AI-enhanced iterative solvers for accelerating the solution of large-scale parametrized systems

Stefanos Nikolopoulos | Ioannis Kalogeris | George Stavroulakis | Vissarion Papadopoulos

Institute of Structural Analysis and Antiseismic Research, School of Civil Engineering, National Technical University of Athens, Athens, Greece

Correspondence
Ioannis Kalogeris, Institute of Structural Analysis and Antiseismic Research, School of Civil Engineering, National Technical University of Athens, Athens, Greece.
Email: ikalog@mail.ntua.gr

Funding information
European High Performance Computing Joint Undertaking, Grant/Award Number: 956201-H2020-JTI-EuroHPC-2019-1

Summary
Recent advances in the field of machine learning open a new era in high performance computing for challenging computational science and engineering applications. In this framework, the use of advanced machine learning algorithms for the development of accurate and cost-efficient surrogate models of complex physical processes has already attracted major attention from scientists. However, despite their powerful approximation capabilities, surrogate model predictions are still far from being near to the 'exact' solution of the problem. To address this issue, the present work proposes the use of up-to-date machine learning tools in order to equip a new generation of iterative solvers of linear equation systems, capable of very efficiently solving large-scale parametrized problems at any desired level of accuracy. The proposed approach consists of the following two steps. At first, a reduced set of model evaluations is performed using a standard finite element methodology and the corresponding solutions are used to establish an approximate mapping from the problem’s parametric space to its solution space using a combination of deep feedforward neural networks and convolutional autoencoders. This mapping serves as a means of obtaining very accurate initial predictions of the system’s response to new query points at negligible computational cost. Subsequently, an iterative solver inspired by the Algebraic Multigrid method in combination with Proper Orthogonal Decomposition, termed POD-2G, is developed that successively refines the initial predictions of the surrogate model towards the exact solution. The application of POD-2G as a standalone solver or as preconditioner in the context of preconditioned conjugate gradient methods is demonstrated on several numerical examples of large scale systems, with the results indicating its strong superiority over conventional iterative solution schemes.

KEYWORDS
algebraic multigrid method, convolutional neural networks, iterative solvers, large-scale parametrized systems, preconditioned conjugate gradient method
1 | INTRODUCTION

In scientific computing, there is a constant need for solving larger and computationally more demanding problems with increased accuracy and improved numerical performance. This holds particularly true in multi-query scenarios such as optimization, uncertainty quantification, inverse problems and optimal control, where the problems under investigation need to be solved for numerous different parameter instances with high accuracy and efficiency. In this regard, constructing efficient numerical solvers for complex systems described by partial differential equations is crucial for many scientific disciplines. The preconditioned conjugate gradient method (PCG) \(^1\)-\(^4\) and the preconditioned generalised minimal residual method (PGMRES) \(^5\)-\(^7\) are amongst the most powerful and versatile approaches to treat such problems. In these methods, the choice of a suitable preconditioner plays a major role on the convergence and scalability of the solvers and notable examples include the incomplete Cholesky factorization \(^8\) and domain decomposition methods, \(^9\),\(^10\) such as the popular FETI methods \(^11\)-\(^13\) and the additive Schwarz methods. \(^14\),\(^15\) In a similar fashion, Algebraic and Geometric Multigrid (AMG, GMG, resp.) \(^16\) are equally well-established methods that are commonly employed for accelerating standard iterative solvers and may also service as highly efficient preconditioners for PCG \(^17\)-\(^19\) or PGMRES. \(^20\)-\(^22\)

Nevertheless, optimizing the aforementioned solvers so as to attain a uniformly fast convergence for multiple parameter instances, as required in multi-query problems, remains a challenging task to this day. To tackle this problem, several works suggest the use of interpolation methods tasked with constructing approximations of the system’s inverse operator for different parameter values, \(^23\)-\(^25\) which can then be used as preconditioners. Another approach can be found in Reference \(^26\), where primal and dual FETI decomposition methods with customized preconditioners are developed in order to accelerate the solution of stochastic problems in the context of Monte Carlo simulation, as well as intrusive Galerkin methods. Augmented Krylov Subspace methods showed great promise in handling sequences of linear systems, \(^27\) such as those arising in parametrized PDEs, however, the augmentation of the usual Krylov subspace with data from multiple previous solves led in certain cases to disproportional computational and memory requirements. To alleviate this cost, optimal truncation strategies have been proposed in References \(^28\),\(^29\), as well as deflation techniques. \(^30\)-\(^32\)

In recent days, the rapid advancements in the field of machine learning (ML) have offered researchers new tools to tackle challenging problems in multi-query scenarios. For instance, deep feedforward neural networks (FFNNs) have been successfully employed to construct response surfaces of quantities of interest in complex problems. \(^33\)-\(^37\) Convolutional neural networks (CNNs) in conjunction with FFNNs have been employed to predict the high-dimensional system response at different parameter instances. \(^38\)-\(^44\) In addition, recurrent neural networks demonstrated great potential in transient problems for propagating the state of the system forward in time without the need of solving systems of equations. \(^45\)-\(^47\) All these non-intrusive approaches utilize a reduced set of system responses to build an emulator of the system’s input-output relation for different parameter values. As such, they are particularly cheap to evaluate and can be very accurate in certain cases. However, these methods can be characterized as physics-agnostic in the sense that the derived solutions do not satisfy any physical laws. This problem is remedied to some extent from intrusive approaches based on reduced basis methods, such as Principal Orthogonal Decomposition (POD) \(^48\)-\(^50\) and proper Generalized Decomposition. \(^51\)-\(^53\) These methods rely on the premise that a small set of appropriately selected basis vectors suffices to construct a low-dimensional subspace of the system’s high-dimensional solution space and the projection of the governing equations to this subspace will come at minimum error. In addition, several recent works have investigated the combination of either linear or nonlinear dimensionality reduction algorithms and non-intrusive interpolation schemes to construct cheap emulators of complex systems. \(^54\)-\(^62\) Nevertheless, none of these surrogate modelling schemes can guarantee convergence to the exact solution of the problem.

In the effort to combine the best of two worlds, a newly emergent research direction is that of enhancing linear algebra solvers with machine-learning algorithms. For instance, POD has been successfully employed to truncate the augmented Krylov subspace and retain only the high-energy modes \(^65\) for efficiently solving sequences of linear systems of equations characterized by varying right-hand sides and symmetric-positive-definite matrices. In Reference \(^64\), neural networks were trained for predicting the geometric location of constraints in the context of domain decomposition methods, leading to enhanced algorithm robustness. Moreover, the close connection between multigrid methods and CNNs has been studied in several recent works, which managed to accelerate their convergence by providing data-driven smoothers, \(^65\) prolongation and restriction operators. \(^66\)

The present work aims at bridging the gap between machine learning and linear algebra algorithms for accelerating the solution of real-life computational mechanics problems in multi-query scenarios. To this end, a novel strategy is proposed to utilize ML tools in order to obtain system solutions within a prescribed accuracy threshold, with faster convergence rates than conventional solvers. The proposed approach consists of two steps. Initially, a reduced set
of model evaluations is performed and the corresponding solutions are used to establish an approximate mapping from the problem’s parametric space to its solution space using a combination of deep FFNNs and convolutional autoencoders (CAEs). This mapping serves a means of acquiring very accurate initial predictions of the system’s response to new query points at negligible computational cost. The error in these predictions, however, may or may not satisfy the prescribed accuracy threshold. Therefore, a second step is proposed herein, which further utilizes the knowledge from the already available system solutions, in order to construct a data-driven iterative solver. This solver is inspired by the idea of the Algebraic Multigrid method combined with Proper Orthogonal Decomposition, termed POD-2G, that successively refines the initial prediction of the surrogate model towards the exact system solutions with significantly faster convergence rates.

The paper is organised as follows. In Section 2 the basic principles of the PCG and AMG iterative solvers are illustrated. In Section 3, the elaborated methodology for developing an AI-enhanced linear algebra solver is presented. Section 4 presents a series of numerical examples that showcase the performance of the method compared to conventional iterative solvers. Section 5 summarizes the outcomes of this work and discusses possible extensions.

2 | ITERATIVE SOLVERS FOR FINITE ELEMENT SYSTEMS

This work focuses on linear elliptic PDEs defined on a domain \( \Omega \subset \mathbb{R}^{dim} \), \( dim = 1, 2, 3 \), which are parametrized by a vector of parameters \( \theta \in \Theta \), with \( \Theta \subset \mathbb{R}^{n} \) being the parameter space. The variational formulation of the PDE can be stated as: given \( \theta \in \Theta \), find the solution \( v = v(\theta) \) from the Hilbert space \( V = V(\Omega) \) such that

\[
\kappa(v, w; \theta) = f(w; \theta)
\]

for every \( w \in V(\Omega) \) with compact support in \( \Omega \). The Lax-Milgram lemma proves that Equation (1) has a unique solution for every \( \theta \), provided that the bilinear form \( \kappa(\cdot, \cdot; \theta) \) is continuous and coercive and \( f(\cdot; \theta) \) is a continuous one-form.

In practice, however, obtaining an exact solution \( v \) is not feasible for most applications of interest and instead, an approximate solution is sought using numerical discretization techniques, such as the finite element method (FEM) focused in this work. In the context of FEM, Equation (1) is converted into a \( d \times d \) linear system of equations parameterized by \( \theta \):

\[
K(\theta)u(\theta) = f(\theta)
\]

where \( K \in \mathbb{R}^{d \times d} \) is a real symmetric positive definite matrix, \( u \in \mathbb{R}^{d} \) is the solution vector containing the unknown nodal values at specific locations of the PDE’s domain, \( f \in \mathbb{R}^{d} \) is the equivalent force vector and \( d \) is the total number of unknowns in the system.

Solving such a linear system for a detailed discretization \( (d \gg 1) \) can be computationally intensive, particularly in multi-query problems that require numerous system evaluations for various instances of parameters \( \theta \). These include optimization problems, parameter inference, uncertainty propagation, sensitivity analysis, and so forth. Therefore, it becomes evident that efficient numerical solvers for linear systems of equations are of vital importance in the analysis of large scale real-world problems. In the remainder of this section, we revisit the basic ideas behind two of most efficient methods for solving such systems, namely, the PCG and the AMG methods.

