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. 2023. hal-03927379

HAL Id: hal-03927379
https://hal.science/hal-03927379
Preprint submitted on 6 Jan 2023
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 [15, 14, 30, 11]. 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.  

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, 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), radial basis functions (RBF), and cubic spline interpolation (CSI) 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), a hybrid strategy is devised in 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 and Hessian feature maps have attracted much attention in the past several years. Assuming that data points lie in a low dimensional manifold embedded in a higher dimensional Euclidean space, manifold learning 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. 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. 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. As an extension of ordinary autoencoders, convolutional autoencoders (CAEs) 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 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 , 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 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 , 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 and 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. 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 we briefly introduce the time-domain Maxwell equations and the DGTD scheme to generate the snapshots. Section provides a description for the two-step POD, from which a RB basis is constructed. In section we present how a CAE is employed to further reduce the dimension of intrinsic coordinates. In section 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 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

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

(1)

where \(\Omega\) 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, \(\varepsilon_r\) and \(\mu_r\) 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]

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

(2)

where \(\partial \Omega\) is the boundary of \(\Omega\), \(\mathbf{n}\) denotes the outer unit normal vector along \(\partial \Omega\), \(\mathbf{E}^{\text{inc}}\) and \(\mathbf{H}^{\text{inc}}\) are the incident fields, and \(Z = \sqrt{\mu_r/\varepsilon_r}\). 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 \(0 = t_0 < t_1 < \cdots < t_m = T_f\) with \(t_n = n\Delta t\) \((n = 0, 1, \cdots, m)\), and \(\Delta t\) 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

\[
\begin{align}
\begin{cases}
M^{\varepsilon_r} \mathbf{E}_h(t_{n+1}) - \mathbf{E}_h(t_n) = (\mathbb{I} - S^r) \mathbf{H}_h(t_{n+\frac{1}{2}}) - S^h \mathbf{H}_h(t_{n+\frac{1}{2}}) - \mathbf{B}^{\varepsilon}(n\Delta t), \\
M^{\mu_r} \mathbf{H}_h(t_{n+\frac{1}{2}}) - \mathbf{H}_h(t_{n+\frac{1}{2}}) = (-\mathbb{I} + S^r) \mathbf{E}_h(t_{n+1}) + S^h \mathbf{E}_h(t_{n+1}) + \mathbf{B}^{\mu}((n + \frac{1}{2})\Delta t).
\end{cases}
\end{align}
\]

(3)

Here \(M^{\varepsilon_r}\) and \(M^{\mu_r}\) are the mass matrices, \(\mathbb{I}\) is the stiffness matrix, \(S^r\) is the surface matrix for the interior faces, and \(S^h\) and \(S^r\) are the boundary face matrices. For more detailed definition of these matrices we refer to [10] [29].

3 Two-step POD method

3.1 Snapshot-based data collection

For the parameterized time-dependent problem, let \(\mathcal{P}_h^\mu = \{\mu_1, \mu_2, \cdots, \mu_{N^\mu}\}\) 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 \(\mu \in \mathcal{P}_h^\mu\) in a time sampling \(\mathcal{T}_h = \{t_1, t_2, \cdots, t_m\}\). Then we uniformly select \(N_t\) transient solutions in \(\mathcal{T}_h^\mathbf{r} = \{t_{n_1}, t_{n_2}, \cdots, t_{n_{N_t}}\} \subseteq \mathcal{T}_h\) to form the time trajectory matrix \(\mathbf{S}_u \in \mathbb{R}^{N_t \times N_t}\), which takes the form

\[
\mathbf{S}_u = \begin{bmatrix}
\mathbf{u}_{h,1}(t_{n_1}, \mu_j) & \mathbf{u}_{h,1}(t_{n_2}, \mu_j) & \cdots & \mathbf{u}_{h,1}(t_{n_{N_t}}, \mu_j) \\
\mathbf{u}_{h,2}(t_{n_1}, \mu_j) & \mathbf{u}_{h,2}(t_{n_2}, \mu_j) & \cdots & \mathbf{u}_{h,2}(t_{n_{N_t}}, \mu_j) \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{u}_{h,N_t}(t_{n_1}, \mu_j) & \mathbf{u}_{h,N_t}(t_{n_2}, \mu_j) & \cdots & \mathbf{u}_{h,N_t}(t_{n_{N_t}}, \mu_j)
\end{bmatrix}
\]

