Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem

Qian Wang[a,*], Jan S. Hesthaven[a], Deep Ray[a]

[a]*Chair of Computational Mathematics and Simulation Science, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland*

Abstract

A non-intrusive reduced-basis (RB) method is proposed for parametrized unsteady flows. A set of reduced basis functions are extracted from a collection of high-fidelity solutions via a proper orthogonal decomposition (POD), and the coefficients of the reduced basis functions are recovered by a feedforward neural network (NN). As a regression model of the RB method for unsteady flows, the neural network approximates the map between the time/parameter value and the projection coefficients of the high-fidelity solution onto the reduced space. The generation of the reduced basis and the training of the NN are accomplished in the offline stage, thus the RB solution of a new time/parameter value can be recovered via direct outputs of the NN in the online stage. Due to its non-intrusive nature, the proposed RB method, referred as the POD-NN, fully decouples the online stage and the high-fidelity scheme, and is thus able to provide fast and reliable solutions of complex unsteady flows. To test this assertion, the POD-NN method is applied to the reduced order modeling (ROM) of the quasi-one dimensional Continuously Variable Resonance Combustor (CVRC) flow. Numerical results demonstrate the efficiency and robustness of the POD-NN method.

Keywords: non-intrusive reduced basis method, unsteady flow, proper orthogonal decomposition, feedforward neural network, combustion

1. Introduction

Computational fluid dynamics (CFD) is one of the few available tools to investigate complex flow problems of scientific interest or of industrial value. However, obtaining *high-fidelity* solutions of complex flow problems is often expensive in terms of both CPU time and memory demands, since the accurate solution of the discretized Euler/Navier-Stokes equations requires a large amount of degree of freedoms (DOFs). Therefore, *reduced order modeling* (ROM) which aims at building low-dimensional and efficient models that are fast to solve while being able to accurately approximate the underlying high-fidelity solutions, fills a critical need in problems of design, control, optimization and uncertainty quantification, all of which require repeated model evaluations over a potentially large range of parameter values [1]. These parameters characterize fluid properties, source terms, geometry, initial and boundary conditions of the flow.

Reduced basis (RB) methods [1–3] are a class of well-known and widely-used ROM techniques, which are generally implemented in an offline-online paradigm [4]. In the *offline* stage, a set of RB functions are extracted from a collection of high-fidelity solutions, which are also called *snapshots*. The *reduced space*, spanned by the RB functions, represents the main dynamics of the full-order

[*]Corresponding author.

Email addresses: qian.wang@epfl.ch (Qian Wang), Jan.Hesthaven@epfl.ch (Jan S. Hesthaven), deep.ray@epfl.ch (Deep Ray) model. The two most popular methods to construct the reduced basis are the *proper orthogonal decomposition* (POD) [5–7] and the *greedy algorithm* [8–10]. The POD extracts the reduced basis by a singular value decomposition (SVD) of the *snapshot matrix*, i.e., the collection of high-fidelity solutions, obtained by a deterministic or random sampling in parameter space. On the contrary, a greedy algorithm carefully selects a set of snapshots as the reduced basis, according to some optimal criterion. For unsteady problems, the POD-greedy sampling algorithm [3, 11], combining POD in time with a greedy approach in parameter space, is widely used to construct the reduced basis. However, it should be noted that the greedy/POD-greedy method is not feasible for problems without a natural criterion for the selection of snapshots [12].

During the *online* stage, the expansion coefficients, also called the *reduced coefficients*, of the RB functions are determined. Based on the methodology of computing the reduced coefficients, RB methods are classified into two categories: *intrusive* and *non-intrusive* RB methods. The intrusive RB methods determine the reduced coefficients by solving a reduced order model, i.e., a projection of the full-order model onto the reduced space. The Galerkin procedure is the most popular choice for the projection [13, 14]. For problems with an affine dependence on the parameters, or with a non-affine dependence on the parameters but for which one may recover an affine expansion of the differential operator through the empirical interpolation method (EIM) [15] or discrete empirical interpolation method (DEIM) [16–18], the evaluation of the projection-based reduced order model is independent on the number of DOFs of the high-fidelity solution, and thus much faster than the full-order model [19, 20]. However, EIM and DEIM are non-trivial for general nonlinear problems with a non-affine dependence on the parameters, leading to limited computational gain by projection-based RB model with respect to the full-order model. Furthermore, the projection-based RB methods can suffer from instabilities [12, 21, 22].

Non-intrusive RB methods approximate the reduced coefficients over the parameter domain by using a database of reduced order information [23]. The high-fidelity solver is used to generate snapshots, leading to a full decoupling of the online stage and the high-fidelity scheme. Although being a natural choice, the use of standard interpolation techniques may fail if only a small number of samples [24, 25] are available. Regression-based non-intrusive RB methods [26, 27] have recently been developed for steady-state problems. In these methods, the reduced coefficients are determined online by rapid evaluations of *regression models*. The regression models are approximate maps between the parameter value and the projection coefficients of the underlying high-fidelity solution onto the reduced space, and are trained by high-fidelity data in a *supervised learning* paradigm [28] in the offline stage. In [26], artificial neural networks (ANNs) are utilized as the regression models for the non-intrusive ROM of the nonlinear Poisson and steady-state incompressible Navier-Stokes equations. In [27], Gaussian processes are utilized as the regression models for nonlinear structural analysis.

In this paper, the non-intrusive RB method using artificial neural networks, proposed in [26] and referred as POD-NN, is extended to *time-dependent* nonlinear problems, in particular the *unsteady flows*. Two important issues need to be addressed with regards to the extension of the non-intrusive RB method from steady-state to unsteady problems. Firstly, for problems with many time steps, the snapshot matrix of the direct POD method becomes very large, which leads to an expensive SVD. To overcome this difficulty, we use a two-step POD algorithm, in which i) the *time-trajectory* of each parameter value is compressed using POD, and ii) the reduced basis is extracted from the collection of the compressed time-trajectories using POD. By performing the POD in time and parameter space separately, the dimensions of the matrices requiring SVD are significantly reduced.

The second issue arises from the *unsteadiness*, which requires the regression model to be able to predict the RB solution at arbitrary time and physical/geometrical parameters within the given time and parameter domains. This issue is addressed by treating the time coordinate as another parameter. In other words, the input of the ANN consists of the temporal/physical/geometrical parameters, while the output is the set of the reduced coefficients.

The POD-NN method is applied to the quasi-1D Continuously Variable Resonance Combustor (CVRC) flow, to test its efficiency and robustness. CVRC is a model rocket combustor designed and operated at Purdue University (Indiana, U.S.) for investigating combustion instabilities [29]. This setup is called Continuously Variable Resonance Combustor (CVRC) because the length of the oxidizer injector can be varied continuously, allowing for a detailed investigation of the coupling between acoustics and combustion in the chamber [30]. The CVRC has been extensively investigated, including experiments [31–33] and 2D/3D detailed simulations [30, 34–36]. As the 2D/3D high-fidelity simulations are expensive, the quasi-1D model based on simplified assumptions for the flow modeling while still capable of remaining the dominating features, has been proposed by Smith et al. [37], and further developed by Frezzotti et al. [38–40]. The ROM of the CVRC, in particular the POD-Galerkin method for the quasi-1D model [41, 42], has also been studied. However, POD-Galerkin method for the quasi-1D CVRC model can suffer from instabilities [41]. In this paper, we apply the POD-NN method to build a fast, accurate and robust reduced order model of the quasi-1D CVRC flow.

In this paper, three artificial neural networks are constructed and trained, one for each of the following objectives:

- the approximation of the RB coefficients for the global ROM of the quasi-1D CVRC flow ;
- the approximation of the RB coefficients for the ROM localized at the combustion chamber;
- the regression of the power spectral density (PSD) for the stability map validation.

The remainder of this paper is organized as follows. Section 2 presents the framework of the non-intrusive POD-NN RB method for unsteady flow problems, including the generation of the reduced basis and the regression of the reduced coefficients. Section 3 presents the construction and training of the feedforward neural networks. Section 4 introduces the quasi-1D CVRC model and the high-fidelity solver. Section 5 presents the numerical results, and Section 6 gathers the relevant conclusions.

2. A non-intrusive reduced basis method using neural networks

The parametrized Euler/Navier-Stokes equations, governing unsteady flows, can be expressed in the following conservative form:

$$\frac{\partial \mathbf{u}\left(\mathbf{x}, t ; \boldsymbol{\mu}\right)}{\partial t}+\nabla \cdot \mathbf{f}\left[\mathbf{u}\left(\mathbf{x}, t ; \boldsymbol{\mu}\right) ; \boldsymbol{\mu}\right]=\mathbf{s}\left[\mathbf{x}, t, \mathbf{u}\left(\mathbf{x}, t ; \boldsymbol{\mu}\right) ; \boldsymbol{\mu}\right], \quad \forall(\mathbf{x}, t, \boldsymbol{\mu}) \in \Omega \times \mathcal{T} \times \mathcal{P}, \tag{1}$$

where $\mathbf{u}$, $\mathbf{f}$, $\mathbf{s}$ and $\boldsymbol{\mu}$ are the vectors of conserved variables, flux, source term and parameters, respectively. $\Omega \subset \mathbb{R}^3$, $\mathcal{T} \subset \mathbb{R}^+$ and $\mathcal{P} \subset \mathbb{R}^d$ are the space, time and parameter domains, respectively, where d is the number of parameters.

2.1. Generation of reduced basis

Given a parameter sampling $\mathcal{P}_h = \left\{\boldsymbol{\mu}^{(1)}, \cdots, \boldsymbol{\mu}^{(N)}\right\} \subset \mathcal{P}$, a collection of high-fidelity solutions of (1) can be obtained by running a solver with different parameter values in $\mathcal{P}_h$. The high-fidelity solution of parameter $\boldsymbol{\mu}^{(k)}$ at time step $j \in [1, \cdots, N_t]$ is denoted as $\mathbf{u}_h(t_j; \boldsymbol{\mu}_k) \in \mathbb{R}^{N_h}$, where N_h is the number of DOFs. The time-trajectory matrix that collects the high-fidelity solutions of parameter $\boldsymbol{\mu}^{(k)}$ at all the time steps is

$$\mathbb{S}_k = [\mathbf{u}_h\left(t_1; \boldsymbol{\mu}^{(k)}\right) | \cdots | \mathbf{u}_h\left(t_{N_t}; \boldsymbol{\mu}^{(k)}\right)] \in \mathbb{R}^{N_h \times N_t},$$

and the snapshot matrix that collects all the high-fidelity solutions is

$$\mathbb{S} = [\mathbb{S}_1 | \cdots | \mathbb{S}_N] \in \mathbb{R}^{N_h \times N_t N}.$$

Given the high-fidelity solutions, we seek to find a parameter-independent reduced basis $\{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_L\} \subset \mathbb{R}^{N_h}$ $(L \ll N_h)$, to construct a linear space that provides a low-rank approximation of the high-fidelity solutions. The reduced space, spanned by the reduced basis functions, is

$$V_{rb} = span\,\{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_L\}\,.$$

A reduced basis solution $\mathbf{u}_L\,(t;\boldsymbol{\mu})$ is an approximation of the high-fidelity solution $\mathbf{u}_h(t;\boldsymbol{\mu})$ in V_{rb} and can be expressed as

$$\mathbf{u}_L\,(t;\boldsymbol{\mu}) = \sum_{l=1}^{L} u_{rb}^{(l)}\,(t;\boldsymbol{\mu})\,\boldsymbol{\psi}_l \in V_{rb}, \tag{2}$$

where $\mathbf{u}_{rb}\,(t;\boldsymbol{\mu}) = [u_{rb}^{(1)}\,(t;\boldsymbol{\mu})\,, \cdots, u_{rb}^{(L)}\,(t;\boldsymbol{\mu})]^\top \in \mathbb{R}^L$ are the reduced coefficients for the expansion of the RB solution in the reduced basis functions. If we introduce the matrix

$$\mathbb{V} = [\boldsymbol{\psi}_1|\cdots|\boldsymbol{\psi}_L] \in \mathbb{R}^{N_h \times L},$$

the reduced basis solution in (2) can be expressed as

$$\mathbf{u}_L\,(t;\boldsymbol{\mu}) = \mathbb{V}\mathbf{u}_{rb}\,(t;\boldsymbol{\mu})\,.$$

The reduced basis can be extracted directly via POD, based on the SVD of the snapshot matrix, i.e.,

$$\mathbb{S} = \mathbb{W}\begin{bmatrix}\mathbb{D} & 0\\ 0 & 0\end{bmatrix}\mathbb{Z}^\top, \tag{3}$$

where $\mathbb{W} = [\mathbf{w}_1|\cdots|\mathbf{w}_{N_h}] \in \mathbb{R}^{N_h \times N_h}$ and $\mathbb{Z} = [\mathbf{z}_1|\cdots|\mathbf{z}_{N_t N}] \in \mathbb{R}^{N_t N \times N_t N}$ are orthogonal matrices, $\mathbb{D} = diag(\sigma_1, \cdots, \sigma_r) \in \mathbb{R}^{r\times r}$ is a diagonal matrix, with *singular values* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Here, r is the number of non-zero singular values and $r \leq min\,\{N_h, N_t N\}$. The columns of $\mathbb{W}$ are called *left singular vectors* of $\mathbb{S}$, and the columns of $\mathbb{Z}$ are called *right singular vectors* of $\mathbb{S}$.