2.1 | The preconditioned conjugate gradient method

As an iterative technique, the Conjugate Gradient method starts by an initial guess \( u^{(0)} \) for the system \( Ku = f \), with \( K \) being a symmetric positive definite matrix, and constructs a sequence of vectors \( \{u^{(1)}, u^{(2)}, \ldots\} \) that converge to true solution \( K^{-1}f \) in, at most, \( d \) iterations. In practice, however, the algorithm may terminate after \( k < d \) iterations, provided that the condition for the residual after the \( k \)-th iteration \( \|r^{(k)}\| = \|Ku^{(k)} - f\| \leq \delta \), with \( \delta \) being a prescribed accuracy threshold, is met.

Critically, the improvement in the approximations \( u^{(k)} \) is determined by the condition number \( c(K) \) of the system matrix \( K \); the larger \( c(K) \) is, the slower the improvement. A standard approach to enhance the convergence of the CG method is through preconditioning (PCG), namely the application of a linear transformation to the system with a matrix \( T \), called the preconditioner, in order to reduce the condition number of the problem. Thus, the original system \( Ku - f = 0 \) is replaced with \( T^{-1}(Ku - f) = 0 \), such that \( c(T^{-1}K) \) is smaller than \( c(K) \). The steps of the PCG algorithm are presented in Algorithm 1.

The choice of the preconditioner \( T \) in PCG plays a crucial role in the fast convergence of the algorithm. Some generic choices include the Jacobi (diagonal) preconditioner \( T = \text{diag}(K) \) and the incomplete Cholesky factorization \( T = \hat{L}\hat{L}^{T} \),
with \( \hat{L} \) being a sparse lower triangular matrix such that \( K \approx \hat{L} \hat{L}^T \). Another popular choice is the incomplete LU factorization \( T = \hat{L} \hat{U} \), with \( \hat{L} \) being a lower unitriangular matrix and \( \hat{U} \) an upper triangular, such that \( K \approx \hat{L} \hat{U} \). Moreover, multigrid methods such as the AMG, elaborated on the next section, apart from standalone iterative schemes, are also very effective as preconditioners to the CG method.

### 2.2 Algebraic multigrid method

AMG methods have been successfully applied to numerous problems in computational mechanics, including PDEs, sparse Markov chains and problems involving graph Laplacians (e.g., References 67-71). The key idea in AMG algorithms is to employ a hierarchy of progressively coarser approximations to the linear system under consideration in order to accelerate the convergence of classical simple and cheap iterative processes, such as the damped Jacobi or Gauss-Seidel. These methods, commonly referred to as relaxation or smoothing, are very efficient in eliminating the high-frequency error modes, but inefficient in resolving the low-energy modes. AMG overcomes this problem through the coarse-level correction, as elaborated below.

Let us consider the linear system of Equation (2), which describes the fine problem and let \( u^{(0)} \) be an initial solution to it. The two-level AMG defines a prolongation operator \( P \), which is a full-column rank matrix in \( \mathbb{R}^{d_{c} \times d} \), \( d_{c} < d \) and

### Algorithm 2. Two-level AMG algorithm

1: **Input**: \( K \in \mathbb{R}^{d \times d} \), rhs \( f \in \mathbb{R}^d \), prolongation operator \( P \in \mathbb{R}^{d_{c} \times d} \), a relaxation scheme denoted as \( \mathcal{G} \), residual tolerance \( \delta \) and an initial approximation \( u^{(0)} \)
2: set \( k = 0 \), initial residual \( r^{(0)} = f - Ku^{(0)} \)
3: **while** \( \| r^{(k)} \| < \delta \) **do**
4: Pre-relaxation: Perform \( r_1 \) iterations of the relaxation scheme on the current approximation and obtain \( u^{(k)} \) as:
   \[ u^{(k)} \leftarrow \mathcal{G}(u^{(k)}; r_1) \]
5: Update the residual: \( r^{(k)} = f - Ku^{(k)} \)
6: Restrict the residual to the coarser level and solve the coarse level system \( K_c e^{(k)}_c = P^T r^{(k)} \), where \( K_c = P^T K P \in \mathbb{R}^{d_{c} \times d_{c}} \)
7: Prolongate the coarse grid error \( e^{(k)}_c = P e^{(k)}_c \)
8: Correct the fine grid solution: \( u^{(k+1)} = u^{(k)} + e^{(k)}_c \)
9: Post-relaxation: Perform additional \( r_2 \) relaxation iterations and obtain \( u^{(k+1)} \leftarrow \mathcal{G}(u^{(k+1)}; r_2) \)
10: \( k = k + 1 \)
11: **end while**
Algorithm 3. AMG preconditioned PCG algorithm

1: Input: \( K \in \mathbb{R}^{d \times d} \), rhs \( f \in \mathbb{R}^d \), AMG scheme, residual tolerance \( \delta \) and an initial approximation \( u^{(0)} \)
2: set \( k = 0 \), initial residual \( r^{(0)} = f - K u^{(0)} \)
3: \( s_0 = \text{AMG}(u^{(0)}; K, r^{(0)}, r_1, r_2) \)
4: \( p_0 = s_0 \)
5: while \( \| r^{(k)} \| < \delta \) do
6: \( \alpha_k = \frac{(r^{(k)})^T s_k}{p_k^T K p_k} \)
7: \( u^{(k+1)} = u^{(k)} + \alpha_k p_k \)
8: \( r^{(k+1)} = r^{(k)} - \alpha_k K p_k \)
9: \( s_{k+1} = \text{AMG}(u^{(k+1)}; K, r^{(k+1)}, r_1, r_2) \)
10: \( \beta_k = \frac{(r^{(k)})^T s_{k+1}}{(r^{(k)})^T s_k} \)
11: \( p_{k+1} = s_{k+1} + \beta_k p_k \)
12: \( k = k + 1 \)
13: end while

a relaxation scheme such as the Gauss-Seidel (GS). Then, the two-level AMG algorithm consists in the steps shown in Algorithm 2:

In this algorithm, lines 4–10 describe what is known as a \( V \)-cycle, schematically depicted in Figure 1A. The multi-level version of the above algorithm is easily obtained as the result of recursively applying the two-level algorithm, as shown in Figure 1B for the three-level setting. The notation

\[
u^{(k+1)} = \text{AMG}(u^{(k)}, K, f, r_1, r_2)
\]

will be used to denote the application of one AMG cycle.

It can be proven that the error at the end of the \( k \)-th cycle of the two-level AMG can be computed as:

\[
\epsilon^{(k)} = M^e C M^e \epsilon^{(k-1)}
\]

with

\[
C = I - P (P^T K)^{-1} P^T K
\]
being the coarse grid correction and \( M \) a matrix that depends on the relaxation scheme. In this work, we chose the GS algorithm as the relaxation scheme, thus \( M = I - L^{-1}K \), where \( I \) is the \( d \times d \) identity matrix and \( L \) is the lower triangular part of \( K \) including the diagonal. As such, \( M^{r_2} \) in Equation (4) refers to the post-relaxation matrix after \( r_2 \) sweeps and \( M^{r_1} \) to the pre-relaxation after \( r_1 \) sweeps.

From Equation (4) it becomes evident that the matrix \( M^{r_2}CM^{r_1} \) determines the convergence behavior of the two-level cycle. The relaxation matrix \( M \) plays a role, however, in practice the selection of the prolongation operator \( P \) is the key to designing an efficient algorithm. In this regard, the most popular variations of AMG include the Ruge-Stüben method\(^{72}\) and the smoothed aggregation based (SA) AMG.\(^{73}\) Another important factor the affects the number of iterations in AMG to reach the prescribed threshold of accuracy, is the choice of the initial solution. In absence of other information, \( u^{(0)} = 0 \) is usually considered. Lastly, several works have investigated the use of the AMG as a preconditioner within the context of PCG, according to Algorithm 3.

3 | MACHINE LEARNING ACCELERATED ITERATIVE SOLVERS

3.1 | Problem statement

The aim in this section is to develop an efficient data-driven and AI-enhanced solver for the parametrized system of Equation (2), by combining linear algebra-based solvers with machine learning algorithms. More specifically, the idea proposed herein, is to utilize a reduced set of high-fidelity system solutions, obtained after solving Equation (2) for specified parameter instances, in two different yet complementary ways. First, a surrogate model will be established in the form of a ‘cheap-to-evaluate’ nonlinear mapping from the problem’s parameter space to its solution space using convolutional autoencoders (CAEs) and feedforward neural networks (FFNNs). Even though CAEs and FFNNs have been shown to produce astonishing results even for challenging applications,\(^{39,40,74}\) nevertheless, their black-box and physics-agnostic nature doesn’t provide any means to improve the solutions they produce. To combat this problem, POD is performed on this data set of solutions and an efficient iterative solver is developed based on the idea of AMG, where in this case the prolongation operator is substituted by the projection matrix to the POD reduced space, similar to the ones proposed in References\(^{75-77}\).

3.2 | Construction of surrogate model

A surrogate model is an imitation of the original high fidelity model and serves as a ‘cheap’ mapping from the parametric space \( \theta \in \mathbb{R}^n \) to the solution space \( u \in \mathbb{R}^d \). In general, it is built upon an initial dataset \( \{u_i\}_{i=1}^N \), which is created by solving the problem for a small, yet sufficient number, \( N \), of parameter values. Many surrogate modeling techniques have been introduced over the past years, including linear\(^{49,50,53}\) and nonlinear\(^{38,39,41,43,56,62}\) dimensionality reduction methods. In general, the selection of an appropriate surrogate modelling method is problem dependent, however, in this work, we will employ a surrogate modeling scheme based on CAEs and FFNNs, tailor-made for parametrized large-scale linear system arising through the numerical discretization of PDEs.

