



HAL
open science

Model order reduction for parameterized electromagnetic problems using matrix decomposition and deep neural networks

Xiao-Feng He, Liang Li, Stéphane Lanteri, Kun Li

► **To cite this version:**

Xiao-Feng He, Liang Li, Stéphane Lanteri, Kun Li. Model order reduction for parameterized electromagnetic problems using matrix decomposition and deep neural networks. *Journal of Computational and Applied Mathematics*, 2023, 431, pp.115271. 10.1016/j.cam.2023.115271 . hal-03927379

HAL Id: hal-03927379

<https://hal.science/hal-03927379>

Submitted on 6 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Model order reduction for parameterized electromagnetic problems using matrix decomposition and deep neural networks

Xiao-Feng He^a, Liang Li^a, Stéphane Lanteri^b, Kun Li^c

a. School of Mathematical Sciences,

University of Electronic Science and Technology of China, Chengdu, Sichuan, P. R. China

b. Université Côte d'Azur, Inria, CNRS, LJAD, Inria,

2004 Route des Lucioles, BP 93, 06902, Sophia Antipolis Cedex, France

c. School of Economic Mathematics,

Southwestern University of Finance and Economics, Chengdu, P.R. China

Abstract

A non-intrusive model order reduction (MOR) method for solving parameterized electromagnetic scattering problems is proposed in this paper. A database collecting snapshots of high-fidelity solutions is built by solving the parameterized time-domain Maxwell equations for some values of the material parameters using a fullwave solver based on a high order discontinuous Galerkin time-domain (DGTD) method. To perform a prior dimensionality reduction, a set of reduced basis (RB) functions are extracted from the database via a two-step proper orthogonal decomposition (POD) method. Projection coefficients of the reduced basis functions are further compressed through a convolutional autoencoder (CAE) network. Singular value decomposition (SVD) is then used to extract the principal components of the reduced-order matrices generated by CAE, and a cubic spline interpolation-based (CSI) approach is employed for approximating the dominating time- and parameter-modes of the reduced-order matrices. The generation of the reduced basis and the training of the CAE and CSI are accomplished in the offline stage, thus the RB solution for given time/parameter values can be quickly recovered via outputs of the interpolation model and decoder network. In particular, the offline and online stages of the proposed RB method are completely decoupled, which ensures the validity of the method. The performance of the proposed CAE-CSI ROM is illustrated with numerical experiments for scattering of a plane wave by a 2-D dielectric disk and a multi-layer heterogeneous medium.

Key words: non-intrusive reduced-order modeling; parameterized electromagnetic scattering; proper orthogonal decomposition; convolutional autoencoder; cubic spline interpolation

1 Introduction

Electromagnetic wave interaction with heterogeneous media is generally modeled by the system of parameterized time-domain Maxwell equations [22]. The Finite Difference Time-Domain (FDTD) method [46, 39, 10] is the most popular method for simulating these time-domain electromagnetic wave propagation problems. The discontinuous Galerkin time-domain (DGTD) method is an alternative approach that has emerged during the last 20 years [20, 8]. Compared with the FDTD method, the DGTD method has several attractive advantages such as local approximation strategy, easy adaption to complex geometry and material composition [20]. With the DGTD method, the number of degree of freedom (DOFs) is determined by the underlying mesh and the polynomial order used for approximating the EM field components. This number is usually high to guarantee the accuracy of the solution of Maxwell's equations. Therefore, it makes sense to study the model order reduction method to reduce the computational burden of CPU time and memory, when repeatedly simulating Maxwell's equations with different parameters. Model order reduction (MOR) techniques [3] aim to replace a full-order model (FOM) by a reduced-order model (ROM), featuring a much lower dimension, thereby reducing computational cost under acceptable accuracy conditions. A reduced basis (RB) method based on an offline-online procedure is a widely used model order reduction technique [18, 14, 30, 41]. During the offline stage, a reduced space spanned by a set of time- and parameter-independent RB functions is constructed from the full-order solutions (snapshots), which are for instance generated by the DGTD method at different parameter values. The reduced-order solution for a new time and parameter value can be expressed as a linear

combination of the RB functions where the combination coefficients, also called the intrinsic coordinates, need to be calculated during the online stage. The high-fidelity solver is only used to generate the snapshots, thus guaranteeing a complete decoupling between the online evaluation and the offline training [44, 30].

More recently, non-intrusive ROM combining RB techniques and machine learning have been developed for solving large-scale complex physical problems. The advantage of these methods is that they are data-driven and do not require access to the governing equations of the original FOM. For instance, a non-intrusive RB method combining POD and feed forward neural networks (FNNs) has been proposed for parameterized unsteady flows [19, 41], where a neural network is trained to approximate the mapping between the time/parameter values and the projection coefficients. Moreover, different regression and interpolation methods such as Gaussian process regression (GPR) [13, 14, 48, 23], radial basis functions (RBF) [43, 27] and cubic spline interpolation (CSI) [30, 4, 35] have also been considered in place of FNNs to approximate the mapping from the time/parameter to the reduced coefficients. By combining an ANN-based regression model with a physics-informed neural network (PINN) [37, 31], a hybrid strategy is devised in [5] to train a network by minimizing the residual loss of the reduced-order equation at a set of points in the parameter space.

However, linear reduction methods such as POD hardly capture the complex dynamics of highly nonlinear systems, therefore nonlinear manifold learning methods such as Kernel PCA [49] and Hessian feature maps [45] have attracted much attention in the past several years. Assuming that data points lie in a low dimensional manifold embedded in an higher dimensional Euclidean space, manifold learning [32] aims to identify the intrinsic dimensionality, equal to the number of parameters that describe the system, and thus obtain low dimensional representations of the data points. Although kernel PCA and Hessian feature maps are effective in providing low dimensional representations for high dimensional data points, their main drawback is that they do not provide an analytical formula to decode the compressed data back to their high dimensional representation in the original space [34]. An autoencoder (AE) overcomes this disadvantage by learning how to compress (encode) a high dimensional data to a low dimensional code and then reconstruct (decode) the code to a representation as close to the original input as possible [34]. The encoder and decoder parts of an autoencoder are trained simultaneously but can be used separately, which provides an opportunity to build a mapping between the input time/parameter and encoded representation.

Directly applying large-scale simulation data (snapshots) to a fully-connected autoencoder is not only computationally prohibitive, but also ignores the opportunity to exploit the structure of features in high dimensional data [12, 9]. As an extension of ordinary autoencoders, convolutional autoencoders (CAEs) [25, 15] are characterized by shared parameters and local connectivity which help to reduce the memory as well as computational costs. For example, projection-based ROMs proposed in [26] project dynamic systems onto nonlinear manifolds by means of autoencoders, which still require the assembling and solution of a ROM as in traditional POD-Galerkin ROMs. In [12], a reduced trial manifold is generated via a deep convolutional recurrent AE, which is then used to train a long short-term memory (LSTM) network that models the reduced dynamics. A POD-DL-ROM approach proposed in [9] combines and improves the previous two methods as the nonlinear trial manifold is learnt by using the decoder function of a CAE while the dynamics on the reduced manifold is modeled through a DFNN and the encoder function of a CAE. In [34], the authors utilize a CAE in conjunction with a FNN to establish a mapping from the problem's parametric space to its solution space. The differences between [9] and [34] are that the former uses the POD method for pre-dimensionality reduction to reduce training time and trains CAE and FNN at the same time.

In this paper, a non-intrusive reduced-order modeling strategy is proposed to solve electromagnetic scattering problems governed by parameterized time-domain Maxwell equations. Firstly, the approach performs a prior dimensionality reduction by a two-step POD method [30]. Then, relying on its powerful dimensionality reduction properties, we use a CAE to encode and decode the intrinsic coordinates. Furthermore, a CSI-based model is devised to approximate a mapping from the problem's parameter space to its low dimensional encoded vector space. Using this approach, the encoded representation of the solution at a new time/parameter value is recovered by the CSI model, while the RB solution in the original high dimensional space is obtained by the decoder and projection matrix. The resulting CAE-CSI ROM provides a very fast and accurate evaluation of the entire system response.

The paper is organized as follows. In section 2, we briefly introduce the time-domain Maxwell equations and the DGTD scheme to generate the snapshots. Section 3 provides a description for the two-step POD, from which a RB basis is constructed. In section 4, we present how a CAE is employed to further reduce the dimension of intrinsic coordinates. In section 5, we show how to use CSI to build an approximate mapping between the problem's parameters and the low dimensional encoded vectors. The overall CAE-CSI ROM for parameterized electromagnetic scattering problems is also presented in this section. In Section 6, two numerical

experiments for testing the method are provided. And conclusion remarks are drawn in section 7.

2 Full-order model and DGTD method

Unsteady electromagnetic scattering problems are governed by the following normalized form of the time-domain Maxwell equations

$$\begin{cases} \nu_r \frac{\partial \mathbf{H}(\mathbf{x}, t)}{\partial t} + \text{curl}(\mathbf{E}(\mathbf{x}, t)) = 0, & \forall (\mathbf{x}, t) \in \Omega \times \mathcal{T}, \\ \varepsilon_r \frac{\partial \mathbf{E}(\mathbf{x}, t)}{\partial t} - \text{curl}(\mathbf{H}(\mathbf{x}, t)) = 0, & \forall (\mathbf{x}, t) \in \Omega \times \mathcal{T}, \\ \mathcal{L}(\mathbf{E}(\mathbf{x}, t), \mathbf{H}(\mathbf{x}, t)) = \mathcal{L}(\mathbf{E}^{\text{inc}}(\mathbf{x}, t), \mathbf{H}^{\text{inc}}(\mathbf{x}, t)), & \forall (\mathbf{x}, t) \in \partial\Omega \times \mathcal{T}, \\ \mathbf{E}(\mathbf{x}, 0) = \mathbf{E}_0(\mathbf{x}), \mathbf{H}(\mathbf{x}, 0) = \mathbf{H}_0(\mathbf{x}), & \forall \mathbf{x} \in \Omega, \end{cases} \quad (1)$$

where Ω is the spatial domain and $\mathcal{T} = [0, T_f]$ is the time interval, $\mathbf{E} = (E_x, E_y, E_z)^T$ and $\mathbf{H} = (H_x, H_y, H_z)^T$ are the electric field and the magnetic field respectively, ε_r and ν_r denote the relative electric permittivity and magnetic permeability parameters respectively. In this work we consider the first-order Silver-Müller absorbing boundary condition (ABC) [33]

$$\mathcal{L}(\mathbf{E}(\mathbf{x}, t), \mathbf{H}(\mathbf{x}, t)) = \mathbf{n} \times \mathbf{E}(\mathbf{x}, t) + Z \mathbf{n} \times (\mathbf{n} \times \mathbf{H}(\mathbf{x}, t)), \text{ on } \partial\Omega, \quad (2)$$

where $\partial\Omega$ is the boundary of Ω , \mathbf{n} denotes the outer unit normal vector along $\partial\Omega$, \mathbf{E}^{inc} and \mathbf{H}^{inc} are the incident fields, and $Z = \sqrt{\nu_r/\varepsilon_r}$. Our goal is to solve equation (1) with varying parameter $\varepsilon_r \in \mathcal{P}$, where \mathcal{P} denotes the parameter domain.

System (1) is discretized in space by means of discontinuous Galerkin method, and in time by means of a second-order leapfrog scheme. The time interval $\mathcal{T} = [0, T_f]$ is divided into m equal subintervals as $0 = t_0 < t_1 < \dots < t_m = T_f$ with $t_n = n\Delta t$ ($n = 0, 1, \dots, m$), and Δt denotes the time step size. The fully discrete scheme of DGTD based on centered fluxes and second-order Leap-Frog time stepping [8] is given by

$$\begin{cases} \mathbb{M}^{\varepsilon_r} \frac{\underline{\mathbf{E}}_h(t_{n+1}) - \underline{\mathbf{E}}_h(t_n)}{\Delta t} = (\mathbb{K} - \mathbb{S}^i) \underline{\mathbf{H}}_h(t_{n+\frac{1}{2}}) - \mathbb{S}^h \underline{\mathbf{H}}_h(t_{n+\frac{1}{2}}) - \mathbf{B}^h(n\Delta t), \\ \mathbb{M}^{\nu_r} \frac{\underline{\mathbf{H}}_h(t_{n+\frac{3}{2}}) - \underline{\mathbf{H}}_h(t_{n+\frac{1}{2}})}{\Delta t} = (-\mathbb{K} + \mathbb{S}^i) \underline{\mathbf{E}}_h(t_{n+1}) + \mathbb{S}^e \widehat{\underline{\mathbf{E}}}_h(t_{n+1}) + \mathbf{B}^e((n + \frac{1}{2})\Delta t). \end{cases} \quad (3)$$

Here $\mathbb{M}^{\varepsilon_r}$ and \mathbb{M}^{ν_r} are the mass matrices, \mathbb{K} is the stiffness matrix, \mathbb{S}^i is the surface matrix for the interior faces, and \mathbb{S}^h and \mathbb{S}^e are the boundary face matrices. For more detailed definition of these matrices we refer to [40, 29].

3 Two-step POD method

3.1 Snapshot-based data collection

For the parameterized time-dependent problem, let $\mathcal{P}_h^{tr} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_{\mathcal{N}_p}\}$ be a parameter sampling over the parameter domain \mathcal{P} . A collection of high-fidelity solutions can be obtained by solving system (1) with the DGTD solver for different parameter values $\boldsymbol{\mu} \in \mathcal{P}_h^{tr}$ in a time sampling $\mathcal{T}_h = \{t_1, t_2, \dots, t_m\}$. Then we uniformly select \mathcal{N}_t transient solutions in $\mathcal{T}_h^{tr} = \{t_{n_1}, t_{n_2}, \dots, t_{n_{\mathcal{N}_t}}\} \subseteq \mathcal{T}_h$ to form the time trajectory matrix $\mathbf{S}_{\mathbf{u}}^j \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}_t}$, which takes the form

$$\mathbf{S}_{\mathbf{u}}^j = \begin{pmatrix} \mathbf{u}_{h,1}(t_{n_1}, \boldsymbol{\mu}_j) & \mathbf{u}_{h,1}(t_{n_2}, \boldsymbol{\mu}_j) & \cdots & \mathbf{u}_{h,1}(t_{n_{\mathcal{N}_t}}, \boldsymbol{\mu}_j) \\ \mathbf{u}_{h,2}(t_{n_1}, \boldsymbol{\mu}_j) & \mathbf{u}_{h,2}(t_{n_2}, \boldsymbol{\mu}_j) & \cdots & \mathbf{u}_{h,2}(t_{n_{\mathcal{N}_t}}, \boldsymbol{\mu}_j) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_{h,\mathcal{N}_h}(t_{n_1}, \boldsymbol{\mu}_j) & \mathbf{u}_{h,\mathcal{N}_h}(t_{n_2}, \boldsymbol{\mu}_j) & \cdots & \mathbf{u}_{h,\mathcal{N}_h}(t_{n_{\mathcal{N}_t}}, \boldsymbol{\mu}_j) \end{pmatrix} \text{ with } j = 1, 2, \dots, \mathcal{N}_p, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}, \quad (4)$$

and the snapshot matrix for all parameters in \mathcal{P}_h^{tr} is

$$\mathbf{S}_{\mathbf{u}} = \left[\mathbf{S}_{\mathbf{u}}^1 \mid \mathbf{S}_{\mathbf{u}}^2 \mid \cdots \mid \mathbf{S}_{\mathbf{u}}^{\mathcal{N}_p} \right] \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}_s}, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}, \quad (5)$$

where $\mathcal{N}_s = \mathcal{N}_t \cdot \mathcal{N}_p$, $\mathbf{u}_h(t, \boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}_h}$ is the high-fidelity solution, and \mathcal{N}_h is the number of DOFs, which is determined by the underlying mesh and polynomial order of the discretization scheme.