At the algebraic level, our goal is to find $L \ll r$ orthonormal vectors $\{\tilde{\mathbf{w}}_1, \cdots, \tilde{\mathbf{w}}_L\}$ that approximate each column $\mathbf{s}_n, n = 1, \cdots, N_t N$ of $\mathbb{S}$ by the projection

$$\sum_{l=1}^{L}(\mathbf{s}_n, \tilde{\mathbf{w}}_l)\tilde{\mathbf{w}}_l.$$

According to Schmidt-Eckart-Young theorem [43, 44], the *POD basis* of rank L consisting of the first L left singular vectors of $\mathbb{S}$, minimizes the projection error defined by

$$\epsilon\,(\tilde{\mathbf{w}}_1, \cdots, \tilde{\mathbf{w}}_L) = \sum_{n=1}^{N_t N} \parallel \mathbf{s}_n - \sum_{l=1}^{L}(\mathbf{s}_n, \tilde{\mathbf{w}}_n)\,\tilde{\mathbf{w}}_l \parallel_{\mathbb{R}^{N_h}}^2,$$

among all the orthonormal bases $\{\tilde{\mathbf{w}}_1, \cdots, \tilde{\mathbf{w}}_L\}$ of $\mathbb{R}^L$. Therefore, $\{\mathbf{w}_1, \cdots, \mathbf{w}_L\}$ is taken as the reduced basis, i.e., $\boldsymbol{\psi}_l = \mathbf{w}_l$ for all $l = 1, \cdots, L$.

The dimension L of the basis is determined by the criterion:

$$\frac{\sum_{l=L+1}^{r}\sigma_l^2}{\sum_{l=1}^{r}\sigma_l^2} \leq \varepsilon, \tag{4}$$

where ε is the relative error tolerance used to control the accuracy of the POD.

The POD algorithm described by (3)-(4) can not be directly applied to unsteady problems with a large number of time steps, because the second dimension $N_t N$ of the snapshot matrix is large, making the SVD of the snapshot matrix very expensive.

To overcome this, we use a two-step POD algorithm, in which the POD in time and parameter space are performed separately, resulting in a significant reduction of the dimension of the snapshot matrix. The two-step POD algorithm is as follows:

(1) The time-trajectory of each parameter value is compressed via POD. With a relative error tolerance ε_t, for parameter $\boldsymbol{\mu}^{(k)}$ $(k = 1, \cdots, N)$, L_k basis functions are obtained via the POD of the time-trajectory, i.e., $\mathbb{T}_k = [\boldsymbol{\zeta}_1^{\boldsymbol{\mu}^{(k)}} | \cdots | \boldsymbol{\zeta}_{L_k}^{\boldsymbol{\mu}^{(k)}}] = \text{POD}\left([\mathbf{u}_h\left(t_1; \boldsymbol{\mu}^{(k)}\right), \cdots, \mathbf{u}_h\left(t_{N_t}; \boldsymbol{\mu}^{(k)}\right)], \varepsilon_t\right)$.

(2) A reduced basis is extracted by POD of the compressed time-trajectories. With a relative error tolerance $\varepsilon_{\boldsymbol{\mu}}$, L reduced basis functions $\{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_L\}$ are obtained via POD of the collection of the compressed time-trajectories, i.e., $\mathbb{V} = [\boldsymbol{\psi}_1 | \cdots | \boldsymbol{\psi}_L] = \text{POD}([\mathbb{T}_1 | \cdots | \mathbb{T}_N], \varepsilon_{\boldsymbol{\mu}})$.

The POD and the two-step POD algorithms are described in Algorithm 1.

Algorithm 1 Reduced basis generation of unsteady flow problems.

```
 1: function $\mathbb{V}$=POD($\mathbb{S}, \varepsilon$)
 2:     Perform SVD: $[\mathbb{W}, \mathbb{D}, \mathbb{Z}] = svd(\mathbb{S})$
 3:     Determine the truncation order $L$ using (4)
 4:     Set the basis functions: $\boldsymbol{\psi}_l = \mathbf{w}_l$, $l = 1, \cdots, L$
 5:     Assemble $\mathbb{V} = [\boldsymbol{\psi}_1 | \cdots | \boldsymbol{\psi}_L]$
 6: end function

 7: function $\mathbb{V}$=POD_TWO-STEP($\mathbb{S}, \varepsilon_t, \varepsilon_{\boldsymbol{\mu}}$)
 8:     Compress time-trajectory of each parameter value: $\mathbb{T}_k = [\boldsymbol{\zeta}_1^{\boldsymbol{\mu}^{(k)}} | \cdots | \boldsymbol{\zeta}_{L_k}^{\boldsymbol{\mu}^{(k)}}] = \text{POD}$
    $([\mathbf{u}_h(t_1; \boldsymbol{\mu}^{(k)}) | \cdots | \mathbf{u}_h(t_{N_t}; \boldsymbol{\mu}^{(k)})], \varepsilon_t)$, for $k = 1, \cdots, N$.
 9:     Extract reduced basis functions: $\mathbb{V} = [\boldsymbol{\psi}_1 | \cdots | \boldsymbol{\psi}_L] = \text{POD}([\mathbb{T}_1 | \cdots | \mathbb{T}_N], \varepsilon_{\boldsymbol{\mu}})$.
10: end function
```

2.2. Regression of reduced coefficients

In a non-intrusive RB method, the RB solution of a new parameter value is sought online by evaluating a regression model. For unsteady flows, the inputs of the regression model are the time and parameters $(t, \boldsymbol{\mu})$, and the outputs are the reduced coefficients $\mathbf{u}_{rb}(t; \boldsymbol{\mu}) = [u_{rb}^{(1)}(t; \boldsymbol{\mu}), \cdots, u_{rb}^{(L)}(t; \boldsymbol{\mu})]^\top$.

The "best" reduced basis solution for a certain time/parameter value $(t, \boldsymbol{\mu})$, understood as the best approximation of the high-fidelity solution $\mathbf{u}_h(t; \boldsymbol{\mu})$ in the reduced space V_{rb}, is the projection of $\mathbf{u}_h(t; \boldsymbol{\mu})$ onto V_{rb}, i.e.,

$$\mathbf{u}_h^{\mathbb{V}}(t; \boldsymbol{\mu}) = \mathbb{V}\mathbb{V}^\top \mathbf{u}_h(t; \boldsymbol{\mu}).$$

The corresponding reduced coefficients are the projection coefficients $\mathbb{V}^\top \mathbf{u}_h(t; \boldsymbol{\mu})$. An ideal regression $\boldsymbol{\pi}$ maps the input vector $(t, \boldsymbol{\mu})$ to the projection coefficients $\mathbb{V}^\top \mathbf{u}_h(t; \boldsymbol{\mu})$, i.e.,

$$\begin{aligned} \boldsymbol{\pi} : \mathcal{T} \times \mathcal{P} \subset \mathbb{R}^{d+1} &\to \mathbb{R}^L \\ (t, \boldsymbol{\mu}) &\to \mathbb{V}^\top \mathbf{u}_h(t; \boldsymbol{\mu}). \end{aligned}$$

In this paper, we construct and train an artificial neural network (ANN) as the regression model, acting as an approximation $\hat{\boldsymbol{\pi}}_{NN}$ of the map $\boldsymbol{\pi}$. Given a new time/parameter instance $(t, \boldsymbol{\mu})$, the associated RB solution is recovered by the evaluation of $\hat{\boldsymbol{\pi}}_{NN}$ at $(t, \boldsymbol{\mu})$, i.e.,

$$\mathbf{u}_{rb}^{NN}(t, \boldsymbol{\mu}) = \hat{\boldsymbol{\pi}}_{NN}(t, \boldsymbol{\mu}),$$

and, consequently,

$$\mathbf{u}_L^{NN}(t; \boldsymbol{\mu}) = \mathbb{V}\hat{\boldsymbol{\pi}}_{NN}(t, \boldsymbol{\mu}).$$

The construction and training of the ANN, will be presented in the Section 3.

2.3. A non-intrusive reduced basis method for unsteady flows using neural networks: POD-NN

The POD-NN method is implemented pursuing an offline-online paradigm [4]. The offline stage includes the generation of the reduced basis and the construction and training of the ANN. The online stage includes the evaluation of the ANN and a linear combination of the reduced basis functions. The implementation of the offline and online stages of the POD-NN method is described in Algorithm 2.

Algorithm 2 POD-NN RB method for unsteady flow problems.

```
 1: function $[\mathbb{V}, \hat{\boldsymbol{\pi}}]$=POD-NN_OFFLINE$(\mathcal{P}, \Omega, N)$
 2:     Generate the parameter set $\mathcal{P}_h = \{\boldsymbol{\mu}^{(1)}, \cdots, \boldsymbol{\mu}^{(N)}\} \subset \mathcal{P}$
 3:     Compute the high-fidelity solutions $\mathbf{u}_h(t_j; \boldsymbol{\mu}_k), j = 1, \cdots, N_t, k = 1, \cdots, N$
 4:     Generate the reduced basis $\{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_L\}$ using POD
 5:     Assemble $\mathbb{V} = [\boldsymbol{\psi}_1 | \cdots | \boldsymbol{\psi}_L]$
 6:     Construct and train an ANN as a regression model $\hat{\boldsymbol{\pi}}_{NN}$
 7: end function

 8: function $[\mathbf{u}_L^{NN}(t, \boldsymbol{\mu})]$=POD-NN_ONLINE$((t, \boldsymbol{\mu}), \mathbb{V}, \hat{\boldsymbol{\pi}}_{NN})$
 9:     Evaluate the output $\mathbf{u}_{rb}^{NN}(t, \boldsymbol{\mu}) = \hat{\boldsymbol{\pi}}_{NN}(t, \boldsymbol{\mu})$ of the ANN for the input vector $(t, \boldsymbol{\mu})$
10:     Compute the RB solution $\mathbf{u}_L^{NN}(t, \boldsymbol{\mu}) = \mathbb{V}\mathbf{u}_{rb}^{NN}(t, \boldsymbol{\mu})$
11: end function
```

In Algorithm 2, we observe that the offline and the online stages of the POD-NN method are decoupled. The online computation of the RB solution is independent of the high-fidelity scheme, enabling the POD-NN method of computing fast and reliable solutions of general nonlinear problems.