(4)

and the snapshot matrix for all parameters in \(\mathcal{P}_h^\mathbf{r}\) is

\[
\mathbf{S}_u = \begin{bmatrix}
\mathbf{S}_u^1 | \mathbf{S}_u^2 | \cdots | \mathbf{S}_u^{N_\mu}\n\end{bmatrix} \in \mathbb{R}^{N_t \times N_\mu}, \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\},
\]

(5)
where $\mathcal{N}_s = \mathcal{N}_f \cdot \mathcal{N}_p$, $\mathbf{u}_h(t, \mathbf{\mu}) \in \mathbb{R}^{\mathcal{N}_h}$ 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 $(t, \mathbf{\mu})$ 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 $\mathcal{V}_{u,r}$ 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

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

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

$$\min_{\mathbf{V}_u \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}}} \|\mathbf{S}_u - \mathbf{V}_u \mathbf{V}_u^T \mathbf{S}_u\|_F,$$

s.t. $\mathbf{V}_u^T \mathbf{V}_u = \mathbf{I}$,

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)

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

where $\mathbf{W}_u = [\mathbf{w}_{u,1}, \mathbf{w}_{u,2}, \cdots, \mathbf{w}_{u,N}]$ and $\mathbf{Z}_u = [\mathbf{z}_{u,1}, \mathbf{z}_{u,2}, \cdots, \mathbf{z}_{u,N}]$ are $\mathcal{N}_h \times \mathcal{N}_h$ and $\mathcal{N}_s \times \mathcal{N}_s$ orthogonal matrices respectively, and $\sum_{\mathcal{N}_r = \mathcal{N}_h}^{\mathcal{N}_u} = \text{diag}(\sigma_{u,1}, \sigma_{u,2}, \cdots, \sigma_{u,\mathcal{N}_r})$. The singular values $\sigma_{u,1} \geq \sigma_{u,2} \geq \cdots \geq \sigma_{u,\mathcal{N}_r} \geq 0$ of $\mathbf{S}_u$ are sorted in descending order, and $\mathbf{u}_i$ is the rank of $\mathbf{S}_u$. The POD basis with $\mathcal{N} (\mathcal{N} \ll \mathcal{N}_r)$ 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

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

where $\mathbf{V}_u = [\mathbf{v}_{u,1}, \cdots, \mathbf{v}_{u,\mathcal{N}}], \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}$. 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 $\mathcal{N}_s = \mathcal{N}_f \cdot \mathcal{N}_p$ 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 6.

1. For each time trajectory matrix $\mathbf{S}_u^j, j = 1, 2, \cdots, \mathcal{N}_p$, the POD basis $\{\gamma_{u,i}^j\}_{i=1}^{\mathcal{N}_r}$ are obtained through SVD, then assemble them to a matrix $\mathbf{T}_u^j = [\gamma_{u,1}^j, \gamma_{u,2}^j, \cdots, \gamma_{u,\mathcal{N}_r}^j]$.

2. Assemble a composite matrix $\mathbf{T}_u = [\mathbf{T}_u^1 | \mathbf{T}_u^2 | \cdots | \mathbf{T}_u^{\mathcal{N}_p}]$ 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., $\mathbf{V}_u = [\mathbf{v}_{u,1}, \cdots, \mathbf{v}_{u,\mathcal{N}}]$. 


According to the projection theory \cite{28}, one can obtain the reduced-order solution and the projection coefficients or intrinsic coordinates as

\[
\begin{align*}
\mathbf{u}_h^u(t, \mu) = \mathbf{V}_u (\mathbf{V}_u^T \mathbf{V}_u)^{-1} \mathbf{V}_u^T \mathbf{u}_h(t, \mu), \\
\alpha_u(t, \mu) \equiv \mathbf{u}_\alpha(t, \mu) = (\mathbf{V}_u^T \mathbf{V}_u)^{-1} \mathbf{V}_u^T \mathbf{u}_h(t, \mu), \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\}.
\end{align*}
\]

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

\[
\mathbf{u}_h^u(t, \mu) = \mathbf{V}_u \alpha_u(t, \mu) = \sum_{i=1}^{N} \alpha_{u,i}(t, \mu) \mathbf{v}_{u,i}, \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\},
\]

