest sample parameter points to $\boldsymbol{\mu}^*$ in $\mathcal{D}_{train}$ as $\mathcal{D}(\boldsymbol{\mu}^*)$. The mathematical description is as follows:

$$\boldsymbol{\Xi}^* = \mathcal{I}(\boldsymbol{\mu}^*;\{\boldsymbol{\Xi}_i\}_{i\in\mathcal{D}(\boldsymbol{\mu}^*)}) = \sum_{\boldsymbol{\mu}_i\in\mathcal{D}(\boldsymbol{\mu}^*)} \varphi_i(\boldsymbol{\mu}^*)\boldsymbol{\Xi}_i, \tag{15}$$

where $\boldsymbol{\Xi}^*$ is the ID model coefficient matrix for the test parameter point $\boldsymbol{\mu}^*$, and $\varphi_i(\boldsymbol{\mu}^*)$ represents the interpolation weights for each trained $\boldsymbol{\Xi}_i$ based on the location of each $\boldsymbol{\mu}_i$. In the KNN method, the distance is determined by the Euclidean distance between the parameter points. Specifically, the distance between parameter points $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}^*$ is defined as

$$d(\boldsymbol{\mu}_i,\boldsymbol{\mu}^*) = \sqrt{\sum_{j=1}^{N_{\boldsymbol{\mu}}}(\boldsymbol{\mu}_{i,j}-\boldsymbol{\mu}_j^*)^2}, \tag{16}$$

where $\boldsymbol{\mu}_{i,j}$ and $\boldsymbol{\mu}_j^*$ are the components of $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}^*$ respectively. The interpolation weights can be expressed as

$$\varphi_i(\boldsymbol{\mu}^*) = \frac{d(\boldsymbol{\mu}_i,\boldsymbol{\mu}^*)^{-p}}{\sum_{\boldsymbol{\mu}_j\in\mathcal{D}(\boldsymbol{\mu}^*)} d(\boldsymbol{\mu}_j,\boldsymbol{\mu}^*)^{-p}}, \tag{17}$$

in which p is the distance weighting index, generally taken as $p=2$.

Note that the interpolation is actually performed on a per-parameter-point basis, and the ID model coefficients are predicted sequentially rather than all at once to ensure accuracy. If there exists a $\boldsymbol{\mu}_i$ such that $d(\boldsymbol{\mu}_i,\boldsymbol{\mu}^*)\approx 0$, then it is concluded that $\boldsymbol{\mu}^*$ is contained in $\mathcal{D}_{train}$, and specific $\boldsymbol{\Xi}_i$ is directly assigned to $\boldsymbol{\Xi}^*$. The choice of k also affects the final prediction results. Following a preliminary screening process, the value k is determined to be 5. The detailed process of the KNN method can be found in Algorithm 1.

6. Data pre-processing

6.1. Data sampling

There are various methods for the selection of the training parameter set $\mathcal{D}_{\text{train}}$. To ensure training efficiency, commonly applied methods include greedy sampling based on training results or physical information[41, 43] and uniform sampling[42]. The goal of these sampling methods is typically to improve the final performance of algorithms with as few parameter points as possible, especially for those based on autoencoders. This is because, compared to TLDNet, these algorithms, while also having an excellent ability to capture underlying dynamics, require more network parameters and take up more GPU memory, resulting in longer computation times and the need for more computational resources.

In our work, the random sampling method is used to choose $\mathcal{D}_{train}$. For convenience and to observe the performance of P-TLDINet across the entire parameter domain $\mathcal{D}$, we uniformly sample a set of discrete parameter points $\mathcal{D}^h\subset\mathcal{D}$, with the number of points far exceeding that of $\mathcal{D}_{train}$ (i.e. $|\mathcal{D}^h|>|\mathcal{D}_{train}|$)), to serve as the testing sample set.

Algorithm 1: IDW interpolation based on KNN

Input: The training parameter set $\mathcal{D}_{train}$, the single testing parameter point $\boldsymbol{\mu}^*$, the k value, and the trained ID model coefficients $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i \in \mathcal{D}_{train}}$;

Output: The interpolated target y^*, k nearest neighbour set $\mathcal{D}(\boldsymbol{\mu}^*) \subset \mathcal{D}_{train}$, and the interpolation weights $\{\varphi_i(\boldsymbol{\mu}^*)\}_{\boldsymbol{\mu}_i \in \mathcal{D}(\boldsymbol{\mu}^*)}$.

Calculate distances between every $\boldsymbol{\mu}_i \in \mathcal{D}_{train}$ and $\boldsymbol{\mu}^*$ by Equation 16;
Rank the distance values in ascending order as a list $Dis(\boldsymbol{\mu}^*)$;
if value 0 is found in the list $Dis(\boldsymbol{\mu}^*)$ **then**
    Indicate that $\boldsymbol{\mu}^*$ lies in $\mathcal{D}_{train}$ and find the index j such that $\boldsymbol{\mu}_j \in \mathcal{D}_{train}$ is nearly equivalent to $\boldsymbol{\mu}^*$;
    Assign the corresponding $y_j \in \{y_i\}_{\boldsymbol{\mu}_i \in \mathcal{D}_{train}}$ to y^* directly;
else
    Search for the k nearest parameter points in $\mathcal{D}_{train}$ to $\boldsymbol{\mu}^*$ and combine them into a set $\mathcal{D}(\boldsymbol{\mu}^*)$;
    Compute the IDW weight $\varphi_i(\boldsymbol{\mu}^*)$ by Equation 17 for $\forall \boldsymbol{\mu}_i \in \mathcal{D}(\boldsymbol{\mu}^*)$;
    Search for the corresponding k target y based on the parameter points $\boldsymbol{\mu} \in \mathcal{D}(\boldsymbol{\mu}^*)$ and denote them as $\{\boldsymbol{\Xi}_i\}_{i=1}^k$;
    Compute the target interpolated ID model coefficient $\boldsymbol{\Xi}*$ by

$$\boldsymbol{\Xi}^* = \sum_{i=1}^{k} \varphi_i(\boldsymbol{\mu}^*)\boldsymbol{\Xi}_i. \tag{18}$$

end if
return $\boldsymbol{\Xi}^*$, $\mathcal{D}(\boldsymbol{\mu}^*)$, $\{\varphi_i(\boldsymbol{\mu}^*)\}_{\boldsymbol{\mu}_i \in \mathcal{D}(\boldsymbol{\mu}^*)}$.

6.2. Normalisation

To enhance the training efficiency of the networks, we preprocess the training data. Especially, there is a need to normalise the inputs and outputs of P-TLDINet. These data include the parameter vector $\boldsymbol{\mu}$, the scalar field $\boldsymbol{u}$, and the spatial grid coordinate variable $\boldsymbol{x}$. Each element is scaled to fit within a specific closed interval, usually chosen as $[-1, 1]$. Based on the data distribution, the normalisation for each variable is conducted independently. The mathematical process is defined as follows:

$$\overline{\beta} = (\beta - \beta_{ref})/\beta_w, \tag{19}$$

where β_{ref} is a reference value and β_w is a reference variable range. Particularly, if a variable takes values in the closed interval $[\alpha_{min}, \alpha_{max}]$, then

$$\begin{cases} \beta_{ref} = (\alpha_{min} + \alpha_{max})/2 \\ \beta_w = (\alpha_{max} - \alpha_{min})/2. \end{cases} \tag{20}$$

If $\alpha_{min} = -\alpha_{max}$, indicating that the origin is centred in $[\alpha_{min}, \alpha_{max}]$, a range hyper-parameter $\mathcal{L}_{range}$ can be set to adjust the normalisation range as follows:

$$\begin{cases} \beta_{ref} = \mathcal{L}_{range}(\alpha_{min} + \alpha_{max})/2 \\ \beta_w = \mathcal{L}_{range}(\alpha_{max} - \alpha_{min})/2. \end{cases} \tag{21}$$

After normalisation, the neural network can be expressed as:

$$\begin{cases} \widetilde{NN}_{dyn}(\boldsymbol{z}(t), \boldsymbol{\mu}) = NN_{dyn}(\boldsymbol{z}(t), (\boldsymbol{\mu} - \boldsymbol{\mu}_{ref}) \odot (1/\boldsymbol{\mu}_w)) \\ \widetilde{NN}_{z_0}(\boldsymbol{\mu}) = NN_{z_0}((\boldsymbol{\mu} - \boldsymbol{\mu}_{ref}) \odot (1/\boldsymbol{\mu}_w)) \\ \widetilde{NN}_{rec}(\boldsymbol{z}(t), \boldsymbol{\mu}, \boldsymbol{x}) = NN_{rec}(\boldsymbol{z}(t), \boldsymbol{\mu}, (\boldsymbol{x} - \boldsymbol{x}_0) \odot (1/\boldsymbol{x}_0)), \end{cases} \tag{22}$$

where $\odot$ denotes the Hadamard product, i.e. element-wise product. This normalisation ensures that the input variables are within a standardised range, thereby facilitating more efficient network training. If normalisation is removed, the networks can generally still achieve the fitting effect. However, it often requires more epochs of training and computational time.

7. The framework of P-TLDINet

The individual components of P-TLDINet have been described above. After preprocessing data, P-TLDINet focuses on obtaining a family of ODE coefficients that can be used for parametric learning by training both the TLDNet and the ID model simultaneously. That is achieved by the total loss function as:

$$Loss_{model} = Loss_{rec}(W_{rec}, W_{dyn}) + \omega_{z_0} Loss_{z_0}(W_{z_0}) + \omega_{ID} Loss_{dz/dt}(\boldsymbol{\Xi}) + \omega_{coef} \frac{1}{N_{\boldsymbol{\mu}}} \sum_{i=1}^{N_{\boldsymbol{\mu}}} ||\boldsymbol{\Xi}_i||_2^2, \tag{23}$$

where the last term $\frac{1}{N_{\boldsymbol{\mu}}} \sum ||\boldsymbol{\Xi}_i||_2^2$ is a Ridge regularization penalty on the SINDy coefficient matrices. In standard SINDy, Lasso is commonly used to enforce sparsity by driving small coefficients exactly to zero. However, in this work, we employ a first-order SINDy library, which contains relatively few candidate terms. Our primary goal is therefore to obtain a stable and simple family of SINDy coefficients that faithfully captures the essential dynamics, rather than to pursue extreme sparsity at the cost of stability. Ridge regularization provides better numerical stability and preserves the dominant dynamical features, which is crucial for the subsequent parametric learning in the coefficient space.

The training process is computationally intensive but is performed entirely offline; once the model is trained, online predictions are obtained almost instantaneously without high computational cost. Then, the KNN interpolation method is used based on the distribution characteristics of the testing parameters, and a series of latent parametric governing equations for each training parameter $\boldsymbol{\mu}$ are solved to predict latent state variables $\tilde{\boldsymbol{z}}$ The high-dimensional approximate flow field $\tilde{\boldsymbol{u}}$ can be quickly simulated using the NN_{rec} model and $\tilde{\boldsymbol{z}}$. This part is an online process, and it only takes a short time to reach the desired results. In addition, trained P-TLDINet allows for direct predictions of $\boldsymbol{u}$ based on the inputs of NN_{rec}, which are independent spatial coordinates $\boldsymbol{x}^*$ and the testing parameter point $\boldsymbol{\mu}^*$. The whole process refers to Equation 22.

According to the error between the estimated vector $\tilde{\boldsymbol{u}}$ and the true value $\boldsymbol{u}$, the parametric prediction and spatial prediction effectiveness of the model can be assessed by L_2 reconstruction rate as:

$$r_{L_2}(\tilde{\boldsymbol{u}}, \boldsymbol{u}) = \frac{||\tilde{\boldsymbol{u}} - \boldsymbol{u}||_2}{||\boldsymbol{u}||_2} \times 100\%, \tag{24}$$

where $||\cdot||_2$ stands for L_2 norm of matrices.

A stopping criterion is set for $r_{L_2}(\tilde{\boldsymbol{u}}, \boldsymbol{u})$ and $Loss_{model}$, which would stop training when they are smaller than the hyperparameters tol_1 and tol_2 set up in advance, respectively.

7.1. Off-line model training

The details mentioned in the previous sections together form the P-TLDINet algorithm based on parametric learning. In practice, after randomly selecting a parameter points subset $\mathcal{D}_{train} \subset \mathcal{D}$, a small amount of high-fidelity snapshots is generated by simulation techniques as training data. These data are then fed into the P-TLDINet model after going through preprocessing, especially the normalisation process in Section 6.2. A series of latent governing equations with the form of Equation 8 are identified to describe the intrinsic dynamics. The coefficient matrices $\{\boldsymbol{\Xi}_i\}$ of Equation 8 for each $\boldsymbol{\mu}_i \in \mathcal{D}_{train}$, treated as trainable variables within the networks, are derived and compiled into a set for subsequent online computations as a significant ingredient of the parametric prediction. The specific procedure can be referred to as Algorithm 2 and the flow chart is in Figure 1.