The strategy proposed herein consists of two phases, namely the offline and the online phase. The offline phase starts by generating a small yet sufficient number of parameter instances, \( \{\theta_i\}_{i=1}^N \), and feeding them as input to the governing PDE of the problem. Then, using a discretization method such as FEM, the set of high-fidelity solutions \( \{u_i\}_{i=1}^N \), with \( u_i \in \mathbb{R}^d \) corresponding to each parameter instance \( \theta_i \), are obtained (see Figure 2A). For detailed numerical models it is usually the case that \( d \gg 1 \) and this poses significant challenges when trying to establish a direct mapping from the problem’s parametric space to its solution space.

To tackle this, the next step in the procedure is to train a CAE that learns a low-dimensional representation of the high-dimensional solution vectors, as well as reconstruction map that converts them back to their original dimension. The first part is achieved through the CAE’s encoder, which is used to obtain low dimensional (latent) representations, \( z_i \in \mathbb{R}^l \) for each \( u_i \in \mathbb{R}^d \), such that \( l \ll d \). The second part is tasked to the decoder. The CAE is trained over the initial data set \( \{u_i\}_{i=1}^N \) such that it minimizes the objective function:

\[
\mathcal{L}_{CAE} = \frac{1}{N} \sum_{i=1}^{N} ||u_i - \hat{u}_i||_2^2
\]
FIGURE 2 A schematic representation of the offline phase for constructing the surrogate: (A) Generation of training samples by solving the high-fidelity numerical model for different parameter instances, (B) Learning a low-dimensional representation for the data set and a corresponding reconstruction map using a CAE, (C) Learning a mapping from the parameter space to the latent space using a FFNN.

where $\tilde{u}_i$ is the reconstructed input. After the training is completed, the latent space data set $\{z_i\}_{i=1}^N$ is obtained, as shown in Figure 2B.

The third step of the offline phase is the training of the FFNN, which is used to establish a nonlinear mapping from the parametric space of $\theta \in \mathbb{R}^n$ to the latent space $z \in \mathbb{R}^l$, as depicted in Figure 2C. Again, the aim of the training is the minimization of the loss function:

$$L_{FFNN} = \frac{1}{N} \sum_{i=1}^N ||z_i - \tilde{z}_i||_2^2$$ (7)

where $\tilde{z}_i = FFNN(\theta)$ is the network’s output.

An essential step in the proposed methodology is to efficiently span the problem’s parametric space, thus sophisticated sampling methods are better preferred, such as the Latin Hypercube method.78

Subsequently, the online phase utilizes the fully trained surrogate model, which is now capable of delivering accurate predictions of the system’s response for new parameter values $\theta_j$ as follows: First, the FFNN maps $\theta_j$ to its corresponding vector $z_j$ in the latent space and, then, the CAE’s decoder takes this as input in order to deliver the system solution $u_j$, that is:

$$u_j = decoder(FFNN(\theta_j)) := F_{\text{sur}}(\theta_j)$$ (8)

A schematic representation of the surrogate model is presented in Figure 3.

3.3 Multigrid-inspired POD solver

POD, also known as Principal Component Analysis, is a powerful and effective approach for data analysis and dimensionality reduction, aimed at indentifying low-order modes of a system. In conjunction with the Galerkin projection procedure it is commonly utilized as an efficient method to reduce the dimensionality of large linear
The theory and application of POD is covered in many publications, however, to keep this paper as self-contained as possible the POD procedure used within this framework is summarized below. Let us denote with $U \in \mathbb{R}^{d \times N}$ the matrix consisting of $N$ solution vectors $[u_1, ..., u_N]$ for different parameter values $\{\theta_i\}_{i=1}^N$ and with $R = UU^T \in \mathbb{R}^{d \times d}$ the correlation matrix. Then POD consists in the following steps.

1. Compute the eigenvalues and eigenvectors of $R$ that satisfy $R\Phi = \Phi \Lambda$. This step can be very demanding when $d \gg 1$, however, in practice $N \ll d$ and since the eigenvalues of the matrix $U^T U$ are also eigenvalues of the correlation matrix $R$, it is computationally more convenient to solve instead the eigenvalue problem $U^T U \Psi = \Psi \Lambda$. Then, the eigenvectors $\Phi$ and $\Psi$ are linked according to the formula:

$$\Phi = U \Psi \Lambda^{-1/2}$$

2. Form the reduced basis $\Phi_r$, by retaining only the $r$ first columns of $\Phi$, corresponding to the largest eigenvalues.
3. Under the assumption that each solution to Equation (2) can be approximated as:

$$u \equiv \Phi_r u_r$$

with $u_r \in \mathbb{R}^r$ being the unknown coefficients of the projection on the truncated POD basis, then the reduced-order linear system becomes:

$$Ku = f$$
$$\Phi_r^T K \Phi_r u_r = \Phi_r^T f$$
$$K_r u_r = f_r$$

Solving Equation (11) for $u_r$ is significantly easier since $K_r \in \mathbb{R}^{r \times r}$, with $r$ small.
4. Retrieve the solution of the original problem:

$$u = \Phi_r u_r$$

Based on the above, a similarity between the two-level AMG method and POD can be observed, under the identification of $\Phi_r$ as the prolongation operator and $\Phi_r^T$ the corresponding restriction. Then, Algorithm 2 remains practically the same, as shown in Algorithm 4.
Algorithm 4. POD-2G algorithm

1: **Input:** $K \in \mathbb{R}^{d \times d}$, rhs $f \in \mathbb{R}^d$, prolongation operator $\Phi \in \mathbb{R}^{d \times r}$, a relaxation scheme denoted as $\mathcal{G}$, residual tolerance $\delta$ and an initial approximation $u^{(0)}$
2: set $k = 0$, initial residual $r^{(0)} = f - Ku^{(0)}$
3: **while** $\|r^{(k)}\| < \delta$ **do**
4: Pre-relaxation: Perform $r_1$ iterations of the relaxation scheme on the current approximation and obtain $u^{(k)}$ as:
   $u^{(k)} = \mathcal{G}(u^{(k)}; r_1)$
5: Update the residual: $r^{(k)} = f - K u^{(k)}$
6: Restrict the residual to the coarser level and solve the coarse level system $K_c e_c^{(k)} = \Phi^T r^{(k)}$, where $K_c = \Phi^T K \Phi \in \mathbb{R}^{d_c \times d_c}$
7: Prolongate the coarse grid error $e^{(k)}_c = \Phi e^{(k)}_c$
8: Correct the fine grid solution: $u^{(k+1)} = u^{(k)} + e^{(k)}$
9: Post-relaxation: Perform additional $r_2$ relaxation iterations and obtain $u^{(k+1)} = \mathcal{G}(u^{(k+1)}; r_2)$
10: $k = k + 1$
11: **end while**

In this case, the error of the scheme is given by the formula

$$e^{(k)} = M^{r_2} \left( I - \Phi \Phi^T (K \Phi \Phi^T)^{-1} \Phi^T K \right) M^{r_1} e^{(k-1)}$$  \hspace{1cm} (13)

3.4 Proposed data-driven framework for parameterized linear systems

The final step is to combine the surrogate model of section 3.2 and the multigrid-inspired POD solver of the previous section into a unified methodological framework for solving efficiently large-scale parametrized linear systems. In particular, an initial data set of system solutions $\{u_i\}_{i=1}^{N}$ is performed for specified instances of the parameter vector $\{\theta_i\}_{i=1}^{N}$. Then, these solution vectors are utilized as training data for the CAE and FFNN and the surrogate model is established. The error between the exact solution and the surrogate’s prediction for a given $\theta$ can be given as:

$$e^{sur} = u^* - \mathcal{T}^{sur}(\theta)$$  \hspace{1cm} (14)

Despite one’s best efforts, however, $\|e^{sur}\| \neq 0$ and the surrogate’s predictions will not satisfy exactly Equation (2). At this point, instead of simply performing iterations of PCG or AMG to improve the surrogate’s predictions, we propose to further utilize the knowledge available to us from the data set of solution vectors, in order to enhance the performance of these iterative solvers. In particular, we perform POD to the solution matrix $U = [u_1, ..., u_N]$, in order to obtain the projection matrix $\Phi^T$ and apply the AMG method either directly, or as a preconditioner in the PCG algorithm according to Algorithm 3. Algorithm 5 summarizes the steps needed to implement the proposed data-driven solution framework for parametrized linear systems.

Regarding the computational aspects behind the implementation of Algorithm 5, there are a few important remarks to be discussed:

- The main computational cost of the proposed strategy comes from the generation of the initial data set $U \in \mathbb{R}^{d \times N}$, that is, from solving the high-fidelity FE model for $N$ times (Step 1 of Algorithm 5). For practical applications, values of $N$ between $10^2 \div 10^3$ are usually considered adequate.
- The cost of training the CAE in Step 2a of Algorithm 5 is related to the size of the data set $U$. However, in contrast to other NN architectures, CAEs exhibit a robust scaling of this cost with respect to the dimensionality of the data. In addition, most machine learning libraries, such as Tensorflow, which was used in this work, already provide capabilities for GPU accelerated computations during the training of the CAE.
- The cost of training the FFNN in Step 2b of Algorithm 5 is affected by the dimensionality of both the parameter space and the latent space. However, these are small numbers and, thus, this cost is insignificant compared to the cost of Step 1 in the algorithm.
Algorithm 5. Proposed data-driven solution framework

1: **Step 1**: *Generation of the initial data set*

2: Generate $N$ instances of the system parameters $\{\theta_i\}_{i=1}^N$ and solve the detailed FE model to obtain the corresponding system responses $\{u_i\}_{i=1}^N$.

3: **Step 2**: *Construction of the data-driven solution framework*

4: **Step 2a**: Utilize the data set $\{u_i\}_{i=1}^N$ to train a CAE in order to learn a reduced representation $z \in \mathbb{R}^l$ for $u \in \mathbb{R}^d$, as well as a reconstruction map.

5: **Step 2b**: Utilize the data set of $\{\theta_i\}_{i=1}^N - \{z_i\}_{i=1}^N$ to train a FFNN in order to learn a mapping between the system parameters $\theta \in \mathbb{R}^n$ and the reduced representation $z \in \mathbb{R}^l$.