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

In our previous work \cite{48, 30} we constructed the mapping between the time/parameter values and the corresponding orthogonal eigenvectors \( \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r \) with \( r \) being the rank of \( \mathbf{A} \);

\[ \mathbf{V}_u = \text{POD} \left( \mathbf{T}_u \right) \]

where \( \mathbf{r} \) and \( \mathbf{r}' \) are the corresponding singular values. The two-step POD projection error can be bounded as

\[
\begin{align*}
\sum_{j=1}^{N_u} \sum_{i=1}^{N} \| \mathbf{S}_{r}^j(i) - \mathbf{V}_u (\mathbf{T}_u^{(i)})^T \mathbf{S}_{r}^j(i) \|_{\mathbb{R}^{N_u}} & \leq \sum_{j=1}^{N_u} \sum_{i=1}^{N} \| \mathbf{S}_{r}^j(i) - \mathbf{T}_u^{(i)} (\mathbf{T}_u^{(i)})^T \mathbf{S}_{r}^j(i) \|_{\mathbb{R}^{N_u}} \\
& + \sum_{j=1}^{N_u} \sum_{i=1}^{N} \| \mathbf{V}_u (\mathbf{T}_u^{(i)})^T \mathbf{S}_{r}^j(i) - \mathbf{V}_u (\mathbf{T}_u^{(i)})^T \mathbf{S}_{r}^j(i) \|_{\mathbb{R}^{N_u}} \\
& + \sum_{j=1}^{N_u} \sum_{i=1}^{N} \| \mathbf{V}_u (\mathbf{T}_u^{(i)})^T \mathbf{S}_{r}^j(i) - \mathbf{V}_u (\mathbf{T}_u^{(i)})^T \mathbf{S}_{r}^j(i) \|_{\mathbb{R}^{N_u}} \\
& \leq \mathcal{L}_1 + \mathcal{L}_2, \quad \mathbf{u} \in \{\mathbf{E}, \mathbf{H}\},
\end{align*}
\]
where
\[
L_1 = (1 + \| V_u V_u^T \|_F) \sum_{j=1}^{N_t} (N_N)^{i/k+1} (\sigma_{u,i}^2)^{\frac{1}{2}}, \\
L_2 = \max_{1 \leq j \leq N_p} \sum_{i=1}^{N_t} \| S_j (:, i) \|_{\mathbb{R}^N} \max_{1 \leq i \leq k} \| T_j (:, i) \|_{\mathbb{R}^N} (kN_p)^{i/N+1} (\sigma_{u,i}^2)^{\frac{1}{2}}.
\]

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

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 \( \alpha_u(t, \mu) \equiv u_N(t, \mu) \in \mathbb{R}^N \) from \( N \) 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 \( S \) embedded in \( \mathbb{R}^m \), where \( m \) is large \cite{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 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 layer encoder network
\[
h = f_E (x; \theta_E),
\]
where \( \theta_E \) denotes the weights and the biases of the encoder network, \( x \in \mathbb{R}^N \) is the input state, \( h \in \mathbb{R}^n \) is the feature or low dimensional representation vector with \( n \ll N \). A decoder network is then used to reconstruct \( x \) by
\[
\hat{x} = f_D (h; \theta_D).
\]

The objective is how to train this autoencoder and find the parameters that minimize the mean squared error over all training examples
\[
\theta_E^*, \theta_D^* = \arg \min_{\theta_E, \theta_D} \| x - \hat{x} \|^2.
\]

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 [10, 47] for more details of the convolution operations.

For simplicity, we consider a 3-D tensor \( X \in \mathbb{R}^{c \times h \times w} \) 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 \( X \) and then use them to construct a feature map \( \mathcal{G} \). Specifically, the output at the \((l-1)\)th layer is \( \mathcal{X}^{l-1} \in \mathbb{R}^{c' \times h' \times w'} \), and the feature map at \( l \)th layer is denoted as a tensor \( \mathcal{G}^l \in \mathbb{R}^{c''_l \times h''_l \times w''_l} \) with \( c''_l = c' - 1 \) and element \( \mathcal{G}^l_{i,j,k} \) 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 \( \mathcal{W}^l \in \mathbb{R}^{c''_l \times c' \times k_l \times s \times h_l \times w_l} \) with element \( \mathcal{W}^l_{i,j,m,n} \) 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

\[
\mathcal{G}^l_{i,j,k} = \sigma_l \left( \sum_{r,m,n} \mathcal{G}^{l-1}_{i,j,k-1} \ast \mathcal{W}^l_{i,j,m,n} + \mathcal{B}^l_{i,j,k} \right),
\]

where \( \sigma_l \) is a non-linear 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 \( V_u \in \mathbb{R}^{N_u \times N} \) \((u \in \{E, H\})\) generated by the two-step POD method, we compute the projection coefficients or intrinsic coordinates \( \alpha_u(t, \mu) = V_u^T u(t, \mu) = V_u^T u(t, \mu) \) \((u \in \{E, H\})\) to perform a moderate dimensionality reduction, and the projection coefficient matrix \( C_u \) takes the form