7.2. Online stage with the reconstruction model and ID models

The trained P-TLDINet model is then used for efficient predictions of estimated solutions for a given parameter. During the online stage, predictions can be made for unknown parameter points or even spatial grid coordinates that were only partially seen in the training data, with reduced time and cost. We adopt the classical fourth-order Runge-Kutta (RK4) method to solve the identified SINDy equations with a fixed time step size Δt_{online}. By combining $\boldsymbol{\mu}^*$ and $\boldsymbol{x}^*$ information with the set of coefficients $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i \in \mathcal{D}_{train}}$, high-dimensional estimates are obtained through KNN convex interpolation and the trained network NN_{rec} and NN_{z_0}. The detailed process is shown in Algorithm 3 and Figure 2.

The source code implementing the P-TLDINet framework described in this work is publicly available at https://github.com/iSelinG/Parametric_data_discovery_in_latent_space.

Algorithm 2: P-TLDINet trained with random sampling

Input: The training parameter set $\mathcal{D}_{train}$, the high-dimensional simulated field data $\{\boldsymbol{u}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$, the spatial coordinates information $\boldsymbol{x}$, the discrete time t, the max iteration $iter$, the target tolerance tol_1 and tol_2, the number of epochs to test the tolerance N_{tol}, the weight ω_{ID} of $Loss_{dz/dt}$, the weight ω_{z_0} of $Loss_{z_0}$, the weight ω_{coef} of $Loss_{coef}$;

Output: The model coefficient W_{rec}, W_{dyn} and W_{z_0}, the coefficient matrices $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$.

Preprocess $\boldsymbol{\mu}_i \in \mathcal{D}_{train}$, $\{\boldsymbol{u}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$ and $\boldsymbol{x}$ by normalisation in Section 6.2;
Construct P-TLDINet model according to the nature of the problem and feed it with normalised $\boldsymbol{\mu}_i$, $\boldsymbol{u}_i$ and $\boldsymbol{x}$;
while $k \leq iter$ **do**
  Update W_{rec}, W_{dyn}, W_{z_0} and the sparse identification coefficients $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$ simultaneously by minimising $Loss_{model}$ in Equation 23;
  if $k\%N_{tol} = 0$ **then**
    Solve the latent governing equations for each training parameter $\boldsymbol{\mu}_i$ as

$$\frac{d\tilde{\boldsymbol{z}}_i}{dt} = \boldsymbol{\Theta}(\tilde{\boldsymbol{z}}_i)\cdot\boldsymbol{\Xi}_i, \quad \boldsymbol{\mu}_i \in \mathcal{D}_{train} \tag{25}$$

    to obtain the approximated latent trajectories $\{\tilde{\boldsymbol{z}}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$.
    The corresponding high-fidelity field $\tilde{\boldsymbol{u}}_i$ is then reconstructed by feeding $\tilde{\boldsymbol{z}}_i$ through the reconstruction network NN_{rec} as $\tilde{\boldsymbol{u}}_i = NN_{rec}(\tilde{\boldsymbol{z}}_i, \boldsymbol{\mu}_i, \boldsymbol{x})$;
    if $r_{L_2}(\tilde{\boldsymbol{u}}_i, \boldsymbol{u}_i) \leq tol_1$ and $Loss_{model} \leq tol_2$ **then**
      break
    end if
  end if
  $k \leftarrow k+1$
end while
return W_{rec}, W_{dyn}, W_{z_0}, $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$.

Algorithm 3: Evaluation and online stage of the P-TLDINet model

Input: A testing parameter set $\mathcal{D}^h$, the training parameter set $\mathcal{D}_{train}$, the trained reconstruction model coefficient W_{rec}, the trained ODE coefficient matrices $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i\in\mathcal{D}_{train}}$, spatial coordinate information $\boldsymbol{x}^*$ to be predicted ;

Output: The approximate fields $\{\boldsymbol{u}(\boldsymbol{\mu}_j)\}_{\boldsymbol{\mu}_j\in\mathcal{D}^h}$.

for $\forall\boldsymbol{\mu}^* \in \mathcal{D}^h$ **do**
  for the m-th coefficient in ID model coefficient matrix $\boldsymbol{\Xi}$ in Equation 11 **do**
    Search for the corresponding k target coefficients based on the parameter points $\boldsymbol{\mu} \in \mathcal{D}(\boldsymbol{\mu}^*)$ and denote them as $\{\xi_i^{(m)}\}_{i\in\mathcal{D}(\boldsymbol{\mu}^*)}$;
    Compute KNN weights by Algorithm 1 and calculate the m-th interpolated coefficient $\xi^{*(m)}$ by

$$\xi^{*(m)} = \sum_{\boldsymbol{\mu}_i\in\mathcal{D}(\boldsymbol{\mu}^*)} \varphi_i(\boldsymbol{\mu}^*)\xi_i^{(m)}; \tag{26}$$

  end for
  Compute $\boldsymbol{\Xi}^* = \{\xi^{*(m)}\}_{m=1}^{N_s\cdot N_b}$ based on the above calculations;
  Solve Equation 8 by Runge-Kutta method and obtain $\{\tilde{\boldsymbol{z}}_j\}_{\boldsymbol{\mu}_j\in\mathcal{D}^h}$;
  Reconstruct the approximate solution field $\boldsymbol{u}(\boldsymbol{\mu}_j)$ by the reconstruction model NN_{rec} , $\boldsymbol{x}^*$ and inverse normalisation as

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}^*, t; \boldsymbol{\mu}_j) = NN_{rec}(\tilde{\boldsymbol{z}}, \boldsymbol{\mu}, \boldsymbol{x}^*) \tag{27}$$

  .
end for
return $\{\boldsymbol{u}(\boldsymbol{\mu}_j)\}_{\boldsymbol{\mu}_j\in\mathcal{D}^h}$.

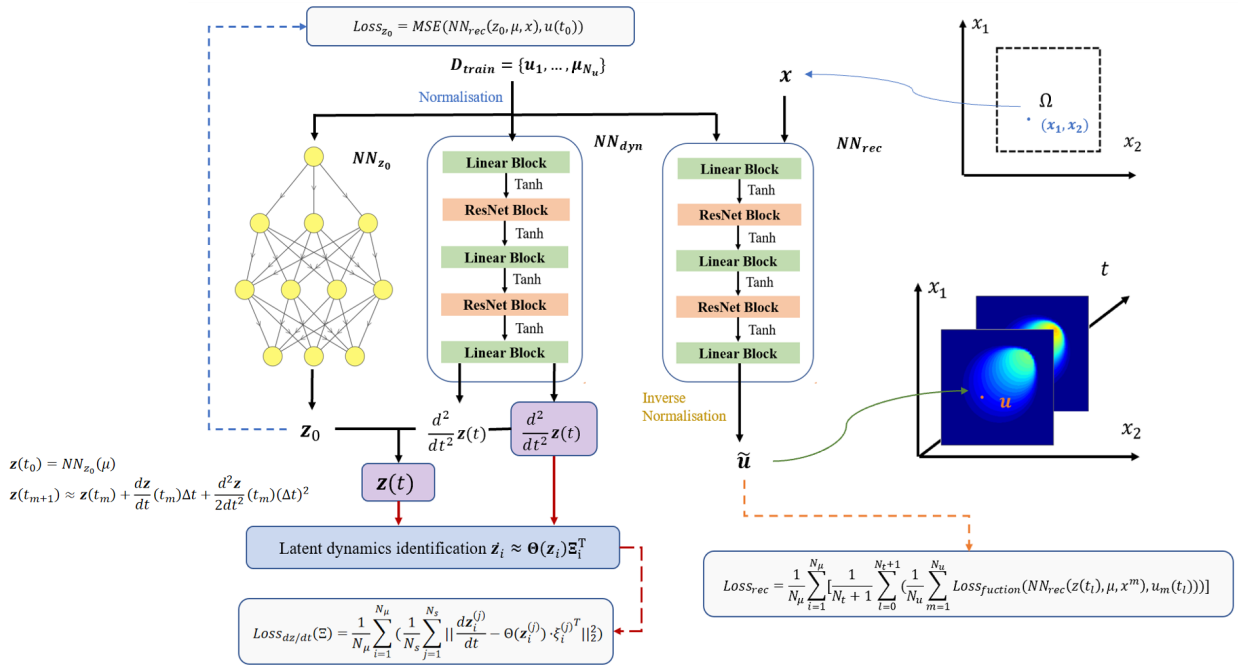


Figure 1: Flowchart of the P-TLDINet algorithm.

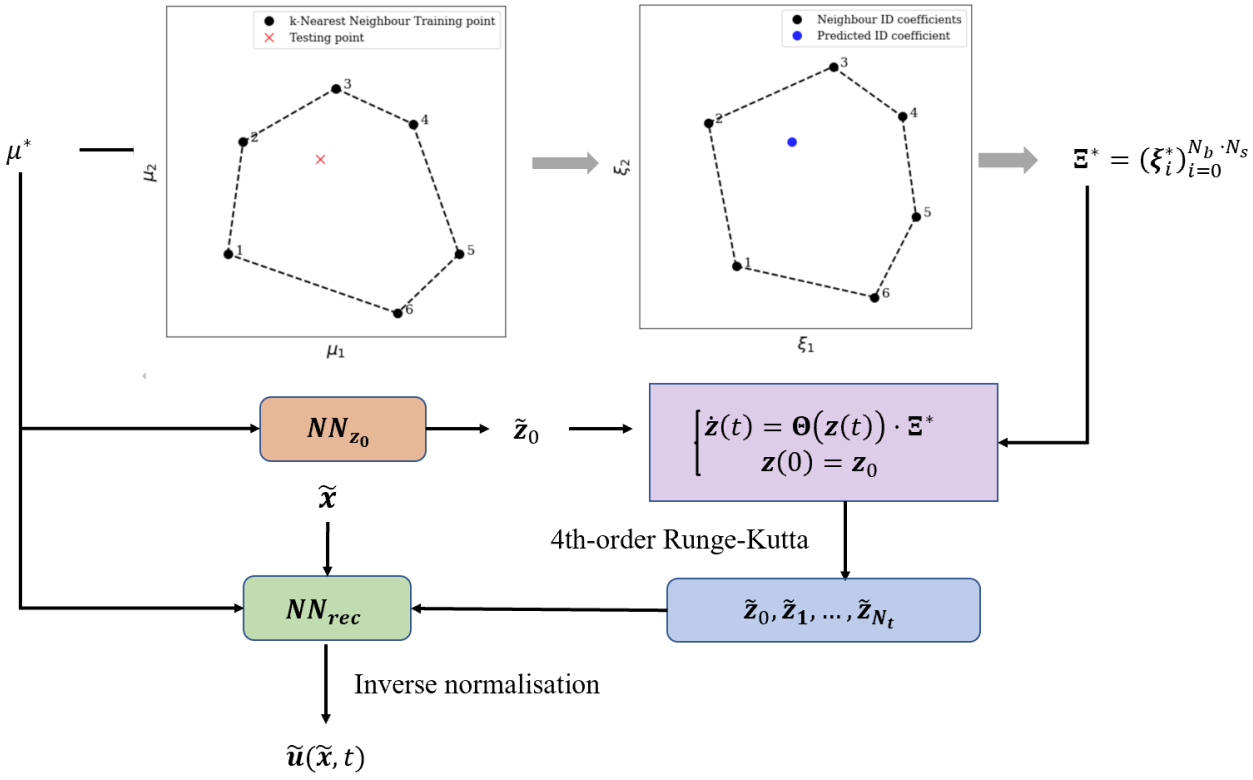


Figure 2: Online stage of the P-TLDINet algorithm.

8. Numerical Results

This section demonstrates the application of P-TLDINet. We explore P-TLDINet using two numerical examples: the 2D-Burgers equation and lock exchange. The Burgers equation uses a uniform grid, whereas the lock-exchange problem uses an unstructured triangular mesh, ensuring that both are capable of high-fidelity numerical solutions. We depict the prediction performance and training speed of P-TLDINet with unknown testing parameters.

In presenting the results of the first example, this method is compared with several dynamics identification methods based on autoencoders. Additionally, two parameter ranges are employed to demonstrate the generalisation ability of this framework. To further illustrate this capability, we include a dedicated subsection for each of the two numerical examples, where the model is trained on data from multiple mesh resolutions.

The training of P-TLDINet and the compared methods is performed on an NVIDIA GeForce RTX 4090 (Ada Lovelace) with a 24 GB GDDR6X GPU.