3. Artificial neural networks

Inspired by biological nervous system, an *artificial neural network* (ANN) is a computational model able to learn from observational data. An ANN consists of a collection of *artificial neurons*, and a set of *weighted synaptic connections* between the neurons. Data travels from the input neurons, following the direction imposed by the synapses, towards the output neurons. A *training* process of a ANN is to adjust the weights of the synapses and the thresholds of the neurons to configure the ANN for a specific application.

In this section, we will introduce the structure and training of the ANN as a regression model of the POD-NN method.

3.1. Artificial neuron

The basic processing unit of the ANN is the artificial neuron that serves to receive/send signals from/to other neurons in the network. The working mechanism of an artificial neuron is illustrated in Fig. 1. The neuron shown in Fig. 1 is the neuron j of an ANN. Neuron j receives signals from m sending neurons $s_k, k = 1, \cdots, m$, and sends signals to n target neurons $r_l, l = 1, \cdots, n$. The input signals are converted into the output signal using three functions: the propagation function f_{prop}, the activation function f_{act}, and the output function f_{out}. The *propagation function* is used to convert the vectorial input $\mathbf{p} = [y_{s_1}, \cdots, y_{s_m}]^\top$ into a scalar u_j that is called *net input*, i.e.,

$$u_j = f_{prop}(w_{s_1,j}, \cdots, w_{s_m,j}, y_{s_1}, \cdots, y_{s_m}).$$

A common choice for the propagation function is the weighted summation

$$f_{prop}(w_{s_1,j}, \cdots, w_{s_m,j}, y_{s_1}, \cdots, y_{s_m}) = \sum_{l=1}^{m} w_{s_l,j} y_{s_l}.$$

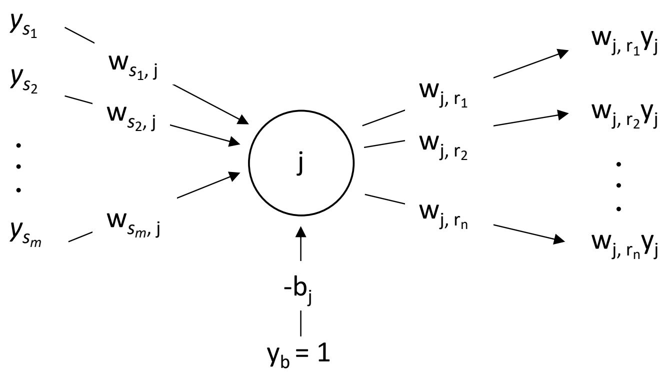


Figure 1: *A single artificial neuron receiving signals from m neurons and sending signals to n neurons.*

The *activation function* is used to decide whether the neuron j is *active*, given the net input u_j and a *threshold* $b_j \in \mathbb{R}$. The activation state a_j is computed by

$$a_j = f_{act}(u_j - b_j) = f_{act}(\sum_{l=1}^{m} w_{s_l,j} y_{s_l} - b_j)$$

The threshold b_j is a parameter of the network and can be adjusted during the training process, similar to the synaptic weights. To ease run-time access of b_j, a common practice is to introduce a *bias neuron* into the network. A bias neuron is a neuron that has a constant output $y_b = 1$, and directly connects to neuron j with a *bias weight* $w_{b,j} = -b_j$, as shown in Fig. 1.

There are various choices for the activation function [45]. In this paper, the activation function is the hyperbolic tangent function

$$f_{act}(v) = tanh(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}.$$

Finally, the *output function* f_{out} is used to calculate the scalar output $y_j \in \mathbb{R}$ of the neuron j based on the activation state a_j, i.e.,

$$y_j = f_{out}(a_j).$$

It is common to set the output function as the identity function, i.e., $y_j = f_{out}(a_j) = a_j$. The output y_j is then sent to the target neurons $r_l, l = 1, \cdots, n$.

Note that, the artificial neuron introduced above is the *computing neuron*. In the feedforward network presented in the following subsection, the neurons in the input layer are *source neurons*, which only provide the network with the input vector without performing any computation. The neurons in the output layer are computing neurons, of which the outputs are the components of the overall output vector of the network. The activation function for neurons in the output layer is often set to be the identity function, and the output function is usually set according to the range of the teaching input, also called the desired output. For example, the desired outputs of the ANNs in Subsection 5.2 and 5.3 are the reduced coefficients $\mathbf{u}_{rb}^{NN} \in \mathbb{R}^L$, the output function is set as the identity function; while, the desired output of the ANN in Subsection 5.5 is the power spectral density (PSD) that is positive, thus the output function is set as $f_{out}(v) = e^v$.

3.2. Feedforward neural network

One of the most important issues for the design of an ANN is the structure, or the topology of the network. Several network architectures have been proposed, among which the *feedforward neural network* [45, 46], also called *perceptron*, has been preferred in function regression tasks.

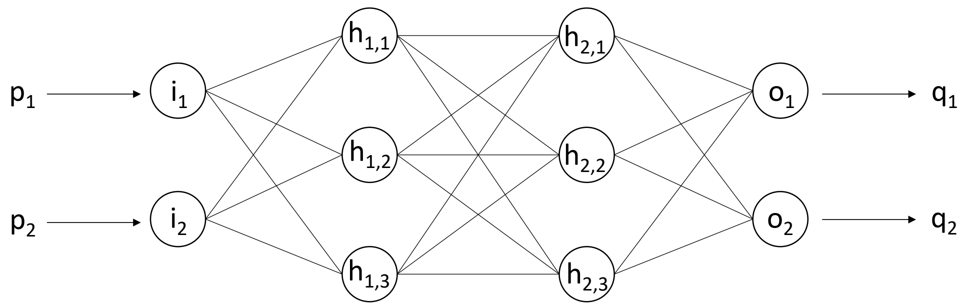


Figure 2: *A feedforward neural network with two hidden layers, two input and two output neurons. Each hidden layer consists of three neurons.*

In a feedforward neural network, neurons are arranged into *layers*. The first layer is called the *input layer*, with M_I source neurons, while the last layer is called the *output layer*, with M_O computing neurons. All the remaining K layers are called *hidden layers*, each one consisting of H_k computing neurons ($k = 1, \cdots, K$). There are $K + 2$ layers in total in the network. The neurons in a given layer receive signals from the preceding layer, and send signals to the succeeding layer. Furthermore, the neurons within the same layer do not communicate with each other. Inside the network, information travels from the input layer, across the hidden layer, towards the output layer. An activation pattern $\mathbf{p} \in \mathbb{R}^{M_I}$ is provided at the input layer, an output $\mathbf{q} \in \mathbb{R}^{M_O}$ is obtained at the output layer. A feedforward neural network establishes a map between the *input space* $\mathbb{R}^{M_I}$ and the *output space* $\mathbb{R}^{M_O}$. Therefore, the feedforward neural network architecture is particularly suitable for function approximation.

A sample feedforward neural network with two hidden layers is shown in Fig. 2. The network has $M_I = 2$ input neurons (denoted with the letter i) and $M_O = 2$ output neurons (denoted with the letter o). Each hidden layer has $H_1 = H_2 = 3$ neurons (denoted with the letter h). This feedforward neural network can be used to a function approximation task that maps an input $\mathbf{p} \in \mathbb{R}^2$ to an output $\mathbf{q} \in \mathbb{R}^2$.

3.3. Training of the feedforward neural network

The principle characteristic of a neural network is its capability of *learning* by encoding the acquired knowledge via its internal parameters, i.e., the synaptic and bias weights. The learning of the network is accomplished through a *training* process on data $\mathcal{D}_{tr}$, by adjusting the synaptic and bias weights according to some performance measure. Therefore, a training procedure is typically iterative. After the training, the network is able to provide reasonable prediction of unknown problems of the same class of the *training data set*, which is called *generalization*. The data used to test the generalization capability of a network during the training is called the *validation data set*, and is denoted by D_{va}.

There are three *learning paradigms*: supervised learning, unsupervised learning and reinforcement learning [45]. The choice of the learning paradigm is task-dependent. For function regression, *supervised learning* is the natural choice.

In supervised learning, a training data set $\mathcal{D}_{tr} = \{\mathbf{p}_i, \mathbf{t}_i\}_{1 \le i \le N_{tr}}$ and a validation data set $\mathcal{D}_{va} = \{\mathbf{p}_i, \mathbf{t}_i\}_{1 \le i \le N_{va}}$ is prepared. Here $\mathbf{p} \in \mathbb{R}^{M_I}$ is the *input pattern*, $\mathbf{t} \in \mathbb{R}^{M_O}$ is the *teaching input*. A component of the input pattern is called a *feature*. A *feature scaling* technique in which all the features are scaled to the same range, can be applied to the data sets to accelerate the training [47]. In this paper, a feature χ is scaled by the mean normalization

$$\tilde{\chi} = \frac{\chi - \bar{\chi}}{\chi_{max} - \chi_{min}},$$

where $\bar{\chi}$, χ_{max} and χ_{min} are the mean, maximum and minimum of χ, respectively.

The training aims at minimizing the error between the prediction and the truth, which can be accomplished by minimizing a *cost function*

$$\mathcal{C} := \mathcal{C}(\mathbf{q}, \mathbf{t}),$$

that measures the difference between the *network prediction* $\mathbf{q} \in \mathbb{R}^{M_O}$ and the *truth* $\mathbf{t} \in \mathbb{R}^{M_O}$. The synaptic and bias weights of the optimal network are obtained using a *stochastic optimization* algorithm, which uses *mini-batches* of size $N_b < N_{tr}$ from the training set $\mathcal{D}_{tr}$, to take a single optimization step by minimizing the cost function

$$\mathcal{C} = \frac{1}{N_b} \sum_{i=1}^{N_b} \parallel \mathbf{q}_i - \mathbf{t}_i \parallel^2_{\mathbb{R}^{M_O}} . \tag{5}$$

More precisely, the full training data set with N_{tr} data-points is shuffled, and N_{tr}/N_b mini-batches are extracted to take N_{tr}/N_b optimization steps. In the s-th optimization step, the weight $w_{i,j}$ for the connection between the neuron pair (i, j) is updated as

$$w^{s+1}_{i,j} = w^s_{i,j} + \Delta w_{i,j}, \ \ \Delta w_{i,j} = -\eta G(\frac{\partial \mathcal{C}}{\partial w_{i,j}}),$$

where η is the learning rate, G is a function that depends on the specific optimizer. Once the entire training data set is exhausted, the training is said to have completed one *training epoch*. Then the training set is reshuffled and the networks is optimized by another N_{tr}/N_b steps, to repeat the process for another epoch. The training is performed for a sufficient number of epochs to get a converged network. The *shuffling* introduces stochasticity to the training data set, leading to a faster convergence [48].

The training of the ANN is performed using the open-source machine learning library TensorFlow [49]. The *Adam stochastic optimizer* [50] is used to minimize the cost function. To accelerate the training, we use a *learning rate decay* as

$$\eta = \frac{\eta_0}{(1 + \theta * epoch)},$$

where η_0 and θ are hyperparameters to control the decay.

To avoid *overfiting* [45], a *regularization* term is introduced to the cost function in (5) to penalize the synaptic weights of the networks. The regularized cost function is

$$\tilde{\mathcal{C}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \parallel \mathbf{q}^{(i)} - \mathbf{t}^{(i)} \parallel^2_{\mathbb{R}^{M_O}} + \lambda \left[\sum_{m=1}^{M_I} \sum_{n=1}^{H_1} w^2_{i_n, h_{1,m}} + \sum_{l=1}^{K-1} \sum_{m=1}^{H_l} \sum_{n=1}^{H_{l+1}} w^2_{h_{l,m}, h_{l+1,n}} + \sum_{m=1}^{H_K} \sum_{n=1}^{M_O} w^2_{h_{K,m}, o_n} \right],$$

where λ is the parameter used to control the effect of regularization.

