Error estimation of the parametric non-intrusive reduced order model using machine learning

D. Xiao[a,*]

[a]*ZCCE, College of Engineering, Swansea University, Bay Campus, Fabian Way, Swansea SA1 8EN, UK*

Abstract

A novel error estimation method for the parametric non-intrusive reduced order model (P-NIROM) based on machine learning is presented. This method relies on constructing a set of response functions for the errors between the high fidelity full model solutions and P-NIROM using machine learning method, particularly, Gaussian process regression method. This yields closer solutions agreement with the high fidelity full model. The novelty of this work is that it is the first time to use machine learning method to derive error estimate for the P-NIROM. The capability of the new error estimation method is demonstrated using three numerical simulation examples: flow past a cylinder, dam break and 3D fluvial channel. It is shown that the results are closer to those of the high fidelity full model when considering error terms. In addition, the interface between two phases of dam break case is captured well if the error estimator is involved in the P-NIROM.

Keywords: NIROM; machine learning; Gaussian process regression; error estimation

1. Introduction

Over the past decades, the reduced order modelling method has proven to be a powerful tool of reducing the dimension of large dynamic systems. Among model reduction techniques, the proper orthogonal decomposition (POD) is the most widely used method. POD is capable of representing large systems using a few number of optimal basis functions and it has been applied successfully to various research and engineering fields such as modal analysis [1], air pollution [2], mesh optimization [3], shape optimization problems [4], varying computational domain [5], cardiac electrophysiology and mechanics [6], porous media [7], shallow water [8, 9, 10], aerospace design [11] and neutron/photon transport problems [12].

In recent years, non-intrusive reduced order modelling (NIROM) methods become popular due to the independence of the source code and physical system [13]. Therefore, the NIROM is easy to modify, maintain and extend to other complicated ap-

[*]Corresponding author
Email address: dunhui.xiao@swansea.ac.uk (D. Xiao)

plications. The NIROM also avoids the issues of instability and non-linear inefficiency of the intrusive reduced order model (IROM) [14, 15]. The methods of improving stability and non-linear efficiency for IROMs can be found in the work of [16, 17, 18, 19, 20, 21, 22, 23]. Over the past few years, a number of NIROMs have been presented. Audouze *et al.* presented a NIROM for nonlinear parametric time dependent Partial Differential Equations (PDEs) using POD and Radial Basis Function (RBF) approximation [24]. Xiao *et al.* presented a number of NIROMs based on interpolations and deep learning methods [14, 25, 26]. Guo and Hesthaven presented a non-intrusive data-driven reduced order modelling method [27, 28]. The NIROMs have been successfully applied into various problems like fluid structure interactions [29], non-linear problems [30], car crash simulation [31] and compressor blades design [32].

However, in some operational models, we hope our fast reduced order models are as accurate as the high fidelity models. In uncertainty quantification or sensitive analysis, it requires a number of simulations of the high fidelity full model at different parameter settings to construct the reduced order model (ROM). After constructing the ROM, we wish to know how accurate the ROM performs. There are a number of error estimation methods being presented for ROMs. Homescu *et al.* presented an approach based on statistical condition estimation and adjoint method [33]. This method defines ranges of perturbations in the high fidelity full model over which the ROM is still appropriate. Chaudhry *et al.* used the adjoint method to do posteriori error estimation. In this method, a hierarchical ROM is used for the adjoint computation to quantify the error of the finite element (FEM) and ROM solutions [34]. Chaturantabut *et al.* derived an error bounds on the state approximations for the POD and Discrete Empirical Interpolation Method (DEIM) based ROM solutions [35]. Wirtz *et al.* presented a local Lipschitz constant estimation method and a A-Posteriori error estimator for ROMs and DEIM based ROMs respectively [36, 37]. Moosavi *et al.*[38] introduced multivariate input-output models to predict the errors of local parametric Proper Orthogonal Decomposition reduced-order models and used Gaussian processes and artificial neural networks to approximate them. Stefanescu *et al.* used these multivariate input-output models to generate decompositions of one dimensional parametric domains [39]. Alexandrov *et al.* presented a multi-fidelity correction method, which makes the solutions of low fidelity models converge globally to the original high fidelity models [40]. Drohmann *et al.* presented a statistical ROMES method for modeling errors introduced by ROMs [41].

In this work, a set of error functions was constructed to represent the remaining (residual) errors between the high fidelity full model and P-NIROM using a machine learning method. The machine learning method has been shown a great success in a number of areas. In this work, the Gaussian Process Regression (GPR) machine learning method is used. The GPR captures the uncertainty in the training data directly, and fits the data accurately when training data is not big [42]. In addition, the GPR is a probabilistic and non-parametric model, which means it can provide a reliable response [43]. Firstly, we run the fidelity full model and the P-NIROM separately for different simulations with different parameters. The P-NIROM is constructed using the method described in the section 3. After obtaining the solutions of the P-NIROM, the errors between the P-NIROM and the high fidelity full model can be calculated. The errors (residual values between snapshots of the high fidelity full model and P-NIROM) are used as the training data of constructing the error functions. After constructing the error response functions using the GPR method, then we can use them to predict the errors for any given new simulations with different parameters.

The structure of the paper is as follows: section 2 presents the general parametric reduced order Partial Differential Equations (PDEs); section 3 briefly describes a general parametric non-intrusive reduced order model (P-NIROM); section 4 briefly describes a GPR machine learning method; section 5 derives an error estimator for P-NIROM using a Gaussian process regression method; section 6 demonstrates the capability of this method using three numerical examples: flow past a cylinder, dam break cases and 3D fluvial channel; Finally in section 7, summary and conclusions are presented.

2. General parametric reduced order PDEs

In this section, a general parametric space-time linear and nonlinear partial differential equation (PDE) is given, and the parametric ROM is derived. The parametric space-time linear and nonlinear PDE has a general form of:

$$\mathcal{F}(\boldsymbol{u}(\mathbf{x},t,\boldsymbol{\beta}),\mathbf{x},t,\boldsymbol{\beta}) = s(\mathbf{x},t,\boldsymbol{\beta}), \tag{1}$$

where $\boldsymbol{u} \in \mathcal{R}^{D\times N}$ denotes a state variable vector including, for example, velocity components, pressure, density and etc. D denotes the number of scalars and N the number of nodes in the computational domain. $\mathbf{x}$ denotes a spatial coordinate system. t is the time and s is a source term. $\boldsymbol{\beta} \in \mathcal{R}^{Q}$ is a parameter vector with a dimensional size of Q.

In reduced order modelling, the state variable $\boldsymbol{u}$ can be described as an expansion of POD basis functions $\Phi(\mathbf{x},\boldsymbol{\beta}) = (\Phi_1,\ldots,\Phi_m,\ldots,\Phi_M)$ ($m \in (1,\ldots,M)$, M is the number of basis functions and $M << N$):

$$\boldsymbol{u}(\mathbf{x},t,\boldsymbol{\beta}) = \Phi\boldsymbol{u}^r, \tag{2}$$

where $\boldsymbol{u}^r(t,\boldsymbol{\beta}) \in \mathcal{R}^M$ denotes a reduced state variable vector (the superscript r indicates an operator or variable associated with the ROM). By using a proper orthogonal decomposition (POD) method, the basis functions Φ of the variable are derived optimally from the snapshots sampled at time instants $\{t_1,\ldots,t_i,\ldots,t_{N_t}\}$:

$$\Phi_m(\mathbf{x},\boldsymbol{\beta}) = \sum_{i=1}^{N_t}\boldsymbol{u}(\mathbf{x},t_i,\boldsymbol{\beta})\gamma_{m,i}, \quad m \in (1,\cdots,M), \tag{3}$$

subject to

$$\sum_{m=1}^{M} | <\Phi_m,\Phi_m>_{L^2} |^2 = 1, \tag{4}$$

where $<\cdot,\cdot>_{L^2}$ denotes the canonical inner product in L^2 norm, and $\gamma_{m,i}$ is obtained via singular value decomposition (SVD):

$$\mathcal{B}\gamma_m = \lambda_m\gamma_m, \tag{5}$$

where $\gamma_m = (\gamma_{m,1}, \ldots, \gamma_{m,i}, \ldots, \gamma_{m,N_t})$ and the matrix $\mathcal{B}$ has a form of,

$$\mathcal{B}_{k,n} = \frac{1}{N_t} \int_{\Omega} \boldsymbol{u}(\cdot, t_k, \cdot) \boldsymbol{u}(\cdot, t_n, \cdot)^* dx, \quad k, n \in (1, \ldots, N_t), \tag{6}$$

where $*$ denotes the transpose and Ω is a space that includes the functions $\boldsymbol{u}(\cdot, t_k, \cdot)$ and $\boldsymbol{u}(\cdot, t_n, \cdot)$.

The singular values $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m, \ldots, \lambda_M)$ are listed in a decreasing order. Projecting Equation (1) onto a reduced space, yields,

$$\Phi^T \mathcal{F}(\Phi \boldsymbol{u}^r(\boldsymbol{\beta}, \mathbf{x}, t), \mathbf{x}, t, \boldsymbol{\beta}) = \Phi^T s(\mathbf{x}, t, \boldsymbol{\beta}). \tag{7}$$

The parametric ROM in (7) can be re-formulated as:

$$\mathcal{F}^r(\boldsymbol{u}^r(\boldsymbol{\beta}, \mathbf{x}, t), \mathbf{x}, t, \boldsymbol{\beta}) = s^r(\mathbf{x}, t, \boldsymbol{\beta}). \tag{8}$$

3. Parametric non-intrusive reduced order modelling (P-NIROM) method

This section describes a general parametric non-intrusive reduced order model (P-NIROM) method for a general parametric space-time linear and nonlinear PDE. Unlike the traditional parametric ROM, as described in Equation (8), the work of [26] presented a P-NIROM, which is independent of the source code of the original physical full system. The P-NIROM can predict solutions of the simulation given unseen or different parameters (for example, initial conditions or boundary conditions). The P-NIROM is constructed based on a number of simulations and each simulation has a different set of parameters *i.e.* initial conditions or boundary conditions. The key idea of P-NIROM is to construct a set of response functions for the parameter space and a state space separately. The parameters and state space can be approximated using either Smolyak spare grid interpolation method or Radial Basis Function (RBF) interpolation method.

The P-NIROM includes offline and online procedures. The offline procedure has two main procedures: (1) constructing a set of interpolation functions for the parametric space; (2) constructing a set of response interpolation functions for the state space (we will use hyper-surfaces in the following paragraphs to represent this). In the first procedure, a Smolyak sparse grid is used to represent the distribution of the parameters (can be any varying variables for the model). For each node at the Smolyak sparse grid, there is a simulation that is run using a parameter set. A set of basis functions can be obtained from this simulation. In the second procedure, a set of hyper-surfaces are generated to represent the underlying reduced state system using a radial basis function interpolation method. The offline process of constructing the P-NIROM can be summarised in Algorithm (1),

Algorithm 1: Offline procedure of constructing the P-NIROM

```
(i) Determining the parameter space and choose a number of parameter sets using
    Smolyak sparse grid;
(ii) Running the high fidelity full model for each parameter set, and each parameter
    set is a simulation;
(iii) Calculating a set of basis functions and POD coefficients for each set of
    snapshots generated by simulations using SVD;
(iv) Constructing a set of interpolation functions representing the parameter space
    using RBF or Smolyak sparse grid interpolation method based on the snapshots
    and basis functions;
(v) Constructing a set of hyper-surfaces representing the state using RBF
    interpolation method;
```

In Algorithm (1) step (i), the parameter space includes different varying parameters, for example, different initial or boundary conditions. One parameter set consists of a combination of those different varying parameters in the parameter space for each simulation. The number of parameter sets is equal to the number of Smolyak sparse grid.