\[
C_u = \begin{bmatrix} C_u^1 & C_u^2 & \cdots & C_u^{N_u} \end{bmatrix} \in \mathbb{R}^{N \times N_u}, \ u \in \{E, H\}, \tag{17}
\]

where \( C_u^j = V_u^T S_u^j \in \mathbb{R}^{N \times N_u} \), \( j = 1, 2, \cdots, N_u \). Next, we randomly shuffle \( C_u \) by column and split it into training set and validation set according to a training-validation splitting fraction \( \lambda \), i.e., \( C_u = [C_u^{(tr)}, C_u^{(val)}] \), \( u \in \{E, H\} \), where \( C_u^{(tr)} \in \mathbb{R}^{N \times \lambda N_u} \). 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 \( C_u^{(tr)} \) and \( C_u^{(val)} \) are normalized to \([0, 1]\) through the following affine transformation

\[
C_u^{(tr)}(i, j) = \frac{C_u^{(tr)}(i, j) - \min_{1 \leq i \leq N, 1 \leq j \leq \lambda N_u} C_u^{(tr)}(i, j)}{\max_{1 \leq i \leq N, 1 \leq j \leq \lambda N_u} C_u^{(tr)}(i, j) - \min_{1 \leq i \leq N, 1 \leq j \leq \lambda N_u} C_u^{(tr)}(i, j)}. \tag{18}
\]

We then reshape each column of \( C_u \) into a square matrix of dimension \((\sqrt{N}, \sqrt{N})\), where \( N = m^2 \) with \( m \in \mathbb{N} \). Then stack the components of \( H \) and \( E \) 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 \((\sqrt{N}, \sqrt{N}, d)\), 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 \( u_N(t, \mu) = V_u^T u(t, \mu) \) and its reconstruction \( \tilde{u}_N(t, \mu, \theta_E, \theta_D) \) is defined as follow

\[
\min_{\theta} \mathcal{J}(\theta) = \min_{\theta} \frac{1}{N_u} \sum_{j=1}^{N_u} \sum_{i=1}^{N_d} \| V_u^T u(t_i, \mu_j) - \tilde{u}_N(t_i, \mu_j, \theta_E, \theta_D) \|^2, \tag{19}
\]
and $\theta = (\theta_E, \theta_D)$. The CAE is implemented in PyTorch and the Adam optimizer is used in the training stage. We set the initial learning rate to $\eta_0 = 10^{-4}$, the mini-batch size to $N_{mb} = 50$, and maximum number of epochs to $N_{epo} = 5000$. A learning rate decay with $\eta = \eta_0/(1 + \alpha \cdot \text{epoch})$ is used to accelerate the training with hyperparameter $\alpha = 0.05$. 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.

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

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. The training process of the CAE is shown in Algorithm 2.

**Algorithm 2: Training process of the CAE**

**Input:** Snapshot matrix $S_u$ and POD basis matrix $V_u$ with $u \in \{H, E\}$, training-validation splitting fraction $\lambda$, initial learning rate $\eta_0$, learning rate decay parameter $\alpha$, mini-batch size $N_{mb}$ and maximum number of epochs $N_{epo}$.

**Output:** Optimal model weights $\theta^* = (\theta_E^*, \theta_D^*)$.

1. Compute intrinsic coordinate matrix $C_u^j = \nabla_u^j S_u$ for $j = 1, 2, \cdots, N_p, u \in \{H, E\}$;
2. Shuffle $C_u = [C_u^{1:2}, \cdots, C_u^{N_p}]$ randomly by column;
3. Split data into training and validation set $C_u = [C_u^{tr}, C_u^{val}]$ with $C_u^{tr} \in \mathbb{R}^{N_x \times \lambda N_r}$;
4. Obtain the optimal model parameters $[W^*, b^*] = \text{CAE}_\text{TRAINING}(C_u, N_{epo}, N_b, \eta_0)$;