At the beginning of training, the synaptic and bias weights are randomly initialized using normal distributions [51]. Therefore, the training needs to be performed several times, following a *multiple restarts* approach [52], to prevent the training results from depending on the initialization of the weights. Ten restarts are performed for the training of each ANN in this paper, and the trained model with the best validation accuracy is selected as the final model. The accuracy of a network on a data set of size S can be measured by

$$Acc = -\frac{1}{S} \sum_{i=1}^{S} \parallel \mathbf{q}_i - \mathbf{t}_i \parallel^2_{\mathbb{R}^{M_O}} .$$

The training process of the ANN in this paper is described in Algorithm 3.

Algorithm 3 Training process of the network.

```
1: function $[\mathbf{w}_{rb}, tr\_acc_{rb}, va\_acc_{rb}]$=POD-NN_TRAINING($\mathbf{w}, N_{res}, N_{epo}, N_b, \eta_0, \theta$)
2:     for $r \leftarrow 1, N_{res}$ do                              ▷ Multiple restarts
3:         $\mathbf{w} \leftarrow init(\mathbf{w})$                  ▷ initialize the synaptic and bias weights $\mathbf{w}$
4:         for $epoch \leftarrow 1, N_{epo}$ do                      ▷ training epoch loop
5:             $\eta \leftarrow \eta_0/(1 + \theta * epoch)$          ▷ learning rate decay
6:             $\mathcal{D}_{tr} \leftarrow shuffle(\mathcal{D}_{tr})$  ▷ shuffle the training data set
7:             for $s \leftarrow 1, N_{tr}/N_b$ do                    ▷ mini-batch loop
8:                 $\mathcal{D}_{tr,s} \leftarrow \mathcal{D}_{tr}[(s-1) * N_b + 1 : s * N_b]$  ▷ the $s$-th mini-batch
9:                 $\Delta\mathbf{w} \leftarrow -\eta G_{Adam}(\frac{\partial \tilde{\mathcal{C}}}{\partial \mathbf{w}})$  ▷ Adam optimizer
10:                $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$  ▷ update the weights
11:            end for
12:        end for
13:        $tr\_acc \leftarrow Acc(\mathcal{D}_{tr})$                 ▷ evaluate training accuracy
14:        $va\_acc \leftarrow Acc(\mathcal{D}_{va})$                 ▷ evaluate validation accuracy
15:        if $r = 1$ or $va\_acc > va\_acc_{rb}$ then                ▷ if the validation accuracy is improved
16:            $\mathbf{w}_{rb} \leftarrow \mathbf{w}$                ▷ save the weights
17:            $tr\_acc_{rb} \leftarrow tr\_acc$                      ▷ save the training accuracy
18:            $va\_acc_{rb} \leftarrow va\_acc$                      ▷ save the validation accuracy
19:        end if
20:    end for
21: end function
```

3.4. Feedforward neural network as the regression model of the non-intrusive RB method

The inputs of the ANN are the time and parameters $(t, \boldsymbol{\mu})$, the outputs are the reduced coefficients $\mathbf{u}_{rb}(t;\boldsymbol{\mu}) = [u_{rb}^{(1)}(t;\boldsymbol{\mu}), \cdots, u_{rb}^{(L)}(t;\boldsymbol{\mu})]^\top$.

The training data $\mathcal{D}_{tr}$ and the validation data $\mathcal{D}_{va}$ are prepared before the training by projecting two collections of high-fidelity solutions onto the reduced space V_{rb}.

The structure of the ANN used is the feedforward neural network with several hidden layers. As the number of outputs is usually much larger than the number of inputs, the numbers of neurons of the layers increase from the input to the output layer, to increase the complexity of the ANN. The ANN of the POD-NN method is trained using Algorithm 3.

During the online phase, as described in Algorithm 2, the RB solution of a new time/parameter value can be computed by evaluating the ANN. The evaluation of the ANN primarily involves matrix-vector multiplications and is thus computationally efficient if suitably vectorized, making the ANN an efficient regression tool for the non-intrusive RB methods.

4. Quasi-1D Continuously Variable Resonance Combustor (CVRC) model

CVRC is a model rocket combustor designed and operated at Purdue University (Indiana, U.S.) to investigate *combustion instabilities* [29]. This setup is called the Continuously Variable Resonance Combustor (CVRC) because the length of the oxidizer injector can be varied continuously, allowing for a detailed investigation of the coupling between acoustics and combustion in the chamber [30]. The 2D/3D high-fidelity simulations of CVRC are expensive. Thus to get a fast analysis tool, a *quasi-1D model* has been proposed by Smith et al. [37] and further developed by Frezzotti et al. [38–40].

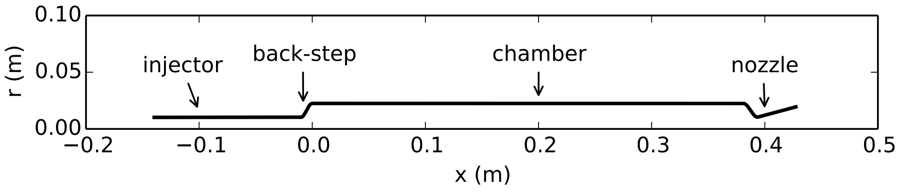


Figure 3: *Geometry of quasi-1D CVRC model.*

4.1. Quasi-1D CVRC setup

The CVRC consists of three parts: oxidizer post, combustion chamber and exit nozzle, as shown in Fig. 3. The oxidizer is injected from the left end of the oxidizer post and meets the fuel that is injected through an annular ring around the oxidizer injector, at the back-step. The combustion happens in a region around the back-step. The combustion products flow through the chamber and exit the system from the nozzle. Both the injector and the nozzle are operated at choked condition during the experiment. The length of the oxidizer post L_{op} of the CVRC can be varied continuously, leading to different behavior of the combustion stability. In this paper, we will focus on the case with $L_{op} = 14.0$ cm, in which the combustion is unstable.

The geometry parameters of the quasi-1D CVRC with a oxidizer post length $L_{op} = 14.0$ cm are shown in Table 1. The back-step and the converging part of the nozzle are sinusoidally contoured to avoid discontinuity of the radius that will invalidate the quasi-1D governing equations presented in the next subsection.

Table 1: *Geometry parameters of the quasi-1D CVRC with an oxidizer post length $L_{op} = 14$ cm.*

Section	Oxidizer post		Chamber	Nozzle	
	injector	back-step		converging part	diverging part
Length (cm)	12.99	1.01	38.1	1.27	3.4
Radius (cm)	1.02	$1.02 \sim 2.25$	2.25	$2.25 \sim 1.04$	$1.04 \sim 1.95$

The fuel is pure gaseous methane. The oxidizer is a mixture of 42% oxygen and 58% water (per unit mass). The oxidizer is injected in the oxidizer post at a temperature $T_{ox} = 1030$ K so that both water and oxygen are in the gaseous phase. The operating conditions are listed in Table 2.

Table 2: *CVRC operating conditions.*

Parameter	Unit	Value
Fuel mass flow rate, $\dot{m}_f$	kg/s	0.027
Fuel temperature, T_f	K	300
Oxidizer mass flow rate, $\dot{m}_{ox}$	kg/s	0.32
Oxidizer temperature, T_{ox}	K	1030
O_2 mass fraction in oxidizer, Y_{O_2}	–	42.4%
H_2O mass fraction in oxidizer, Y_{H_2O}	–	57.6%
Mean chamber pressure	MPa	1.34
Equivalence ratio, E_r	–	0.8

For the combustion, we consider the one-step reaction model

$$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$$

We assume that the fuel reacts instantaneously to form products, allowing us to neglect intermediate species and finite reaction rates. As the equivalence ratio is less than one, there is oxidizer left after the combustion. Therefore, only two species need to be considered: oxidizer and combustion products.

4.2. Full-order model

The governing equations that describe the conservation of mass, momentum, and energy of the quasi-1D CVRC flow, are the quasi-1D unsteady Euler equations for multiple species, expressed in conservative form as

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} = \mathbf{s}_A + \mathbf{s}_f + \mathbf{s}_q.$$

The conserved variable vector $\mathbf{u}$ and the convective flux vector $\mathbf{f}$ are

$$\mathbf{u} = \begin{pmatrix} \rho A \\ \rho u A \\ \rho E A \\ \rho Y_{ox} A \end{pmatrix}, \mathbf{f} = \begin{pmatrix} \rho u A \\ \left(\rho u^2 + p\right) A \\ \left(\rho E + p\right) u A \\ \rho u Y_{ox} A \end{pmatrix},$$

where ρ is the density, u is the velocity, p is the pressure, E is the total energy, Y_{ox} is the mass fraction of oxidizer, and $A = A(x)$ is the cross section area of the duct. The pressure p can be computed using the conserved variables as

$$E = \frac{p}{\rho(\gamma - 1)} + \frac{u^2}{2} - C_p T_{ref},$$

where T_{ref} is the reference temperature and is set as 298.15 K in this paper. The temperature T is recovered from the equation of state $p = \rho R T$. The gas properties C_p, R and γ are computed as $C_p = \sum C_{pi} Y_i$, $R = \sum R_i Y_i$ and $\gamma = C_p/(C_p - R)$, respectively.

The source terms are

$$\mathbf{s}_A = \begin{pmatrix} 0 \\ p\dfrac{dA}{dx} \\ 0 \\ 0 \end{pmatrix}, \mathbf{s}_f = \begin{pmatrix} \dot{\omega}_f \\ \dot{\omega}_f u \\ \dot{\omega}_f \left(h_0^f + \Delta h_0^{rel}\right) \\ \dot{\omega}_{ox} \end{pmatrix}, \mathbf{s}_q = \begin{pmatrix} 0 \\ 0 \\ q' \\ 0 \end{pmatrix},$$

where $\dot{\omega}_f$ is the depletion rate of the fuel, $\dot{\omega}_{ox}$ is the depletion rate of the oxidizer, h_0^f is the total enthalpy of the fuel, Δh_0^{rel} is the heat of reaction per unit mass of fuel and q' is the *unsteady heat release term*. $\mathbf{s}_A$ accounts for area variations, $\mathbf{s}_f$ and $\mathbf{s}_q$ are related to the combustion. $\mathbf{s}_f$ represents the addition of the fuel and its combustion with the oxidizer, which in turn results in the creation of the combustion products. The depletion rate of the fuel is

$$\dot{\omega}_f = \frac{k_f \dot{m}_f Y_{ox} \left(1 + sin\xi\right)}{l_f - l_s}, \tag{6}$$

where

$$\xi = -\frac{\pi}{2} + 2\pi \frac{x - l_s}{l_f - l_s}, \quad \forall \ l_s < x < l_f.$$

The setting of the fuel injection restricts the combustion to the region $l_s < x < l_f$. The reaction constant k_f is selected to insure that the fuel is consumed within the specified *combustion zone.* The depletion rate of the oxidizer is computed by

$$\dot{\omega}_{ox} = C_{o/f}\dot{\omega}_f,$$

where $C_{o/f}$ is the oxidizer-to-fuel ratio.