In step (iv), a set of interpolation functions representing the parameter space (P_j) is constructed for POD coefficients and POD basis functions. The interpolation function can be constructed by the Radial Basis Function (RBF) interpolant. Given a set of G distinct data points $(\mathbf{k}_{j,i})_{i=1}^{G}$, and corresponding data values $\boldsymbol{\zeta}_{j,i}$, the RBF interpolant $P_j(\mathbf{k}_j)$ for the j^{th} POD coefficient is given by

$$P_j(\mathbf{k}_j) = \sum_{i=1}^{G} \varpi_{j,i}\, \phi_{j,i}(||\mathbf{k}_j - \boldsymbol{c}_{j,i}||), \quad j \in \{1, 2, \cdots, B\} \tag{9}$$

where $\mathbf{k}_j$ is an independent variable (a parameter set vector) including varying parameters such as varying initial conditions or boundary conditions $\mathbf{k}_j = \left\{k_{j,1}, k_{j,2}, \cdots, k_{j,\mathcal{G}}\right\}$. $\mathcal{G}$ is the number of varying parameters (dimension of the parameter space). $\boldsymbol{c}_j$ is a vector containing the centre of the j^{th} RBF function. In this work, the centres are chosen such that $\mathbf{c}_{j,i} = \mathbf{k}_{j,i}$. G is the chosen number of Smolyak sparse grid and it is the number of scatter points for RBF interpolation. B is the number of POD coefficients or basis functions and $\phi_{j,i}$ is the i^{th} point's Radial Basis Function (RBF) for the j^{th} POD coefficient. The RBF can be Gaussian, Linear Spline, Multi-Quadratic and etc. In this work, The RBF is chosen to be a Gaussian function. If we need to use the interpolant $P_j(\mathbf{k}_j)$ to interpolate a new unseen point $\mathbf{k}_j$, we need to solve the right hand side of Equation (9). This means the expansion coefficients (weights of RBFs) $\varpi_{j,i}$ should be determined. They can be determined from the interpolation conditions $P_j(\mathbf{k}_{j,i}) = \zeta_{j,i}$, $i \in \{1, 2, \ldots, G\}$, which leads to solving the following linear system:

$$A_j \boldsymbol{\varpi}_j = \boldsymbol{\zeta}_j, \tag{10}$$

where

$$A_j = \begin{bmatrix} \phi_j(\|\mathbf{k}_1 - \mathbf{k}_1\|_2) & \phi_j(\|\mathbf{k}_1 - \mathbf{k}_2\|_2) & \cdots \phi_j(\|\mathbf{k}_1 - \mathbf{k}_G\|_2) \\ \phi_j(\|\mathbf{k}_2 - \mathbf{k}_1\|_2) & \phi_j(\|\mathbf{k}_2 - \mathbf{k}_2\|_2) & \cdots \phi_j(\|\mathbf{k}_2 - \mathbf{k}_G\|_2) \\ \vdots & \vdots & \vdots \\ \phi_j(\|\mathbf{k}_G - \mathbf{k}_1\|_2) & \phi_j(\|\mathbf{k}_G - \mathbf{k}_2\|_2) & \cdots \phi_j(\|\mathbf{k}_G - \mathbf{k}_G\|_2) \end{bmatrix}, \tag{11}$$

$$\boldsymbol{\varpi}_j = [\varpi_{j,1}, \cdots, \varpi_{j,G}]^T, \boldsymbol{\zeta}_j = [\zeta_{j,1}, \cdots, \zeta_{j,G}]^T. \tag{12}$$

The matrix A_j for the j^{th} POD coefficient is a distance matrix. It is known that the distance matrix A_j based on the Euclidean distance $\|\cdot\|_2$ is always non-singular, and therefore, the interpolation problem is well-posed [44]. $\boldsymbol{\zeta}_{j,G}$ is the target functional values corresponding to the j^{th} POD coefficients for the simulation set G. For example, $\boldsymbol{\zeta}_{j,1}$ is the j^{th} POD coefficient associated with the first node at Smolyak sparse grid (the simulation set 1). They are POD coefficients when calculating the weights for POD coefficients. $\|\cdot\|_2$ denotes the ℓ_2 Euclidean norm. After obtaining the weights $\boldsymbol{\varpi}_j$, the RBF interpolant $P_j(\mathbf{k}_j)$ is known, and we can use it to do interpolation for any given unseen points.

In Algorithm (1) step (v), a set of hyper-surfaces f_m, $m \in (1, \ldots, M)$, are constructed to represent the physical state of the PDEs over the reduced order space:

$$\boldsymbol{u}_m^{r,n+1} = f_m\left(\boldsymbol{\alpha}_1^{r,n}, \ldots, \boldsymbol{\alpha}_m^{r,n}, \ldots, \boldsymbol{\alpha}_M^{r,n}\right) = \sum_{n_t=0}^{N_t-1} \boldsymbol{w}_m \, \phi_{m,n_t}(\|\boldsymbol{\alpha}^{r,n} - \boldsymbol{c}_m\|), \quad m \in (1, \ldots, M), \tag{13}$$

where M denotes the number of basis functions. $\boldsymbol{\alpha}$ is the POD coefficient $n_t, \in (0, \ldots, N_t - 1)$ and N_t is the total number of time levels. $\boldsymbol{c}_m$ is a vector containing the centre of the m^{th} RBF function. In this work, the centres are chosen such that $\mathbf{c}_m = \boldsymbol{\alpha}^{r,n}$. If we need to use the interpolant f_m to interpolate solutions of current time level, the weights $\boldsymbol{w}_m$ have to be determined. The $\boldsymbol{w}_m$ are weights in physical state, which are different from the weights $\boldsymbol{\varpi}_j$ in the parametric space in Equation (10). The interpolation conditions for determining them are $f_m\left(\boldsymbol{\alpha}_1^{r,n}, \ldots, \boldsymbol{\alpha}_m^{r,n}, \ldots, \boldsymbol{\alpha}_M^{r,n}\right) = \alpha_m^{r,n+1}$, which leads to solving the following linear system:

$$\begin{bmatrix} \phi_m\left(\left\|\boldsymbol{\alpha}^{r,0} - \boldsymbol{\alpha}^{r,0}\right\|_2\right) & \phi_m\left(\left\|\boldsymbol{\alpha}^{r,0} - \boldsymbol{\alpha}^{r,1}\right\|_2\right) & \cdots \phi_m\left(\left\|\boldsymbol{\alpha}^{r,0} - \boldsymbol{\alpha}^{r,N_t-1}\right\|_2\right) \\ \phi_m\left(\left\|\boldsymbol{\alpha}^{r,1} - \boldsymbol{\alpha}^{r,0}\right\|_2\right) & \phi_m\left(\left\|\boldsymbol{\alpha}^{r,1} - \boldsymbol{\alpha}^{r,1}\right\|_2\right) & \cdots \phi_m\left(\left\|\boldsymbol{\alpha}^{r,1} - \boldsymbol{\alpha}^{r,N_t-1}\right\|_2\right) \\ \vdots & \vdots & \vdots \\ \phi_m\left(\left\|\boldsymbol{\alpha}^{r,N_t-1} - \boldsymbol{\alpha}^{r,0}\right\|_2\right) & \phi_m\left(\left\|\boldsymbol{\alpha}^{r,N_t-1} - \boldsymbol{\alpha}^{r,1}\right\|_2\right) & \cdots \phi_m\left(\left\|\boldsymbol{\alpha}^{r,N_t-1} - \boldsymbol{\alpha}^{r,N_t-1}\right\|_2\right) \end{bmatrix} \begin{pmatrix} w_m^0 \\ w_m^1 \\ \vdots \\ w_m^{N_t-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha}_m^{r,1} \\ \boldsymbol{\alpha}_m^{r,2} \\ \vdots \\ \boldsymbol{\alpha}_m^{r,N_t} \end{pmatrix}, \tag{14}$$

After constructing the P-NIROM, an arbitrary point, representing a new unseen parameter set, in the Smolyak sparse grid can be given to the P-NIROM to predict the solutions. This online process can be found in Algorithm (2).

```
Algorithm 2: Online procedure of solving the P-NIROM
Calculate reduced numerical solutions at the current time step (here $N_t$ is the
  number of time levels, $\overline{\boldsymbol{u}}$ is the mean of snapshots):
for $n = 1$ to $N_t$ do
    for $m = 1$ to $M$ do
        (i) Assign a complete set of the reduced solution $\boldsymbol{\alpha}^{r,n} = (\alpha_1^{r,n}, \ldots, \alpha_M^{r,n})$ at previous
            time level $n$ into the hyper-surface $f_m$:
                $f_m \leftarrow (\alpha_1^{r,n}, \ldots, \alpha_m^{r,n}, \ldots, \alpha_M^{r,n})$
        (ii) Calculate $\alpha_m^{r,n+1}$ at the current time level $n+1$ using:
                $\alpha_m^{r,n+1} = f_m\left(\alpha_1^{r,n}, \ldots, \alpha_m^{r,n}, \ldots, \alpha_M^{r,n}\right)$     (15)
    endfor
    Obtain the approximation of the high fidelity solution $\boldsymbol{u}$ at the current time
      level $n+1$ by projecting $\boldsymbol{\alpha}^{r,n+1}$ onto the full space using:
                $\boldsymbol{u}^{n+1} = \overline{\boldsymbol{u}} + \sum_{m=1}^{M} \alpha_m^{r,n+1} \Phi_m$
endfor
```

4. Gaussian process regression

The Gaussian process regression is used as a machine learning tool to construct a set of error response functions representing the residual errors between the high fidelity full model and the P-NIROM. The GPR is a non-parametric model, which assumes that the data distribution is defined in terms of an infinite set of parameters. Compared to the parametric model, the GPR depends more on the data points and is robust to such changes. It can capture more information when given bigger training data, which is more flexible [45].

The Gaussian process regression predicts the output $\boldsymbol{\alpha}^{\boldsymbol{re}}$ associated to input data $\boldsymbol{k}$ (varying parameter vector) based on the η sets of training data (η input-output pairs) $S \equiv (\boldsymbol{k}_1, \alpha_1^{re}), (\boldsymbol{k}_2, \alpha_2^{re}), \cdots, (\boldsymbol{k}_\eta, \alpha_\eta^{re})$. The outputs $\boldsymbol{\alpha}^{\boldsymbol{re}}$ are the POD coefficients of the errors ΔF in Equation (24).

Assuming that the outputs α_j^{re} are calculated by a latent function $g(\boldsymbol{k})$ and corrupted by a mean function $\mu(\boldsymbol{k})$ and a Gaussian noise of constant variance σ_η^2,

$$\alpha_j^{re} = g(\boldsymbol{k}_j) + \delta_j, \quad \delta_j \sim \mathcal{N}(\mu, \sigma_\eta^2), \tag{16}$$

where $g(\boldsymbol{k}_j)$ is the regression model in $\boldsymbol{k}$ (also known as the trend function). The regression method aims to make inference about the function $g(\boldsymbol{k})$. In this work, it is assumed that the observational error δ_j is identically and normal independent distributed with zero mean ($\mu(\boldsymbol{k})$=0).