6: **Step 2c**: Construct the matrix $U = [u_1, \ldots, u_N]$ of solution vectors and perform POD by computing the eigenvalues and eigenvectors of the correlation matrix $R = UU^T$, or equivalently, the matrix $U^TU$. From the matrix of eigenvectors $\Phi$, form the reduced basis $\Phi_r$ by retaining only the $r$ most important eigenvectors.

7: **Step 3**: *Application of the framework to additional model simulations*

8: **Step 3a**: For a new parameter instance $\theta$ use the FFNN to obtain the corresponding reduced representation $z$.

9: **Step 3b**: Use the CAE’s decoder to map $z$ to the approximate solution vector $u$.

10: **Step 3c**: Perform additional iterations of the POD-2G algorithm with $u$ as the initial solution $u^{(0)}$ until a prescribed accuracy threshold has been reached.

**FIGURE 4** ITs test: A diametrically point loaded disk.

- Similarly to the above, the cost for constructing the POD basis in Step 2c of Algorithm 5 is also very small. This part requires the eigen decomposition of the correlation matrix $R \in \mathbb{R}^{d \times d}$, or preferably, the matrix $U^TU \in \mathbb{R}^{N \times N}$, since in most applications $N \ll d$.

- The selection of the optimal network hyperparameters for both the FFNN and the CAE is a very cumbersome task that, in general, requires a lot of trial-and-error attempts since there are no established relations between optimal NN architectures and the problem’s features. In this work the network architectures were chosen without any immediate considerations and this is, in fact, another strong point of the proposed methodology. Even if the surrogate is not perfectly accurate, its predictions can still be corrected with some additional iterations of the iterative solver. This approach overcomes the need to experiment with different network hyperparameters searching for the perfect combination.

In the following section, we test the solver on numerical applications of scientific interest and assess its performance in comparison with conventional solvers.

### 4 Numerical Applications

The proposed methodology is tested on three large scale parametrized systems. The first case is the indirect tensile strength (ITS) test, which is treated with the theory of 2D linear elasticity. The second case is a 3D elasticity problem from solid mechanics and the third case is a 3D deformable porous medium problem, also known as Biot problem.
### 4.1 | Indirect tensile strength test

A popular test to measure the tensile strength of concrete or asphalt materials is the ITS test. As shown in Figure 4, the test contains a cylindrical specimen loaded across its diameter to failure. The specimen is usually loaded at a constant deformation rate and measuring the load response. When the developed tensile stress in the specimen under loading exceeds its tensile strength then the specimen will fail. In this application, we restrict our analysis to the linear regime and model the cylinder as a 2D disk under plain strain assumptions, as shown in Figure 4. In this case, the weak form of the problem reads: Find $v \in \mathcal{V}(\Omega)$ such that

$$
\int_{\Omega} \sigma(v) : c(w) \, d\Omega = \int_{\Omega} f \cdot w \, d\Omega, \quad \forall w \in \mathcal{V}_c(\Omega)
$$

where,

$$
\sigma = \lambda \text{tr}(\varepsilon) I + 2\mu \varepsilon
$$

(15)

and the strain tensor $\varepsilon$ and $f$ the body forces, which are zero in this example. Also, $\mu$ and $\lambda$ are the Lamé’s constants, which are linked to the Young modulus $E$ and the Poisson ratio according to equations (17):

$$
\mu = \frac{E}{2(1 + \nu)}
$$

$$
\lambda = \frac{E \nu}{(1 + \nu)(1 - 2\nu)}
$$

(17)

In this example, the specimen has a diameter of 150 mm and due to symmetry in geometry and loading we only need to model one quarter of the disk, as illustrated in Figure 5. The solution of Equation (15) is obtained using a finite element mesh that consists of triangular plane-strain finite elements with a total of $d = 89440$ dofs. The Young modulus $E$ and

| Parameter | Distribution | Mean   | Standard deviation |
|-----------|--------------|--------|--------------------|
| $E$ (MPa) | Lognormal    | 2000   | 600                |
| $P$ (N)   | Lognormal    | $-1000$| 300                |

TABLE 1 Random parameters of the ITS test.

**FIGURE 5** Displacement magnitude $\|u\|$ for $E = 2000$ MPa and $P = -1000$ N.
the load $P$ are considered uncorrelated random variables following the lognormal distribution as described in Table 1. The Poisson ratio is considered to be a constant parameter $\nu = 0.3$. Figure 5 displays the contour plot of the displacement norm $||u||$ for the mean value of the random parameters, that is $E = 2000$ MPa and $P = -1000$ N.

The first step of the proposed procedure is to generate a sufficient number of offline samples. To this purpose, the Latin Hypercube sampling method was utilized to generate $N = 200$ parameter samples $\{[E_i, P_i]\}_{i=1}^{N}$. Subsequently, the corresponding problems are solved with the finite element method and the solution vectors obtained, $\{u_i\}_{i=1}^{N}$, are regarded as 'exact' solutions. Next, a surrogate model is trained over these solutions in order to establish a 'cheap' mapping from the parametric to solution space. The methodology for the surrogate model is described in section 3.2. The details of the selected CAE and FFNN architectures are presented in Figure 6.

To tackle the problem of overfitting, the standard hold-out approach was employed. In particular, the data set was randomly divided into train and validation subsets with a ratio of 70%–30% and each network’s performance on the validation data set was assessed in order to avoid overfitting. The CAE is trained for 500 epochs with a batch size of 20 and a learning rate of 0.001, while the FFNN is trained for 3000 epochs with a batch size of 20 and a learning rate of 0.0001. The average normalized $l_2$ norm error of the surrogate model on the test data set is 0.96%.

The second step is to form the POD basis $\Phi_r$ by performing eigen decomposition on the correlation matrix $U^T U$, with $U = [u_1, \ldots, u_N]$ being the solution matrix. In this case, the number of eigenvectors kept is $r = 8$, which correspond to over 99.99% of the variance in the training data. Subsequently, when all components of the proposed POD-2G solver are defined and fully trained, the methodology described in section 3.4 can be applied to obtain new system’s solutions for different parameter values.

In order to test the proposed POD-based solver, a number of $N_{\text{test}} = 500$ test parameter samples $\{[E_j, P_j]\}_{j=1}^{N_{\text{test}}}$ were generated according to their distribution. The corresponding problems were solved with the Ruge-Stüben AMG solver for 3, 4 and 5 grids (termed AMG-3G, -4G, -5G respectively), as well as the proposed POD-2G solver for different values of tolerance. The size of the system of equations at the coarsest level for each of these solvers is given in Table 2. The mean

**Figure 6** Surrogate model architecture.

**Table 2** Size of the problem at the coarsest grid for the different solvers.

| System size          |                |
|----------------------|----------------|
| Initial problem      | $89440 \times 89440$ |
| AMG-2G               | $24294 \times 24294$ |
| AMG-3G               | $5466 \times 5466$   |
| AMG-4G               | $1357 \times 1357$   |
| AMG-5G               | $372 \times 372$     |
| POD-2G               | $8 \times 8$        |

The load $P$ are considered uncorrelated random variables following the lognormal distribution as described in Table 1. The Poisson ratio is considered to be a constant parameter $\nu = 0.3$. Figure 5 displays the contour plot of the displacement norm $||u||$ for the mean value of the random parameters, that is $E = 2000$ MPa and $P = -1000$ N.

The first step of the proposed procedure is to generate a sufficient number of offline samples. To this purpose, the Latin Hypercube sampling method was utilized to generate $N = 200$ parameter samples $\{[E_i, P_i]\}_{i=1}^{N}$. Subsequently, the corresponding problems are solved with the finite element method and the solution vectors obtained, $\{u_i\}_{i=1}^{N}$, are regarded as 'exact' solutions. Next, a surrogate model is trained over these solutions in order to establish a 'cheap' mapping from the parametric to solution space. The methodology for the surrogate model is described in section 3.2. The details of the selected CAE and FFNN architectures are presented in Figure 6.

To tackle the problem of overfitting, the standard hold-out approach was employed. In particular, the data set was randomly divided into train and validation subsets with a ratio of 70%–30% and each network’s performance on the validation data set was assessed in order to avoid overfitting. The CAE is trained for 500 epochs with a batch size of 20 and a learning rate of 0.001, while the FFNN is trained for 3000 epochs with a batch size of 20 and a learning rate of 0.0001. The average normalized $l_2$ norm error of the surrogate model on the test data set is 0.96%.

The second step is to form the POD basis $\Phi_r$ by performing eigen decomposition on the correlation matrix $U^T U$, with $U = [u_1, \ldots, u_N]$ being the solution matrix. In this case, the number of eigenvectors kept is $r = 8$, which correspond to over 99.99% of the variance in the training data. Subsequently, when all components of the proposed POD-2G solver are defined and fully trained, the methodology described in section 3.4 can be applied to obtain new system’s solutions for different parameter values.

In order to test the proposed POD-based solver, a number of $N_{\text{test}} = 500$ test parameter samples $\{[E_j, P_j]\}_{j=1}^{N_{\text{test}}}$ were generated according to their distribution. The corresponding problems were solved with the Ruge-Stüben AMG solver for 3, 4 and 5 grids (termed AMG-3G, -4G, -5G respectively), as well as the proposed POD-2G solver for different values of tolerance. The size of the system of equations at the coarsest level for each of these solvers is given in Table 2. The mean
value of the CPU time and the number of cycles required for convergence to the desired tolerance are displayed in Figure 7 for the 3 AMG solvers and the proposed POD-2G with initial \( u(0) = 0 \), as well as \( u(0) = u_{\text{sur}} \), namely the solution delivered by the surrogate model.

