Received: 14 August 2020 | Revised: 7 February 2021 | Accepted: 24 March 2021

DOI: 10.1002/nme.6681

RESEARCH ARTICLE

WILEY

An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion

Toby R. F. Phillips[1] | **Claire E. Heaney**[1] | **Paul N. Smith**[2] | **Christopher C. Pain**[1]

[1]Applied Modelling and Computation Group, Department of Earth Science and Engineering, Imperial College London, London, UK

[2]Jacobs, Poundbury, UK

Correspondence
Claire E. Heaney, Applied Modelling and Computation Group, Imperial College London, London, UK.
Email: c.heaney@imperial.ac.uk

Funding information
EPSRC, Grant/Award Numbers: MUFFINS (EP/P033180/1), MAGIC (EP/N010221/1), INHALE (EP/T003189/1), PREMIERE (EP/T000414/1)

Abstract

Using an autoencoder for dimensionality reduction, this article presents a novel projection-based reduced-order model for eigenvalue problems. Reduced-order modeling relies on finding suitable basis functions which define a low-dimensional space in which a high-dimensional system is approximated. Proper orthogonal decomposition (POD) and singular value decomposition (SVD) are often used for this purpose and yield an optimal linear subspace. Autoencoders provide a nonlinear alternative to POD/SVD, that may capture, more efficiently, features or patterns in the high-fidelity model results. Reduced-order models based on an autoencoder and a novel hybrid SVD-autoencoder are developed. These methods are compared with the standard POD-Galerkin approach and are applied to two test cases taken from the field of nuclear reactor physics.

KEYWORDS

autoencoder, machine learning, reduced-order modeling, model reduction, neutron diffusion equation, reactor physics

1 | INTRODUCTION

Reduced-order modeling or model reduction[1] is a numerical technique that has made a significant impact across a broad range of fields, including aerodynamics,[2] hemodynamics,[3] fracture,[4,5] porous media,[6,7] and molecular dynamics.[8] Its aim is to produce a low-dimensional model which is a good approximation of a high-dimensional model (high-fidelity model), but which can be solved in a fraction of the computational time required by the high-fidelity model (HFM). Main applications for reduced-order modeling are multiquery problems, such as optimization and control,[2,9] and applications which require real-time solutions.[10,11] The computational efficiency of reduced-order models is achieved by an offline/online split of the computational work. There are typically three parts to the offline stage. First, the high-fidelity model is solved for different parameter values. Second, dimensionality reduction techniques are applied to the HFM solutions (referred to as snapshots) producing basis functions that span the low-dimensional space or reduced space. Finally, the discretized governing equations are approximated in the reduced space. In the online stage, the reduced-order model is solved for an unseen parameter value in a relatively fast time.

A number of dimensionality reduction methods have been proposed for finding the basis of the reduced space, however proper orthogonal decomposition[12,13] (POD) is by far the most common. Also known as principal component analysis

(PCA), the Karhunen–Loève transform, empirical orthogonal functions, and the Hotelling transform, POD was introduced by Lumley[14] as a method to extract coherent structures from turbulent flow data, and was made tractable with the method of snapshots by Sirovich.[15] The POD basis functions are found by taking a singular value decomposition (SVD) of the snapshots, or by solving a related eigenvalue problem. Using the basis functions, a Galerkin projection can be applied to the discretized governing equations (i.e., the high-fidelity model) to produce the reduced system of equations. For HFMs whose operators can be affinely decomposed, the Galerkin projection will be applied to all the matrices in the decomposition. Finding the reduced system for an unseen parameter can be done by interpolating the resulting reduced system matrices. Reduced-order models produced by this method are referred to as projection-based reduced-order models.[16] For non-affine or nonlinear systems, this process is more complex, and generally more sampling is required to represent the non-affine nature or nonlinearities of the system. Hyperreduction[17,18] and discrete empirical interpolation[19] are two methods that can be used for this.

In some circumstances, the projection step is challenging, as the matrices representing the high-fidelity model matrices are not easily available, for example, when using licensed software or for complex multiphysics problems. To alleviate this difficulty, an alternative method has been developed and is referred to variously as "non-intrusive reduced-order modeling" (NIROM), "POD with interpolation," "Galerkin-free," "surrogate POD" (see Reference 20 and references within), "data-driven reduced-order modeling"[21-23] and "system/model identification."[24,25] As in projection-based methods, these non-intrusive methods take a set of high-fidelity model solutions (snapshots) and find a reduced basis, often by using POD. However, to find solutions for unseen parameter values, projection of the high-fidelity system onto the basis functions is replaced by interpolating between snapshots (or snapshots that have themselves been projected onto the basis functions). Although cubic interpolation[26] or radial basis functions[7] were originally used for the interpolation, NIROM methods have since embraced machine learning, using a variety of neural networks for this purpose, including autoencoders in combination with long short-term memory networks,[27,28] multilayer perceptrons,[29] and Gaussian process regression.[22,30] When using neural networks, the offline stage involves training a neural network to learn how the solutions depend on various model parameters, and the online stage involves evaluating the neural network for an unseen parameter value or values.

Although machine learning techniques are becoming more prevalent for predicting the behavior of the HFM (third part of offline stage and online stage in NIROM-type methods), it is much less common affto find machine learning algorithms being used in the dimensionality reduction stage (second part of offline stage in both projection-based or NIROM methods). One type of neural network that is ideal for dimensionality reduction is the autoencoder. A type of unsupervised (or self-supervised) feed-forward neural network, the autoencoder learns the identity map with a network architecture that includes a bottleneck. The central bottleneck layer forces the autoencoder to learn a reduced or low-dimensional representation of the data. The neurons in this layer determine the so-called latent space and are referred to as latent variables. These types of networks are common in image classification and identification,[31,32] and have been used with great success to fill in gaps in images[33] and to remove noise from data.[34] Several authors[35-37] have highlighted the connection between autoencoders and SVD (or PCA), namely that an autoencoder with one hidden layer and *linear* activation functions can produce basis functions than span the same space as basis functions derived from an SVD. This connection between the autoencoder and the SVD paves the way for an autoencoder to be seen as a tool to perform *nonlinear* dimensionality reduction (by choosing nonlinear activation functions) and therefore to be used as an alternative to the SVD/PCA.[31,38] Among those early to recognize the potential of the autoencoder for dimensionality reduction within a model reduction framework were Milano et al.,[39] who reconstructed the near wall field from computational solutions for pressure and shear stress at the wall. The suitability of autoencoders for dimensionality reduction having been established, a number of authors have since explored their use in reduced-order models, mostly non-intrusive reduced-order models. This is because the strength of the autoencoder is that it is a nonlinear embedding, but this also means that the reduced-order system will be nonlinear and is therefore more complex to solve than for a POD-based ROM. For non-intrusive models, this nonlinearity does not present any additional challenge and no change in the algorithm is required, hence the greater uptake of autoencoders in NIROM-type methods. For example, Gonzalez et al.[27] and Wiewel et al.[28] both use a convolutional autoencoder to reduce the dimension of their problems and long short-term memory networks to learn the dynamics. The former demonstrate their method on interacting vortices in a 2D box and 2D lid-driven cavity flow. The interacting-vortices problem aims to demonstrate the location invariance properties of convolutional networks. The long-term statistics of the flow, as embodied by the turbulent kinetic energy, are captured much better by the autoencoder-based method than by the projection-based POD-Galerkin model. Coming from the field of computer graphics, Wiewel et al.[28] solve 3D problems with millions of spatial degrees of freedom. With their reduced-order model, they are able to produce realistic-looking simulations of complex dynamical systems such as

sloshing waves, colliding bodies of fluid, and smoke convection. Their reduced model runs approximately two orders of magnitude faster than their high fidelity solver.

As previously mentioned, the substitution of an autoencoder for an SVD is less straightforward for projection-based reduced-order models than it is for non-intrusive reduced-order models. There are a couple of simple examples of the incorporation of autoencoders in a projection-based reduced-order model,[40,41] however, the most comprehensive to date is a paper by Lee et al.[42] They propose a framework for projecting dynamical systems onto nonlinear manifolds using deep convolutional autoencoders and derive a posteriori error bounds. Focusing on advection-dominated benchmark problems known to be challenging for POD-based ROMs due to a slow decay of singular values or Kolmogorov n-width,[43] they model the convection, diffusion and chemical reactions of a flame as it evolves in time over a range of values for two parameters which occur in the nonlinear reaction source term. The error in the autoencoder-based method is over 40 times less than that of the standard POD-based projection method, showing that, although their method requires a higher offline computational time (for the training of the autoencoder), they can produce much more accurate results than projection-based methods for some nonlinear problems. In this article, we develop autoencoder-based ROMs for eigenvalue problems, using a standard autoencoder and a hybrid SVD-autoencoder.

The field of nuclear engineering has some extremely demanding applications such as neutron transport within a reactor core, the geometry of which presents extremely challenging aspect ratios. A recent discretization of a Westinghouse PWR-900 core with 44 energy groups used over 1.7 trillion degrees of freedom[44,45] precluding the use of optimization, safety-analysis or real-time analysis. It comes as no surprise, therefore, that reduced-order modeling has been applied to reactor physics problems. Based on the multigroup neutron diffusion equation for criticality, Chunyu et al.[46] applied a reduced basis method to 2D and 3D pressurized water reactor (PWR) benchmark problems. Buchan et al.[47] also tackled criticality by transforming the eigenvalue problem into a time-dependent problem and applied this to simple 1D and 2D models of nuclear reactors (test cases based on these are used in this article). These papers[46,47] used a single POD basis for all the energy groups, the so-called monolithic approach. Heaney et al.[48] developed a method to model control-rod movement with a POD-based ROM for a PWR fuel assembly, in which POD basis functions were calculated independently for each energy group. German et al.[49] demonstrated that such an approach produced more accurate results than the monolithic approach. All the methods discussed in this paragraph show computational gains of at least three orders of magnitude over the respective HFMs. Outside of criticality problems, ROM has also been applied to time-dependent movement of control rods,[50] and coupled neutronics and heat transfer,[51] both using a reduced basis method based on POD.

Although, in many areas, adding diffusion or solving diffusion problems often leads to fields that are smooth and well able to be represented accurately with POD basis functions, in other areas, such as reactor physics, one is often confronted by problems with abruptly changing fields similar to advection problems. A good example is a control rod that is partially inserted into a reactor, where one would see a near zero flux within the control rod and a much higher flux a small distance away from the control rod; see the scalar flux solutions shown in Buchan et al.[52] for instance. This similarity between advection and diffusion problems is also seen in discretization methods such as the self-adjoint angular flux method,[53] in which the first-order transport equations are transformed into a series of diffusion-like equations with the application of a least squares principle and a particular weighting. Although two relatively simple test cases are presented in this article, our future work will involve increasingly complicated/more realistic assembly or reactor configurations which will benefit from the improved modeling capabilities that autoencoders provide.

This article reports, what the authors believe to be, the first application of an autoencoder-based reduced-order model to an eigenvalue problem. Also presented is a novel hybrid SVD-autoencoder method that combines an SVD with an autoencoder. This is useful as training a fully connected autoencoder with a long input vector is computationally expensive. Without the SVD, the length of the input vector would be equal to the number of degrees of freedom of the problem. Applying the SVD reduces the length of the input vector to, at most, the number of snapshots.

The remainder of the article is organized as follows. In Section 2, the governing equations and their control-volume discretization are presented. Used to find the dominant eigenvalues and eigenvectors of eigenvalue problems, the power method is also described. Section 3 contains background information relevant to reduced-order modeling techniques, including descriptions of POD, SVD, selecting basis functions and constructing projection-based reduced-order models. A description of autoencoders is included in Section 4 along with theory on how to include these networks in a projection-based ROM. In Section 5 results of the 1D and 2D test cases are given for several different autoencoder-based reduced-order models and they are compared with a standard POD-based reduced-order model. Concluding remarks are made in the final section.

2 | GOVERNING EQUATIONS AND THEIR DISCRETIZATION

This section introduces the equations that govern criticality and their control-volume discretization, referred to as the high-fidelity model. A description is then given of the power method, which can be used to solve generalized eigenvalue problems such as that arising in criticality.

2.1 | Diffusion equation

The one-group steady-state diffusion equation for criticality takes the form of a generalized eigenvalue problem and can be written as

$$-\nabla \cdot (D\nabla\phi) + \Sigma^a\phi = \lambda\nu\Sigma^f\phi, \tag{1}$$

where ϕ is the scalar flux of the neutron population, Σ^a represents the absorption cross-section, Σ^f represents the fission cross-section and ν is the average number of neutrons produced per fission event. The diffusion coefficient, D, is given by:

$$D = \frac{1}{3(\Sigma^a + \Sigma^s)}, \tag{2}$$

where Σ^s is the scattering coefficient. The eigenvalue, λ, is defined as the reciprocal of k_{eff}, that is $\lambda = \frac{1}{k_{\text{eff}}}$, where:

$$k_{\text{eff}} = \frac{\text{number of neutrons in one generation}}{\text{number of neutrons in the preceding generation}}. \tag{3}$$

The boundary conditions are for reflection:

$$D\frac{\partial\phi}{\partial n} = 0, \tag{4}$$

and for a vacuum or bare-surface:

$$-D\frac{\partial\phi}{\partial n} = \frac{1}{2}\phi, \tag{5}$$

in which n is the outward-pointing normal to the boundary. In this article, autoencoder-based reduced-order models are developed for the one-group equations, however the methods described here could easily be applied to the multigroup equations.

2.2 | Discretization

A control-volume discretization of the diffusion equation in 2D with a regular $N_x \times N_y$ mesh can be written as:

$$\begin{aligned}
&\frac{1}{2}\max\left\{D_{i-1,j} + D_{i,j}, 0\right\}\frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x^2} + \frac{1}{2}\max\left\{D_{i,j} + D_{i+1,j}, 0\right\}\frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x^2} \\
&+ \frac{1}{2}\max\left\{D_{i,j-1} + D_{i,j}, 0\right\}\frac{\phi_{i,j} - \phi_{i,j-1}}{\Delta y^2} + \frac{1}{2}\max\left\{D_{i,j} + D_{i,j+1}, 0\right\}\frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y^2} + {\Sigma^a}_{i,j}\phi_{i,j} \\
&= \lambda\nu{\Sigma^f_{i,j}}\phi_{i,j}, \quad \forall i \in \{2, 3, \ldots, N_x - 1\}, \ \forall j \in \{2, 3, .., N_y - 1\},
\end{aligned} \tag{6}$$

in which Δx and Δy are the uniform cell widths in the x- and y-directions, respectively, N_x and N_y are the numbers of cells in the x- and y-directions, respectively, the subscripts i, j refer to the cells in the x- and y-directions, respectively, and $\phi_{i,j}$ represents the scalar flux in cell i, j. In this expression, the first and last cells are omitted in order to apply the

boundary conditions efficiently. To apply a reflective boundary condition, see Equation (4), the diffusion coefficients in the outermost cells are set to a large negative number. To set a bare surface boundary condition, see Equation (5), the diffusion coefficients in the outermost cells are again set to a large negative number and the absorption term is modified as now described. To apply such a boundary condition to the left or right edges (where the normal to the boundary is aligned with the x-direction):

$$\Sigma^a_{i,j} \leftarrow \Sigma^a_{i,j} + \frac{1}{2\Delta x}, \tag{7}$$

and to apply this to the top or bottom edges (where the normal to the boundary is aligned with the y-direction):

$$\Sigma^a_{i,j} \leftarrow \Sigma^a_{i,j} + \frac{1}{2\Delta y}. \tag{8}$$

For cells that have both boundary conditions the following modification is made:

$$\Sigma^a_{i,j} \leftarrow \Sigma^a_{i,j} + \frac{1}{2\Delta x} + \frac{1}{2\Delta y}. \tag{9}$$

The discretized form of Equation (1) can therefore be written as:

$$\boldsymbol{A\phi} = \lambda \boldsymbol{B\phi}. \tag{10}$$

where matrix $\boldsymbol{A}$ contains the transport terms (scattering, diffusion) from the left-hand side of Equation (6), matrix $\boldsymbol{B}$ represents the fission terms given in the right-hand side of Equation (6) and the vector $\boldsymbol{\phi}$ contains the values of the scalar flux for each cell. The matrices are of size $(N_x - 2)(N_y - 2)$ by $(N_x - 2)(N_y - 2)$. Although a 2D discretization is given here, the methods can also be applied in 1D or 3D.

2.3 | The power method

The power method[54] is an algorithm which finds the dominant eigenvalue and eigenvector of an eigenvalue problem. In certain circumstances the method can be slow to converge, however, it is possible to accelerate convergence.[55] The power method has a broad range of application and has been used to rank scientific journals[56] and to make suggestions to Twitter users of whom to follow[57] as well as being applied to the criticality eigenvalue problem.[58] Described in Algorithm 1, the power method first finds an approximation to the scalar flux solution within the so-called inner iteration loop. Approximations of the flux and eigenvalue (from the outer iterations) are used to form the right-hand side of Equation (10) and the resulting system is solved with the forward backward Gauss Seidel method (line 3). The scalar flux solution is passed to the outer iteration loop where it is normalized, and the associated eigenvalue is then calculated. The outer iterations continue until convergence is reached, which, in this article, is either when the difference in the eigenvalue in successive iterations is less than 10^{-8} or when 1000 iterations have been completed. The outer iterations are detailed in Algorithm 2, in which $\boldsymbol{b}$ is a vector whose entries all equal one. The normalization applied here, seen on line 9, differs from the normalization usually applied in the power method, however, results in a unit number of fissions, which is often preferred in nuclear applications. The parenthesized superscript on the flux solutions and the eigenvalues indicates the outer iteration index.

Algorithm 1. Power method: Inner Iterations

1: **function** INNER_ITERATIONS($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\phi}^{\text{guess}}$, λ^{guess})
2: $\boldsymbol{s} = \lambda^{\text{guess}} \boldsymbol{B}\phi^{\text{guess}}$
3: solve $\boldsymbol{A\phi} = \boldsymbol{s}$
4: **return** $\boldsymbol{\phi}$
5: **end function**

Algorithm 2. Power method: Outer iterations

function OUTER_ITERATIONS($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\phi}^{\text{guess}}$, λ^{guess})
  $\boldsymbol{\phi}^{(0)} = \boldsymbol{\phi}^{\text{guess}}$, $\lambda^{(0)} = \lambda^{\text{guess}}$
  $i\_max = 1000$
  $k\_tol = 10^{-8}$
  $i = 0$
  `not_converged = True`
  while `not_converged` **do**
    $\boldsymbol{\phi}^{(i+1)} =$ INNER_ITERATIONS($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\phi}^{(i)}$, $\lambda^{(i)}$) ! *to solve Equation (10)*
    $\boldsymbol{\phi}^{(i+1)} \leftarrow \dfrac{\boldsymbol{\phi}^{(i+1)}}{\boldsymbol{b}^T\boldsymbol{B}\boldsymbol{\phi}^{(i+1)}}$! *normalising the flux*
    $\lambda^{(i+1)} = \dfrac{\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{\phi}^{(i+1)}}{\boldsymbol{b}^T\boldsymbol{B}\boldsymbol{\phi}^{(i+1)}}$
    if ($i = i\_max$ **or** $|k_{\text{eff}}^{(i+1)} - k_{\text{eff}}^{(i)}| < k\_tol$) **then**
      `not_converged = False`
      $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}^{(i+1)}$
      $\lambda \leftarrow \lambda^{(i+1)}$
    else
      $i \leftarrow i + 1$
    end if
  end while
  return $\boldsymbol{\phi}$, λ
end function

3 | PROJECTION-BASED REDUCED-ORDER MODELING

This section briefly describes the key stages of constructing a projection-based reduced-order model, including explanations of proper orthogonal decomposition (POD) and the method of snapshots, singular value decomposition, selecting POD basis functions, and constructing a projection-based reduced-order model.