8.1. Case 1: 2D-Burgers equation

The 2D parametric Burgers equation is widely used to describe shock waves and turbulence phenomena in viscous fluids. We consider this equation with a viscous diffusion term, which includes a governing equation and essential boundary conditions. It can be summarised as

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} &= \frac{1}{Re} \Delta \boldsymbol{u}, \quad (x_1, x_2) \in \Omega, t \in [0, 1] \\ \boldsymbol{u}(x_1, x_2, t = 0; \boldsymbol{\mu}) &= 0, \quad (x_1, x_2) \in \partial\Omega \end{aligned} \tag{28}$$

where $\Omega = [-3, 3] \times [-3, 3]$ and the Reynolds number $Re = 10000$. The symbol $\boldsymbol{u} = (u, v)$ represents a vector field, where u and v are the fluid velocity components in the x and y directions, respectively. The initial condition determines the evolution of Equation 28. In this case, it is parametrised by $\boldsymbol{\mu} = \{a, \omega\}$ and defined as

$$\boldsymbol{u}(t = 0, x_1, x_2) = a \exp\left(-\frac{x_1^2 + x_2^2}{2\omega^2}\right), \tag{29}$$

causing the initial values to remain unequal even after normalisation. In this test case, both velocity components u and v are assigned the same initial condition given by Equation 29. This choice simplifies the problem while preserving its essential nonlinear characteristics, and is consistent with the typical symmetric setup of the 2D Burgers problem. The high-fidelity solver uses a backward finite-difference discretisation in space and an implicit backward Euler integration scheme in time. In the domain Ω, a spatial step size of 50 is used for each edge, and a temporal step number of 200 is used, resulting in $\Delta x_1 = \Delta x_2 = 0.12$ and $\Delta t = 5 \times 10^{-3}$.

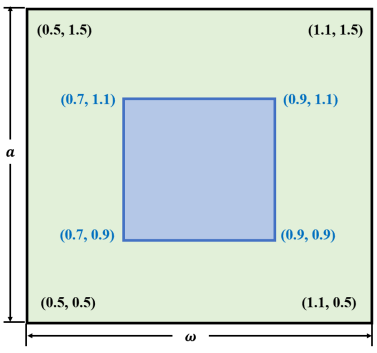


Figure 3: **2D-Burgers**: The inner small blue square represents the range of the small parameter set; the outer large green square represents the range of a large parameter set (includes the blue square area).

To demonstrate the generality of the P-TLDINet method, we choose two parameter ranges for comparison, namely a small dataset $\mathcal{D}_1 = [0.7, 0.9] \times [0.9, 1.1]$ and a large dataset $\mathcal{D}_2 = [0.5, 1.1] \times [0.5, 1.5]$, as shown in detail in Figure 3. Here, $\mathcal{D}_1$ represents the parameter range selected for testing the performance of the model in [41] and [43]. We have further expanded this range into $\mathcal{D}_2$ in our work.

8.1.1. Trained on a small dataset $\mathcal{D}_1$

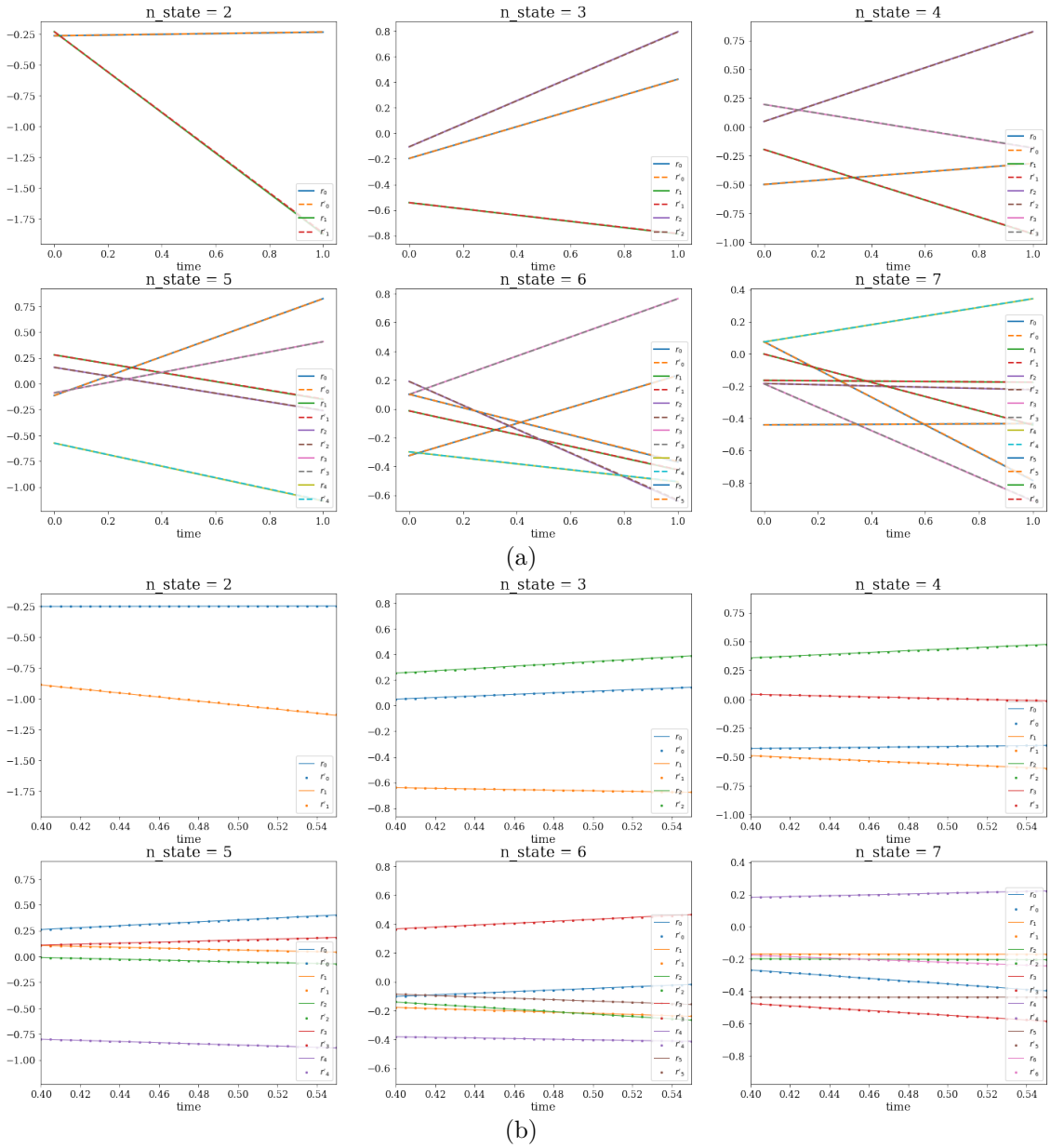


Figure 4: **2D-burgers**: Low-dimensional dynamics and the solutions of the identified equations in the small dataset $\mathcal{D}_1$. (a) and (b) show the dimension of the latent space N_s that ranges from 2 to 7. (a) represents the results for the entire time frame $t \in [0, 1.0]$, while (b) shows only in $t \in [0.4, 0.6]$.

In the first example, P-TLDINet is trained on a small data set $\mathcal{D}_1$, where the dimensionality of the latent space N_s ranges from 2 to 7 to reveal some special features of the reduction mapping in TLDNet. The input size of the dynamical network NN_{dyn} is $N_s + N_D = N_s + 2$ and the output size is $2N_s$. The specific structure of NN_{dyn} is:

$$\begin{aligned} NN_{dyn}:&\text{Linear}(\mathrm{N_s}+2,6) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(6,6) \xrightarrow{\text{Tanh}} \text{Linear}(6,6) \xrightarrow{\text{Tanh}} \\ &\text{ResNetBlock}(6,6) \xrightarrow{\text{Tanh}} \text{Linear}(6,2\mathrm{N_s}), \end{aligned} \tag{30}$$

where "Linear" is an FCNN and "ResNetBlock" refers to a ResNet block. The input size of the reconstruction network NN_{rec} is $N_S + 2 + N_D = N_s + 4$ (2 denotes the dimension of the spatial domain Ω), and the output

size is the number of flow fields in $\boldsymbol{u}$, which is 2 in this problem. The specific structure of NN_{rec} is:

$$\begin{aligned} NN_{rec}:&\text{Linear}(\mathrm{N_s}+4,11) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(11,15) \xrightarrow{\text{Tanh}} \text{Linear}(15,15) \xrightarrow{\text{Tanh}} \\ &\text{ResNetBlock}(15,15) \xrightarrow{\text{Tanh}} \text{Linear}(15,15) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(15,11) \xrightarrow{\text{Tanh}} \\ &\text{Linear}(11,2). \end{aligned} \tag{31}$$

The initial mapping network NN_{z_0} is expressed as:

$$NN_{z_0}: \text{Linear}(2,6) \xrightarrow{\text{Tanh}} \text{Linear}(6,6) \xrightarrow{\text{Tanh}} \text{Linear}(6,6) \xrightarrow{\text{Tanh}} \text{Linear}(6,\mathrm{N_s}). \tag{32}$$

To identify dynamics in the latent space $\mathcal{S}$, linear terms and a constant term comprise the library $\boldsymbol{\Theta}(\cdot)$ in Equation 8, leading to $N_b = 6$. The coefficient matrix $\boldsymbol{\Xi}$ is contained in the network as trainable variables. Compared to the autoencoder, this is a sufficiently lightweight structure with fewer model trainable parameters, leading to more rapid model training and less use of random access units.

The P-TLDINet model is trained by the Adam optimiser for 6000 iterations, with $N_{tol} = 500$. To further refine the training, a learning rate scheduler (StepLR) is implemented with an initial learning rate $\alpha = 0.05$, which decreases α by a factor of 0.6 every 500 iterations. 25 samples are randomly selected to estimate the latent dynamics. The loss weights are set as $\omega_{DI} = 0.05$, $\omega_{coef} = 0$, and $\omega_{z_0} = 0.5$. For simplicity and symmetry, all velocity fields shown in the following figures correspond to the u velocity component; the v component exhibits similar behavior and is therefore omitted for brevity.

N_s	2	3	4	5	6	7
L_2 error (%)	0.7504%	0.7010%	0.7531%	0.6623%	0.8467%	0.7951%

Table 2: **2D-Burgers**: Total L_2 errors on testing samples $\mathcal{D}^h \subset \mathcal{D}_1$ of different N_s, which are the dimensions of the latent dynamics.

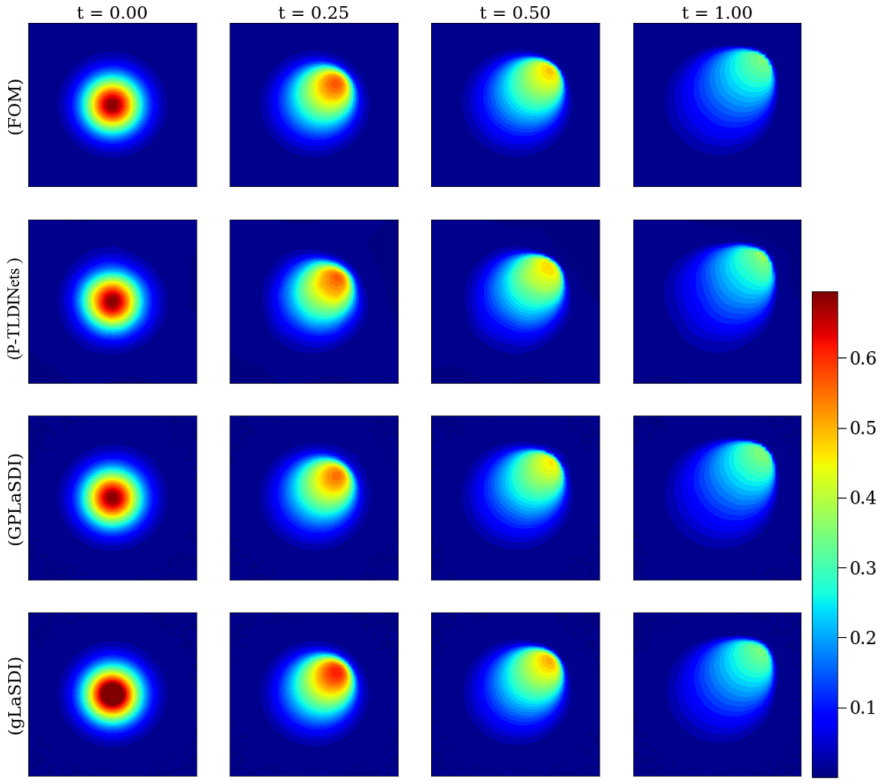
The results of different N_s are shown in Figure 4. To evaluate the reconstruction performance of NN_{rec} and $\{\boldsymbol{\Xi}_i\}_{\boldsymbol{\mu}_i \in \mathcal{D}_1}$ with the KNN interpolation method, 225 test samples in $\mathcal{D}_1$ are adopted as $\mathcal{D}_1^h$. The overall L_2 reconstruction errors are shown in Table 2. It can be seen that P-TLDINet still demonstrates sufficient compression and learning capabilities even with different N_s, but the reconstruction efficiency did not improve significantly with the increase of N_s. It takes 1.3 hours to obtain high-dimensional solutions for the entire testing set $\mathcal{D}_1^h$ for the solver, whereas P-TLDINet only requires 36.35 minutes of training time and 57.02 seconds to predict on $\mathcal{D}_1^h$, achieving a striking acceleration of 94.7× speed-up. From Table 2, P-TLDINet reaches a reconstruction accuracy of 99% on the test dataset for every latent dimension.