**Function** $[W_{tr}, b_{tr}] = \text{CAE}_\text{TRAINING}(D, N_{epo}, N_b, \eta_0)$

6. $W, b \leftarrow \text{Initialize}(W, b)$ ▷ initialize weights and biases
7. $N_b \leftarrow N_b/N_{mb}$ ▷ number of batches
8. for epoch $\leftarrow 1$ to $N_{epo}$ do
9. $\eta \leftarrow \text{LearningRateDecay}(\eta_0, \alpha, \text{epoch})$ ▷ learning rate decay
10. for s $\leftarrow 1$ to $N_b$ do
11. $D_i \leftarrow D[(s-1) \cdot N_{mb} + 1 : s \cdot N_{mb}]$ ▷ the s-th mini-batch
12. $D_i \leftarrow \text{Reshape}(D_i, (N_{mb}, \sqrt{N_x}, \sqrt{N}, d))$ ▷ reshape to $d$ channels
13. $J \leftarrow \frac{1}{N_x} \sum_{j=1}^{N_x} \sum_{i=1}^{N_{mb}} \|V_u \nabla u_i(t_i, \mu_j) - \tilde{u}_i(t_i, \mu_j)\|^2$ ▷ compute loss
14. $\Delta W \leftarrow -\eta G_{\text{Adam}}(\nabla W J), \Delta b \leftarrow -\eta G_{\text{Adam}}(\nabla b J)$ ▷ Adam optimizer step
15. $W \leftarrow W + \Delta W, b \leftarrow b + \Delta b$ ▷ update weights and biases
16. end
17. $W_{tr}, b_{tr} \leftarrow W, b,$ ▷ save optimal weights
18. end

5 Approximation of the reduced-order matrices based on CSI

5.1 Cubic spline interpolation

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

$$
\begin{align*}
S(x) &\in C^2[a, b], \\
S(x_k) &= f(x_k) = y_k, k = 0, 1, 2, \cdots, n, \\
S(x) &\text{ is a cubic polynomial in } [x_k, x_{k+1}], k = 0, 1, 2, \cdots, n-1.
\end{align*}
$$

(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 $\tilde{u}_n(t, \mu, \theta_E)$ is extracted via the encoder from the projection coefficient $u_N(t, \mu)$. The decoder reconstructs an approximate projection coefficient $\tilde{u}_N(t, \mu, \theta_E, \theta_D)$ from the low dimensional code $\tilde{u}_n(t, \mu, \theta_E)$ by performing the inverse operations of the encoder.

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

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

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

$$S_k^{(i)}(x_k^{-}) = S_k^{(i)}(x_k^{+}), S_k^{(n)}(x_k^{-}) = S_k^{(n)}(x_k^{+}), k = 1, 2, \ldots, n-1.$$

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

$$S_k^{(i)}(x_0^{-}) = S_k^{(i)}(x_1^{+}), S_k^{(n)}(x_0^{-}) = S_k^{(n)}(x_1^{+}).$$

Combined with the conditions (22)-(24), the cubic polynomial $S_k(x)$ ($k = 0, 1, \ldots, n-1$) 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, 18]. 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 $\tilde{u}_n(t, \mu, \mu_j) = [\omega_1(t, \mu_j), \omega_2(t, \mu_j), \ldots, \omega_n(t, \mu_j)] (i = 1, 2, \ldots, N_t, j = 1, 2, \ldots, N_p)$. After the convolutional autoencoder network is trained, we get all the $N_s$ low dimensional coded representation vectors
\( \tilde{u}_n(t_i, \mu_j) \in \mathbb{R}^n \). Reshape them to \( n \) reduced-order matrices with the form

\[
\mathbf{P}_l = \begin{pmatrix}
    \omega_1(t_{n_1}, \mu_1) & \omega_1(t_{n_1}, \mu_2) & \cdots & \omega_1(t_{n_1}, \mu_{N_p}) \\
    \omega_2(t_{n_2}, \mu_1) & \omega_2(t_{n_2}, \mu_2) & \cdots & \omega_2(t_{n_2}, \mu_{N_p}) \\
    \vdots & \vdots & \ddots & \vdots \\
    \omega_l(t_{n_N}, \mu_1) & \omega_l(t_{n_N}, \mu_2) & \cdots & \omega_l(t_{n_N}, \mu_{N_p})
\end{pmatrix} \in \mathbb{R}^{N_t \times N_p}, l = 1, 2, \ldots, n. \tag{25}
\]