Gaussian process regression is a probabilistic and non-parametric Bayesian method. For any input $\boldsymbol{k}_i, i \in (1, 2, \cdots \eta)$, the corresponding vector of function $\boldsymbol{g} = [g(\boldsymbol{k}_1), \cdots, g(\boldsymbol{k}_\eta)]^T$ has a joint Gaussian distribution:

$$p(\boldsymbol{g}|\boldsymbol{k}_i) = \mathcal{N}(0, \mathbb{C}), \tag{17}$$

where 0 means setting the mean of the process to zero. The positive semi-definite covariance matrix $\mathbb{C}$ is governed by the covariance function C: $[\mathbb{C}]_{i,j} = C(\boldsymbol{k}_i, \boldsymbol{k}_j) = E[g(\boldsymbol{k}_i)g(\boldsymbol{k}_j)]$. In this work, a Radial Basis Function (RBF) kernel is used, which is also known as squared exponential kernel. It is given by

$$C(\boldsymbol{k}_i, \boldsymbol{k}_j) = exp(-\frac{1}{2}d(\frac{\boldsymbol{k}_i}{l}, \frac{\boldsymbol{k}_j}{l})^2), \tag{18}$$

where d denotes the distance. l is a length-scale parameter, which can either be a scalar or a vector with the same dimensional size with the inputs $\boldsymbol{k}_i$. The predictive distribution $p(\boldsymbol{\alpha}_*^{\boldsymbol{re}}|\boldsymbol{k}_*)$ can be obtained by conditioning on the training outputs:

$$p(\boldsymbol{\alpha}_*^{\boldsymbol{re}}|\boldsymbol{k}_*) = \mathcal{N}(\mu_*, \sigma_*^2), \tag{19}$$

$$\mu_* = \mathbb{C}(\boldsymbol{k}_*, \boldsymbol{k})(\mathbb{C}(\boldsymbol{k}, \boldsymbol{k}) + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{\alpha}^{\boldsymbol{re}} \tag{20}$$

$$\sigma_*^2 = \mathbb{C}(\boldsymbol{k}_*, \boldsymbol{k}_*) - \sigma_n^2\boldsymbol{I} - \mathbb{C}(\boldsymbol{k}_*, \boldsymbol{k})(\mathbb{C}(\boldsymbol{k}, \boldsymbol{k}) + \sigma_n^2\boldsymbol{I})^{-1}\mathbb{C}(\boldsymbol{k}, \boldsymbol{k}_*) \tag{21}$$

where $\boldsymbol{k}$ and $\boldsymbol{\alpha}^{\boldsymbol{re}}$ are the input training data vector and and output training data vector respectively. $\boldsymbol{k}_*$ and $\boldsymbol{\alpha}_*^{\boldsymbol{re}}$ are the input and output test data vectors respectively. $\mathbb{C}(\boldsymbol{k}_*, \boldsymbol{k})$ denotes a $\eta \times \eta_*$ covariance matrix evaluated at all pairs of η training datasets and η test datasets, and this is similarly true for values of $\mathbb{C}(\boldsymbol{k}, \boldsymbol{k})$, $\mathbb{C}(\boldsymbol{k}_*, \boldsymbol{k})$ and $\mathbb{C}(\boldsymbol{k}_*, \boldsymbol{k}_*)$. $\boldsymbol{I}$ is the $\eta \times \eta$ identity. The hyper-parameters in covariance functions determine how quickly covariances decay with distances between inputs.

During the training process of Gaussian process regression, suitable covariance functions and parameters must be chosen. The Gaussian process regression model can be trained by performing Bayesian inference, that is, maximising the logarithmic marginal likelihood, this leads to the minimisation of the negative logarithmic posterior:

$$L(\sigma^2|C) = \frac{1}{2}\boldsymbol{\alpha}^{\boldsymbol{re}T}(\mathbb{C} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\alpha}^{\boldsymbol{re}} + \frac{1}{2}\log|\mathbb{C} + \sigma^2\mathbf{I}| - \log L(\sigma^2) - \log L(C) \tag{22}$$

The hyper-parameters can be obtained by performing partial derivative of Equation (22) with respect to σ^2 and $\mathbb{C}$. The log-marginal-likelihood (LML) can be maximised by a gradient-based search method [46, 47]. As the optimisation of the LML may have an issue of local minima, however, it is usually not a big issue with few hyper-parameters [48]. In addition, it is advised that to start from several random positions in the hyper-parameter space to avoid the local minima problem [49].In this work, scikit-learn machine learning library is used. In Keras, when performing the optimisation, the first run is carried out starting from the kernel's initial hyper-parameter values. In order to avoid multiple local optimal, the subsequent runs in Keras library are undertaken from randomly chosen hyper-parameter values with a range of allowed values [50].

5. Error estimation for P-NIROM using Gaussian process regression

In this section, an error estimator for P-NIROM is derived using the Gaussian process regression method. The essence of the method lies in constructing a set of error response functions $\boldsymbol{f}^{re}$ approximating the remaining (residual) errors ΔF between the high fidelity full model and the P-NIROM.

$$\Delta F = F_{full} - F_{P-NIROM}, \tag{23}$$

where F_{full} and $F_{P-NIROM}$ denote solutions of the high fidelity full model and P-NIROM. There are two types of error in P-NIROM: (i) error from the POD basis functions; (ii) error from the parametric interpolation method. Both of these errors are absorbed in $\boldsymbol{f}^{re}$. The process of deriving an error estimator function using Gaussian process regression machine learning method can be summarised in algorithm 3.

Algorithm 3: Deriving error estimator functions

```
(1) Running a number of simulations for different parameter sets using high fidelity
    full model;
(2) Constructing a P-NIROM using the method described in section 3.
(3) Running a number of simulations for different parameter sets using P-NIROM;
(4) Calculating errors ($\Delta F$) between the high fidelity full model and P-NIROM for
    each simulation;
(5) Calculating the POD basis functions and POD coefficients of errors ($\Delta F$) using
    the POD method.
(6) Constructing a set of error response functions $\boldsymbol{f}^{re}$ to represent the errors using
    machine learning method: Gaussian process regression.
(7) For a given unseen set of parameters, estimate the errors of P-NIROM using the
    error functions $\boldsymbol{f}^{re}$.
(8) For a given unseen set of parameter $\boldsymbol{k}^{un}$, calculate the solutions using P-NIROM
    and update the solutions by considering the errors.
```

The errors (ΔF) in algorithm 3 step (4) are essentially a set of residual snapshots, and therefore have the same dimensional size N with the full model. In order to reduce the intensive computational cost of calculating the errors for a new simulation with a different parameter set, a P-NIROM is constructed using the residual snapshots ΔF. A set of basis functions $\boldsymbol{\phi}^{re} = (\phi_1^{re}, \cdots, \phi_Q^{re})$ and POD coefficients $\boldsymbol{\alpha}^{re} = (\alpha_1^{re}, \cdots, \alpha_Q^{re})$ can be obtained by projecting the ΔF in the full space into a reduced space. The errors (ΔF) can be represented as:

$$\Delta F = \overline{\psi}^{re} + \sum_{j=1}^{Q} \alpha_j^{re} \phi_j^{re}, \tag{24}$$

where $\overline{\psi}^{re}$ is the mean of snapshots of ΔF. α_j^{re} is the POD coefficients of errors (ΔF). ϕ_j^{re} is the POD basis functions of ΔF and Q is the number of basis functions. The POD basis functions ϕ_j^{re} are generated from the residual snapshots ΔF considering all of the training simulation sets. This set of global POD basis functions are used for calculating the training POD coefficients. A set of error response functions $\boldsymbol{f}^{re}$ is constructed for each POD coefficients α_j^{re} using GPR machine learning method. Once the response functions are constructed, then POD coefficients of the new simulation can be predicted by the response functions. The error snapshots of the new simulation can be obtained by projecting the POD coefficients back into the full space using the global POD basis functions ϕ_j^{re}.

In algorithm 3, step (6), the training datasets for constructing the response functions $\boldsymbol{f}^{re}$ are the distribution of parameter sets and POD coefficients of the ΔF (ΔF is the errors between the high fidelity full model and P-NIROM for each simulation). For each single POD coefficient $\alpha_{1,j}^{re}$, a GPR network is constructed and it has G sets of training data: $(\boldsymbol{k}_1, \alpha_{1,j}^{re}), (\boldsymbol{k}_2, \alpha_{2,j}^{re}), \cdots, (\boldsymbol{k}_G, \alpha_{G,j}^{re})$. One single training sample for GPR network, for example $\boldsymbol{k}_1$, is $(k_1, k_2, \cdots, k_{\mathcal{G}})$. The target (output) for this single training sample is α_1^{re}, for example. The inputs and outputs that are used to construct the $\boldsymbol{f}_j^{re}, \forall j \in \{1, 2, \ldots, B\}$ using GPR are:

$$\text{input:} \qquad \boldsymbol{k} = (k_1, k_2, \ldots, k_{\mathcal{G}}) \tag{25}$$

$$\text{output:} \qquad \alpha_j^{re}\ , \tag{26}$$

We have B pairs of inputs and outputs for determining each of the B error response functions, $\boldsymbol{f}_j^{re}$. This involves training of the GPR. The calculation of basis functions for the error snapshots ΔF and the GPR network training are offline procedures, which means they are precomputed. After obtaining the error response functions, $\boldsymbol{f}_j^{re}$, we can use them to calculate the POD coefficients for any new unknown parameter sets using the Equation (27) below,

$$\alpha_j^{\boldsymbol{k}^{un}} = \boldsymbol{f}_j^{re}(\boldsymbol{k}^{un}) = \boldsymbol{f}_j^{re}(\boldsymbol{k}_1^{un}, \boldsymbol{k}_2^{un}, \ldots, \boldsymbol{k}_{\mathcal{G}}^{un}), \qquad \forall j \in \{1, 2, \ldots, B\} \tag{27}$$

where $\boldsymbol{k}^{un}$ is the unknown new parameter set, and $\alpha_j^{\boldsymbol{k}^{un}}$ is the POD coefficient need to be predicted associated with the unknown new parameter set.

In algorithm 3, step (8), the final solutions will consider an additional error term,

$$F_{final} = F_{P-NIROM} + \Delta F = F_{P-NIROM} + \overline{\psi}^{re} + \sum_{j=1}^{Q} \alpha_j^{re} \phi_j^{re} \tag{28}$$

The POD coefficients and basis functions are obtained by the Proper Orthogonal Decomposition method, which involves a singular value decomposition of matrix $\mathcal{E}$ of errors (ΔF) between the high fidelity full model and P-NIROM.

$$\mathcal{E} = U \Sigma V^T. \tag{29}$$

The U and V are the matrices containing the orthogonal vectors for $\mathcal{E}\mathcal{E}^T$ and $\mathcal{E}^T\mathcal{E}$, respectively and Σ is a diagonal matrix with a size of $N \times O$ consisting of singular values λ. The POD basis functions are defined to be [51],

$$\phi_i = \mathcal{E} V_{:,i} / \sqrt{\lambda_i}, \qquad \text{for } i \in \{1, 2 \ldots O\}, \tag{30}$$

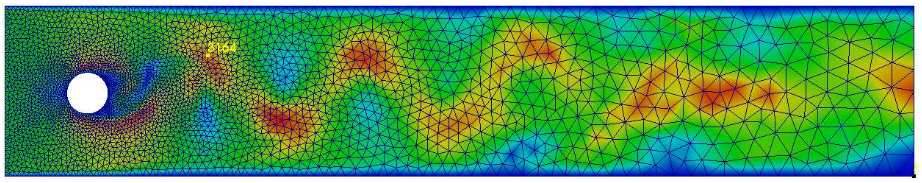


Figure 1: Case 1: computational domain of flow past a cylinder.

An optimal set of functions corresponding to the largest singular values are chosen to approximate the snapshot matrix $\mathcal{E}$. Thus, using POD method, the ΔF can be expressed by, $\Delta F = \overline{\psi}^{re} + \sum_{j=1}^{Q} \alpha_j^{re} \phi_j^{re}$.

6. Illustrative numerical examples

The error estimation method based on machine learning has been implemented in an advanced three-dimensional unstructured finite element mesh fluid model (Fluidity)[52]. The Fluidity model solves the Navier-Stoke and accompanying field equations on a 3D finite element mesh. The Fluidity model is used to generate a number of snapshots for different varying parameter sets, for example, different initial velocity, initial pressure. And the results generated by the Fluidity model are also used to be compared with the results of new presented method. Lapack, scikit-learning and Keras library are used to perform SVD and Gaussian Regression Process [53, 54, 50].

6.1. Case 1: flow past a cylinder

In the first example, a 2D flow past a cylinder case is used to demonstrate the capability of the newly present error estimation method. The computational domain of the flow past a cylinder case is given in Figure 1. The domain has a rectangle size of 2.2×0.41 and a cylinder with a radius of 0.05 is centered at the point (0.2, 0.2). The units are non-dimensional. An inlet velocity is given in the left side of the domain and drives the fluid flows from the left and to the right side. The open boundary condition is applied to the right side of the domain and through the right side of the domain. No slip and zero outward flow conditions are specified at the upper and lower edges and the Dirichlet boundary conditions are applied to the cylinder's wall. The simulation was carried out using Fluidity during the time period $[0, 6]$ with a time step size of 0.01. In this example, 300 snapshots were taken at an equal time interval of $\Delta t = 0.02$ to construct the ROM. The number of nodes in the computational domain is 3213.

