

# Parametric Dynamic Mode Decomposition for nonlinear parametric dynamical systems

Shuwen Sun\*    Lihong Feng\*    Hoon Seng Chan<sup>†</sup>    Tamara Miličić\*  
 Tanja Vidaković-Koch\*    Peter Benner\*

\*Max Planck Institute for Dynamics of Complex Technical Systems

Email: [ssun@mpi-magdeburg.mpg.de](mailto:ssun@mpi-magdeburg.mpg.de);

<sup>†</sup>Karlsruhe Institute of Technology

**Abstract:** A non-intrusive model order reduction (MOR) method that combines features of the dynamic mode decomposition (DMD) and the radial basis function (RBF) network is proposed to predict the dynamics of parametric nonlinear systems. In many applications, we have limited access to the information of the whole system, which motivates non-intrusive model reduction. One bottleneck is capturing the dynamics of the solution without knowing the physics inside the “black-box” system. DMD is a powerful tool to mimic the dynamics of the system and give a reliable approximation of the solution in the time domain using only the dominant DMD modes. However, DMD cannot reproduce the parametric behavior of the dynamics. Our contribution focuses on extending DMD to parametric DMD by RBF interpolation. Specifically, an RBF network is first trained using snapshot matrices at limited parameter samples. The snapshot matrix at any new parameter sample can be quickly learned from the RBF network. DMD will use the newly generated snapshot matrix at the online stage to predict the time patterns of the dynamics corresponding to the new parameter sample. The proposed framework and algorithm are tested and validated by numerical examples including models with parametrized and time-varying inputs.

**Keywords:** Model reduction, Dynamic mode decomposition, Radial basis function, nonlinear systems with parametrized inputs

**Novelty statement:** Parametric dynamic mode decomposition using RBF network for non-intrusive MOR of parametric nonlinear dynamical systems including systems with parametrized and time-varying inputs.

## 1 Introduction

Nonlinear dynamical systems arise from many physical and engineering applications. Solving systems with nonlinear effects and parameter variations indeed costs a lot of time and effort, which motivates model order reduction, a technique of constructing compact surrogates of nonlinear systems to realize accelerated computation with acceptable accuracy. The computational efforts in constructing the surrogate, i.e., the reduced-order model (ROM) is usually constructed at the offline stage, while the process of employing the ROM for simulation or any other multi-query tasks is known as the online stage. When the online stage is fast enough, it can be stated as “real-time” computation and is

promising for real applications. There are various subtopics and methods in MOR aiming at different applications, such as modal truncation, balanced truncation [20,32], Krylov subspace methods (moment matching) [17], local linear embedding (LLE) [41], proper orthogonal decomposition (POD, also known as principal component analysis in the statistical area or Karhunen-Loeve expansion in the stochastic area) [30,31,36,44], reduced basis methods, dynamic mode decomposition (DMD) [45], data-driven and machine learning approaches [2,5,10,18,19,22,24,34,37–39,51,53].

When a dynamic system is seen as a “black box” so that the only information of the system are the inputs and its corresponding outputs, intrusive MOR based on projection is impossible, and non-intrusive MOR is preferred. Efficient MOR for nonlinear time-evolution systems parametrized with some physical or geometrical parameters is challenging. Although, intrusive MOR based on projection for such systems has achieved much success [6–8], non-intrusive MOR methods that are robust for systems characterized by all the above three properties, i.e., nonlinear, parametric and time-dependent, are still not fully explored, though some are proposed [5,9,13,16,18,21,24,38,39,50,52]. At present, more and more non-intrusive MOR methods are based on machine learning to tackle such systems with strong nonlinearity [14,16,21,39,50,52]. Furthermore, many of the existing methods assume that the solution space is of low dimension, and a global reduced space over the whole parameter domain is assumed [10,18,37–39]. Fewer non-intrusive methods are successful for systems in which the solution is non-smooth in the parameter domain [14,16,50,52]. Non-intrusive MOR methods with emphasis on treating non-smooth or convection-dominated problems are also proposed [26,33,42,47]. To the best of the authors’ knowledge, many of them are only applicable to either parametric steady problems or time-evolution problems without parameters.

Dynamic mode decomposition can provide a way of discovering low-rank space-time patterns of the dynamics in an equation-free manner [11]. DMD is first introduced to realize the nonlinear evolution of fluid dynamics. Based on the snapshot matrix from the system, DMD computes a linear operator that maps the snapshots one time step further. It appeared firstly in [43] and then it was later used for model order reduction. There exist different variants of the DMD method to overcome the different drawbacks of the standard DMD, such as Extended DMD [48] and Kernel DMD [49]. DMD is also combined with autoencoder for non-intrusive model reduction of nonlinear dynamical systems [35].

This work focuses on extending DMD to parametric DMD by combining DMD with the RBF network to achieve fast approximation of both the parametric behavior and time-evolution of the dynamics in a non-intrusive way. Compared with the existing methods based on deep learning, our proposed method is much faster to train, since the RBF network is known as a shallow neural network with much fewer parameters to be optimized during network training. Yet, the derived ROMs are still of acceptable accuracy. Some closely related methods are proposed in [50], where the RBF network is combined with POD and is also used for prediction in the time domain. Due to the limitation of the RBF interpolation only in the time domain, the method in [50] cannot predict the solution at a future time that is outside of the time interval used for training. Another recent work on parametric DMD in [23] aims at reaching the same goal as our proposed method in different ways. Two different parametric DMD methods are proposed. The first method interpolates the eigenpairs associated with the projected Koopman matrices at different parameter samples. The second method instead interpolates the projected Koopman matrices corresponding to different parameter samples. Each method necessitates the second stage of interpolation: interpolating the associated DMD modes in the parameter domain to recover the solution in the original space. However, some limitations are also mentioned in this paper. The most restricting limitations are the following assumptions. Given the polynomial interpolation method used in [23], the smoothness of the eigenpairs over the parameter domain must be satisfied for the accuracy of the first method, and smoothness of the projected Koopman operator w.r.t the parameters is required for the second method to be successful. Another limitation lies in the fact that projected DMD used in [23] can not assure the dynamic modes are exactly the eigenvectors of the original Koopman matrix. In the latest paper [29], similar work has been done using DMD for non-intrusive MOR of parametric systems, the reduced-order model at any testing parameter sample is obtained from manifold-interpolation of the left singular vectors at training parameter samples and manifold-interpolation of the projected Koopman matrices. Some hyperparameters need to be

heuristically tuned to achieve success, for example, the reference configuration  $i_0$ , which could lead to failure of the method if not optimally chosen. Furthermore, the proposed DMD method can only reconstruct the observables of the solution. The solution needs to be recovered by implementing an inverse mapping from the observables to the state space. For observables with a complex expression, it is unclear how the inverse mapping can be computed.

In this work, the power of DMD for time-dependent problems is combined with the RBF network to derive a method that is robust for prediction in both the parameter domain and the time domain. When compared to the existing DMD-based methods for MOR of the parametric dynamical system, the RBF network that is applied for snapshot interpolation leads to the proposed parametric DMD method with much less constraints.

The remaining part of the work is organized as follows. In [Section 2](#), a general overview of DMD is provided. The algorithm of the exact DMD and the kernel DMD are presented for use in the next sections. In [Section 3](#), radial basis function (RBF) is shortly introduced. Then the proposed method, a practical algorithm, and some discussions are given. In [Section 4](#), three examples from real applications are presented to demonstrate the robustness of the proposed method. We conclude the work in [Section 5](#) with further outlook.

## 2 Dynamic Mode Decomposition

DMD is a non-intrusive MOR method for time-dependent systems. It provides a low-dimensional representation of the system solution via spatiotemporal decomposition of the dynamics. The main tool is the singular value decomposition (SVD) of a large data matrix and the eigendecomposition of a small projected data matrix. Suppose we have a nonlinear dynamic system of ordinary differential equations (ODEs):

$$u'(t) = g(u(t)), \quad (1)$$

where the state vector  $u(t) \in \mathbb{R}^n$ ,  $g: \mathbb{R}^n \rightarrow \mathbb{R}^n$  is a nonlinear operator. Applying explicit time integration scheme to (1) results in the following nonlinear evolution,

$$u_{i+1} = F(u_i), \quad i = 0, \dots, m-1. \quad (2)$$

Note that  $F$  may also depend on  $u_{i-1}$ , etc. for a multi-step integration scheme. For simplicity of explanation, those dependencies are omitted here.

Consider the snapshot matrix  $\mathbf{X}_0$  and the shifted snapshot matrix  $\mathbf{X}_1$  as follows:

$$\mathbf{X}_0 = \begin{bmatrix} | & | & \cdots & | \\ u_0 & u_1 & \cdots & u_{m-1} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times m}, \mathbf{X}_1 = \begin{bmatrix} | & | & \cdots & | \\ u_1 & u_2 & \cdots & u_m \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times m}, \quad (3)$$

where  $u_k = u(t_k)$ ,  $k = 1, \dots, m$ , are state vectors at time  $t_k$  within a certain time interval. They are also known as snapshots. It is worth pointing out that these snapshots do not necessarily follow the order of their appearance during simulation, which means  $t_{k-1}$  is not necessarily smaller (earlier) than  $t_k$ ,  $\forall k \leq m$  [45]. DMD uses a linear time evolution to approximate the nonlinear evolution in (2), i.e.

$$\mathbf{X}_1 = \mathbf{K}\mathbf{X}_0. \quad (4)$$

Then it finds the best fit  $\mathbf{A}$  for the linear operator  $\mathbf{K} \in \mathbb{R}^{n \times n}$ . Mathematically, we have

$$\mathbf{A} = \operatorname{argmin}_{\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}} \left\| \mathbf{X}_1 - \tilde{\mathbf{A}}\mathbf{X}_0 \right\|_F = \mathbf{X}_1 \mathbf{X}_0^\dagger, \quad (5)$$

where  $\|\cdot\|_F$  is the Frobenius norm and  $\dagger$  is the pseudo-inverse operator. When  $\mathbf{X}_0$  and  $\mathbf{X}_1$  are linearly consistent, i.e., whenever  $\mathbf{X}_0 c = 0$ , then  $\mathbf{X}_1 c = 0$ , then it is proved in [45] that  $\mathbf{A}$  satisfies (4), i.e.,  $\mathbf{X}_1 = \mathbf{A}\mathbf{X}_0$ . From the eigendecomposition of  $\mathbf{A}$  we obtain the eigenvalues and eigenvectors of  $\mathbf{A}$ . The

eigenvectors are also known as the DMD modes [45]. Reconstruction of the state can be done using these DMD modes and their evolution configured by the eigenvalues. Each eigenvalue represents the growth/decay rate (real part of the complex value) and oscillation with different frequencies (imaginary part of the value) of the corresponding mode. When  $n$  is large, the eigendecomposition of  $\mathbf{A}$  becomes inefficient. The practical algorithm of implementing DMD takes use of dimension reduction via SVD of the initial snapshot matrix  $\mathbf{X}_0$  to compute the dominant DMD modes from the (truncated) left singular vectors  $\mathbf{U}$  and the eigendecomposition of the small projected matrix  $\hat{\mathbf{A}} = \mathbf{U}^* \mathbf{A} \mathbf{U}$ . Algorithm 1 presents the detailed procedure of the exact DMD algorithm first proposed in [45].