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 [13]

\[
\mathbf{P}_l \approx \tilde{\mathbf{P}}_l = \sum_{k=1}^{q_l} \sigma_k^l \psi_k^l (\phi_k^l)^T, 1 \leq l \leq n, \tag{26}
\]

where \( \psi_k^l \) and \( \phi_k^l \) are the \( k \)th discrete time- and parameter- modes for the \( l \)th reduced-order matrix respectively, \( \sigma_k^l \) is the \( k \)th singular value, and \( q_l \) is the truncation rank corresponding to the error tolerance \( \delta \), i.e., \( q_l = \text{argmin} \{ \pi(q_l) : \pi(q_l) \geq 1 - \delta \} \) with \( \pi(q_l) = \sum_{k=1}^{q_l} (\sigma_k^l)^2 / \sum_{k=1}^{\infty} (\sigma_k^l)^2 \) 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

\[
t \mapsto \tilde{\psi}_k^l(t), \text{ trained from } \{(t_i, (\psi_k^l))_i, i = 1, 2, \ldots, N_t\}, \tag{27}
\]

\[
\mu \mapsto \tilde{\phi}_k^l(\mu), \text{ trained from } \{(\mu_j, (\phi_k^l))_j, j = 1, 2, \ldots, N_p\}.
\]

Hence, we have

\[
(P_l)_{ij} \approx \sum_{k=1}^{q_l} \sigma_k^l \psi_k^l (t_i) \phi_k^l (\mu_j), \tag{28}
\]

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

\[
u_h(t^*, \mu^*) \approx u_h^*(t^*, \mu^*) = V \tilde{\alpha}_n(t^*, \mu^*), u \in \{E, H\}. \tag{29}
\]
Figure 3: Architecture of the CSI and decoder in the online stage. The CSI-based model produces a low-dimensional code \( \tilde{u}_{\mu}(t, \mu) \) for a new input \((t, \mu)\), then the decoder reconstructs an approximate projection coefficient \( \tilde{u}^N(t, \mu, \theta_D) \) from the code \( \tilde{u}_{\mu}(t, \mu) \).

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

\[
\begin{align*}
\nabla_r \partial H_x \partial t + \partial E_z \partial y &= 0, \\
\nabla_r \partial H_y \partial t - \partial E_z \partial x &= 0, \\
\varepsilon_r \partial E_z \partial t - \partial H_y \partial x + \partial H_x \partial y &= 0.
\end{align*}
\]

The excitation is an incident plane wave which is defined as

\[
\begin{align*}
H^\text{inc}_x(x, y, t) &= 0, \\
H^\text{inc}_y(x, y, t) &= -\cos(\omega t - kx), \\
E^\text{inc}_z(x, y, t) &= \cos(\omega t - kx),
\end{align*}
\]

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 \( N \) in the two-step POD method by the following error indicator

\[
\tilde{e}_{u,\text{POD}} = \frac{\sum_{(t, \mu) \in T^t \times P^\mu} \| u_h(t, \mu) - \mathbf{V}_u \mathbf{V}_u^T u_h(t, \mu) \|_{\mathbb{R}^{N_h}}} {\| u_h(t, \mu) \|_{\mathbb{R}^{N_h}}},
\]

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

\[
e_{u,\text{CAE-CSI}}(t, \mu) = \frac{\| u_h(t, \mu) - \tilde{u}^N_h(t, \mu) \|_{\mathbb{R}^{N_h}}} {\| u_h(t, \mu) \|_{\mathbb{R}^{N_h}}} = \frac{\| u_h(t, \mu) - \mathbf{V}_u \tilde{u}_h(t, \mu) \|_{\mathbb{R}^{N_h}}} {\| u_h(t, \mu) \|_{\mathbb{R}^{N_h}}}, \quad u \in \{ \mathbf{E}, \mathbf{H} \},
\]

which will be compared with the relative projection error

\[
e_{u,\text{P}}(t, \mu) = \frac{\| u_h(t, \mu) - \tilde{u}^N_h(t, \mu) \|_{\mathbb{R}^{N_h}}} {\| u_h(t, \mu) \|_{\mathbb{R}^{N_h}}} = \frac{\| u_h(t, \mu) - \mathbf{V}_u \tilde{u}_h(t, \mu) \|_{\mathbb{R}^{N_h}}} {\| u_h(t, \mu) \|_{\mathbb{R}^{N_h}}}, \quad u \in \{ \mathbf{E}, \mathbf{H} \}.
\]
The above errors are evaluated on a testing time/parameter sampling $T_h^{te} \times P_h^{te}$ of size $N_{te}$. Furthermore, the average relative errors are used to measure the accuracy of the ROM in our numerical experiments, which are defined as

$$\tilde{e}_{u,\text{CAE-CSI}} = \frac{\sum_{(t, \mu) \in T_h^{te} \times P_h^{te}} e_{u,\text{CAE-CSI}}(t, \mu)}{N_{te}}, \quad \tilde{e}_{u,\text{Pro}} = \frac{\sum_{(t, \mu) \in T_h^{te} \times P_h^{te}} e_{u,\text{Pro}}(t, \mu)}{N_{te}}. \quad (35)$$

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 × 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 $\mu_r = 1.0$, i.e., we consider a nonmagnetic material. The medium outside to the dielectric disk is assumed to be vacuum, i.e., $\varepsilon_r = \mu_r = 1.0$. 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 $P_2$ approximation, resulting in $N_e = 30264$ DOFs for the FOM.