In Figure 5, we compare the model with advanced reduced-order algorithms, GPLaSDI[43] and gLaSDI[41], which are based on autoencoders and the identification of dynamics. We pick $\boldsymbol{\mu}^* = [0.71, 1.04]$ randomly to show the predicted solutions of different methods at several time points when $N_s = 5$. Since u and v in $\boldsymbol{u}$ are highly similar in the initial conditions of Equation 29, only the variation of u is shown here. All three methods roughly restore the trends of the flow fields, and Figure 5(b) shows that their errors remain within a very narrow range. Their reconstruction rates for all training samples are shown in Table 3. However, as shown in Figure 6, gLaSDI and GPLaSDI contain far more trainable parameters and require more training time compared to P-TLDINet. This is due to the nature of autoencoders. To further demonstrate the effects of P-TLDINet, a large parameter dataset $\mathcal{D}_2$ is employed in Section 8.1.2.

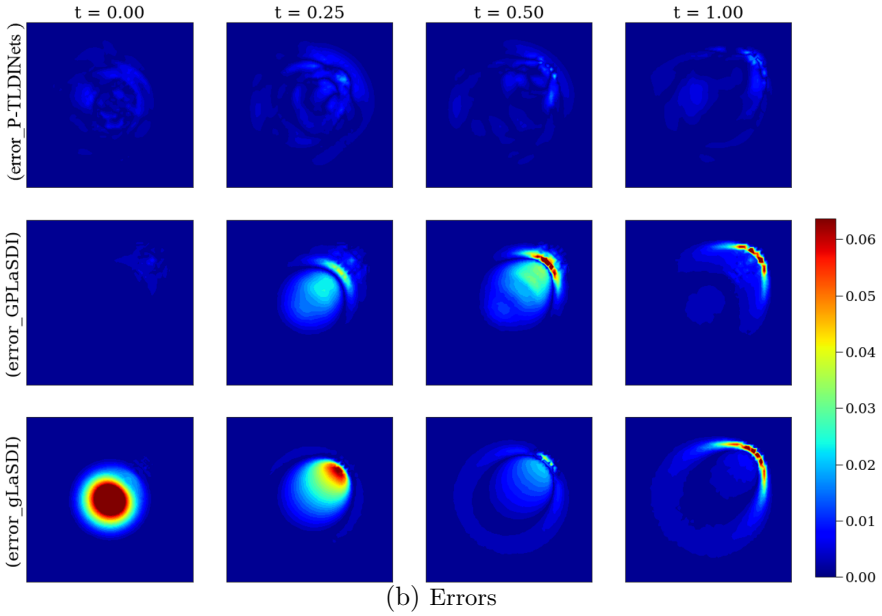
8.1.2. Trained on a large dataset $\mathcal{D}_2$

For the Burgers equation, a larger range of parameters implies more complex dynamics. The network structure and training information are the same as those in the previous section. A new testing parameter point set, $\mathcal{D}_2^h$, also containing 225 sample parameters, is selected from $\mathcal{D}_2$.

Figure 7 illustrates the low-dimensional temporal dynamics of P-TLDNet for N_s ranging from 2 to 7 as well. Figure 8 presents the prediction results of GPLaSDI, gLaSDI and P-TLDINet for the testing parameter $\boldsymbol{\mu} = [1.0, 1.29]$ when $N_s = 5$. In $\mathcal{D}_2$, notice the change in the upper and lower bounds of the colour bar in Figure 8 (b), which implies that methods based on autoencoders fail to accurately generate high-dimensional



(a) Solutions



(b) Errors

Figure 5: **2D-burgers**: Results for the u velocity component. (a) shows high-fidelity velocity solutions and reconstruction results of P-TLDINet, gLaSDI and GPLaSDI on $\boldsymbol{\mu}^* = [0.71, 1.04]$ in $\mathcal{D}_1^h$ when $N_s = 5$. (b) demonstrates point-wise errors between three types of methods and the true value.

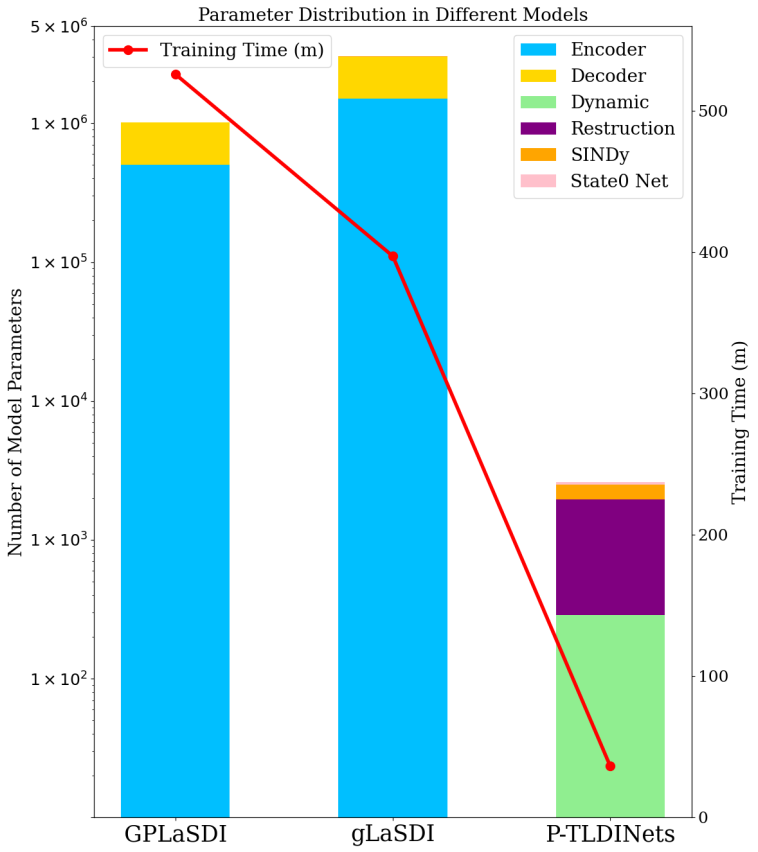


Figure 6: **2D-Burgers**: The bar chart shows the trainable model parameters and their compositions for GPLaSDI, gLaSDI, and P-TLDINet; the red line indicates the required model training time. GPLaSDI and gLaSDI consist of an encoder and a decoder, whereas P-TLDINet replaces these with a combination of a dynamic network, a reconstruction network, and z_0 network.

approximate solutions as t evolves. In contrast, P-TLDINet continues to demonstrate reconstruction capabilities similar to those on the smaller datasets $\mathcal{D}_1$. Table 3 shows the reconstruction rates for three methods. It is obvious that when the parameter range becomes larger, P-TLDINet maintains its efficiency, while the other methods may fail.

To delve deeper into the causes of this phenomenon, Figure 9 depicts the low-dimensional states of the other two methods. It reveals that in the larger parameter space $\mathcal{D}_2$, the states learnt by autoencoders show more complex forms, which limit the identification ability of models and potentially cause divergence when solving the latent governing equations. Comparing Figure 4 with Figure 7, it is clear that the latent performances of P-TLDINet are consistently smoother than those learnt by autoencoders, regardless of the dimensionality of the latent space. This is due to the properties of TLDNet. Therefore, this method is more generalisable.

L_2 error (%)	2D-Burgers ($\mathcal{D}_1$)	2D-Burgers ($\mathcal{D}_2$)
P-TLDINet	0.66%	0.86%
GPLaSDI	0.19%	35.90%
gLaSDI	1.27%	43.07%

Table 3: **2D-Burgers**: L_2 errors on all testing samples on $\mathcal{D}^h \subset \mathcal{D}_1$ and $\mathcal{D}^h \subset \mathcal{D}_2$ for different methods, when $N_s = 5$.

A notable observation from Figure 4 and Figure 7 is that the latent trajectories of P-TLDINet exhibit approximately linear behaviour, which requires further discussion. This near-linearity does *not* indicate that the dynamics network NN_{dyn} has collapsed to a constant; rather, it suggests that for the Burgers equation

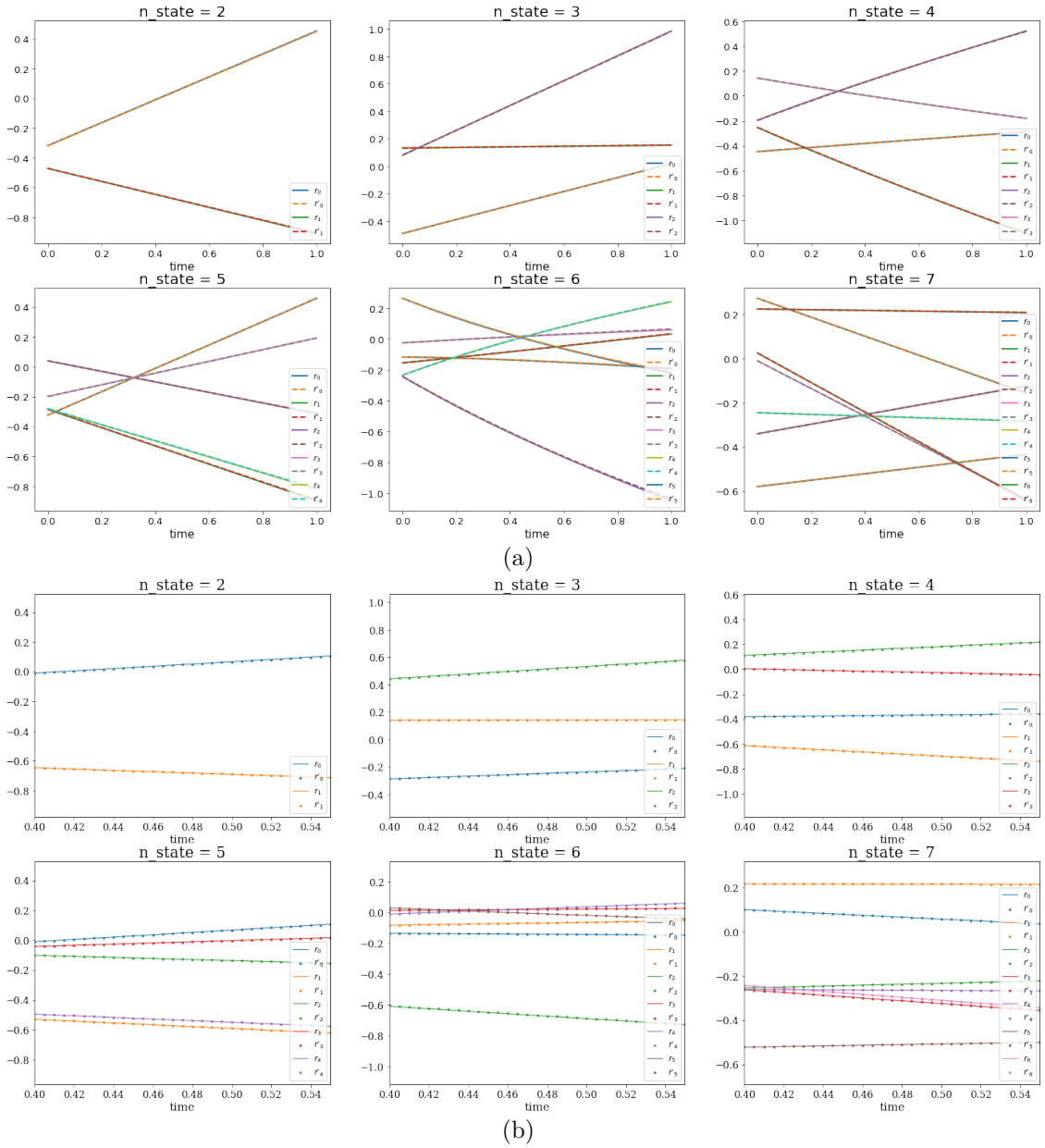


Figure 7: **2D-burgers**: Low-dimensional dynamics and the solutions of the identified equations on the large dataset $\mathcal{D}_2$. (a) and (b) show the dimension of latent space N_s ranging from 2 to 7. (a) represents results for the entire time frame $t \in [0, 1.0]$, while (b) shows only in $t \in [0.4, 0.6]$.

under the chosen parameter regime, the system's essential dynamics in the low-dimensional latent space are indeed nearly linear.

Although the 2D Burgers equation is inherently nonlinear due to the advection term $(\mathbf{u} \cdot \nabla)\mathbf{u}$, this does not preclude its latent dynamics from exhibiting linear characteristics. Linear latent dynamical models have also been used for the nonlinear 2D Burgers problem in LaSDI[42]. In gLaSDI[41], it can be observed that under the same example, the underlying dynamics exhibit similar quasi-linear characteristics as those shown in Fig. 4 and Fig. 7. From a theoretical perspective, the classical Cole-Hopf transformation exactly linearises the nonlinear Burgers equation into the linear heat equation, providing a mathematical foundation for the emergence of linear behaviour in this system.