Directly reducing the dimensionality of the snapshots through a neural network will result in an overwhelming computational burden when the FOM dimension \mathcal{N}_h becomes moderately large. The CAE-CSI technique proposed here can be considered as a non-intrusive ROM technique in which a three-step dimensionality reduction is performed. First, a two-step POD strategy is applied on a set of FOM snapshots. Then a convolutional autoencoder is utilized to reduce the dimensionality of the projection coefficients (also called intrinsic coordinates) generated by POD. Lastly, a CSI-based model is built to approximate the mapping between input parameters $(t, \boldsymbol{\mu})$ and the reduced-order matrices.

3.2 Two-step POD for dimensionality reduction

In this subsection, we perform a low-rank approximation to the snapshot matrix \mathbf{S}_u and construct a low dimensional space $\mathcal{V}_{u,rb}$ with dimension $\mathcal{N} \ll \min\{\mathcal{N}_h, \mathcal{N}_s\}$. Spanned by a group of time- and parameter-independent RB functions, the reduced space is expressed as

$$\mathcal{V}_{u,rb} = \text{span} \{ \mathbf{v}_{u,1}, \mathbf{v}_{u,2}, \dots, \mathbf{v}_{u,\mathcal{N}} \}, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}. \quad (6)$$

The POD method is a popular technique to compress data and extract an optimal set of RB functions in the least-squares sense. The POD basis of size \mathcal{N} is the solution to the minimization problem

$$\begin{aligned} \min_{\mathbf{V}_u \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}}} & \|\mathbf{S}_u - \mathbf{V}_u \mathbf{V}_u^T \mathbf{S}_u\|_F, \\ \text{s.t.} & \quad \mathbf{V}_u^T \mathbf{V}_u = \mathbf{I}, \end{aligned} \quad (7)$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbf{I} \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ is an identity matrix.

According to the Eckart-Young theorem [38, 7], the solution of (7) is given by the first \mathcal{N} left singular vectors of the matrix \mathbf{S}_u , which can be obtained through the singular value decomposition (SVD)

$$\mathbf{W}_u^T \mathbf{S}_u \mathbf{Z}_u = \begin{pmatrix} \Sigma_{r_u \times r_u}^u & \mathbf{O}_{r_u \times (\mathcal{N}_s - r_u)} \\ \mathbf{O}_{(\mathcal{N}_h - r_u) \times r_u} & \mathbf{O}_{(\mathcal{N}_h - r_u) \times (\mathcal{N}_s - r_u)} \end{pmatrix} = \mathbf{D}_u, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}, \quad (8)$$

where $\mathbf{W}_u = [\mathbf{w}_{u,1}, \mathbf{w}_{u,2}, \dots, \mathbf{w}_{u,\mathcal{N}_h}]$ and $\mathbf{Z}_u = [\mathbf{z}_{u,1}, \mathbf{z}_{u,2}, \dots, \mathbf{z}_{u,\mathcal{N}_s}]$ are $\mathcal{N}_h \times \mathcal{N}_h$ and $\mathcal{N}_s \times \mathcal{N}_s$ orthogonal matrices respectively, and $\Sigma_{r_u \times r_u}^u = \text{diag}(\sigma_{u,1}, \sigma_{u,2}, \dots, \sigma_{u,r_u})$. The singular values $\sigma_{u,1} \geq \sigma_{u,2} \geq \dots \geq \sigma_{u,r_u} \geq 0$ of \mathbf{S}_u are sorted in descending order, and r_u is the rank of \mathbf{S}_u . The POD basis with \mathcal{N} ($\mathcal{N} \ll r_u$) vectors is the set $\{\mathbf{v}_{u,i}\}_{i=1}^{\mathcal{N}}$ with $\mathbf{v}_{u,i} = \mathbf{w}_{u,i}$, which can minimize the projection error of the snapshots among all \mathcal{N} -dimensional orthogonal basis in $\mathbb{R}^{\mathcal{N}_h}$. The error bound can be evaluated using the singular values

$$\sum_{i=1}^{\mathcal{N}_s} \|\mathbf{S}_u(:, i) - \mathbf{V}_u \mathbf{V}_u^T \mathbf{S}_u(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}}^2 = \sum_{j=\mathcal{N}+1}^{r_u} \sigma_{u,j}^2, \quad (9)$$

where $\mathbf{V}_u = [\mathbf{v}_{u,1}, \dots, \mathbf{v}_{u,\mathcal{N}}]$, $\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$. According to (9), it is clear that the error in the POD basis is equal to the sum of the squares of the neglected singular values, i.e., the error can be controlled by \mathcal{N} . In this work, POD is utilized to perform a moderate dimensionality reduction of the snapshots data, thus yielding a linear subspace of dimension $\mathcal{N} \ll \mathcal{N}_h$.

However, since $\mathcal{N}_s = \mathcal{N}_t \cdot \mathcal{N}_p$ is large, performing SVD on such a large-scale snapshot matrix directly is extremely expensive. We adopt a two-step POD method to effectively save the computational cost. The detailed process is as follow and also shown in Algorithm 1:

1. For each time trajectory matrix \mathbf{S}_u^j , $j = 1, 2, \dots, \mathcal{N}_p$, the POD basis $\{\gamma_{u,i}^j\}_{i=1}^k$ are obtained through SVD, then assemble them to a matrix $\mathbf{T}_u^j = [\gamma_{u,1}^j, \gamma_{u,2}^j, \dots, \gamma_{u,k}^j]$;
2. Assemble a composite matrix $\mathbf{T}_u = [\mathbf{T}_u^1 \mid \mathbf{T}_u^2 \mid \dots \mid \mathbf{T}_u^{\mathcal{N}_p}]$ with the matrices generated in the first step, then perform POD on \mathbf{T}_u with truncation \mathcal{N} . The POD basis $\{\mathbf{v}_{u,i}\}_{i=1}^{\mathcal{N}}$ is obtained, i.e., $\mathbf{V}_u = [\mathbf{v}_{u,1}, \dots, \mathbf{v}_{u,\mathcal{N}}]$.

According to the projection theory [28], one can obtain the reduced-order solution and the projection coefficients or intrinsic coordinates as

$$\begin{cases} \mathbf{u}_h^r(t, \boldsymbol{\mu}) = \mathbf{V}_u (\mathbf{V}_u^T \mathbf{V}_u)^{-1} \mathbf{V}_u^T \mathbf{u}_h(t, \boldsymbol{\mu}) = \mathbf{V}_u \mathbf{V}_u^T \mathbf{u}_h(t, \boldsymbol{\mu}), \\ \alpha_u(t, \boldsymbol{\mu}) \equiv \mathbf{u}_N(t, \boldsymbol{\mu}) = (\mathbf{V}_u^T \mathbf{V}_u)^{-1} \mathbf{V}_u^T \mathbf{u}_h(t, \boldsymbol{\mu}) = \mathbf{V}_u^T \mathbf{u}_h(t, \boldsymbol{\mu}), \end{cases} \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}. \quad (10)$$

Thus, the reduced-order solution $\mathbf{u}_h^r(t, \boldsymbol{\mu})$ serves as an approximation to the high-fidelity solution $\mathbf{u}_h(t, \boldsymbol{\mu})$ and can be represented as

$$\mathbf{u}_h^r(t, \boldsymbol{\mu}) = \mathbf{V}_u \alpha_u(t, \boldsymbol{\mu}) = \sum_{i=1}^{\mathcal{N}} \alpha_{u,i}(t, \boldsymbol{\mu}) \mathbf{v}_{u,i}, \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}, \quad (11)$$

where $\alpha_u(t, \boldsymbol{\mu}) = [\alpha_{u,1}(t, \boldsymbol{\mu}), \alpha_{u,2}(t, \boldsymbol{\mu}), \dots, \alpha_{u,\mathcal{N}}(t, \boldsymbol{\mu})]^T \in \mathbb{R}^{\mathcal{N}}$ collects the combination coefficients.

In our previous work [48, 30] we constructed the mapping between the time/parameter values and the projection coefficients by decoupling time- and parameters-modes through SVD. However, these ROMs need to create too many regression or interpolation models. The approach we proposed here firstly reduces the length of projection coefficients $\alpha_u(t, \boldsymbol{\mu})$ or intrinsic coordinates $\mathbf{u}_N(t, \boldsymbol{\mu})$ from \mathcal{N} to a fairly small n through a convolutional autoencoder, then constructs the mapping between the time/parameter values and the low-dimensional coded representation using cubic spline interpolation, thus reducing the number of interpolation models as well as test time online.

Algorithm 1: Two-step POD method

Input : Time trajectory matrices \mathbf{S}_u^j ($j = 1, 2, \dots, \mathcal{N}_p$, $\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$), truncation parameter k and size parameter \mathcal{N}

Output: POD basis matrix \mathbf{V}_u ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$)

- 1 Compute the compressed matrices $\mathbf{T}_u^j = \text{POD}(\mathbf{S}_u^j, k)$ for $j = 1, 2, \dots, \mathcal{N}_p$, $\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$;
 - 2 Assemble the matrix $\mathbf{T}_u = [\mathbf{T}_u^1 \mid \mathbf{T}_u^2 \mid \dots \mid \mathbf{T}_u^{\mathcal{N}_p}]$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) for all parameter values;
 - 3 Calculate the POD basis matrix $\mathbf{V}_u = \text{POD}(\mathbf{T}_u, \mathcal{N})$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$)
 - 4 **Function** $\mathbf{V} = \text{POD}(\mathbf{A}, k)$
 - 5 | Perform $\text{eig}(\mathbf{A}^T \mathbf{A})$ in MATLAB, and get the eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$ and the corresponding orthogonal eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$ with r being the rank of \mathbf{A} ;
 - 6 | Compute the POD basis functions $\mathbf{v}_i = \frac{1}{\sqrt{\lambda_i}} \mathbf{A} \mathbf{u}_i$, $i = 1, 2, \dots, k$, $k \ll r$;
 - 7 | Obtain the POD basis matrix $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k]$.
 - 8 **end**
-

Remark 1. According to (9), the error bounds in the first and second steps of the two-step POD algorithm are written as

$$\begin{cases} \sum_{i=1}^{\mathcal{N}_t} \|\mathbf{S}_u^j(:, i) - \mathbf{T}_u^j (\mathbf{T}_u^j)^T \mathbf{S}_u^j(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}}^2 = \sum_{i=k+1}^{r_u^j} (\sigma_{u,i}^j)^2, 1 \leq j \leq \mathcal{N}_p, \\ \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^k \|\mathbf{T}_u^j(:, i) - \mathbf{V}_u \mathbf{V}_u^T \mathbf{T}_u^j(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}}^2 = \sum_{i=\mathcal{N}+1}^{r_u} (\sigma_{u,i})^2, \end{cases} \quad (12)$$

where r_u and r_u^j ($j = 1, 2, \dots, \mathcal{N}_p$, $\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) are the rank of \mathbf{S}_u and \mathbf{S}_u^j , and $\{\sigma_{u,i}\}_{i=1}^{r_u}$ as well as $\{\sigma_{u,i}^j\}_{i=1}^{r_u^j}$ are the corresponding singular values. The two-step POD projection error can be bounded as

$$\begin{aligned} \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_t} \|\mathbf{S}_u^j(:, i) - \mathbf{V}_u \mathbf{V}_u^T \mathbf{S}_u^j(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}} &\leq \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_t} \|\mathbf{S}_u^j(:, i) - \mathbf{T}_u^j (\mathbf{T}_u^j)^T \mathbf{S}_u^j(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}} \\ &+ \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_t} \|\mathbf{T}_u^j (\mathbf{T}_u^j)^T \mathbf{S}_u^j(:, i) - \mathbf{V}_u \mathbf{V}_u^T \mathbf{T}_u^j (\mathbf{T}_u^j)^T \mathbf{S}_u^j(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}} \\ &+ \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_t} \|\mathbf{V}_u \mathbf{V}_u^T \mathbf{T}_u^j (\mathbf{T}_u^j)^T \mathbf{S}_u^j(:, i) - \mathbf{V}_u \mathbf{V}_u^T \mathbf{S}_u^j(:, i)\|_{\mathbb{R}^{\mathcal{N}_h}} \\ &\leq \mathcal{L}_1 + \mathcal{L}_2, \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}, \end{aligned}$$

where

$$\begin{cases} \mathcal{L}_1 = (1 + \|\mathbf{V}_u \mathbf{V}_u^T\|_F) \sum_{j=1}^{\mathcal{N}_p} (\mathcal{N}_t \sum_{i=k+1}^{r_u^j} (\sigma_{u,i}^j)^2)^{\frac{1}{2}}, \\ \mathcal{L}_2 = \max_{1 \leq j \leq \mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_t} \|\mathbf{S}_u^j(\cdot, i)\|_{\mathbb{R}^{\mathcal{N}_h}} \max_{\substack{1 \leq i \leq k \\ 1 \leq j \leq \mathcal{N}_p}} \|\mathbf{T}_u^j(\cdot, i)\|_{\mathbb{R}^{\mathcal{N}_h}} (k \mathcal{N}_p \sum_{i=N+1}^{r_u} (\sigma_{u,i})^2)^{\frac{1}{2}}. \end{cases}$$

Therefore, the accuracy of the two-step POD can be controlled by the truncation parameter k and size parameter \mathcal{N} .

4 Convolutional autoencoders for model reduction

In this section, we present an approach to construct a nonlinear manifold, which compresses the dimensionality of projection coefficients $\alpha_u(t, \boldsymbol{\mu}) \equiv \mathbf{u}_{\mathcal{N}}(t, \boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) from \mathcal{N} to a fairly small n . In data-driven sciences, the manifold hypothesis presumes that real-world high dimensional data lie near a low dimensional manifold \mathcal{S} embedded in \mathbb{R}^m , where m is large [2]. As a result POD has been widely used for dimensionality reduction of physical systems. However the main drawback is that POD can only construct an optimal linear manifold while data sampled from complex real-world systems tends to be strongly nonlinear.