3.1 | Proper orthogonal decomposition

Proper orthogonal decomposition was introduced by Lumley[14] to identify the most energetic coherent structures within turbulent flow fields. The method can be applied to data from experiments as well as from numerical simulations. In the latter case, the numerical data are gathered into a matrix, $\boldsymbol{S}$, whose columns are individual snapshots pertaining to the solution of the high-fidelity model for a particular set of parameters. For a problem with N spatial degrees of freedom and M snapshots, the snapshots' matrix will be of size N by M. According to POD, a snapshot can be decomposed into a finite number of coefficients (α_i) and POD basis functions or modes ($\boldsymbol{\psi}_i$) as follows:

$$\boldsymbol{\phi} = \sum_i \alpha_i \boldsymbol{\psi}_i, \tag{11}$$

where α_i is a scalar and $\boldsymbol{\phi},\ \boldsymbol{\psi}_i \in \mathbb{R}^N$. POD gives the optimal linear representation of the data in the snapshots' matrix. In order to find the POD basis functions, the average squared error between the snapshots and their projection onto the basis functions is minimized.[59] This leads to an eigenvalue problem, which was originally written in the form

$$\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{\psi}_j = \mu_j\boldsymbol{\psi}_j \quad \forall j \in N, \tag{12}$$

where μ_j is the eigenvalue, $\boldsymbol{\psi}_j$ is the eigenvector, and, without loss of generality, it is assumed that the eigenvalues are given in descending order and that the eigenvectors have been orthonormalized. For a large number of degrees of freedom,

this problem quickly becomes challenging to solve, and it was Sirovich[15] who noted that the same (nonzero) eigenvalues could be found by instead solving the following, more tractable problem

$$\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{\varphi}_k = \mu_k\boldsymbol{\varphi}_k \quad \forall k \in M, \tag{13}$$

for eigenvectors $\boldsymbol{\varphi}_k$. Called the method of snapshots, its lower computational burden has led to a wide uptake of POD for analyzing results from numerical simulations.[12,13] Assuming, again, that the eigenvalues are arranged in descending order, the eigenvectors of the two problems (12) and (13) are related through

$$\boldsymbol{\psi}_k = \frac{1}{\sqrt{\mu_k}}\boldsymbol{S}\boldsymbol{\varphi}_k \quad \forall k \in M. \tag{14}$$

Finding the POD basis functions can also be done by taking a singular value decomposition of the snapshots' matrix,[59] which we now describe briefly. For a real $N \times M$ matrix S, where $N \geqslant M$, the singular value decomposition of $\boldsymbol{S}$ is as follows:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}, \tag{15}$$

where $\boldsymbol{U}$ is an $N \times N$ matrix consisting of orthonormalized eigenvectors associated with the M largest eigenvalues of $\boldsymbol{S}\boldsymbol{S}^T$ and $\boldsymbol{V}$ is an $M \times M$ matrix consisting of orthonormalized eigenvectors associated with $\boldsymbol{S}^T\boldsymbol{S}$. Here, the POD basis functions are the columns of $\boldsymbol{U}$. The matrix Σ contains the nonnegative square roots of the eigenvalues of $\boldsymbol{S}^T\boldsymbol{S}$ (called the singular values) on its diagonal. These singular values are ordered as follows:

$$\sigma_1 \geqslant \sigma_2 \geqslant \ \ldots \ \geqslant \sigma_M \geqslant 0. \tag{16}$$

A reduced rank approximation to $\boldsymbol{S}$ can be found by setting to zero all but the first P largest singular values in the matrix Σ, and then pre- and postmultiplying by $\boldsymbol{U}$ and $\boldsymbol{V}^T$, respectively. This will give the optimal rank P approximation of the matrix $\boldsymbol{S}$ according to the Eckart–Young–Mirsky theorem.[60]

3.2 | POD basis functions

For some problems, a few basis functions will capture much of the behavior of the system (as represented by the snapshots), and this is seen by the magnitude of the first few singular values being much larger than the remaining singular values. In such cases, the POD basis can be truncated without introducing much error. To determine how many basis functions should be kept, an empirical expression can be used that relates the number of basis functions retained to the fraction of information captured by them. The amount of information carried by each basis function depends on the square of its singular value. Suppose that the fraction of information of the original system to be captured is γ, where $0 \leqslant \gamma \leqslant 1$, then the lowest integer value of P is sought, such that the following is satisfied:

$$\frac{\sum_{i=1}^{P}\sigma_i^2}{\sum_{i=1}^{M}\sigma_i^2} \geqslant \gamma. \tag{17}$$

After truncation is applied (if indeed it is), the basis functions are stored in the matrix $\boldsymbol{R} \in \mathbb{R}^{N\times P}$. If the SVD has been used then the first P basis functions from $\boldsymbol{U}$ are retained to make up the columns of the matrix $\boldsymbol{R}$. If no truncation is applied then $P = M$ (i.e., the number of basis functions is equal to the number of snapshots). If either of the related eigenvalue problems are solved to determine the basis functions, then, once orthonormalized, the first P eigenvectors are retained to make up $\boldsymbol{R}$, either from Equation (12) or Equations (13) and (14). The complexity of solving the eigenvalue problems is $\mathcal{O}(N^3)$ for Equation (12) and $\mathcal{O}(M^3)$ for Equation (13). For an SVD, the complexity is $\mathcal{O}(NM^2)$ (see Golub et al.[54]), thus, for large problems (many snapshots but where $N \gg M$), the eigenvalue problem in Equation (13) will be the cheapest to solve.

Once $\boldsymbol{R}$ has been determined, the P reduced variables $\boldsymbol{\alpha}$ associated with a snapshot $\boldsymbol{\phi}$ can be determined from the basis functions by:

$$\boldsymbol{\alpha} = \boldsymbol{R}^T\boldsymbol{\phi}, \tag{18}$$

and the cell-based scalar flux values can be recovered from the reduced variables by:

$$\boldsymbol{\phi} = \boldsymbol{R}\boldsymbol{\alpha}, \tag{19}$$

where $\boldsymbol{R}$ represents the POD basis functions and $\boldsymbol{\alpha}$ the POD coefficients or reduced variables. Equation (19) is the matrix equivalent of Equation (11).

3.3 | Constructing a reduced-order model for criticality with POD

Having found the POD basis functions for the system, the discretized governing equations can be projected onto the reduced space. By inserting Equation (19) into Equation (10) and premultiplying by $\boldsymbol{R}^T$, the reduced system of equations is found:

$$\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{\alpha} = \lambda\boldsymbol{R}^T\boldsymbol{B}\boldsymbol{R}\boldsymbol{\alpha}, \tag{20}$$

where the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ both depend on the material parameters Σ^a, Σ^s, Σ^f. This reduced system of equations has matrices of size P by P, whereas the original system given in Equation (10) is N by N where $N \gg P$.

In this article, the aim is to construct a reduced-order model based on a set of snapshots, each of which correspond to a particular control-rod configuration, and to use this model to predict solutions for previously unseen control-rod configurations. An important point to address is how to modify the reduced system in Equation (20) for unseen parameters. One possibility is to exploit an affine decomposition of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, by calculating $\boldsymbol{R}^T\boldsymbol{A}_i\boldsymbol{R}$ and $\boldsymbol{R}^T\boldsymbol{B}_i\boldsymbol{R}$ as part of the offline stage, where $\boldsymbol{A}_i$ is the ith matrix in the affine decomposition of $\boldsymbol{A}$, similarly for $\boldsymbol{B}_i$. These matrices could be interpolated in order to approximate the matrices corresponding to an unseen parameter.[48] Although accurate for the problem solved by Heaney et al.,[48] this method can in general be inaccurate for unseen parameters.[61] Another approach would be to use the Matrix Discrete Empirical Interpolation Method,[62] which involves sampling the high-fidelity model matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ at a relatively low number of points before pre- and postmultiplying by $\boldsymbol{R}^T$ and $\boldsymbol{R}$, respectively, in order to estimate $\boldsymbol{A}$ and $\boldsymbol{B}$ for the unseen parameter. For both these methods, the sampling of $\boldsymbol{A}$ and $\boldsymbol{B}$, and their pre- and post-multiplication by the POD basis functions would be part of the offline stage, whereas the interpolation of the resulting reduced matrices would be part of the online stage. As the aim of this article is to demonstrate an autoencoder-based ROM for eigenvalue problems, a simple method is chosen which avoids approximation due to sampling, but would be impractical for large problems. To evaluate $\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}$ and $\boldsymbol{R}^T\boldsymbol{B}\boldsymbol{R}$ for an unseen parameter, the high-fidelity model is used to assemble $\boldsymbol{A}$ and $\boldsymbol{B}$ (Equations 6 and 10) and then pre- and postmultiplied by $\boldsymbol{R}^T$ and $\boldsymbol{R}$, respectively. This would not be practical for systems with large number of degrees of freedom, as the online stage, in which $\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}$ and $\boldsymbol{R}^T\boldsymbol{B}\boldsymbol{R}$ are approximated for the unseen parameter, should be independent of the high-fidelity model for reasons of computational efficiency. Future work will involve larger scale problems, for which more computationally efficient methods will be investigated.

For the POD-based ROM, the offline stage consists of solving the high-fidelity model many times for different material parameters (cross-sections), see Equation (10) and Algorithm 2. In this article, the POD basis functions are determined by solving the eigenvalue problem associated with the method of snapshots, see Equation (13). The online stage of the POD-based ROM consists of assembling the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ for a given set of parameters, projecting these matrices onto the reduced space and then solving with the power method. The outer iteration algorithm for the POD-based ROM is very similar to that of the high-fidelity model (Algorithm 2), except for the passing down of the POD basis functions from the outer to the inner iterations. The so-called inner iterations are described for POD-based ROM in Algorithm 3. (Although there are no iterations, the description "inner iterations" is still used for Algorithm 3). Here the reduced-order model, Equation (20), is used to solve for the scalar fluxes. The right-hand side of Equation (20) is evaluated using the current value of scalar flux, resulting in a system that can be solved for the reduced variables. As the system is small, the problem is solved directly with Gaussian elimination. There is no need to use the eigenvalue in the source term (see

Algorithm 3. POD-based reduced-order model: Inner iterations

1: **function** POD_INNER_ITERATIONS($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\phi}$, λ, $\boldsymbol{R}$)
2: $\boldsymbol{s} = \lambda \left(\boldsymbol{R}^T \boldsymbol{B}\right) \boldsymbol{\phi}$! *set a source with values from the outer iterations*
3: solve the reduced-order model for $\boldsymbol{\alpha}$:
4: $\left(\boldsymbol{R}^T \boldsymbol{A} \boldsymbol{R}\right) \boldsymbol{\alpha} = \boldsymbol{s}$
5: $\boldsymbol{\phi} = \boldsymbol{R}\boldsymbol{\alpha}$! *find the updated scalar flux from the reduced variables*
6: **return** $\boldsymbol{\phi}$
7: **end function**

line 2) as, once passed back to the outer iterations, the flux is normalized. However, it is included here as it is needed in the inner iterations for the autoencoder-based reduced order models.

4 | AUTOENCODERS AND THEIR INCLUSION IN PROJECTION-BASED ROMS

First, the two types of autoencoder used in this article are described: a standard, fully connected autoencoder and an SVD-autoencoder. Then, their incorporation into projection-based reduced-order models is explained. The use of autoencoders for dimensionality reduction in eigenvalue problems represents the main novelty in this article. To the authors' best knowledge, the hybrid SVD-autoencoder is also a new method.

4.1 | Autoencoders

4.1.1 | A standard autoencoder

An autoencoder is a special type of feed-forward neural network that is trained to learn the identity map. A bottleneck at its central layer forces the autoencoder to learn a reduced representation of the data. The simplest (vanilla) autoencoder has an input layer with n neurons, a hidden layer with m neurons where $m < n$, and an output layer with n neurons. The values of the neurons in the hidden layer define the latent space and can be referred to as the latent variables. Networks with additional hidden layers can learn more complicated patterns creating a more effective representation of complex data. An autoencoder can be split into two networks: the encoder, which maps an input to the latent variables, and a decoder, which maps from the latent variables to the output. If the input vector is given by $\boldsymbol{x} \in \mathbb{R}^n$, the output vector by $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ and the latent variables of the hidden layer by $\boldsymbol{y} \in \mathbb{R}^m$, then the autoencoder can be written as a composition of functions as follows

$$\hat{\boldsymbol{x}} = f^{\mathrm{AE}}(\boldsymbol{x}; \boldsymbol{w}^{\mathrm{AE}}) = f^{\mathrm{DEC}}\left(\boldsymbol{y}; \boldsymbol{w}^{\mathrm{DEC}}\right) = f^{\mathrm{DEC}}\left(f^{\mathrm{ENC}}(\boldsymbol{x}; \boldsymbol{w}^{\mathrm{ENC}}); \boldsymbol{w}^{\mathrm{DEC}}\right), \tag{21}$$

where f^{AE}, f^{ENC}, and f^{DEC} represent the autoencoder, encoder, and decoder, respectively, with associated weights $\boldsymbol{w}^{\mathrm{AE}}$, $\boldsymbol{w}^{\mathrm{ENC}}$ and $\boldsymbol{w}^{\mathrm{DEC}}$.

Figure 1 illustrates a fully connected autoencoder with six layers (only layers with trainable weights are included in the total), five of these are hidden layers. Neurons can have connections with sending neurons from the previous layer (except for those in the input layer) and with receiving neurons from the following layer (except for those in the output layer). In this diagram, the input and output layers have six neurons each and the latent space is of dimension 1. The gray arrows indicate how the information passes through the autoencoder from left to right. The input to a neuron is calculated as a weighted sum of the outputs of the sending neurons to which it is connected and a bias term. The output of the neuron is the evaluation of the activation function for this weighted sum. For example, the output of the ith neuron of the hidden layer of the vanilla autoencoder described above, is given by

$$y_i = g\left(\sum_{j=1}^{n} w_{ij}^{\mathrm{ENC}} x_j + b_i\right), \tag{22}$$

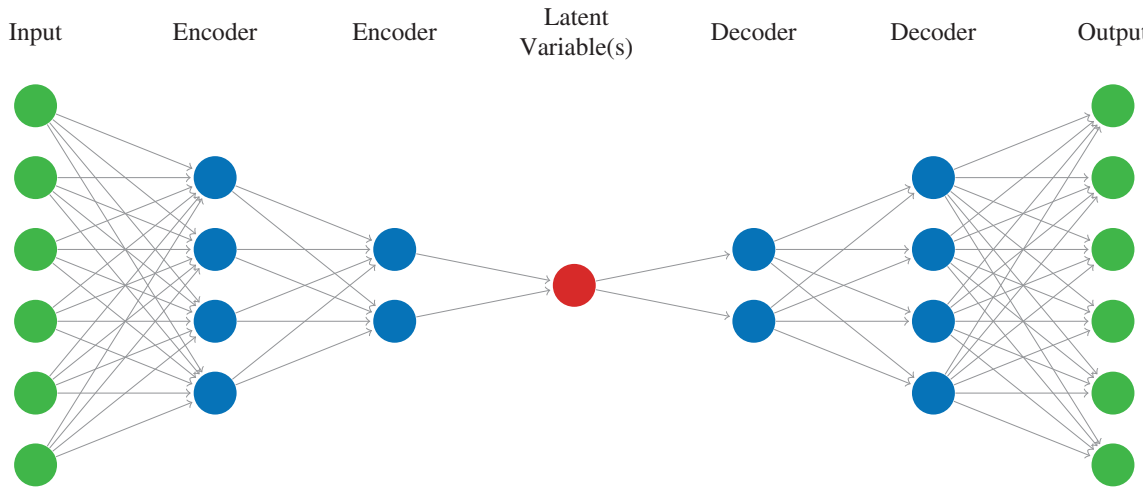


FIGURE 1 The architecture of a fully connected autoencoder comprising of three encoder layers and three decoder layers. This network compresses six variables into one latent variable and then expands back out to the original six variables

where g is the activation function, b_i is the bias of the ith neuron, $\{x_j\}_{j=1}^{n}$ are the values of the sending neurons connected to the ith neuron and w_{ij} is the weight of the jth sending neuron of the ith neuron in the hidden layer.

The data available to build the network are customarily split into three parts: training data, validation data, and test data. The use of these data sets in the training process will now be described. The activation function and number(s) of neurons in the hidden layer(s) are examples of hyperparameters of the network, and the weights and biases are referred to as parameters of the network. For given hyperparameters, the objective of training the autoencoder is to find weights that minimize the so-called loss function, which, for regression problems, is often taken as the mean squared error of the reconstruction:

$$\frac{1}{N^{\text{ex}}}\sum_{j=1}^{N^{\text{ex}}}||\boldsymbol{x}_j-\hat{\boldsymbol{x}}_j||^2=\frac{1}{N^{\text{ex}}}\sum_{j=1}^{N^{\text{ex}}}||\boldsymbol{x}_j-f^{\text{ae}}\left(\boldsymbol{x}_j;\boldsymbol{w}^{\text{ae}}\right)||^2, \tag{23}$$

where N^{ex} is the number of samples or examples used to train the autoencoder and the norm used is the Euclidean norm. Based on gradient descent-type methods, a back-propagation algorithm is used to optimize the weights by solving

$$\boldsymbol{w}^{\text{ae}}=\arg\min_{\tilde{\boldsymbol{w}}^{\text{ae}}}\frac{1}{2}\sum_{j=1}^{N^{\text{ex}}}||\boldsymbol{x}_j-f^{\text{ae}}\left(\boldsymbol{x}_j;\tilde{\boldsymbol{w}}^{\text{ae}}\right)||^2. \tag{24}$$

In order to find the optimal hyperparameters, less sophisticated approaches than back-propagation are employed, such as cross-validation.[63] This involves training a number of networks each with a different set of hyperparameters and comparing the performance of these networks on the validation data. The best-performing network is selected and can be retrained with both the training and validation data combined. The performance of this network is then assessed on the test data.

