Embedded symmetric positive semi-definite machine-learned elements for reduced-order modeling in finite-element simulations with application to threaded fasteners

Eric Parish[1*], Payton Lindsay[2], Timothy Shelton[2], John Mersch[2]

[1*]Sandia National Laboratories, 7011 East Ave, Livermore, 94550, CA, United States.
[2*]Sandia National Laboratories, 1515 Eubank Blvd SE, Albuquerque, 87123, NM, United States.

*Corresponding author(s). E-mail(s): ejparis@sandia.gov;
Contributing authors: plindsa@sandia.gov; trshelt@sandia.gov; jpmersc@sandia.gov;

Abstract

We present a machine-learning strategy for finite element analysis of solid mechanics wherein we replace complex portions of a computational domain with a data-driven surrogate. In the proposed strategy, we decompose a computational domain into an "outer" coarse-scale domain that we resolve using a finite element method (FEM) and an "inner" fine-scale domain. We then develop a machine-learned (ML) model for the impact of the inner domain on the outer domain. In essence, for solid mechanics, our machine-learned surrogate performs static condensation of the inner domain degrees of freedom. This is achieved by learning the map from displacements on the inner-outer domain interface boundary to forces contributed by the inner domain to the outer domain on the same interface boundary. We consider two such mappings, one that directly maps from displacements to forces without constraints, and one that maps from displacements to forces by virtue of learning a symmetric positive semi-definite (SPSD) stiffness matrix. We demonstrate, in a simplified setting, that learning an SPSD stiffness matrix results in a coarse-scale problem that is well-posed with a unique solution. We present numerical experiments on several exemplars, ranging from finite deformations of a cube to finite deformations with contact of a fastener-bushing geometry. We demonstrate that enforcing an SPSD stiffness matrix drastically improves the

1

robustness and accuracy of FEM–ML coupled simulations, and that the resulting methods can accurately characterize out-of-sample loading configurations with significant speedups over the standard FEM simulations.

1 Introduction

Finite element analysis of component and assembly-level systems remains computationally intensive despite the tremendous advancements in algorithmic research and computing power that have occurred over the past several decades. This problem is exacerbated when the underlying problem is multiscale in nature, wherein the physical discretization of the governing equations requires resolving a wide range of length and/or time scales. One such example that motivates the present work is that of systems-level models that involve threaded fasteners. In such systems, fasteners can be an integral connector of many sub-components, and accurately modeling the behavior of the fastener is critical for many analyses. Unfortunately, directly resolving the full fastener within a finite element model is difficult: the geometries are challenging to mesh, the resulting meshes can require hundreds of thousands of degrees of freedom, and the required mechanics that need to be simulated are complex as they often involve contact, friction, etc. As an example, Figure 1 depicts a schematic of a ratcheting mechanism; fasteners are an essential part of this complex mechanism and must capture relevant mechanics to properly assess quantities of interest. In general, directly resolving each fastener is often not feasible in systems-level models, and approaches capable of reducing this computational burden are needed.

Various approaches have been developed to reduce computational complexities in finite element methods. These methods include the variational multiscale method [21], Guyan reduction [18], Craig–Bampton reduction [13], the generalized finite element method [17], multiscale methods [20], and reduced basis/proper-orthogonal decomposition reduced-order models [10, 12, 29, 33]. Machine learning-based techniques have garnered significant attention recently and are the focus of the present literature review. "Smart" finite elements were introduced in [11], wherein a machine-learned regression model was used to learn a direct relationship between the internal state of an element and its forces. The approach enforces frame indifference and conservation of linear and angular momentum and was shown to reduce the computational cost as opposed to a traditional finite element model. Refs. [5, 34] proposed additional techniques for computing the tangent stiffness matrix of smart elements based on finite differences, automatic differentiation, and neural networks. In addition, [34] proposes a composite loss function with terms for both the element forces as well as the tangent stiffness. Similarly, [25] proposes nonlinear meta elements (i.e., element patches) using deep learning. Unlike [11, 34] this approach learns generalizable meta elements (where each meta element comprises multiple finite elements). The primary input to the regression model defining the meta element is the displacement field on element boundary, while the outputs are the displacements, stresses, and forces within the meta element. Lastly, [8] explores quantifying uncertainties in machine-learned

elements that map from displacements to forces by virtue of an extension to deep ensembles [26].