A nonlinear generalization of POD is the autoencoder, which takes the form of a fully-connected neural network shown in Fig. 1. Specifically, an autoencoder includes two parts: the encoder maps a high dimensional input to a low-dimensional latent vector, known as encoding; then the decoder learns how to reconstruct the original input from the latent vector with a minimum error. A basic autoencoder consists of a single or multiple

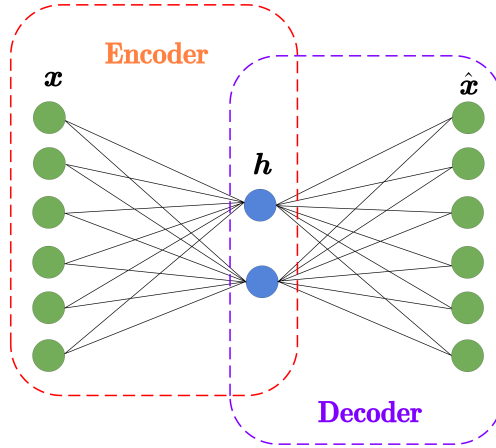


Figure 1: Network architecture of a fully-connected autoencoder.

layer encoder network

$$\mathbf{h} = f_E(\mathbf{x}; \boldsymbol{\theta}_E), \quad (13)$$

where $\boldsymbol{\theta}_E$ denotes the weights and the biases of the encoder network, $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$ is the input state, $\mathbf{h} \in \mathbb{R}^n$ is the feature or low dimensional representation vector with $n \ll \mathcal{N}$. A decoder network is then used to reconstruct \mathbf{x} by

$$\hat{\mathbf{x}} = f_D(\mathbf{h}; \boldsymbol{\theta}_D). \quad (14)$$

The objective is how to train this autoencoder and find the parameters that minimize the mean squared error over all training examples

$$\boldsymbol{\theta}_E^*, \boldsymbol{\theta}_D^* = \arg \min_{\boldsymbol{\theta}_E, \boldsymbol{\theta}_D} \|\mathbf{x} - \hat{\mathbf{x}}\|^2. \quad (15)$$

Note that directly applying large-scale snapshots to fully connected autoencoders is not only computationally expensive, but also ignores opportunities to exploit feature structures in high dimensional data. So we adopt an convolutional autoencoder which is characterized by shared parameters and local connectivity, thus reduce the memory as well as computational cost.

4.1 Basics of convolutional and deconvolutional layers

Some basic operations in convolutional and transposed convolutional layers are introduced in this subsection. The reader is referred to [6, 47] for more details of the convolution operations.

For simplicity, we consider a 3-D tensor $\mathcal{X} \in \mathbb{R}^{c \times h \times w}$ as an input of the convolutional neural network, with c , h and w being the number of channels, height and width respectively. The convolution operations aim to extract the most important features from the input \mathcal{X} and then use them to construct a feature map \mathcal{G} . Specifically, the output at the $(l-1)$ th layer is $\mathcal{X}^{l-1} \in \mathbb{R}^{c_q^{l-1} \times h^{l-1} \times w^{l-1}}$, and the feature map at l th layer is denoted as a tensor $\mathcal{G}^l \in \mathbb{R}^{c_p^l \times n_1^l \times n_2^l}$ with $c_p^l = c_q^{l-1}$ and element $\mathcal{G}_{i,j,k}^l$ representing a unit within channel i at row j and column k . The filter banks collecting a set of kernels at l th layer can be considered as a 4-dimensional tensor $\mathcal{W}^l \in \mathbb{R}^{c_p^l \times c_q^l \times k_h^l \times k_w^l}$ with element $\mathcal{W}_{i,j,m,n}^l$ connecting between a unit in channel i of the output and a unit in channel j of the input, with an offset of m rows and n columns between the output unit and the input unit. c_p denotes the number of kernels in the filter bank, and the kernel size is denoted by k_h and k_w . Convolution between a feature map \mathcal{G}^{l-1} and a filter bank \mathcal{W}^l can be expressed as

$$\mathcal{G}_{i,j,k}^l = \sigma_l \left(\sum_{r,m,n} \mathcal{G}_{r,(j-1) \times s + m, (k-1) \times s + n}^{l-1} \mathcal{W}_{i,r,m,n}^l + \mathcal{B}_{i,j,k}^l \right), \quad (16)$$

where σ_l is a nonlinear activation function and \mathcal{B}^l is a bias. Here, s denotes the stride, which determines the downsampling rate of each convolution. The dimension of the next feature map can be reduced by a factor s in each direction when $s > 1$. The kernel size $[k_h, k_w]$, the number of filters c_p and the stride s are the hyperparameters while the filter banks \mathcal{W} and the biases \mathcal{B} are learnable parameters. A deconvolutional layer performs the reverse operations of convolution, called transposed convolution, and it is used to construct decoding layers [26]. In summary, the architecture of CAE shown in Fig. 2 is composed of convolutional, deconvolutional and dense layers, which is used for dimensionality reduction and reconstruction.

4.2 Data preparation

In this subsection, we present a procedure to generate a data set for training. Provided the high-fidelity snapshots and the orthogonal basis $\mathbf{V}_{\mathbf{u}} \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) generated by the two-step POD method, we compute the projection coefficients or intrinsic coordinates $\alpha_{\mathbf{u}}(t, \boldsymbol{\mu}) = \mathbf{u}_{\mathcal{N}}(t, \boldsymbol{\mu}) = \mathbf{V}_{\mathbf{u}}^T \mathbf{u}_h(t, \boldsymbol{\mu})$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) to perform a moderate dimensionality reduction, and the projection coefficient matrix $\mathbf{C}_{\mathbf{u}}$ takes the form

$$\mathbf{C}_{\mathbf{u}} = \left[\mathbf{C}_{\mathbf{u}}^1 \mid \mathbf{C}_{\mathbf{u}}^2 \mid \dots \mid \mathbf{C}_{\mathbf{u}}^{\mathcal{N}_p} \right] \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}_s}, \quad \mathbf{u} \in \{\mathbf{H}, \mathbf{E}\}, \quad (17)$$

where $\mathbf{C}_{\mathbf{u}}^j = \mathbf{V}_{\mathbf{u}}^T \mathbf{S}_{\mathbf{u}}^j \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}_t}$, $j = 1, 2, \dots, \mathcal{N}_p$. Next, we randomly shuffle $\mathbf{C}_{\mathbf{u}}$ by column and split it into training set and validation set according to a training-validation splitting fraction λ , i.e., $\mathbf{C}_{\mathbf{u}} = [\mathbf{C}_{\mathbf{u}}^{\text{tr}}, \mathbf{C}_{\mathbf{u}}^{\text{val}}]$, $\mathbf{u} \in \{\mathbf{H}, \mathbf{E}\}$, where $\mathbf{C}_{\mathbf{u}}^{\text{tr}} \in \mathbb{R}^{\mathcal{N} \times \lambda \mathcal{N}_s}$. As the training speed of neural network is affected by the range of input/output [21], feature scaling is required before feeding the training data into the network. The input data $\mathbf{C}_{\mathbf{u}}^{\text{tr}}$ and $\mathbf{C}_{\mathbf{u}}^{\text{val}}$ are normalized to $[0, 1]$ through the following affine transformation

$$\mathbf{C}_{\mathbf{u}}^{\text{tr}}(i, j) = \frac{\mathbf{C}_{\mathbf{u}}^{\text{tr}}(i, j) - \min_{1 \leq i \leq \mathcal{N}} \min_{1 \leq j \leq \lambda \mathcal{N}_s} \mathbf{C}_{\mathbf{u}}^{\text{tr}}(i, j)}{\max_{1 \leq i \leq \mathcal{N}} \max_{1 \leq j \leq \lambda \mathcal{N}_s} \mathbf{C}_{\mathbf{u}}^{\text{tr}}(i, j) - \min_{1 \leq i \leq \mathcal{N}} \min_{1 \leq j \leq \lambda \mathcal{N}_s} \mathbf{C}_{\mathbf{u}}^{\text{tr}}(i, j)}. \quad (18)$$

We then reshape each column of $\mathbf{C}_{\mathbf{u}}$ into a square matrix of dimension $(\sqrt{\mathcal{N}}, \sqrt{\mathcal{N}})$, where $\mathcal{N} = m^2$ with $m \in \mathbb{N}$. Then stack the components of \mathbf{H} and \mathbf{E} together to form a tensor with d channels, where d denotes the number of vectorial components of the solution of system (1). Thus, the input of the autoencoder network is a tensor of dimension $(\sqrt{\mathcal{N}}, \sqrt{\mathcal{N}}, d)$, which allows to reduce the number of parameters, thus save the time of training and testing the network.

4.3 Dimensionality reduction via convolutional autoencoders

The architecture of the CAE, employed at training stage, is shown in Fig. 2. The loss function to measure the discrepancy between the input $\mathbf{u}_{\mathcal{N}}(t, \boldsymbol{\mu}) = \mathbf{V}_{\mathbf{u}}^T \mathbf{u}_h(t, \boldsymbol{\mu})$ and its reconstruction $\tilde{\mathbf{u}}_{\mathcal{N}}(t, \boldsymbol{\mu}, \boldsymbol{\theta}_E, \boldsymbol{\theta}_D)$ is defined as follow

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{\mathcal{N}_s} \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_t} \left\| \mathbf{V}_{\mathbf{u}}^T \mathbf{u}_h(t_i, \boldsymbol{\mu}_j) - \tilde{\mathbf{u}}_{\mathcal{N}}(t_i, \boldsymbol{\mu}_j, \boldsymbol{\theta}_E, \boldsymbol{\theta}_D) \right\|^2, \quad (19)$$

and $\boldsymbol{\theta} = (\boldsymbol{\theta}_E, \boldsymbol{\theta}_D)$. The CAE is implemented in PyTorch [36] and the Adam optimizer [24] is used in the training stage. We set the initial learning rate to $\eta_0 = 10^{-4}$, the mini-batch size to $\mathcal{N}_{\text{mb}} = 50$, and maximum number of epochs to $\mathcal{N}_{\text{epo}} = 5000$. A learning rate decay with $\eta = \eta_0 / (1 + \alpha * \text{epoch})$ is used to accelerate the training with hyperparameter $\alpha = 0.05$ [41]. The dataset is divided into training and validation set with a proportion 8 : 2, i.e., $\lambda = 0.8$. To avoid overfitting, we stop the training if the loss on validation set does not decrease over 500 epochs. The ELU function defined as follow is utilized as the nonlinear activation function

$$\sigma(z) = \begin{cases} z & z \geq 0 \\ \exp(z) - 1 & z < 0 \end{cases}. \quad (20)$$

No activation function is applied at the last convolutional layer of the decoder neural network. The weights and biases of the network are initialized through the Xavier initialization [11]. The training process of the CAE is shown in Algorithm 2.

Algorithm 2: Training process of the CAE

Input: Snapshot matrix $\mathbf{S}_{\mathbf{u}}$ and POD basis matrix $\mathbf{V}_{\mathbf{u}}$ with $\mathbf{u} \in \{\mathbf{H}, \mathbf{E}\}$, training-validation splitting fraction λ , initial learning rate η_0 , learning rate decay parameter α , mini-batch size \mathcal{N}_{mb} and maximum number of epochs \mathcal{N}_{epo} .

Output: Optimal model weights $\boldsymbol{\theta}^* = (\boldsymbol{\theta}_E^*, \boldsymbol{\theta}_D^*)$.

- 1 Compute intrinsic coordinate matrix $\mathbf{C}_{\mathbf{u}}^j = \mathbf{V}_{\mathbf{u}}^T \mathbf{S}_{\mathbf{u}}^j$ for $j = 1, 2, \dots, \mathcal{N}_p$, $\mathbf{u} \in \{\mathbf{H}, \mathbf{E}\}$;
 - 2 Shuffle $\mathbf{C}_{\mathbf{u}} = [\mathbf{C}_{\mathbf{u}}^1, \mathbf{C}_{\mathbf{u}}^2, \dots, \mathbf{C}_{\mathbf{u}}^{\mathcal{N}_p}]$ randomly by column;
 - 3 Split data into training and validation set $\mathbf{C}_{\mathbf{u}} = [\mathbf{C}_{\mathbf{u}}^{\text{tr}}, \mathbf{C}_{\mathbf{u}}^{\text{val}}]$ with $\mathbf{C}_{\mathbf{u}}^{\text{tr}} \in \mathbb{R}^{\mathcal{N}_h \times \lambda \mathcal{N}_s}$;
 - 4 Obtain the optimal model parameters $[\mathbf{W}^*, \mathbf{b}^*] = \text{CAE_TRAINING}(\mathbf{C}_{\mathbf{u}}, \mathcal{N}_{\text{epo}}, \mathcal{N}_{\text{b}}, \eta_0)$;
 - 5 **Function** $[\mathbf{W}_{\text{tr}}, \mathbf{b}_{\text{tr}}] = \text{CAE_TRAINING}(\mathcal{D}, \mathcal{N}_{\text{epo}}, \mathcal{N}_{\text{b}}, \eta_0)$
 - 6 $\mathbf{W}, \mathbf{b} \leftarrow \text{Initialize}(\mathbf{W}, \mathbf{b})$ ▷ initialize weights and biases
 - 7 $\mathcal{N}_{\text{b}} \leftarrow \mathcal{N}_s / \mathcal{N}_{\text{mb}}$ ▷ number of batches
 - 8 **for** epoch $\leftarrow 1$ **to** \mathcal{N}_{epo} **do**
 - 9 $\eta \leftarrow \text{Learning_rate_decay}(\eta_0, \alpha, \text{epoch})$ ▷ learning rate decay
 - 10 **for** $s \leftarrow 1$ **to** \mathcal{N}_{b} **do**
 - 11 $\mathcal{D}_i \leftarrow \mathcal{D}[(s-1) * \mathcal{N}_{\text{mb}} + 1 : s * \mathcal{N}_{\text{mb}}]$ ▷ the s-th mini-batch
 - 12 $\mathcal{D}_i \leftarrow \text{Reshape}(\mathcal{D}_i, (\mathcal{N}_{\text{mb}}, \sqrt{\mathcal{N}}, \sqrt{\mathcal{N}}, d))$ ▷ reshape to d channels
 - 13 $\mathcal{J} \leftarrow \frac{1}{\mathcal{N}_s} \sum_{j=1}^{\mathcal{N}_p} \sum_{i=1}^{\mathcal{N}_i} \|\mathbf{V}_{\mathbf{u}}^T \mathbf{u}_h(t_i, \boldsymbol{\mu}_j) - \tilde{\mathbf{u}}_{\mathcal{N}}(t_i, \boldsymbol{\mu}_j)\|^2$ ▷ compute loss
 - 14 $\Delta \mathbf{W} \leftarrow -\eta \text{G}_{\text{Adam}}(\nabla_{\mathbf{W}} \mathcal{J}), \Delta \mathbf{b} \leftarrow -\eta \text{G}_{\text{Adam}}(\nabla_{\mathbf{b}} \mathcal{J})$ ▷ Adam optimizer step
 - 15 $\mathbf{W} \leftarrow \mathbf{W} + \Delta \mathbf{W}, \mathbf{b} \leftarrow \mathbf{b} + \Delta \mathbf{b}$ ▷ update weights and biases
 - 16 **end**
 - 17 **end**
 - 18 $\mathbf{W}_{\text{tr}} \leftarrow \mathbf{W}, \mathbf{b}_{\text{tr}} \leftarrow \mathbf{b}$ ▷ save optimal weights
 - 19 **end**
-

5 Approximation of the reduced-order matrices based on CSI

5.1 Cubic spline interpolation

Given some interpolation nodes $a = x_0 < x_1 < \dots < x_n = b$, and their corresponding function values $f(x_k) = y_k, k = 0, 1, 2, \dots, n$. A smooth function $S(x)$ is said to be a cubic spline function of $f(x)$ if $S(x)$ satisfies the following three conditions

$$\begin{cases} S(x) \in \mathbb{C}^2[a, b], \\ S(x_k) = f(x_k) = y_k, k = 0, 1, 2, \dots, n, \\ S(x) \text{ is a cubic polynomial in } [x_k, x_{k+1}], k = 0, 1, 2, \dots, n-1. \end{cases} \quad (21)$$