4.1.2 | SVD autoencoder

The length of the input to the autoencoder is one factor that governs the difficulty of training an autoencoder. The longer these vectors are, the more parameters (weights) there are to be adjusted during optimization, which increases the computational cost. To combat this, the dimensionality reduction can be split into two steps. The first step is to apply an SVD to the inputs of the autoencoder (i.e., the snapshots). A basis is obtained and the snapshots are projected onto the resulting low-dimensional space yielding reduced variables $\boldsymbol{\alpha}$ (just as is done in POD-based ROM), see Equation (18). This reduces the length of the input vectors from the number of degrees of freedom of the problem (N) to the number of basis functions (P where $P \ll N$). The second step is to train a fully connected autoencoder with the reduced variables of the snapshots (or the subset of these that have been selected for training), thereby further reducing the dimensionality.

4.2 | Projection-based ROMs with autoencoders

POD defines a linear map between the scalar fluxes $\boldsymbol{\phi}$, and the corresponding reduced variables, $\boldsymbol{\alpha}$, see Equation (18). Using an autoencoder for the dimensionality reduction means that the map between these variables will, in general, be nonlinear. For eigenvalue problems, which are inherently nonlinear,[64] using an autoencoder rather than POD for the dimensionality reduction introduces an additional level of nonlinearity. To construct a reduced-order model, the nonlinear mapping between the reduced variables and the scalar fluxes needs to be linearized and the reduced system of equations needs to be derived. Both points are now elucidated.

Suppose that we have a known state of reduced variables $\tilde{\boldsymbol{\alpha}}$ and scalar fluxes $\tilde{\boldsymbol{\phi}}$, and wish to find the change in the scalar flux $\Delta\boldsymbol{\phi}$ due to a small change in the reduced variables $\Delta\boldsymbol{\alpha}$. We can linearize the map $\boldsymbol{C}$ between these variables using a first-order Taylor series as follows:

$$\Delta\boldsymbol{\phi} = \boldsymbol{C}(\tilde{\boldsymbol{\alpha}})\ \Delta\boldsymbol{\alpha}, \tag{25}$$

$$\text{where } \Delta\boldsymbol{\phi} = \boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}, \tag{26}$$

$$\Delta\boldsymbol{\alpha} = \boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}}, \tag{27}$$

$$\text{and } \boldsymbol{C}(\tilde{\boldsymbol{\alpha}}) = \frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}\boldsymbol{\alpha}}\Big|_{\tilde{\alpha}}. \tag{28}$$

The scalar flux $\boldsymbol{\phi}$ can therefore be approximated as

$$\boldsymbol{\phi} = \boldsymbol{C}(\tilde{\boldsymbol{\alpha}})\boldsymbol{\alpha} - \boldsymbol{C}(\tilde{\boldsymbol{\alpha}})\tilde{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\phi}}. \tag{29}$$

In order to calculate $\boldsymbol{C}(\tilde{\boldsymbol{\alpha}})$, perturbations are made to $\tilde{\boldsymbol{\alpha}}$ from which the entries of $\boldsymbol{C}$ can be inferred. Each of the P entries in $\tilde{\boldsymbol{\alpha}}$ are perturbed in turn by a small value, which yields a column of entries in $\boldsymbol{C}$. The kth perturbation simply adds a small number ϵ to the kth component of $\tilde{\boldsymbol{\alpha}}$, so, for example, the second perturbation is given by

$${}^{2}\Delta\boldsymbol{\alpha} = \underbrace{(0, \epsilon, 0, 0, \ \dots\ , 0)^{T}}_{P \text{ entries}}. \tag{30}$$

The decoder can be used to calculate the scalar flux corresponding to the perturbed reduced variables ${}^{k}\boldsymbol{\alpha}$:

$${}^{k}\boldsymbol{\phi} = f^{\mathrm{DEC}}\left({}^{k}\boldsymbol{\alpha}\right) = f^{\mathrm{DEC}}\left(\tilde{\boldsymbol{\alpha}} + {}^{k}\Delta\boldsymbol{\alpha}\right). \tag{31}$$

The change in the scalar flux can be calculated by subtracting the known state $\tilde{\boldsymbol{\phi}}$ from the above. The kth column of $\boldsymbol{C}$ is now known:

$$C_{jk}(\tilde{\boldsymbol{\alpha}}) = \frac{{}^{k}\phi_j - \tilde{\phi}_j}{\epsilon} = \frac{{}^{k}\Delta\phi_j}{\epsilon} \qquad \forall j, \tag{32}$$

where ϕ_j is the jth component of the scalar flux. Once every component of the reduced variables $\boldsymbol{\alpha}$ has been perturbed, the $\boldsymbol{C}$ matrix will be fully determined. Having found $\boldsymbol{C}$, the map between the reduced variables, $\boldsymbol{\alpha}$, and the high-fidelity model variables, $\boldsymbol{\phi}$, can be calculated using Equation (29). In this equation, the nonlinearity is introduced by $\boldsymbol{C}$ as its values depend on the reduced variables. When using a dimensionality-reduction method such as POD, this map is linear and is independent of the reduced variables, see Equation (19).

With the map between the reduced variables and the high-fidelity model variables now established, the projection-based ROM, using an autoencoder, can now be developed. When solving the high-fidelity model, an inner iteration loop solves for the scalar flux, and an outer iteration loop normalizes the flux and solves for the eigenvalue. Instead of solving a reduced-order model of the generalized eigenvalue problem, directly, shown in Equation (20), we replace the inner iterations with a reduced-order model to approximate the flux, but use the outer iterations, unmodified, to normalize the flux and find the corresponding eigenvalue. Therefore, we look to develop a reduced-order model for the equation solved in the inner iterations of the power method, see line 3 of Algorithm 1,

$$\boldsymbol{A}\boldsymbol{\phi} = \boldsymbol{s}, \tag{33}$$

where the right-hand side term, $\boldsymbol{s} = \lambda^{(\cdot)}\boldsymbol{B}\boldsymbol{\phi}^{(\cdot)}$, uses the latest values for the eigenvalue and scalar flux that have been calculated in the outer iterations. Substituting Equation (29) into Equation (33), rearranging and premultiplying the result by $\boldsymbol{C}^T$ gives the reduced system of equations for the autoencoder-based ROM:

$$\left(\boldsymbol{C}^T(\tilde{\boldsymbol{\alpha}})\boldsymbol{A}\boldsymbol{C}(\tilde{\boldsymbol{\alpha}})\right)\Delta\boldsymbol{\alpha} = \boldsymbol{C}^T(\tilde{\boldsymbol{\alpha}})\boldsymbol{s} - \boldsymbol{C}^T(\tilde{\boldsymbol{\alpha}})\boldsymbol{A}\tilde{\boldsymbol{\phi}}. \tag{34}$$

As this system is relatively small for the examples used in this article, it is solved with Gaussian elimination. The algorithm for the inner iterations, which solve for the scalar flux, is given in Algorithm 4 in which the dependence of $\boldsymbol{C}$ on $\tilde{\boldsymbol{\alpha}}$ has been dropped for readability. Regularization has been applied to the reduced-order model by adding a small number, ε, to the diagonal of $\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C}$ as, without this, there is no guarantee that this matrix will be nonsingular. The regularization will also ensure that the iterative change in reduced variables ($\Delta\boldsymbol{\alpha}$) will not become too large. Note that this is a modified power iteration, as the eigenvalue is included in the source term. For the standard power method this is not required when determining the flux, as normalization renders this unnecessary. However, in line 11 of the algorithm, the right-hand side of this expression has a source term and an additional term, meaning that the size of the source term is now important. Both autoencoders described in Sections 4.1.1 and 4.1.2 can be used in the reduced-order model framework described in this section.

We remark that in order to solve the intrinsically nonlinear eigenvalue problem, (whether from the high-fidelity model or reduced model) the power method is deployed. This method uses an inner iteration to solve for the scalar flux, and an outer iteration to normalize the flux and solve for the eigenvalue. Due to the linear map $\boldsymbol{R}$ between the reduced variables and the high-fidelity model variables, the inner iterations for the high-fidelity model and POD-based ROM solve a linear system (line 3 in Algorithm 1 and line 4 in Algorithm 3, respectively). However, for the AE-based ROM, the map between the high-fidelity model variables and the reduced variables is nonlinear, which results in an additional iteration loop (lines 8 to 21 in Algorithm 4), within which, is the linear solve (line 11).

Algorithm 4. AE-based reduced-order model: Inner iterations

1: **function** INNER_ITERATIONS($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\phi}$, λ)
2: $k\_max = 100$
3: $\alpha\_tol = 10^{-8}$
4: $\boldsymbol{s} = \lambda\boldsymbol{B}\boldsymbol{\phi}$! *set a source with values from the outer iterations*
5: $\boldsymbol{\phi}^{(0)} = \boldsymbol{\phi}$; $\lambda^{(0)} = \lambda$; $\boldsymbol{\alpha}^{(0)} = f^{\mathrm{ENC}}(\boldsymbol{\phi}^{(0)})$! *initialise with values from the outer iterations*
6: $k = 0$
7: `not_converged = True`
8: **while** `not_converged` **do**
9: calculate $\boldsymbol{C}^{(k)}$ for $\boldsymbol{\alpha}^{(k)}$ using Equation (32)
10: solve the reduced-order model:
11: $\left((\boldsymbol{C}^{(k)})^T\boldsymbol{A}\boldsymbol{C}^{(k)} + \varepsilon\boldsymbol{I}\right)\Delta\boldsymbol{\alpha}^{(k+1)} = (\boldsymbol{C}^{(k)})^T\boldsymbol{s} - (\boldsymbol{C}^{(k)})^T\boldsymbol{A}\boldsymbol{\phi}^{(k)}$
12: $\boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} + \Delta\boldsymbol{\alpha}^{(k+1)}$! *update variables*
13: $\Delta\boldsymbol{\phi}^{(k+1)} = \boldsymbol{C}^{(k)}\Delta\boldsymbol{\alpha}^{(k+1)}$
14: $\boldsymbol{\phi}^{(k+1)} = \boldsymbol{\phi}^{(k)} + \Delta\boldsymbol{\phi}^{(k+1)}$
15: **if** ($\Delta\boldsymbol{\alpha}^{(k+1)} < \alpha\_tol$ **or** $k = k\_max$) **then**
16: `not_converged = False`
17: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}^{(k+1)}$
18: **else**
19: $k \leftarrow k + 1$
20: **end if**
21: **end while**
22: **return** $\boldsymbol{\phi}$
23: **end function**

5 | RESULTS

The methods outlined in Sections 3 (POD-based ROM) and 4 (autoencoder-based ROMs) are applied to two test cases. Before presenting the results, the error measures that are used to compare the methods are introduced, followed by an explanation of how the material parameters (cross-sections) are homogenized. The first test case is a 1D slab reactor and results from a POD-based ROM and an AE-based ROM are presented. Further comparisons are made between a POD-based ROM and an AE-based ROM constructed with fewer reduced variables. The second test case is a simplified 2D reactor. For this case, results are presented for ROMs based on POD, on an autoencoder and a hybrid SVD-autoencoder. The reduced-order models were developed in python using Keras[65] with the TensorFlow backend.[66]

5.1 | Error measures

To assess the accuracy of the results, a number of error measures are introduced. In all the expressions, the reduced-order model solutions are measured against the high-fidelity model results. First, the error committed in the dimensionality reduction step is evaluated. For the POD-based ROM, this error is incurred by projecting the high-fidelity model solutions onto the reduced space and for the AE-based ROM this error occurs during the compression and decompression of the high-fidelity model solutions by the autoencoder. For the SVD-AE-based ROM, the error is due to both projecting the solution onto a space defined by POD basis functions, and compressing and decompressing with an autoencoder. The normalized maximum errors in the reconstruction of the solution are defined here using

$$e_{\max}^{\text{POD}}(\boldsymbol{\phi}^{\text{HFM}}) = \frac{\phi_k^{\text{HFM}} - \left(\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{\phi}^{\text{HFM}}\right)_k}{\|\boldsymbol{\phi}^{\text{HFM}}\|_\infty} \quad \text{where } k = \underset{i \in \text{cells}}{\arg\max}\left|\phi_i^{\text{HFM}} - \left(\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{\phi}^{\text{HFM}}\right)_i\right|, \tag{35}$$

$$e_{\max}^{\text{AE}}(\boldsymbol{\phi}^{\text{HFM}}) = \frac{\phi_k^{\text{HFM}} - \left(f^{\text{AE}}(\boldsymbol{\phi}^{\text{HFM}})\right)_k}{\|\boldsymbol{\phi}^{\text{HFM}}\|_\infty} \quad \text{where } k = \underset{i \in \text{cells}}{\arg\max}\left|\phi_i^{\text{HFM}} - \left(f^{\text{AE}}(\boldsymbol{\phi}^{\text{HFM}})\right)_i\right|, \tag{36}$$

$$e_{\max}^{\text{SVD-AE}}(\boldsymbol{\phi}^{\text{HFM}}) = \frac{\phi_k^{\text{HFM}} - \left(\boldsymbol{R}f^{\text{AE}}\left(\boldsymbol{R}^T\boldsymbol{\phi}^{\text{HFM}}\right)\right)_k}{\|\boldsymbol{\phi}^{\text{HFM}}\|_\infty} \quad \text{where } k = \underset{i \in \text{cells}}{\arg\max}\left|\phi_i^{\text{HFM}} - \left(\boldsymbol{R}f^{\text{AE}}\left(\boldsymbol{R}^T\boldsymbol{\phi}^{\text{HFM}}\right)\right)_i\right|, \tag{37}$$

for the POD-based ROM, the AE-based ROM, and SVD-AE-based ROM, respectively, where $\|\cdot\|_\infty$ represents the maximum norm, i and k represent cell indices, and the superscript "HFM" indicates the flux is associated with the high-fidelity model.

Second, the error in the scalar flux of the reduced-order models is calculated. The normalized maximum error in the flux profile is determined by

$$e_{\max}(\boldsymbol{\phi}^{\text{ROM}}) = \frac{\phi_j^{\text{HFM}} - \phi_j^{\text{ROM}}}{\|\boldsymbol{\phi}^{\text{HFM}}\|_\infty} \quad \text{where } j = \underset{i \in \text{cells}}{\arg\max}|\phi_i^{\text{HFM}} - \phi_i^{\text{ROM}}|, \tag{38}$$

in which the superscript "ROM" indicates the solution is from a reduced-order model (either POD-based or autoencoder-based), and i and j represent cell indices.

Calculating the error in k_{eff} is done by comparing the value calculated by the high-fidelity model with that of the reduced-order model:

$$e(k_{\text{eff}}^{\text{ROM}}) = k_{\text{eff}}^{\text{HFM}} - k_{\text{eff}}^{\text{ROM}}. \tag{39}$$

As k_{eff} is usually close to one, no normalization is applied.

Finally, an average maximum error is defined, where, for N sets of material parameters resulting in N solutions of the high-fidelity model and N solutions of one of the reduced-order models:

$$\bar{e}_{\max}(\boldsymbol{\phi}^{\text{ROM}}) = \frac{1}{N}\sum_{l=1}^{N}\frac{\left|\phi_j^{\text{HFM}}(\boldsymbol{\mu}_l) - \phi_j^{\text{ROM}}(\boldsymbol{\mu}_l)\right|}{\|\phi^{\text{HFM}}(\boldsymbol{\mu}_l)\|_\infty}, \tag{40}$$

where $\boldsymbol{\mu}_l$ refers to the parameters (i.e., the macroscopic cross-sections) used for the lth problem. Until this point, dependence of the solution on the material parameters has not been explicitly indicated (for readability), however here it is required. A similar error measure can also be used to find the average maximum error in k_{eff}:

$$\bar{e}(k_{\text{eff}}^{\text{ROM}}) = \frac{1}{N}\sum_{l=1}^{N} \left| \, k_{\text{eff}}^{\text{HFM}}(\boldsymbol{\mu}_l) - k_{\text{eff}}^{\text{ROM}}(\boldsymbol{\mu}_l) \, \right|, \tag{41}$$

and the average maximum error in the reconstruction:

$$\bar{e}_{\text{max}}^{\text{DR}}(\boldsymbol{\phi}^{\text{HFM}}) = \frac{1}{N}\sum_{l=1}^{N} \left| \, e_{\text{max}}^{\text{DR}}(\boldsymbol{\phi}^{\text{HFM}}) \, \right|, \tag{42}$$

where DR represents one of the three dimensionality reduction methods used: POD, AE, or SVD-AE.

5.2 | Homogenizing the material parameters

In the test cases that follow, there are fuel regions and also control-rod regions, within which the control rods can be fully inserted, completely withdrawn or partially inserted. In the latter case, the cross-sections will be calculated by combining the cross-sections for fuel and control rod. Control-rod regions are assumed to have a uniform distribution of the averaged properties of the mixture within the system based on a mixing coefficient for each region. This can be written as follows

$$\Sigma_{\text{hom}}^{a} = r\Sigma_{\text{cr}}^{a} + (1-r)\Sigma_{\text{fuel}}^{a}, \tag{43}$$
$$\Sigma_{\text{hom}}^{s} = r\Sigma_{\text{cr}}^{s} + (1-r)\Sigma_{\text{fuel}}^{s}, \tag{44}$$

where the mixing coefficient, r, for a given region, lies between 0 and 1, and the subscripts "hom," "cr," and "fuel" indicate a homogenized cross-section, and cross-sections of the control rod and the fuel, respectively. If r were chosen to be proportional to the amount of control rod present in a control-rod region, the high absorption of the control rods would dominate the homogenized cross-section. In order to address this, the reciprocal of the absorption cross-sections are averaged:

$$\frac{1}{\Sigma_{\text{hom}}^{a}} = \frac{z}{\Sigma_{\text{cr}}^{a}} + \frac{1-z}{\Sigma_{\text{fuel}}^{a}} \qquad \text{where } z \in [0,1]. \tag{45}$$

By combining Equations (43) and (45) it can be shown that the mixing coefficient is given by

$$r = \frac{z\Sigma_{\text{fuel}}^{a}}{z\Sigma_{\text{fuel}}^{a} + (1-z)\Sigma_{\text{cr}}^{a}}. \tag{46}$$

In this way, the desired amount of control rod in a region can be chosen by setting z, converting this into a value for r by using Equation (46), and then calculating the homogenized absorption and scattering cross-sections from Equations (43) and (44).