The aforementioned approaches seek to learn a generalizable (meta-)element that can be applied repeatedly throughout the domain. Another similar body of work focuses on *domain-decomposition* in where a computational domain is divided into subdomains. Once partitioned, each subdomain can be solved independently, and a global solution is obtained by coupling the various domains, e.g., with the Schwarz method [31]. Data-driven techniques have received significant attention in this field as well, where the general idea is to use a cheaper-to-evaluate surrogate to model a complex subdomain. Refs. [27, 28], for instance, proposed D3M and DeepDDM, both of which employ physics-informed neural networks (PINNs) to learn subdomain-
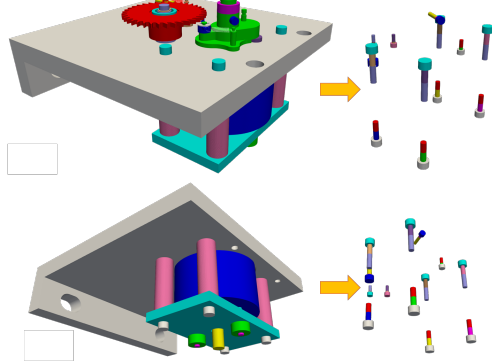

Fig. 1: Schematic of a ratcheting mechanism. Fasteners are an integral part of this complex mechanism and must capture relevant mechanics to properly assess quantities of interest.

solutions conditioned on boundary data, and demonstrated their approaches on Poisson's equation and Schrodinger's equation. We note that D3M and DeepDDM employ PINNs for all subdomains, as opposed to a FEM for some of the subdomains. Refs. [6, 14, 15, 19, 22–24, 30] propose domain decomposition strategies leveraging reduced basis and proper-orthogonal decomposition (Petrov-)Galerkin reduced-order models. These methods operate by projecting the finite element model onto low-dimensional subspaces computed, e.g., from training data. The different subdomains are then coupled via, e.g., Lagrange multipliers [19], the Schwarz method [31, 32]. Lastly, one final approach of relevance was proposed in [3] for multiscale problems, where the authors aim to resolve only the domain of interest by learning an interface condition leveraging long short term memory (LSTM) networks and ideas from upwinding. The setup was demonstrated on the transient Burgers' equation and Euler equations in one dimension.

In the present work, we develop a machine learning strategy for replacing portions of a computational domain, such as threaded fasteners, with a data-driven surrogate. We develop the proposed strategy within the context of hypoelastic material models for static solid mechanics (SM) FEM simulations of bodies under finite deformations exposed to contact and friction; we leave the extension of the approach to nonlinear material models and dynamic simulations as future work. In the proposed strategy we decompose a computational domain into an "outer" domain that we resolve with an FEM, and an "inner" domain. We then develop a machine-learned model for the

impact of the inner domain on the outer domain. In essence, for SM our machine-learned surrogate learns the map from displacements on the interface boundary to forces contributed by the inner domain to the outer domain on the same interface boundary (we refer to these forces as internal forces).