The parameter space in this example includes two variables: the inlet velocity k_1 and viscosity k_2. In this case, each parameter point $\mathbf{k}$ has two variables, the inlet velocity k_1 and viscosity k_2, that is $\mathbf{k}_i = \{k_{i,1}, k_{i,2}\}$. A number of random points $(\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_G)$ were generated within the domain $(0.45 \leqslant k_1 \leqslant 0.55, 0.333 \times 10^{-4} \leqslant k_2 \leqslant 1.000 \times 10^{-4})$.

Table 1 lists a combination of two variables (the inlet velocity k_1 and viscosity k_2) for the case of the flow past a cylinder - labelled A1-A20 are the training parameter points. In table 1, k_1 and k_2 denote the inlet velocity and viscosity respectively.

We run Fluidity to generate 20 simulations (A1-A20), and then run NIROM for each simulation. After obtaining the solutions from the high fidelity full model and NIROM, the errors between them can be calculated. We treated those errors as residual snapshots for constructing a Residual Parametric NIROM (RP-NIROM). A set of basis functions and POD coefficients can be obtained by projecting the residual snapshots into a reduced space by POD method. We then use the POD coefficients and parameter sets in Table 1 to train the GPR network. The inputs of the GPR network are the parameter sets and the output of the GPR is a POD coefficient from the complete set of POD coefficient at a time level, see Equations (25) and (26). In this case, the size of the training set for each POD coefficients is 20. After training, the GPR networks can be used to predict the POD coefficients of any unseen parameter set. After obtaining the POD coefficients of the new unseen parameter set, the errors ΔF of the simulation with the new parameter set can be obtained by projecting the POD coefficients back into the full space by Equation (24). After obtaining the ΔF, we can obtain the final results by absorbing the ΔF into the P-NIROM.

A new untrained parameter point ($k_1 = 0.520$, $k_2 = 0.800{\times}10^{-4}$) - labelled T1 is chosen to demonstrate the capability of the Residual parametric NIROM (RP-NIROM). In the following texts, RP-NIROM will be used to denote the P-NIROM considering remaining errors between the high fidelity full model and P-NIROM.

Table 1: Case 1: a list of combination of two variables for the flow past a cylinder (parameter I k_1: inlet velocity; parameter II k_2: viscosity)

Cases	k_1	k_2	Reynolds	cases	k_1	k_2	Reynolds
A1	0.4800	$0.800{\times}10^{-4}$	1200	A11	0.515	$0.800{\times}10^{-4}$	1288
A2	0.5000	$0.650{\times}10^{-4}$	1538	A12	0.450	$0.350{\times}10^{-4}$	2571
A3	0.4500	$0.650{\times}10^{-4}$	1384	A13	0.550	$0.350{\times}10^{-4}$	3143
A4	0.5000	$0.500{\times}10^{-4}$	2000	A14	0.500	$0.800{\times}10^{-4}$	1250
A5	0.5500	$0.650{\times}10^{-4}$	1692	A15	0.450	$1.000{\times}10^{-4}$	900
A6	0.5000	$0.350{\times}10^{-4}$	2857	A16	0.550	$1.000{\times}10^{-4}$	1100
A7	0.5000	$1.000{\times}10^{-4}$	1000	A17	0.500	$0.450{\times}10^{-4}$	2222
A8	0.4250	$0.650{\times}10^{-4}$	1308	A18	0.500	$0.900{\times}10^{-4}$	1111
A9	0.5354	$0.650{\times}10^{-4}$	1647	A19	0.480	$0.500{\times}10^{-4}$	1920
A10	0.2354	$0.650{\times}10^{-4}$	724	A20	0.515	$0.500{\times}10^{-4}$	2060
T1	0.520	$0.800{\times}10^{-4}$	1300				

Figure 3 shows the singular values and logarithmic singular values of a flow past a cylinder case: $k_1 = 0.450$, $k_2 = 1.000{\times}10^{-4}$ (the sample training case A15 in Table 1). The figure provides a criterion of choosing number of POD basis functions. The larger number of POD basis functions is chosen, the more accurate results of NIROM can be obtained. In this work, 12 basis functions were chosen to construct the P-NIROM and demonstrate the capability of the RP-NIROM. After obtaining the solutions of P-NIROM, the residual solution errors between the high fidelity full model and P-NIROM can be calculated. And those solution errors constitute the residual error snapshots. There are 20 training sets, and each training set has 300 residual error snapshots. Therefore, there are 6000 residual error snapshots in total. In order to reduce the dimensional size, POD method was used to generate a reduced representation of those residual solution error snapshots. 12 POD basis functions were generated from those residual error snapshots. The first, second, third, 7th, 9th and 12nd POD basis functions generated from the residual error snapshots are given in Figure 13. The figure shows us the energy captured by the basis functions from the residual error snapshots. The singular values and logarithmic singular values of the residual error snapshots are given in Figure 4. It provides a criterion of choosing number of POD basis functions for the residual error snapshots. The singular values in Figure 4 decrease slower than those in Figure 3. As such, for the residual error snapshots, a larger number of POD basis functions is needed to be chosen.

The predicted POD coefficients of the residual error snapshots (ΔF) using GPR is compared with the true values (standard ROM), in Figure 5, which shows the first and second POD coefficients from the true and solutions predicted using GPR. 80% of the data is used as the training, while the remaining 20 % is used for cross validation. After projecting the POD coefficients back into the full space, the predicted errors using GPR can be obtained. Figure 6 shows exact ΔF results (top) and predicted ΔF using GPR (bottom). It can be seen that the predicted solutions of ΔF have close agreement with exact ΔF and the GPR predicts well using training datasets with a moderate size. This is an advantage of Gaussian-process kernel, which converges well even using a moderately sized set of training points [41]. In order to see the performance of GPR prediction, the prediction errors of all nodes in the computational domain is considered. The prediction errors of GPR is analysed using root-mean-square error (RMSE) and correlation coefficients. The correlation coefficient (CC) is defined as

$$\mathrm{CC}(\varphi(t), \varphi_o(t)) = \frac{\mathrm{cov}(\varphi(t), \varphi_o(t))}{\sigma_{\varphi(t)}\sigma_{\varphi_o(t)}} = \frac{\mathrm{E}(\varphi(t) - \mu_{\varphi(t)})(\varphi_o(t) - \mu_{\varphi_o(t)})}{\sigma_{\varphi(t)}\sigma_{\varphi_o(t)}}. \tag{31}$$

where $\mu_{\varphi(t)}$ and $\mu_{\varphi_o(t)}$ are expected values; $\sigma_{\varphi(t)}$ and $\sigma_{\varphi_o(t)}$ are standard deviations; $\varphi_i(t)$ and $\varphi_{o,i}(t)$ denote the predicted solution and the exact solution at node i and time t, respectively; The RMSE is defined as

$$\mathrm{RMSE}(t) = \sqrt{\frac{\sum_{i=1}^{N}(\varphi_i(t) - \varphi_{o,i}(t))^2}{N}}, \tag{32}$$

Figure 7 shows RMSE and correlation coefficients of predicted GPR results. The RMSE shows that the predicted error of GPR is small and the the correlation coefficients are very closer to 1, which indicates that the predicted results using GPR are strongly associated to the exact solutions.

Figure 8 shows velocity solutions obtained from the high fidelity full model, P-NIROM and RP-NIROM with 12 POD basis functions at t = 3s and t = 5s. As shown in the figure, the results of RP-NIROM are closer to the high fidelity full model than those of P-NIROM. In order to see the difference clearly of three models (high fidelity full model, P-NIROM, RP-NIROM), velocity solutions from those three models at a particular node (see the yellow point in Figure 11) within the domain (x = 0.49111, y = 0.29193) are shown in Figure 9. Again, it shows that the RP-NIROM performs better than P-NIROM.

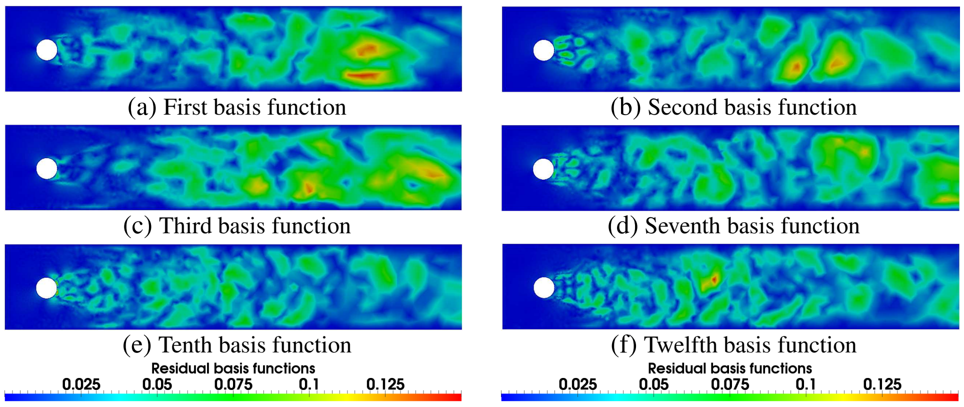


Figure 2: Case 1: the figures displayed above show the first, second, third, seventh, tenth and twelfth POD basis functions of the residual errors between the high fidelity full model and P-NIROM.

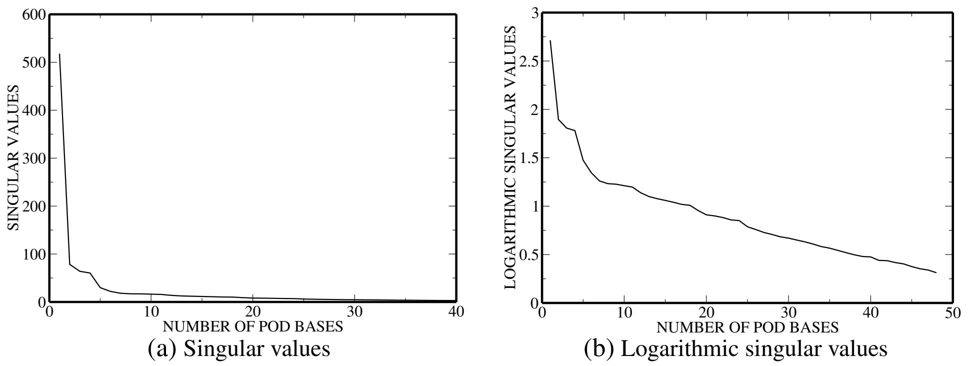
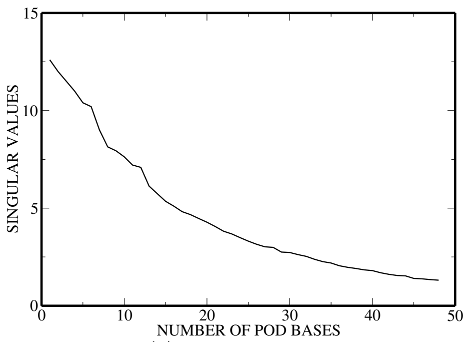
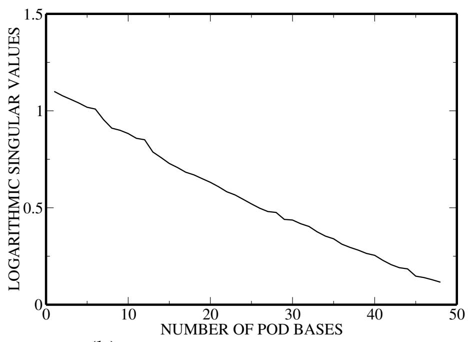


Figure 3: Case 1: The figure shows the singular values and logarithmic singular values of the test case: k_1 = 0.450, k_2 = 1.000×10^{-4}.