The main difference between exact DMD and a previously proposed standard DMD (also known as projected DMD) lies in the way of computing the DMD modes. For the standard DMD, a DMD mode is computed from the matrix  $\mathbf{U}$  of left singular eigenvector:

$$\hat{\varphi} = \mathbf{U}w, \quad (6)$$

where  $w$  is an eigenvector of  $\hat{\mathbf{A}}$ , corresponding to an eigenvalue  $\lambda$ . However, for the exact DMD, the DMD mode  $\varphi$  is defined as lying in the image of  $\mathbf{X}_1$  instead of that of  $\mathbf{X}_0$ . It is computed as follows:

$$\varphi = \frac{1}{\lambda} \mathbf{X}_1 \mathbf{V} \Sigma^{-1} w. \quad (7)$$

The aim of computing  $\varphi$  following (7) is to make sure that  $\varphi$  is the eigenvector of the original linear operator  $\mathbf{A}$ , i.e.,  $\mathbf{A}\varphi = \lambda\varphi$ . This property is used in Step 6 in Algorithm 1 for the reconstruction of the dynamics. Whereas,  $\hat{\varphi}$  in (6) doesn't meet such a requirement. A detailed explanation can be found in [45].

After the DMD modes are computed in Algorithm 1, the solution at any future time  $t_i$  can be reconstructed from the DMD modes, and their initial amplitudes  $b_k$  computed based on the initial solution, see Steps 5-6 in Algorithm 1.

*Remark 1.* The truncation in Step 2 of Algorithm 1 did not appear in the original exact DMD in [45] but was included in the exact DMD algorithm presented in [27] so that the computational cost of the eigendecomposition of  $\hat{\mathbf{A}}$  is further reduced. The truncation rank  $r$  is determined according to the energy criteria:

$$\frac{\sum_{i=r+1}^d \sigma_i}{\sum_{i=1}^d \sigma_i} \leq \eta, \quad (8)$$

where the  $\eta$  is the tolerance decided by the user. This may introduce truncation errors, however, we found in the numerical tests that when  $r \ll d$ , the DMD still produces results with acceptable accuracy. Furthermore, once the truncation is introduced, the DMD modes computed in Step 5 are not the eigenvectors of  $\mathbf{A}$  anymore.

## 2.1 Extended and kernel DMD

DMD uses a linear evolution scheme (4) to approximate the nonlinear evolution (2), which might cause big errors for some problems with strong nonlinearities. To improve the accuracy of DMD, extended DMD (EDMD) is proposed in [48]. Assuming that the state vector  $u$  in (1) can be spanned by  $s$  eigenfunctions  $\phi_k(u)$ ,  $k = 1, \dots, s$  of the Koopman operator  $\mathcal{K}$ , i.e.,

$$u = \sum_{k=1}^s v_k \phi_k(u), \quad (9)$$

then the nonlinear evolution (2) can be fully described by the Koopman operator via its eigenfunctions, eigenvalues and modes (see [48] for detailed derivation), i.e.,

$$F(u) = \sum_{k=1}^s v_k (\mathcal{K} \phi_k)(u) = \sum_{k=1}^s \lambda_k v_k \phi_k(u). \quad (10)$$

**Algorithm 1** Exact DMD [27, 45]

- 1: Collect the snapshots for the snapshot matrix pair  $\{\mathbf{X}_0, \mathbf{X}_1\}$  in (3).
- 2: Compute the compact SVD of the data snapshot matrix  $\mathbf{X}_0 = \mathbf{U}\Sigma\mathbf{V}^T$ ,  $\mathbf{U} \in \mathbb{R}^{n \times d}$ ,  $\Sigma \in \mathbb{R}^{d \times d}$ ,  $\mathbf{V} \in \mathbb{R}^{m \times d}$ ,  $d \leq \min(m, n)$  is the rank of  $\mathbf{X}_0$ . Make truncation and remain only the  $r < d$  leading eigenvalues and the corresponding eigenvectors, so that  $\mathbf{X}_0 \approx \mathbf{U}_r \Sigma_r \mathbf{V}_r^T$ , where  $\mathbf{U}_r \in \mathbb{R}^{n \times r}$ ,  $\Sigma_r \in \mathbb{R}^{r \times r}$ , and  $\mathbf{V}_r \in \mathbb{R}^{m \times r}$ .
- 3: Compute  $\hat{\mathbf{A}} = \mathbf{U}_r^T \mathbf{A} \mathbf{U}_r$  by replacing  $\hat{\mathbf{A}}$  with  $\mathbf{X}_1 \mathbf{X}_0^\dagger$ , and  $\mathbf{X}_0$  with its SVD in Step 2, i.e.,  $\hat{\mathbf{A}} = \mathbf{U}_r^T \mathbf{A} \mathbf{U}_r = \mathbf{U}_r^T \mathbf{X}_1 \mathbf{V}_r \Sigma_r^{-1} \mathbf{U}_r^T \mathbf{U}_r = \mathbf{U}_r^T \mathbf{X}_1 \mathbf{V}_r \Sigma_r^{-1}$ .
- 4: Compute eigendecomposition of  $\hat{\mathbf{A}}$ :  $\hat{\mathbf{A}} \mathbf{W} = \mathbf{W} \Lambda$ , with  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_r)$ .
- 5: Compute the DMD modes  $\Phi = [\varphi_1, \dots, \varphi_r]$  by  $\Phi = \mathbf{X}_1 \mathbf{V} \Sigma_r^{-1} \mathbf{W}$ . Given the initial solution  $u_0$  and suppose it can be represented by the DMD modes, i.e.,  $u_0 = \Phi b$ , then the vector of weights (coefficient)  $b = (b_1, \dots, b_r)^T$  can be computed as  $b = \Phi^\dagger u_0$ .
- 6: Reconstruct of the solution at any future time  $t_i > 0$  using the DMD modes:  $u_i = \mathbf{A}^i u_0 = \sum_{k=1}^r \varphi_k b_k \lambda_k^i$ .

Here,  $\mathcal{K}$  is the Koopman operator,  $\phi_k(u)$  are the Koopman eigenfunctions,  $v_k$  are the Koopman modes, and  $\lambda_k$  are the Koopman eigenvalues. Motivated by (10), EDMD tries to approximate the nonlinear evolution (2) via approximating the Koopman operator, its eigenfunctions and modes. The Koopman operator is approximated by using not only the data matrices  $\mathbf{X}_0, \mathbf{X}_1$  but also a dictionary of functions of the state vector (observables),  $\{\psi_1(u), \psi_2(u), \dots, \psi_M(u)\}$ . Finally, the Koopman operator is approximated by a matrix  $\tilde{\mathbf{K}}$ :

$$\tilde{\mathbf{K}} = \Psi_0^\dagger \Psi_1, \quad (11)$$

where  $\Psi_0$  and  $\Psi_1 \in \mathbb{R}^{m \times M}$  can be written in the following form:

$$\Psi_0 = \begin{bmatrix} \psi_1(u_0) & \cdots & \psi_M(u_0) \\ \psi_1(u_1) & \cdots & \psi_M(u_1) \\ \vdots & & \vdots \\ \psi_1(u_{m-1}) & \cdots & \psi_M(u_{m-1}) \end{bmatrix}, \quad \Psi_1 = \begin{bmatrix} \psi_1(u_1) & \cdots & \psi_M(u_1) \\ \psi_1(u_2) & \cdots & \psi_M(u_2) \\ \vdots & & \vdots \\ \psi_1(u_m) & \cdots & \psi_M(u_m) \end{bmatrix}. \quad (12)$$

The eigenfunctions of the Koopman operator and the Koopman modes then can be computed from the right eigenvectors and left eigenvectors of  $\tilde{\mathbf{K}}$ , respectively. The eigenvalues of  $\tilde{\mathbf{K}}$  are the approximation of the eigenvalues of  $\mathcal{K}$ . For detailed derivation see [48]. A computational issue with EDMD is the expensive cost of computing the eigendecomposition of  $\tilde{\mathbf{K}} \in \mathbb{R}^{M \times M}$  when  $M \gg m$ , which is often the case in many applications.

Kernel DMD is proposed in [49] to reduce the computational cost of EDMD. This is done by using the compact SVD of the matrix  $\Psi_0 = \mathbf{Q}\Sigma\mathbf{Z}^T$ ,  $\mathbf{Q}, \Sigma \in \mathbb{R}^{m \times m}$ ,  $\mathbf{Z} \in \mathbb{R}^{M \times m}$ . It is then proved in [49] that  $\tilde{\mathbf{K}}$  has the same eigenvalues with a smaller matrix  $\hat{\mathbf{K}} = (\Sigma^{-1}\mathbf{Q}^T)(\Psi_1\Psi_0^T)(\mathbf{Q}\Sigma^{-1}) \in \mathbb{R}^{m \times m}$ . Any right eigenvector  $v$  of  $\tilde{\mathbf{K}}$  corresponding to an eigenvalue  $\lambda$  can be computed from the right eigenvector  $\hat{v}$  of  $\hat{\mathbf{K}}$  by  $\mathbf{v} = \mathbf{Z}\hat{v}$ . From the SVD of  $\Psi_0$ , it is noticed that the eigendecomposition of  $\Psi_0\Psi_0^T$  is,

$$\Psi_0\Psi_0^T = \mathbf{Q}\Sigma^2\mathbf{Q}^T \in \mathbb{R}^{m \times m}. \quad (13)$$

Therefore, if we can compute the eigendecomposition of  $\Psi_0\Psi_0^T$  and get  $\mathbf{Q}, \Sigma$ , then  $\hat{\mathbf{K}}$  can be derived without SVD of  $\Psi_0$ . The eigendecomposition of  $\Psi_0\Psi_0^T$  is of complexity  $\mathcal{O}(m^3)$ , which is less than  $\mathcal{O}(Mm^2)$ , the SVD cost of  $\Psi_0\Psi_0^T$ . It is further noticed that computing  $\Psi_0\Psi_0^T$  and  $\Psi_1\Psi_0^T$  is essentially implementing inner products of the two vectors  $\psi(u_i) := [\psi_1(u_i), \dots, \psi_M(u_i)]^T$  and  $\psi(u_j) := [\psi_1(u_j), \dots, \psi_M(u_j)]^T$ ,  $i, j = 0, 1, \dots, m$ . When  $M$  is large, the computational cost of these inner products cannot be neglected. Usually, the observables include both the state variables and functions of them, making even  $M \gg n$ . The kernel function is then used to compute these inner

products. As a result, the inner products in  $\mathbb{R}^M$  is equivalently transformed to inner products in  $\mathbb{R}^n$ . This reduces the computations of directly computing the inner products  $\psi(u_i)^T \psi(u_j)$ . Please refer to [49] for a detailed explanation using illustrative examples. The final kernel DMD algorithm is reviewed in Algorithm 2, where  $f(u_i, u_j)$  is a kernel function. Since  $\mathbf{Z}$  can be represented by  $\Psi_0$ ,  $\mathbf{Q}$  and  $\Sigma$ , the eigenvectors of  $\tilde{\mathbf{K}}$  can also be recovered without SVD of  $\Psi_0$ . Furthermore, the eigenmodes of  $\tilde{\mathbf{K}}$  are also computed independently of the SVD of  $\Psi_0$ , see Step 7 in Algorithm 2. For a detailed derivation of it, please refer to [49]. Some common kernel functions that can be chosen are the polynomial kernel  $f(x, y) = (1 + y^T x)^\alpha$  or Gaussian kernel  $f(x, y) = \exp(-\|x - y\|^2 / \sigma^2)$ .