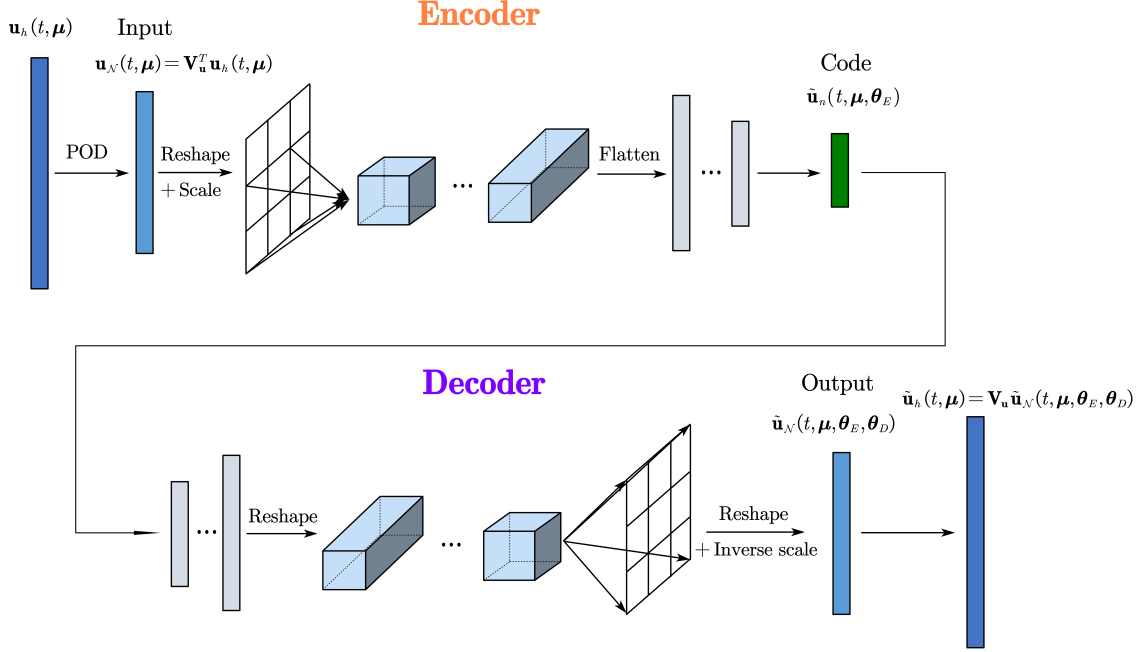


Figure 2: Architecture of a deep convolutional autoencoder network used for parameterized electromagnetic scattering problems, which takes a projection coefficient as an input and produces an approximate projection coefficient as an output. A low dimensional code $\tilde{\mathbf{u}}_n(t, \boldsymbol{\mu}, \boldsymbol{\theta}_E)$ is extracted via the encoder from the projection coefficient $\mathbf{u}_N(t, \boldsymbol{\mu})$. The decoder reconstructs an approximate projection coefficient $\tilde{\mathbf{u}}_N(t, \boldsymbol{\mu}, \boldsymbol{\theta}_E, \boldsymbol{\theta}_D)$ from the low dimensional code $\tilde{\mathbf{u}}_n(t, \boldsymbol{\mu}, \boldsymbol{\theta}_E)$ by performing the inverse operations of the encoder.

Let $S_k(x)$ be the expression of $S(x)$ in the interval $[x_k, x_{k+1}]$, $k = 0, 1, 2, \dots, n-1$, which includes $4n$ unknowns. Based on (21), we have $2n$ interpolation conditions

$$S_k(x_k) = y_k, S_k(x_{k+1}) = y_{k+1}, k = 0, 1, 2, \dots, n-1, \quad (22)$$

and $2(n-1)$ differential continuity conditions

$$S'_{k-1}(x_k^-) = S'_k(x_k^+), S''_{k-1}(x_k^-) = S''_k(x_k^+), k = 1, 2, \dots, n-1. \quad (23)$$

We cannot obtain the cubic spline function $S(x)$ according to (22) and (23) because two more conditions are required. In this study, we consider the not-a-knot conditions [1]

$$S'''_0(x_1^-) = S'''_1(x_1^+), S'''_{n-2}(x_{n-1}^-) = S'''_{n-1}(x_{n-1}^+). \quad (24)$$

Combined with the conditions (22)-(24), the cubic polynomial $S_k(x)$ ($k = 0, 1, \dots, n-1$) can be obtained by solving a system of linear equations of order $n+1$. In particular, we can extend the same multivariate analysis through the widely used tensor product formulation [17, 16]. To determine the interpolated value at a desired point, we use cubic interpolation of the values at the closest knot points in each respective dimension. The number of independent variables in the multivariate CSI method is not limited. For more detailed definition of multivariate CSI method we refer to [42].

5.2 CSI-based approximation of reduced-order matrix

In this subsection, we construct a mapping from time/parameters to the low dimensional coded representation vectors $\tilde{\mathbf{u}}_n(t_i, \boldsymbol{\mu}_j) = [\omega_1(t_i, \boldsymbol{\mu}_j), \omega_2(t_i, \boldsymbol{\mu}_j), \dots, \omega_n(t_i, \boldsymbol{\mu}_j)]$ ($i = 1, 2, \dots, \mathcal{N}_t, j = 1, 2, \dots, \mathcal{N}_p$). After the convolutional autoencoder network is trained, we get all the \mathcal{N}_s low dimensional coded representation vectors

$\tilde{\mathbf{u}}_n(t_i, \boldsymbol{\mu}_j) \in \mathbb{R}^n$. Reshape them to n reduced-order matrices with the form

$$\mathbf{P}_l = \begin{pmatrix} \omega_l(t_{n_1}, \boldsymbol{\mu}_1) & \omega_l(t_{n_1}, \boldsymbol{\mu}_2) & \cdots & \omega_l(t_{n_1}, \boldsymbol{\mu}_{\mathcal{N}_p}) \\ \omega_l(t_{n_2}, \boldsymbol{\mu}_1) & \omega_l(t_{n_2}, \boldsymbol{\mu}_2) & \cdots & \omega_l(t_{n_2}, \boldsymbol{\mu}_{\mathcal{N}_p}) \\ \vdots & \vdots & \ddots & \vdots \\ \omega_l(t_{n_{\mathcal{N}_t}}, \boldsymbol{\mu}_1) & \omega_l(t_{n_{\mathcal{N}_t}}, \boldsymbol{\mu}_2) & \cdots & \omega_l(t_{n_{\mathcal{N}_t}}, \boldsymbol{\mu}_{\mathcal{N}_p}) \end{pmatrix} \in \mathbb{R}^{\mathcal{N}_t \times \mathcal{N}_p}, l = 1, 2, \dots, n. \quad (25)$$

It is difficult to directly fit the above matrix in two dimensions under acceptable accuracy conditions. So we resort to a SVD to decompose \mathbf{P}_l into several time- and parameter-modes [14]

$$\mathbf{P}_l \approx \tilde{\mathbf{P}}_l = \sum_{k=1}^{q_l} \sigma_k^l \boldsymbol{\psi}_k^l (\boldsymbol{\phi}_k^l)^T, \quad 1 \leq l \leq n, \quad (26)$$

where $\boldsymbol{\psi}_k^l$ and $\boldsymbol{\phi}_k^l$ are the k th discrete time- and parameter- modes for the l th reduced-order matrix respectively, σ_k^l is the k th singular value, and q_l is the truncation rank corresponding to the error tolerance δ , i.e., $q_l = \operatorname{argmin} \{\pi(q_l) : \pi(q_l) \geq 1 - \delta\}$ with $\pi(q_l) = \sum_{k=1}^{q_l} (\sigma_k^l)^2 / \sum_{k=1}^{r_l} (\sigma_k^l)^2$ and r_l being the rank of \mathbf{P}_l . With the discrete modes database, CSI models can be trained to approximate the continuous modes as

$$\begin{aligned} t &\mapsto \hat{\boldsymbol{\psi}}_k^l(t), \text{ trained from } \{(t_i, (\boldsymbol{\psi}_k^l)_i), i = 1, 2, \dots, \mathcal{N}_t\}, \\ \boldsymbol{\mu} &\mapsto \hat{\boldsymbol{\phi}}_k^l(\boldsymbol{\mu}), \text{ trained from } \{(\boldsymbol{\mu}_j, (\boldsymbol{\phi}_k^l)_j), j = 1, 2, \dots, \mathcal{N}_p\}. \end{aligned} \quad (27)$$

Hence, we have

$$(\mathbf{P}_l)_{ij} \approx \sum_{k=1}^{q_l} \sigma_k^l \hat{\boldsymbol{\psi}}_k^l(t_i) \hat{\boldsymbol{\phi}}_k^l(\boldsymbol{\mu}_j), \quad (28)$$

with $1 \leq i \leq \mathcal{N}_t, 1 \leq j \leq \mathcal{N}_p$. The architecture of the CSI and decoder in the online stage is shown in Fig. 3. For a new time/parameter value $(t^*, \boldsymbol{\mu}^*)$, we can rapidly get the low-dimensional coded representation $\tilde{\mathbf{u}}_n(t^*, \boldsymbol{\mu}^*) = [\omega_1(t^*, \boldsymbol{\mu}^*), \omega_2(t^*, \boldsymbol{\mu}^*), \dots, \omega_n(t^*, \boldsymbol{\mu}^*)]$ via CSI. Then the approximation of the projection coefficient $\hat{\alpha}_{\mathbf{u}}(t^*, \boldsymbol{\mu}^*)$ is obtained by the decoder network. The reduced-order solution $\mathbf{u}_h^r(t, \boldsymbol{\mu})$ served as an approximation to the high-fidelity solution $\mathbf{u}_h(t, \boldsymbol{\mu})$ can be recovered by

$$\mathbf{u}_h(t^*, \boldsymbol{\mu}^*) \approx \mathbf{u}_h^r(t^*, \boldsymbol{\mu}^*) = \mathbf{V}_{\mathbf{u}} \hat{\alpha}_{\mathbf{u}}(t^*, \boldsymbol{\mu}^*), \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}. \quad (29)$$

Algorithm 3: POD-CAE-CSI method for electromagnetic simulations

1 **Function** $[\mathbf{V}_{\mathbf{u}}, f_E, f_D, \hat{\omega}] = \text{POD-CAE-CSI-Offline}(\mathcal{P}, \mathcal{T}, \Omega)$

2 Prepare the parameter sampling $\mathcal{P}_h^{tr} \subset \mathcal{P}$;

3 Calculate the high-fidelity solutions $\mathbf{u}_h(t_i, \boldsymbol{\mu}_j)$ via the DGTD solver in the time domain \mathcal{T} ,
 $j = 1, 2, \dots, \mathcal{N}_p, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$;

4 Form the snapshot matrix $\mathbf{S}_{\mathbf{u}}^j$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$), and prepare the time sampling $\mathcal{T}_h^{tr} \subset \mathcal{T}$;

5 Generate the POD basis matrix $\mathbf{V}_{\mathbf{u}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) via Algorithm 1;

6 Train the autoencoder network f_E and f_D via Algorithm 2 ;

7 Build the CSI-based interpolation model $\hat{\omega}$.

8 **end**

9 **Function** $\mathbf{u}_h^r(t^*, \boldsymbol{\mu}^*) = \text{POD-CAE-CSI-Online}(\mathbf{V}_{\mathbf{u}}, f_D, \hat{\omega}, (t^*, \boldsymbol{\mu}^*))$

10 Compute the approximate low dimensional coded representation $\tilde{\mathbf{u}}_n(t^*, \boldsymbol{\mu}^*)$ for a new input
 $(t^*, \boldsymbol{\mu}^*)$ through the CSI-based model $\hat{\omega}$;

11 Recover the approximate projection coefficient $\hat{\alpha}_{\mathbf{u}}(t^*, \boldsymbol{\mu}^*)$ through the decoder f_D ;

12 Evaluate the reduced-order solution $\mathbf{u}_h^r(t^*, \boldsymbol{\mu}^*)$ based on 29.

13 **end**

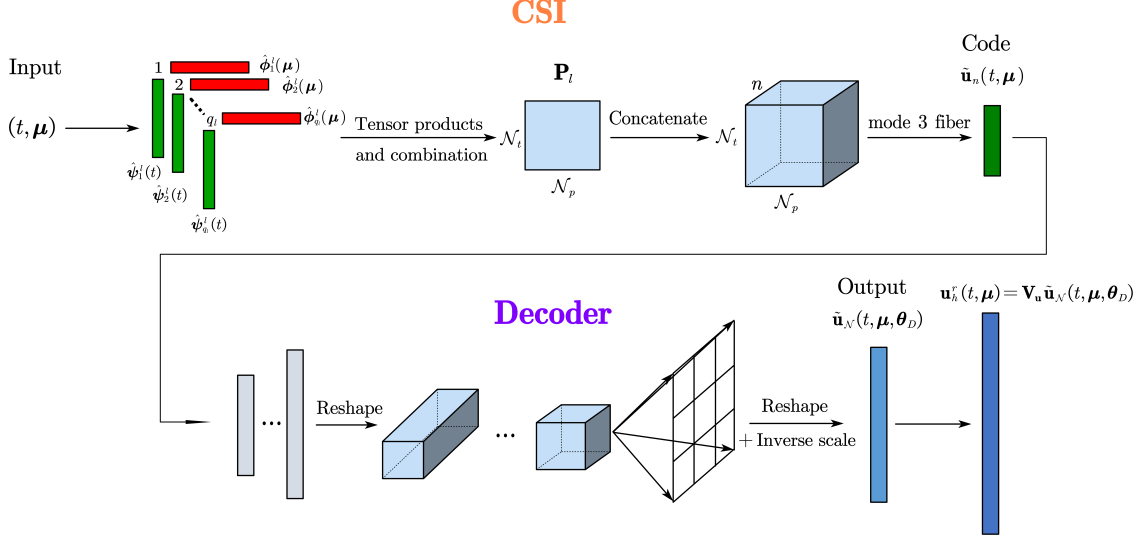


Figure 3: Architecture of the CSI and decoder in the online stage. The CSI-based model produces a low dimensional code $\tilde{\mathbf{u}}_n(t, \boldsymbol{\mu})$ for a new input $(t, \boldsymbol{\mu})$, then the decoder reconstructs an approximate projection coefficient $\tilde{\mathbf{u}}_{\mathcal{N}}(t, \boldsymbol{\mu}, \boldsymbol{\theta}_D)$ from the code $\tilde{\mathbf{u}}_n(t, \boldsymbol{\mu})$.

6 Numerical experiments

In this section, numerical experiments for electromagnetic scattering problems are used to evaluate the proposed CAE-CSI ROM. We consider the 2-D time-domain Maxwell equations in the case of transverse magnetic (TM) waves

$$\begin{cases} \nu_r \frac{\partial H_x}{\partial t} + \frac{\partial E_z}{\partial y} = 0, \\ \nu_r \frac{\partial H_y}{\partial t} - \frac{\partial E_z}{\partial x} = 0, \\ \varepsilon_r \frac{\partial E_z}{\partial t} - \frac{\partial H_y}{\partial x} + \frac{\partial H_x}{\partial y} = 0. \end{cases} \quad (30)$$

The excitation is an incident plane wave which is defined as

$$\begin{cases} H_x^{\text{inc}}(x, y, t) = 0, \\ H_y^{\text{inc}}(x, y, t) = -\cos(\omega t - kx), \\ E_z^{\text{inc}}(x, y, t) = \cos(\omega t - kx), \end{cases} \quad (31)$$

where $\omega = 2\pi f$ denotes the angular frequency with the wave frequency $f = 30$ GHz, and $k = \frac{\omega}{c}$ is the wave number with c being the wave speed in vacuum.