The error analysis is carried out by RMSE and correlation coefficients considering all nodes in the computational domain. In Figure 10, it shows the RMSE and correlation coefficients of test case: $k_1 = 0.520$, $k_2 = 0.800\times10^{-4}$. As shown in the figure, the results of P-NIROM are improved by considering the residual errors.

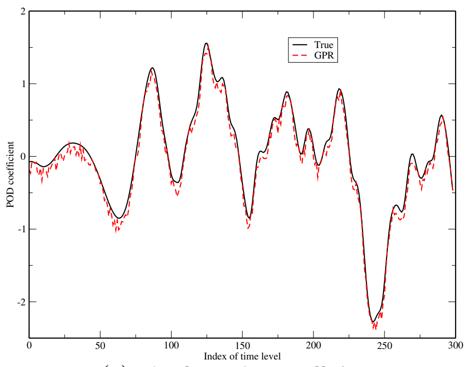


(a) Singular values

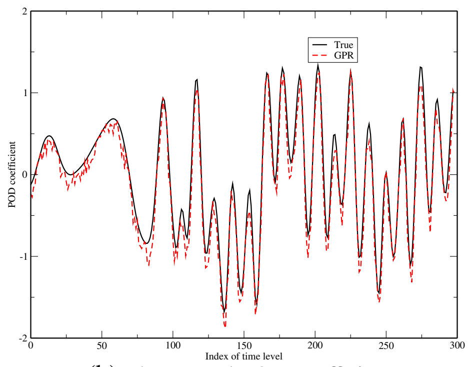


(b) Logarithmic singular values

Figure 4: Case 1: the figures displayed above show the singular values and logarithmic singular values of errors between the high fidelity full model and P-NIROM from test case: $k_1 = 0.520$, $k_2 = 0.800 \times 10^{-4}$.



(a) The first POD coefficient



(b) The second POD coefficient

Figure 5: Case 1: Comparisons between the true and predicted values using GPR of the first and second POD coefficients (the black line: true, the red dash line: GPR.

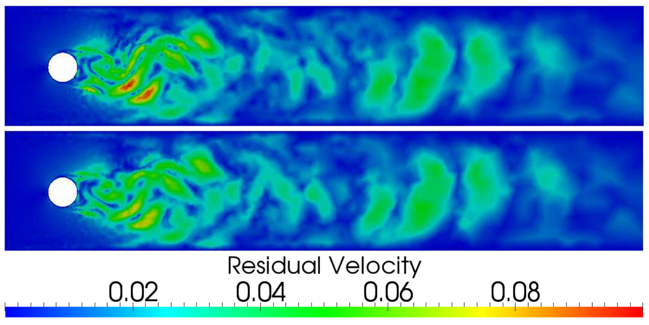
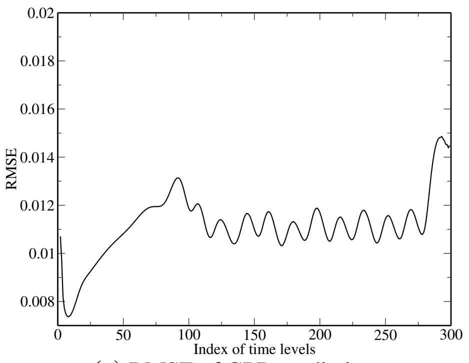
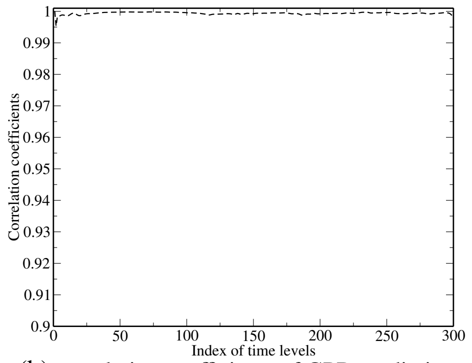


Figure 6: Case 1: The figure shows the ΔF results from exact solutions and predicted solutions using GPR. Top: standard ROM; Bottom: GPR

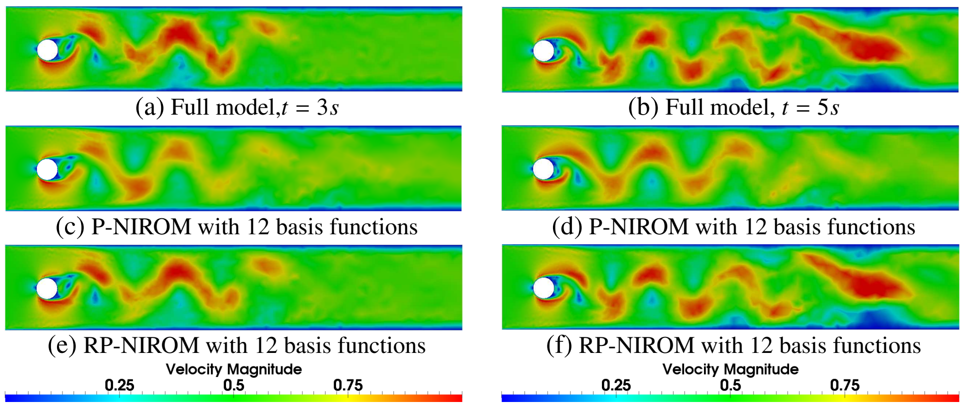


(a) RMSE of GPR prediction.



(b) correlation coefficients of GPR prediction.

Figure 7: Case 1: The figure shows the RMSE and correlation coefficients of GPR prediction for ΔF.



(a) Full model,t = 3s

(b) Full model, t = 5s

(c) P-NIROM with 12 basis functions

(d) P-NIROM with 12 basis functions

(e) RP-NIROM with 12 basis functions

(f) RP-NIROM with 12 basis functions

Figure 8: Case 1: the figures displayed above show velocity solutions obtained from the high fidelity full model, P-NIROM and RP-NIROM with 12 POD basis functions at t = 3s and t = 5s.

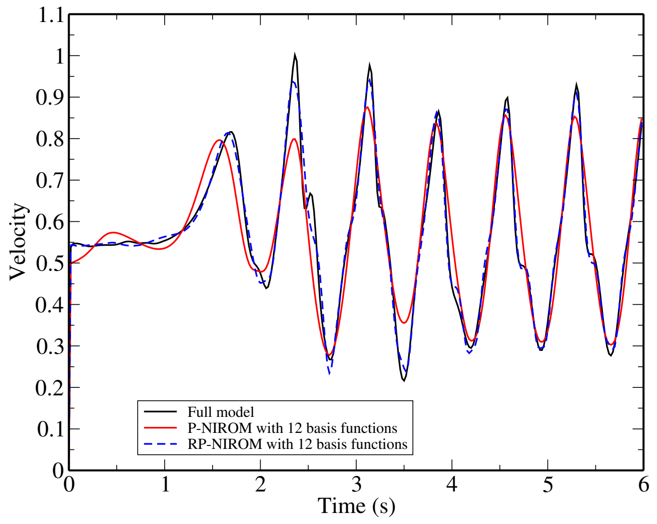


Figure 9: Case 1: Comparison of the velocity solutions at a particular node within the domain (x = 0.49111, y = 0.29193).

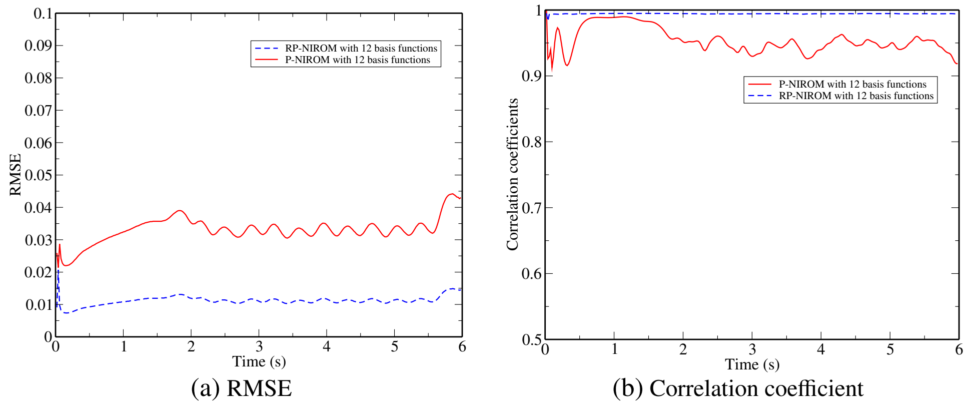


Figure 10: Case 1: The figure shows the RMSE and correlation coefficients from P-NIROM and RP-NIROM for the test case: $k_1 = 0.520$, $k_2 = 0.800 \times 10^{-4}$.

6.2. *Case 2: water dam break*

The second example is a dam break problem, which has a collapsing column of liquid, normally water, within a vacuum or atmosphere. In this example, a reservoir of water is held behind a barrier that separates the water from the rest of the tank. There is a cylinder inside the tank. The barrier is then instantaneously removed and the water column collapses due to the gravity (g=9.81)[55, 56]. The computational domain of the dam break case is given in figure 11. The domain has a rectangle size of 1×1 and a cylinder with a radius of 0.1 is centered at the point (0.5, 0.125). Dirichlet boundary condition was specified at the bottom, left and right side. The initial velocity of water is 0. The simulation was carried out using Fluidity during the time period $[0, 7.5]$ with a time step size of 0.0001. In this example, 150 snapshots were taken at an equal time interval of $\Delta t = 0.05$ to construct the ROM. The number of nodes in the computational domain is 4625. The parameter space in this example is the initial height of water. 12 different initial water heights between 0 and 1 were chosen to run the Fluidity, see Table 2. Those 12 simulations constitute the training data of the P-NIROM. Each simulation has 150 snapshots. A new untrained initial water height 0.55 is chosen to demonstrate the capability of the RP-NIROM. This example also shows the RP-NIROM's capability of interface capturing.

Table 2: Case 2: a list of cases (parameter I k_1: initial height of water)

Cases	k_1	Cases	k_1	cases	k_1	Cases	k_1
A1	0.1	A4	0.35	A7	0.6	A10	0.9
A2	0.2	A5	0.4	A8	0.7	A11	0.95
A3	0.3	A6	0.5	A9	0.8	A12	1
T1	0.55						

The singular values and logarithmic singular values of the training case A6 ($k_1 = 0.5$) in Table 2 in order of decreasing magnitude are presented in Figure 12. The figure provides a criterion of choosing number of POD basis functions for this case. In this work, 48 basis functions were chosen to construct the P-NIROM and demonstrate the capability of the RP-NIROM.