---

**Algorithm 2** Kernel DMD [49]
 

---

- 1: Compute elements of two matrices  $\Psi_0 \Psi_0^T$  and  $\Psi_1 \Psi_0^T$  by kernel function:  $(\Psi_0 \Psi_0^T)_{ij} = f(u_i, u_j)$  and  $(\Psi_1 \Psi_0^T) = f(u_{i+1}, u_j)$ , with  $i, j = 0, \dots, m - 1$ ;
  - 2: Compute the eigendecomposition of  $\Psi_0 \Psi_0^T$  via (13) to get  $\mathbf{Q}$ , and  $\Sigma$ .
  - 3: (optional) Choose the truncation rank  $r$  that is smaller than the rank of  $\Psi_0 \Psi_0^T$  to achieve a further reduction of the computation. Truncate the matrices  $\mathbf{Q}$ , and  $\Sigma$  by keeping the first  $r$  columns of  $\mathbf{Q}$  and first  $r$  diagonal blocks  $\Sigma$  to obtain  $\mathbf{Q}_r$  and  $\Sigma_r$ .
  - 4: Compute  $\hat{\mathbf{K}} = (\Sigma_r^{-1} \mathbf{Q}_r^T) (\Psi_1 \Psi_0^T) (\mathbf{Q}_r \Sigma_r^{-1})$ .
  - 5: Compute the eigendecomposition of  $\hat{\mathbf{K}} \hat{\mathbf{W}} = \hat{\mathbf{W}} \hat{\Lambda}$  with  $\hat{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_r)$ .
  - 6: Compute the eigenfunction matrix  $\Phi = \mathbf{Q}_r \Sigma_r \hat{\mathbf{W}}$ .
  - 7: Set the Koopman modes as  $v_i = (\hat{\xi}_i^* \Sigma_r^{-1} \mathbf{Q}_r^T \mathbf{X}_1)^T$ , where  $\hat{\xi}_i$  is the left eigenvector of the matrix  $\hat{\mathbf{K}}$ , and  $\hat{\xi}_i^* \hat{w}_i = 1$ .
  - 8: With eigenvalues, eigenfunctions and Koopman modes  $\lambda_i, \phi_i, v_i$ , the approximation of the evolution can be done via (10).
- 

However, either exact DMD or extended/kernel DMD cannot be straightforwardly applied to parametric problems, where the solution depends not only on the initial solution but also on the parameter variations. The parametric behavior of the solution usually cannot be captured by the DMD modes corresponding to any fixed value of the parameter provided by the DMD method. In the next section, we extend DMD to parametric DMD based on the RBF network.

### 3 Proposed Parametric DMD

In many applications, parametric systems are widely used in multi-query tasks, such as optimal design, control, or uncertainty quantification. In this work, we consider parametric systems in a general form as,

$$\begin{aligned} \frac{du(t, \mu)}{dt} &= g(u(t, \mu), \mu), & u(t_0) &= u_0, \\ y(t, \mu) &= s(u(t, \mu)). \end{aligned} \quad (14)$$

where  $\mu$  is the vector of parameters,  $u(t, \mu) \in \mathbb{R}^n$  is the vector of states, and  $y(t, \mu) \in \mathbb{R}^{n_0}$  is the quantity of interest, also called the output. Existing DMD methods could not compute DMD modes which are also parametric, and as a result, they can only reconstruct the dynamics corresponding to a fixed value of  $\mu$ . Whenever the parameter value changes, DMD has to be reimplemented from scratch. In this section, we propose combining DMD with the RBF network to construct non-intrusive ROMs for parametric systems, which can predict the system's dynamics in both the parameter domain and the time domain. In Section 3.1, we first review the RBF network, then in Section 3.2, we connect it with DMD to realize parametric DMD.

#### 3.1 Radial Basis Function Network

The RBF method uses the weighted kernel function to approximate a given function  $f(x): \mathbb{R}^\ell \rightarrow \mathbb{R}$  based on the data of  $f(x)$ . The approximate function  $\hat{f}(x)$  constructed by RBF can be written as

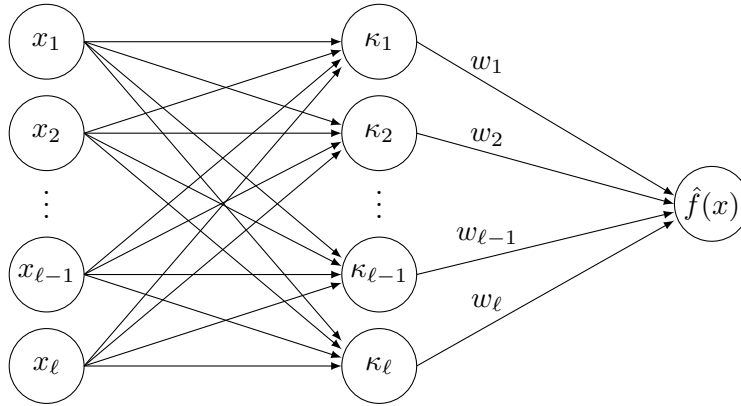


Figure 1: RBF interpolation as a shallow neural network.

weighted summation of the RBFs, i.e.,

$$f(x) \approx \hat{f}(x) = \sum_{i=1}^{\ell} w_i \kappa(\|x - x_i\|). \quad (15)$$

The kernel function is radially symmetric based on Euclidean distance  $\|x - x_i\|$  or comparable metrics. The coefficients or weights  $w_i, i = 1, \dots, \ell$  are determined by solving a linear system in (16). The whole process of computing  $\hat{f}(x)$  works like a shallow neural network shown in Figure 1, where  $\kappa_i = \kappa(\|x - x_i\|)$ . After the weights are fixed, the interpolation can be completed simply using the weighted summation in (15). The detailed process of computing the approximate function  $\hat{f}(x)$  is presented in Algorithm 3.

$$\begin{bmatrix} \kappa(x_1 - x_1) & \dots & \kappa(x_1 - x_\ell) \\ \vdots & \vdots & \vdots \\ \kappa(x_\ell - x_1) & \dots & \kappa(x_\ell - x_\ell) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_\ell \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_\ell) \end{bmatrix} \quad (16)$$

In this work, when training the RBF network, the data points  $x_i$  are the samples of the parameters  $\mu_i$ , and  $f(x)$  corresponds to each entry of the solution vector  $u(t, \mu)$  at any time instance  $t_k \leq T_0$  and any training sample of  $\mu$ , i.e., each entry in the snapshot matrices  $\mathbf{X}_0$  and  $\mathbf{X}_1$  in (3). It will be further clarified in later sections.

---

**Algorithm 3** RBF network construction
 

---

- 1: Choose the appropriate kernel function  $\kappa$  and its shape factor  $\varepsilon$  when necessary.
  - 2: Compute the Euclidean distance  $r = \|x - x_i\|$  between data points.
  - 3: Compute the radial basis function  $\kappa(r)$  and construct the coefficient matrix in (16).
  - 4: Determine the coefficients  $w_i$  by solving a linear system in (16).
- 

The kernel functions can be chosen widely, such as splines, Gaussian, Multi-quadrics, and so on. Table 1 provides a chart with some commonly used basis functions. In this work, inverse multi-quadrics (IMQ) is used and its shape factor  $\varepsilon$  is set as 1/30.

### 3.2 Parametric DMD framework

In this section, we propose the parametric DMD framework. After the collection of snapshots at limited samples of training parameters, the RBF network is first trained using these snapshots. The trained RBF network can then predict snapshots at any new parameter. After the snapshot matrices  $\mathbf{X}_0, \mathbf{X}_1$

Linear splines	$\ x - x_i\ $
Thin plate splines	$\ x - x_i\ ^k \ln \ x - x_i\ ; k \in [2, 4, \dots]$
Cubic splines	$\ x - x_i\ ^3$
Gaussian	$\exp\left(-\varepsilon^2 \ x - x_i\ ^2\right)$
Multi-quadrics	$\left(1 + \varepsilon^2 \ x - x_i\ ^2\right)^{1/2}$
Inverse multiquadrics	$\left(1 + \varepsilon^2 \ x - x_i\ ^2\right)^{-1/2}$

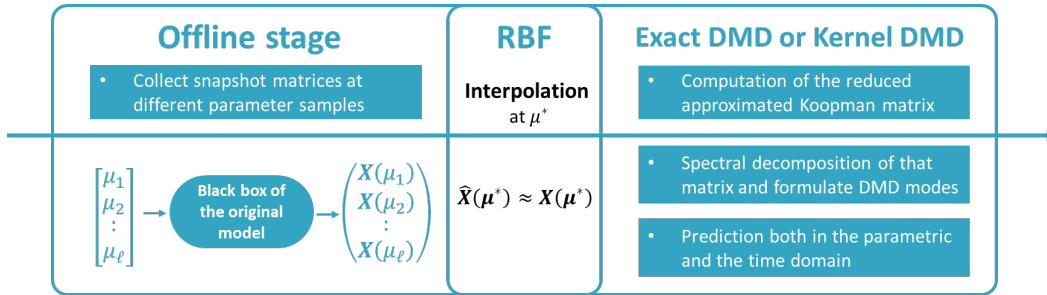
Table 1: Some commonly used RBFs [25],  $x_i$  denotes the  $i$ -th center of the RBF.

Figure 2: Flowchart of the parametric DMD algorithm.

corresponding to the new parameter is computed, then DMD is implemented on the new snapshot matrices to generate the DMD modes for predicting the solution in the time domain. The whole flow chart of the parametric DMD framework can be seen in Figure 2. At the offline stage, the snapshot matrices  $\mathbf{X}(\mu_k) := [u_0(\mu_k), \dots, u_m(\mu_k)]$  corresponding to different samples  $\mu_k, k = 1, \dots, \ell$  of the parameter  $\mu$  are first computed via, e.g., black-box simulation of a dynamical system. These are used as training data for the RBF network. Then the RBF network is used to construct an approximate function  $\hat{\mathbf{X}}_{ij}(\mu) : \mu \mapsto \mathbb{R}$  for each entry  $\mathbf{X}_{ij}(\mu), i = 1, \dots, n, j = 1, \dots, m + 1$  of a snapshot matrix function  $\mathbf{X}(\mu)$ . More specifically,  $\hat{f}(x)$  in (15) now becomes  $\hat{\mathbf{X}}_{ij}(\mu)$ , and  $f(x)$  is now  $\mathbf{X}_{ij}(\mu)$ . The RBF network is used to learn the  $i, j$ -th entry of  $\mathbf{X}(\mu)$  using the data  $\mathbf{X}_{ij}(\mu_k)$ , i.e., the  $i, j$ -th entry of the snapshot matrices  $\mathbf{X}(\mu_k)$  at the  $\ell$  parameter samples  $\mu_k, k = 1, \dots, \ell$ . The weights  $w_i$  in (15) are computed once for each entry  $\mathbf{X}_{ij}(\mu)$ . After the weights are computed, the RBF network  $\hat{\mathbf{X}}_{ij}(\mu)$  for the  $i, j$ -th entry is trained and is ready to be used at the online stage. The predicted snapshot matrix at  $\mu^*$  is nothing but  $\hat{\mathbf{X}}(\mu^*)$ .