8.1.3. Compared to POD-NN and LDNets

To further situate the proposed P-TLDINet within the existing literature, we compare it with two additional non-intrusive reduced-order modelling approaches: POD-NN[33] and LDNets[36].

POD-NN combines proper orthogonal decomposition (POD) with a neural network that maps the parameters $\boldsymbol{\mu}$ and time t directly to the POD coefficients, bypassing any dynamical system identification. This approach is conceptually simple and computationally efficient at test time, but it lacks any explicit modelling of the underlying dynamics, which limits its accuracy for highly nonlinear problems. LDNets, on the other hand, learns a latent dynamical system purely through neural network fitting, similar in spirit to our approach but without the subsequent sparsity-promoting distillation step. Its dynamics remain encoded in a black-box neural network, and the final model does not provide an interpretable governing equation.

Both methods are trained and tested on the same 2D-Burgers dataset under the large parameter domain $\mathcal{D}_2 = [0.5, 1.1] \times [0.5, 1.5]$, using the identical 47 training parameters and 225 test parameters as described in Section 8.1. For a fair comparison, the latent dimension is set to $N_s = 5$ for all methods, and the same number of training iterations is used. Table 4 reports the relative L_2 errors on the test set $\mathcal{D}_2^h$, together with the offline training time and the online prediction time for all 225 test parameters and 200 time nodes.

Method	L_2 error (%)	Training time (min)	Prediction time (s)
P-TLDINet	0.86%	36.35	57.02
POD-NN	3.92%	12.10	1.26
LDNets	0.92%	35.64	89.30

Table 4: **2D-Burgers** ($\mathcal{D}_2$): Performance comparison among P-TLDINet, POD-NN and LDNets.

POD-NN achieves the shortest training and prediction times because it avoids any time integration and directly outputs the POD coefficients via a single forward pass. On the large parameter domain $\mathcal{D}_2$, POD-NN achieves substantially lower errors than GPLaSDI (35.90%) and gLaSDI (43.07%), highlighting the strength of non-intrusive regression-based approaches for large parameter domains. However, its error of 3.92% remains higher than those of P-TLDINet (0.86%) and LDNets (0.92%).

LDNets achieves a moderate overall error that is comparable to that of P-TLDINet, which are 0.92% and 0.86%. The training times of the two methods are also close. However, a notable difference appears in the online prediction stage. LDNets requires both the dynamics network NN_{dyn} and the reconstruction network NN_{rec} to produce the final output at each query point. In contrast, P-TLDINet, after distillation, only needs the explicit SINDy coefficient matrix $\boldsymbol{\Xi}$ (which defines the governing equation), the reconstruction network NN_{rec}, and the initial state network NN_{z_0}, which means that the dynamics network NN_{dyn} is no longer required during inference. This difference leads to faster online prediction for P-TLDINet, as shown in Table 4.

Although both P-TLDINet and LDNets operate in the framework of learning latent-space dynamics, their methodological foundations and final outputs differ substantially in the following aspects.

First, and most fundamentally, the nature of the final surrogate model is different. LDNets retain the trained neural ODE (NN_{dyn}) as a black-box predictor, meaning that its dynamics are implicitly encoded in the network weights and cannot be expressed in closed form. In contrast, P-TLDINet introduces a dedicated distillation part that replaces the neural ODE with a sparse, explicit SINDy model. In the present case, Burgers functions are reduced to a simple affine form. This yields a closed-form dynamical system that is not only interpretable but also allows the complete low-dimensional state to be obtained directly without querying a black-box network. In other words, LDNets approximates the dynamics through pure neural network regression, whereas P-TLDINet approximates the dynamics through explicit functional forms, which is conceptually closer to function approximation than to black-box fitting.

Second, the treatment of the initial latent state differs fundamentally. LDNets assumes, without loss of generality, that the initial latent state is fixed to zero, i.e. $z(0) = 0$[36]. This assumption is justified by the hidden nature of $z(t)$, but it implies that all information about the physical initial condition must be encoded in the reconstruction network NN_{rec}. While this works well for problems with fixed initial conditions, it becomes problematic when the initial condition itself varies with parameters. For example, the case for the 2D-Burgers equation considered in this work, where the initial condition is parameterized by $\boldsymbol{\mu} = \{a, \omega\}$ (Equation 29). By contrast, P-TLDINet employs a dedicated network NN_{z_0} that explicitly predicts the

initial latent state $\mathbf{z}(0)$ from the input parameters $\boldsymbol{\mu}$:

$$\mathbf{z}(0) = NN_{z_0}(\boldsymbol{\mu}). \tag{33}$$

This design is physically more intuitive and directly handles parameter-dependent initial conditions. To quantify this difference, we conducted an additional experiment where LDNets was applied to the 2D-Burgers problem with parameterized initial conditions. As shown in Table 5, the overall prediction errors of the two methods are comparable. However, a notable difference appears in the reconstruction of the initial condition: LDNets exhibits significantly larger errors when the initial condition varies with parameters, whereas P-TLDINet maintains its accuracy. This discrepancy arises from the different treatments of the initial latent state. LDNets assumes $s(0) = 0$ and relies entirely on NN_{rec} to implicitly infer the initial physical field from the parameters, a task that becomes challenging when the initial condition is parameter-dependent. In contrast, P-TLDINet explicitly predicts $\mathbf{z}(0)$ from $\boldsymbol{\mu}$ via the dedicated network NN_{z_0}, which is trained specifically for this purpose, leading to more accurate initial field reconstruction. Despite this difference in initial condition recovery, both methods achieve similar overall errors, suggesting that the influence of the initial error diminishes as the system evolves.

Method	L_2 error at t_0 (%)	L_2 error of overall (%)
P-TLDINet	0.67%	0.86%
LDNets	2.73%	0.92%

Table 5: **2D-Burgers**: Comparison of reconstruction errors for P-TLDINet and LDNets in case of 2D-burgers on $\mathcal{D}_2$.

Third, the parameterization learning strategies are different. LDNets incorporates parameters $\boldsymbol{\mu}$ as inputs to both the dynamics and reconstruction networks, relying on the network's generalization capability to handle unseen parameters. P-TLDINet, in addition to using parameters as network inputs, further combines the distilled SINDy coefficients with a KNN interpolation scheme in the coefficient space. This enables smooth and robust interpolation across unseen parameters, which is particularly beneficial for problems with parameter-dependent initial conditions. Moreover, because the distilled SINDy model is evaluated directly without ODE integration, P-TLDINet achieves faster online prediction than LDNets, as discussed earlier in Table 4. The advantage becomes more pronounced when predictions are required at many time instants or over long time horizons, as each time step in LDNets requires an additional forward pass through NN_{dyn}, whereas P-TLDINet evaluates the closed-form SINDy model directly.

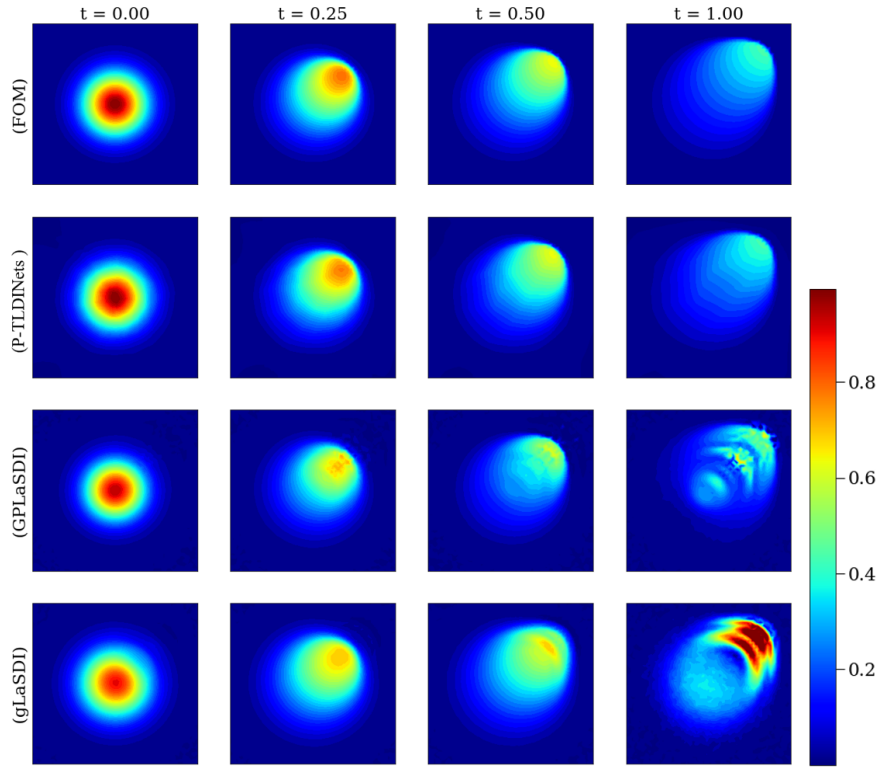
Fourth, the time discretization schemes differ. LDNets employs a first-order forward Euler scheme for time advancement, while P-TLDINet instead uses a second-order Taylor-series expansion. The Taylor-series scheme has a smaller local truncation error ($O(\Delta t^3)$) compared to the forward Euler scheme ($O(\Delta t^2)$). While this does not lead to a significant reduction in training time, it theoretically reduces the accumulation of errors for long-time predictions.

In summary, P-TLDINet strikes the best balance between accuracy and computational efficiency. It outperforms POD-NN in accuracy by a large margin and surpasses LDNets in both predicting speed and interpretability, while maintaining excellent generalisation from the small parameter domain $\mathcal{D}_1$ to the much larger domain $\mathcal{D}_2$.

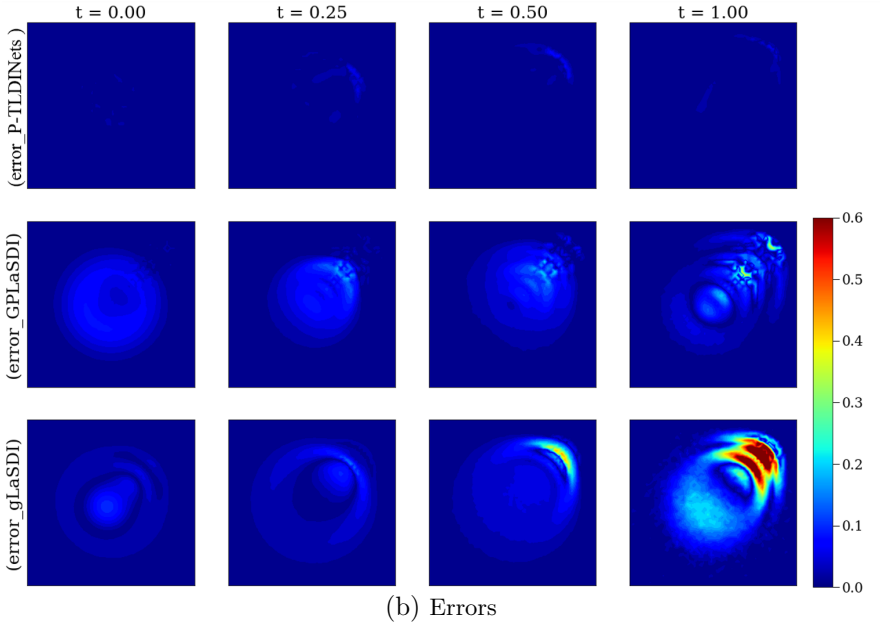
8.1.4. Trained on different resolutions of the meshes

This section demonstrates another capability of P-TLDINet, which is learning on different resolutions of the meshes. P-PDE problems with multiple resolutions are commonly found in physical simulations and engineering applications[53].

In this example, three types of square grids with different resolutions are fed into the model. The grids are shown in Figure 10. For simplicity, these grids are uniform and divided into 50, 60, and 70 segments along the x_1 axis and the x_2 axis respectively, resulting in 2500, 3600, and 4900 mesh points in total. The training data $\{\boldsymbol{u}\}$ are generated according to the random selection in $\mathcal{D}_2$. The other conditions and network structure are the same as those in the previous training. The trained model is then applied to make predictions on the grid (c) in Figure 10, which has 4900 nodes, achieving a reconstruction accuracy of 98.73%. Figure 11 shows the specific prediction results of P-TLDINet at $\boldsymbol{\mu}^* = [0.64, 1.21]$.



(a) velocity solutions



(b) Errors

Figure 8: **2D-Burgers**: Results for the u velocity component. (a) shows high-fidelity velocity solutions and reconstruction results of P-TLDINet, gLaSDI and GPLaSDI on $\boldsymbol{\mu}^* = [1.0, 1.29]$ in $\mathcal{D}_2^h$ when $N_s = 5$. (b) demonstrates point-wise errors between three types of methods and true values.