After obtaining the P-NIROM, the solution errors between the high fidelity full model and P-NIROM can be calculated. And those solution errors constitute the residual error snapshots. In order to reduce the dimensional size, POD method was used to generate a reduced representation of the residual solution errors. Figure 13 shows the first, second, third, 7th, 9th and 12nd POD basis functions of the residual errors between the high fidelity full model and P-NIROM. It can be seen that those basis functions contain more information around the interface area (interface between the water and the air) than other areas. Generally, larger number of basis functions are required to capture the interface area. In this work, the error estimator is considered into the P-NIROM in order to capture the more details of the interface area. The error estimator is constructed by the GPR networks. The GPR networks are trained for each POD coefficients of the new case with new different parameter sets. The size of the training set for each POD coefficient in this case is 12. Figure 14 shows the singular values and logarithmic singular values of the residual error solutions in order of

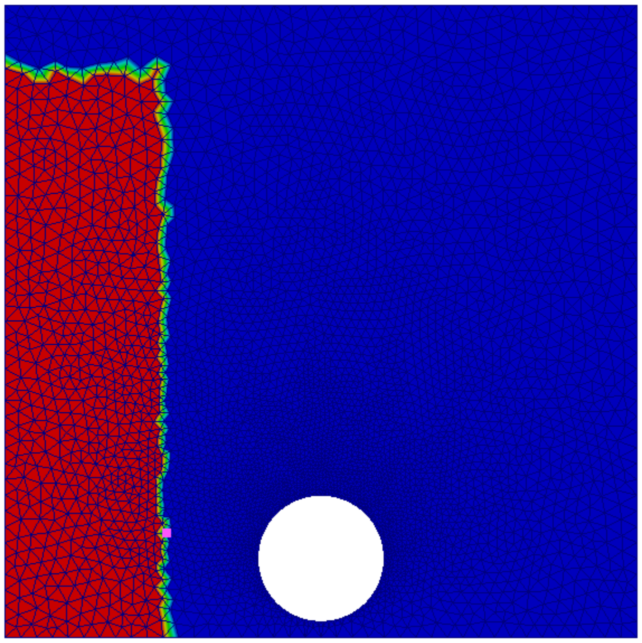
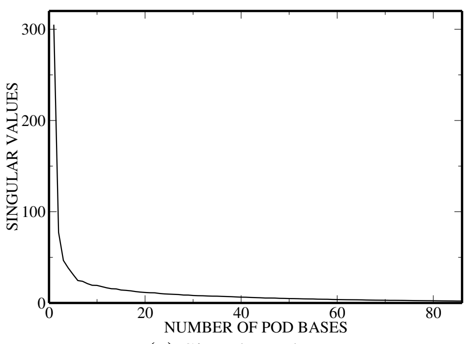
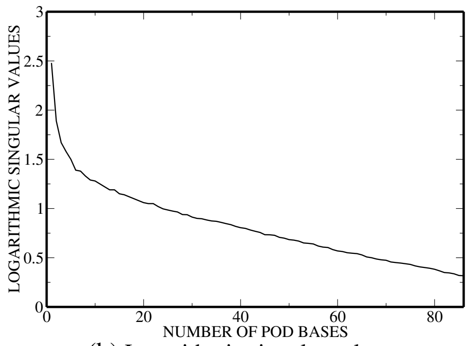


Figure 11: Case 2: Computational domain of water height = 0.9.



(a) Singular values



(b) Logarithmic singular values

Figure 12: Case 2: The figure shows the singular values and logarithmic singular values of training case (A6) in Table 2: water height = 0.5.

decreasing magnitude.

Figure 15 shows the solutions obtained from the high fidelity full model, P-NIROM and RP-NIROM with 48 basis functions at $t = 0.05$s and $t = 0.125$s. As shown in the figure, the results obtained from both P-NIROM and RP-NIROM are close to those of the high fidelity full model. In order to see the difference, the volume fraction solutions at a particular node (see the pink point in Figure 11) within the computational domain ($x = 0.32289, y = 0.34007$) are given in Figure 16. As shown in the figure, the P-NIROM considering error estimate (RP-NIROM) perform better than P-NIROM.

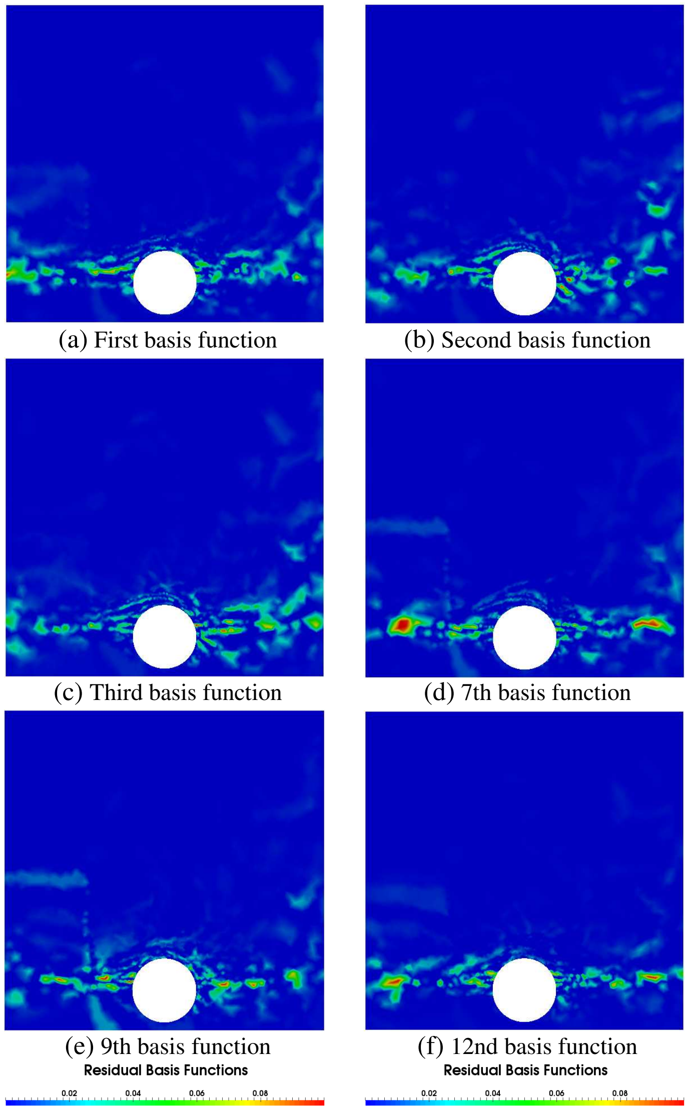
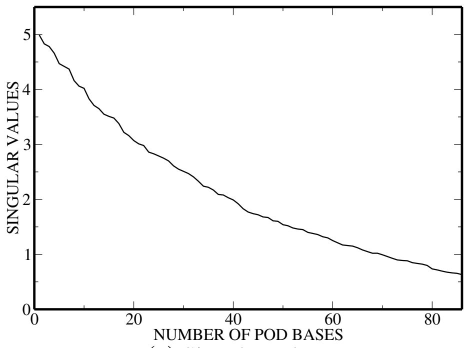
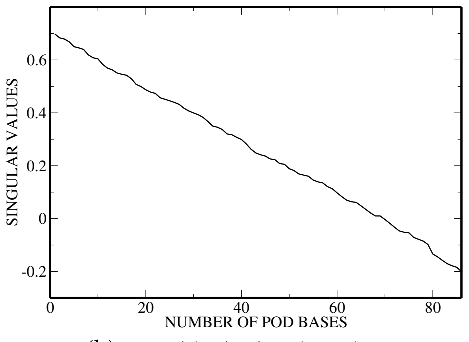


Figure 13: Case 2: the figures displayed above show the first, second, third, 7th, 9th and 12nd POD basis functions of the residual errors between the high fidelity full model and P-NIROM.

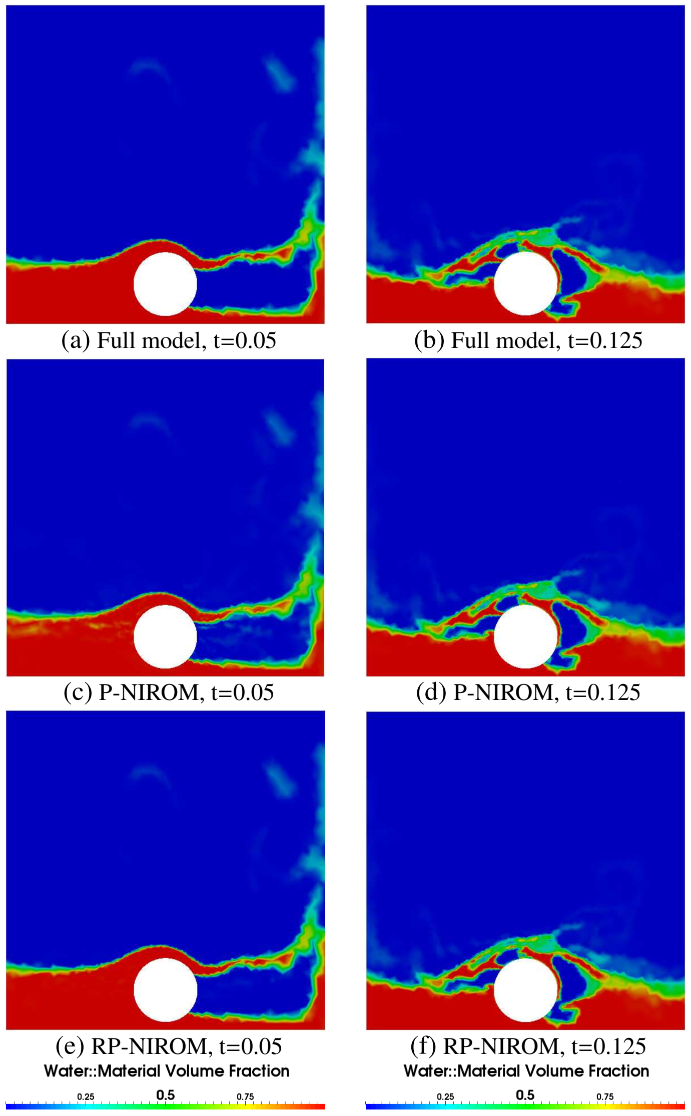


(a) Singular values



(b) Logarithmic singular values

Figure 14: Case 2: the figures displayed above show the singular values and logarithmic singular values of errors between the high fidelity full model and P-NIROM from the test case: water height = 0.5.



(a) Full model, t=0.05 (b) Full model, t=0.125

(c) P-NIROM, t=0.05 (d) P-NIROM, t=0.125

(e) RP-NIROM, t=0.05 (f) RP-NIROM, t=0.125

Figure 15: Case 2: the figures displayed above show the solutions obtained from the high fidelity full model, P-NIROM and RP-NIROM with 48 basis functions at t=0.05 and t=0.125.

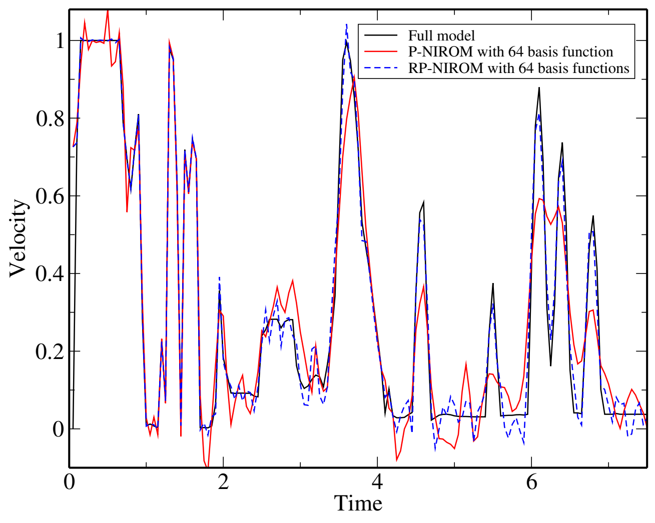


Figure 16: Case 2: Comparison of the volume fraction solutions at a particular node within the domain (x = 0.32289, y = 0.34007).

6.3. Case 3: 3D fluvial channel case

The third example is a three dimensional fluvial channel reservoir case. The reservoir contains three channels and each channel has a different permeability, as shown in different color in Figure 17. The flow passes through the three channels from the right side to the left due to the injected water pressure in the right side. The porosity is 0.2 in the computational domain and the viscosities of the irreducible water and residual oil are 0.004 and 0.001 $Pa.s$ respectively. The geometry of the reservoir is constructed using B-splines (NURBS) surfaces and curves, which is an efficient way to represent geological heterogeneity [57]. The simulations were run on IC-FERST, which is an multiphase simulator based on Fluidity [58]. The simulation period is $[0, 1000]$ days with a fixed time step of 10 days. A snapshot is taken every 25 days and a total of 40 snapshots of solutions were taken for each solution variable.

The parameter space in this 3D case is the permeability of each channel. 30 training simulations (A1-A30 in Table 3) were generated using IC-FERST and each simulation includes 40 snapshots. A new unseen permeability set (T1 in Table 3) is chosen to demonstrate the capability of the RP-NIROM. The error estimator is constructed using the GPR networks. The GPR networks are trained for each POD coefficients of the new case with new different parameter sets. The size of the training set for each POD coefficient in this case is 30.