The unsteady heat release term q', also called the *combustion response function*, models the coupling between acoustics and combustion. In this paper, we use the combustion response function designed by Frezzotti et al. [39, 40], which is a function of the velocity, sampled at specific abscissa $\hat{x}$ that is almost coincident with the antinode of the first longitudinal modal shape, with a certain time lag τ, i.e.,

$$q'\left(x,t\right) = \alpha g\left(x\right) A\left(x\right)\left[u\left(\hat{x}, t-\tau\right) - \bar{u}\left(\hat{x}\right)\right]. \tag{7}$$

Here $\bar{u}$ is the time averaged velocity, estimated with the steady-state quasi-1D model assuming $q' = 0$, and $g(x)$ is a Gaussian distribution

$$g\left(x\right) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}},$$

where μ is the mean and σ is the standard deviation. The amount of heat release due to velocity oscillations is controlled by the parameter α.

The boundary conditions for the quasi-1D CVRC flow include the fixed mass flow rate and the stagnation temperature at the head-end of the oxidizer injector, and the supersonic outflow at the exit of the nozzle.

Prior to unsteady simulation, the quasi-1D CVRC needs to be excited, which can be achieved by adding a *perturbation* to the steady-state solution. The perturbation is added by forcing the mass flow rate with a multi-sine signal

$$\dot{m}_{ox}\left(t\right) = \dot{m}_{ox,0}\left[1 + \delta\sum_{k=1}^{K} sin\left(2\pi k\Delta f t\right)\right], \tag{8}$$

where $\dot{m}_{ox,0}$ is the oxidizer mass flow rate in Table 2, Δf is the frequency resolution and K is the number of frequencies. In this paper, $\Delta f = 50$ Hz and $K = 140$, resulting in a minimal frequency of 50 Hz and a maximal frequency of 7000 Hz. δ is required to be small to control the amplitude of the perturbation and is set as 0.1%.

The procedure of the unsteady simulation of the quasi-1D CVRC flow includes three steps:

(1) Compute the steady-state solution by setting $\dot{m}_{ox} = \dot{m}_{ox,0}$ and $q' = 0$.

(2) Excite the system by adding a perturbation to the oxidizer mass flow rate according to (8) and setting $q' = 0$.

(3) Perform the unsteady simulation by turning on the combustion response function q' in (7) and turning off the oxidizer mass flow rate perturbation by setting $\dot{m}_{ox} = \dot{m}_{ox,0}$.

4.3. High-fidelity solver

A high-fidelity quasi-1D CVRC flow solver is built by employing the MUSCL reconstruction [53], the Lax-Friedrichs flux [54] and the strong stability preserving, three-stage Runge-Kutta (SSP RK3) time stepping [55].

A difficulty in implementing the solver is that, the reaction constant k_f in (6), selected to consume all the fuel within the specified combustion zone, can not be computed without the steady-state distribution of Y_{ox}. Therefore, it is determined by trial and error in previous research [37, 39]. In this paper, an iteration algorithm for k_f in the steady-state simulation is designed to overcome this difficulty. In the j-th time step of the steady-state simulation, k_f is computed by

$$k_f^{(j)} = \frac{l_f - l_s}{\int_{l_s}^{l_f} Y_{ox}^{(j)} \left(1 + sin\xi\right) dx},$$

obtained by the condition that all fuel is consumed within the combustion zone $l_s < x < l_f$. This significantly improves the efficiency of the generation of high-fidelity solutions, since the trial and error work is avoided when evaluating the solver with large number of parameter values.

5. Numerical results

Numerical results including high-fidelity solver validation, reduced order modeling (ROM), and stability map validation of the quasi-1D CVRC flow, are presented in this section.

The following metrics are used to evaluate the accuracy of the results:

(1) the relative projection error,

$$\varepsilon_{\mathbb{V}}(t, \boldsymbol{\mu}) = \frac{\| \mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbb{V}\mathbb{V}^\top \mathbf{u}_h(t, \boldsymbol{\mu}) \|_{\mathbb{R}^{N_h}}}{\| \mathbf{u}_h(t, \boldsymbol{\mu}) \|_{\mathbb{R}^{N_h}}}$$

(2) the POD-NN relative error,

$$\varepsilon_{PODNN}(t, \boldsymbol{\mu}) = \frac{\| \mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbf{u}_L^{NN}(t, \boldsymbol{\mu}) \|_{\mathbb{R}^{N_h}}}{\| \mathbf{u}_h(t, \boldsymbol{\mu}) \|_{\mathbb{R}^{N_h}}} = \frac{\| \mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbb{V}\mathbf{u}_{rb}^{NN}(t, \boldsymbol{\mu}) \|_{\mathbb{R}^{N_h}}}{\| \mathbf{u}_h(t, \boldsymbol{\mu}) \|_{\mathbb{R}^{N_h}}}$$

In the numerical experiments, the accuracy of the reduced order models built by the POD-NN method, will be evaluated on *test data*. The average of the relative errors are used to measure the accuracy of the models. On test data $\mathcal{D}_{te}$ of size N_{te}, the average relative errors are defined as

$$\bar{\varepsilon}_{\mathbb{V}}(t, \boldsymbol{\mu}) = \frac{\sum_{(t,\boldsymbol{\mu}) \in \mathcal{T}_{te} \times \mathcal{P}_{te}} \varepsilon_{\mathbb{V}}(t, \boldsymbol{\mu})}{N_{te}}, \;\; \bar{\varepsilon}_{PODNN}(t, \boldsymbol{\mu}) = \frac{\sum_{(t,\boldsymbol{\mu}) \in \mathcal{T}_{te} \times \mathcal{P}_{te}} \varepsilon_{PODNN}(t, \boldsymbol{\mu})}{N_{te}},$$

where $\mathcal{T}_{te} \times \mathcal{P}_{te}$ is the time/parameter sampling of $\mathcal{D}_{te}$.

5.1. High-fidelity solver validation

In this subsection, both steady-state and unsteady solutions of the full-order quasi-1D model are presented and compared with the existing experimental and computational results to validate the high-fidelity solver presented in Subsection 4.3.

The simulations are performed on a uniform mesh with 1200 cells. In the steady-state simulation, the CFL number is set as 1. The time step for the converged steady-state simulation is 1.604×10^{-7} s. The computed steady-state pressure and temperature distributions are presented in Fig. 4. The computed mean chamber pressure is compared with the results of the experiment [30] and a reference solver [56] using the same number of cells in Table 3. The comparison show that the mean chamber pressure computed by the present solver is comparable to that of the reference solver.

The unsteady simulation is performed until $t = 1.2$ s using a time step $\Delta t = 10^{-7}$ s. The perturbation of the mass flow rate is added at $0 < t < 0.02$ s. The parameters of the combustion response function in (7) are set as $\alpha = 2.97$ MPa, $\tau = 0.475$ ms, $\hat{x} = 21.5$ cm, $\mu = 4$ cm, and $\sigma = 2.24$ cm. These values are extracted by Frezzotti et al. [39, 40] from the 2D detailed hybrid RANS/LES simulation results [34]. The computed pressure signal at $x = 36.8$ cm is shown in Fig. 5

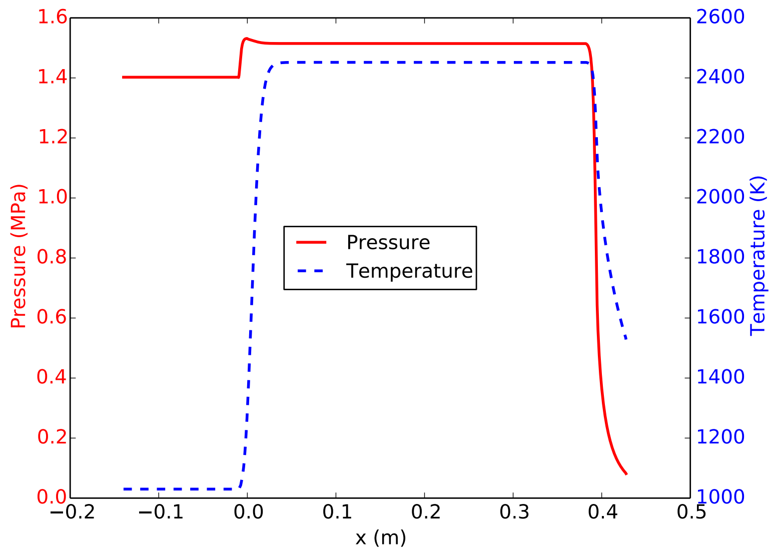


Figure 4: *Steady-state solution.*

Table 3: *Comparison of computational and experimental mean chamber pressure.*

Method	Present solver	Reference solver [56]	Experiment [30]
Mean chamber pressure (MPa)	1.515	1.531	1.34

to illustrate the unsteadiness of the quasi-1D CVRC flow. The choice of the location of the pressure signals is due to the availability of a pressure probe at this location in the experiment. A fast Fourier transformation (FFT) of the pressure signal is performed to compute the resonance frequencies and the corresponding amplitudes. Single-sided peak-to-peak amplitude spectrum of the pressure signal is shown in Fig. 6. The computed resonance frequencies and peak-to-peak amplitudes are listed and compared with the results of the experiment of [36] and the solver of [40] in Table 4. The comparison shows that the results of the present solver agree with the experiment and are comparable to that of the reference solver.

Table 4: *Comparison of computational and experimental resonance frequencies and peak-to-peak amplitudes.*

	Present solver			Reference solver [40]		Experiment [36]		
Mode	f (Hz)	p'_{ptp} (kPa)	f_i/f_1	f (Hz)	f_i/f_1	f (Hz)	p'_{ptp} (kPa)	f_i/f_1
1	1514	340.07	1.00	1546	1.00	1324	387.15	1.00
2	3029	30.97	2.00	3098	2.00	2655	89.29	2.01
3	4543	17.23	3.00	4646	3.01	3979	46.37	3.01

Based on the steady-state and unsteady results, we conclude that the solver, presented in Subsection 4.3, can compute high-fidelity solutions of the quasi-1D CVRC flow. The solver is used as the basis of the ROM of the quasi-1D CVRC flow in the rest of this section.

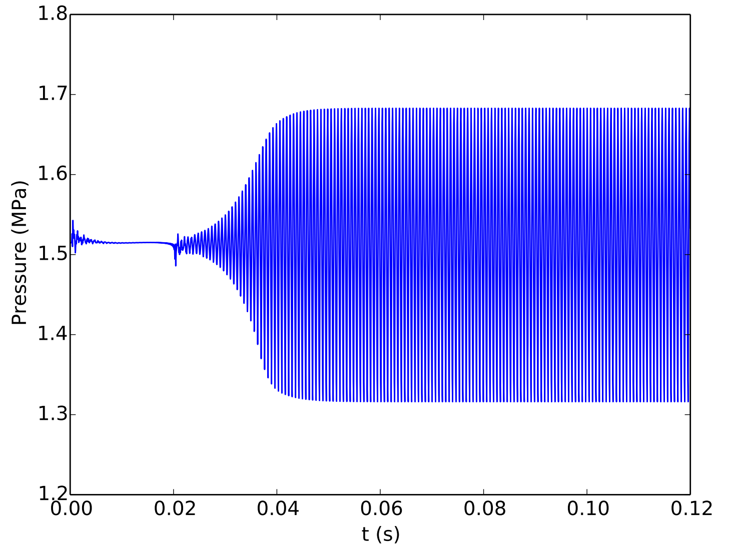


Figure 5: *Pressure signal at $x = 36.8$ cm.*

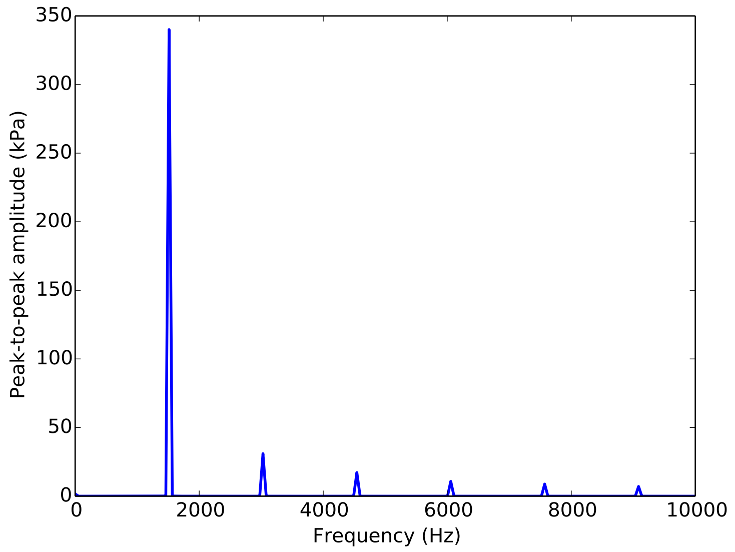


Figure 6: *Single-sided peak-to-peak amplitude spectrum of the pressure signal of Fig. 5.*

5.2. ROM of the CVRC

This subsection presents the results of the ROM of the quasi-1D CVRC pressure field using the POD-NN method.