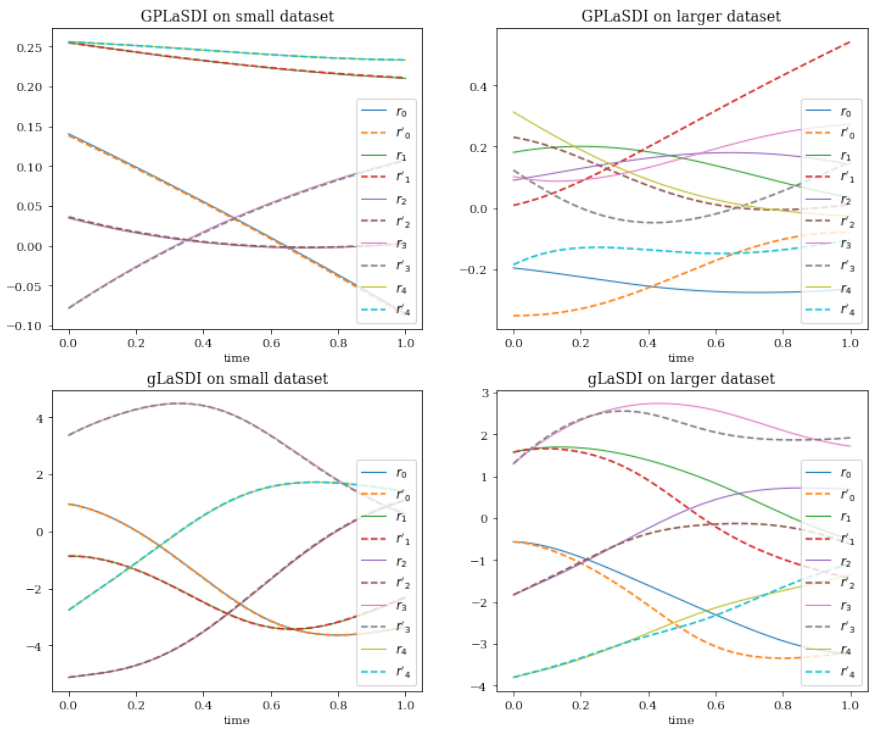
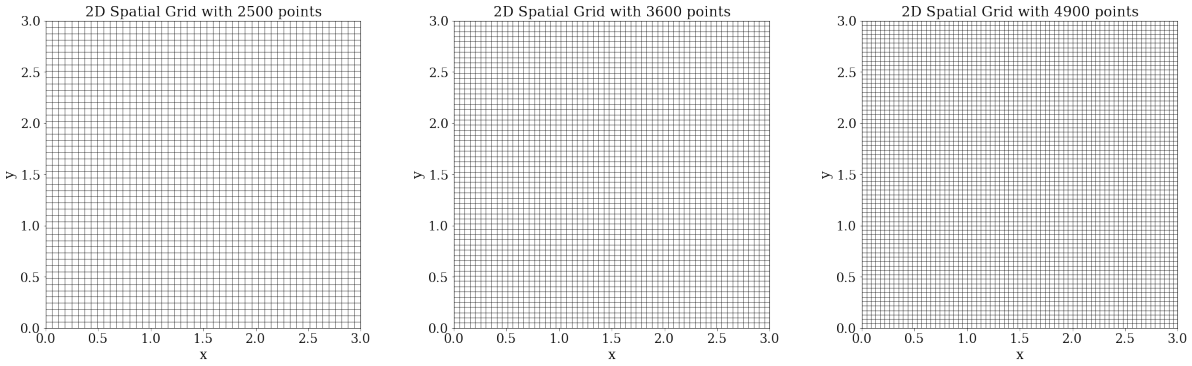


Figure 9: **2D-Burgers**: Latent states trained by GPLaSDI and gLaSDI on dataset $\mathcal{D}_1$ and $\mathcal{D}_2$.



(a) 2500 nodes (b) 3600 nodes (c) 4900 nodes

Figure 10: **2D-burgers**: Three types of square grids with different resolutions.

8.2. Case 2: Lock exchange

Lock-exchange is a classic numerical experiment in fluid mechanics, designed to analyse the mixing and flow behaviours between fluids of varying densities. A 2D rectangular flat-bottomed tank is filled with a source fluid on one side and an ambient fluid on the other, separated by a vertical gate in the middle. The source fluid has a lower temperature and higher density, while the ambient fluid is the opposite. After the gate is removed, the denser fluid collapses under the lighter one. The two fluids gradually mix under gravity until viscosity dominates. The density interface between the source fluid and the ambient fluid forms Kelvin–Helmholtz billows[54], due to instability.

Assuming the temperature on the right side is T_0 and on the left side is $-T_0$, the involved equations include several formulas from fluid mechanics. The incompressible fluid model can be written as:

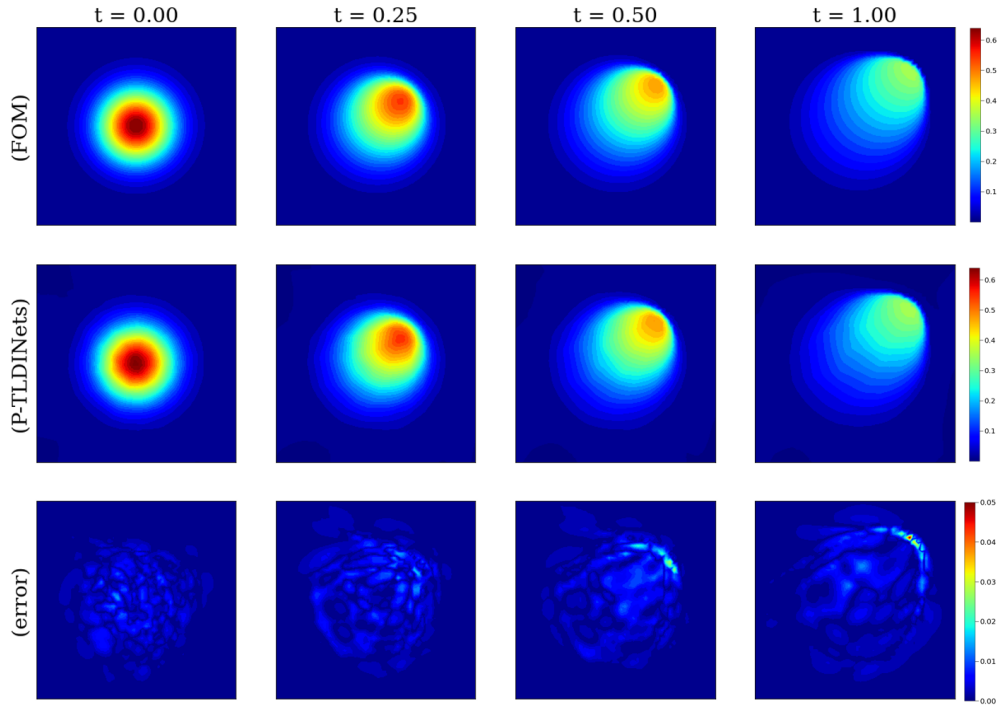


Figure 11: **2D-Burgers**: The high-fidelity velocity solutions, predicted solutions of P-TLDINet and errors for multiple resolutions of the grids at $\boldsymbol{\mu}^* = [0.64, 1.21]$.

$$\begin{cases} \rho\left(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \nu\nabla^2\mathbf{u} + \rho\mathbf{g} \\ \nabla\cdot\mathbf{u} = 0 \\ \dfrac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T = \alpha\nabla^2 T \quad (x_1, x_2, t) \in \Omega \times [0\text{s}, 8\text{s}] \end{cases} \tag{34}$$

where the first equation is the Navier-Stokes equation, the second represents the incompressibility condition, and the third is the energy equation. $\mathbf{u}$ is the velocity vector, and p is the pressure. T is temperature, which is the scalar data observed and learned in this case. $\Omega = [0\,\text{m}, 0.1\,\text{m}] \times [0\,\text{m}, 0.8\,\text{m}]$. The density ρ is related to temperature through the Boussinesq approximation:

$$\rho(T) = \rho_0\left[1 - \beta(T - T_0)\right], \tag{35}$$

where ρ_0 is the reference density at temperature T_0, and β is the thermal expansion coefficient. Here, thermal diffusion is not considered.

The initial condition for velocity $\mathbf{u}$ and temperature T is summarised as follows:

$$\begin{cases} \mathbf{u}(x_1, x_2, 0) = 0 \\ T(x_1, x_2, 0) = \begin{cases} -T_0, & x_1 < 0.4\,\text{m} \\ T_0. & x_1 \geq 0.4\,\text{m} \end{cases} \end{cases} \tag{36}$$

The lock exchange adopts a no-slip boundary condition, meaning that the velocity remains zero at the bottom of the domain. The other relevant physical configurations are shown in Table 6.

The high-fidelity solutions are obtained using an unstructured mesh finite element model, Fluidity[55]. The time interval is set to $\Delta t = 0.005$s. The grid is non-uniform and remains constant over time. Considering that the flow characteristics of the lock exchange may present more dynamic variability in the middle section, a grid refinement is employed between $0.2\text{m} \leq \text{x} \leq 0.6\text{m}$.

In this case, $\boldsymbol{\mu} = \{T_0, \nu\} \in \mathcal{D} = [4.9, 5.1] \times [10^{-6}, 10^{-5}]$ forms the parameter space.

Physical parameters (unit)	Notation	Value
kinematic viscosity coefficient (m^2s^{-1})	ν	10^{-6}
gravitational acceleration (ms^{-2})	g	10
thermal diffusivity (ms^{-2})	α	0
thermal expansion coefficient ($°C^{-1}$)	β	10^{-3}

Table 6: **Lock-exchange**: Configurations of simulation in Fluidity.

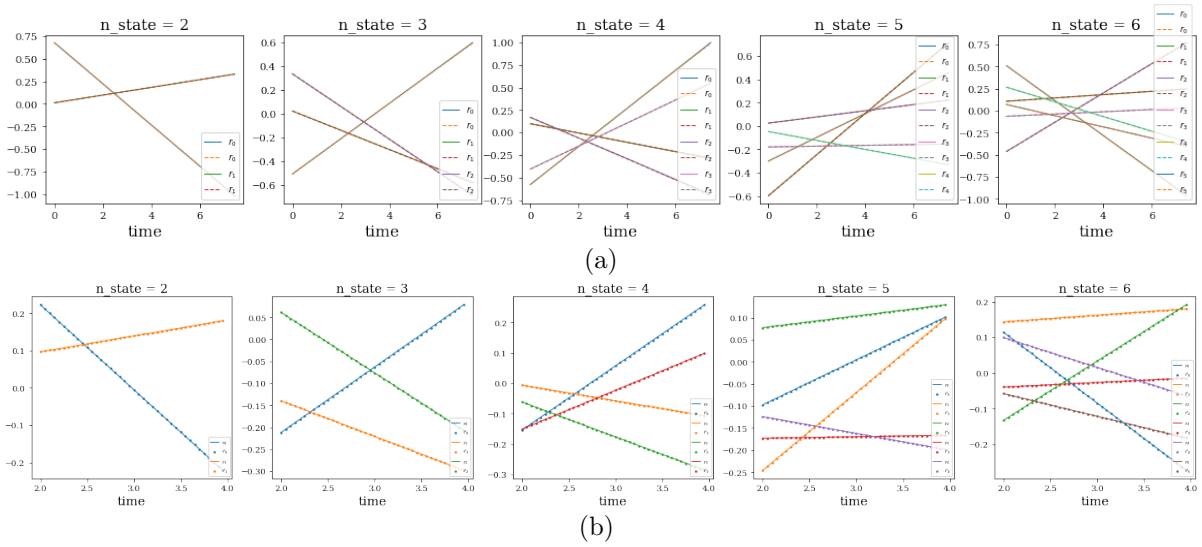


Figure 12: **Lock-exchange**: Low-dimensional dynamics and the solutions of the identified equations at $\boldsymbol{\mu} = [5.0, 10^{-6}]$. (a) and (b) show the dimension of latent space N_s ranging from 2 to 6. (a) represents results for the entire time frame $t \in [0, 0.8]$, while (b) shows only in $t \in [0.2, 0.4]$.