Figure 18 shows the solutions obtained from the high fidelity full model, P-NIROM and RP-NIROM with 12 basis functions at day 250 and 750. As shown in the figure, the results obtained from both P-NIROM and RP-NIROM are close to those of the high fidelity full model. In order to see the difference, the solutions at a particular node within the computational domain ($x = 223.79, y = 496.56, z = 140$) are given in Figure 20. As shown in the figure, the P-NIROM considering error estimate (RP-NIROM) perform better than P-NIROM. Figure 19 shows exact ΔF results (left) and predicted ΔF using GPR (right). It can be seen that the predicted solutions of ΔF have close agreement with exact ΔF and the GPR predicts well using training datasets with a moderate size.

6.4. Computational efficiency

Table 4 shows the online CPU cost required for simulating the high-fidelity full model and NIROM for each time step. It is worth noting that the online CPU time (dimensionless) required for running the NIROM during one time step is only 0.004s, while the full model for the dam break case is 238 seconds. The simulations were performed on 12 cores of a workstation with Intel(R) Xeon(R) X5680 CPU processors of 3.3GHz and 64GB RAM. The CPU cost of the full model is dependent on the resolution of mesh, which means the computation time increases when a finer mesh is used.

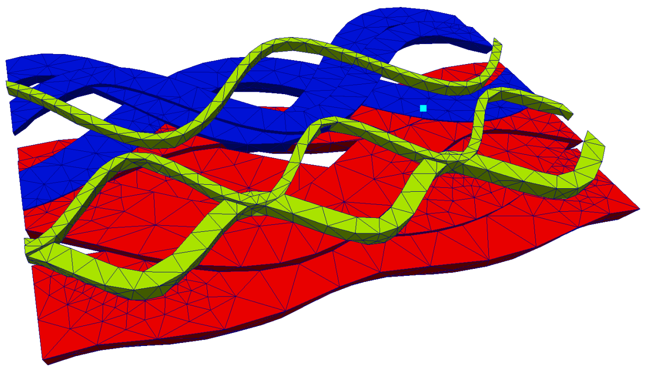


Figure 17: Case 3: Computational domain of 3D fluvial channel case. It includes three channels: big one (red), middle one(blue) and small one (cyan).

Table 3: Case 3: a list of combination of three permeabilities of the three channels for the 3D fluvial channel case.

Cases	k_1	k_2	k_3	cases	k_1	k_2	k_3
A1	4.93×10^{-14}	4.93×10^{-14}	4.93×10^{-14}	A16	1.97×10^{-13}	3.95×10^{-13}	9.87×10^{-13}
A2	4.93×10^{-14}	9.87×10^{-13}	9.87×10^{-14}	A17	5.92×10^{-13}	4.93×10^{-13}	4.93×10^{-13}
A3	9.87×10^{-13}	4.93×10^{-13}	4.93×10^{-13}	A18	2.76×10^{-13}	9.87×10^{-13}	1.97×10^{-13}
A4	9.87×10^{-13}	9.87×10^{-13}	3.95×10^{-13}	A19	3.45×10^{-13}	3.45×10^{-13}	9.87×10^{-13}
A5	4.69×10^{-13}	4.69×10^{-13}	2.96×10^{-13}	A20	7.40×10^{-13}	7.40×10^{-13}	9.87×10^{-13}
A6	2.47×10^{-13}	2.47×10^{-13}	2.47×10^{-13}	A21	3.45×10^{-13}	6.42×10^{-13}	8.39×10^{-13}
A7	2.47×10^{-13}	7.90×10^{-13}	2.47×10^{-13}	A22	5.43×10^{-13}	1.48×10^{-13}	6.42×10^{-13}
A8	7.90×10^{-13}	2.47×10^{-13}	8.88×10^{-13}	A23	8.39×10^{-13}	3.45×10^{-13}	8.88×10^{-13}
A9	7.90×10^{-13}	7.90×10^{-13}	7.90×10^{-13}	A24	2.96×10^{-13}	1.97×10^{-13}	6.42×10^{-13}
A10	6.42×10^{-13}	8.88×10^{-13}	2.47×10^{-13}	A25	8.39×10^{-13}	4.93×10^{-13}	6.91×10^{-13}
A11	4.44×10^{-13}	3.45×10^{-13}	4.93×10^{-13}	A26	1.97×10^{-13}	6.91×10^{-13}	9.87×10^{-13}
A12	6.42×10^{-13}	5.43×10^{-13}	7.40×10^{-13}	A27	4.93×10^{-13}	4.93×10^{-13}	4.93×10^{-13}
A13	1.48×10^{-13}	5.43×10^{-13}	4.93×10^{-13}	A28	7.40×10^{-13}	7.40×10^{-13}	7.40×10^{-13}
A14	9.67×10^{-13}	5.53×10^{-13}	1.97×10^{-13}	A29	9.87×10^{-13}	9.87×10^{-13}	9.87×10^{-13}
A15	4.93×10^{-13}	4.93×10^{-13}	8.39×10^{-13}	A30	9.87×10^{-14}	7.40×10^{-13}	9.87×10^{-14}
T1	3.8×10^{-13}	3.0×10^{-14}	9.87×10^{-13}				

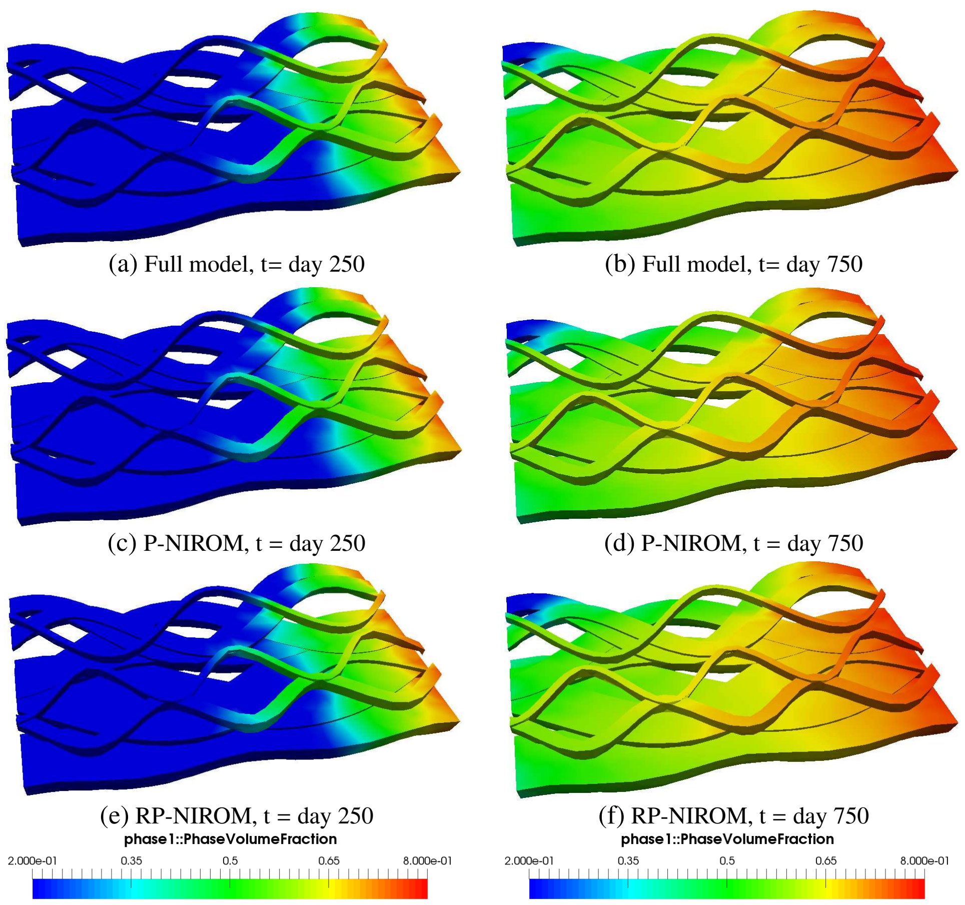

(a) Full model, t= day 250
(b) Full model, t= day 750
(c) P-NIROM, t = day 250
(d) P-NIROM, t = day 750
(e) RP-NIROM, t = day 250
(f) RP-NIROM, t = day 750
phase1::PhaseVolumeFraction
2.000e-01
0.35
0.5
0.65
8.000e-01
phase1::PhaseVolumeFraction
2.000e-01
0.35
0.5
0.65
8.000e-01

Figure 18: Case 3: the figures displayed above show the solutions obtained from the high fidelity full model, P-NIROM and RP-NIROM with 12 basis functions at t= t = day 250 and t = day 750.

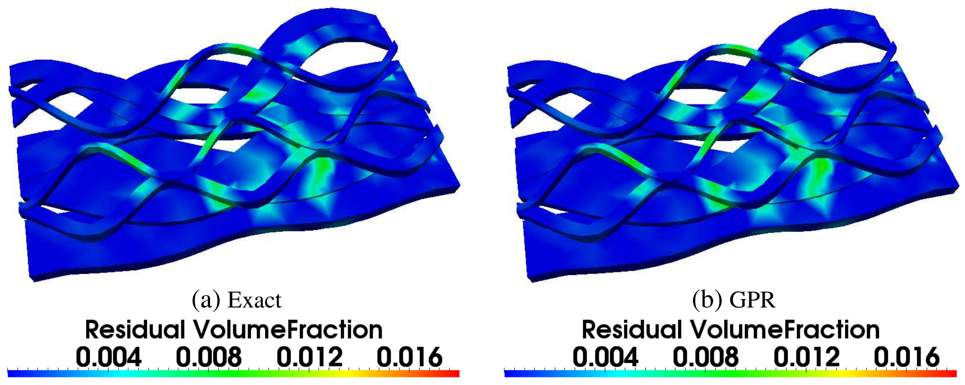


Figure 19: Case 3: The figure shows the ΔF results from exact solutions and predicted solutions using GPR. Left: standard ROM; Right: GPR.

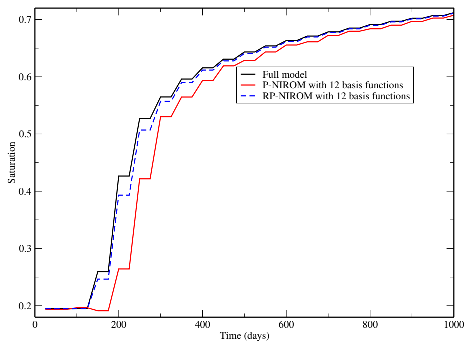


Figure 20: Case 3: Comparison of the volume fraction solutions at a particular node (aqua point in Figure 17) within the domain (x = 223.79, y = 496.56, z=140).

Table 4: Comparison of the online CPU time (dimensionless) required for running the full model and NIROM during one time step.

Cases	Model	assembling and solving	projection	interpolation	total
Flow past a cylinder	Full model NIROM	3.11 0	0 0.003	0 0.001	3.11 0.0040
Water dam break	Full model NIROM	238 0	0 0.003	0 0.001	238 0.0040
3D channel	Full model NIROM	74.9200 0	0 0.003	0 0.001	74.9200 0.0040

7. Conclusion

An error estimator for P-NIROM based on a Gaussian Processing Regression machine learning method has been, for the first time, presented in this paper. A GPR machine learning method has been used to construct a set of error response functions for remaining errors between the high fidelity full model and P-NIROM. The error estimator has been implemented in the framework of an advanced 3-D unstructured finite element mesh fluid model (Fluidity). The performance of RP-NIROM considering the error estimator has been illustrated by three numerical examples: flow past a cylinder, water dam break and 3D fluvial channel test cases. A detailed comparison between the RP-NIROM and high fidelity full model has been made. The numerical examples show that the RP-NIROM performs better than P-NIROM when absorbing the remaining errors between the high fidelity full model and P-NIROM. A significant CPU speed up is also obtained compared to the high fidelity full model. The advantage of the proposed method is that it can find valuable information from the residual error data. The valuable information can be used to improve the accuracy of the P-NIROM. However, compared to standard P-NIROM, an extra procedure of constructing a set of error response functions using machine learning methods is required. This extra procedure includes offline and online procedures. The offline procedure involves constructing the error response functions, which is precomputed. The online procedure involves using the error functions to predict, which is fast. Future work will investigate the performance of this error estimator for more complex problems such as air pollution, large city scale urban flows and flooding problems.