At the online stage, instead of repeated black-box simulation of the large-scale model in (14), the maps  $\hat{\mathbf{X}}_{ij}(\mu)$  constructed by the RBF networks are called to compute the approximated snapshot matrix  $\hat{\mathbf{X}}(\mu^*)$  at any new parameter sample  $\mu^*$ .  $\hat{\mathbf{X}}(\mu^*)$  is then split into two snapshot matrices  $\hat{\mathbf{X}}_0 \in \mathbb{R}^{n \times m}$  and  $\hat{\mathbf{X}}_1 \in \mathbb{R}^{n \times m}$ . For example, if  $\hat{\mathbf{X}}(\mu^*) \in \mathbb{R}^{n \times m+1}$  approximates  $\mathbf{X}(\mu^*) := [u_0(\mu^*), \dots, u_m(\mu^*)]$ , then  $\hat{\mathbf{X}}_0(\mu^*) = \hat{\mathbf{X}}(\mu^*)[:, 1:m]$ , and  $\hat{\mathbf{X}}_1(\mu^*) = \hat{\mathbf{X}}(\mu^*)[:, 2:m+1]$ . Here we use the MATLAB notation for matrix blocks. The exact DMD or the kernel DMD is then applied to  $\hat{\mathbf{X}}_0(\mu^*)$  and  $\hat{\mathbf{X}}_1(\mu^*)$  to predict the time evolution of the solution corresponding to  $\mu^*$ . In summary, the RBF networks are used to predict the dynamics in the parameter domain and the DMD is employed for the time-evolution prediction. This process of parametric DMD is detailed in Algorithm 4.



**Algorithm 4** Parametric DMD framework for parametric DMD

- 1: Sampling (equidistant or random) the parameters in the range of the parameter space.
- 2: Generate the snapshot matrices  $\mathbf{X}(\mu_i)$  for each parameter sample  $\mu_i$ ;
- 3: Train the RBF network for each entry of  $\mathbf{X}(\mu)$  by [Algorithm 3](#), and get the approximate snapshot matrix function  $\hat{\mathbf{X}}(\mu)$  in the form of RBF networks.
- 4: For any new parameter sample  $\mu^*$ , evaluate the approximate snapshot matrix function at  $\mu^*$  and take  $\hat{\mathbf{X}}(\mu^*)$  as the new snapshot matrix corresponding to  $\mu^*$ . Split it into  $\hat{\mathbf{X}}_0(\mu^*)$  and  $\hat{\mathbf{X}}_1(\mu^*)$ .
- 5: Apply the exact DMD [Algorithm 1](#) or the kernel DMD [Algorithm 2](#) to  $\hat{\mathbf{X}}_0(\mu^*)$  and  $\hat{\mathbf{X}}_1(\mu^*)$  to reconstruct and predict the time-evolution of the dynamics corresponding to  $\mu^*$ .

## 4 Numerical examples

In this section, we test the performance of the proposed parametric DMD method with some models from engineering applications. Two examples are related to electrochemical processes. The first one considers lithium-ion battery model. Lithium-ion batteries are of high importance in the context of electromobility. Understanding of their dynamics is of high interest. The second example is a ferrocyanide reduction oxidation reaction. This is a common model system in electrochemistry which exemplifies diffusion controlled fast electrochemical process. As the last example, the FitzHugh-Nagumo model is a prototype of an excitable system, for example, a neuron. A common feature of all the models is that they are systems with parameters and time-varying inputs that can be considered as time-varying parameters. In the following subsections, we discuss the numerical tests on each of them separately. In all the figures for the numerical result, ‘‘RBF-DMD’’ represents parametric DMD, and ‘‘reference’’ refers to the solution computed by directly simulating the original model. According to the error computation in the numerical examples, we use relative error at any testing parameter  $\mu^*$  defined as follows:

$$\epsilon_i(t, \mu^*) = \frac{|y_i(t, \mu^*) - \hat{y}_i(t, \mu^*)|}{\max_{0 \leq t_j \leq T} |y_i(t_j, \mu^*)|}. \quad (17)$$

Here the index  $i$  means the  $i$ -th output, i.e., the  $i$ -th entry of  $y(t, \mu^*) \in \mathbb{R}^{n_0}$ . To evaluate the performance of the proposed method in the parameter domain, the time-average relative error is used and is defined as:

$$\epsilon_i^{ave}(\mu^*) = \frac{1}{n_T} \sum_{j=0}^{n_T-1} \epsilon_i(t_j, \mu^*). \quad (18)$$

As for the computation time, on the one hand, the snapshot generation and the RBF training are run only once at the offline stage. On the other hand, the runtimes of the RBF prediction, the DMD prediction and the FOM simulation at the online phase are respectively the average values of the runtimes over all the testing parameters.

### 4.1 Lithium-ion Battery Model

As an example for validating of the proposed methodology, we consider the widely implemented yet complicated mathematical model of a lithium-ion battery, the pseudo-two-dimensional (P2D) battery model, which was previously introduced in [\[15\]](#). [Figure 3a](#) depicts the schematic of the P2D battery model. As the name suggests, the P2D battery model comprises two modelling scales: the computation of lithium concentration and potential gradients across the battery model (macro-scale) as well as the diffusion of lithium ions within the electrode (micro-scale).

Further complexities arise in the P2D battery model when one considers a distribution of different particle sizes in the electrode (anode), which has been introduced by Röder et al. [\[40\]](#). Considering different sizes of the solid particles, the battery dynamics at a wider operational condition can be

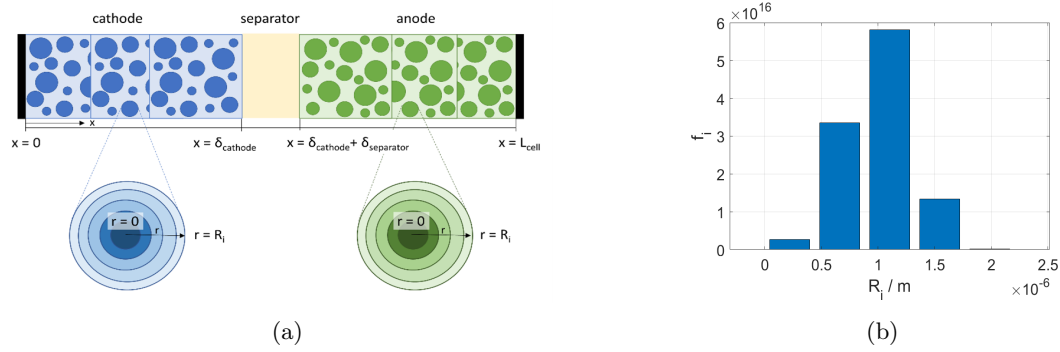


Figure 3: Mathematical model of the lithium-ion battery. (a) Schematic of the modelling domains of the battery model. (b) Particle size distribution of electrode (anode) assumed in P2D battery model simulation.

better reproduced via a model-based approach. Here, we assume the particle size distribution within the electrode follows a Weibull distribution density defined as:

$$h(R_i; c_1, c_2) = c_1 c_2 (c_1 R_i)^{c_2 - 1} e^{-(c_1 R_i)^{c_2}}, \quad (19)$$

where  $c_1$  is the scaling factor and  $c_2$  is the form factor of the distribution density.  $R_i$  is the particle radius of the  $i$ -th particle size class in the electrode. Figure 3b shows the simulated particle size distribution of the electrode with five different radius classes with  $c_1 = 9.064 \times 10^5$  and  $c_2 = 4$ . Summing up the surface and volume densities across every particle radius class yields the surface area ratio as well as the volume fractions of the total active materials in the battery.

$$\begin{aligned} a_s &= \int_0^\infty f_{\text{area}}(R_i) dR_i \\ \varepsilon_s &= \int_0^\infty f_{\text{vol}}(R_i) dR_i \end{aligned} \quad (20)$$

The governing equations for the P2D-PSD model are derived from the conservation laws of species and charge transport. The governing equations of the P2D battery model incorporating the effect of particle size distribution are detailed in Table 2.

It is also seen that all the governing equations are coupled with each other. Due to the high complexity of the coupled governing equations, it is almost impossible to extract the discretized system matrices and nonlinear terms from the spatial discretization of the PDEs given all the parameters are fixed, not to mention their parametrized forms. Consequently, projection-based MOR methods cannot be applied to MOR for this model, and the non-intrusive MOR is the only possible choice. That leads to the application of our proposed parametric DMD to this example. The input  $I(\omega, t)$  is the current with a certain frequency  $\omega \in [10^{-2}, 10^4]$  and the output  $E(\omega, t)$  is the voltage, which can be shown as the difference of potential at the current collectors between the anode and the cathode. Both are shown in (21). The whole general in-output model is shown in Figure 4.

$$\begin{aligned} I(\omega, t) &= \frac{I}{A_{\text{cell}}} \sin(\omega t) \\ E(\omega, t) &= \phi_s(x=0) - \phi_s(x=L_{\text{cell}}) \end{aligned} \quad (21)$$

The original spatially discretized ODE model has  $n = 325$  states. We use 100 snapshot matrices corresponding to 100 frequency training samples  $\omega_i, i = 1, \dots, 100$  with 10-base logarithmic spacing in  $[-2, 4]$ . At the offline stage, the RBF network is trained with these parameters. The snapshots corresponding to each frequency sample in a limited time interval  $[0, T_0]$  are computed by an ODE

Table 2: Governing equations of the P2D battery model with the corresponding boundary conditions [28]. Subscript  $i$  describes the  $i^{th}$  particle size classes,  $s$  for solid phase,  $e$  for electrolyte phase.

Model equations	Boundary conditions
$\frac{\partial c_s(r, R_i)}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left( D_{s,a/c} \cdot r^2 \cdot \frac{\partial c_s(r, R_i)}{\partial r} \right)$	$\frac{\partial c_s(r=0, R_i)}{\partial r} = 0$ $\frac{\partial c_s(r=R_i, R_i)}{\partial r} = \frac{j^{Li}(x, R_i)}{a_s D_s F}$
$\varepsilon_e \frac{\partial C_e(x)}{\partial t} = \frac{\partial}{\partial x} \left( D_{e,eff} \cdot \frac{\partial C_e(x)}{\partial x} \right) + (1 - t_p) \cdot \frac{j^{Li}(x)}{F}$	$\frac{\partial c_e(x=0)}{\partial x} = 0$ $\frac{\partial c_e(x=L_{cell})}{\partial x} = 0$
$J_s(x) = -\sigma_s \varepsilon_s \frac{\partial \phi_s(x)}{\partial x}$ $\frac{\partial J_s(x)}{\partial x} = - (j^{Li}(x) + j^{DL}(x))$	$\frac{\partial J_s(x=0)}{\partial x} = a_s \frac{I}{A_{cell}}$ $\frac{\partial J_s(x=L_{cell})}{\partial x} = -a_s \frac{I}{A_{cell}}$
$J_e(x) = -\sigma_e(x) \frac{\varepsilon_e}{\tau} \frac{\partial \phi_e(x)}{\partial x} - \sigma_{De} \frac{\varepsilon_e}{\tau} \frac{\partial \ln(c_e(x))}{\partial x}$ $\frac{\partial J_e(x)}{\partial x} = j^{Li}(x) + j^{DL}(x)$	$\phi_e(x=0) = 0$ $\frac{\partial \phi_e(x=L_{cell})}{\partial x} = 0$
$j^{DL}(x) = a_s C_{DL} \frac{\partial (\Delta \phi(x))}{\partial t}$	
$j^{Li}(x, R_i) = a_{s,j}(R_i) j_0(R_i) \left( \exp\left(\frac{\alpha \eta(x, R_i) F}{RT}\right) - \exp\left(\frac{(1-\alpha) \eta(x, R_i) F}{RT}\right) \right)$ $\eta(x, R_i) = \phi_s(x) - \phi_e(x) - U(x, R_i)$ $j^{Li}(x) = \int j^{Li}(x, R_i) dR_i$	

Table 3: Parameter set for P2D battery model simulation.