5.3 | 1D slab reactor

The length of the slab reactor is 10 cm, and it consists of fuel and two control-rod regions labeled as 1 and 2 in Figure 2. Region 1 is located between 2.2 and 2.5 cm on the x-axis, region 2, between 7.5 and 7.8 cm. This test case was used by Buchan et al.[47]

Table 1 contains the macroscopic cross-sections used for the control rods and the fuel. The control rods can be fully inserted, completely withdrawn or partially inserted into regions 1 and 2. In the latter case, the cross-sections for these regions will be calculated by combining the cross-sections for fuel and control rod through mixing coefficients as described in the previous section. The amount of control rod in region 1 will, in general, be different from that of region 2, henceforth,

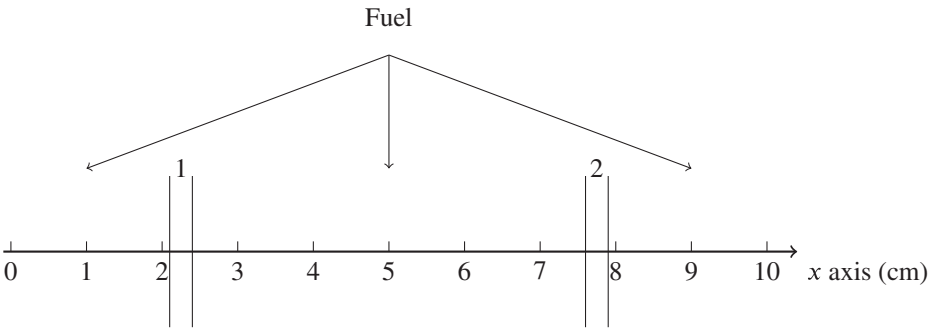


FIGURE 2 Geometry of a 1D slab reactor consisting of fuel and two control-rod regions (regions 1 and 2)

TABLE 1 Macroscopic cross-sections for the 1D slab reactor (units cm^{-1})

	Σ^a	Σ^s	Σ^f
Fuel	0.45	2	0.5
Control rod	0.9	2	0

z_1 and r_1 refer to region 1, and z_2 and r_2 refer to region 2. One hundred and forty-four cells were used in the control-volume discretization although 44 of these were used to enforce the boundary conditions, resulting in a system with 100 degrees of freedom.

To generate the data necessary to build the reduced-order models, 200 problems were solved using the high-fidelity model with different control-rod settings. This was done by choosing values for z_1 and z_2 at random from the interval [0, 1] and using Equations (46), (43), and (44) to calculate the corresponding cross-sections. The control-volume discretization along with the cross-sections determine the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in Equation (10). Given an initial guess for the scalar flux, the system in (10) is solved with the power method as described in Section 2.3 yielding 200 solutions of the high-fidelity model. One hundred of the high-fidelity model solutions were treated as snapshots and were used to generate basis functions (either by applying POD or by training an autoencoder). These snapshots are referred to as "seen" data (in the context of ROM) or as training data (in the context of neural networks). The remaining 100 solutions were used in the online stage to test how good the models were at predicting solutions for unseen data (i.e., parameter sets whose corresponding solutions were not included in the snapshots). This data are referred to as unseen data (in the context of ROM) and as test data (in the context of neural networks). Two reduced-order models are compared: a POD-based ROM and an autoencoder-based ROM. The models have 10 reduced variables (10 POD coefficients for the POD-based model and 10 latent variables for the autoencoder-based model). Following this, a POD-based model and an autoencoder-based model are developed with just two reduced variables.

5.3.1 | POD-based reduced-order model

The first method to be tested is a POD Galerkin method, which uses POD and the SVD to determine basis functions, projects the discretized governing equations onto the subspace defined by the POD basis functions, and results in a generalized eigenvalue problem of reduced dimension, Equation (20), to be solved in the online stage for a given set of macroscopic cross-sections. This method is described in Section 3.3. The 200 sets of parameters that generated the high-fidelity model results are used again to obtain the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. The system of equations describing the POD-based ROM, given in (20), is solved for each problem using the power method described in Algorithms 2 and 3, and Section 3.3 using numpy's Gaussian elimination function to solve for the reduced variables. Half of the results have been seen by the model and half have not been seen.

5.3.2 | Autoencoder-based reduced-order model

In this method the dimensionality reduction (or compression) is performed by an autoencoder instead of an SVD. Making this change requires an additional iteration loop when solving the reduced-order model as explained in Section 4.2. For

the online prediction, an initial flux guess is compressed using the encoder. Following this, the $\boldsymbol{C}$ matrix is formed by using Equation (28). The system in Equation (32) is then solved, the flux is decoded and passed to the outer iterations, and the whole process is repeated until k_{eff} converges. In order to compare with results from the previous section, the autoencoder compresses to 10 latent variables.

Autoencoders with different architectures were trained in order to optimize the hyperparameters. Once trained, the networks were evaluated on the validation data and the network with the lowest loss (or mean square error in this case) was assumed to have the optimal hyperparameters. From four to 16 layers were tested with up to 100 neurons in each layer. Smaller networks were seen to outperform larger networks, although the effect was less significant for model accuracy than the choice of activation function and batch size. It was found that the exponential linear unit (ELU)[67] was the best performing activation function, giving better results than the rectified linear unit (ReLU), the scaled exponential linear unit (SELU), and the hyperbolic tangent activation function (tanh). Batch sizes of 10, 25, and 100 were also compared during training and it was found that the smallest batch size resulted in the lowest validation loss. The initial learning rate of the optimizer was set at 10^{-3} and the minimum value to which this can reduce, upon stagnation of the loss, was varied from 10^{-4} to 10^{-7}.

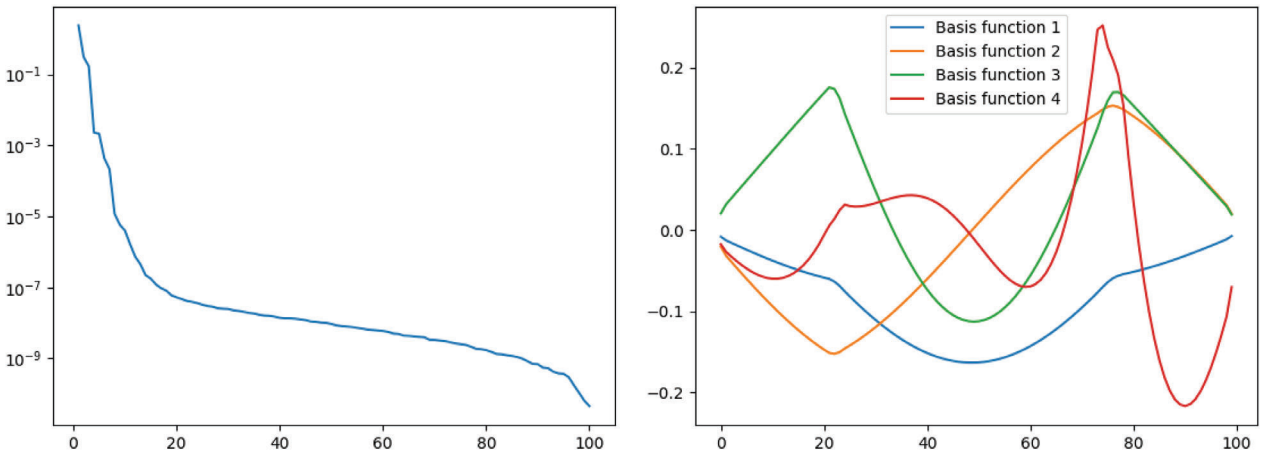
The best performing architecture comprised four fully connected layers with the following number of neurons in each layer:

$$100 \rightarrow 32 \rightarrow 10 \rightarrow 32 \rightarrow 100.$$

The activation function for every layer was the exponential linear unit (ELU); the optimizer used was "Nadam"[68] with an initial learning rate of 10^{-3} and a minimum learning rate of 10^{-6}; and the loss function was defined to be the mean squared reconstruction error. The snapshots were scaled between 0 and 1. The network was trained over 20,000 epochs using mini-batch gradient descent with a batch size of 10.

5.3.3 | Comparison of POD and AE as compression methods

For the POD-based ROM, an SVD is applied to the 100 snapshots and from a possible 100 basis functions, 10 are retained, which corresponds to capturing 99.999% of the information contained in the snapshots, see Figure 3(A). As can be seen from this figure, the magnitude of the singular values decreases rapidly from the first to the 20th singular value (by about seven orders of magnitude) before leveling off, and it can be expected from this that a POD model with a sufficient number of basis functions will perform well. The first four basis functions can be seen in Figure 3(B). Notice that the fourth basis function has more detail or structure than the other basis functions, following the general trend that the higher order basis functions tend to have more structure.



(A) Singular value against singular value index.

(B) First four basis functions.

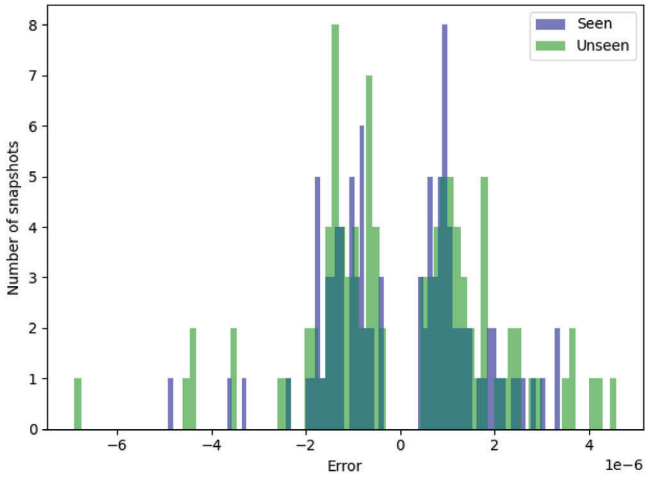
FIGURE 3 Singular values and basis functions of snapshots from the 1D slab reactor for the POD-based ROM

Figure 4(A) shows the reconstruction or compression error, $e_{\max}^{\mathrm{POD}}(\boldsymbol{\phi}^{\mathrm{HFM}})$, resulting from projecting high-fidelity model solutions onto the reduced space for both the seen data and the unseen data, see Equation (35). Both seen and unseen data sets yield a similar range of errors and the errors themselves are low: $\mathcal{O}(10^{-6})$.

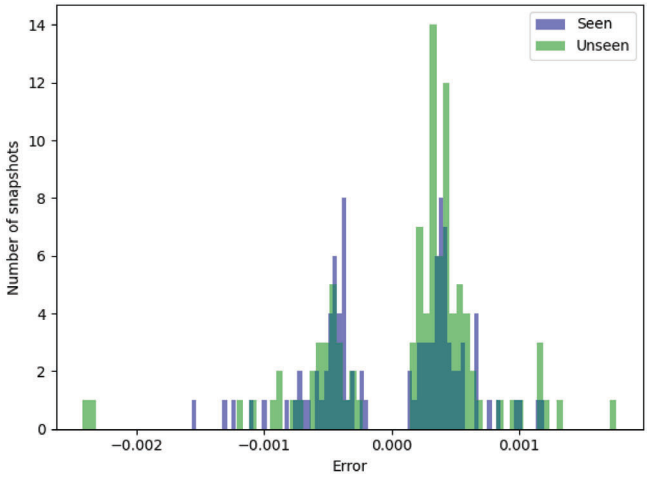
The autoencoder was trained on the same data to which the SVD was applied, that is, the 100 snapshots. The unseen high-fidelity model solutions were used as test data, to assess the prediction power of the autoencoder. Figure 4(B) shows the error (as defined by Equation (36)) between each HFM solution and the corresponding output of the autoencoder for seen and unseen data sets. The autoencoder performs well, achieving, at worst, an absolute error of just over 0.2% in the flux, with almost all of the absolute values of the error less than 0.15% . Although the seen and unseen data sets have a similar range of errors, it should be noted that these errors are much larger than those of the SVD (see Figure 4(A)). Shown in Figure 5 is a box and whisker plot of the values of the 10 latent variables over the training data set. It can be observed that five of the latent variables show very little variation across the seen data set.

5.3.4 | Comparison of POD-based and AE-based ROMs

Scalar fluxes and k_{eff} values from the POD-based and AE-based reduced-order models are now compared, where both ROMs have 10 reduced variables. Figure 6 shows the flux profile and the convergence of k_{eff} for the mixing coefficients $r_1 = 0.957$ and $r_2 = 0.115$. The high-fidelity model solution corresponding to these values was included in the snapshots, so this is a seen solution. Figure 6(A) shows that the flux profiles of the POD-based ROM and AE-based ROM very agree



(A) Reconstruction error from the SVD for the 1D test case. Seen data (snapshots) in blue, unseen data in green.



(B) Compression error from the autoencoder for the 1D test case. Seen data (snapshots) in blue, unseen data in green.

FIGURE 4 Errors for the two methods of compression used here for the 1D test case, compressing from 100 variables to 10

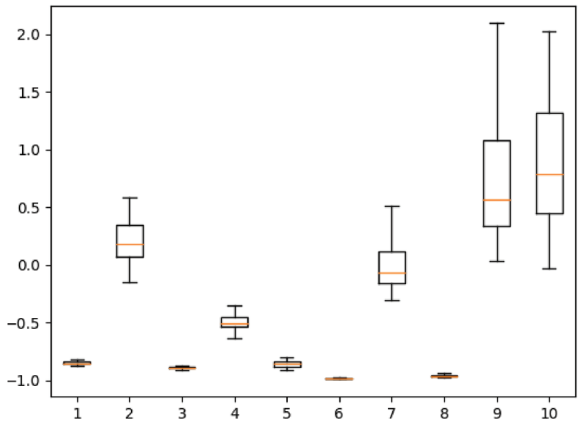
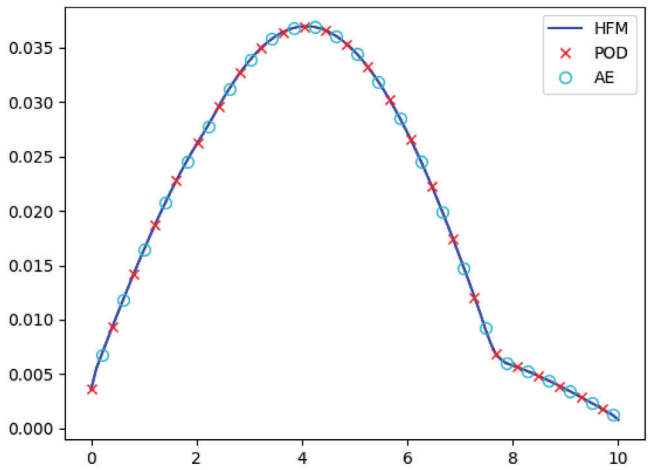


FIGURE 5 Box and whisker plot of the 10 latent variables generated from the autoencoder over the training data for the 1D test case. The orange line shows the median value, the box shows the upper and lower quartiles, and the whiskers indicate the minimum and maximum values

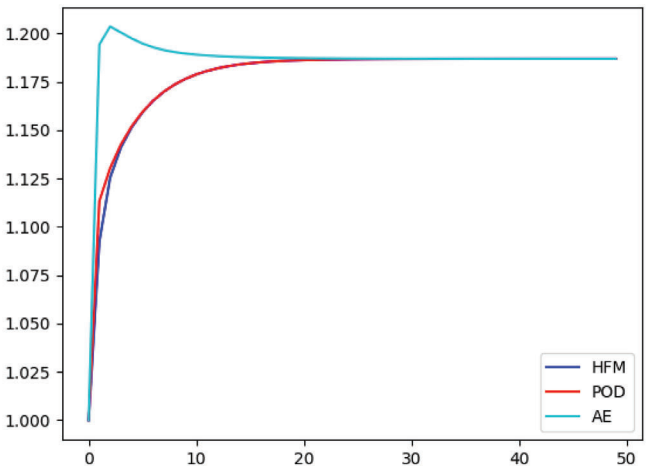
well with the high-fidelity model solution. Both ROMs also converge to the k_{eff} predicted by the high-fidelity model as shown in Figure 6(B). The rates of convergence are similar, however the AE-based ROM converges from above, whereas the high-fidelity model and POD-based ROM converge from below.

Figure 7 shows the flux profile and the convergence of k_{eff} for the material parameters $r_1 = 0.458$ and $r_2 = 0.932$. The solution corresponding to these values has not been seen by the reduced-order models (i.e., is unseen). From Figure 7(A), it can be seen that the flux profiles of both POD-based and AE-based ROMs demonstrate excellent agreement with the high-fidelity model. The convergence of k_{eff} for both ROMs is also in good agreement with the value predicted by the high-fidelity model, see Figure 7(B), and the rates of convergence are also similar.

To further analyze the results, the 200 solutions from both the seen and unseen data sets for the POD-based and AE-based ROMs were compared with solutions from the high-fidelity model using the error measures for the scalar flux and k_{eff} as given by Equations (38) and (39), respectively. Figure 8(A) shows a histogram of the error in the scalar flux profile and Figure 8(B) shows a histogram of the error in k_{eff}. Excluding outliers, most of the flux solutions have an absolute

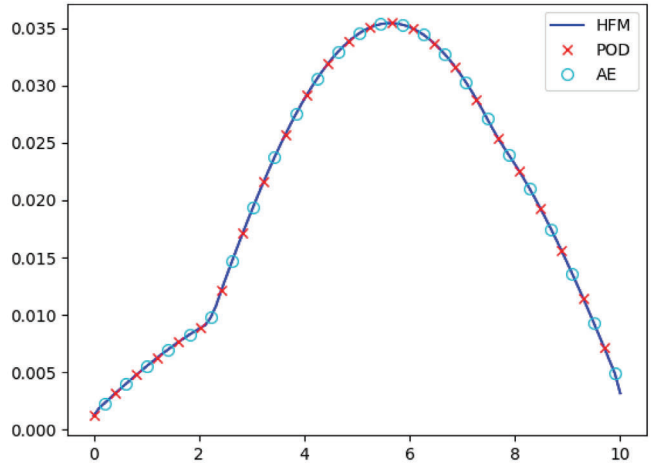


(A) Scalar flux (neutrons cm^{-2} s^{-1}) vs x (cm), HFM solution in blue, POD-based ROM in red crosses and AE-based ROM in light blue circles.

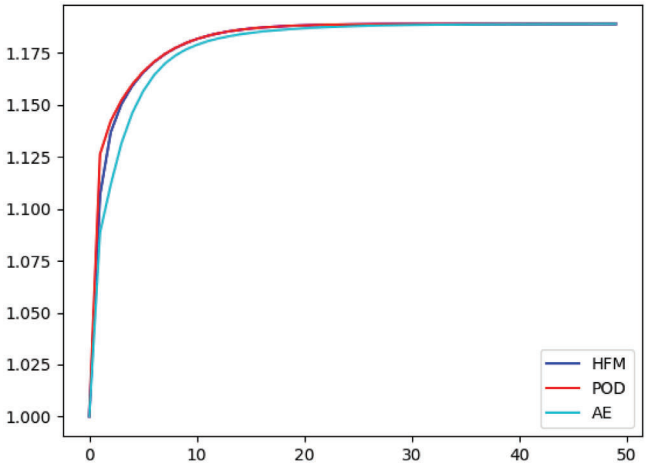


(B) k_{eff} vs iteration number, convergence of the POD-based ROM (red) and and AE-based ROM (light blue) to the converged HFM value (blue).