The parameter space is set to involve $d = 4$ independent physical parameters $(T_f, T_{ox}, Y_{O_2}, E_r)$ of the operating conditions listed in Table 2. The range and the number of points of the time/parameter sampling is listed in Table 5. The number of the parameter points is $N = 3^4 = 81$. The number of the time points is $N_t = 1000$, which means that snapshots are obtained every 100 time steps within the time range $0.10 \leq t \leq 0.11$. The total number of snapshots is $N_t N = 81000$.

Table 5: *Time and parameter settings for snapshots of the ROM of the quasi-1D CVRC pressure field.*

		Parameters			
	t	T_f	T_{ox}	Y_{O_2}	E_r
Range	$0.10 \sim 0.11$	$290 \sim 310$	$1020 \sim 1040$	$0.414 \sim 0.434$	$0.79 \sim 0.81$
Points	1000, uniform	3, uniform	3, uniform	3, uniform	3, uniform

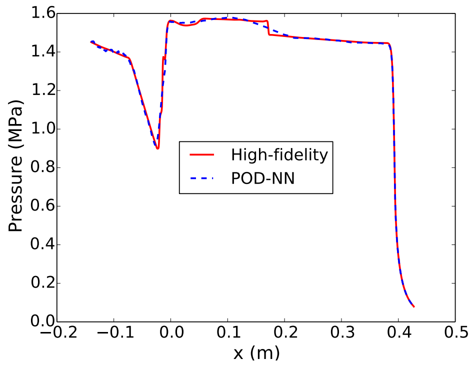
The reduced basis functions are extracted from the snapshots using the two-step POD algorithm presented in Subsection 2.1. The error bounds for the two-step POD are $\varepsilon_t = 5.0 \times 10^{-6}$ and $\varepsilon_{\boldsymbol{\mu}} = 5.0 \times 10^{-4}$, which results in $L = 206$ reduced basis functions and a projection error $\bar{\varepsilon}_{\mathbb{V}} = 0.083\%$, as shown in Table 8.

An ANN is constructed for the regression of the reduced coefficients. The inputs of the ANN are $(t; \boldsymbol{\mu}) = (t, T_f, T_{ox}, Y_{O_2}, E_r)$, the outputs are $\mathbf{u}_{rb}(t; \boldsymbol{\mu}) = [u_{rb}^{(1)}(t; \boldsymbol{\mu}), \cdots, u_{rb}^{(L)}(t; \boldsymbol{\mu})]^\top$. Therefore, the dimension of inputs is $d+1 = 5$, the dimension of outputs is $L = 206$. We construct a feedforward neural network with an input layer, an output layer, and five hidden layers. The number of neurons of each layer is listed in Table 6. For the hidden layers, the activation function is the hyperbolic tangent function, the output function is the identity function; for the output layer, both the activation and the output functions are the identity function. The training data $\mathcal{D}_{tr}$ and validation data $\mathcal{D}_{va}$ are prepared for the training of the ANN. The settings for the training and validation data are listed in Table 7. The training data is obtained from the 81 high-fidelity simulations that are used for the snapshots. The validation data is obtained from 24 high-fidelity simulations with randomly chosen parameters inside the parameter domain in Table 5. For the training, the mini-batch size is $N_b = 810$, the regularization parameter is $\lambda = 10^{-10}$, the number of training epochs is $N_{epo} = 6000$, and the learning rate decay is $\eta = 0.1/(1 + 0.5 * epoch)$. The ANN is trained with $N_{res} = 10$ restarts.

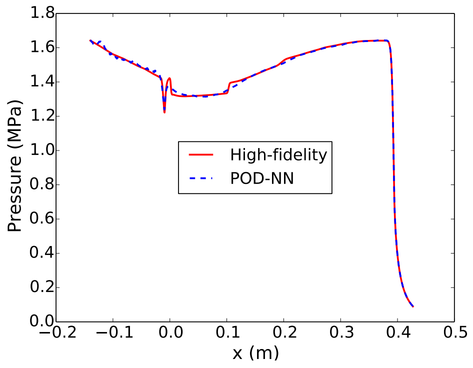
Table 6: *Architecture of the ANN used for the ROM of the quasi-1D CVRC pressure field.*

Layer	i	h_1	h_2	h_3	h_4	h_5	o
Number of neurons	5	50	55	60	65	70	206

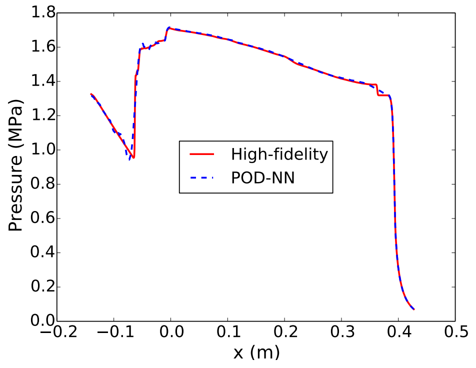
The reduced order model built by the POD-NN method is tested on a data set $\mathcal{D}_{te}$, obtained from 12 high-fidelity simulations with randomly chosen parameters, as listed in Table 7. The average projection error and the average POD-NN error are listed in Table 8. The comparison between the predicted and the high-fidelity solutions for 4 different time/parameter values is shown in Fig. 7. The results in Fig. 7 show that the predicted solutions agree with the high-fidelity solutions quite accurately.



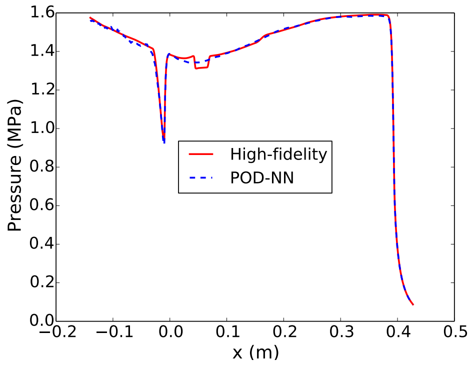
(a) $(t, \boldsymbol{\mu}) = (0.1081, 303.2893, 1032.1611, 0.4309, 0.7908)$



(b) $(t, \boldsymbol{\mu}) = (0.1006, 303.2893, 1035.8901, 0.4328, 0.7908)$



(c) $(t, \boldsymbol{\mu}) = (0.1075, 304.4604, 1035.8901, 0.4309, 0.7928)$



(d) $(t, \boldsymbol{\mu}) = (0.1015, 303.2893, 1035.8901, 0.4172, 0.8059)$

Figure 7: *Comparison between the predicted quasi-1D CVRC pressure fields by POD-NN method and the high-fidelity solutions.*

Table 7: *Settings for the training, validation, and test data sets.*

Data set	Training	Validation	Test
Parameter points	81, uniform	24, random	12, random
Time points	500, uniform	20, random	20, random
Size	40500	480	240

Table 8: *Test results of the POD-NN method for the ROM of the quasi-1D CVRC pressure field.*

Average projection error, $\bar{\varepsilon}_{\mathbb{V}}$	Average POD-NN error, $\bar{\varepsilon}_{PODNN}$
0.083 %	1.28 %

5.3. ROM of the CVRC chamber

A big advantage of the non-intrusive RB method over the intrusive RB method is that the former method can be used to build a reduced order model of a *local* space/time domain of interest, while the latter method can only be used to build a reduced order model of the entire space/time domain. The ROM of the local domain is more efficient than that of the entire domain, due to the reduction of the number of DOFs. In this subsection, ROM of the CVRC chamber pressure field is presented. The domain of the CVRC chamber is $0 \leq x \leq 38.1$ cm, and discretized into 806 cells.

The snapshots, training, validation and test data are set in the same way as that in the ROM of the entire combustor in Subsection 5.2. The reduced basis functions are extracted from the snapshots using the two-step POD algorithm. The error bounds for the two-step POD are $\varepsilon_t = 5.0 \times 10^{-6}$ and $\varepsilon_{\boldsymbol{\mu}} = 1.0 \times 10^{-4}$, which results in $L = 83$ reduced basis functions and a projection error $\bar{\varepsilon}_{\mathbb{V}} = 0.095\%$, as shown in Table 10.

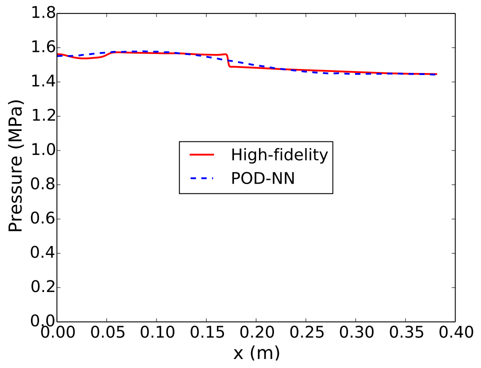
The ANN architecture is presented in Table 9. For the training, the mini-batch size is 810, the regularization parameter is $\lambda = 10^{-7}$, the number of training epochs is 6000, and the learning rate decay is $\eta = 0.1/(1 + 0.5 * epoch)$. The ANN is trained with 10 restarts.

Table 9: *Architecture of the ANN used for the local ROM of the quasi-1D CVRC chamber pressure field.*

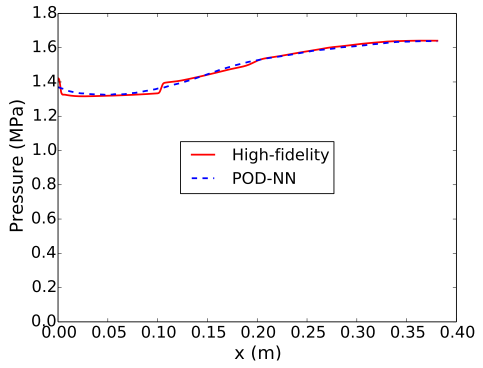
Layer	i	h_1	h_2	h_3	h_4	h_5	o
Number of neurons	5	10	20	30	40	50	83

The average projection error and the average POD-NN error are listed in Table 10. The average POD-NN error is about 7 times larger than the average projection error, while for the global ROM in Subsection 5.2, the average POD-NN error is about 15 times larger than the average projection error. This highlights the benefit of the local ROM over the global ROM with respect to the prediction accuracy. The comparison between the predicted and the high-fidelity solutions for 4 different time/parameter values is shown in Fig. 8. The results in Fig. 8 show that the predicted solutions agree with the high-fidelity solutions very well, but achieved with a smaller basis as compared to the global ROM.