We determine the truncation parameter k and size parameter \mathcal{N} in the two-step POD method by the following error indicator

$$\bar{e}_{\mathbf{u}, \text{POD}} = \frac{\sum_{(t, \boldsymbol{\mu}) \in \mathcal{T}_h^{tr} \times \mathcal{P}_h^{tr}} \|\mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbf{V}_{\mathbf{u}} \mathbf{V}_{\mathbf{u}}^T \mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}{\mathcal{N}_s \|\mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}. \quad (32)$$

The relative error between the reduced-order solution generated by CAE-CSI and the high-fidelity solution is used as the metric to evaluate the accuracy of the results

$$e_{\mathbf{u}, \text{CAE-CSI}}(t, \boldsymbol{\mu}) = \frac{\|\mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbf{u}_h^r(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}{\|\mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}} = \frac{\|\mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbf{V}_{\mathbf{u}} \hat{\alpha}_{\mathbf{u}}(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}{\|\mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}, \quad (33)$$

which will be compared with the relative projection error

$$e_{\mathbf{u}, \text{Pro}}(t, \boldsymbol{\mu}) = \frac{\|\mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbf{u}_h^p(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}{\|\mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}} = \frac{\|\mathbf{u}_h(t, \boldsymbol{\mu}) - \mathbf{V}_{\mathbf{u}} \mathbf{V}_{\mathbf{u}}^T \mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}{\|\mathbf{u}_h(t, \boldsymbol{\mu})\|_{\mathbb{R}^{\mathcal{N}_h}}}, \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}. \quad (34)$$

The above errors are evaluated on a testing time/parameter sampling $\mathcal{T}_h^{te} \times \mathcal{P}_h^{te}$ of size \mathcal{N}_{te} . Furthermore, the average relative errors are used to measure the accuracy of the ROM in our numerical experiments, which are defined as

$$\bar{e}_{\mathbf{u},\text{CAE-CSI}} = \frac{\sum_{(t,\boldsymbol{\mu}) \in \mathcal{T}_h^{te} \times \mathcal{P}_h^{te}} e_{\mathbf{u},\text{CAE-CSI}}(t, \boldsymbol{\mu})}{\mathcal{N}_{te}}, \quad \bar{e}_{\mathbf{u},\text{Pro}} = \frac{\sum_{(t,\boldsymbol{\mu}) \in \mathcal{T}_h^{te} \times \mathcal{P}_h^{te}} e_{\mathbf{u},\text{Pro}}(t, \boldsymbol{\mu})}{\mathcal{N}_{te}}. \quad (35)$$

The CSI-based approximation of the reduced-order matrices are built via the SciPy functions CubicSpline and RegularGridInterpolator in the first and second numerical experiments respectively. The DGTD and two-step POD methods are implemented in MATLAB, while CAE-CSI is developed in Python. All simulations are run on a computer equipped with an Intel Core i9 10-core 2.8 GHz \times 20 CPU with 64 GB RAM.

6.1 Scattering of a plane wave by a dielectric disk

The first numerical experiment is the electromagnetic scattering of a plane wave by a dielectric disk. The computational domain is artificially bounded by a square $\Omega = [-2.6 \text{ m}, 2.6 \text{ m}] \times [-2.6 \text{ m}, 2.6 \text{ m}]$ where we impose the Silver-Müller ABC boundary condition. The range of the relative permittivity is $\varepsilon_r \in [1.0, 5.0]$ (i.e., $\mathcal{P} = [1.0, 5.0]$) with the relative permeability $\nu_r = 1.0$, i.e., we consider a nonmagnetic material. The medium outside to the dielectric disk is assumed to be vacuum, i.e. $\varepsilon_r = \nu_r = 1.0$. The high-fidelity simulations are performed on an unstructured triangular mesh with 2575 nodes and 5044 elements, in which 1092 elements are located inside the disk. And we use a DGTD method with \mathbb{P}_2 approximation, resulting in $\mathcal{N}_h = 30264$ DOFs for the FOM.

During the offline stage, the DGTD solver is used to generate a collection of high-fidelity solutions at $\mathcal{N}_p = 81$ equidistant parameter sampling points (i.e., $\boldsymbol{\mu} \in \mathcal{P}_h^{tr} = \{1.0, 1.05, \dots, 4.95, 5.0\}$) with $\mathcal{N}_t = 263$ points in the last oscillation period (i.e., $t \in \mathcal{T}_h^{tr} = \{49.0024, 49.006, \dots, 49.9623, 49.966\}$). A test parameter set $\mathcal{P}_h^{te} = \{1.215, 2.215, 3.215, 4.215\}$ and test time set $\mathcal{T}_h^{te} = \mathcal{T}_h^{tr}$ are used to evaluate the proposed method.

To perform a moderate dimensionality reduction, we utilize the two-step POD method to extract the basis functions from the the snapshot matrix. Fig. 4 shows the convergence histories of $\bar{e}_{\mathbf{H},\text{POD}}$ and $\bar{e}_{\mathbf{E},\text{POD}}$ with the choice of k and \mathcal{N} in the two-step POD method. We obtain the POD basis matrices $\mathbf{V}_{\mathbf{u}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) with $k = 4$ and $\mathcal{N} = 196$. Then we train the CAE network, and Fig. 5 left shows the behavior of the loss on validation set with respect to the reduced dimension n . By increasing the dimension n of the low-dimensional coded representation vector $\tilde{\mathbf{u}}(t, \boldsymbol{\mu})$ from 3 to 20, there is a significant improvement of the CAE performance. However, when $n > 20$, the loss on the validation set does not change significantly. Thus we set the dimensionality of the low-representation coded vector to $n = 20$. Fig. 5 right shows the impact of the size of the convolutional kernels on the loss over the validation set. Tabs. 1 and 2 summarize attributes of the convolutional and transposed convolutional layers. As for the training of CSI models, we set the SVD truncation tolerance to $\delta = 1 \times 10^{-4}$. In order to assess the performance of CAE-CSI model, the reduced-

Table 1: Architecture of convolutional and fully-connected layers in the encoder f_E .

Layer	Input Dim	Output dim	Kernel size	Number of kernels	Stride	Padding
1	[14, 14, 3]	[14, 14, 8]	[5, 5]	8	1	2
2	[14, 14, 8]	[8, 8, 16]	[5, 5]	16	2	3
3	[8, 8, 16]	[4, 4, 32]	[5, 5]	32	2	2
4	[4, 4, 32]	[2, 2, 64]	[5, 5]	64	2	2
5	256	256				
6	256	256				
7	256	n				

order solutions are compared with the corresponding DGTD high-fidelity solutions on the test parameter set $\mathcal{P}_h^{te} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3, \boldsymbol{\mu}_4\}$ where $\boldsymbol{\mu}_1 = \varepsilon_1 = 1.215$, $\boldsymbol{\mu}_2 = \varepsilon_2 = 2.215$, $\boldsymbol{\mu}_3 = \varepsilon_3 = 3.215$ and $\boldsymbol{\mu}_4 = \varepsilon_4 = 4.215$. Firstly, Fig. 6 shows the exact reduced-order matrices and approximate reduced-order matrices based on CSI. Over the Fourier domain during the last oscillation period of the incident wave, we display in Fig. 7 the 1-D x -wise evolution along $y = 0$ of the real part of H_y and E_z , as well as their 2-D distribution in Fig. 8 and Fig. 9. The time evolution of the relative projection error $e_{\mathbf{u},\text{Pro}}$ and CAE-CSI error $e_{\mathbf{u},\text{CAE-CSI}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) for the test parameter instances $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3$ and $\boldsymbol{\mu}_4$ are shown in Fig. 10. The average relative error $\bar{e}_{\mathbf{u},\text{Pro}}$ and

Table 2: Architecture of fully-connected and transposed convolutional layers in the decoder f_D .

Layer	Input Dim	Output dim	Kernel size	Number of kernels	Stride	Padding
1	n	256				
2	256	256				
3	256	256				
4	[2, 2, 64]	[4, 4, 64]	[5, 5]	64	1	1
5	[4, 4, 64]	[8, 8, 32]	[5, 5]	32	1	0
6	[8, 8, 32]	[14, 14, 16]	[5, 5]	16	3	6
7	[14, 14, 16]	[14, 14, 3]	[5, 5]	8	1	2

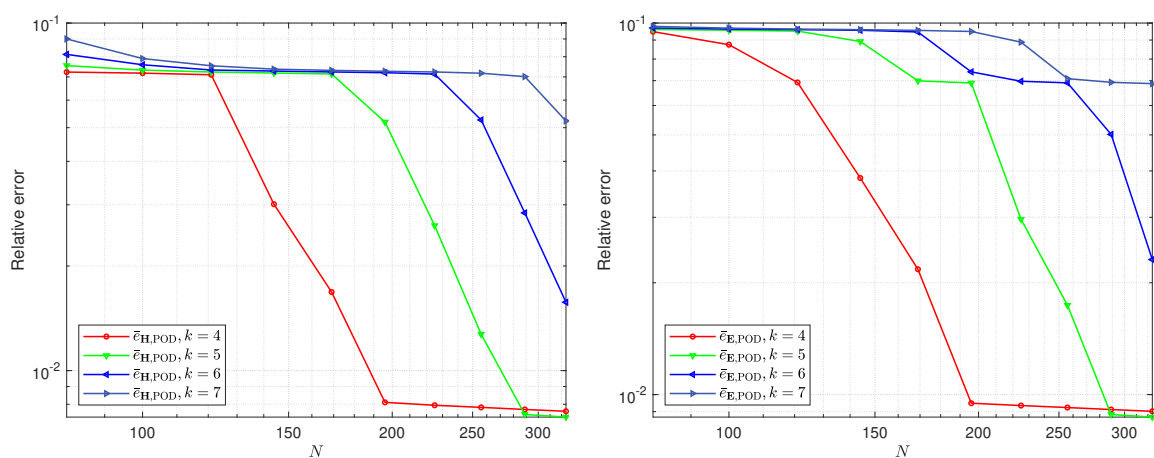


Figure 4: Scattering of a plane wave by a dielectric disk. Convergence histories of $\bar{e}_{\mathbf{H},\text{POD}}$ (left) and $\bar{e}_{\mathbf{E},\text{POD}}$ (right) with different truncation parameters k and N .

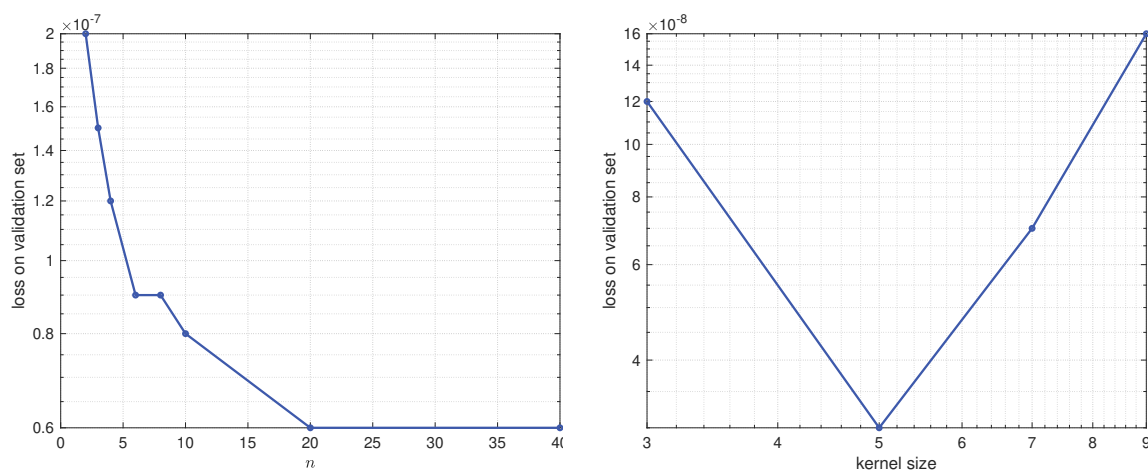


Figure 5: Left: loss on validation set versus n . Right: loss on validation set versus kernel size.

$\bar{\epsilon}_{\mathbf{u},\text{CAE-CSI}}(\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\})$ for the four test parameters are shown in Tab. 3. It can be seen that the reduced-order solutions and the DGTD solutions match each other very well. Secondly, Tab. 4 presents the time performance comparison of DGTD and CAE-CSI, where we record the average online test time of DGTD solver and ROM for the above four test cases, as well as the training time of the CAE-CSI model. It is worthwhile to take a long time to build a surrogate model, since the online test time of the ROM is greatly shortened compared to the DGTD solver, achieving a speed-up of 3387. Note that the online test time of CAE-CSI is shortened compared to previous work (POD-CSI) [30].

Table 3: Scattering of a plane wave by a dielectric disk. Average relative error on the test set.

$\bar{\epsilon}_{\mathbf{H},\text{Pro}}$	$\bar{\epsilon}_{\mathbf{H},\text{CAE-CSI}}$	$\bar{\epsilon}_{\mathbf{E},\text{Pro}}$	$\bar{\epsilon}_{\mathbf{E},\text{CAE-CSI}}$
1.15%	1.37%	1.56%	1.69%

Table 4: Scattering of a plane wave by a dielectric disk. Comparison between the CAE-CSI (offline and online) and DGTD methods in terms of CPU time. The unit of time is second.

Offline			Online		
Snapshots	Two-step POD	CAE-CSI	CAE-CSI	POD-CSI	DGTD
2.088×10^4	6.0530	1.9042×10^3	0.0761	0.2561	2.5778×10^2

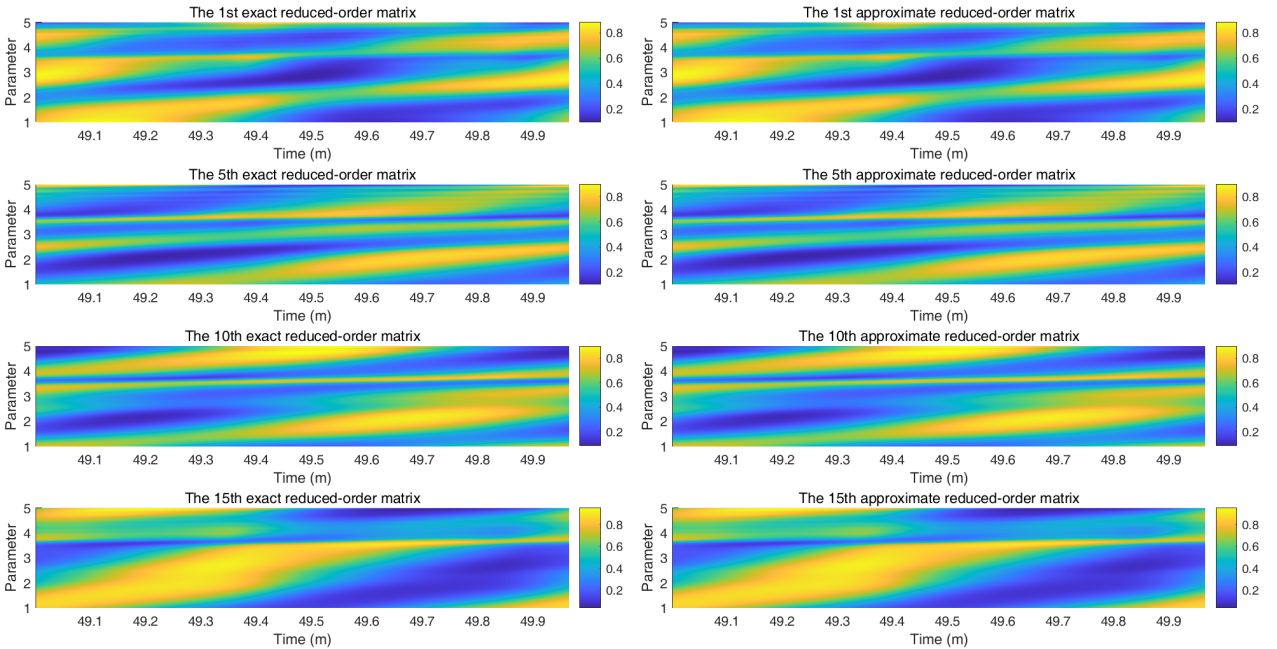


Figure 6: Scattering of a plane wave by a dielectric disk. The 1-st, 5-th, 10-th, and 15-th exact and approximate reduced-order matrices based on CSI.