8.2.1. Trained on a single-resolution grid

In this case, P-TLDINet is first trained on a single-resolution grid with 1491 node points. 25 samples are randomly selected as the training parameter set $\mathcal{D}_{train}$, and the discrete test parameter set $\mathcal{D}^h$ is composed of 225 samples. We trained P-TLDINet from $N_s = 2$ to $N_s = 6$, employing the second-order Taylor expansion in Equation 3 for time advancement. The structure of the dynamical network NN_{dyn} is

$$\begin{aligned} NN_{dyn}: &\text{Linear}(\mathrm{N_s}+2, 9) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(9,9) \xrightarrow{\text{Tanh}} \text{Linear}(9,9) \xrightarrow{\text{Tanh}} \\ &\text{ResNetBlock}(9,9) \xrightarrow{\text{Tanh}} \text{Linear}(9, \mathrm{N_s}). \end{aligned} \tag{37}$$

The structure of the reconstruction network NN_{rec} is

$$\begin{aligned} NN_{rec}: &\text{Linear}(\mathrm{N_s}+4, 11) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(11,20) \xrightarrow{\text{Tanh}} \text{Linear}(20,20) \xrightarrow{\text{Tanh}} \\ &\text{ResNetBlock}(20,20) \xrightarrow{\text{Tanh}} \text{Linear}(20,20) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(20,11) \xrightarrow{\text{Tanh}} \\ &\text{Linear}(20,20) \xrightarrow{\text{Tanh}} \text{ResNetBlock}(20,20) \xrightarrow{\text{Tanh}} \text{Linear}(20,20) \xrightarrow{\text{Tanh}} \\ &\text{ResNet}(20,20) \xrightarrow{\text{Tanh}} \text{Linear}(20,11) \xrightarrow{\text{Tanh}} \text{Linear}(11,2). \end{aligned} \tag{38}$$

The initial mapping network NN_{z_0} has the form of:

$$NN_{z_0}: \text{Linear}(2,9) \xrightarrow{\text{Tanh}} \text{Linear}(9,9) \xrightarrow{\text{Tanh}} \text{Linear}(9,9) \xrightarrow{\text{Tanh}} \text{Linear}(9, \mathrm{N_s}). \tag{39}$$

In the library of basis functions $\Theta(\boldsymbol{z})$, only linear and constant terms are taken into account. The optimiser is configured to the Adam optimiser with a learning rate of 0.05, decaying by a factor of 0.6 every 500 iterations and $N_{tol} = 500$. The loss hyperparameters are set as $\omega_{DI} = 0.5$, $\omega_{coef} = 0$ and $\omega_{z_0} = 0.5$.

N_s	2	3	4	5	6
L_2 error (%)	0.8289	0.6977	1.310	0.8827	0.8368

Table 7: **Lock-exchange**: Total L_2 errors on testing samples $\mathcal{D}^h \subset \mathcal{D}_{trian}$ of different N_s.

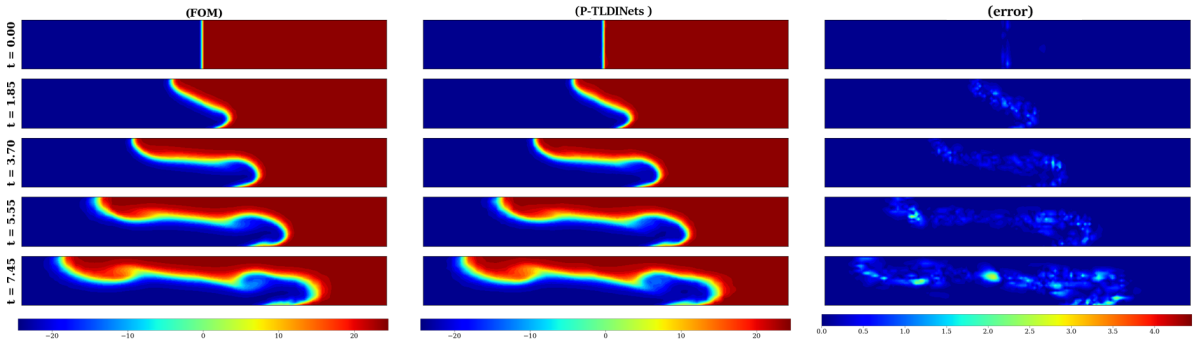


Figure 13: **Lock-exchange**: High-fidelity temperature solutions, reconstruction results and errors of P-TLDINet at $\boldsymbol{\mu}^* = [4.929, 8.07^{-6}]$ vary with t.

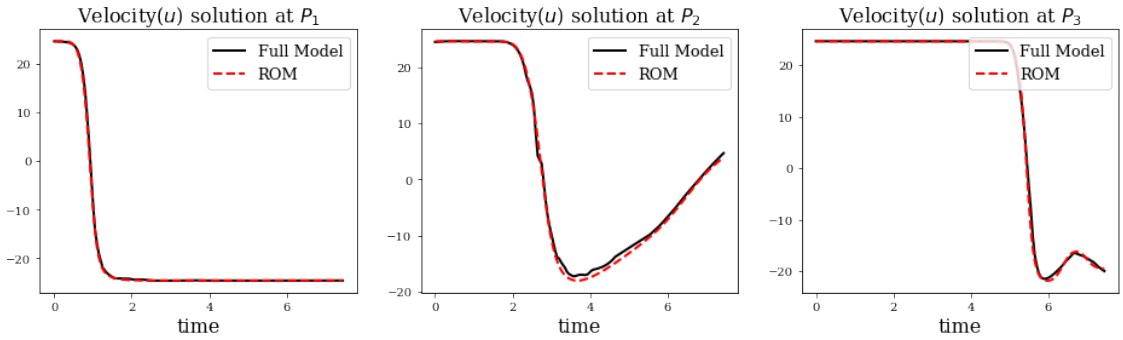


Figure 14: **Lock-exchange**: The flow fields reconstructed by P-TLDINet at $\boldsymbol{\mu}^* = [4.929, 8.07^{-6}]$ for P_1, P_2 and P_3.

Figure 12 and Table 7 illustrate the dimensionality reduction performances across various low-dimensional spaces. It is clear that P-TLDINet still shows greater smoothness, even under more intricate dynamical systems in this case. This feature enhances the effectiveness of subsequent equation identification and facilitates predictions with a 176.5× speed-up.

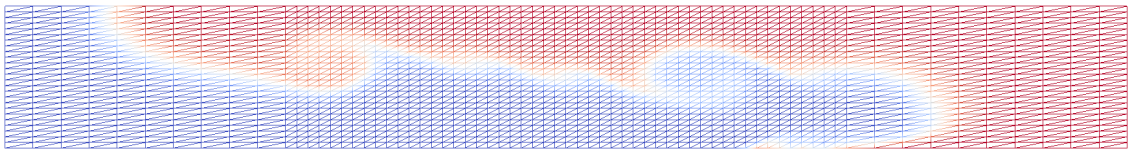
Figure 13 shows the prediction results for $\boldsymbol{\mu}^* = [4.929, 8.07^{-6}]$ in D^h. More detailed results are shown in Figure 14, showing the flow field reconstructions at three single points, P_1, P_2 and P_3, with the same $\boldsymbol{\mu}^*$. It is apparent that P-TLDINet exhibits excellent reconstruction capability.

8.2.2. Trained on different resolutions of the meshes

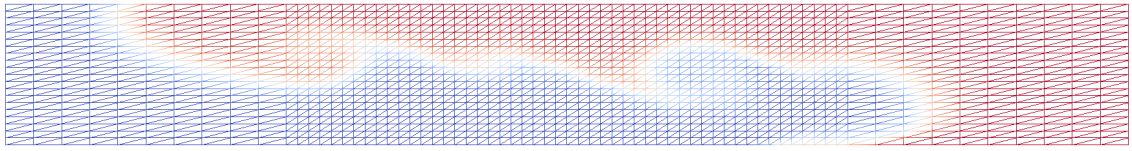
In the lock-exchange scenario, grids of different resolutions are also tested. These grids obey the sparsity characteristics described previously and are depicted in Figure 15. These three grids have 1894, 1491, and 1278 spatial points, respectively. D_{train} is randomly divided into three parts, and each part corresponds to a different grid size. The high-dimensional data $\{\boldsymbol{u}\}$ are generated by Fluidity on these three grids for network training as input. Finally, predictions are made on the grid (c) in Figure 15. The prediction results achieve 97.00% recovery accuracy. Figure 16 presents the specific prediction results of P-TLDINet at $\boldsymbol{\mu}^* = [5.00, 4.214 \times 10^{-6}]$.

9. Conclusion

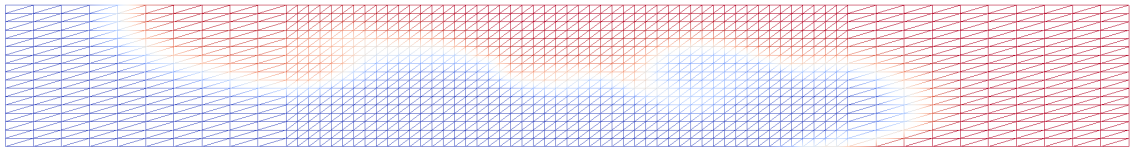
This paper introduces a Parametric Taylor series-based Latent Dynamics Identification Network (P-TLDINet), a novel dimensionality reduction neural network designed for robust, accurate, and computation-



(a) 1864 nodes



(b) 1491 nodes



(c) 1278 nodes

Figure 15: **Lock-exchange**: Three types of unstructured grids with different resolutions.

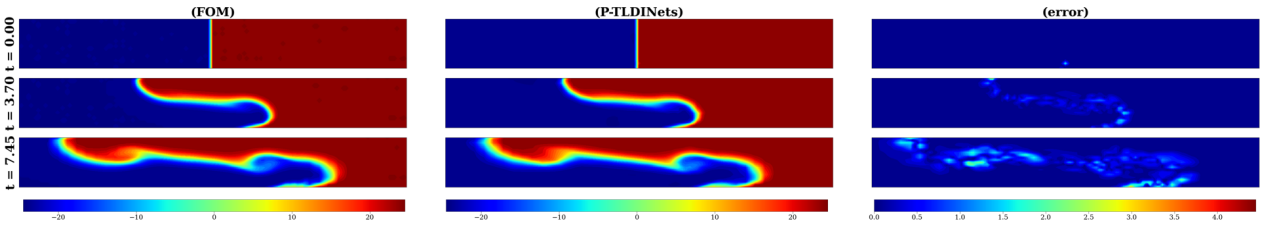


Figure 16: **Lock-exchange**: The high-fidelity temperature solutions, predicted solutions of P-TLDINet and errors for different resolutions of the grids at $\boldsymbol{\mu}^* = [5.00, 4.214 \times 10^{-6}]$.

ally efficient data-driven modelling of parametric dynamical systems. The proposed model integrates with Taylor series-based Latent Dynamic Networks (TLDNet) and Identification of Dynamics (ID models), which are also the core strengths of our work. Three sub-networks and the coefficients of ID models are trained simultaneously and interactively, facilitating a simple and smoother discovery of latent dynamics. In order to enhance the exploration of parameter spaces, a state-of-the-art k nearest neighbour (KNN) method based on inverse distance weighting (IDW) is used to predict approximate scalar fields of unknown parameter samples, incorporating spatial information.

To demonstrate the utility of P-TLDINet, it has been applied to dynamical problems with nonlinear characteristics, including 2D Burgers' equations and the lock-exchange case. Numerical results show that, compared to autoencoder-based latent identification methods (GPLaSDI and gLaSDI), P-TLDINets have a lighter structure and superior performance, tending to reflect better learning capabilities in more complex parameter spaces with less training time. In addition, in contrast to standard autoencoder-based methods that are typically restricted to a fixed grid resolution due to their reliance on a fixed input dimension, P-TLDINet can handle multiple mesh resolutions within a single trained network. This is achieved through the coordinate-based input of NN_{rec}, which decouples the model from any specific discretization. The L_2 errors of the P-TLDINets remain consistently below 2%, achieving a significant computational speedup. P-TLDINet supports training on multiple mesh resolutions within a single trained network, which is a feature that arises naturally from the coordinate-based design of NN_{rec}, which does not depend on any fixed grid structure. By contrast, standard autoencoder-based approaches typically rely on a fixed input dimension, making them inherently limited to the resolution on which they were trained. The L_2 errors of P-TLDINet remain consistently below 2%, while achieving a significant computational speedup.

In comparison to LDNets, which encodes all information within a black-box neural network, P-TLDINet introduces a fundamentally different strategy by extracting an explicit, closed-form SINDy model from the latent dynamics. This key feature not only provides interpretability but also opens up new possibilities

for parameterisation learning in the low-dimensional space. For instance, in this work, we adopt a KNN interpolation scheme in the SINDy coefficient space to achieve robust generalisation across unseen parameters. Beyond that, the proposed parametric framework is inherently flexible and can accommodate other sampling or interpolation strategies, such as greedy sampling or radial basis function interpolation, depending on the specific requirements of the problem. When the initial condition varies with parameters, LDNets lead to less accurate reconstruction near the initial time, as the entire initial field must be implicitly inferred by NN_{rec}. To address this issue, we introduce a dedicated initial state network NN_{z_0} that explicitly predicts $z(0)$ from the input parameters, effectively compensating for this deficiency. Additionally, we employ a second-order Taylor-series scheme for time advancement, which offers a smaller local truncation error compared to the first-order forward Euler scheme used in LDNets, theoretically reducing the accumulation of errors over long-time predictions.