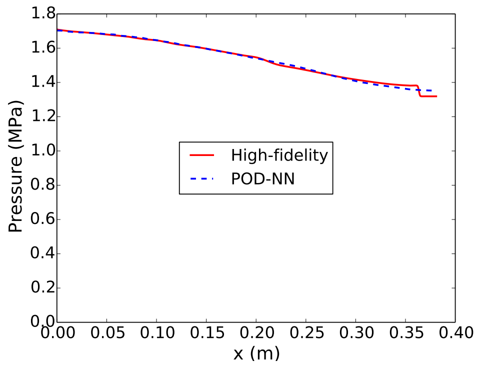
Compared with the ROM of the entire combustor, the ROM of the chamber requires a smaller reduced basis, while achieving the same level of accuracy as the global ROM. This indicates that a non-intrusive RB method can further improve the efficiency of the ROM by building a ROM of the local space/time domain of interest.



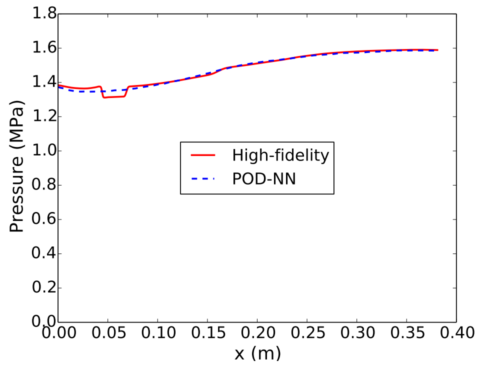
(a) $(t, \boldsymbol{\mu}) = (0.1081, 303.2893, 1032.1611, 0.4309, 0.7908)$



(b) $(t, \boldsymbol{\mu}) = (0.1006, 303.2893, 1035.8901, 0.4328, 0.7908)$



(c) $(t, \boldsymbol{\mu}) = (0.1075, 304.4604, 1035.8901, 0.4309, 0.7928)$



(d) $(t, \boldsymbol{\mu}) = (0.1015, 303.2893, 1035.8901, 0.4172, 0.8059)$

Figure 8: *Comparison between the predicted quasi-1D CVRC chamber pressure fields by a local POD-NN method and the high-fidelity solutions.*

Table 10: *Test results of the POD-NN method for the local ROM of the quasi-1D CVRC chamber pressure field.*

Average projection error, $\bar{\varepsilon}_{\mathbb{V}}$	Average POD-NN error, $\bar{\varepsilon}_{PODNN}$
0.095 %	0.67 %

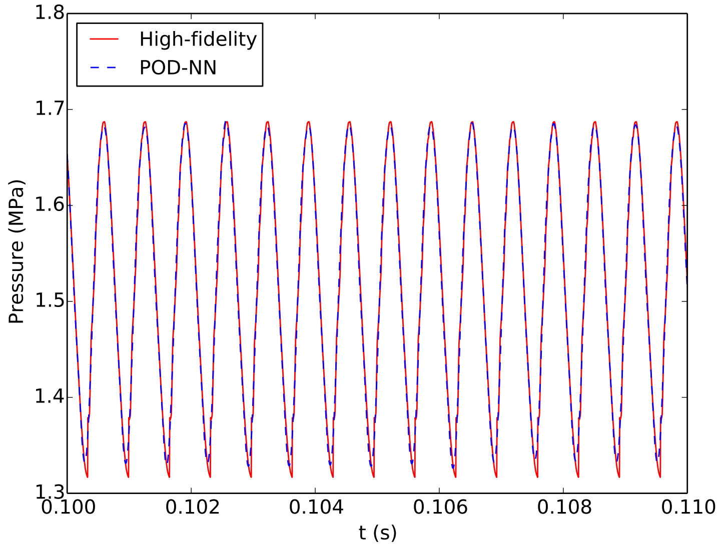


Figure 9: *Comparison between the predicted pressure signal by the POD-NN method and the high-fidelity solution at* $0.10 \leq t \leq 0.11$ *with parameters* $\boldsymbol{\mu} = (303.2893, 1032.1611, 0.4309, 0.7908)$.

5.4. Prediction of pressure signal

To evaluate the unsteady flow, we use the RB model, built in Subsection 5.3, to predict the pressure signal by evaluating the RB solutions in the training time range $0.10 \leq t \leq 0.11$ at $x = 36.8$ cm. The predicted pressure signal of a certain parameter value is shown and compared with that of the high-fidelity solution in Fig. 9. The comparison shows that the predicted pressure signal is accurate.

As we only train the ANN using the data within a certain time range, the direct prediction of long-time behavior of the unsteady-state flows is a difficult problem. However, for periodic problems, we can use the FFT to extend the solution beyond the time range on which the network is trained. The extended predicted pressure signal of Fig. 9 to time range $0.10 \leq t \leq 0.15$ is shown and compared with that of the high-fidelity solution in Fig. 10. The results in Fig. 10 show that the pressure signal obtained by this approach agrees well with the high-fidelity solution. The average relative errors of the predicted pressure signals of the 12 parameter values of the test data used in Subsection 5.2 and 5.3 are listed in Table 11. The results in Table 11 show that the average relative error of the predicted pressure signals of the extended time range is only slightly larger than that of the training time range, which indicates that the POD-NN method can provide accurate prediction of the quasi-1D CVRC flow for a long time range.

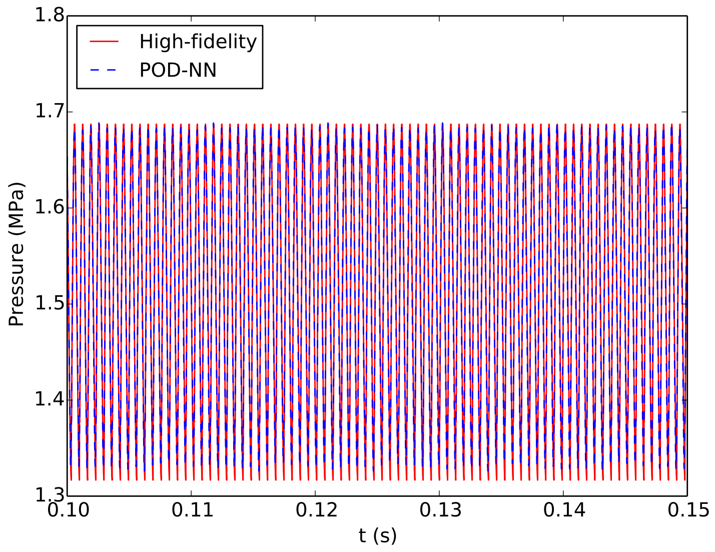


Figure 10: *Comparison between the predicted pressure signal by the POD-NN method and the high-fidelity solution at* $0.10 \le t \le 0.15$ *with parameters* $\boldsymbol{\mu} = (303.2893, 1032.1611, 0.4309, 0.7908)$.

Table 11: *Test results of pressure signal prediction.*

Time range	$0.1 \le t \le 0.11$	$0.1 \le t \le 0.15$
Average relative error	0.63 %	0.86 %

5.5. Stability map validation

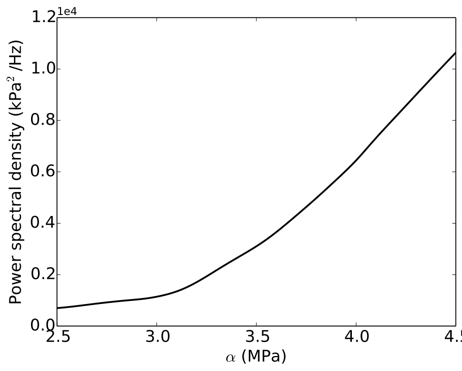
A task of the quasi-1D modeling of CVRC is to predict the effect of the oxidizer post length on the combustion stability. As there are free parameters in the combustion response function, it is necessary to perform a *sensitivity analysis* to investigate the stability behaviors of the system with different parameters. In this section, we study the combustion stability of the CVRC with 13 oxidizer post lengths $L_{op} \in [9, 10, \cdots, 21]$ cm.

The three parameters, α, τ and σ of the combustion response function in (7), have an influence on the combustion stability. According to the sensitivity analysis in [40], α and τ exert a major influence on the solution, controlling its tendency to be unstable. Therefore, the sensitivity study is performed on the parameter space of $\alpha \in [2.5, 4.5]$ MPa and $\tau \in [0.4, 0.5]$ ms.

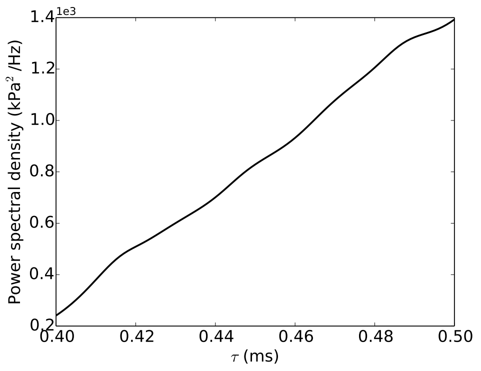
An ANN is built to provide an approximate map from the parameters $\boldsymbol{\mu} = (\alpha, \tau)$ to the power spectral density (PSD) of the most excited frequency of the pressure signal at the location $x = 36.8$ cm. The inputs of the ANN are the parameters (α, τ), the outputs are the PSD values of the most excited frequencies of the pressure signals with 13 different oxidizer post lengths. The architecture of the ANN is presented in Table 12. The training data are collected from 45 high-fidelity simulations, corresponding to a tensor product sampling of 9 points in the α direction and 5 points in the τ direction of the parameter space. The validation and test data are collected from 15 and 8 high-fidelity simulations with randomly chosen parameters. For the training, the batch size is 45, the regularization parameter is $\lambda = 10^{-4}$, the number of training epochs is 9000, and the learning rate decay is $\eta = 0.1/(1 + 0.1 * epoch)$. The ANN is trained with 10 restarts. The average relative error on the test data set is 4.1%.

Table 12: *Architecture of the ANN used for the regression of the PSD.*

Layer	i	h_1	h_2	h_3	h_4	o
Number of neurons	2	20	30	40	50	13



(a) Sensitivity analysis on α



(b) Sensitivity analysis on τ

Figure 11: *Sensitivity analysis on α and τ.*

A sensitivity analysis of the PSD of the oxidizer post length $L_{op} = 14$ cm on α and τ is shown in Fig. 11. The experimental PSD of this case is about 1400 kPa^2/Hz. The sensitivity results show that the PSD increases with either of the two parameters, with a greater sensitivity with respect to α. The PSD is too large compared to the experimental value when $\alpha > 3.5$ MPa, therefore we change the range of α from [2.5, 4.5] MPa to [2.5, 3.5] MPa.

A sensitivity of the *stability map*, obtained by using the Monte Carlo method with 10^6 sampling points, is shown and compared with the experimental results [36] in Fig. 12. The results agree well with the experiment for the oxidizer post length range $L_{op} \leq 9$ cm and $L_{op} \geq 11$ cm, while there is a difference of the stability behavior at $L_{op} = 10$ cm. This difference is likely due to the effects not accounted for in the quasi-1D model.

5.6. Computational cost

In this paper, all computations are performed on a desktop with 40 Intel® Xeon® Gold 6148 @ 2.40 GHz CPUs, and 2 NVIDIA® 10DE:15F0 GPUs. It should be noted that the programs of high-fidelity simulation and POD basis generation are implemented serially. Although we can run several programs at the same time, to avoid confusion, we give the summation of CPU time of each individual simulation as the time cost. The training of the ANNs are performed on 2 GPUs using the TensorFlow library.

The offline cost of the ROM and the sensitivity investigation is presented in Table 13. The data in Table 13 indicates that most of the offline cost is due to the generation of the high-fidelity solutions. The online cost of the ROM and UQ is presented in Table 14. The results in Table 14 highlight that the online evaluation of the ROM of a new parameter value is extremely fast, demonstrating the efficiency of the non-intrusive POD-NN RB method.