We propose two displacement-to-force mappings in the present work. First, we propose a direct displacement-to-force mapping with dimension reduction. In this approach, we identify low-dimensional subspaces for compact approximations to the interface displacements and internal forces via proper orthogonal decomposition (POD) [9]. A mapping between reduced coordinates for these fields is then learned via regression; i.e., for interface displacements $\mathbf{u}$ and internal forces $\mathbf{f}$, we learn a mapping $\mathbf{f} = \mathbf{g}(\mathbf{u})$ where $\mathbf{g}$ is the learned surrogate. The second approach we propose is a structure-preserving displacement-to-stiffness mapping with dimension reduction. In this approach, we again identify low-dimensional structure for the displacements and internal forces on the interface, but instead learn a displacement-to-force mapping in the form of a stiffness matrix, i.e., for interface displacements $\mathbf{u}$ and internal forces $\mathbf{f}$, we learn a mapping $\mathbf{f} = \mathbf{K}(\mathbf{u})\mathbf{u}$ where $\mathbf{K}(\mathbf{u})$ is a learned stiffness which can be a nonlinear function of the displacements. We further enforce the learned stiffness matrix to be symmetric positive semi-definite (SPSD)[1]. We show, in a simplified setting, that the addition of this SPSD model results in a coarse-scale problem that is symmetric positive definite (SPD) and well-posed. Further, enforcing SPSD structure allows our ML model to be interfaced with common SM solvers such as conjugate gradient. Thematically similar ideas were pursued in Ref. [4, 37] within the context of learning constitutive relations. As will be shown, we find this latter approach is critical for successful integration of the method into our finite element solver.

The current work has commonalities with several existing efforts. For example, the present approach can be viewed as a smart element or a "meta-element" strategy in which we are learning a meta-element for our inner domain(s). This synergy is particularly relevant if the inner domain is a component which is repeated throughout the computational domain, like a threaded fastener. We propose a more limited approach than [11, 25, 34], however: we aim to remove certain hard-to-resolve portions of a domain rather than develop a general smart element. In this sense, our proposed methodology can be interpreted as a type of Guyan reduction that is applicable to the nonlinear regime. Second, our approach can be viewed as a domain-decomposition method, where we employ an ML model to simulate the impact of fine-scale domain on the coarse-scale domain. Comparing the present work to the literature, we highlight several novelties:

1. We propose regression models equipped with a dimension reduction strategy to reduce both the training cost of the model as well as the number of datum required to learn an accurate and generalizable model. This strategy enables our approach to be applied to systems where the ML model must predict thousands of degrees of freedom.

---

[1] We note that our ML model is only learning a boundary coupling term and enforcing SPSD structure is sufficient to guarantee that the full assembled stiffness matrix in the coarse-scale problem is symmetric positive definite.

2. We propose two types of displacement-to-force mappings: one that directly maps from displacement to forces, and one that maps from displacements to forces by virtue of learning an SPSD stiffness matrix. This second approach is novel. Refs. [11, 25, 34], for example, only consider direct displacement-to-force mappings, and the stiffness matrix associated with these mappings is extracted via, e.g., automatic differentiation. By directly learning a stiffness matrix we can enforce structure in our learned models. Our numerical analysis and experiments demonstrate that this embedded structure results in increased robustness. Additionally, the proposed displacement-to-stiffness mapping addresses the issue of stiffness computation studied in [34]. In our setting the stiffness is a direct output of the neural network and does not need to be probed with finite differences, computed via back propagation, or estimated with an auxiliary network.

This manuscript proceeds as follows. In Section 2 we provide the general formulation for the governing equations of solid mechanics, while in Section 3 we present an equivalent decomposed formulation. Section 4 outlines our machine-learning framework, and Section 5 provides analysis of the coupled FEM-ML systems when applied to the equations of linear elasticity in one dimension. Section 6 provides numerical experiments, while Section 7 provides conclusions and perspectives.

2 Monolithic formulation

We begin by outlining the governing equations for finite-deformation solid mechanics. While we consider physics such as friction and contact in our numerical experiments, these details are not presented here for brevity. We refer the interested reader to the Sierra/SM theory manual [7] for more details.