From Figure 7, we notice that all AMG solvers require almost the same mean number of cycles, yet, AMG-4G is the most efficient scheme in terms of CPU time. This is because the CPU time is affected by both the size of the coarse problem, the number of times the prolongation/restriction operators are applied within a cycle, as well as the sparsity pattern of the system’s matrix at the coarsest level. In this regard, AMG-4G provides the optimal number of grids needed for this problem. However, a significant improvement on both the speedup and the number of iterations can be observed when applying the two POD solvers instead of the AMG solvers (see Table 3). This performance gain is increased with increasing tolerance \( \varepsilon \), reaching a speedup of \( \times 8.44 \) and \( \times 150.83 \) for the POD solvers with \( u(0) = 0 \) and \( u(0) = u_{\text{sur}} \) for \( \varepsilon = 10^{-4} \), respectively. On the other hand, for smaller values of \( \varepsilon \) such as \( 10^{-8} \), the speedup in CPU time obtained with POD-2G with \( u(0) = u_{\text{sur}} \) is \( \times 4.79 \), when compared with the case of AMG-3G with \( u(0) = 0 \). Even though the gain achieved in this case is much smaller than for the case of \( \varepsilon = 10^{-4} \), yet, it is still notable. Based on these results, the conclusion is drawn that a key component of the proposed methodology is to obtain a close estimation of the solution by the surrogate model, \( u(0) = u_{\text{sur}} \), since an initial solution \( u(0) \) from an accurately trained surrogate is capable of drastically reducing the computational cost.

Furthermore, the convergence behaviour of the proposed method when used as a preconditioner in the context of the PCG method is presented in Figure 8 and Table 4. Again, the results demonstrated that the proposed methodology is superior than classic AMG preconditioners. In particular, for \( \varepsilon = 10^{-4} \) and \( u(0) = 0 \), a reduction of computational cost of \( \times 2.10 \) is observed between the proposed method and the 4-grid AMG. In addition, the initial solution delivered by the surrogate model, \( u(0) = u_{\text{sur}} \), is again a crucial factor of fast convergence, and can lead to a speedup of up to \( \times 12.96 \) for the same case.

Finally, in order to highlight the computational gain of the proposed framework in the context of the Monte Carlo method, \( N_{\text{MC}} = 10^5 \) simulations are performed to determine the probability density function (PDF) of the vertical
**Figure 8** Comparison of mean CPU time and mean number of PCG iterations over 500 analyses for different preconditioners.

**Table 4** Computational speedup of different preconditioners compared to the AMG-4G preconditioner.

|                  | $\varepsilon = 10^{-4}$ | $\varepsilon = 10^{-5}$ | $\varepsilon = 10^{-6}$ | $\varepsilon = 10^{-7}$ | $\varepsilon = 10^{-8}$ |
|------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| AMG-3G ($\boldsymbol{u}^{(0)} = \mathbf{0}$) | $\times 1.00$            | $\times 1.00$            | $\times 1.00$            | $\times 1.00$            | $\times 1.00$            |
| POD-2G ($\boldsymbol{u}^{(0)} = \mathbf{0}$) | $\times 2.10$            | $\times 1.71$            | $\times 1.59$            | $\times 1.53$            | $\times 1.47$            |
| POD-2G ($\boldsymbol{u}^{(0)} = \boldsymbol{u_{sur}}$) | $\times 12.96$           | $\times 4.62$            | $\times 2.75$            | $\times 2.15$            | $\times 1.81$            |

**Figure 9** (A) PDF of $\boldsymbol{u}_{y}^{\text{top}}$ for $10^5$ MC simulations obtained using only the surrogate's predictions and its refined predictions with additional POD-2G iterations (exact solution) and (B) comparison of computational cost.
displacement $u_{y}^{\text{top}}$ of the top node, where the load $P$ is applied. Figure 9A displays the PDF obtained using the surrogate model solutions (blue line) and the solutions after the refinement with additional POD-2G iterations (red line). The latter serves as the exact PDF for the problem, and even though the two curves are in a relatively good agreement, yet, the POD-2G iterations are necessary in order to attain perfect accuracy. Regarding the computational gains, each simulation is solved with PCG and two different preconditioners, namely the proposed POD-2G method and a standard four grid Ruge-Stüben AMG preconditioner. The results are displayed in Figure 9B and verify that the proposed method is superior to classic AMG when dealing with parametrized systems. In particular, the conventional method needed $3.77 \times 10^5$ s to complete the $10^5$ simulations, while the proposed data-driven solver required $1.37 \times 10^5$ s for the same task including the offline cost (initial simulations and training of the surrogate model) for an accuracy threshold of $\epsilon = 10^{-6}$. This translates to a noticeable decrease in CPU time of $\times 2.75$, but this value would have been even greater by setting $\epsilon$ to a larger value, as evidenced by the results in Table 4.

### 4.2 3D elasticity problem

As our second test case we consider the 3D elasticity problem shown in Figure 10. The weak form of the problem reads: Find $\mathbf{v} \in \mathcal{Y}(\Omega)$ such that

$$
\int_{\Omega} \sigma(\mathbf{v}) : \mathbf{e}(\mathbf{w}) d\Omega = \int_{\Omega} \mathbf{f} : \mathbf{w} d\Omega, \quad \forall \mathbf{w} \in \mathcal{V}(\Omega)
$$

$$
\sigma = \lambda \mathbf{tr}(\mathbf{e}) \mathbf{I} + 2\mu \mathbf{e}
$$

where,

$$
\mathbf{e} = \begin{bmatrix}
e_{xx} & e_{xy} & e_{xz} \\
e_{yx} & e_{yy} & e_{yz} \\
e_{zx} & e_{zy} & e_{zz}
\end{bmatrix}
$$

is the strain tensor and $\mathbf{f}$ are the body forces, which are zero in this example.

In this example, each side of the cube has a length of 1.0 m and it is subjected to a pressure load $p$ at the center of its top surface as shown in 10. The solution of Equation (18) is obtained using a finite element mesh that consists of hexaedral finite elements with a total of $d = 50700$ (free) dofs. The Young modulus $E$ and the pressure load $p$ are
TABLE 6 Size of the problem at the coarsest grid for the different solvers.

| System size         |  |
|---------------------|---|
| Initial problem     | 50700 × 50700 |
| AMG-2G              | 10432 × 10432 |
| AMG-3G              | 1341 × 1341   |
| AMG-4G              | 252 × 252     |
| AMG-5G              | 9 × 9          |
| POD-2G              | 8 × 8          |

considered uncorrelated random variables following the lognormal distribution as described in Table 5. The Poisson ratio is considered to be a constant parameter $\nu = 0.3$.

To generate a sufficient number of offline samples, the Latin Hypercube sampling method was utilized to generate $N = 300$ parameter samples $\{[E_i, P_j]\}^{N}_{i=1}$ and the corresponding problems were solved with the finite element method to obtain the solution vectors $\{u_i\}^{N}_{i=1}$. The surrogate model is trained over these solutions and the selected CAE and FFNN architectures are presented in Figure 11. The CAE is trained for 500 epochs with a batch size of 20 and a learning rate of 0.001, while the FFNN is trained for 3000 epochs with a batch size of 20 and a learning rate of 0.0001. The average normalized $l_2$ norm error of the surrogate model on the test data set is 1.48%. Next, the POD basis $\Psi$ is produced by performing eigendecomposition on the matrix $U^T U$, and keeping $r = 8$ eigenvectors.

Having established all the components of the POD-2G solver it is then tested on a number of $N_{\text{test}} = 500$ 'unseen' parameter samples $\{[E_j, P_k]\}^{N_{\text{test}}}_{j=1}$, which were generated according to their distribution. The corresponding problems were solved with the Ruge-Stüben AMG solver for 3 and 4 grids (termed AMG-3G, -4G respec.), as well as the proposed POD-2G solver for different values of tolerance. The size of the system of equations at the coarsest level for each of these solvers is given in Table 6. The mean value of the CPU time and the number of cycles required for convergence to the desired tolerance are displayed in Figure 12 for the 2 AMG solvers and the proposed POD-2G with initial $u^{(0)} = 0$, as well as $u^{(0)} = u_{\text{sur}}$, namely the solution delivered by the surrogate model.

From Figure 12, we notice that both AMG solvers require almost the same mean number of cycles and CPU time. On the other hand, the two POD solvers show a significant improvement on both the number of iterations and CPU time, as shown in Table 7. This performance gain is increased with increasing tolerance $\epsilon$, reaching a speedup of $\times 5.08$ and $\times 53.72$ for the POD solvers with $u^{(0)} = 0$ and $u^{(0)} = u_{\text{sur}}$ for $\epsilon = 10^{-4}$, respectively. On the other hand, for smaller values of $\epsilon$ such as $10^{-8}$, the speedup in CPU time obtained with POD-2G with $u^{(0)} = u_{\text{sur}}$ is reduced.

The convergence behaviour of the proposed method when used as a preconditioner in the context of the PCG method is presented in Figure 13. Again, the results delivered by the proposed methodology showed its superior performance not
only over AMG preconditioners but also over incomplete Cholesky and Jacobi preconditioners. In this case, for $\epsilon = 10^{-5}$ and $\mathbf{u}^{(0)} = \mathbf{u}_{\text{surf}}$, a reduction of computational cost of $\times 10.05$ is observed between the proposed method and the 3-grid AMG, of $\times 7.07$ with the Cholesky and of $\times 3.17$ with the Jacobi preconditioners (See Table 8).

Finally, a Monte Carlo simulation is performed on this example as well, using $N_{\text{MC}} = 2 \times 10^5$ simulations to determine the PDF of the vertical displacement $u_{\text{top}}$ of the monitored node (see Figure 10). Figure 14A displays the PDF obtained using the surrogate model solutions (blue line) and the solutions after the refinement with additional POD-2G iterations (red line). As in the previous example, each simulation is solved with PCG and two different preconditioners, namely the proposed POD-2G and the Jacobi preconditioner. Again, the results obtained by the proposed methods demonstrate a significant computational advantage over conventional preconditioners. In particular, the Jacobi preconditioner needed $1.97 \times 10^5$ s to complete $2 \times 10^5$ simulations, while the proposed data-driven solver required $0.95 \times 10^5$ s for the same task including the offline cost (initial simulations and training of the surrogate model) for an accuracy threshold of $\epsilon = 10^{-6}$. This translates to a decrease in CPU time of $\times 2.07$.