FIGURE 6 Scalar flux and convergence of k_{eff} for a seen problem with material parameters (cross-sections) determined by $r_1 = 0.957$ and $r_2 = 0.115$, comparing the POD-based and AE-based ROMs with the high-fidelity model. Compression from 100 variables to 10



(A) Scalar flux (neutrons cm^{-2} s^{-1}) vs x (cm), HFM solution in blue, POD-based ROM in red crosses and AE-based ROM in light blue circles.

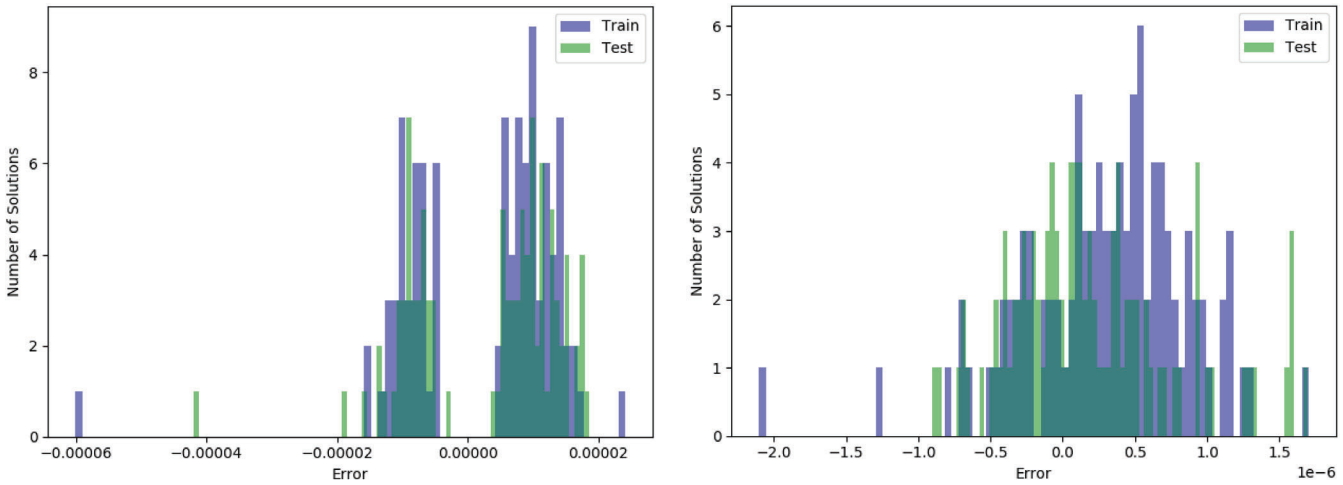


(B) k_{eff} vs iteration number, convergence of the POD-based ROM (red) and and AE-based ROM (light blue) to the converged HFM value (blue).

FIGURE 7 Scalar flux and convergence of k_{eff} for a unseen problem with material parameters (cross-sections) determined by $r_1 = 0.458$ and $r_2 = 0.932$, comparing the POD-based and AE-based ROMs with the high-fidelity model. Compression from 100 variables to 10

error of 0.002% or less, and the absolute error in the k_{eff} values is less than 1.5×10^{-6}. The plots demonstrate both the accuracy of the POD-based model (for seen data) and its ability to predict (for unseen parameters). The ranges of values of the errors for both the seen and unseen parameters are similar. Histograms were also produced to study the error of the AE-based ROM. Figure 9(A) shows the error in flux and Figure 9(B) shows the error in k_{eff} based on the error measures defined in Equations (38) and (39). Excluding outliers, most of the flux solutions have an absolute error of 0.2% or less, and the absolute error in the k_{eff} values is less than 10^{-4}. These values are higher than for the POD-based reduced-order model, and there is also a larger spread of errors for the AE-based model than for the POD-based method, although the ranges of the values of the errors are also similar for both seen and unseen data sets.

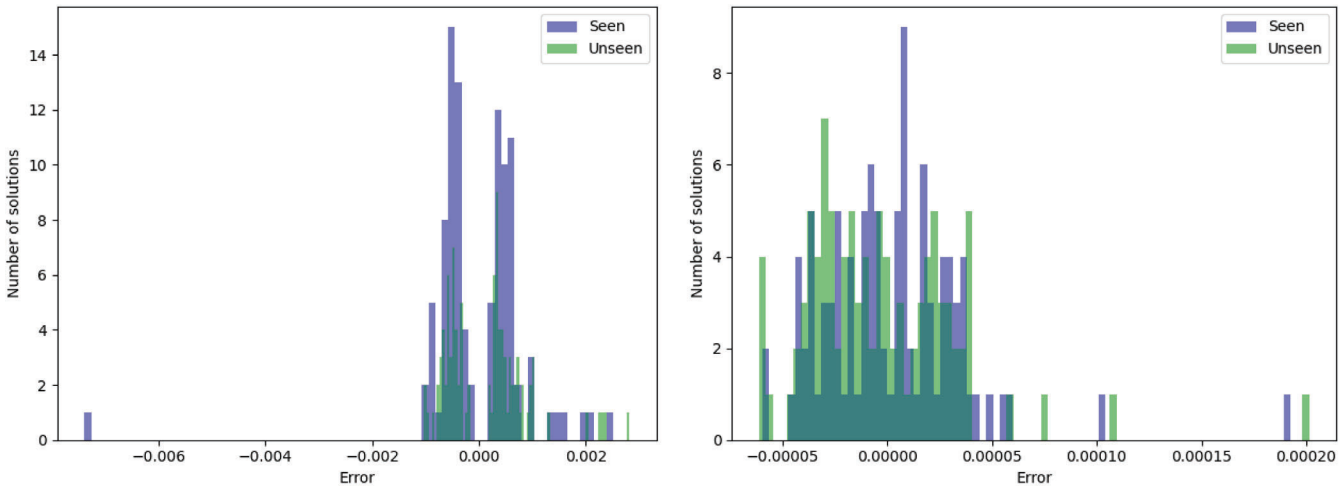
Table 2 displays the average maximum errors in the compression, scalar flux and k_{eff} for POD-based and AE-based ROMs as defined in Equations (40), (41), and (42). The reconstruction or compression error (for one solution) is given by Equation (35) for POD and Equation (36) for the autoencoder. The error in the scalar flux solution will contain this error and other numerical errors made when solving the reduced generalized eigenvalue problem. Therefore we expect that the compression error would be lower than the flux error, borne out by the results shown in Table 2. Also noteworthy is



(A) Histogram of the error in the flux profile.

(B) Histogram of the error in k_{eff} (scale of x-axis , 10^{-6}).

FIGURE 8 Errors in the seen (blue) and unseen (green) results from the POD-based reduced-order model for the 1D test case compressing from 100 variables to 10



(A) Histogram of the error in the flux profile.

(B) Histogram of the error in k_{eff}.

FIGURE 9 Errors in the seen and unseen results from the autoencoder-based reduced-order model for the 1D test case compressing from 100 variables to 10

the similarity between the range of errors for the seen and unseen data. This may mean that the snapshots represent well the behavior that was tested by the 100 unseen problems. For k_{eff}, the error for the unseen results of the POD-based ROM is very slightly lower that than of the seen results. This is surprising but could be explained by several outliers and the averaging of the maximum errors.

For this test case, with two control-rod regions, one could postulate that having a low-dimensional space of dimension 2 should be enough to describe the behavior of the system. The next section compares a projection-based ROM with two basis functions and an autoencoder-based ROM with two latent variables.

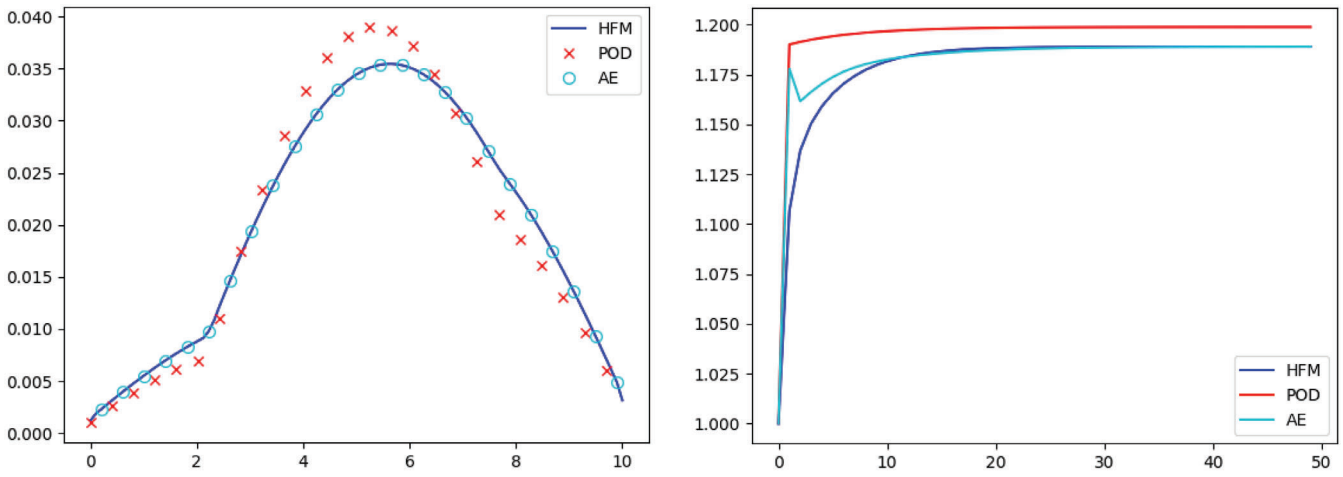
5.3.5 | Reduction to two variables

In order to see how well the models perform when reducing to a low-dimensional space of dimension 2, a POD-based ROM and an AE-based ROM were constructed using just two POD basis functions for POD, and two latent variables for the autoencoder. Figure 10 shows the flux profile and convergence of k_{eff} for both POD-based and AE-based ROMs compared to the high-fidelity model for an unseen set of parameters, $r_1 = 0.458$ and $r_2 = 0.932$. Here it can be seen clearly that the AE-based ROM agrees more closely with the high-fidelity model than the POD-based ROM. Also, the k_{eff} value of the AE-based ROM converges to the value predicted by the high-fidelity model, whereas the POD-based model does not converge to this value.

Table 3 shows the errors in flux and k_{eff} when the system is compressed from 100 variables to two variables. For this extreme case, it can be seen that the autoencoder performs better than POD, with flux errors that are two orders of magnitude lower than those for POD and k_{eff} errors that are more than two orders of magnitude lower than for POD. This suggests that the autoencoder can capture more of the features than POD with a given number of basis functions/latent variables.

TABLE 2 Average maximum errors for seen and unseen data using the POD-based reduced-order model and the autoencoder-based reduced-order model for the 1D test case compressing from 100 variables to 10

	POD-based results			Autoencoder-based results		
		Seen	Unseen		Seen	Unseen
Compression error	$\bar{e}^{\text{POD}}_{\max}(\boldsymbol{\phi}^{\text{HFM}})$	1.4344×10^{-6}	1.7004×10^{-6}	$\bar{e}^{\text{AE}}_{\max}(\boldsymbol{\phi}^{\text{HFM}})$	4.9554×10^{4}	5.4829×10^{-4}
Flux error	$\bar{e}_{\max}(\boldsymbol{\phi}^{\text{POD}})$	1.0270×10^{-5}	1.0401×10^{-5}	$\bar{e}_{\max}(\boldsymbol{\phi}^{\text{AE}})$	6.5843×10^{-4}	6.3030×10^{-4}
k_{eff} error	$\bar{e}(k^{\text{POD}}_{\text{eff}})$	5.2956×10^{-7}	4.4348×10^{-7}	$\bar{e}(k^{\text{AE}}_{\text{eff}})$	2.4642×10^{-5}	2.7540×10^{-5}



(A) Scalar flux (neutrons $\text{cm}^{-2}\ \text{s}^{-1}$) vs x (cm), HFM solution in blue, POD-based ROM in red crosses and AE-based ROM in light blue circles.

(B) k_{eff} vs iteration number, convergence of the POD-based ROM (red) and and AE-based ROM (light blue) to the converged HFM value (blue).

FIGURE 10 Scalar flux and convergence of k_{eff} for an unseen problem with material parameters (cross-sections) determined by $r_1 = 0.458$ and $r_2 = 0.932$, comparing the POD-based and AE-based ROMs with the high-fidelity model. Compression from 100 variables to 2

To investigate at which point the AE-based ROM becomes more accurate than the POD-based ROM, a number of ROMs were constructed with reduced space dimensions of 2, 3, and so on, up to 10. To generate the AE-based results, several networks were trained for each level of compression with various number of layers and neurons in each layer. All other parameters in training and the architecture were the same as for the autoencoder in Section 5.3.2. The network with the lowest validation loss was selected for each level of compression. Figure 11 shows the average maximum errors in the scalar flux and in k_{eff} for levels of compression between two and 10 variables. The accuracy of the POD-based ROM reduces as the number of POD coefficients decreases, and for four POD coefficients and fewer, the error is higher than that of the AE-based ROM. Once again, the errors for the seen and unseen data sets are very similar. It can also be observed that the errors from the AE-based ROMs do not vary much with the number of latent variables, so that compression to two variables incurs a similar error to the reduction to 10 variables.

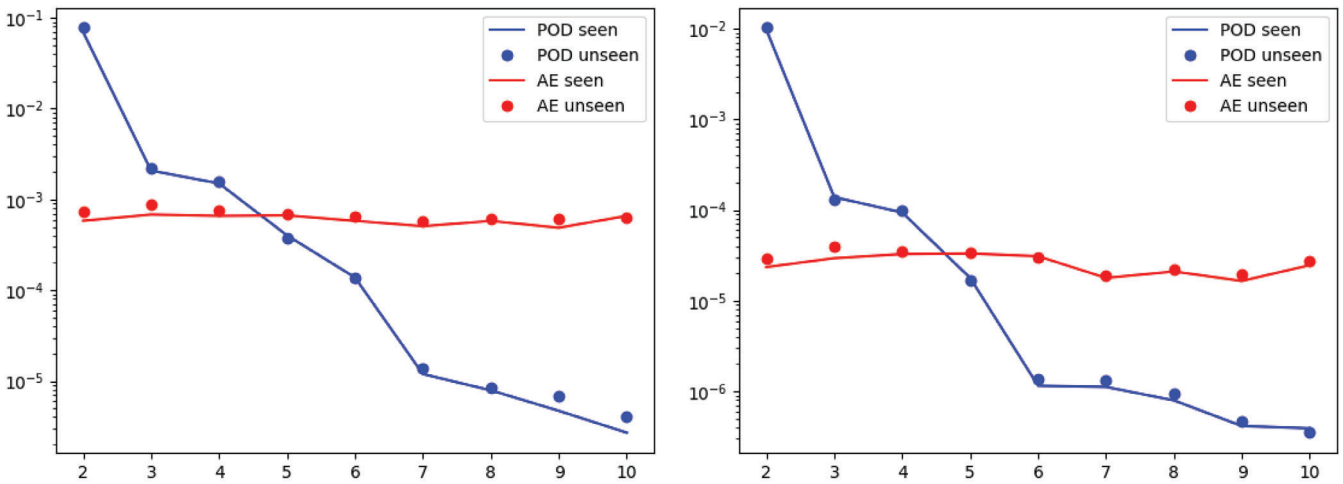
5.4 | 2D reactor core eigenvalue problem

The methods are now applied to a simple 2D reactor. The domain is square-shaped with sides of length 90 cm and there are four different materials in this mock reactor, see Figure 12. The domain is discretized with 90 cells in both directions, each cell is of length 1 cm, making 8100 cells. In addition to this, 364 cells were used to enforce the boundary conditions. Once again, this test case was used by Buchan et al.[47] who based the cross-sections on IAEA benchmarks. The same macroscopic cross-sections are used here and their values are given in Table 4.

In Figure 12 the blue region represents graphite, the red area contains the fuel and the four control-rod regions are white and labeled 1, 2, 3, and 4. Each control-rod region is 10 cm square with centers at (25,35), (55,35), (25,65), and (55,65), respectively. These are assumed to have a uniform distribution of the averaged properties of the mixture within the system based on a mixing coefficient for each region chosen. The method for determining the mixing coefficient is

TABLE 3 Average maximum errors for seen and unseen results using POD and a standard autoencoder to reduce from 100 variables to two variables

	POD-based results			Autoencoder-based results		
		Seen	Unseen		Seen	Unseen
Compression error	$\bar{e}^{\text{POD}}_{\max}(\boldsymbol{\phi}^{\text{HFM}})$	6.3880×10^{-2}	7.1400×10^{-2}	$\bar{e}^{\text{AE}}_{\max}(\boldsymbol{\phi}^{\text{HFM}})$	5.0533×10^{-4}	6.221×10^{-4}
Flux error	$\bar{e}_{\max}(\boldsymbol{\phi}^{\text{POD}})$	6.9400×10^{-2}	7.7714×10^{-2}	$\bar{e}_{\max}(\boldsymbol{\phi}^{\text{AE}})$	5.8252×10^{-4}	7.3427×10^{-4}
k_{eff} error	$\bar{e}(k^{\text{POD}}_{\text{eff}})$	9.8282×10^{-3}	1.0248×10^{-2}	$\bar{e}(k^{\text{AE}}_{\text{eff}})$	2.3471×10^{-5}	2.9228×10^{-5}



(A) Scalar flux error against number of POD basis functions.

(B) k_{eff} error against number of latent variables.

FIGURE 11 Average maximum errors associated with POD-based and AE-based ROMs for compression to between two and 10 POD coefficients (POD-based method) or latent variables (AE-based method) for the 1D test case

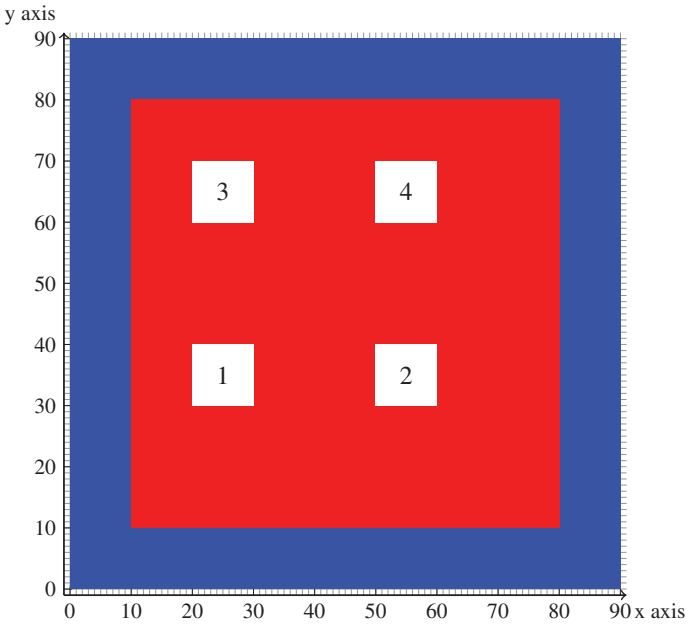


FIGURE 12 Geometry of a 2D reactor core with graphite (blue), fuel (red), and four control-rod regions labeled 1, 2, 3, 4 (white)

TABLE 4 Macroscopic cross-sections for the materials in the 2D reactor core (units cm^{-1})

	Σ_a	Σ_s	Σ_f
Fuel	0.075	0.53	0.79
Water	0.01	0.89	0
Control rod	0.38	0.2	0
Graphite	0.15	0.5	0

described in Section 5.2, however, instead of mixing fuel and control rod materials in the control-rod regions as done for the 1D test case, here, water and control rod are mixed, with more weight given to the water in order to prevent domination of the strong absorption of the control rods.