6.2 Scattering of a plane wave by a multi-layer heterogeneous medium

In this subsection, we consider a multi-layer heterogeneous medium which is illuminated by an incident plane wave as shown in Fig. 11. The computational domain is artificially bounded by a square $\Omega = [-3.2 \text{ m}, 3.2 \text{ m}] \times [-3.2 \text{ m}, 3.2 \text{ m}]$ where we impose the Silver-Müller ABC boundary condition. Tab. 5 summarizes the distribution and range of material parameters considered in this study. The mesh consists of 3256 nodes and 6206

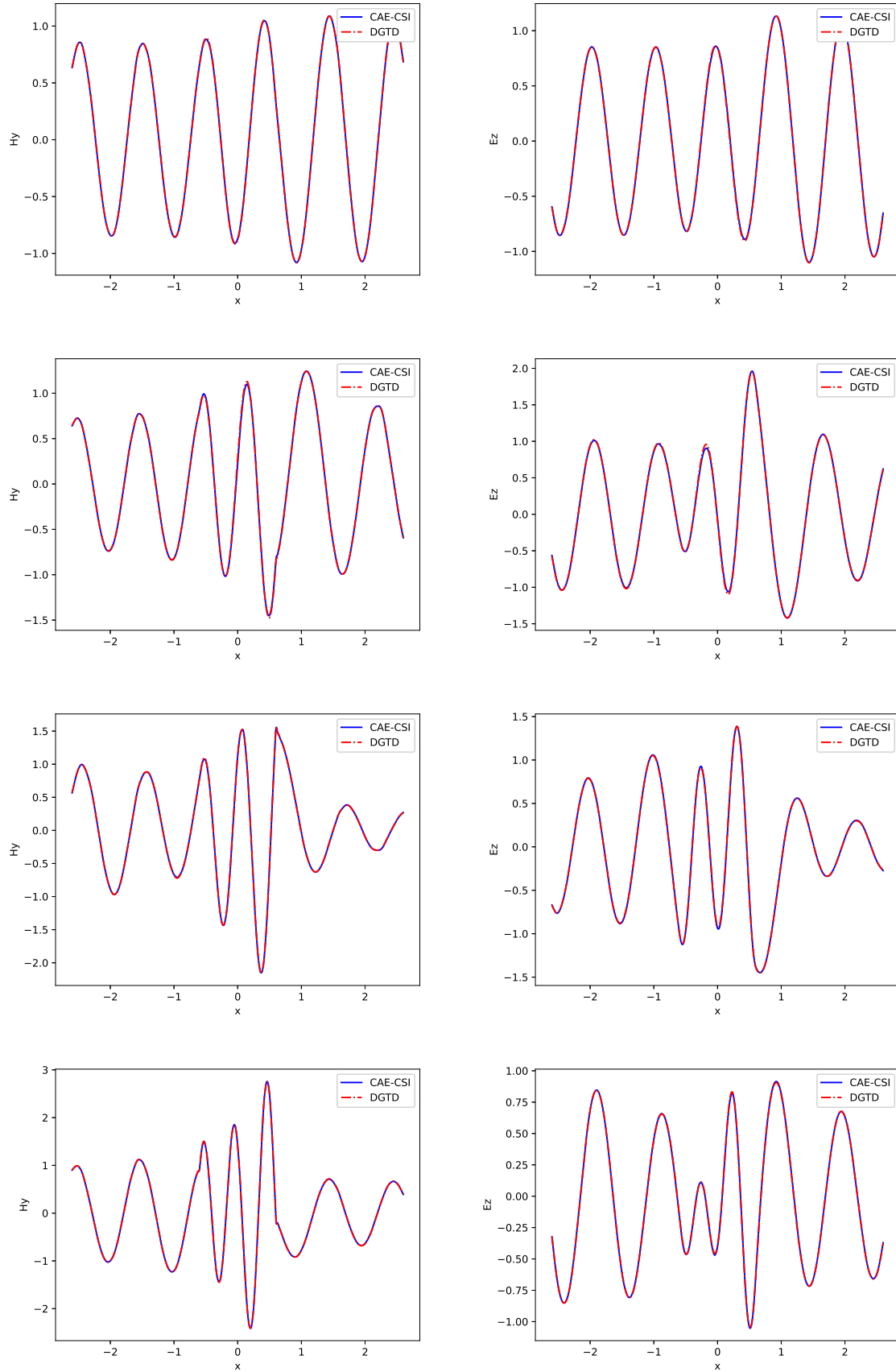


Figure 7: Scattering of a plane wave by a dielectric disk. Comparison of the 1-D x -wise distribution along $y = 0$ of the real part of H_y (left) and E_z (right) for the testing parameter instances: $\varepsilon_1 = 1.215$ (1st row), $\varepsilon_2 = 2.215$ (2th row), $\varepsilon_3 = 3.215$ (3th row), $\varepsilon_4 = 4.215$ (4th row).

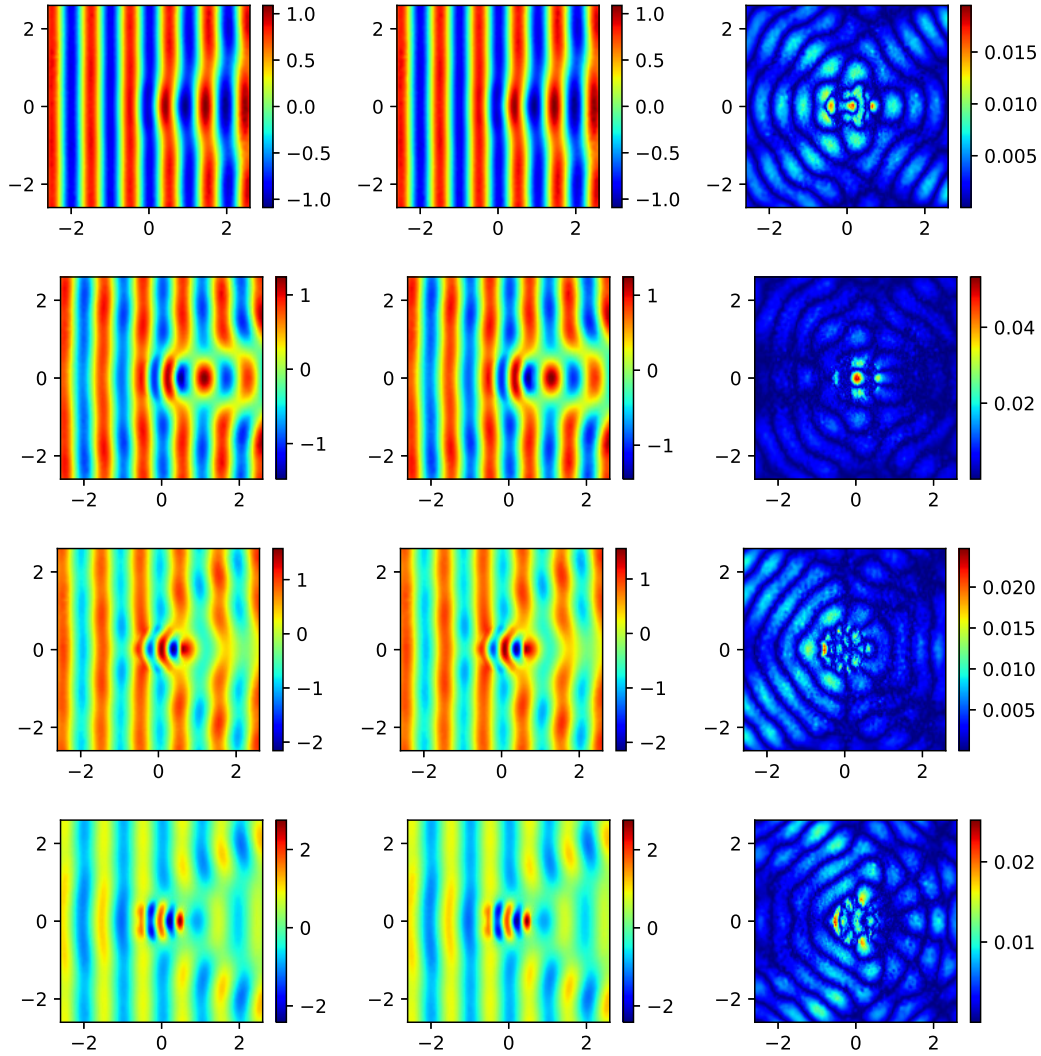


Figure 8: Scattering of a plane wave by a dielectric disk. Comparison of the 2-D distribution of the real part of H_y between DGTD (left), CAE-CSI (middle) and relative error (right) for the testing parameter instances: $\varepsilon_1 = 1.215$ (1st row), $\varepsilon_2 = 2.215$ (2th row), $\varepsilon_3 = 3.215$ (3th row), $\varepsilon_4 = 4.215$ (4th row).

Table 5: Scattering of plane wave by a multi-layer heterogeneous medium. Distribution and range of material parameters.

Layer i	\mathcal{P}^i	$\nu_{r,i}$	r_i
1	$\varepsilon_{r,1} \in [5.0, 5.6]$	1.0	0.15
2	$\varepsilon_{r,2} \in [3.25, 3.75]$	1.0	0.3
3	$\varepsilon_{r,3} \in [2.0, 2.5]$	1.0	0.45
4	$\varepsilon_{r,4} \in [1.25, 1.75]$	1.0	0.6

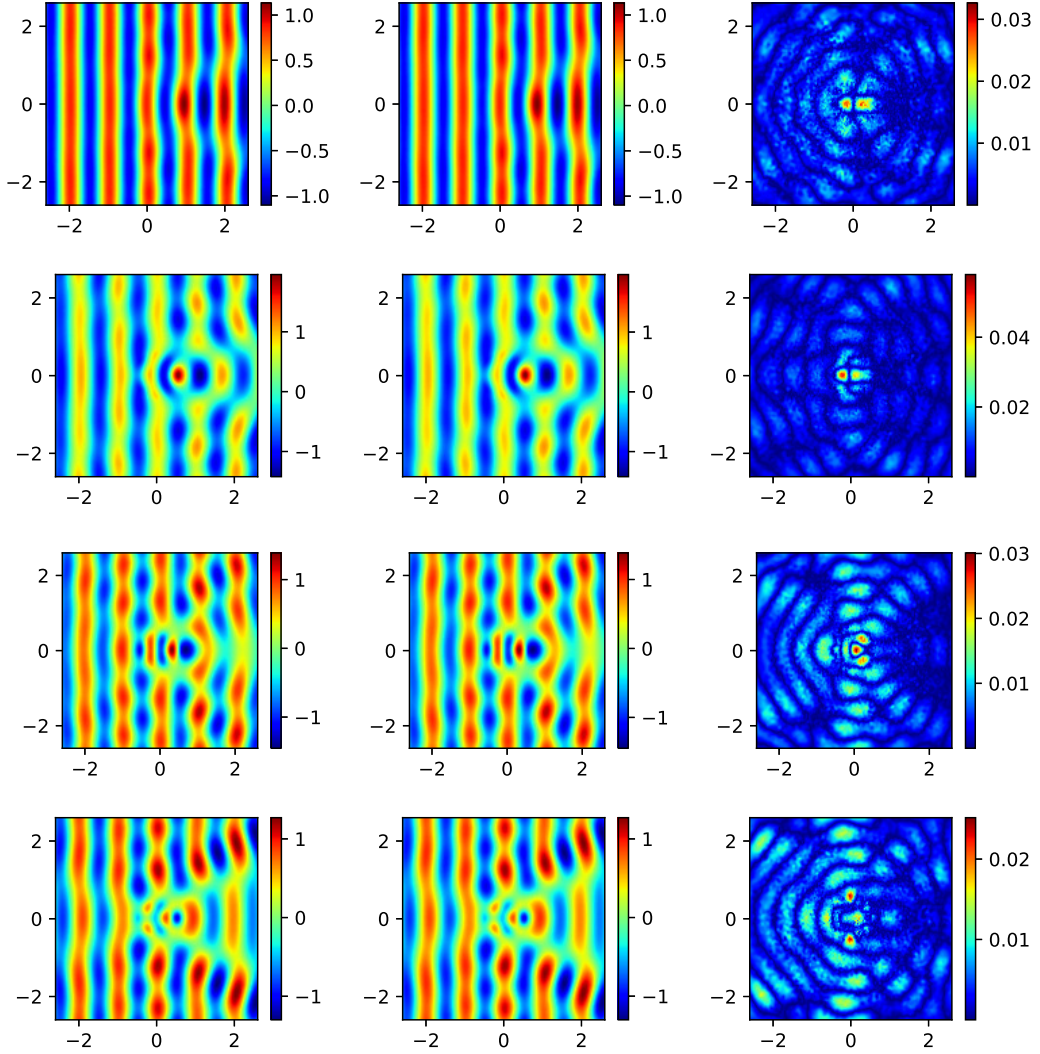


Figure 9: Scattering of a plane wave by a dielectric disk. Comparison of the 2-D distribution of the real part of E_z between DGTD (left), CAE-CSI (middle) and relative error (right) for the testing parameter instances: $\varepsilon_1 = 1.215$ (1st row), $\varepsilon_2 = 2.215$ (2th row), $\varepsilon_3 = 3.215$ (3th row), $\varepsilon_4 = 4.215$ (4th row).