### 4.3 Biot problem–Deformable porous medium

Biot’s theory describes wave propagation in a porous saturated medium, that is, a medium made of a solid matrix, fully soaked with a fluid. This formulation does not take into account the microscopic level and assumes that continuum mechanics can be applied to measurable macroscopic quantities. In weak form, Biot problem can be stated as: Find $\mathbf{v} \in \mathcal{V}(\Omega; \mathbb{R}^3)$ and $p \in \mathcal{V}(\Omega; \mathbb{R})$ such that

\[
\begin{align*}
\int_{\Omega} \mathbf{\sigma} : \mathbf{v} d\Omega - \int_{\Omega} p A : \mathbf{\epsilon}(\mathbf{v}) d\Omega &= 0, \forall \mathbf{v} \in \mathcal{V}_c(\Omega; \mathbb{R}^3) \\
\int_{\Omega} \mathbf{q} A : \mathbf{\epsilon}(\mathbf{v}) d\Omega + \sum_{\Omega} \nabla q \cdot D(\nabla p)^T d\Omega &= 0, \forall q \in \mathcal{V}_c(\Omega; \mathbb{R})
\end{align*}
\]

\[
\mathbf{\sigma} = \lambda \text{tr}(\mathbf{\epsilon}) I + 2\mu \mathbf{\epsilon}
\]
NIKOLOPoulos et al.

**FIGURE 13** Comparison of mean CPU time and mean number of PCG iterations over 500 analyses for different preconditioners.

**TABLE 8** Computational speedup of different preconditioners compared to the AMG-3G preconditioner.

| Preconditioner | $\varepsilon = 10^{-4}$ | $\varepsilon = 10^{-5}$ | $\varepsilon = 10^{-6}$ | $\varepsilon = 10^{-7}$ | $\varepsilon = 10^{-8}$ |
|----------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| AMG-3G ($u^{(0)} = 0$) | x1.00                    | x1.00                    | x1.00                    | x1.00                    | x1.00                    |
| IChol ($u^{(0)} = 0$)   | x1.28                    | x1.42                    | x1.70                    | x1.83                    | x1.99                    |
| Jacobi ($u^{(0)} = 0$)  | x2.87                    | x3.17                    | x3.63                    | x3.79                    | x4.05                    |
| POD-2G ($u^{(0)} = 0$)  | x3.68                    | x3.39                    | x3.50                    | x3.40                    | x3.47                    |
| POD-2G ($u^{(0)} = u_{sur}$) | x26.18               | x10.05                   | x6.42                    | x5.15                    | x4.54                    |

with $A$, $D$ being the Biot coefficient tensor and diffusion tensor, respectively. In this test case, the domain $\Omega$ is a cube and each side has a length of $L = 1.00 \text{ m}$. Regarding the boundary conditions, a pressure distribution $p^{|x|=0} = 1.0 \text{ MPa}$ is applied on the left face of the cube along with a displacement load $u_{\text{top}}^{|z|=1} = 0.20 \text{ m}$ on the top face, while all displacements $u_x$, $u_y$, and $u_z$ are restrained in the bottom face ($z = 0$). The problem definition is presented in Figure 15.

It is important to mention that the $u-p$ formulation of the Biot problem leads to non-symmetric matrices, which can also be indefinite in certain cases (e.g., compressible fluid flow). In the literature, several iterative solvers have been proposed targeted at accelerating the solution to this problem. However, due to the specific choice of boundary conditions in our study case, the resulting matrices are positive definite, meaning that all eigenvalues have strictly positive real part, and the skew symmetric part of these matrices is 'negligible' in the sense that the imaginary part of the eigenvalues is significantly smaller than their real part. For these reasons, we can still apply the AMG and PCG methods and obtain a very accurate approximation of the solution without the need to resort to more sophisticated solution techniques.

In this example, the finite element mesh contains 3-d hexahedral elements and the solution vector $u \in \mathbb{R}^d$ consists of the nodal values of displacements and pressure, where in this case the total number of free dofs is $d = 34839$. The Lame’s constants $\mu$ and $\lambda$ are considered uncorrelated random variables following the lognormal distribution as described in Table 9. The Poisson ratio $\nu$ is determined by:

$$\nu = \frac{\lambda}{2(\lambda + \mu)} < 0.5$$

We further assumed that the Biot coefficient tensor $A$ and $D$ are constant, taking the values:

$$A = \begin{bmatrix} 0.132 & 0.046 & 0.046 \\ 0.046 & 0.132 & 0.046 \\ 0.046 & 0.046 & 0.132 \end{bmatrix}, \quad D = \begin{bmatrix} 2.0 & 0.2 & 0 \\ 0.2 & 1.0 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

$\lambda$ and $\mu$ being the Biot coefficient tensor and diffusion tensor, respectively. In this test case, the domain $\Omega$ is a cube and each side has a length of $L = 1.00 \text{ m}$. Regarding the boundary conditions, a pressure distribution $p^{|x|=0} = 1.0 \text{ MPa}$ is applied on the left face of the cube along with a displacement load $u_{\text{top}}^{|z|=1} = 0.20 \text{ m}$ on the top face, while all displacements $u_x$, $u_y$, and $u_z$ are restrained in the bottom face ($z = 0$). The problem definition is presented in Figure 15.

It is important to mention that the $u-p$ formulation of the Biot problem leads to non-symmetric matrices, which can also be indefinite in certain cases (e.g., compressible fluid flow). In the literature, several iterative solvers have been proposed targeted at accelerating the solution to this problem. However, due to the specific choice of boundary conditions in our study case, the resulting matrices are positive definite, meaning that all eigenvalues have strictly positive real part, and the skew symmetric part of these matrices is 'negligible' in the sense that the imaginary part of the eigenvalues is significantly smaller than their real part. For these reasons, we can still apply the AMG and PCG methods and obtain a very accurate approximation of the solution without the need to resort to more sophisticated solution techniques.

In this example, the finite element mesh contains 3-d hexahedral elements and the solution vector $u \in \mathbb{R}^d$ consists of the nodal values of displacements and pressure, where in this case the total number of free dofs is $d = 34839$. The Lame’s constants $\mu$ and $\lambda$ are considered uncorrelated random variables following the lognormal distribution as described in Table 9. The Poisson ratio $\nu$ is determined by:

$$\nu = \frac{\lambda}{2(\lambda + \mu)} < 0.5$$

We further assumed that the Biot coefficient tensor $A$ and $D$ are constant, taking the values:

$$A = \begin{bmatrix} 0.132 & 0.046 & 0.046 \\ 0.046 & 0.132 & 0.046 \\ 0.046 & 0.046 & 0.132 \end{bmatrix}, \quad D = \begin{bmatrix} 2.0 & 0.2 & 0 \\ 0.2 & 1.0 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$
FIGURE 14  (A) PDF of $u_{y}^{\text{top}}$ for $2 \times 10^5$ MC simulations obtained using only the surrogate’s predictions and its refined predictions with additional POD-2G iterations (exact solution) and (B) comparison of computational cost.

FIGURE 15  Geometry and boundary conditions of Biot problem.

TABLE 9  Random parameters of the Biot problem.

| Parameter | Distribution | Mean  | Standard deviation |
|-----------|--------------|-------|--------------------|
| $\mu$(MPa) | Lognormal    | 0.30  | 0.09               |
| $\lambda$(MPa) | Lognormal    | 1.70  | 0.51               |
As a first step, the Latin Hypercube sampling method was utilized to generate $N = 300$ parameter samples $\{\{\mu_i, \lambda_i\}\}_{i=1}^N$. The next steps are similar with those of the previous numerical examples. The surrogate’s architecture is presented in Figure 17. The CAE is trained for 100 epochs with a batch size of 10 and a learning rate of $10^{-3}$, while the FFNN is trained for 5000 epochs with a batch size of 20 and a learning rate of $10^{-4}$. The average normalized $l_2$ norm error of the surrogate model in the test dataset is 0.68%.

As in the previous numerical examples, a number of $N_{\text{test}} = 500$ parameter vectors $\{\{\mu_j, \lambda_j\}\}_{j=1}^{N_{\text{test}}}$ were generated according to their distribution and the corresponding problems were solved with the proposed POD-based solver and different Ruge-Stüben AMG solvers, with the number of grids ranging from 2 to 6. The size of the system of equations at the coarsest level for each of these solvers is presented in Table 10. For this example, eight eigenvectors were retained in the POD expansion, as these were sufficient for capturing 99.99% of the dataset’s variance.

The mean value of the CPU time and the number of cycles required for convergence to the desired number of tolerance are displayed in Figure 18 and Table 11. The results are very promising in terms of computational cost. For instance, for $\epsilon = 10^{-5}$ and $u^{(0)} = 0$, a reduction of computational cost of $\times 7.32$ is achieved when comparing the proposed solver with the 3-grid AMG solver. Furthermore, obtaining an accurate initial solution $u^{(0)}$ is again a very important component of the proposed framework. Specifically, by considering $u^{(0)} = u_{\text{sur}}$ instead of $u^{(0)} = 0$ for $\epsilon = 10^{-5}$, an additional decrease in CPU time of $\times 4.31$ can be achieved.
Table 10  Size of the problem at the coarsest grid for the different solvers.

| System      | Initial problem | AMG-2G | AMG-3G | AMG-4G | AMG-5G | AMG-6G | POD-2G |
|-------------|-----------------|--------|--------|--------|--------|--------|--------|
| System size | 34839 x 34839   | 8625 x 8625 | 1421 x 1421 | 229 x 229 | 47 x 47 | 9 x 9 | 8 x 8 |

Figure 18  Comparison of mean CPU time and mean number of cycles over 500 analyses for different multigrid solvers.