During the offline stage, the DGTD solver is used to generate a collection of high-fidelity solutions at $N_p = 81$ equidistant parameter sampling points (i.e., $\mu \in P_h^{tr} = \{1.0, 1.05, \cdots, 4.95, 5.0\}$) with $N_t = 263$ points in the last oscillation period (i.e., $t \in T_h^{tr} = [49.0024, 49.006, \cdots, 49.9623, 49.966]$). A test parameter set $P_h^{te} = \{1.215, 2.215, 3.215, 4.215\}$ and test time set $T_h^{te} = T_h^{tr}$ 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 $\tilde{e}_{u,\text{POD}}$ and $\tilde{e}_{u,\text{POD}}$ with the choice of $k$ and $N$ in the two-step POD method. We obtain the POD basis matrices $V_u (u \in \{E, H\})$ with $k = 4$ and $N = 196$. Then we train the CAE network, and Fig. 5 shows the trajectory 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 $\tilde{u}(t, \mu)$ 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>
<thead>
<tr>
<th>Layer</th>
<th>Input Dim</th>
<th>Output dim</th>
<th>Kernel size</th>
<th>Number of kernels</th>
<th>Stride</th>
<th>Padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[14, 14, 3]</td>
<td>[14, 14, 8]</td>
<td>[5, 5]</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>[14, 14, 8]</td>
<td>[8, 8, 16]</td>
<td>[5, 5]</td>
<td>16</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[8, 8, 16]</td>
<td>[4, 4, 32]</td>
<td>[5, 5]</td>
<td>32</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>[4, 4, 32]</td>
<td>[2, 2, 64]</td>
<td>[5, 5]</td>
<td>64</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

order solutions are compared with the corresponding DGTD high-fidelity solutions on the test parameter set $P_h^{te} = \{\mu_1, \mu_2, \mu_3, \mu_4\}$ where $\mu_1 = \varepsilon_1 = 1.215, \mu_2 = \varepsilon_2 = 2.215, \mu_3 = \varepsilon_3 = 3.215$ and $\mu_4 = \varepsilon_4 = 4.215$. 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 $e_{u,\text{Pro}}$ and CAE-CSI error $e_{u,\text{CAE-CSI}}(u \in \{E, H\})$ for the test parameter instances $\mu_1, \mu_2, \mu_3$ and $\mu_4$ are shown in Fig. 10. The average relative error $\tilde{e}_{u,\text{Pro}}$ and
Table 2: Architecture of fully-connected and transposed convolutional layers in the decoder $f_D$.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input Dim</th>
<th>Output dim</th>
<th>Kernel size</th>
<th>Number of kernels</th>
<th>Stride</th>
<th>Padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n$</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[2, 2, 64]</td>
<td>[4, 4, 64]</td>
<td>[5, 5]</td>
<td>64</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>[4, 4, 64]</td>
<td>[8, 8, 32]</td>
<td>[5, 5]</td>
<td>32</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>[8, 8, 32]</td>
<td>[14, 14, 16]</td>
<td>[5, 5]</td>
<td>16</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>[14, 14, 16]</td>
<td>[14, 14, 3]</td>
<td>[5, 5]</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 4: Scattering of a plane wave by a dielectric disk. Convergence histories of $\varepsilon_{H,POD}$ (left) and $\varepsilon_{E,POD}$ (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.
\( e_{\text{u,CAE-CSI}}(u \in \{E, H\}) \) 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>
<thead>
<tr>
<th>Table 3: Scattering of a plane wave by a dielectric disk. Average relative error on the test set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_{\text{H,Pro}} )</td>
</tr>
<tr>
<td>1.15%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
</tr>
<tr>
<td>Snapshots</td>
</tr>
<tr>
<td>2.088 ( \times ) 10^4</td>
</tr>
</tbody>
</table>

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), $\varepsilon_2 = 2.215$ (2nd row), $\varepsilon_3 = 3.215$ (3rd 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.