For this system 800 high-fidelity model solutions are generated, each with four distinct values of r, one for each control-rod region. The method for generating the solutions is similar to Section 5.3. Of the 800 solutions, 400 are used during the offline phase as the snapshots (or seen data). The other 400 results make up the unseen data. These high-fidelity model solutions are used to construct a POD-based ROM and two autoencoder-based ROMs: the first uses a fully connected autoencoder, the second uses the hybrid SVD-autoencoder. The number of degrees of freedom of the high-fidelity model solutions is 8100, and, for all ROMs, the number of POD coefficients or latent variables is chosen as 4, as this is perhaps the minimum number required to represent the system.

5.4.1 | POD-based ROM

For the POD-based ROM, an SVD is used to compress the snapshots. From a possible 400 basis functions, four are retained, which corresponds to capturing 99.878% of the information contained in the snapshots. In the online stage the parameter values are chosen and matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the high-fidelity model are formed. The first four basis functions are used to create $\boldsymbol{R}$ and Equation (20) is formed. This is solved with the power method as described in Algorithm 3 and Section 3.1.

5.4.2 | AE-based ROM

In the autoencoder-based ROM, the SVD is replaced by an autoencoder, which, for this case, compresses from 8100 variables to four latent variables. For this problem a range of networks were trained varying the numbers of layers

(up to 16 layers), neurons in each layer (to a maximum of 100), batch size (10, 50, 400) and minimum learning rate (from 10^{-4} to 10^{-7}). The network with the lowest validation loss was used.

The best performing autoencoder comprised of 16 fully connected layers with the following number of neurons in each layer:

$$8100 \to 100 \to 86 \to 72 \to 58 \to 44 \to 30 \to 16 \to 4 \to 16 \to 30 \to 44 \to 58 \to 72 \to 86 \to 100 \to 8100.$$

The activation function for every layer is the exponential linear unit; the optimizer used is "Nadam"; the initial learning rate is 10^{-3} and the minimum learning rate is 10^{-6}; and the loss function is defined to be the mean squared reconstruction error. The snapshots are scaled between 0 and 1, and the network was trained using mini-batch gradient descent with a batch size of 50. To prevent overfitting an early stopping parameter was used to terminate training if the validation loss did not decrease over 200 successive epochs. In this example, training was stopped due to this at 24,000 epochs. Note here, the large reduction in neurons between the input and the first hidden layer. Although needed to reduce the number of layers which would otherwise be too high, this is expected to cause some difficulties with the accuracy of the network's predictions.

5.4.3 | Combined singular value decomposition and autoencoder approach

As an alternative to having a large difference between the number of neurons in the input and the first hidden layer, a combination of an SVD and an autoencoder is explored. First the SVD is applied to the snapshots, after which, 100 of the 400 possible basis functions are retained. The snapshots are projected onto the basis functions (see Equation (18)) and the resulting snapshot coefficients are used to train the autoencoder. As with the autoencoder of the previous section, the network has four latent variables. To find the optimal architecture, different networks were trained with different numbers of layers and different numbers of neurons in each layer. The network with the lowest validation loss was then selected.

The best performing autoencoder consisted of 14 fully connected layers with the number of neurons in each layer being:

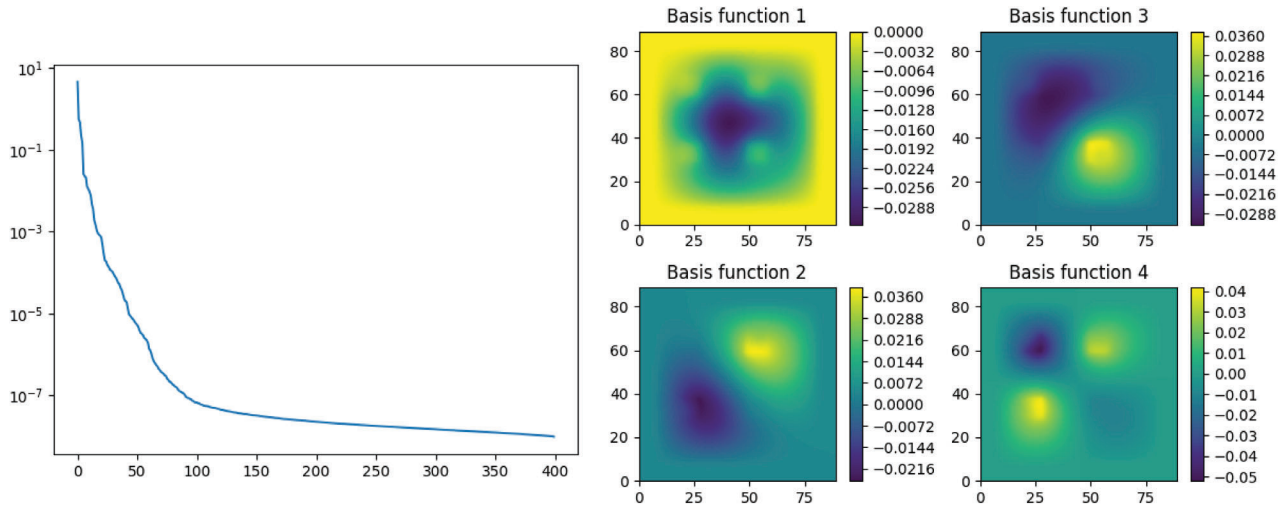
$$100 \to 100 \to 100 \to 45 \to 20 \to 9 \to 4 \to 9 \to 20 \to 45 \to 100 \to 100 \to 100.$$

The activation function for every layer was the exponential linear unit; the optimizer used was "Nadam" and the loss function was defined to be the mean squared reconstruction error. The snapshots were not scaled, and the network was trained using mini-batch gradient descent with a batch size of 50. The number of epochs was 30,000. We note that, instead of combining an SVD with the autoencoder, two additional linear layers could be added to the autoencoder, one after the input and the other before the output layer, effectively replacing the SVD. However, extra layers would make the training process more difficult and, in any case, the using the SVD is a computationally efficient way of calculating this particular transformation.

5.4.4 | Compression using POD, an autoencoder, and the SVD-AE

Here we compare POD, an autoencoder, and the SVD-AE as methods of compression. Figure 13(A) shows the singular values of the snapshots matrix, of which the first 100 decrease rapidly, the remaining values reach a plateau. The first four POD basis functions can be seen in Figure 13(B). Figure 14 shows four basis functions of the nonlinear map associated with the autoencoder, linearized at convergence, for both the seen and unseen sets of parameter values. These basis functions are columns of the $\boldsymbol{C}$ matrix. Figure 15 shows four basis functions of the nonlinear map associated with the SVD-autoencoder, linearized at the point of convergence, for both the seen and unseen sets of parameter values. From all three compression methods, it can be seen the basis functions capture variation associated with the control-rod regions.

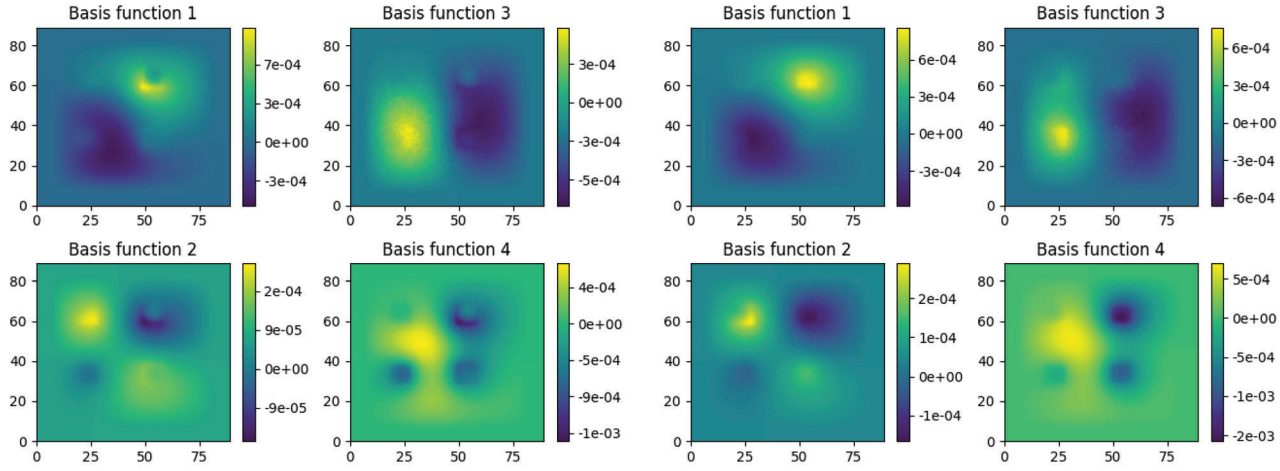
Figure 16 shows the compression errors for the POD method (see Equation (35)), the autoencoder (see Equation (36)), and the hybrid SVD-AE (see Equation (37)). For POD, this represents the error which occurs when truncating the POD basis; for the AE, this represents the difference between each high-fidelity model solution and the corresponding output of the autoencoder after training; and for the SVD-AE, this represents the difference between the high-fidelity model solution



(A) Singular value against singular value index.

(B) First four basis functions.

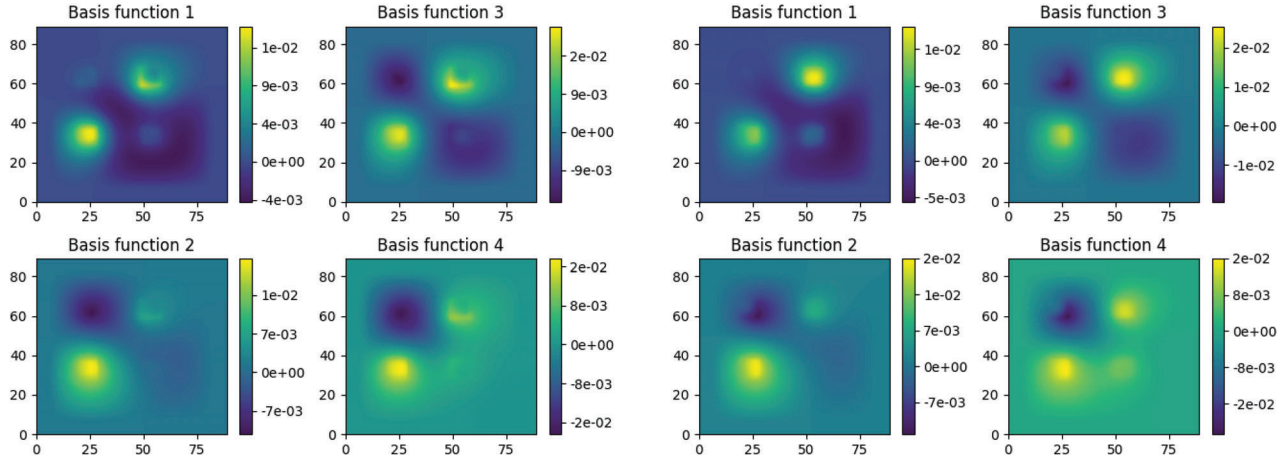
FIGURE 13 Singular values and four basis functions for the 2D test case



(A) For the seen problem with parameters $r_1 = 0{:}9782$, $r_2 = 0{:}9891$, $r_3 = 0{:}7006$ and $r_4 = 0{:}8316$.

(B) For the unseen problem with parameters $r_1 = 0{:}9452$, $r_2 = 0{:}8647$, $r_3 = 0{:}9996$ and $r_4 = 0{:}9776$.

FIGURE 14 The four basis functions from the linearized $\boldsymbol{C}$ matrix generated by the autoencoder at the point of convergence



(A) For the seen problem with parameters $r_1 = 0{:}9782$, $r_2 = 0{:}9891$, $r_3 = 0{:}7006$ and $r_4 = 0{:}8316$.

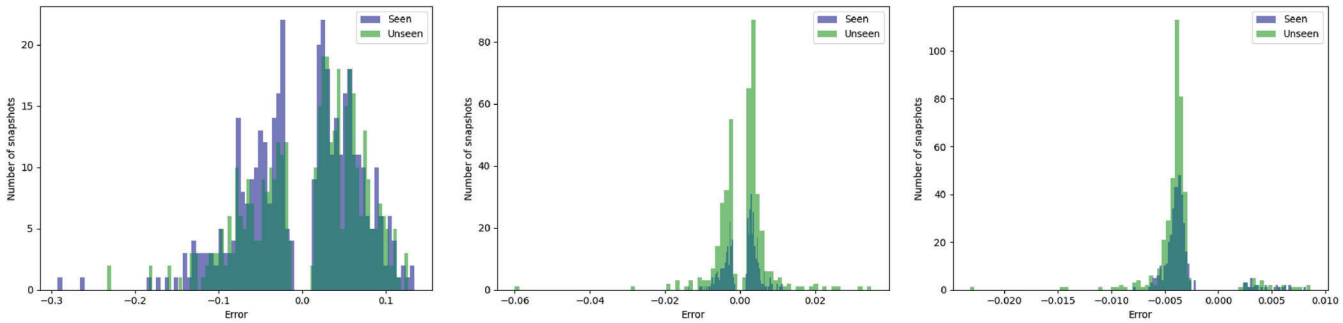
(B) For the unseen problem with parameters $r_1 = 0{:}9452$, $r_2 = 0{:}8647$, $r_3 = 0{:}9996$ and $r_4 = 0{:}9776$.

FIGURE 15 The four basis functions from the linearized $\boldsymbol{C}$ matrix generated by the SVD-autoencoder at the point of convergence

and the result of transforming these solutions by the linear map of the SVD, passing them through the autoencoder and applying the inverse linear map of the SVD. In all three cases, the ranges of errors for seen and unseen data sets is similar, indicating for the two AE-based methods, that overfitting has been avoided. Excluding outliers, compressing with POD introduces, at worst, an error of about 15%, whereas the corresponding errors for the AE and SVD-AE are much lower at 2% and 1%, respectively. The errors for POD also exhibit a much wider distribution than for the two AE-based methods, which have relatively narrow distributions with much higher peak values. Figure 17 shows box and whiskers plots of the values of the four latent variables over the seen data for the autoencoder (Figure 17(A)) and the SVD-AE (Figure 17(B)). For the autoencoder, two of the variables take a limited range of values, whereas, for the SVD-AE, all four variables have a similar range of values. Intuition indicates that at least four variables are required to describe this problem, so the fact that the SVD-AE fully exploits all four variables suggests that it will perform better than the AE-based method.

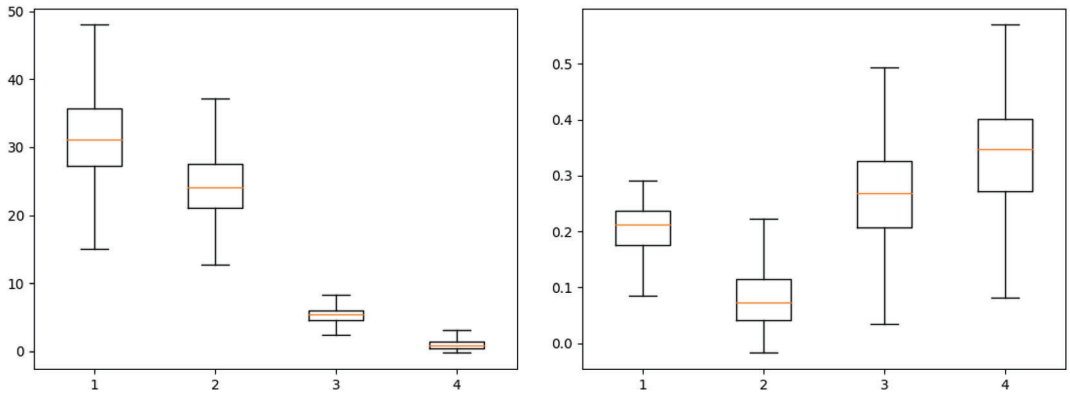
5.4.5 | Comparing results from the ROMs based on POD, AE, and SVD-AE

First, the performance of the ROMs is tested on a seen set of parameter values: $r_1 = 0.9782$, $r_2 = 0.9891$, $r_3 = 0.7006$, and $r_4 = 0.8316$. Figure 18 shows the scalar flux across the domain for the high-fidelity model and the three reduced-order models using POD, AE and SVD-AE for compression. All ROMs show excellent agreement with the high-fidelity model. Figure 19 shows the pointwise error for the three reduced-order models. The error is concentrated around the control rods. For the SVD-AE-based ROM, the error appears to be much more localized than for the POD-based and AE-based models. The maximum error for the SVD-AE-based model is also lower than for the other two models. Figure 20 shows



(A) Compression error of POD. (B) Compression error of the AE. (C) Compression error of the SVD-AE.

FIGURE 16 Compression error in the flux profile for seen (blue) and unseen (green) data sets for the 2D test case (note the different x-axis scales)



(A) Values taken by the latent variables of the AE. (B) Values taken by the latent variables of the SVD-AE.