P-TLDINet provides a transformation from leading-edge modelling reductions (such as methods based on POD and autoencoders). In detail, instead of explicitly constructing an autoencoder based on high-dimensional discrete coordinates, P-TLDINet automatically generates a latent representation of the system's states and implements a direct query for individual spatial coordinates. Additionally, it addresses the limitations of LDNets in handling problems with varying initial conditions, thereby broadening its generalisation performance. The ID model improves interpretability in low-dimensional spaces, and the concept of locally identifiable predictions makes P-TLDINets more competitive.

The presented parametric framework is generalisable, allowing for the integration of other sampling (such as greedy sampling) and interpolation methods. P-TLDINet can not only accommodate input with varying grids during the training procedure but also enhance spatial flexibility in the online reconstruction process. This enables P-TLDINet to handle different resolutions of the meshes relatively easily, paving the way for future applications in adaptive meshing and complex problems.

While the current P-TLDINet framework has demonstrated strong performance on the test cases considered in this work, its applicability to more complex scenarios, such as stiff dynamics or noisy data, remains to be further explored. These more challenging cases introduce new difficulties for both the stability of time integration and the accumulation of errors over long-time predictions. To address these challenges in future work, perturbation and denoising strategies could be incorporated into the P-TLDINet framework. These techniques have the potential to enhance the robustness of the model against numerical instabilities and improve its long-term predictive capability, thereby extending the applicability of P-TLDINet to a wider range of complex dynamical systems.

Acknowledgments

The authors acknowledge the support of the Fundamental Research Funds for the Central Universities, the Top Discipline Plan of Shanghai Universities-Class I, NSFC (Grant No.12671529), Shanghai Gaofeng Project for University Academic Program Development, National Key *R&D* Program of China(NO. 2022YFE0208000, 2024YFC2816400, and 2024YFC2816401). This work is also supported in part by grants from the Shanghai Engineering Research Center (No. 19DZ2255100) and the Shanghai Institute of Intelligent Science and Technology, Tongji University.

References

[1] Federico Negri, Andrea Manzoni, and David Amsallem. Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *Journal of Computational Physics*, 303:431–454, 2015. doi:10.1016/j.jcp.2015.09.046.

[2] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015. doi:10.1137/130932715.

[3] Zhenghai Wang, Dunhui Xiao, Fangxin Fang, Rajesh Govindan, Christopher C Pain, and Yike Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018. doi:10.1002/fld.4416.

[4] Hanyu Zhou, Haochen Li, and Yaomin Zhao. Identification of partial differential equations from noisy data with integrated knowledge discovery and embedding using evolutionary neural networks. *Theoretical and Applied Mechanics Letters*, 14(2):100511, 2024. doi:10.1016/j.taml.2024.100511.

[5] Wenbo Cao and Weiwei Zhang. Machine learning of partial differential equations from noise data. *Theoretical and Applied Mechanics Letters*, 13(6):100480, 2023. doi:10.1016/j.taml.2023.100480.

[6] Christopher J Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205(1):131–156, 2005. doi:10.1016/j.jcp.2004.10.036.

[7] Richard H. Gallagher. *Finite Element Analysis: Fundamentals*. Prentice-Hall, Englewood Cliffs, NJ, 1975. URL: https://opac.um.edu.uy/index.php?lvl=notice_display&id=19023.

[8] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. In *Handbook of Numerical Analysis*, volume 7, pages 713–1018. Elsevier, 2000. doi:10.1016/s1570-8659(00)07005-8.

[9] Suraj Pawar, Shady E Ahmed, Omer San, and Adil Rasheed. Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, 32(3), 2020. doi:10.1063/5.0002051.

[10] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909:A9, 2021. doi:10.1017/jfm.2020.948.

[11] Rui Fu, Dunhui Xiao, Ionel Michael Navon, Fangxin Fang, Liang Yang, C Wang, and S Cheng. A nonlinear non-intrusive reduced order model of fluid flow by auto-encoder and self-attention deep learning methods. *International Journal for Numerical Methods in Engineering*, 124(13):3087–3111, 2023. doi:10.1002/nme.7240.

[12] Kai Fukami and Kunihiko Taira. Grasping extreme aerodynamics on a low-dimensional manifold. *Nature Communications*, 14(1):6480, 2023. doi:10.1038/s41467-023-42213-6.

[13] S Kounouho, R Dingreville, and J Guilleminot. Stochastic symplectic reduced-order modeling for model-form uncertainty quantification in molecular dynamics simulations in various statistical ensembles. *Computer Methods in Applied Mechanics and Engineering*, 431:117323, 2024. doi:10.1016/j.cma.2024.117323.

[14] Ivan Cimrák, Markus Gusenbauer, and Thomas Schrefl. Modelling and simulation of processes in microfluidic devices for biomedical applications. *Computers & Mathematics with Applications*, 64(3):278–288, 2012. doi:10.1016/j.camwa.2012.01.062.

[15] Jinfeng Li, Yuanhang Zhang, Xuesong Wang, and Yuh-Shan Ho. Bibliometric analysis of atmospheric simulation trends in meteorology and atmospheric science journals. *Croatica Chemica Acta*, 82(3):695–705, 2009. URL: https://hrcak.srce.hr/en/45477.

[16] Jinlong Fu, Dunhui Xiao, Rui Fu, Chenfeng Li, Chuanhua Zhu, Rossella Arcucci, and Ionel M Navon. Physics-data combined machine learning for parametric reduced-order modelling of nonlinear dynamical systems in small-data regimes. *Computer Methods in Applied Mechanics and Engineering*, 404:115771, 2023. doi:10.1016/j.cma.2022.115771.

[17] Xinyu Pan and Dunhui Xiao. Domain decomposition for physics-data combined neural network based parametric reduced order modelling. *Journal of Computational Physics*, 519:113452, 2024. doi:10.1016/j.jcp.2024.113452.

[18] Victor Matray, Faisal Amlani, Frédéric Feyel, and David Néron. A hybrid numerical methodology coupling reduced order modeling and graph neural networks for non-parametric geometries: Applications to structural dynamics problems. *Computer Methods in Applied Mechanics and Engineering*, 430:117243, 2024. doi:10.1016/j.cma.2024.117243.

[19] Utku Eren, Anna Prach, Başaran Bahadır Koçer, Saša V Raković, Erdal Kayacan, and Behçet Açıkmeşe. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017. doi:10.2514/1.g002507.

[20] Indranil Nayak, Mrinal Kumar, and Fernando L Teixeira. Detection and prediction of equilibrium states in kinetic plasma simulations via mode tracking using reduced-order dynamic mode decomposition. *Journal of Computational Physics*, 447:110671, 2021. doi:10.1016/j.jcp.2021.110671.

[21] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993. doi:10.1146/annurev.fl.25.010193.002543.

[22] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016. doi:10.1137/15m1013857.

[23] Yvon Maday and Einar M Rønquist. A reduced-basis element method. *Journal of scientific computing*, 17:447–459, 2002. doi:10.1023/a:1015197908587.

[24] Federico Pichi, Beatriz Moya, and Jan S Hesthaven. A graph convolutional autoencoder approach to model order reduction for parametrized pdes. *Journal of Computational Physics*, 501:112762, 2024. doi:10.1016/j.jcp.2024.112762.

[25] Kazuo Yonekura and Katsuyuki Suzuki. Data-driven design exploration method using conditional variational autoencoder for airfoil design. *Structural and Multidisciplinary Optimization*, 64(2):613–624, 2021. doi:10.1007/s00158-021-02851-0.

[26] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020. doi:10.1016/j.jcp.2019.108973.

[27] Kai Fukami, Taichi Nakamura, and Koji Fukagata. Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Physics of Fluids*, 32(9), 2020. doi:10.1063/5.0020721.

[28] Alberto Solera-Rico, Carlos Sanmiguel Vila, Miguel Gómez-López, Yuning Wang, Abdulrahman Almashjary, Scott TM Dawson, and Ricardo Vinuesa. β-variational autoencoders and transformers for reduced-order modelling of fluid flows. *Nature Communications*, 15(1):1361, 2024. doi:10.1038/s41467-024-45578-4.

[29] Alan A Kaptanoglu, Kyle D Morgan, Chris J Hansen, and Steven L Brunton. Physics-constrained, low-dimensional models for magnetohydrodynamics: First-principles and data-driven approaches. *Physical Review E*, 104(1):015206, 2021. doi:10.1103/physreve.104.015206.

[30] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021. doi:10.1038/s42256-021-00302-5.

[31] Wei Cui, Lin Zhao, and Yaojun Ge. Wind-induced buffeting vibration of long-span bridge considering geometric and aerodynamic nonlinearity based on reduced-order modeling. *Journal of Structural Engineering*, 149(11):04023160, 2023. doi:10.1061/jsendh.steng-11543.

[32] Dunhui Xiao, Fangxin Fang, Christopher Pain, and Guangwei Hu. Non-intrusive reduced-order modelling of the Navier–Stokes equations based on RBF interpolation. *International Journal for Numerical Methods in Fluids*, 79(11):580–595, 2015. doi:10.1002/fld.4066.

[33] J.S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018. doi:10.1016/j.jcp.2018.02.037.

[34] Yu-Eop Kang, Soonho Shon, and Kwanjung Yee. Local non-intrusive reduced order modeling based on soft clustering and classification algorithm. *International Journal for Numerical Methods in Engineering*, 123(10):2237–2261, 2022. doi:10.1002/nme.6934.

[35] Jiaming Guo and Dunhui Xiao. Nonlinear model reduction by probabilistic manifold decomposition. *SIAM Journal on Scientific Computing*, 48(1):A209–A235, 2026. doi:10.1137/25M1738863.

[36] Francesco Regazzoni, Stefano Pagani, Matteo Salvador, Luca Dede', and Alfio Quarteroni. Learning the intrinsic dynamics of spatio-temporal processes through latent dynamics networks. *Nature Communications*, 15(1):1834, 2024. doi:10.1038/s41467-024-45323-x.

[37] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007. doi:10.1073/pnas.0609476104.

[38] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009. doi:10.1126/science.1165893.

[39] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 113(15):3932–3937, 2016. doi:10.1073/pnas.1517384113.

[40] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019. doi:10.1073/pnas.1906995116.

[41] Xiaolong He, Youngsoo Choi, William D Fries, Jonathan L Belof, and Jiun-Shyan Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 489:112267, 2023. doi:10.1016/j.jcp.2023.112267.

[42] William D Fries, Xiaolong He, and Youngsoo Choi. Lasdi: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 399:115436, 2022. doi:10.1016/j.cma.2022.115436.

[43] Christophe Bonneville, Youngsoo Choi, Debojyoti Ghosh, and Jonathan L Belof. Gplasdi: Gaussian process-based interpretable latent space dynamics identification through deep autoencoder. *Computer Methods in Applied Mechanics and Engineering*, 418:116535, 2024. doi:10.1016/j.cma.2023.116535.

[44] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017. doi:10.1126/sciadv.1602614.

[45] Mourad Oulghelou, Amine Ammar, and Rama Ayoub. Greedy identification of latent dynamics from parametric flow data. *Computer Methods in Applied Mechanics and Engineering*, 432:117332, 2024. doi:10.1016/j.cma.2024.117332.

[46] Xinlei Lin, Dunhui Xiao, Min Luo, Xuejun Xu, Shuyu Sun, Lijian Jiang, and Haibao Wen. Data discovery of lower dimensional equations of turbulent flows. *International Journal for Numerical Methods in Engineering*, 126(23):e70198, 2025. doi:10.1002/nme.70198.

[47] Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Taylor-lagrange neural ordinary differential equations: Toward fast training and evaluation of neural odes. *arXiv preprint arXiv:2201.05715*, 2022. doi:10.48550/arXiv.2201.05715.

[48] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020. URL: https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

[49] Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021. doi:10.1017/s0962492921000052.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. doi:10.1109/cvpr.2016.90.

[51] Jean-Christophe Loiseau, Bernd R Noack, and Steven L Brunton. Sparse reduced-order modelling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018. doi:10.1017/jfm.2018.147.

[52] Oliver Kramer. K-nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*, pages 13–23. Springer, 2013. doi:10.1007/978-3-642-38652-7_2.

[53] Shady E Ahmed, Omer San, Adil Rasheed, Traian Iliescu, and Alessandro Veneziani. Physics guided machine learning for variational multiscale reduced order modeling. *SIAM Journal on Scientific Computing*, 45(3):B283–B313, 2023. doi:10.1137/22m1496360.

[54] IPD De Silva, HJS Fernando, F Eaton, and D Hebert. Evolution of kelvin-helmholtz billows in nature and laboratory. *Earth and Planetary Science Letters*, 143(1-4):217–231, 1996. doi:10.1016/0012-821x(96)00129-x.

[55] Fluidity: Open-source computational fluid dynamics. URL: https://fluidityproject.github.io/.