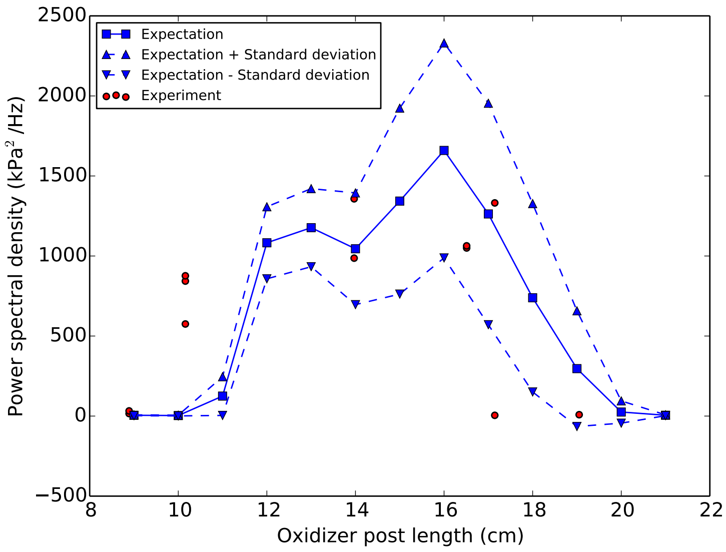


Figure 12: *Stability map of the quasi-1D CVRC model.*

Table 13: *Offline time cost. The unit of the time cost is second (s).*

	ROM of CVRC	ROM of CVRC chamber	Sensitivity of stability map
Snapshots	3.57×10^5	3.57×10^5	2.63×10^6
POD	5.82×10^2	5.60×10^2	–
Training	1.03×10^4	9.5×10^3	4.27×10^2

6. Conclusions

This paper proposes a non-intrusive POD-NN RB method for unsteady flows. The reduced basis is extracted from a collection of high-fidelity solutions via proper orthogonal decomposition (POD), and the reduced coefficients are approximated by an artificial neural network (ANN). The offline stage consists of the generation of the reduced basis and the training of the ANN, while the online stage only performs evaluation of the ANN and linear combination of the RB functions, leading to an efficient POD-NN method. Numerical results of the ROM of the quasi-1D Continuous Variable Resonance Combustor (CVRC) flow, demonstrate the efficiency and robustness of the POD-NN method for a complex unsteady problem.

Table 14: *Online time cost. The unit of the time cost is second (s).*

High-fidelity	ROM of CVRC	ROM of CVRC chamber	Sensitivity of stability map
4.42×10^3	9.42×10^{-4}	3.41×10^{-4}	4.10×10^{-8}

Acknowledgments

The authors gratefully acknowledge Maria L. Frezzotti (Sapienza University of Rome, Italy) and Jiayang Xu (University of Michigan, U.S.) for the help and advice on the CVRC solver.

References

[1] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Review 57 (4) (2015) 483–531.

[2] A. Quarteroni, G. Rozza, et al., Reduced order methods for modeling and computational reduction, Vol. 9, Springer, 2014.

[3] J. S. Hesthaven, G. Rozza, B. Stamm, et al., Certified reduced basis methods for parametrized partial differential equations, Springer, 2016.

[4] Y. Maday, Reduced basis method for the rapid and reliable solution of partial differential equations, in: Proceedings of the International Congress of Mathematicians: Madrid, Spain, 2006, pp. 1255–1269.

[5] G. Berkooz, P. Holmes, J. L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annual Review of Fluid Mechanics 25 (1) (1993) 539–575.

[6] A. Chatterjee, An introduction to the proper orthogonal decomposition, Current Science (2000) 808–817.

[7] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, C. Wu, Proper orthogonal decomposition and its applications-Part I: Theory, Journal of Sound and Vibration 252 (3) (2002) 527–544.

[8] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, P. Wojtaszczyk, Convergence rates for greedy algorithms in reduced basis methods, SIAM Journal on Mathematical Analysis 43 (3) (2011) 1457–1472.

[9] A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme, G. Turinici, A priori convergence of the greedy algorithm for the parametrized reduced basis method, ESAIM: Mathematical Modelling and Numerical Analysis 46 (3) (2012) 595–603.

[10] J. S. Hesthaven, B. Stamm, S. Zhang, Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods, ESAIM: Mathematical Modelling and Numerical Analysis 48 (1) (2014) 259–283.

[11] B. Haasdonk, M. Ohlberger, Reduced basis method for finite volume approximations of parametrized linear evolution equations, ESAIM: Mathematical Modelling and Numerical Analysis 42 (2) (2008) 277–302.

[12] F. Ballarin, A. Manzoni, A. Quarteroni, G. Rozza, Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible navier–stokes equations, International Journal for Numerical Methods in Engineering 102 (5) (2015) 1136–1161.

[13] M. Couplet, C. Basdevant, P. Sagaut, Calibrated reduced-order POD-Galerkin system for fluid flow modelling, Journal of Computational Physics 207 (1) (2005) 192–220.

[14] A. Quarteroni, G. Rozza, A. Manzoni, Certified reduced basis approximation for parametrized partial differential equations and applications, Journal of Mathematics in Industry 1 (1) (2011) 3.

[15] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Mathematique 339 (9) (2004) 667–672.

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

[17] B. Peherstorfer, D. Butnaru, K. Willcox, H.-J. Bungartz, Localized discrete empirical interpolation method, SIAM Journal on Scientific Computing 36 (1) (2014) A168–A192.

[18] F. Negri, A. Manzoni, D. Amsallem, Efficient model reduction of parametrized systems by matrix discrete empirical interpolation, Journal of Computational Physics 303 (2015) 431–454.

[19] T. Lieu, C. Farhat, M. Lesoinne, Reduced-order fluid/structure modeling of a complete aircraft configuration, Computer Methods in Applied Mechanics and Engineering 195 (41-43) (2006) 5730–5742.

[20] P. Benner, L. Feng, S. Li, Y. Zhang, Reduced-order modeling and ROM-based optimization of batch chromatography, in: Numerical Mathematics and Advanced Applications-ENUMATH 2013, Springer, 2015, pp. 427–435.

[21] A. Iollo, S. Lanteri, J.-A. Désidéri, Stability properties of POD–Galerkin approximations for the compressible Navier–Stokes equations, Theoretical and Computational Fluid Dynamics 13 (6) (2000) 377–396.

[22] M. F. Barone, I. Kalashnikova, D. J. Segalman, H. K. Thornquist, Stable Galerkin reduced order models for linearized compressible flow, Journal of Computational Physics 228 (6) (2009) 1932–1946.

[23] F. Casenave, A. Ern, T. Lelièvre, A nonintrusive reduced basis method applied to aeroacoustic simulations, Advances in Computational Mathematics 41 (5) (2015) 961–986.

[24] V. Barthelmann, E. Novak, K. Ritter, High dimensional polynomial interpolation on sparse grids, Advances in Computational Mathematics 12 (4) (2000) 273–288.

[25] D. Amsallem, Interpolation on manifolds of CFD-based fluid and finite element-based structural reduced-order models for on-line aeroelastic predictions, Ph.D. thesis, Stanford University (2010).

[26] J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, Journal of Computational Physics 363 (2018) 55 – 78.

[27] M. Guo, J. S. Hesthaven, Reduced order modeling for nonlinear structural analysis using gaussian process regression, accepted by Computer Methods in Applied Mechanics and Engineering.

[28] M. Riedmiller, Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms, Computer Standards & Interfaces 16 (3) (1994) 265–278.

[29] Y. Yu, S. Koeglmeier, J. Sisco, W. Anderson, Combustion instability of gaseous fuels in a continuously variable resonance chamber (CVRC), in: 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 2008, p. 4657.

[30] R. Garby, Simulations of flame stabilization and stability in high-pressure propulsion systems, Ph.D. thesis, INPT (2013).

[31] Y. Yu, L. O'Hara, J. Sisco, W. Anderson, Experimental study of high-frequency combustion instability in a continuously variable resonance combustor (CVRC), in: 47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2009, p. 234.

[32] J. C. Sisco, Y. Yu, V. Sankaran, W. E. Anderson, Examination of mode shapes in an unstable model combustor, Journal of Sound and Vibration 330 (1) (2011) 61–74.

[33] Y. Yu, J. C. Sisco, S. Rosen, A. Madhav, W. E. Anderson, Spontaneous longitudinal combustion instability in a continuously-variable resonance combustor, Journal of Propulsion and Power 28 (5) (2012) 876–887.

[34] M. E. Harvazinski, W. E. Anderson, C. L. Merkle, Analysis of self-excited combustion instabilities using two-and three-dimensional simulations, Journal of Propulsion and Power 29 (2) (2013) 396–409.

[35] L. Selle, R. Blouquin, M. Théron, L.-H. Dorey, M. Schmid, W. Anderson, Prediction and analysis of combustion instabilities in a model rocket engine, Journal of Propulsion and Power 30 (4) (2014) 978–990.

[36] M. E. Harvazinski, C. Huang, V. Sankaran, T. W. Feldman, W. E. Anderson, C. L. Merkle, D. G. Talley, Coupling between hydrodynamics, acoustics, and heat release in a self-excited unstable combustor, Physics of Fluids 27 (4) (2015) 045102.

[37] R. Smith, M. Ellis, G. Xia, V. Sankaran, W. Anderson, C. Merkle, Computational investigation of acoustics and instabilities in a longitudinal-mode rocket combustor, AIAA Journal 46 (11) (2008) 2659–2673.

[38] M. L. Frezzotti, F. Nasuti, C. Huang, C. Merkle, W. E. Anderson, Determination of heat release response function from 2D hybrid RANS-LES data for the CVRC combustor, in: 51st AIAA/SAE/ASEE Joint Propulsion Conference, 2015, p. 3841.

[39] M. L. Frezzotti, S. D'Alessandro, B. Favini, F. Nasuti, Numerical issues in modeling combustion instability by quasi-1D euler equations, International Journal of Spray and Combustion Dynamics 9 (4) (2017) 349–366.

[40] M. L. Frezzotti, F. Nasuti, C. Huang, C. L. Merkle, W. E. Anderson, Quasi-1D modeling of heat release for the study of longitudinal combustion instability, Aerospace Science and Technology 75 (2018) 261–270.

[41] C. Huang, W. E. Anderson, C. L. Merkle, V. Sankaran, Investigation of the Stability of POD-Galerkin Techniques for Reduced Order Model Development, Tech. rep., AFRL/RQR Edwards AFB United States (2016).

[42] J. Xu, K. Duraisamy, Reduced-order modeling of model rocket combustors, in: 53rd AIAA/SAE/ASEE Joint Propulsion Conference, 2017, p. 4918.

[43] E. Schmidt, On the Theory of Linear and Nonlinear Integral Equations. Part I: Development of arbitrary function according to systems prescribed, Mathematical Annals 63 (1907) 433–476.

[44] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.

[45] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[46] M. W. Gardner, S. Dorling, Artificial neural networks (the multilayer perceptron)-a review of applications in the atmospheric sciences, Atmospheric Environment 32 (14-15) (1998) 2627–2636.

[47] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[48] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 437–478.

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `https://www.tensorflow.org/`

[50] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[51] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[52] K.-l. Hsu, H. V. Gupta, S. Sorooshian, Artificial neural network modeling of the rainfall-runoff process, Water Resources Research 31 (10) (1995) 2517–2530.

[53] A. Delis, C. Skeels, TVD schemes for open channel flow, International Journal for Numerical Methods in Fluids 26 (7) (1998) 791–809.

[54] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, Journal of Computational Physics 77 (2) (1988) 439–471.

[55] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, Journal of Computational Physics 126 (1) (1996) 202–228.

[56] M. Frezzotti, F. Nasuti, C. Huang, C. Merkle, W. Anderson, Parametric Analysis of Response Function in Modeling Combustion Instability by a Quasi-1D Solver, in: 6th EUROPEAN CONFERENCE FOR AEROSPACE SCIENCES, 2015.