Symbol	Parameter	Unit	Value
$R$	Gas constant	$Jmol^{-1}K^{-1}$	8.314
$F$	Faraday constant	$Cmol^{-1}$	96485
$T$	Temperature	$K$	298
$R_c$	Radius of cathode	$\mu m$	1
$\delta_{anode}$	Anode's thickness	$\mu m$	50
$\delta_{separator}$	Separator's thickness	$\mu m$	26.4
$\delta_{cathode}$	Cathode's thickness	$\mu m$	25.4
$D_{s,anode}$	Diffusion coefficient anode	$m^2s^{-1}$	$2 \times 10^{-16}$
$D_{s,cathode}$	Diffusion coefficient cathode	$m^2s^{-1}$	$3.7 \times 10^{-16}$
$\tau_{anode}$	Tortuosity anode	-	3.67
$\tau_{separator}$	Tortuosity separator	-	1.4
$\tau_{cathode}$	Tortuosity cathode	-	3.67
$\varepsilon_{e,anode}$	Volume fraction electrolyte anode	-	0.33
$\varepsilon_{e,cathode}$	Volume fraction electrolyte cathode	-	0.33
$\varepsilon_{e,separator}$	Volume fraction electrolyte separator	-	0.5
$\varepsilon_{s,cathode}$	Volume fraction cathode	-	0.5
$\alpha$	Charge transfer coefficient	-	0.5
$C_{DL,anode}$	Double layer capacitance anode	$Fm^{-2}$	0.2
$C_{DL,cathode}$	Double layer capacitance cathode	$Fm^{-2}$	0.2
$t_p$	Transference number	-	0.37
$\sigma_{s,anode}$	Electrical conductivity anode	$Sm^{-1}$	100
$\sigma_{s,cathode}$	Electrical conductivity cathode	$Sm^{-1}$	10

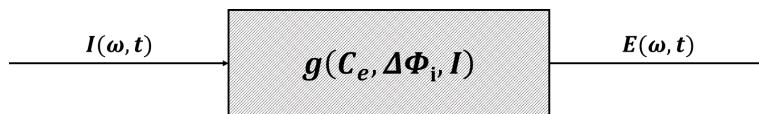


Figure 4: Graphical model of a Lithium-ion battery.

solver: ode15s in MATLAB. Here  $T_0 = T/2$ , with  $T$  being the final simulation time. That means the original model is simulated till half of the final simulation time to get the snapshot matrices. The dynamics corresponding to time span  $[0, T]$  at any testing frequency will be predicted. At the online stage, the snapshot matrix function  $\hat{\mathbf{X}}(\omega)$  in the form of RBF networks is evaluated at a new frequency sample  $\omega^*$  to get an approximate snapshot matrix  $\hat{\mathbf{X}}(\omega^*)$  that is considered as the new snapshot matrix. DMD is then applied to  $\hat{\mathbf{X}}(\omega^*)$  to predict the output voltage  $E(\omega, t)$  at any time  $t > T_0$ .

Exact DMD is employed in the proposed parametric DMD for this model. The results are derived by 14 dominant DMD modes, i.e.,  $r = 14$  in Algorithm 1. The time-evolution of the output voltage at  $\omega^*$  computed by the ODE solver is considered as the reference solution. Both the reference solution and the output computed by the parametric DMD are presented in Figure 5. The RBF-DMD solution is the voltage derived by the proposed parametric DMD. The voltage in  $[0, T_0]$  is predicted by the RBF network. Based on this, DMD then predicts the evolution in  $[T_0, T]$ . The relative error between the reference voltage and the RBF-DMD voltage is presented in Figure 6. It can be observed that the maximal relative error is under 0.03%. The plot for the time-average relative error at different testing frequencies  $\omega^* = 0.025, 0.1, 0.271, 1, 2.239, 3.690, 10, 100, 1000, 3689.776 \text{ Hz}$  is shown in Figure 7. RBF-DMD predicts the voltage with no more than 0.007% relative error compared to the reference solution both in low and high testing frequencies. Electrochemical impedance spectroscopy (EIS) is commonly used to monitor the performance of the lithium-ion battery. When the input is the current with different frequencies, the output voltage is transformed from time to frequency domain by Fast Fourier Transformation (FFT) to analyse the model. In this example, the results of EIS are shown in Nyquist and Bode diagrams, see Figure 8. The subfigure above is the Nyquist plot presenting the imaginary part of the complex impedance as a function of the real part of it. It can be observed that there exists a semicircle at the high frequency range and a non-vertical line at the intermediate frequency range, which can be interpreted as the resistance of the electrolyte and the resistance of the diffusive layer respectively in the practical application. In this subfigure, the complex impedance computed by parametric DMD at the new frequency  $\omega^* = 3.69 \text{ Hz}$  conforms to the pattern from the reference solution. The bottom-left one shows the relationship between the impedance and the frequency, while the bottom-right one is the phase shift changing with the frequency. Both subfigures show the great matching between the solution from the parametric DMD and the reference solution at testing frequency  $\omega^*$ . We can conclude that the proposed method delivers satisfactory accuracy in the parameter space and the time domain.

The runtime comparison for this example is shown in Table 4. The computation time of parametric DMD includes the offline stage and the online stage. The offline stage of computing all the snapshots at 100 training samples takes 171.581 seconds. Training the RBF network at the offline stage takes 0.445 seconds. The online RBF prediction at a new parameter sample in the training time interval costs 0.007 seconds. The online DMD prediction in the future time interval takes 1.697 seconds. Computing the reference solution at one testing sample of  $\omega^*$  via ODE solver i.e., the FOM simulation needs 4.787 seconds. The online speed-up is around 2–3 times faster. It is clear that if the original model needs to be simulated to get the output response at more than 40 different values of  $\omega$ , the proposed parametric DMD method will outperform the direct simulation without MOR.

## 4.2 Coupled electrochemical kinetics and diffusion model

This section presents the performance of the parametric DMD on a model of the ferrocyanide redox reaction. The reaction kinetics under the influence of the rotation rate of the rotating disc electrode is of

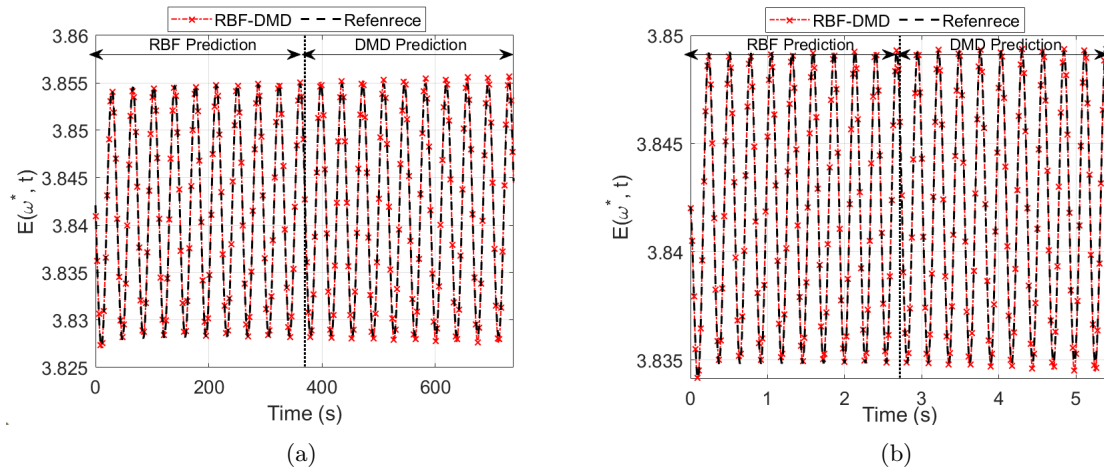


Figure 5: Lithium-ion battery model: the parametric DMD solution vs the reference solution for  $E(\omega^*, t)$ . (a)  $\omega^* = 0.025 \text{ Hz}$ . (b)  $\omega^* = 3.69 \text{ Hz}$ .

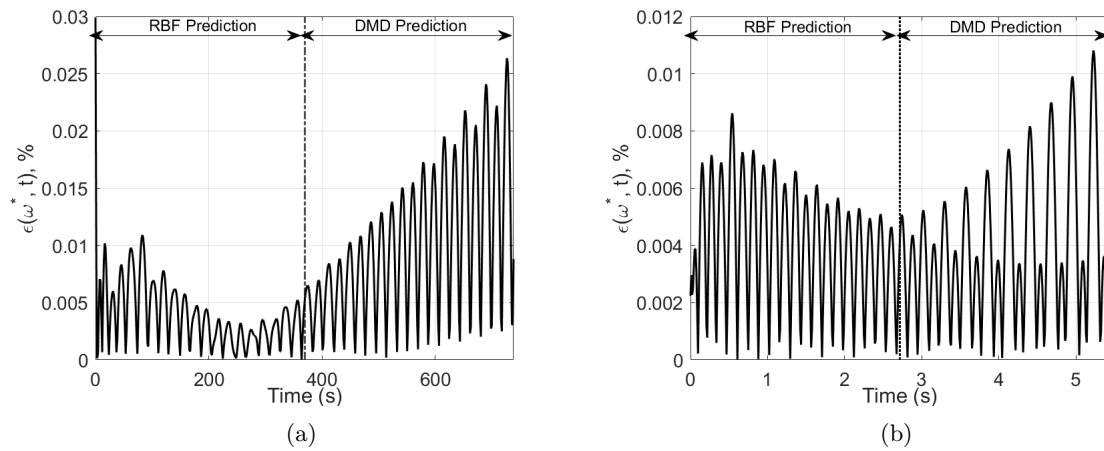


Figure 6: Lithium-ion battery model: the relative error of the parametric DMD solution for  $E(\omega^*, t)$ . (a)  $\omega^* = 0.025 \text{ Hz}$ . (b)  $\omega^* = 3.69 \text{ Hz}$ .

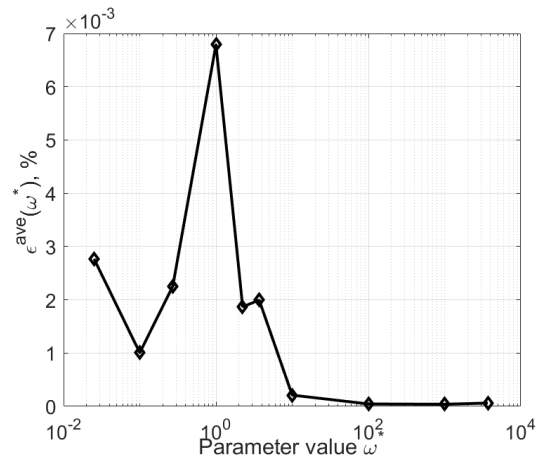


Figure 7: Lithium-ion battery model: the time-average relative error of the parametric DMD for  $E(\omega^*, t)$  at different testing frequencies  $\omega^*$ .

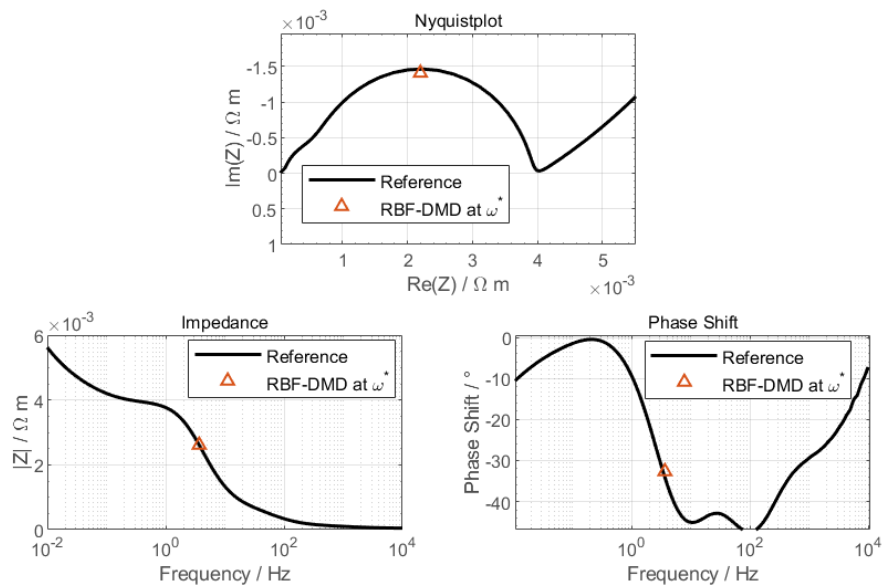


Figure 8: Lithium-ion battery model: Nyquist and bode representations of the parametric DMD solution at  $\omega^* = 3.69 \text{ Hz}$  vs the reference solution.