Consider a body defined on the reference domain $\Omega \subset \mathbb{R}^3$ with boundary Γ undergoing finite deformations described by $\boldsymbol{x} = \boldsymbol{\psi}(\boldsymbol{X})$, where $\boldsymbol{\psi} : \Omega \to \Omega^*$ is the deformation map, $\boldsymbol{X} \in \Omega$ are the reference coordinates, $\boldsymbol{x} \in \Omega^*$ are the deformed coordinates, and $\Omega^* = \boldsymbol{\psi}(\Omega)$ is the deformed domain. In what follows we use $\boldsymbol{\psi} \equiv \boldsymbol{\psi}(\boldsymbol{X})$ for brevity. The governing equations for a quasi-static mechanical problem are given as

$$-\nabla \cdot \boldsymbol{P}(\boldsymbol{\psi}) = \boldsymbol{b} \text{ on } \Omega. \tag{1}$$

In the above, $\boldsymbol{P} : \boldsymbol{\psi} \mapsto J(\boldsymbol{\psi})\boldsymbol{\sigma}(\boldsymbol{\psi})\boldsymbol{F}^{-T}(\boldsymbol{\psi})$ is the first Piola–Kirchhoff stress tensor, $\boldsymbol{F}(\boldsymbol{\psi}) = \nabla\boldsymbol{\psi}$ is the deformation gradient, $J(\boldsymbol{\psi}) = \det(\boldsymbol{F}(\boldsymbol{\psi}))$, $\boldsymbol{\sigma}(\boldsymbol{\psi})$ is the Cauchy stress tensor, and $\boldsymbol{b} : \Omega \to \mathbb{R}^3$ is the body forcing per unit reference volume. We emphasize that the first Piola-Kirchoff stress tensor is a nonlinear function of the deformation map and contains the constitutive relationship. We assume the domain boundary Γ can be decomposed as $\Gamma = \Gamma_{\boldsymbol{\psi}} \cup \Gamma_{\boldsymbol{P}}$ with $\Gamma_{\boldsymbol{\psi}} \cap \Gamma_{\boldsymbol{P}} = \emptyset$, where $\Gamma_{\boldsymbol{\psi}}$ comprises a Dirichlet boundary and $\Gamma_{\boldsymbol{P}}$ a traction boundary such that Eq. (1) is supplemented with the boundary conditions

$$\boldsymbol{P}(\boldsymbol{\psi}) \cdot \mathbf{n} = \boldsymbol{T}_{\Gamma} \text{ on } \Gamma_{\boldsymbol{P}}$$
$$\boldsymbol{\psi} = \boldsymbol{\psi}_{\Gamma} \text{ on } \Gamma_{\boldsymbol{\psi}}.$$

5

In the above, $\boldsymbol{T}_\Gamma$ denotes the prescribed boundary tractions, $\boldsymbol{\psi}_\Gamma$ the prescribed Dirichlet boundary conditions, and $\mathbf{n}$ defines the normal vector to the boundary Ω.

We consider the standard weighted residual form of Eq. (1), which reads as follows: find $\boldsymbol{\psi} \in \mathcal{V}$ such that

$$\int_\Omega \boldsymbol{P}\left(\boldsymbol{\psi}\right) : \nabla \boldsymbol{v}\, dV - \int_{\Gamma_{\boldsymbol{P}}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}\right) \cdot \mathbf{n}\right) \cdot \boldsymbol{v}\, dS = \int_\Omega \boldsymbol{b} \cdot \boldsymbol{v}\, dV, \qquad \forall \boldsymbol{v} \in \mathcal{W}, \qquad (2)$$

where $\mathcal{V}$ denotes a suitable vector-valued trial space on Ω satisfying the Dirichlet boundary condition $\boldsymbol{\psi}_\Gamma$ on $\Gamma_{\boldsymbol{\psi}}$ and $\mathcal{W}$ denotes a suitable vector-valued test space on Ω vanishing on $\Gamma_{\boldsymbol{\psi}}$. The Galerkin method with Lagrange finite elements comprises a standard approach to solve Eq. (2). To this end, let $\mathcal{V}_\mathrm{h} \subset \mathcal{V}$ denote a conforming trial space obtained via a Lagrange finite element discretization of the reference domain Ω into N_el non-overlapping elements $\Omega_k, k = 1, \ldots, N_\mathrm{el}$, and let $\mathcal{W}_\mathrm{h}$ be the corresponding test space. The spatially discrete counterpart to Eq. (2) reads: find $\boldsymbol{\psi}_\mathrm{h} \in \mathcal{V}_\mathrm{h}$ such that