<table>
<thead>
<tr>
<th>Layer $i$</th>
<th>$\mathcal{P}_i$</th>
<th>$\nu_{r,i}$</th>
<th>$r_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\varepsilon_{r,1} \in [5.0, 5.6]$</td>
<td>1.0</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>$\varepsilon_{r,2} \in [3.25, 3.75]$</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>$\varepsilon_{r,3} \in [2.0, 2.5]$</td>
<td>1.0</td>
<td>0.45</td>
</tr>
<tr>
<td>4</td>
<td>$\varepsilon_{r,4} \in [1.25, 1.75]$</td>
<td>1.0</td>
<td>0.6</td>
</tr>
</tbody>
</table>
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), $\varepsilon_2 = 2.215$ (2nd row), $\varepsilon_3 = 3.215$ (3rd 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 $N_h = 37236$ 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 $\mu \in \mathcal{P}$, 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 $N_{\Delta p} = 3$ uniform points for each parameter to form a training parameter samples $\mathcal{P}_h^\varepsilon$, resulting in $N_p = 81$ points. Each choice of the parameter is sampled for $N_t = 253$ snapshots in time at the last period, i.e., $T_{tr} = \{49.0002, 49.0041, \ldots, 49.9630, 49.9669\}$. Fig. 12 shows the convergence histories of $\bar{e}_{H, \text{POD}}$ and $\bar{e}_{E, \text{POD}}$ with different truncation parameter $k$ and size parameter $N$ in the two-step POD method. The POD basis matrices $V_\mu (\mu \in \{\mathbf{E}, \mathbf{H}\})$ are generated by the two-step POD method with $k = 4$ and $N = 196$. 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 $P^r = \{\mu_1, \mu_2, \mu_3\}$ with $\mu_1 = \{(5.1, 3.4, 2.1, 1.4)\}$, $\mu_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ and $\mu_3 = \{(5.5, 3.7, 2.4, 1.7)\}$. 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 $e_{u, \text{Pro}}$ and CAE-CSI error $e_{u, \text{CAE-CSI}} (\mu \in \{\mathbf{E}, \mathbf{H}\})$ for 3 test parameters $\mu_1$, $\mu_2$, and $\mu_3$ are displayed in Fig. 16. The average projection error $\bar{e}_{u, \text{Pro}}$ and the average CAE-CSI error $\bar{e}_{u, \text{CAE-CSI}} (\mu \in \{\mathbf{E}, \mathbf{H}\})$ for the 3 test parameters are listed in Tab. 6. Performance results of the CAE-CSI and DGTD methods with $P_2$ approximation are summarized in Tab. 7. The CPU time of the DGTD method is $3.0477 \times 10^2$ 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.

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

<table>
<thead>
<tr>
<th>$\bar{e}_{H, \text{Pro}}$</th>
<th>$\bar{e}_{H, \text{CAE-CSI}}$</th>
<th>$\bar{e}_{E, \text{Pro}}$</th>
<th>$\bar{e}_{E, \text{CAE-CSI}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60%</td>
<td>0.89%</td>
<td>0.69%</td>
<td>1.03%</td>
</tr>
</tbody>
</table>

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

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: $\mu_1 = \{(5.1,3.4,2.1,1.4)\}$ (1st row), $\mu_2 = \{(5.4,3.4,2.3,1.3)\}$ (2th row) and $\mu_3 = \{(5.5,3.7,2.4,1.7)\}$ (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.

<table>
<thead>
<tr>
<th></th>
<th>Offline</th>
<th>Online</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapshots</td>
<td>2.4686 $\times 10^4$</td>
<td>6.5544</td>
</tr>
<tr>
<td>Two-step POD</td>
<td>2.0432 $\times 10^3$</td>
<td>0.2923</td>
</tr>
<tr>
<td>CAE-CSI</td>
<td>1.4856</td>
<td>3.0477 $\times 10^2$</td>
</tr>
<tr>
<td>CAE-CSI POD-CSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGTD</td>
<td></td>
<td>0.2923</td>
</tr>
</tbody>
</table>
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: $\mu_1 = \{(5.1, 3.4, 2.1, 1.4)\}$ (1st row), $\mu_2 = \{(5.4, 3.4, 2.3, 1.3)\}$ (2th row) and $\mu_3 = \{(5.5, 3.7, 2.4, 1.7)\}$ (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: $\mu_1 = \{(5.1,3.4,2.1,1.4)\}$ (1st row), $\mu_2 = \{(5.4,3.4,2.3,1.3)\}$ (2th row) and $\mu_3 = \{(5.5,3.7,2.4,1.7)\}$ (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