Table 4: Lithium-ion battery model: The computation time (seconds) of parametric DMD and that of the FOM simulation.

Snapshot generation	RBF training	RBF prediction	DMD prediction	FOM simulation
171.581	0.445	0.007	1.697	4.787

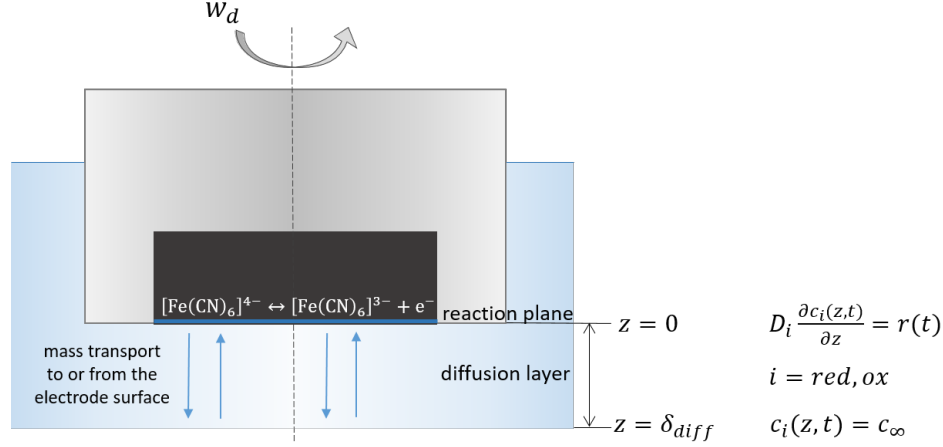


Figure 9: Schematic of coupled electrochemical kinetics and diffusion model

interest [46]. A schematic representation of the investigated system is shown in Figure 9. This reaction can be considered as a model reaction with coupled electrochemical kinetics and mass transport. Similar to the first battery model, the governing equations of this model are based on mass and charge conservation laws as well.

The mass conservation law is described by the second Fick's law assuming that convective term can be neglected, see (22).

$$\frac{\partial c_i(z, w_d, t)}{\partial t} = D_i \frac{\partial^2 c_i(z, w_d, t)}{\partial z^2}, i = red, ox, \quad (22)$$

where the subscript  $i$  stands either for the reduced (ferrocyanide,  $[\text{Fe}(\text{CN})_6]^{4-}$ ) or oxidized (ferricyanide,  $[\text{Fe}(\text{CN})_6]^{3-}$ ) form, and  $c_i, D_i$  are their corresponding concentration and diffusion coefficients, respectively.

The charge balance can be described as,

$$C_{dl} \frac{dE(w_d, t)}{dt} = J(w_d, t) - Fr(w_d, t), \quad (23)$$

where  $E(w_d, t)$  is the electrode potential,  $C_{dl}$  is the double-layer capacitance,  $J(w_d, t)$  is the cell current density,  $F$  is Faraday constant, and  $r(w_d, t)$  is the nonlinear reaction rate, computed by Butler-Volmer kinetics,

$$r(w_d, t) = k \left\{ \frac{c_{red}(0, w_d, t)}{c_{red, \infty}} \exp(\beta f (E(w_d, t) - E_r)) - \frac{c_{ox}(0, w_d, t)}{c_{ox, \infty}} \exp(-(1 - \beta) f (E(w_d, t) - E_r)) \right\}. \quad (24)$$

Here,  $E_r$  is the equilibrium electrode potential,  $\beta$  is the charge transfer coefficient, and  $f$  is determined as  $F/RT$ , where  $T$  is the temperature, and  $R$  is the universal gas constant.

Table 5: Parameters in the model for ferrocyanide reaction.

Parameters	Variables	Value Range
rotation rate	$w_d$ (rpm)	[500, 5000]
input potential	$E_{total}$ (V)	[-0.4, 1.0]
double layer capacity	$C_{dl}$ (F/m <sup>2</sup> )	0.2
charge transfer coefficient	$\beta$	0.5
reaction rate constant	$k$ (m/s)	[10 <sup>-7</sup> , 10 <sup>-2</sup> ]
ohmic resistance of the electrolyte	$R_{\Omega}$ ( $\Omega$ )	[1, 100]
diffusivity coefficient for the ferrocyanide	$D_{red}$ (m <sup>2</sup> /s)	[10 <sup>-10</sup> , 9 × 10 <sup>-10</sup> ]
diffusivity coefficient for the ferricyanide	$D_{ox}$ (m <sup>2</sup> /s)	[10 <sup>-10</sup> , 9 × 10 <sup>-10</sup> ]

The main source of the nonlinearity comes from  $r(t)$  and its coupling with the diffusion of the reacting species through the boundary conditions (given in [Figure 9](#) and [\(25\)](#)).

$$D_i \frac{\partial c_i(z, w_d, t)}{\partial z} \Big|_{z=0} = \pm r(w_d, t), \quad i = \text{red or ox} . \quad (25)$$

We list in [Table 5](#) all the important parameters used to construct the model and their ranges of change.

We study the influence of the rotation rate  $w_d$  ([Figure 9](#)) on the system output (current density  $J(w_d, t)$ ). The rotation speed  $w_d$  of the rotating disc electrode determinates the thickness of diffusion layer  $\delta_{diff}$  for ox or red, as shown below:

$$\delta_{diff} = 1.61 D_i^{1/3} \nu^{1/6} \omega_d^{-1/2}, \quad i = \text{red or ox} , \quad (26)$$

where  $\nu$  is the kinematic viscosity. The thickness of the diffusion layer further has impacts on the concentration in [\(22\)](#) and its boundary as  $c_i(\delta_{diff}, t) = c_{i,\infty}$ ,  $i = \text{red or ox}$  (see also [Figure 9](#)). The equation in [\(23\)](#) and [\(24\)](#) are discretized in space using finite difference method. The dimension of the discretized system is  $n = 4003$  while the simulation time is set as 10s with 10 periods. 20 different rotation rates as training parameters are uniformly sampled in the range of [500, 5000] rpm. The kernel DMD with Gaussian kernel in our parametric DMD method ([Algorithm 2](#)) is selected in this example.  $r$  in step 2 of [Algorithm 2](#) is chosen according to the criteria in [\(8\)](#) with  $\eta = 0.5\%$ . [Figure 10](#) presents the current density computed by the parametric DMD and the reference solution. The relative error changing with time at two testing samples of  $w_d$  and the time average relative errors at 10 different testing rotation rates are plotted in [Figure 11](#) and [Figure 12](#) respectively. In [Figure 11](#), the relative errors at all time instances are all below 5%. In [Figure 12](#), the time average relative error at all testing  $w_d^*$  samples, i.e.,  $w_d^* = 600, 800, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4800$  rpm are under 0.7% when using parametric DMD.

The runtime comparison for this model is listed in [Table 6](#). At the offline stage, generating snapshots and training RBF network take 1452.29 seconds and 0.082 seconds, respectively. The online runtime is the average value over 10 testing samples of different rotation rates. RBF predicts the current in  $[0, T_0]$  using 0.082 seconds and in  $[T_0, T]$ , DMD uses 7.312 seconds. The total runtime at the online stage is around 7.4 seconds, which is much less than that of solving the original system (FOM simulation) by an ODE solver with 77.703 seconds.

### 4.3 FitzHugh–Nagumo model

We further consider the nonlinear Fitz–Hugh Nagumo model as a benchmark example used in many existing works [[1](#), [3](#), [4](#), [12](#)]. This model is designed to simulate the spike generation in an excitable



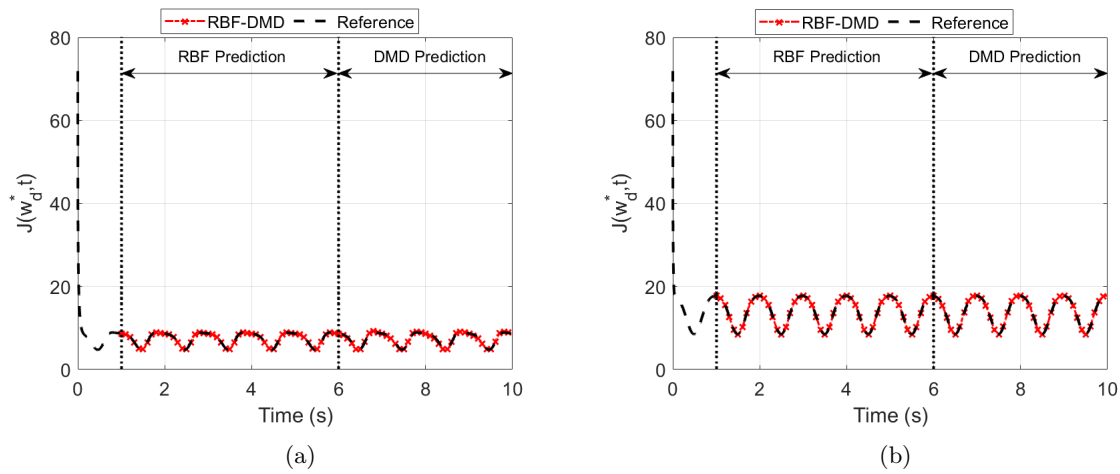


Figure 10: Ferrocyanide reaction model: the parametric DMD solution vs the reference solution for the current density  $J(w_d^*, t)$ . (a)  $w_d^* = 1000 \text{ rpm}$ . (b)  $w_d^* = 4800 \text{ rpm}$ .

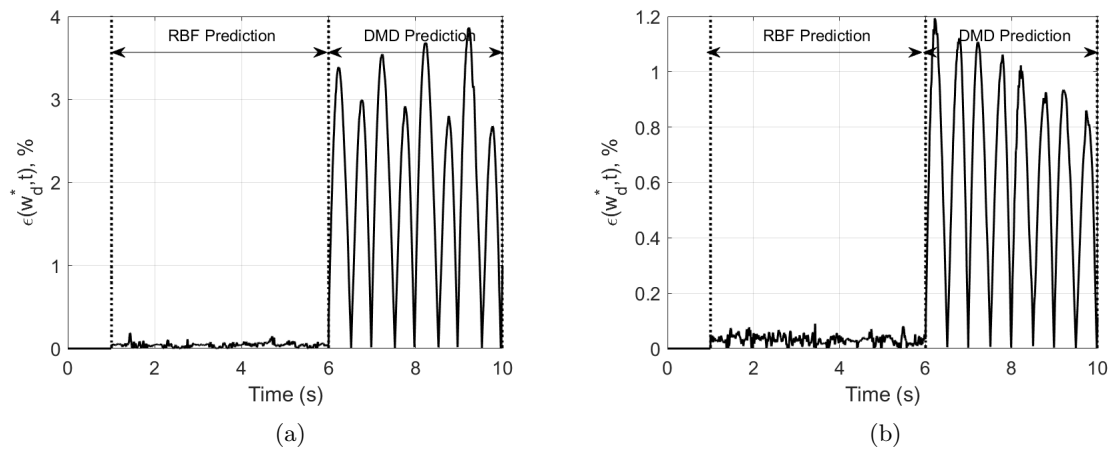


Figure 11: Ferrocyanide reaction model: the relative error of the parametric DMD solution for the current density  $J(w_d^*, t)$ . (a)  $w_d^* = 1000 \text{ rpm}$ . (b)  $w_d^* = 4800 \text{ rpm}$ .