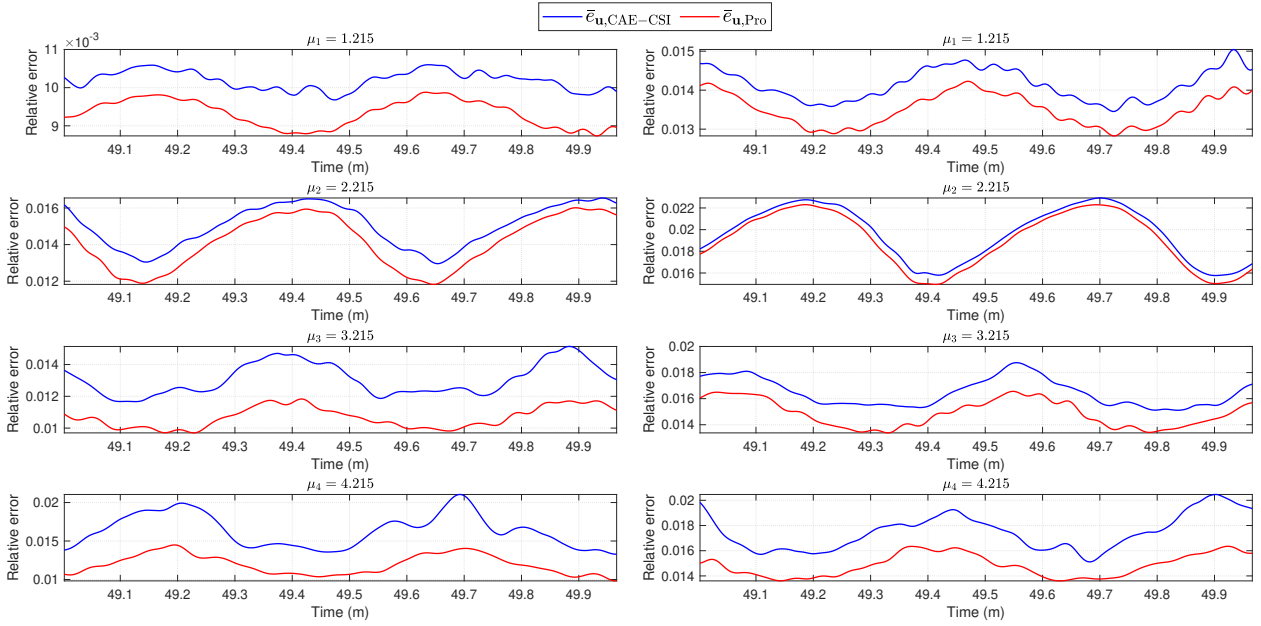


Figure 10: Scattering of a plane wave by a dielectric disk. Comparison of the relative error between DGTD and CAE-CSI for the field \mathbf{H} (left) and \mathbf{E} (right) with the testing parameter instances $\varepsilon_1 = 1.215$ (1st row), $\varepsilon_2 = 2.215$ (2th row), $\varepsilon_3 = 3.215$ (3th row), $\varepsilon_4 = 4.215$ (4th row).

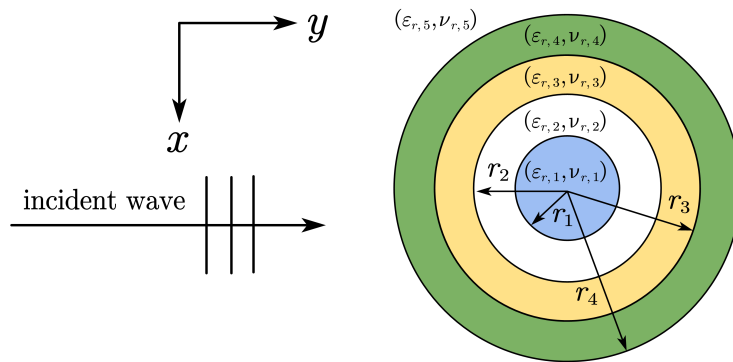


Figure 11: Scattering of plane wave by a multi-layer heterogeneous medium. Geometry of the multi-layer heterogeneous medium.

elements, resulting in $\mathcal{N}_h = 37236$ DOFs of the FOM. The DGTD solver is used to generate the snapshots, and the numerical simulation time is set to 50 periods of the incident wave oscillation with time step $\Delta t = 0.0038$ s for each 4-dimensional parameter $\boldsymbol{\mu} \in \mathcal{P}$, where $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \times \mathcal{P}^3 \times \mathcal{P}^4 \subset \mathbb{R}^4$ with $\varepsilon_{r,i} \in \mathcal{P}^i$ ($i = 1, 2, 3, 4$). We adopt a grid sampling of tensor product with $\mathcal{N}_{\Delta p} = 3$ uniform points for each parameter to form a training parameter samples \mathcal{P}_h^{tr} , resulting in $\mathcal{N}_p = 81$ points. Each choice of the parameter is sampled for $\mathcal{N}_t = 253$ snapshots in time at the last period, i.e., $\mathcal{T}_h^{tr} = \{49.0002, 49.0041, \dots, 49.9630, 49.9669\}$. Fig. 12 shows the convergence histories of $\bar{e}_{\mathbf{H},\text{POD}}$ and $\bar{e}_{\mathbf{E},\text{POD}}$ with different truncation parameter k and size parameter \mathcal{N} in the two-step POD method. The POD basis matrices $\mathbf{V}_{\mathbf{u}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) are generated by the two-step POD method with $k = 4$ and $\mathcal{N} = 196$. The architecture of the CAE network is the same as the one used in the first experiment. After performing a SVD to all reduced-order matrices with a truncation tolerance $\delta = 1 \times 10^{-4}$, the CSI models are built as the combination of time- and parameter-modes. To evaluate the CAE-CSI method, we compare the reduced-order solution with the high-fidelity solution generated by the DGTD method on a test parameter set $\mathcal{P}_h^{te} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3\}$ with $\boldsymbol{\mu}_1 = \{(5.1, 3.4, 2.1, 1.4)\}$, $\boldsymbol{\mu}_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ and $\boldsymbol{\mu}_3 = \{(5.5, 3.7, 2.4, 1.7)\}$. Fig. 13 shows the 1-D x -wise distribution along $y = 0$ of the real part of H_y and E_z in the Fourier domain on the last period of simulation. Fig. 14 and Fig. 15 display the 2-D distribution of the real part of H_y and E_z . The time evolution of the relative projection error $e_{\mathbf{u},\text{Pro}}$ and CAE-CSI error $e_{\mathbf{u},\text{CAE-CSI}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) for 3 test parameters $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_3$ are displayed in Fig. 16. The average projection error $\bar{e}_{\mathbf{u},\text{Pro}}$ and the average CAE-CSI error $\bar{e}_{\mathbf{u},\text{CAE-CSI}}$ ($\mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$) for the 3 test parameters are listed in Tab. 6. Performance results of the CAE-CSI and DGTD methods with \mathbb{P}_2 approximation are summarized

Table 6: Scattering of plane wave by a multi-layer heterogeneous medium. Average relative error on the test set.

$\bar{e}_{\mathbf{H},\text{Pro}}$	$\bar{e}_{\mathbf{H},\text{CAE-CSI}}$	$\bar{e}_{\mathbf{E},\text{Pro}}$	$\bar{e}_{\mathbf{E},\text{CAE-CSI}}$
0.60%	0.89%	0.69%	1.03%

in Tab. 7. The CPU time of the DGTD method is 3.0477×10^2 s and the online cost of the CAE-CSI is 0.2923 s, corresponding to a speed-up of 1042. Note that the online test time of CAE-CSI is again shortened compared to previous work (POD-CSI) [30], which again demonstrates the significantly enhanced efficiency of the CAE-CSI method.

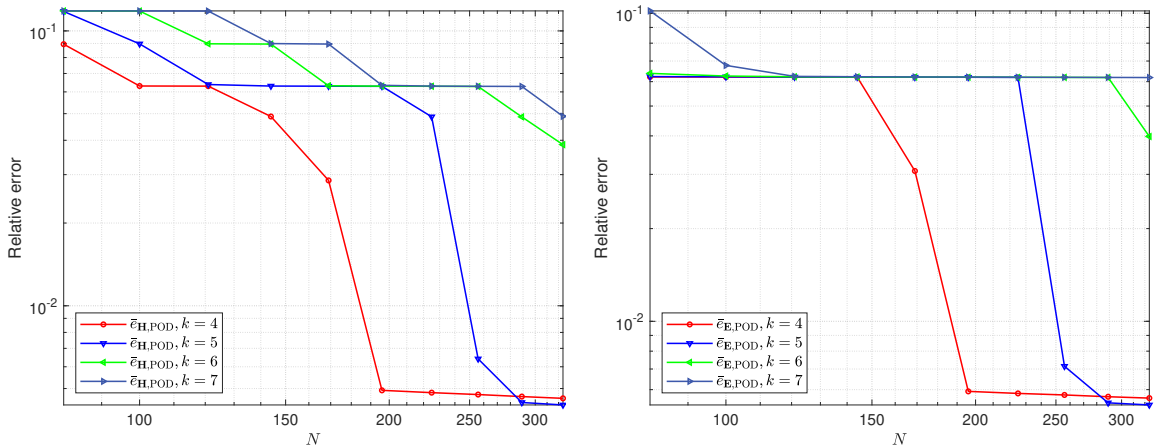


Figure 12: Scattering of plane wave by a multi-layer heterogeneous medium. Convergence histories of $\bar{e}_{\mathbf{H},\text{POD}}$ (left) and $\bar{e}_{\mathbf{E},\text{POD}}$ (right) with the choice of k and \mathcal{N} .

7 Conclusion

A data-driven RB method based on POD, CAE and CSI is proposed to accelerate the solution of parameterized time-domain Maxwell equations. The two-step POD method performs a prior dimensionality reduction

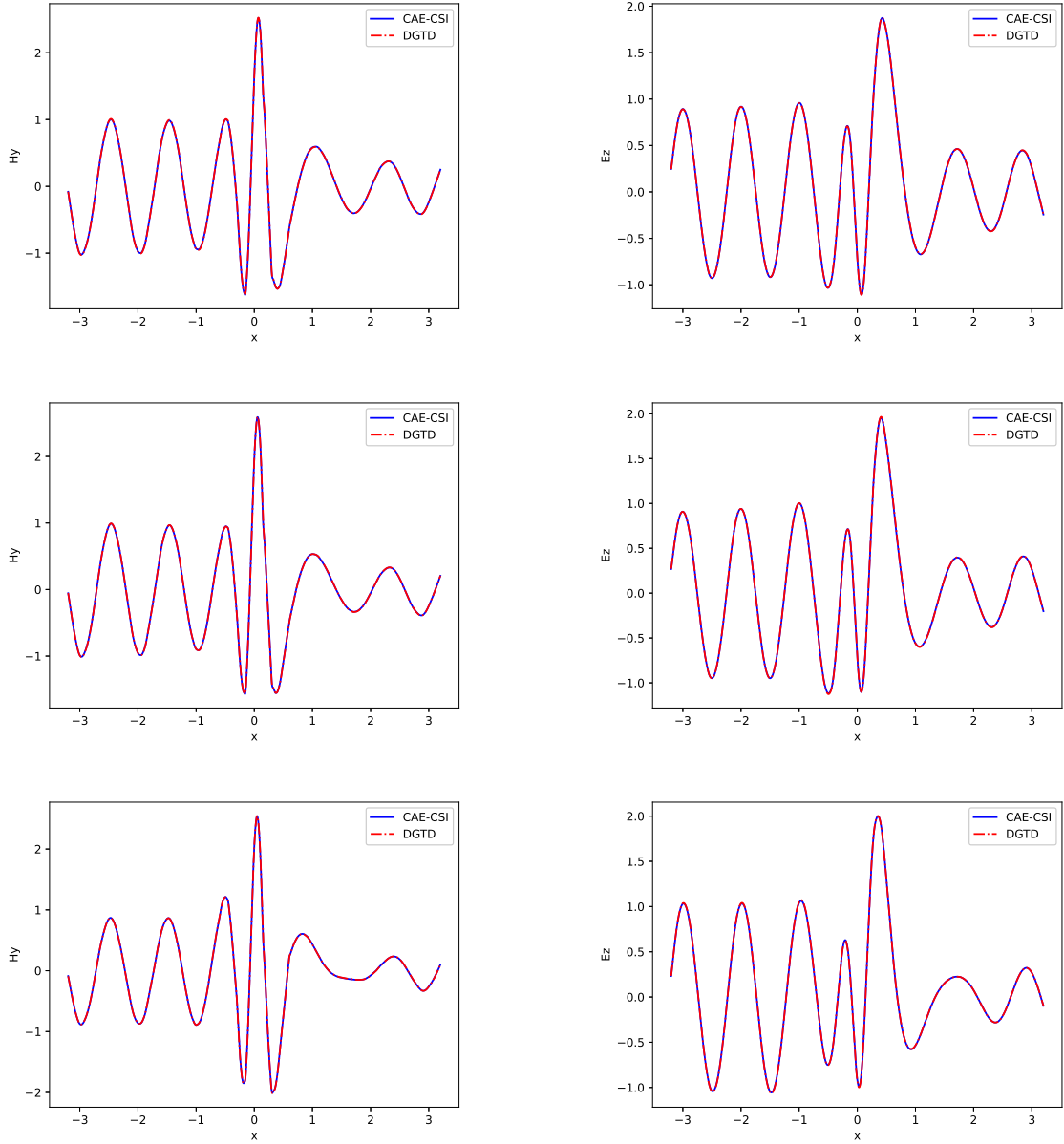


Figure 13: Scattering of plane wave by a multi-layer heterogeneous medium. Comparison of the 1-D x -wise distribution along $y = 0$ of the real part of H_y (left) and E_z (right) for the testing parameter instances: $\boldsymbol{\mu}_1 = \{(5.1, 3.4, 2.1, 1.4)\}$ (1st row), $\boldsymbol{\mu}_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ (2th row) and $\boldsymbol{\mu}_3 = \{(5.5, 3.7, 2.4, 1.7)\}$ (3th row).

Table 7: Scattering of plane wave by a multi-layer heterogeneous medium. Comparison between the CAE-CSI (offline and online) and DGTD methods in terms of CPU time. The unit of time is second.