Table 11  Computational speedup of different solvers compared to AMG-3G.

| ε | AMG-3G (u^{(0)} = 0) | AMG-5G (u^{(0)} = 0) | AMG-6G (u^{(0)} = 0) | POD-2G (u^{(0)} = 0) | POD-2G (u^{(0)} = \text{sur}) |
|---|---------------------|---------------------|---------------------|---------------------|---------------------|
| ε = 10^{-4} | x1.00 | x0.97 | x0.97 | x12.31 | x76.89 |
| ε = 10^{-5} | x1.00 | x0.96 | x0.96 | x7.32  | x31.54 |
| ε = 10^{-6} | x1.00 | x0.96 | x0.96 | x4.89  | x17.90 |
| ε = 10^{-7} | x1.00 | x0.96 | x0.96 | x2.34  | x12.12 |
| ε = 10^{-8} | x1.00 | x0.96 | x0.96 | x1.77  | x4.35  |

The convergence behaviour of the proposed method when used as a preconditioner in the context of the PCG method is presented in Figure 19. Again, the results delivered by the proposed methodology showed its superior performance not only over AMG preconditioners but also over ILU and Jacobi preconditioners. In this case, for ε = 10^{-5} and u^{(0)} = 0, a reduction of computational cost of \times 2.37 is observed between the proposed method and the 3-grid AMG, of \times 1.63 with the ILU and of \times 1.16 with the Jacobi. Last but not least, the initial solution delivered by the surrogate model, u^{(0)} = \text{sur}, managed to further reduce the computational time by \times 2.12 when compared to POD-2G with u^{(0)} = 0 (See Table 12).

Finally, a Monte Carlo simulation is performed on this example as well, using N_{MC} = 2 \times 10^5 simulations to determine the PDF of the displacement magnitude ||u|| of the monitored node (see Figure 15). Figure 20A displays the PDF obtained using the surrogate model solutions (blue line) and the solutions after the refinement with additional POD-2G iterations (red line). As in the previous examples, each simulation is solved with PCG and two different preconditioners, namely
FIGURE 19  Comparison of mean CPU time and mean number of PCG iterations over 500 analyses for different preconditioners.

TABLE 12  Computational speedup of different preconditioners compared to the AMG-3G preconditioner.

| Preconditioner | $\epsilon = 10^{-4}$ | $\epsilon = 10^{-5}$ | $\epsilon = 10^{-6}$ | $\epsilon = 10^{-7}$ | $\epsilon = 10^{-8}$ |
|----------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| AMG-3G ($u(0) = 0$) | ×1.00 | ×1.00 | ×1.00 | ×1.00 | ×1.00 |
| ILU ($u(0) = 0$) | ×1.38 | ×1.45 | ×1.61 | ×1.77 | ×2.63 |
| Jacobi ($u(0) = 0$) | ×2.50 | ×2.04 | ×1.73 | ×1.85 | ×2.70 |
| POD-2G ($u(0) = 0$) | ×2.86 | ×2.37 | ×1.74 | ×1.86 | ×2.71 |
| POD-2G ($u(0) = u_{sur}$) | ×8.88 | ×5.02 | ×3.98 | ×2.64 | ×3.55 |

FIGURE 20  (A) PDF of $||u||$ at monitored dof for $2 \times 10^5$ MC simulations obtained using only the surrogate's predictions and its refined predictions with additional POD-2G iterations (exact solution) and (B) comparison of computational cost.
the proposed POD-2G and a standard three grid Ruge-Stüben AMG preconditioner. Again, the results obtained by the proposed methods demonstrate a significant computational advantage over conventional preconditioners. In particular, the Jacobi preconditioner needed $4.23 \times 10^5$ s to complete $2 \times 10^5$ simulations, while the proposed data-driven solver required $1.75 \times 10^5$ s for the same task including the offline cost (initial simulations and training of the surrogate model). This translates to a decrease in CPU time of $\times 2.42$.

5 | CONCLUSIONS

The present work introduces a framework for accelerating the solution of parametrized problems that require multiple model evaluations. The proposed framework consists of two distinct yet complementary steps. The first step in the methodology is the construction of a ‘cheap-to-evaluate’ metamodel using FFNNs and CAEs, trained over a reduced set of high-fidelity system solutions. Despite giving very accurate predictions at new parameter instances, these predictions are bound to exhibit some discrepancy with respect to the actual system solutions since they are not constrained by any physical laws. The second step in the methodology aims precisely at fixing this by proposing a data-driven iterative solver, inspired by the AMG method, that will refine the metamodel’s predictions until a prescribed level of accuracy has been attained. In particular, using again the already available set of high-fidelity system solutions, POD is performed on this set to identify the subspace that captures most of the variation in the system responses. Next, a 2-level multigrid scheme is developed, termed POD-2G, using the projection operator from POD as the prolongation operator. This scheme was tested on numerical applications as a standalone solver, as well as a preconditioner to PCG, and in both cases, its superior performance with respect to conventional iterative solvers was demonstrated.

ACKNOWLEDGMENTS

This work has been supported by the The European High Performance Computing Joint Undertaking (EuroHPC) grant DComEX (956201-H2020-JTI-EuroHPC-2019-1).

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

DATA AVAILABILITY STATEMENT

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

ORCID

Stefanos Nikolopoulos https://orcid.org/0000-0003-3896-1141
Ioannis Kalogeris https://orcid.org/0000-0001-7237-3605
Vissarion Papadopoulos https://orcid.org/0000-0002-8717-4585

REFERENCES

1. Barrett R, Berry M, Chan TF, et al. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. 2nd ed. SIAM; 1994.
2. Benzi M, Cullum JK, Tuma M. Robust approximate inverse preconditioning for the conjugate gradient method. SIAM J Sci Comput. 2000;22(4):1318-1332. doi:10.1137/S1064827599356900
3. Lin FR, Yang SW, Jin XQ. Preconditioned iterative methods for fractional diffusion equation. J Comput Phys. 2014;256:109-117. doi:10.1016/j.jcp.2013.07.040
4. Herzog R, Sachs E. Preconditioned conjugate gradient method for optimal control problems with control and state constraints. SIAM J Matrix Anal Appl. 2010;31(5):2291-2317. doi:10.1137/090779127
5. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput. 1986;7(3):856-869. doi:10.1137/0907058
6. Shakib F, Hughes TJ, Johan Z. A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis. Comput Methods Appl Mech Eng. 1989;75(1-3):415-456. doi:10.1016/0045-7825(89)90040-6
7. Baglama J, Calvetti D, Golub G, Reichel L. Adaptively preconditioned gmres algorithms. SIAM J Sci Comput. 1998;20(1):243-269. doi:10.1137/S1064827596305258
8. Concis P, Golub GH, Meurant G. Block preconditioning for the conjugate gradient method. SIAM J Sci Stat Comput. 1985;6(1):220-252. doi:10.1137/0906018
9. Toselli A, Widlund OB. Domain Decomposition Methods: Algorithms and Theory. Springer; 2005.
10. Tallec P, Roek Y, Vidrascu M. Domain decomposition methods for large linearly elliptic three-dimensional problems. *J Comput Appl Math.* 1991;34(1):93-117. doi: 10.1016/0374-0273(91)90150-I

11. Farhat C, Roux FX. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int J Numer Methods Eng.* 1991;32(6):1205-1227. doi: 10.1002/nme.1620320604

12. Farhat C, Mandel J, Roux FX. Optimal convergence properties of the FETI domain decomposition method. *Comput Methods Appl Mech Eng.* 1994;115(3):365-385. doi: 10.1016/0045-7825(94)90068-X

13. Fragakis Y, Papadrakakis M. The mosaic of high performance domain decomposition methods for structural mechanics: formulation, interrelation and numerical efficiency of primal and dual methods. *Comput Methods Appl Mech Eng.* 2003;192:3799-3830.

14. Cai XC, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J Sci Comput.* 1999;21(2):792-797. doi: 10.1137/S106482759732678X

15. Al Daas H, Grigori L, Jolivet P, Tournier PH. A multilevel Schwarz preconditioner based on a hierarchy of robust coarse spaces. *SIAM J Sci Comput.* 2020;43(2):A1907-A1928. doi: 10.1137/19M1266964

16. Trottenberg U, Oosterlee C, Schuller A. *Multigrid 1st Edition.* Academic Press; 2000.

17. Iwamura C, Costa FS, Sbarski I, Easton A, Li N. An efficient algebraic multigrid preconditioned conjugate gradient solver. *Comput Methods Appl Mech Eng.* 2003;192(20-21):2299-2318. doi: 10.1016/S0045-7825(02)00378-X

18. Heys J, Manteuffel T, McCormick S, Olson L. Algebraic multigrid for higher-order finite elements. *SIAM J Sci Comput.* 2005;26(5):1909-1926. doi: 10.1137/040607346

19. Ramage A. A multigrid preconditioner for stabilised discretisations of advection-diffusion problems. *J Comput Appl Math.* 1999;110(1):187-203. doi: 10.1016/S0377-0427(99)00234-4

20. Bergamaschi L. A survey of low-rank updates of preconditioners for sequences of symmetric linear systems. *SIAM J Sci Comput.* 2020;42(4):A1907-A1928. doi: 10.1137/19M1266964

21. Chapman A, Saad Y. Deflated and augmented Krylov subspace techniques. *SIAM J Sci Comput.* 2020;42(4):A1044-A1074. doi: 10.1137/20M1331123

22. Chojczynski A, Teixeira A, Neves L, Cardoso J, Guedes SC. Review and application of artificial neural networks models in reliability analysis of steel structures. *Struct Safe.* 2015;52:78-89. doi: 10.1016/j.strusafe.2014.09.002

23. Nikolopoulos S, Kalogeris I, Papadrakakis M. Non-intrusive surrogate modeling for parametrized time-dependent partial differential equations using convolutional autoencoders. *Eng Appl Artif Intell.* 2022;109:104652. doi: 10.1016/j.engappai.2021.104652

24. Nikolopoulos S, Kalogeris I, Papadrakakis M. Machine learning accelerated transient analysis of stochastic nonlinear structures. *Eng Struct.* 2022;257:114020. doi: 10.1016/j.engstruct.2022.114020