Acknowledgments

The author would like to acknowledge the support of IMPACT (Innovative Materials, Processing And numeriCal Technologies, part-funded by the European Regional Development Fund through the Welsh Government and Swansea University), EPSRC grant: Managing Air for Green Inner Cities (MAGIC) (EP/N010221/1) and the Innovate UK Smart-GeoWells consortium (EP/R005761/1). Thank Idris Abebayo for setting up and running the dam break test case and thank Carl Jacquemyn and Pablo Salinas for setting up the 3D channel case.

References

[1] Clarence Rowley and Scott Dawson. Modal analysis of fluid flows using variants of proper orthogonal decomposition. *Bulletin of the American Physical Society*, 62, 2017.

[2] F. Fang, T. Zhang, D. Pavlidis, C.C. Pain, A.G. Buchan, and I.M. Navon. Reduced order modelling of an unstructured mesh air pollution model and application in 2D/3D urban street canyons. *Atmospheric Environment*, 96:96–106, 2014.

[3] F Fang, C.C. Pain, I.M. Navon, GJ Gorman, MD Piggott, PA Allison, and AJH Goddard. A POD goal-oriented error measure for mesh optimization. *International Journal for Numerical Methods in Fluids*, 63(2):185–206, 2010.

[4] M. Diez, E.F. Campana, and F. Stern. Design-space dimensionality reduction in shape optimization by Karhunen–Loève expansion. *Computer Methods in Applied Mechanics and Engineering*, 283:1525–1544, 2015.

[5] Niccolò Dal Santo, Simone Deparis, Andrea Manzoni, and Alfio Quarteroni. An algebraic least squares reduced basis method for the solution of nonaffinely parametrized stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 2018.

[6] Andrea Manzoni, Diana Bonomi, and Alfio Quarteroni. Reduced order modeling for cardiac electrophysiology and mechanics: New methodologies, challenges and perspectives. In *Mathematical and Numerical Modeling of the Cardiovascular System and Applications*, pages 115–166. Springer, 2018.

[7] Manal Alotaibi, Victor M Calo, Yalchin Efendiev, Juan Galvis, and Mehdi Ghommem. Global–local nonlinear model reduction for flows in heterogeneous porous media. *Computer Methods in Applied Mechanics and Engineering*, 292:122–137, 2015.

[8] R.Stefanescu and I.M. Navon. POD/DEIM Nonlinear model order reduction of an ADI implicit shallow water equations model. *Journal of Computational Physics*, 237:95–114, 2013.

[9] R. Stefanescu, A. Sandu, and I.M. Navon. Comparison of POD reduced order strategies for the nonlinear 2D shallow water equations. *International Journal for Numerical Methods in Fluids*, 76(8):497–521, 2014.

[10] D.N. Daescu and I.M. Navon. A Dual-Weighted Approach to Order Reduction in 4D-Var Data Assimilation. *Monthly Weather Review*, 136(3):1026–1041, 2008.

[11] A. Manzoni, F. Salmoiraghi, and L. Heltai. Reduced Basis Isogeometric Methods (RB-IGA) for the real-time simulation of potential flows about parametrized NACA airfoils. *Computer Methods in Applied Mechanics and Engineering*, 284:1147–1180, 2015.

[12] AG Buchan, AA Calloo, MG Goffin, S Dargaville, F Fang, CC Pain, and IM Navon. A POD reduced order model for resolving angular direction in neutron/photon transport problems. *Journal of Computational Physics*, 296:138–157, 2015.

[13] H. Chen. Blackbox stencil interpolation method for model reduction. Master's thesis, Massachusetts Institute of Technology, 2012.

[14] D. Xiao, F. Fang, A.G. Buchan, C.C. Pain, I.M. Navon, and A. Muggeridge. Non-intrusive reduced order modelling of the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293:552–541, 2015.

[15] D. Xiao, P. Yang, F. Fang, J. Xiang, C.C. Pain, I.M. Navon, and M. Chen. A non-intrusive reduced-order model for compressible fluid and fractured solid coupling and its application to blasting. *Journal of Computational Physics*, 330:221–244, 2017.

[16] Michael Schlegel and Bernd R. Noack. On long-term boundedness of Galerkin models. *Journal of Fluid Mechanics*, 765:325–352, 2 2015.

[17] Jan Osth, Bernd R. Noack, SiniÅa Krajnovi, Diogo Barros, and Jacques Bore. On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body. *Journal of Fluid Mechanics*, 747:518–544, 5 2014.

[18] David Amsallem and Charbel Farhat. Stabilization of projection-based reduced-order models. *International Journal for Numerical Methods in Engineering*, 91(4):358–377, 2012.

[19] Leopoldo P. Franca and Sergio L. Frey. Stabilized finite element methods: II. The incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 99(2-3):209–233, 1992.

[20] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput*, 32:2737–2764, 2010.

[21] Feriedoun S. and Alireza J. α Regularization of the POD-Galerkin dynamical systems of the Kuramoto-Sivashinsky equation. *Applied Mathematics and Computation*, 218(10):6012 – 6026, 2012.

[22] D. Xiao, F. Fang, A.G. Buchan, C.C. Pain, I.M. Navon, J. Du, , and G. Hu. Non-linear model reduction for the Navier-Stokes equations using Residual DEIM method. *Journal of Computational Physics*, 263:1–18, 2014.

[23] D. Xiao, F. Fang, J. Du, C.C. Pain, I.M. Navon, A.G. Buchan, A.H. ElSheikh, and G. Hu. Non-linear Petrov-Galerkin methods for reduced order modelling of the Navier-Stokes equations using a mixed finite element pair. *Computer Methods In Applied Mechanics and Engineering*, 255:147–157, 2013.

[24] C. Audouze, F.D. Vuyst, and P.B. Nair. Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. *Numerical Methods for Partial Differential Equations*, 29(5):1587–1628, 2013.

[25] Z. Wang, D. Xiao, F. Fang, R. Govindan, C.C. Pain, and Y. Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[26] D. Xiao, F. Fang, C.C. Pain, and I.M. Navon. A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Computer Methods in Applied Mechanics and Engineering*, 317:868–889, 2017.

[27] Mengwu Guo and Jan S Hesthaven. Reduced order modeling for nonlinear structural analysis using gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 341:807–826, 2018.

[28] Mengwu Guo and Jan S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 2018.

[29] D. Xiao, P. Yang, F. Fang, J. Xiang, C.C. Pain, and I.M. Navon. Non-intrusive reduced order modeling of fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 303:35–54, 2016.

[30] Jan S Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[31] Yves Le Guennec, J-P Brunet, F-Z Daim, Ming Chau, and Yves Tourbier. A parametric and non-intrusive reduced order model of car crash simulation. *Computer Methods in Applied Mechanics and Engineering*, 338:186–207, 2018.

[32] Tariq Benamara, Piotr Breitkopf, Ingrid Lepot, and Caroline Sainvitu. LPC Blade and Non-Axisymmetric Hub Profiling Optimization Using Multi-Fidelity Non-Intrusive POD Surrogates. In *ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition*, pages V02CT47A026–V02CT47A026. American Society of Mechanical Engineers, 2017.

[33] Chris Homescu, Linda R Petzold, and Radu Serban. Error estimation for reduced-order models of dynamical systems. *Siam Review*, 49(2):277–299, 2007.

[34] JH Chaudhry, D Estep, and M Gunzburger. Exploration of efficient reduced-order modeling and a posteriori error estimation. *International Journal for Numerical Methods in Engineering*, 111(2):103–122, 2017.

[35] Saifon Chaturantabut and Danny C Sorensen. A state space error estimate for POD-DEIM nonlinear model reduction. *SIAM Journal on numerical analysis*, 50(1):46–63, 2012.

[36] D. Wirtz and B. Haasdonk. Efficient a-posteriori error estimation for nonlinear kernel-based reduced systems. *Systems and Control Letters*, 61(1):203–211, 2012.

[37] D Wirtz, DC Sorensen, and Bernard Haasdonk. A posteriori error estimation for deim reduced nonlinear dynamical systems. *SIAM Journal on Scientific Computing*, 36(2):A311–A338, 2014.

[38] Azam Moosavi, Răzvan Ştefănescu, and Adrian Sandu. Multivariate predictions of local reduced-order-model errors and dimensions. *International Journal for Numerical Methods in Engineering*, 113(3):512–533, 2018.

[39] Razvan Stefanescu, Azam Moosavi, and Adrian Sandu. Parametric domain decomposition for accurate reduced order models: Applications of MP-LROM methodology. *Journal of Computational and Applied Mathematics*, 340:629–644, 2018.

[40] Natalia M Alexandrov, Robert Michael Lewis, Clyde R Gumbert, Lawrence L Green, and Perry A Newman. Approximation and model management in aerodynamic optimization with variable-fidelity models. *Journal of Aircraft*, 38(6):1093–1101, 2001.

[41] Martin Drohmann and Kevin Carlberg. The ROMES method for statistical modeling of reduced-order-model error. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):116–145, 2015.

[42] C.E. Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[43] Jin Yuan, Kesheng Wang, Tao Yu, and Minglun Fang. Reliable multi-objective optimization of high-speed WEDM process based on Gaussian process regression. *International Journal of Machine Tools and Manufacture*, 48(1):47–60, 2008.

[44] Stefano De Marchi and Emma Perracchione. Lectures on radial basis functions. 2018.

[45] Matthew S Caywood, Daniel M Roberts, Jeffrey B Colombe, Hal S Greenwald, and Monica Z Weiland. Gaussian process regression for predictive but interpretable machine learning models: An example of predicting mental workload across tasks. *Frontiers in human neuroscience*, 10:647, 2017.

[46] Malte Kuss. *Gaussian process models for robust regression, classification, and reinforcement learning*. PhD thesis, Technische Universität, 2006.

[47] Joaquin QuiÃ±onero-Candela, Carl Edward Rasmussen, AnÃbal R Figueiras-Vidal, et al. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11(Jun):1865–1881, 2010.

[48] Ed Snelson. Tutorial: Gaussian process models for machine learning. *Gatsby Computational Neuroscience Unit, UCL*, 2006.

[49] David S Touretzky, Michael C Mozer, and Michael E Hasselmo. *Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference*, volume 8. Mit Press, 1996.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] S. Chaturantabut. Dimension reduction for unsteady nonlinear partial differential equations via empirical interpolation methods. Master's thesis, Rice University, 2008.

[52] C.C. Pain, M.D. Piggott, A.J.H. Goddard, F. Fang, G.J. Gorman, D.P. Marshall, M.D. Eaton, P.W. Power, and C.R.E. De Oliveira. Three-dimensional unstructured mesh ocean modelling. *Ocean Modelling*, 10(1-2):5–33, 2005.

[53] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[54] Franois Chollet. keras. https://github.com/fchollet/keras, 2015.

[55] Deborah M Greaves. Simulation of viscous water column collapse using adapting hierarchical grids. *International Journal for Numerical Methods in Fluids*, 50(6):693–711, 2006.

[56] ZQ Zhou, JO De Kat, and B Buchner. A nonlinear 3-D approach to simulate green water dynamics on deck. In *Proceedings of the 7th International Conference on Numerical Ship Hydrodynamics*, page 5. Nantes, 1999.

[57] Carl Jacquemyn, Matthew D Jackson, and Gary J Hampson. Surface-based geological reservoir modelling using grid-free nurbs curves and surfaces. *Mathematical Geosciences*, pages 1–28, 2018.

[58] Matthew Jackson, James Percival, Peyman Mostaghimi, Brendan Tollit, Dimitrios Pavlidis, Christopher Pain, Jefferson Gomes, Ahmed H Elsheikh, Pablo Salinas, Ann Muggeridge, et al. Reservoir modeling for flow simulation by use of surfaces, adaptive unstructured meshes, and an overlapping-control-volume finite-element method. *SPE Reservoir Evaluation & Engineering*, 18(02):115–132, 2015.