FIGURE 17 Box and whisker plots of the ranges of values taken by the four latent variables of the AE and SVD-AE over the seen data set. The orange line shows the median value, the box shows the upper and lower quartiles, and the whiskers show the minimum and maximum values

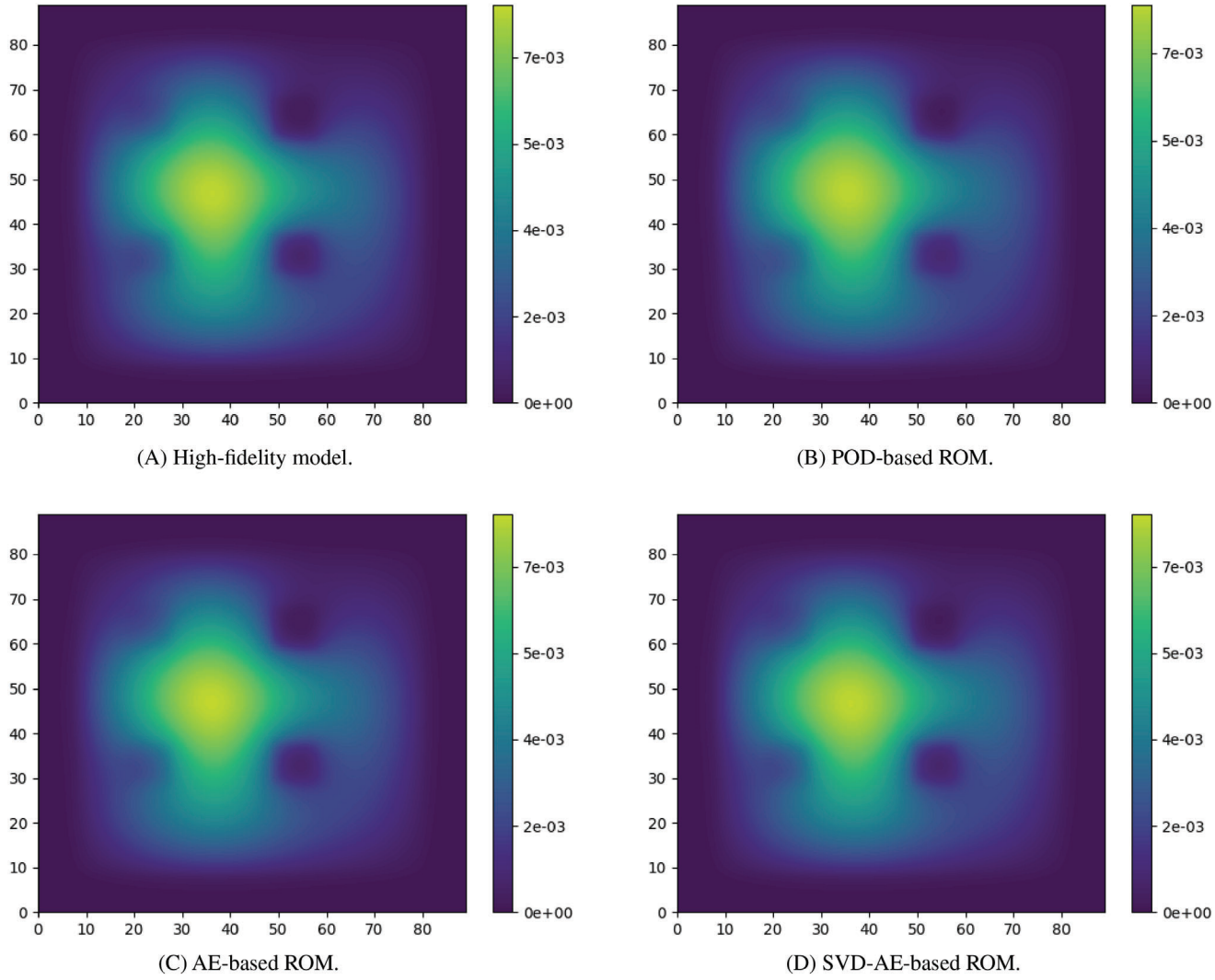


FIGURE 18 Scalar flux (neutrons·cm^{-2} ·s^{-1}) across the reactor for a seen problem with material properties $r_1 = 0.9782$, $r_2 = 0.9891$, $r_3 = 0.7006$, and $r_4 = 0.8316$ for the 2D test case

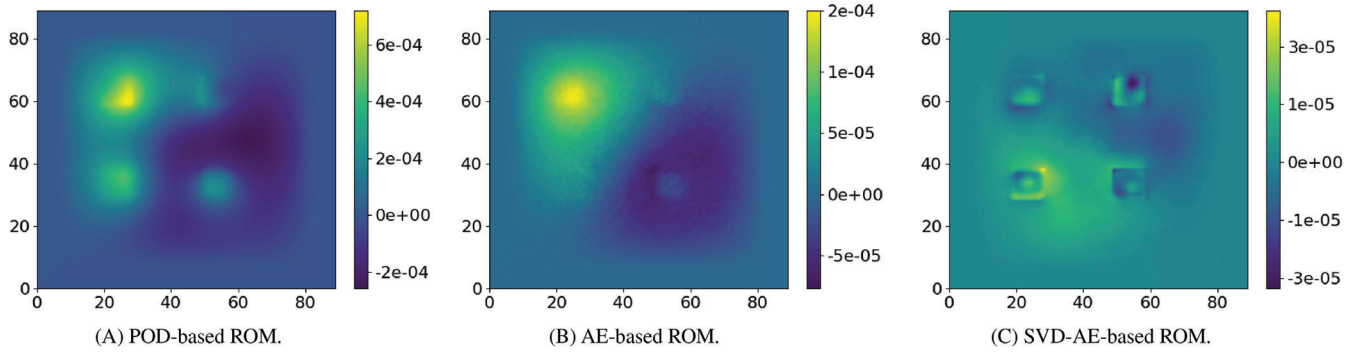


FIGURE 19 Pointwise error of the scalar flux across the reactor for a seen problem with material properties $r_1 = 0.9782$, $r_2 = 0.9891$, $r_3 = 0.7006$, and $r_4 = 0.8316$ for the 2D test case

the convergence of k_{eff} for the three ROMs compared with that of the high-fidelity model. The three ROMs converge faster than the high-fidelity model. The POD-based model has the highest error, finally achieving a value within 0.0035 of that of the high-fidelity model. Both of the autoencoder-based models converge to a value extremely close to that of the high-fidelity model.

The reduced-order models are now tested on a set of unseen values $r_1 = 0.9452$, $r_2 = 0.8647$, $r_3 = 0.9996$, and $r_4 = 0.9776$. The scalar fluxes across the reactor are shown in Figure 21 and, once again, all the ROMs show excellent agreement with the high-fidelity model. The pointwise errors are shown in Figure 22, which demonstrate that most of the error is concentrated around the control-rod regions. The error for the SVD-AE-based model is, once again, very localized. The

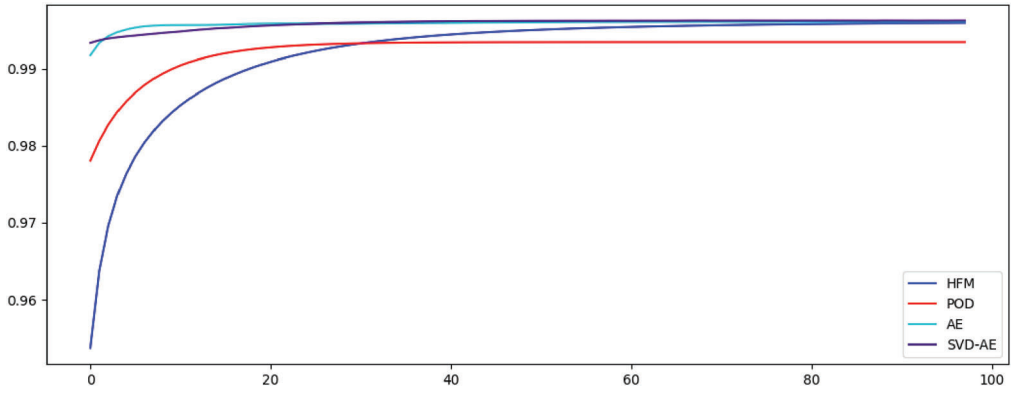


FIGURE 20 Convergence of k_{eff} for the three ROMs for the seen problem with material properties $r_1 = 0.9782$, $r_2 = 0.9891$, $r_3 = 0.7006$ and $r_4 = 0.8316$. Blue represents the high-fidelity model, red the POD-based ROM, cyan the AE-based ROM, and indigo the SVD-AE-based ROM

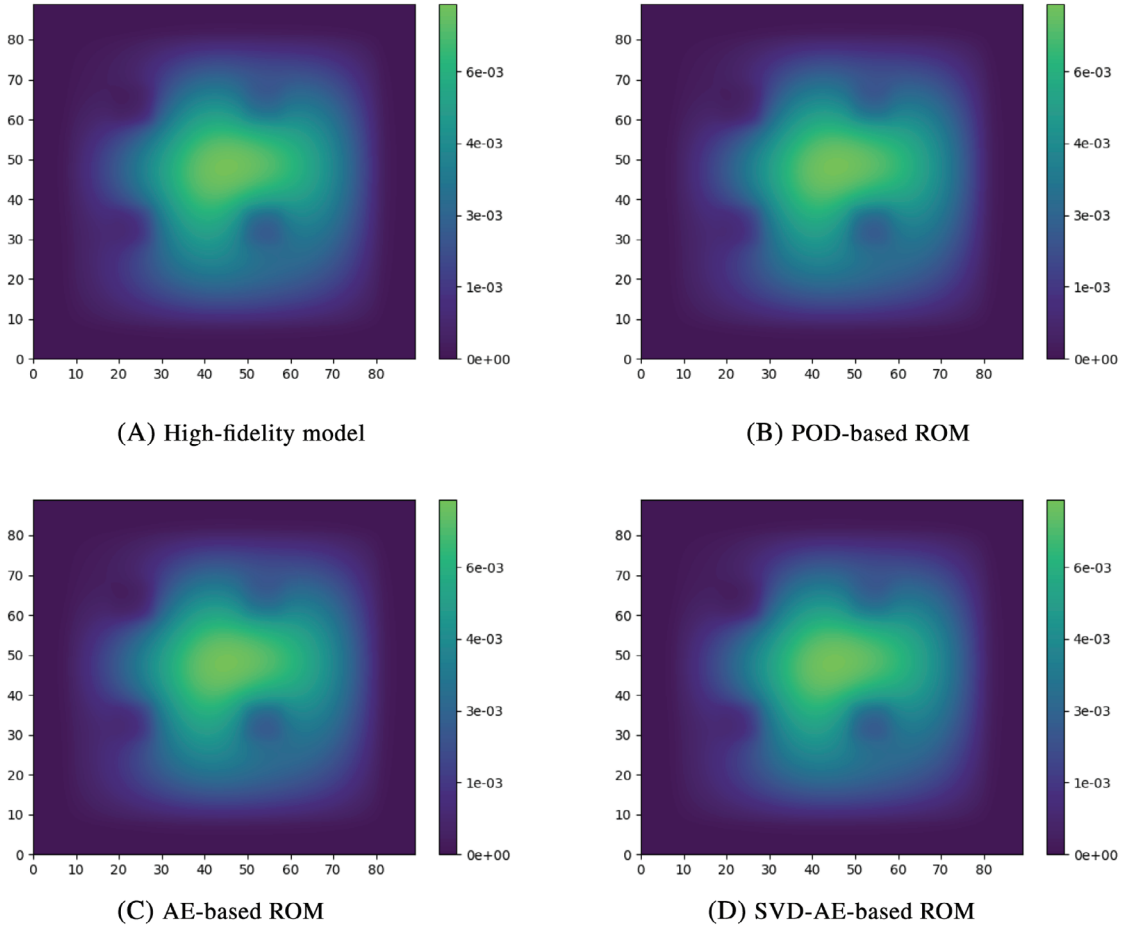
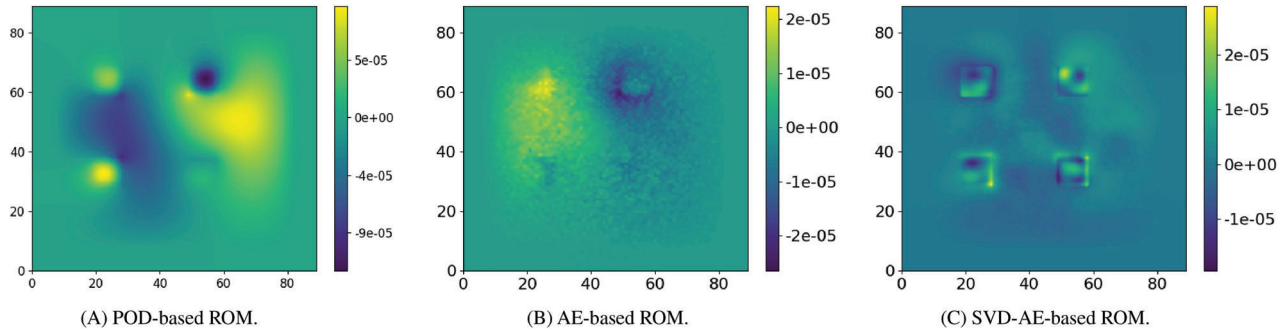


FIGURE 21 Scalar flux (neutrons·cm^{-2} ·s^{-1}) across the reactor for an unseen problem with material properties $r_1 = 0.9452$, $r_2 = 0.8647$, $r_3 = 0.9996$, and $r_4 = 0.9776$ for the 2D test case

error for the AE-based ROM has a similar magnitude to the other ROMs but is more diffuse and more oscillatory. All ROMs have a similar maximum error for this unseen set of parameter values. Figure 23 shows that the k_{eff} values of the three ROMs converge closely, and at a similar rate, to the k_{eff} value predicted by the high-fidelity model.

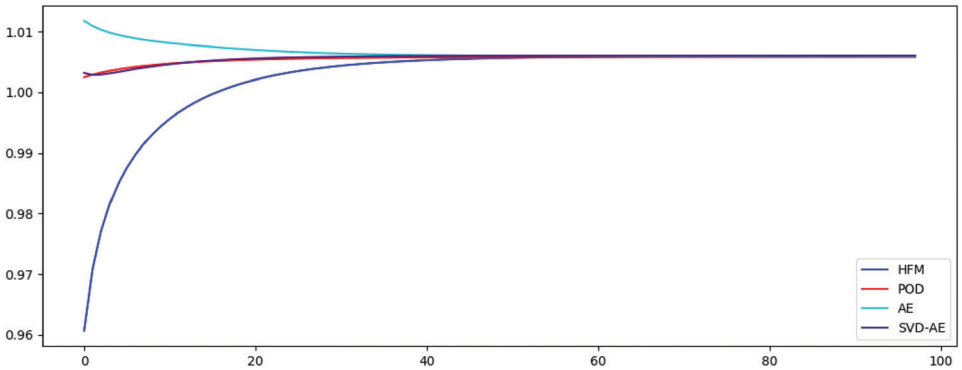
Figure 24 shows histogram plots of the errors from the POD-based ROM for both the 400 snapshots and the 400 unseen HFM solutions. Figure 24(A) shows the error in flux profile calculated using Equation (38), and Figure 24(B) shows the error in k_{eff} calculated using Equation (39). Setting aside some outliers, the error in flux profile has a range similar to the range of errors seen in the truncation of the POD basis, implying that the main contribution to the error for the POD-based

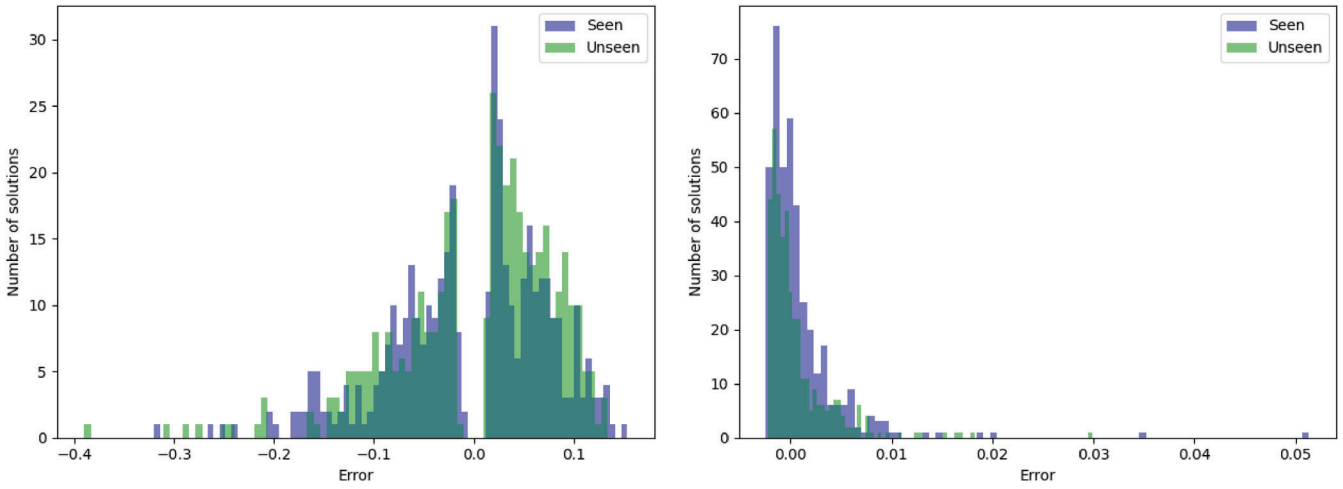


(A) POD-based ROM. (B) AE-based ROM. (C) SVD-AE-based ROM.

FIGURE 22 Pointwise error of the scalar flux across the reactor for an unseen problem with material properties $r_1 = 0.9452$, $r_2 = 0.8647$, $r_3 = 0.9996$, and $r_4 = 0.9776$ for the 2D test case

FIGURE 23 Convergence of k_{eff} for the three ROMs for the unseen problem with material properties $r_1 = 0.9452$, $r_2 = 0.8647$, $r_3 = 0.9996$, and $r_4 = 0.9776$. Blue represents the high-fidelity model, red the POD-based ROM, cyan the AE-based ROM, and indigo the SVD-AE-based ROM





(A) Histogram of the error in the flux profile. (B) Histogram of the error in k_{eff}.

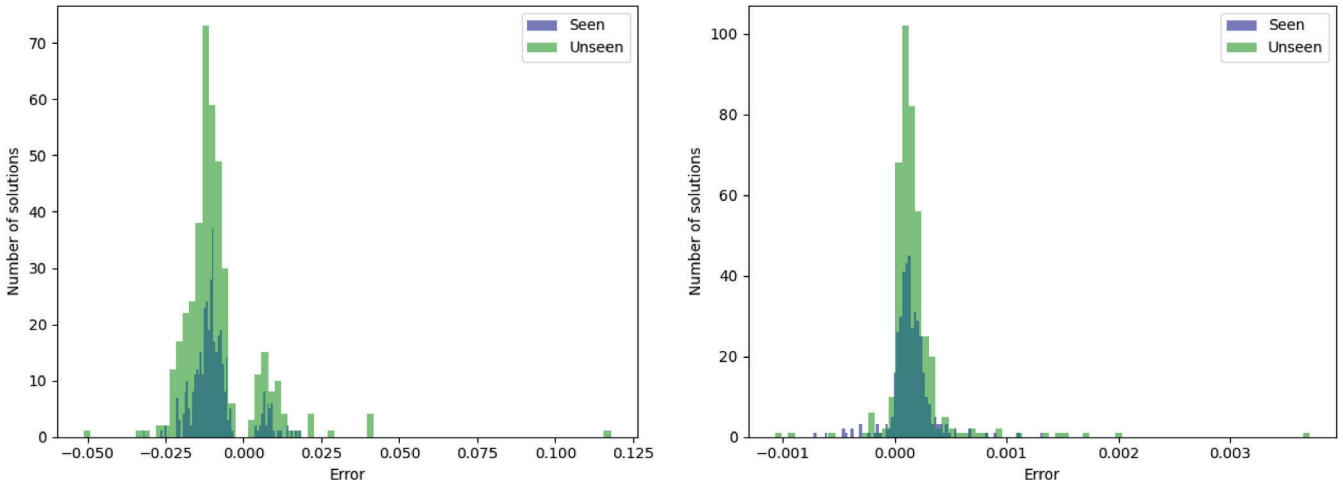
FIGURE 24 Error in the seen (blue) and the unseen (green) results for the POD-based ROM applied to the 2D test case

ROM is due to the truncation of the POD basis. The absolute maximum error in the fluxes is over 20% and for k_{eff} is about 2%.