$$\int_\Omega \boldsymbol{P}\left(\boldsymbol{\psi}_\mathrm{h}\right) : \nabla \boldsymbol{v}_\mathrm{h}\, dV - \int_{\Gamma_{\boldsymbol{P}}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}_\mathrm{h}\right) \cdot \mathbf{n}\right) \cdot \boldsymbol{v}_\mathrm{h}\, dS = \int_\Omega \boldsymbol{b} \cdot \boldsymbol{v}_\mathrm{h}\, dV, \qquad \forall \boldsymbol{v}_\mathrm{h} \in \mathcal{W}_\mathrm{h}. \quad (3)$$

Equation (3) comprises a monolithic set of governing equations. It is often the case, however, that the body Ω is made up of multiple subdomains, where each subdomain may require, e.g., different resolution requirements. One such example that motivates the present work is systems-level FEM models comprising threaded fasteners. In this case, the threaded fastener model can require a much finer discretization than the remainder of the domain and can be a computational bottleneck. To mitigate this issue we propose a data-driven modeling approach.

3 Domain decomposition formulation

We consider a decomposition of the reference domain Ω into two non-overlapping components such that $\Omega = \overline{\Omega} \cup \Omega'$ with the interface $\Gamma_\mathcal{I} = \overline{\Omega} \cap \Omega'$ as in Figure 2. We take $\overline{\Omega}$ to be an "outer" coarse-scale domain which we aim to solve directly with a finite element method, while we take Ω' to be an "inner" fine-scale domain. We note that we assume the inner domain to be completely enclosed by the outer domain such that $\Gamma_\mathcal{I} \cap \Gamma = \emptyset$. We aim to not resolve Ω', but rather model its impact on $\overline{\Omega}$. To this end, we introduce a decomposition of the deformation map on $\overline{\Omega}$ and Ω'

$$\boldsymbol{\psi} = \overline{\boldsymbol{\psi}} + \boldsymbol{\psi}'$$

such that $\overline{\boldsymbol{\psi}} = \boldsymbol{0}$ on Ω', $\boldsymbol{\psi}' = \boldsymbol{0}$ on $\overline{\Omega}$, and $\overline{\boldsymbol{\psi}} = \boldsymbol{\psi}' \equiv \boldsymbol{\psi}$ on $\Gamma_\mathcal{I}$. We note that, by definition, $\boldsymbol{P}(\overline{\boldsymbol{\psi}} + \boldsymbol{\psi}') = \boldsymbol{P}(\overline{\boldsymbol{\psi}})$ on $\overline{\Omega}$ and $\boldsymbol{P}(\overline{\boldsymbol{\psi}} + \boldsymbol{\psi}') = \boldsymbol{P}(\boldsymbol{\psi}')$ on Ω'. With this decomposition, a coarse-scale problem on $\overline{\Omega}$ can be written as: find $\overline{\boldsymbol{\psi}} \in \overline{\mathcal{V}}$ such that $\forall \overline{\boldsymbol{v}} \in \overline{\mathcal{W}}$