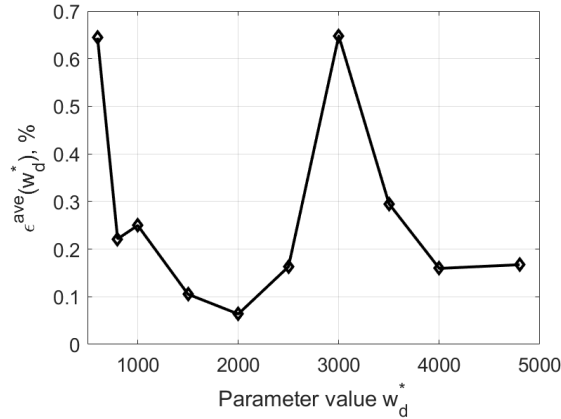


Figure 12: Ferrocyanide reaction model: the time-average relative error of the parametric DMD solution for the current density  $J(w_d^*, t)$  at different testing rotation rates  $w_d^*$ .

Table 6: Ferrocyanide reaction model: The computation time (seconds) of parametric DMD and that of the FOM simulation.

Snapshot generation	RBF training	RBF prediction	DMD prediction	FOM simulation
1452.290	2.453	0.719	6.202	99.825

system, for example in a neuron.

$$\begin{aligned}
 \varepsilon v_t(x, \varepsilon, t) &= \varepsilon^2 v_{xx}(x, \varepsilon, t) + f(v(x, \varepsilon, t)) - w(x, \varepsilon, t) + c \\
 w_t(x, \varepsilon, t) &= bv(x, \varepsilon, t) - \gamma w(x, \varepsilon, t) + c \\
 y(x, \varepsilon, t) &= [v(0, \varepsilon, t), w(0, \varepsilon, t)]^T,
 \end{aligned} \tag{27}$$

with  $f(v) = v(v - 0.1)(1 - v)$  as the cubic nonlinear term and the boundary conditions are:

$$\begin{aligned}
 v(x, \varepsilon, 0) &= 0, & w(x, \varepsilon, 0) &= 0, & x &\in [0, L], \\
 v_x(0, \varepsilon, t) &= -i_o(t), & v_x(L, \varepsilon, t) &= 0, & t &\geq 0,
 \end{aligned}$$

The unknown state variable,  $v(x, \varepsilon, t)$  is the membrane potential, and  $w(x, \varepsilon, t)$  is a recovery of the potential. Parameters are  $b, c, \varepsilon$  and  $\gamma$ . In this numerical test, the operating parameter is  $\varepsilon$ , changing from 0.02 to 0.03, while other parameters are fixed as  $L = 20$ ,  $b = 0.5$ ,  $c = 0.05$  and  $\gamma = 2$ . The input term  $i_o(t) = 50000t^3 e^{-15t}$ . The output vector  $y(x, \varepsilon, t) \in \mathbb{R}^2$  includes two outputs: the membrane potential and the recovery of the potential at the left boundary.

After discretization by the finite difference method, the resulting ODE is solved by an ODE solver `ode15s` in Matlab. The total number of states is  $n = 16384$ . The time span is  $[0, 10]s$  with the time step  $\delta t = 0.01s$ . The snapshots are taken in the time interval  $[0, 8]s$ . The number of the equidistant samples in  $[0.02, 0.03]$  in the training phase is 15. For this example, kernel DMD is chosen in [Algorithm 2](#).

The numerical results are shown in [Figure 13](#). [Figure 13a](#) and [Figure 13c](#) show the evolution of the two outputs  $v(0, \varepsilon^*, t)$  and  $w(0, \varepsilon^*, t)$  when  $\varepsilon^* = 0.0225$  and  $\varepsilon^* = 0.0275$ . As is shown in these figures, at the online stage of the proposed parametric DMD, RBF first predicts the solution at the testing  $\varepsilon^*$  in the time interval  $[0, 8]s$ , then DMD predicts the evolution of the solution in the time period  $[8, 10]s$ . The red line is the parametric DMD results for  $v(0, \varepsilon^*, t)$  and the blue line stands for  $w(0, \varepsilon^*, t)$ . Both lines fit quite well with the black reference solution. [Figure 13b](#) and [Figure 13d](#) are their corresponding phase-space diagrams. [Figure 14](#) is the relative error changing

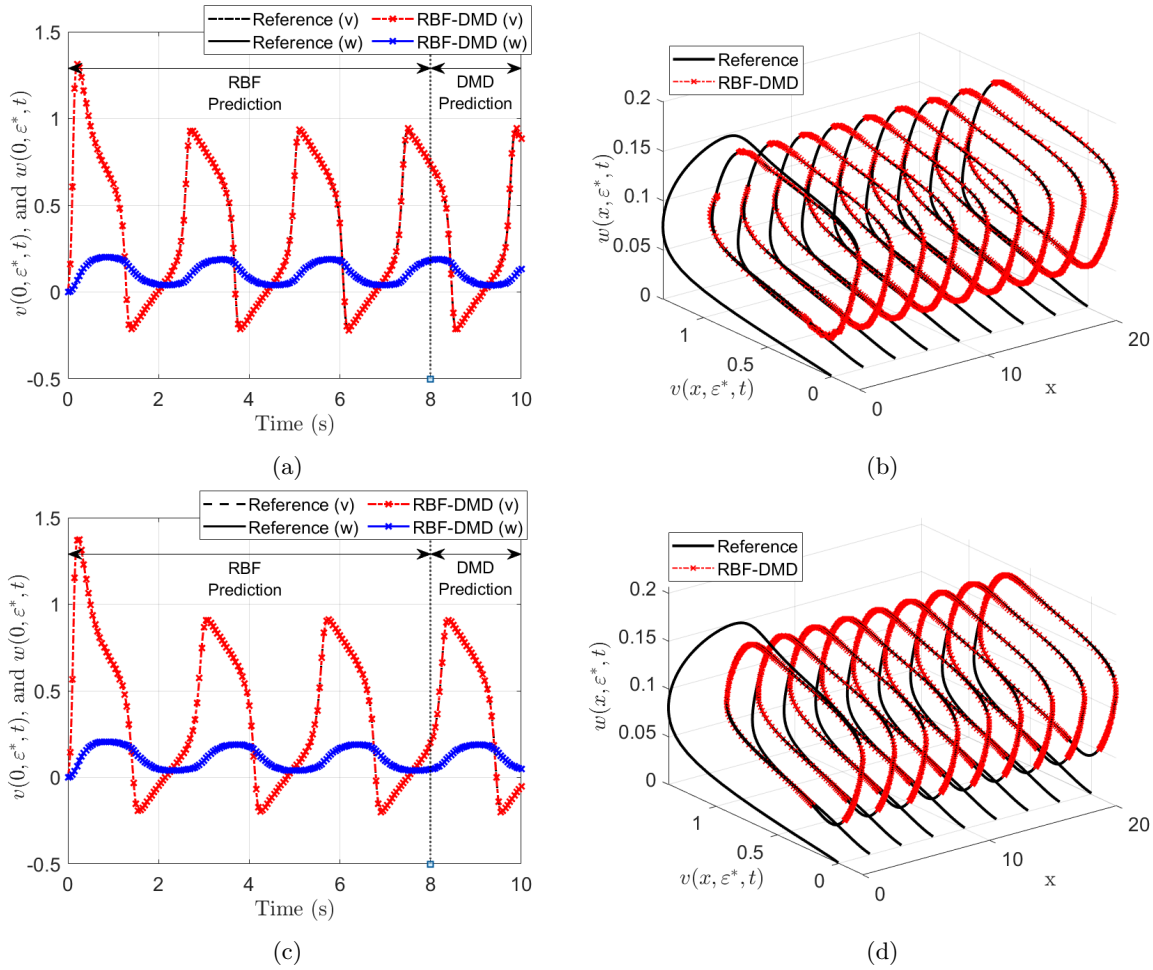


Figure 13: FitzHugh-Nagumo model: the parametric DMD solution vs the reference solution. (a) The outputs  $v(0, \varepsilon^*, t)$  and  $w(0, \varepsilon^*, t)$  when  $\varepsilon^* = 0.0225$ . (b) Limit cycles of  $v(x, \varepsilon^*, t)$  and  $w(x, \varepsilon^*, t)$  when  $\varepsilon^* = 0.0225$ . (c) The outputs  $v(0, \varepsilon^*, t)$  and  $w(0, \varepsilon^*, t)$  when  $\varepsilon^* = 0.0275$ . (d) Limit cycles of  $v(x, \varepsilon^*, t)$  and  $w(x, \varepsilon^*, t)$  when  $\varepsilon^* = 0.0275$ .

with time when  $\varepsilon^* = 0.0225$  and  $\varepsilon^* = 0.0275$ . The maximum relative error of these two cases is around 2 – 3.5%. Figure 15 is the time average of the relative errors over all testing parameters, i.e.,  $\varepsilon^* = 0.021, 0.0225, 0.024, 0.0245, 0.0252, 0.027, 0.0275, 0.029$ . Their values never exceeds 1% in all these testing cases. Through these error plots, it can be confirmed that the proposed method works well for this nonlinear dynamic system.

The computation time is also listed in Table 7. 12656.380 seconds are needed at the offline stage for generating training snapshots for training samples. RBF training takes 1.395 seconds based on these training samples. At the online stage, the RBF prediction costs 0.5435 seconds and the DMD prediction costs 0.694 seconds. Solving the original full order model for a single testing parameter takes around 1057.4 seconds. It can be observed that around 850 times speed-up is achieved at the online phase when using parametric DMD.

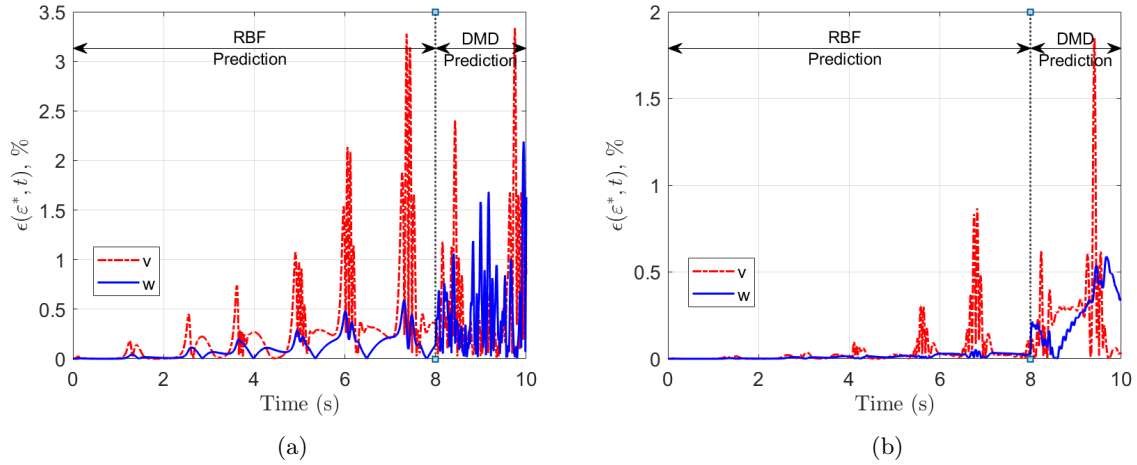


Figure 14: FitzHugh-Nagumo model: the relative error of the parametric DMD for  $v(0, \epsilon^*, t)$  and  $w(0, \epsilon^*, t)$ . (a)  $\epsilon^* = 0.0225$ . (b)  $\epsilon^* = 0.0275$ .