Figure 25 shows histograms of the errors for the AE-based ROM for both seen and unseen data. Both sets of data exhibit similar ranges of errors as calculated by Equation (38) for the error in scalar flux and Equation (39) for the error in k_{eff}, demonstrating that overfitting has been avoided. In comparison with results from the POD-based ROM, the AE-based model shows lower errors (the absolute maximum error in flux being about 2.5% as opposed to 20% for POD). The errors in k_{eff} are an order of magnitude lower those of POD.

Figure 26 shows histograms of the errors for the SVD-AE-based ROM for both seen and unseen data. Both sets of data exhibit similar ranges of errors indicating that the SVD-autoencoder is not overfitting. The errors are much lower than those seen in the POD-based ROM and a little lower than those of the AE-based ROM. The largest flux error is half that of the AE-based model, and the largest k_{eff} is a quarter of that of the AE-based model.

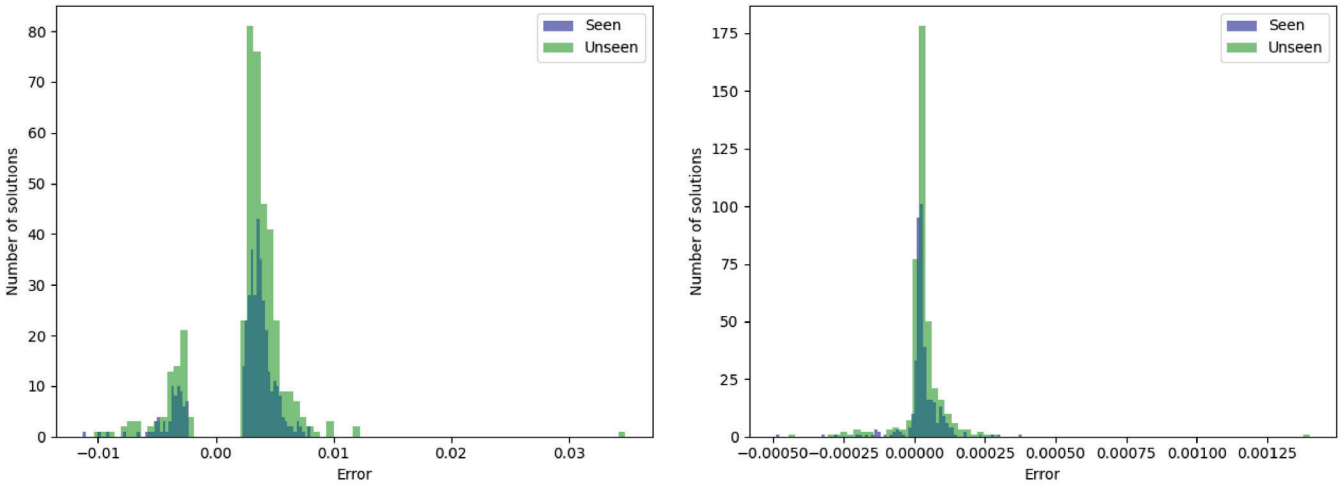
Table 5 shows the average maximum errors, defined in Equations (40), (41), and (42), for all three ROMs applied to the 2D problem. It can be seen that both autoencoder-based ROMs have lower compression errors than POD with both networks having similar compression errors. Both AE-based methods perform better than the POD-based method but the SVD-AE-based method outperforms the pure AE method. The AE-based method achieves k_{eff} errors that are an order of magnitude better than the POD-based method and the SVD-AE has k_{eff} errors lower than the AE-based method.



(A) Histogram of the error in the flux profile.

(B) Histogram of the error in k_{eff}.

FIGURE 25 Error in the seen (blue) and the unseen (green) results for the autoencoder-based ROM applied to the 2D test case



(A) Histogram of the error in the flux profile.

(B) Histogram of the error in k_{eff}.

FIGURE 26 Error in the seen (blue) and the unseen (green) results for the SVD-AE-based ROM applied to the 2D test case

TABLE 5 Average maximum errors for seen and unseen data for the POD-based ROM, AE-based ROM, and SVD-AE-based ROM for the 2D test case

	POD		AE		SVD-AE	
	Seen	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
Compression error	5.7397×10^{-2}	5.7756×10^{-2}	3.9602×10^{-3}	5.0071×10^{-3}	4.0081×10^{-3}	4.2799×10^{-3}
Flux error	6.4829×10^{-2}	6.5539×10^{-2}	1.1773×10^{-2}	1.3256×10^{-2}	4.5225×10^{-3}	4.7314×10^{-3}
k_{eff} error	2.1332×10^{-3}	1.9947×10^{-3}	1.7579×10^{-4}	2.0100×10^{-4}	4.5224×10^{-5}	4.5921×10^{-5}

Note: The number of compressed variables for all methods is four.

6 | CONCLUSIONS

By adopting an autoencoder for dimensionality reduction or compression, a novel projection-based reduced-order model for solving eigenvalue problems is developed. Two simple reactor problems are used as test cases to compare a number of autoencoder-based reduced order models with a reduced-order model (ROM) based on proper orthogonal decomposition (POD). One of the autoencoder-based ROMs uses a novel combination of singular value decomposition (SVD) with an autoencoder.

POD and singular value decomposition (SVD) are often used to obtain basis functions of a low-dimensional or "reduced" space onto which the high-fidelity model is projected. This embedding of a high-dimensional space into a low-dimensional space is linear for POD, whereas the embedding found by an autoencoder (AE) is, in general, nonlinear. This nonlinearity means that an autoencoder can provide a more accurate representation of data which could be important for larger, more complex problems.

For the 1D test case, when using a reduced space of dimension two, the AE-based reduced-order model performs better than the POD-based model, with lower errors in compression and for prediction of the scalar flux profiles. When predicting, the errors in the scalar flux and k_{eff} for the AE-based ROM are both two orders of magnitude lower than those of the POD-based ROM. This supports the claim that each latent variable of the autoencoder can capture more information than each POD basis function.

For the 2D test case, the SVD-AE-based ROM has errors in the flux solution that are one order of magnitude lower than those of the POD-based ROM, and errors in k_{eff} that are two orders of magnitude lower. The errors of the AE-based ROM lie in between those of the POD-based ROM and the SVD-AE-based ROM. Combining an SVD with an autoencoder reduces the length of the inputs to the autoencoder, which, in turn, reduces the complexity of the network, making training faster and more effective, and overfitting less likely.

Although two relatively simple test cases are presented in this article, our future work will involve increasingly complicated (i.e., more realistic) assembly or reactor configurations, which will benefit from the increased modeling capabilities that autoencoders provide. For this eigenvalue problem, it can be seen that there is a clear benefit to using autoencoders over the standard POD approach. For minimal numbers of basis functions, autoencoders can be used to achieve a level of compression that cannot be matched in accuracy by POD.

ACKNOWLEDGMENTS

The authors would like to acknowledge the EPSRC Centre for Doctoral Training in Nuclear Energy (Imperial College, Cambridge University and the Open University, EP/L015900/1) and the following EPSRC grants: MUFFINS (EP/P033180/1); MAGIC (EP/N010221/1), INHALE (EP/T003189/1), and PREMIERE (EP/T000414/1). The authors would also like to thank the reviewers for their insightful comments, which have contributed to improving the paper.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author, Claire E. Heaney, upon reasonable request.

ORCID

Claire E. Heaney https://orcid.org/0000-0002-6555-1423

REFERENCES

1. Schilders WHA, van der Vorst HA, Rommes J. *Model Order Reduction: Theory, Research Aspects and Applications*. The European Consortium for Mathematics in Industry. Vol 13. Berlin/Heidelberg, Germany: Springer; 2008.
2. Zahr MJ, Farhat C. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *Int J Numer Methods Eng*. 2015;102(5):1111-1135.
3. Ballarin F, Faggiano E, Ippolito S, et al. Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a POD-Galerkin method and a vascular shape parametrization. *J Comput Phys*. 2016;315:609-628.
4. Kerfriden P, Passieux JC, Bordas SPA. Local/global model order reduction strategy for the simulation of quasi-brittle fracture. *Int J Numer Methods Eng*. 2012;89(2):154-179.
5. Agathos K, Bordas SPA, Chatzi E. Parametrized reduced order modeling for cracked solids. *Int J Numer Methods Eng*. 2020;121(20):4537-4565.
6. Jansen JD, Durlofsky LJ. Use of reduced-order models in well control optimization. *Optim Eng*. 2017;18(1):105-132.
7. Xiao D, Fang F, Pain CC, Navon IM, Salinas P, Wang Z. Non-intrusive model reduction for a 3D unstructured mesh control volume finite element reservoir model and its application to fluvial channels. *Int J Oil Gas Coal Technol*. 2018;19:316-338.
8. Hoang HC, Fu Y, Song JH. An hp-proper orthogonal decomposition-moving least squares approach for molecular dynamics simulation. *Comput Methods Appl Mech Eng*. 2016;298:548-575.
9. Chinesta F, Leygue A, Bordeu F, et al. PGD-based computational vademecum for efficient design, optimization and control. *Arch Comput Methods Eng*. 2013;20(1):31-59.
10. Niroomandi S, Alfaro I, Gonzalez D, Cueto E, Chinesta F. Real-time simulation of surgery by reduced order modelling and X-FEM techniques. *Int J Numer Methods Biomed Eng*. 2011;28(5):574-588.
11. Aguado JV, Huerta A, Chinesta F, Cueto E. Real-time monitoring of thermal processes by reduced-order modeling. *Int J Numer Methods Eng*. 2015;102(5):991-1017.
12. Taira K, Brunton SL, Dawson STM, et al. Modal analysis of fluid flows: an overview. *AIAA J*. 2017;55(12):4013-4041.
13. Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. *Annu Rev Fluid Mech*. 2020;52:477-508.
14. Lumley JL. The structure of inhomogeneous turbulent flows. In: Yaglom AM, Tatarski VI, eds. *Atmospheric Turbulence and Radio Propagation*. Moscow, Russia: Nauka; 1967:166-178.
15. Sirovich L. Turbulence and the dynamics of coherent structures Parts I, II, III. *Q Appl Math*. 1987;45(3):561-590.
16. Benner P, Gugercin S, Willcox K. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev*. 2015;57(4):483-531.
17. Ryckelynck D. A priori hypereduction method: an adaptive approach. *J Comput Phys*. 2005;2:346-366.
18. Fritzen F, Haasdonk B, Ryckelynck D, Schöps S. An algorithmic comparison of the hyper-reduction and the discrete empirical interpolation method for a nonlinear thermal problem. *Math Comput Appl*. 2018;23(1):8.
19. Chaturantabut S, Sorensen DC. Nonlinear model reduction via discrete empirical interpolation. *SIAM J Sci Comput*. 2010;32(5):2737-2764.
20. Breitkopf P, Lepot I, Sainvitu C, Villon P. Multi-fidelity POD surrogate-assisted optimization: concept and aero-design study. *Struct Multidiscip Optim*. 2017;56:1387-1412.
21. Kaiser E, Noack BR, Cordier L, et al. Cluster-based reduced-order modelling of a mixing layer. *J Fluid Mech*. 2014;754:365-414.
22. Guo M, Hesthaven JS. Data-driven reduced order modeling for time-dependent problems. *Comput Methods Appl Mech Eng*. 2019;345:75-99.
23. Swischuk R, Mainini L, Peherstorfer B, Willcox K. Projection-based model reduction: formulations for physics-based machine learning. *Comput Fluids*. 2019;179:704-717.
24. Polifke W. Black-box system identification for reduced order model construction. *Ann Nucl Energy*. 2014;67:109-128.
25. Wang Z, Xiao D, Fang F, Govindan R, Pain CC, Guo Y. Model identification of reduced order fluid dynamics systems using deep learning. *Int J Numer Methods Fluids*. 2018;86(4):255-268.
26. Bui-Thanh T, Damodaran M, Willcox K. Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics. Paper presented at: Proceedings of the 21st AIAA Applied Aerodynamics Conference, Orlando, Florida; 2003.
27. Gonzalez FJ, Balajewicz M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems; 2018. arXiv:1808.01346 [math.DS].
28. Wiewel S, Becher M, Thuerey N. Latent space physics: towards learning the temporal evolution of fluid flow. *Comput Graph Forum*. 2019;38(2):71-82.
29. Hesthaven JS, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J Comput Phys*. 2018;363:55-78.
30. Xiao D, Heaney CE, Mottet L, et al. A reduced order model for turbulent flows in the urban environment using machine learning. *Build Environ*. 2019;148:323-337.
31. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science*. 2006;313(5786):504-507.
32. Wang J, He H, Prokhorov DV. A folded neural network autoencoder for dimensionality reduction. *Proc Comput Sci*. 2012;13:120-127.
33. Mao X, Shen C, Yang Y-B. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Lee DD, Sugiyama M, UV Luxburg, Guyon I, Garnett R, eds. *Advances in Neural Information Processing Systems* 29, NY: Curran Associates, Inc 2016 (pp. 2802–2810).
34. Pascal V, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J Mach Learn Res*. 2010;11:3371-3408.
35. Oja E. A simplified neuron model as a principal component analyzer. *J Math Biol*. 1982;15:267-273.

36. Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biol Cybern*. 1988;59:291-294.
37. Baldi P, Hornik K. Neural networks and principal component analysis: learning from examples without local minima. *Neural Netw*. 1989;2:53-58.
38. Takane Y. Nonlinear PCA by neural network models. Paper presented at: Proceedings of the 63rd Annual Meeting of the Japan Statistical Society; 1998; 258-260; Japan.
39. Milano M, Koumoutsakos P. Neural network modeling for near wall turbulent flow. *J Comput Phys*. 2002;182:1-26.
40. Hartman D, Mestha LK. A deep learning framework for model reduction of dynamical systems. Paper presented at: Proceedings of the 2017 IEEE Conference on Control Technology and Applications (CCTA), Maui, HI; 2017:1917-1922.
41. Kashima K. Nonlinear model reduction by deep autoencoder of noise response data. Paper presented at: Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas; 2016:5750-5755.
42. Lee K, Carlberg KT. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J Comput Phys*. 2020;404:108973.
43. Ahmed SE, Rahman SM, San O, Rasheed A, Navon IM. Memory embedded non-intrusive reduced order modeling of non-ergodic flows. *Phys Fluids*. 2019;31:126602.
44. Palmtag S, Clarno K, Davidson G, et al. Coupled neutronics and thermal-hydraulic solution of a full core PWR using VERA-CS. In: Chiba G, Yamamoto A, eds. *Proceedings of the International Topical Meeting on Advances in Reactor Physics, PHYSOR*. Vancouver, Canada; 2014.
45. Slaybaugh RN, Ramirez-Zweiger M, Tara P, Hamilton S, Evans TM. Eigenvalue solvers for modeling nuclear reactors on leadership class machines. *Nucl Sci Eng*. 2018;190(1):31-44.
46. Chunyu Z, Gong C. Fast solution of neutron diffusion problem by reduced basis finite element method. *Ann Nucl Energy*. 2018;111:702-708.
47. Buchan AG, Pain CC, Fang F, Navon IM. A POD reduced-order model for eigenvalue problems with application to reactor physics. *Int J Numer Methods Eng*. 2013;95(12):1011-1032.
48. Heaney CE, Buchan AG, Pain CC, Jewer S. Reactor simulators and reduced order modelling. *Nucl Future*. May/June 2018;49-54.
49. German P, Ragusa JC. Reduced-order modeling of parameterized multi-group diffusion *k*-eigenvalue problems. *Ann Nucl Energy*. 2019;134:144-157.
50. Sartori A, Cammi A, Luzzi L, Rozza G. Reduced basis approaches in time-dependent non-coercive settings for modelling the movement of nuclear reactor control rods. *Commun Comput Phys*. 2016;20(1):23-59.
51. Sartori A, Cammi A, Luzzi L, Rozza G. A multi-physics reduced order model for the analysis of lead fast reactor single channel. *Ann Nucl Energy*. 2016;87:198-208.
52. Buchan AG, Dargaville S, Pain CC. A combined immersed body and adaptive mesh method for simulating neutron transport within complex structures. *Ann Nucl Energy*. 2019;134:88-100.
53. Morel JE, McGhee JM. A self-adjoint angular flux equation. *Nucl Sci Eng*. 1999;132(3):312-325.
54. Golub GH, Loan CF. *Matrix Computations*. Baltimore and London: John Hopkins University Press; 1996.
55. Prince ZM, Ragusa JC. Application of proper generalized decomposition to multigroup neutron diffusion eigenvalue calculations. *Prog Nucl Energy*. 2020;121:103232.
56. Pinski G, Narin F. Citation influence for journal aggregates of scientific publications: theory, with application to the literature of physics. *Inf Process Manag*. 1976;12(5):297-312.
57. Gupta P, Goel A, Lin J, Sharma A, Wang D, Zadeh R. WTF: the who to follow service at Twitter. Paper presented at: Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil; 2013:505-514.
58. Scheben F. *Iterative Methods for Criticality Computations in Neutron Transport Theory* [PhD thesis] School of Mathematical Sciences, University of Bath; 2011.
59. Holmes P, Lumley JL, Berkooz G, Rowley CW. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge, MA: Cambridge University Press; 2012.
60. Eckart C, Young G. The approximation of one matrix by another of lower rank. *Psychometrika*. 1936;1:211-218.
61. Baur Ulrike, Benner Peter, Haasdonk Bernard, Himpe Christian, Martini Immanuel, Ohlberger Mario. Chapter 9: comparison of methods for parametric model order reduction of time-dependent problems:377–407. SIAM 2017.
62. Wirtz D, Sorensen DC, Haasdonk B. A posteriori error estimation for DEIM reduced nonlinear dynamical systems. *SIAM J Sci Comput*. 2014;36(2):A311-A338.
63. Kuhn M, Johnson K. *Applied Predictive Modeling*. 2nd ed. New York, NY: Springer; 2018.
64. Golub GH, Vorst HA. Eigenvalue computation in the 20th century. *J Comput Appl Math*. 2000;123(1):35-65.
65. Chollet F. *Keras;* 2015. https://keras.io.
66. Abadi M, Agarwal P, Brevdo E, et al. *TensorFlow: large-scale machine learning on heterogeneous systems*; 2015. www.tensorflow.org.
67. Clevert DA, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by Exponential Linear Units (ELUs); 2015. arXiv:1511.07289 [cs.LG].
68. Dozat T. Incorporating Nesterov momentum into Adam. Paper presented at: Proceedings of the International Conference on Learning Representations; 2016; San Juan, Puerto Rico.

How to cite this article: Phillips TRF, Heaney CE, Smith PN, Pain CC. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *Int J Numer Methods Eng*. 2021;122:3780–3811. https://doi.org/10.1002/nme.6681