25. Xu J, Duraisamy K. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Comput Methods Appl Mech Eng.* 2020;372:113379. doi: 10.1016/j.cma.2020.113379
41. Fresca S, Dede L, Manzoni A. A comprehensive deep learning-based approach to reduced order Modeling of nonlinear time-dependent parametrized PDEs. J Sci Comput. 2021;87(61). doi:10.1007/s10915-021-01462-7
42. Gruber A, Gunzburger M, Ju L, Wang Z. A comparison of neural network architectures for data-driven reduced-order modeling. Comput Methods Appl Mech Eng. 2022;393. doi:10.1016/j.cma.2022.114764
43. Lee K, Carlberg KT. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. J Comput Phys. 2020;404. doi:10.1016/j.jcp.2019.108973
44. San O, Maulik R, Ahmed M. An artificial neural network framework for reduced order modeling of transient flows. Commun Nonlinear Sci Numer Simul. 2019;77:271-287. doi:10.1016/j.cnsns.2019.04.025
45. Yu Y, Yao H, Liu Y. Aircraft dynamics simulation using a novel physics-based learning method. Aerosp Sci Technol. 2019;87:254-264. doi:10.1016/j.ast.2019.02.021
46. Zhou JM, Dong L, Guan W, Yan J. Impact load identification of nonlinear structures using deep recurrent neural network. Mech Syst Signal Process. 2019;133. doi:10.1016/j.ymssp.2019.106292
47. Gonzalez FJ, Balajewicz M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. arXiv. 2018.
48. Carlberg K, Farhat C. A low-cost, goal-oriented ‘compact proper orthogonal decomposition’ basis for model reduction of static systems. Int J Numer Methods Eng. 2011;86(3):381-402. doi:10.1002/nme.3074
49. Zahr MJ, Avery P, Farhat C. A multilevel projection-based model order reduction framework for nonlinear dynamic multiscale problems in structural and solid mechanics. Int J Numer Methods Eng. 2017;112(8):855-881. doi:10.1002/nme.5535
50. Agathos K, Bordas SPA, Chatzi E. Parametrized reduced order modeling for cracked solids. Int J Numer Methods Eng. 2020;121(20):4537-4565. doi:10.1002/nme.6447
51. Chinesta F, Ammar A, Cueto E. Proper generalized decomposition of multiscale models. Int J Numer Methods Eng. 2010;83(8-9):1114-1132. doi:10.1002/nme.2794
52. Ladevèze P, Passieux JC, Néron D. The LATIN multiscale computational method and the proper generalized decomposition. Comput Methods Appl Mech Eng. 2010;199(21-22):1287-1296. doi:10.1016/j.cma.2009.06.023
53. Ladevèze P, Chamoin L. On the verification of model reduction methods based on the proper generalized decomposition. Comput Methods Appl Mech Eng. 2011;200(23-24):2032-2047. doi:10.1016/j.cma.2011.02.019
54. Da l Santo N, Deparis S, Pegolotti L. Data driven approximation of parametrized PDEs by reduced basis and neural networks. J Comput Phys. 2020;416:109550. doi:10.1016/j.jcp.2020.109550
55. Salvador M, Dedè L, Manzoni A. Non intrusive reduced order modeling of parametrized PDEs by kernel POD and neural networks. Comput Math Appl. 2021;104:1-13. doi:10.1016/j.camwa.2021.11.001
56. Kalogeris I, Papadopoulos V. Diffusion maps-aided neural networks for the solution of parametrized PDEs. Comput Methods Appl Mech Eng. 2021;376:113568. doi:10.1016/j.cma.2020.113568
57. Santos KR, Giovannis DG, Shields MD. Grassmannian diffusion maps–based dimension reduction and classification for high-dimensional data. SIAM J Sci Comput. 2022;44(2):B250-B274. doi:10.1137/20M137001X
58. Kadeethum T, Ballarin F, Choi Y, O’Malley D, Yoon H, Bouklas N. Non-intrusive reduced order modeling of natural convection in porous media using convolutional autoencoders: comparison with linear subspace techniques. Adv Water Resour. 2022;160. doi:10.1016/j.adwatre.2021.104098
59. Vlachas K, Tatsis K, Agathos K, Brink AR, Chatzi E. A local basis approximation approach for nonlinear parametric model order reduction. J Sound Vib. 2021;502:116055. doi:10.1016/j.jsv.2021.116055
60. Heinlein A, Klawonn A, Lanser M, Weber J. Machine learning in adaptive domain decomposition methods—predicting the geometric location of constraints. SIAM J Sci Comput. 2020;41(6):A3887-A3912. doi:10.1137/18M1205364
61. Lazzara M, Chevalier M, Colombo M, Garay Garcia J, Lapeyre C, Teste O. Surrogate modeling for an aircraft dynamic land-
62. Fresca S, Manzoni A. POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Comput Methods Appl Mech Eng. 2022;388. doi:10.1016/j.cma.2021.114181
63. Carlb erg K, Forstall V, Tuminaro R. Krylov-subspace recycling via the POD-augmented conjugate-gradient method. SIAM J Matrix Anal Appl. 2016;37(3):1304-1336. doi:10.1137/16M1057693
64. Heinlein A, Klawonn A, Lanser M, Weber J. Machine learning in adaptive domain decomposition methods—predicting the geometric location of constraints. SIAM J Sci Comput. 2019;41(6):A3887-A3912. doi:10.1137/18M1205364
65. Chen Y, Dong B, Xu J. Meta-MgNet: meta multigrid networks for solving parameterized partial differential equations. J Comput Phys. 2022;455:110996. doi:10.1016/j.jcp.2022.110996.
66. Luz I, Galun M, Maron H, Basri R, Yavneh I. Learning algebraic multigrid using graph neural networks. PMLR. 2020 6489-6499.
67. Stüben K. A review of algebraic multigrid. J Comput Appl Math. 2001;128(1):281-309. doi:10.1016/S0377-0427(00)00516-1
68. Brezina M, Cleary AJ, Falgout RD, et al. Algebraic multigrid based on element interpolation (AMGe). SIAM J Sci Comput. 2001;22(5):1570-1592. doi:10.1137/S1064827598344303
69. Treister E, Yavneh I. Square and stretch multigrid for stochastic matrix eigenproblems. Numer Linear Algebra Appl. 2010;17(2-3):229-251. doi:10.1002/nla.708
70. Napov A, Notay Y. An efficient Multigrid method for graph Laplacian systems. Electron Trans Numer Anal. 2016;45:201-218.
71. Facca E, Benzi M. Fast iterative solution of the optimal transport problem on graphs. *SIAM J Sci Comput*. 2021;43(3):A2295-A2319. doi:10.1137/20M137015X
72. Ruge JW, Stüben K. Algebraic Multigrid. 4. Algebraic Multigrid: 73-130. *Frontiers in Applied Mathematics*. SIAM; 1987.
73. Vanek P, Mandel J, Brezina M. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Comput Secur*. 1996;56(3):179-196. doi:10.1007/BF02238511
74. Mo S, Zhu Y, Zabaras N, Shi X, Wu J. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resour Res*. 2019;55(1):703-728. doi:10.1029/2018WR023528
75. Pasetto D, Ferronato M, Putti M. A reduced order model-based preconditioner for the efficient solution of transient diffusion equations. *Int J Numer Methods Eng*. 2017;109(8):1159-1179. doi:10.1002/nme.5320
76. Dal Santo N, Deparis S, Manzoni A, Quarteroni A. Multi space reduced basis preconditioners for large-scale parametrized PDEs. *SIAM J Comput*. 2018;40(2):A954-A983. doi:10.1137/16M1089149
77. Diaz Cortes G, Vuijk C, Jansen J. On POD-based deflation vectors for DPGC applied to porous media problems. *J Comput Appl Math*. 2018;330:193-213. doi:10.1016/j.cam.2017.06.032
78. Olsson AM, Sandberg GE. Latin hypercube sampling for stochastic finite element analysis. *J Eng Mech*. 2002;128(1):121-125.
79. Rathinam M, Petzold LR. A new look at proper orthogonal decomposition. *SIAM J Numer Anal*. 2003;41(5):1893-1925. doi:10.1137/S0036142901389049
80. Lieu T, Farhat C, Lesoinne M. Reduced-order fluid/structure modeling of a complete aircraft configuration. *Comput Methods Appl Mech Eng*. 2006;195(41):5730-5742. doi:10.1016/j.cma.2005.08.026
81. Rapún ML, Vega JM. Reduced order models based on local POD plus Galerkin projection. *J Comput Phys*. 2010;229(8):3046-3063. doi:10.1016/j.jcp.2009.12.029
82. Abadi M, Barham P, Chen J, et al. Tensorflow: a system for large-scale machine learning. Paper presented at: 12th USENIX Symposium on Operating Systems Design and Implementation. 2016 265–283.
83. Carcione JM. Chapter 7-Biot theory for porous media. *Wave Fields in Real Media (Third Edition)*. Elsevier; 2015.
84. Bergamaschi L, Ferronato M, Gambolati G. Novel preconditioners for the iterative solution to FE-discretized coupled consolidation equations. *Comput Methods Appl Mech Eng*. 2007;196(25-28):2647-2656. doi:10.1016/j.cma.2007.01.013
85. Lee JJ, Mardal KA, Winther R. Parameter-robust discretization and preconditioning of Biot’s consolidation model. *SIAM J Sci Comput*. 2017;39(1):A1-A24. doi:10.1137/15M1029473
86. Castelletto N, Klevtsov S, Hajibeygi H, Tchelepi H. Multiscale two-stage solver for Biot’s poroelasticity equations in subsurface media. *Comput Geosci*. 2019;23(2):207-224. doi:10.1007/s10596-018-9791-z
87. Gaspar FJ, Rodrigo C. On the fixed-stress split scheme as smoother in multigrid methods for coupling flow and geomechanics. *Comput Methods Appl Mech Eng*. 2017;326:526-540. doi:10.1016/j.cma.2017.08.025
88. Axelsson O, Karátson J. Superlinearly convergent CG methods via equivalent preconditioning for nonsymmetric elliptic operators. *Numer Math*. 2004;99(2):197-223. doi:10.1007/s00211-004-0557-2
89. Karátson J, Kurics T. Superlinearly convergent PCG algorithms for some nonsymmetric elliptic systems. *J Comput Appl Math*. 2008;212(2):214-230. doi:10.1016/j.cam.2006.12.004

**How to cite this article:** Nikolopoulos S, Kalogeris I, Stavroulakis G, Papadopoulos V. AI-enhanced iterative solvers for accelerating the solution of large-scale parametrized systems. *Int J Numer Methods Eng*. 2024;125(2):e7372. doi:10.1002/nme.7372