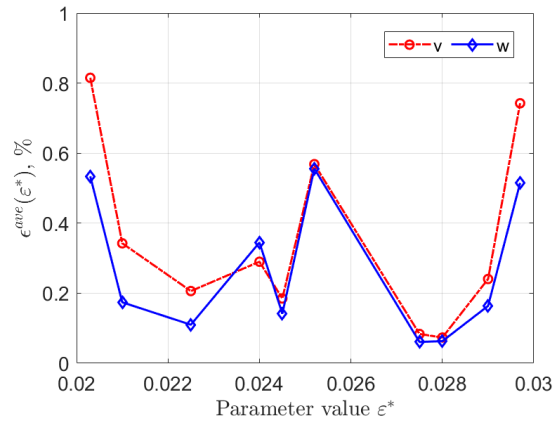


Figure 15: FitzHugh-Nagumo model: the time-average relative error of the parametric DMD solution at different testing  $\epsilon^*$ .

Table 7: FitzHugh-Nagumo model: The computation time (seconds) of parametric DMD and that of the FOM simulation.

Snapshot generation	RBF training	RBF prediction	Online prediction	FOM simulation
12656.380	1.395	0.5435	0.694	1057.400

## 5 Conclusion

We propose a non-intrusive parametric model order reduction method combining the DMD and RBF. When heavy computations are needed for multi-query tasks in the parametric case, especially for predicting the nonlinear dynamics, the proposed parametric DMD is promising for prediction in both the parameter and the time domain.

The proposed method is tested on several examples and their results are compared with the reference solutions obtained by direct simulations of the original models. The results demonstrate that the proposed algorithm is effective. For the P2D battery model, the frequency of the current is the changing parameter, parametric DMD predicts the output potential at a new frequency with high accuracy. The second example of the ferrocyanide redox reaction is parametrized with rotation rates. The numerical results also indicate the high accuracy of the parametric DMD. The FitzHugh-Nagumo model further manifests the effective reduction and acceptable accuracy of the parametric DMD for the large nonlinear dynamic system.

Further improvements can be done in several directions. Firstly, all the numerical examples are based on a single parameter and the data are from simulation. Parametric DMD could also be applied to real experimental data with multiple parameters, which is of high interest in the design of experiments (DoE). Secondly, DMD and its related topic are being developed with a rapid speed, the proposed method could be further extended to new variants of DMD.

## Acknowledgments

This research is supported by the International Max Planck Research School for Advanced Methods in Process and Systems Engineering (IMPRS ProEng), Magdeburg, Germany. Also thanks for the model, code and help with the battery examples from several researchers (Karim Cherifi from TU Berlin, Antonio Sorrentino from EEC group in MPI Magdeburg).

## References

- [1] M. M. A. Asif, M. I. Ahmad, P. Benner, L. Feng, and T. Stykel. Implicit higher-order moment matching technique for model reduction of quadratic-bilinear systems. *Journal of the Franklin Institute*, 358(3):2015–2038, 2021. doi:10.1016/j.jfranklin.2020.11.012.
- [2] C. A. Beattie and S. Gugercin. Model reduction by rational interpolation. In *Model Reduction and Approximation: Theory and Algorithms*. SIAM, 2017. doi:10.1137/1.9781611974829.ch7.
- [3] P. Benner and T. Breiten. Two-sided projection methods for nonlinear model order reduction. *SIAM J. Sci. Comput.*, 37(2):B239–B260, 2015. doi:10.1137/14097255X.
- [4] P. Benner, P. Goyal, and S. Gugercin.  $\mathcal{H}_2$ -quasi-optimal model order reduction for quadratic-bilinear control systems. *SIAM J. Matrix Anal. Appl.*, 39(2):983–1032, 2018. doi:10.1137/16M1098280.

- [5] P. Benner, P. Goyal, B. Kramer, B. Peherstorfer, and K. Willcox. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Comp. Meth. Appl. Mech. Eng.*, 372:113433, 2020. doi:10.1016/j.cma.2020.113433.
- [6] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, editors. *Model Order Reduction, Volume 1: System- and Data-Driven Methods and Algorithms*. De Gruyter, 2021. URL: <https://www.degruyter.com/document/isbn/9783110498967/html>.
- [7] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, editors. *Model Order Reduction, Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, 2021. URL: <https://doi.org/10.1515/9783110671490>, doi:doi:10.1515/9783110671490.
- [8] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, editors. *Model Order Reduction, Volume 3: Applications*. De Gruyter, 2021. URL: <https://doi.org/10.1515/9783110499001>, doi:doi:10.1515/9783110499001.
- [9] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.*, 57(4):483–531, 2015. doi:10.1137/130932715.
- [10] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.
- [11] S. L. Brunton and J. N. Kutz. *7 Data-driven methods for reduced-order modeling*, pages 307–344. De Gruyter, 2020. URL: <https://doi.org/10.1515/9783110671490-007>, doi:doi:10.1515/9783110671490-007.
- [12] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010. doi:10.1137/090766498.
- [13] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of computational physics*, 446:110666, 2021.
- [14] M. Cheng, F. Fang, C. C. Pain, and I. M. Navon. An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. *Computer Methods in Applied Mechanics and Engineering*, 372:113375, 2020.
- [15] M. Doyle, T. F. Fuller, and J. Newman. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *Journal of the Electrochemical Society*, 140(6):1526, 1993.
- [16] S. Fresca, L. Dedè, and A. Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *J. Sci. Comput.*, 87:61, 2021. doi:10.1007/s10915-021-01462-7.
- [17] R. W. Freund. Model reduction methods based on krylov subspaces. *Acta Numerica*, 12:267–319, 2003.
- [18] P. Goyal and P. Benner. LQResNet: A deep neural network architecture for learning dynamic processes. e-print 2103.02249, arXiv, 2021. cs.LG. URL: <https://arxiv.org/abs/2103.02249>.
- [19] S. Grivet-Talocia and B. Gustavsen. *Passive Macromodeling: Theory and Applications*. John Wiley and Sons, 2016. doi:10.1002/9781119140931.
- [20] S. Gugercin and A. C. Antoulas. A survey of model reduction by balanced truncation and some new results. *Internat. J. Control*, 77(8):748–766, 2004. doi:10.1080/00207170410001713448.
- [21] M. Guo and J. S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99, 2019.

- [22] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Trans. Power Del.*, 14(3):1052–1061, 1999. doi:10.1109/61.772353.
- [23] Q. A. Huhn, M. E. Tano, J. C. Ragusa, and Y. Choi. Parametric dynamic mode decomposition for reduced order modeling. *arXiv preprint arXiv:2204.12006*, 2022.
- [24] M. Kast, M. Guo, and J. S. Hesthaven. A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems. *Computer Methods in Applied Mechanics and Engineering*, 364, 2020.
- [25] A. Keane and P. Nair. *Computational approaches for aerospace design: the pursuit of excellence*. John Wiley & Sons, 2005.
- [26] H. Kleikamp, M. Ohlberger, and S. Rave. Nonlinear model order reduction using diffeomorphic transformations of a space-time domain. *arXiv preprint arXiv:2203.05833*, 2022.
- [27] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Society of Industrial and Applied Mathematics, Philadelphia, USA, 2016. doi:10.1137/1.9781611974508.
- [28] N. Legrand, S. Raël, B. Knosp, M. Hinaje, P. Desprez, and F. Lopicque. Including double-layer capacitance in lithium-ion battery mathematical models. *Journal of Power Sources*, 251:370–378, 2014.
- [29] H. Lu and D. M. Tartakovsky. Model reduction via dynamic mode decomposition. *arXiv preprint arXiv:2204.09590*, 2022.
- [30] J. L. Lumley. The structure of inhomogeneous turbulence. *Atmospheric Turbulence and Radio Wave Propagation*, 01 1967.
- [31] J. L. Lumley. Coherent structures in turbulence. In *Transition and turbulence*, pages 215–242. Elsevier, 1981.
- [32] V. Mehrmann and T. Stykel. Balanced truncation model reduction for large-scale systems in descriptor form. In P. Benner, V. Mehrmann, and D. C. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lect. Notes Comput. Sci. Eng.*, pages 83–115. Springer-Verlag, Berlin/Heidelberg, Germany, 2005. doi:10.1007/3-540-27909-1\_3.
- [33] A. Mendible, S. L. Brunton, A. Y. Aravkin, W. Lowrie, and J. N. Kutz. Dimensionality reduction and reduced-order modeling for traveling wave physics. *Theoretical and Computational Fluid Dynamics*, 34(4):385–400, 2020.
- [34] Y. Nakatsukasa, O. Séte, and L. N. Trefethen. The AAA algorithm for rational approximation. *SIAM J. Sci. Comput.*, 40(3):A1494–A1522, 2018. doi:10.1137/16M1106122.
- [35] S. E. Otto and C. W. Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.
- [36] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.
- [37] S. M. Rahman, S. Pawar, O. San, A. Rasheed, and T. Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100(5):053306, 2019.
- [38] F. Regazzoni, L. Dede, and A. Quarteroni. Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics. *Computer Methods in Applied Mechanics and Engineering*, 370, 2020. Available online.

- [39] S. A. Renganathan, R. Maulik, and V. Rao. Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil. *Physics of Fluids*, 32(4):047110, 2020.
- [40] F. Röder, S. Sonntag, D. Schröder, and U. Krewer. Simulating the impact of particle size distribution on the performance of graphite electrodes in lithium-ion batteries. *Energy Technology*, 4(12):1588–1597, 2016.
- [41] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- [42] N. Sarna, J. Giesselmann, and P. Benner. Data-driven snapshot calibration via monotonic feature matching. e-prints 2009.08414, arXiv, 2020. cs.NA. URL: <https://arxiv.org/abs/2009.08414>.
- [43] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.*, 656:5–28, 2010. doi:10.1017/S0022112010001217.
- [44] L. Sirovich. Turbulence and the dynamics of coherent structures. parts I-III. *Quart. Appl. Math.*, 45(3):561–590, 1987. URL: <http://www.jstor.org/stable/43637457>.
- [45] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014. doi:10.3934/jcd.2014.1.391.
- [46] T. Vidaković-Koch, V. Panić, M. Andrić, M. Petkovska, and K. Sundmacher. Nonlinear frequency response analysis of the ferrocyanide oxidation kinetics. part i. a theoretical analysis. *Journal of Physical Chemistry (2011)*, 115, 08 2011. doi:10.1021/jp201297v.
- [47] G. Welper. Interpolation of functions with parameter dependent jumps by transformed snapshots. *SIAM Journal on Scientific Computing*, 39(4):A1225–A1250, 2017.
- [48] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.
- [49] Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel-based method for data-driven koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.
- [50] D. Xiao, F. Fang, 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.
- [51] Y. Q. Xiao, S. Grivet-Talocia, P. Manfredi, and R. Khazaka. A novel framework for parametric loewner matrix interpolation. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 9(12):2404–2417, 2019. doi:10.1109/TCPMT.2019.2948802.
- [52] J. Xu and K. Duraisamy. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372:113379, 2020.
- [53] A. Zanco and S. Grivet-Talocia. Toward fully automated high-dimensional parameterized macromodeling. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 11(9):1402–1416, 2021.