Offline		Online			
Snapshots	Two-step POD	CAE-CSI	CAE-CSI	POD-CSI	DGTD
2.4686×10^4	6.5544	2.0432×10^3	0.2923	1.4856	3.0477×10^2

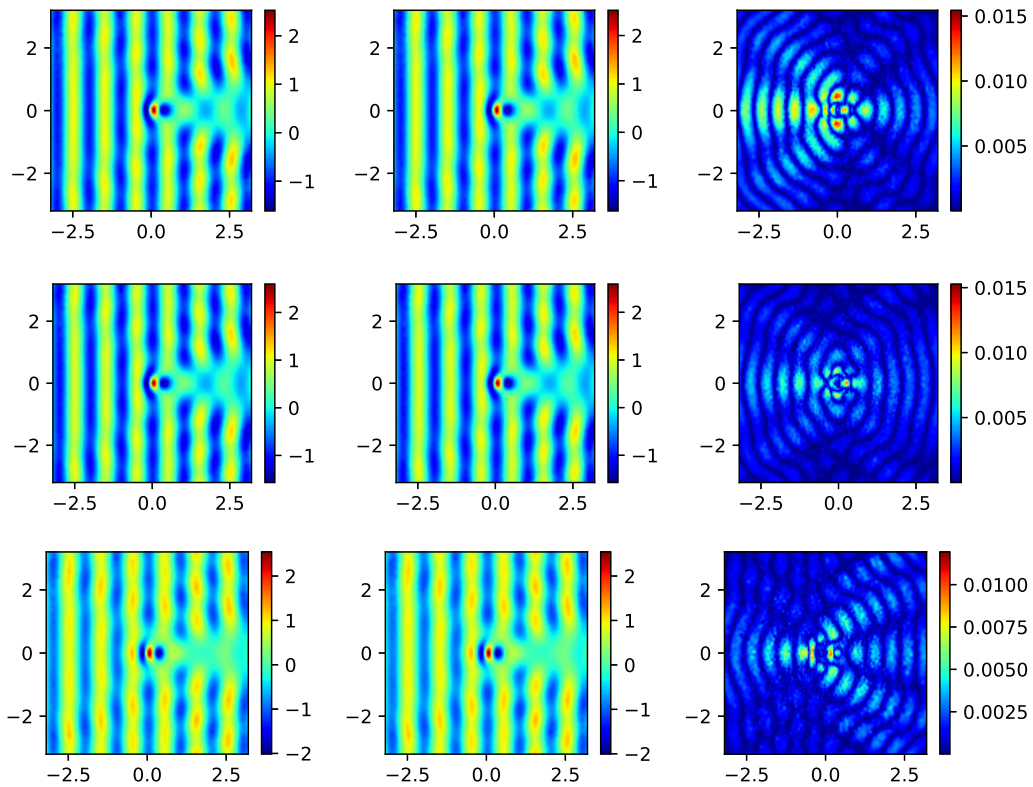


Figure 14: Scattering of plane wave by a multi-layer heterogeneous medium. Comparison of the 2-D distribution of the real part of H_y between DGTD (left), CAE-CSI (middle) and relative error (right) for the testing parameter instances: $\boldsymbol{\mu}_1 = \{(5.1, 3.4, 2.1, 1.4)\}$ (1st row), $\boldsymbol{\mu}_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ (2th row) and $\boldsymbol{\mu}_3 = \{(5.5, 3.7, 2.4, 1.7)\}$ (3th row).

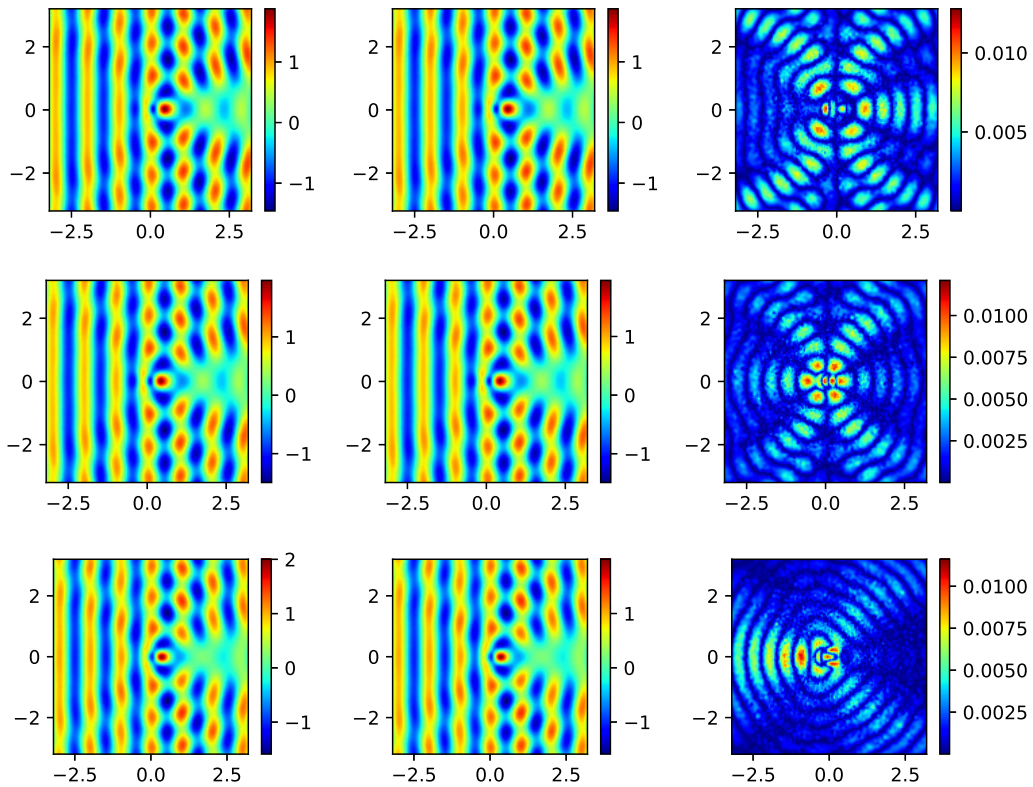


Figure 15: Scattering of plane wave by a multi-layer heterogeneous medium: comparison of the 2-D distribution of the real part of E_z between DGTD (left), CAE-CSI (middle) and relative error (right) for the testing parameter instances: $\boldsymbol{\mu}_1 = \{(5.1, 3.4, 2.1, 1.4)\}$ (1st row), $\boldsymbol{\mu}_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ (2th row) and $\boldsymbol{\mu}_3 = \{(5.5, 3.7, 2.4, 1.7)\}$ (3th row).

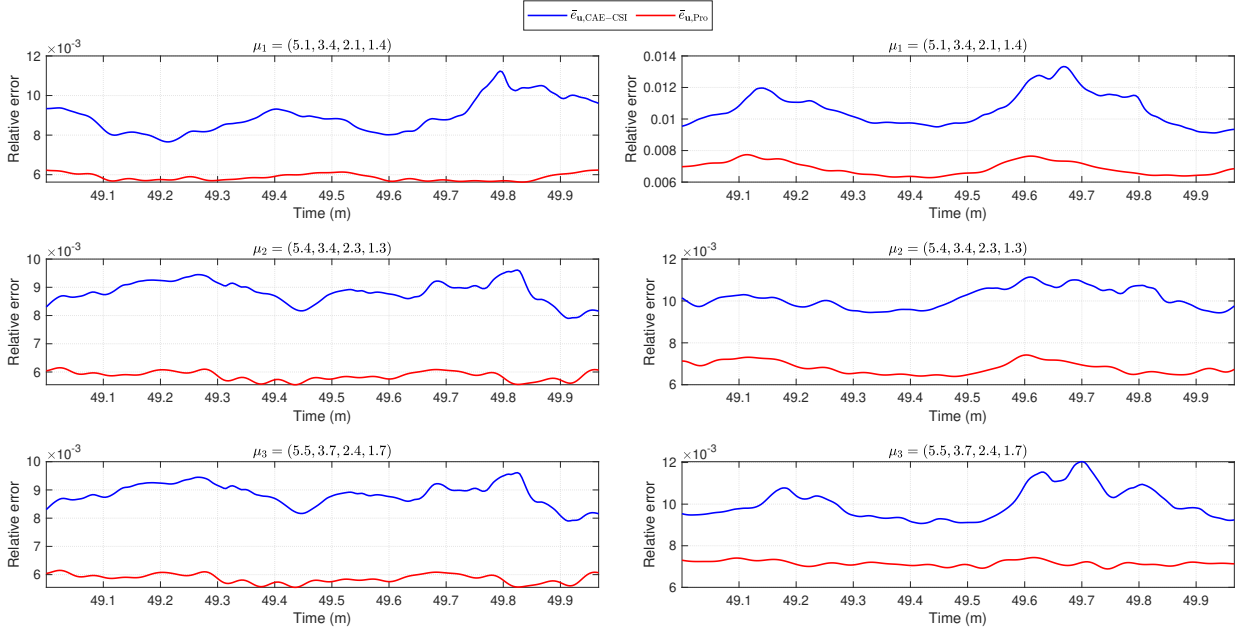


Figure 16: Scattering of plane wave by a multi-layer heterogeneous medium. Comparison of the relative projection error $e_{\mathbf{u},\text{Pro}}$, the CAE-CSI error $e_{\mathbf{u},\text{CAE-CSI}}$ for \mathbf{H} (left) and \mathbf{E} (right) for the testing parameter instances: $\boldsymbol{\mu}_1 = \{(5.1, 3.4, 2.1, 1.4)\}$ (1st row), $\boldsymbol{\mu}_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ (2th row) and $\boldsymbol{\mu}_3 = \{(5.5, 3.7, 2.4, 1.7)\}$ (3th row).

of the snapshots, then the CAE network provides a low dimensional representation of the projection coefficients through its encoder, as well as the inverse map through its decoder. CSI-based models are trained to approximate a mapping from the time/parameters to the low dimensional representations and the decoder is used to reconstruct the system solutions to their original dimension. By combining CSI with a decoder, a ‘simulation-free’ approach is established to obtain a RB solution at a very low cost. The results of the numerical experiments demonstrate the powerful dimensionality reduction and reconstruction capabilities of the CAE-CSI method, and a highly accurate and cheap surrogate model for systems described by PDEs is developed. Future research directions include more realistic 3-D simulations and the reduction of parameterized geometry.

Acknowledgments

The last author is supported by NSFC (Grant No. 12101511).

References

- [1] G Hossein Behforooz. A comparison of theE (3) and not-a-knot cubic splines. *Appl. Math. Comput.*, 72(2-3):219–223, 1995.
- [2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.
- [3] Peter Benner, Mario Ohlberger, Anthony Patera, Gianluigi Rozza, and Karsten Urban. *Model reduction of parametrized systems*. Springer, 2017.
- [4] T Bui-Thanh, Murali Damodaran, and Karen Willcox. Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics. In *Proceedings of the 21st AIAA Applied Aerodynamics Conference*, page 4213, 2003.

- [5] Wenqian Chen, Qian Wang, Jan S Hesthaven, and Chuhua Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *J. Comput. Phys.*, 446:110666, 2021.
- [6] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.
- [7] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- [8] Loula Fezoui, Stéphane Lanteri, Stéphanie Lohrengel, and Serge Piperno. Convergence and stability of a discontinuous Galerkin time-domain method for the 3D heterogeneous Maxwell equations on unstructured meshes. *ESAIM: Math. Model. Numer. Anal.*, 39(6):1149–1176, 2005.
- [9] Stefania Fresca and Andrea Manzoni. POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Comput. Methods Appl. Mech. Eng.*, 388:114181, 2022.
- [10] Stephen D Gedney. Introduction to the finite-difference time-domain (FDTD) method for electromagnetics. *Synthesis Lectures on Computational Electromagnetics*, 6(1):1–250, 2011.
- [11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
- [12] Francisco J Gonzalez and Maciej Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv preprint arXiv:1808.01346*, 2018.
- [13] Mengwu Guo and Jan S Hesthaven. Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Comput. Methods Appl. Mech. Eng.*, 341:807–826, 2018.
- [14] Mengwu Guo and Jan S Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Comput. Methods Appl. Mech. Eng.*, 345:75–99, 2019.
- [15] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490, 2016.
- [16] Md Sakib Hasan, Sherif Amer, Syed K Islam, and Garrett S Rose. Multivariate cubic spline: a versatile DC modeling technique suitable for different deep submicron transistors. In *Proceedings of the 2019 IEEE SoutheastCon Conference*, pages 1–8. IEEE, 2019.
- [17] Md Sakib Hasan, Syed K Islam, and Benjamin J Blalock. Modeling of SOI four-gate transistor (G4FET) using multidimensional spline interpolation method. *Microelectronics J.*, 76:33–42, 2018.
- [18] Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.
- [19] Jan S Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.*, 363:55–78, 2018.
- [20] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.
- [21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR, 2015.
- [22] Jian-Ming Jin. *Theory and computation of electromagnetic fields*. John Wiley & Sons, 2011.
- [23] Mariella Kast, Mengwu Guo, and Jan S Hesthaven. A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems. *Comput. Methods Appl. Mech. Eng.*, 364:112947, 2020.

- [24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.
- [26] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.*, 404:108973, 2020.
- [27] Jichun Li and Bing Nan. Simulating backward wave propagation in metamaterial with radial basis functions. *Results Appl. Math.*, 2:100009, 2019.
- [28] Kun Li, Ting-Zhu Huang, Liang Li, and Stéphane Lanteri. A reduced-order DG formulation based on POD method for the time-domain Maxwell’s equations in dispersive media. *J. Comput. Appl. Math.*, 336:249–266, 2018.
- [29] Kun Li, Ting-Zhu Huang, Liang Li, and Stéphane Lanteri. Pod-based model order reduction with an adaptive snapshot selection for a discontinuous galerkin approximation of the time-domain maxwell’s equations. *J. Comput. Phys.*, 396:106–128, 2019.
- [30] Kun Li, Ting-Zhu Huang, Liang Li, and Stéphane Lanteri. Non-intrusive reduced-order modeling of parameterized electromagnetic scattering problems using cubic spline interpolation. *J. Sci. Comput.*, 87(2):1–29, 2021.
- [31] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.
- [32] Luke Melas-Kyriazi. The mathematical foundations of manifold learning. *arXiv preprint arXiv:2011.01307*, 2020.
- [33] Peter Monk et al. *Finite element methods for Maxwell’s equations*. Oxford University Press, 2003.
- [34] Stefanos Nikolopoulos, Ioannis Kalogeris, and Vissarion Papadopoulos. Non-intrusive surrogate modeling for parametrized time-dependent partial differential equations using convolutional autoencoders. *Eng. Appl. Artif. Intell.*, 109:104652, 2022.
- [35] M Oulghelou and C Allery. Non intrusive method for parametric model order reduction using a bi-calibrated interpolation on the grassmann manifold. *J. Comput. Phys.*, 426:109924, 2021.
- [36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process Syst.*, 32, 2019.
- [37] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.
- [38] E Schmidt. On the theory of linear and nonlinear integral equations. i. development of arbitrary function according to systems prescribed. *Math. Ann.*, 63:433–476, 1907.
- [39] Allen Taflove and Susan Hagness. *Computational electrodynamics: the finite-difference time-domain method - 3rd ed*. Artech House Publishers, 2005.
- [40] Jonathan Viquerat. *Simulation of electromagnetic waves propagation in nano-optics with a high-order discontinuous Galerkin time-domain method*. PhD thesis, Université Nice Sophia Antipolis, 2015.
- [41] Qian Wang, Jan S Hesthaven, and Deep Ray. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *J. Comput. Phys.*, 384:289–307, 2019.
- [42] Ren-Hong Wang. *Multivariate spline functions and their applications*, volume 529. Springer Science & Business Media, 2013.

- [43] D Xiao, F Fang, CC Pain, and IM Navon. A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Comput. Methods Appl. Mech. Eng.*, 317:868–889, 2017.
- [44] Dunhui Xiao, Fangxin Fang, Andrew G Buchan, Christopher C Pain, Ionel M Navon, and Ann Mugeridge. Non-intrusive reduced order modelling of the Navier–Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 293:522–541, 2015.
- [45] Qiang Ye and Weifeng Zhi. Discrete Hessian eigenmaps method for dimensionality reduction. *J. Comput. Appl. Math.*, 278:197–212, 2015.
- [46] Kane Yee. Numerical solution of initial boundary value problems involving Maxwell’s equations in isotropic media. *IEEE Trans. Antennas Propag.*, 14(3):302–307, 1966.
- [47] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.
- [48] Ying Zhao, Liang Li, and Kun Li. Non-intrusive reduced order modeling of parametric electromagnetic scattering problems through Gaussian process regression. *arXiv preprint arXiv:2103.12472*, 2021.
- [49] Tong Zhou and Yongbo Peng. Kernel principal component analysis-based Gaussian process regression modelling for high-dimensional reliability analysis. *Comput. Struct.*, 241:106358, 2020.