$$\int_{\overline{\Omega}} \boldsymbol{P}\left(\overline{\boldsymbol{\psi}}\right) : \nabla \overline{\boldsymbol{v}}\, dV - \int_{\Gamma_{\boldsymbol{P}}} \left(\boldsymbol{P}\left(\overline{\boldsymbol{\psi}}\right) \cdot \mathbf{n}\right) \cdot \overline{\boldsymbol{v}}\, dS = \int_{\overline{\Omega}} \boldsymbol{b} \cdot \overline{\boldsymbol{v}}\, dV - \int_{\Gamma_\mathcal{I}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}'\right) \cdot \mathbf{n}_\mathcal{I}\right) \cdot \overline{\boldsymbol{v}}\, dS. \ (4)$$
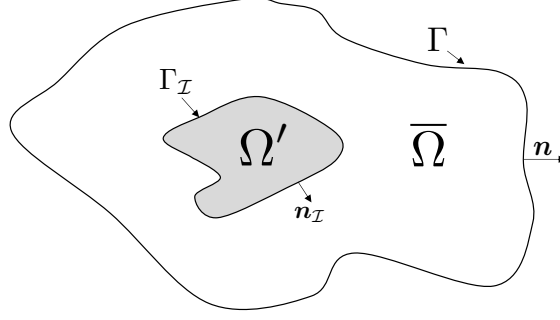
6

Fig. 2: Simplified depiction of domain-decomposition formulation.

In the above, $\overline{\mathcal{V}}$ and $\overline{\mathcal{W}}$ are suitable vector-valued trial and test spaces on $\overline{\Omega}$, $\mathbf{n}_{\mathcal{I}}$ is the outward normal vector on the interface $\Gamma_{\mathcal{I}}$ (see Figure 2), and continuity of the normal stress between the inner and outer domain is enforced weakly. Our goal is to solve the coarse-scale equation. Let $\overline{\mathcal{V}}_{\mathrm{h}} \subset \overline{\mathcal{V}}$ denote a conforming trial space obtained via a Lagrange finite element discretization of the domain $\overline{\Omega}$ into $\overline{N}_{\mathrm{el}}$ non-overlapping elements $\overline{\Omega}_k, k = 1, \ldots, \overline{N}_{\mathrm{el}}$ with the node set $\overline{\mathcal{N}}$, $|\overline{\mathcal{N}}| = \overline{N}$, and let $\overline{\mathcal{W}}_{\mathrm{h}}$ denote the corresponding test space. For notational purposes, we additionally denote the interface boundary node set as $\overline{\mathcal{N}}^{\Gamma_{\mathcal{I}}} \subset \overline{\mathcal{N}}$ with $|\overline{\mathcal{N}}^{\Gamma_{\mathcal{I}}}| = \overline{N}^{\Gamma_{\mathcal{I}}}$. The spatially discrete counterpart to Eq. (4) reads: find $\overline{\boldsymbol{\psi}}_{\mathrm{h}} \in \overline{\mathcal{V}}_{\mathrm{h}}$ such that $\forall \overline{\boldsymbol{v}}_{\mathrm{h}} \in \overline{\mathcal{W}}_{\mathrm{h}}$

$$
\int_{\overline{\Omega}} \boldsymbol{P}\left(\overline{\boldsymbol{\psi}}_{\mathrm{h}}\right) : \nabla \overline{\boldsymbol{v}}_{\mathrm{h}} dV - \int_{\Gamma_{\boldsymbol{P}}} \left(\boldsymbol{P}\left(\overline{\boldsymbol{\psi}}_{\mathrm{h}}\right) \cdot \mathbf{n}\right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}} dS =
$$
$$
\int_{\overline{\Omega}} \boldsymbol{b} \cdot \overline{\boldsymbol{v}}_{\mathrm{h}} dV - \int_{\Gamma_{\mathcal{I}}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}'\right) \cdot \mathbf{n}_{\mathcal{I}}\right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}} dS. \tag{5}
$$

The above can be written as a set of $\overline{N}_{\mathrm{dof}} = 3\overline{N}$ algebraic equations by introducing the vector-valued trial basis functions $\overline{\boldsymbol{\phi}}_{\mathrm{h}i}, i = 1, \ldots, \overline{N}$ and vector-valued test basis functions $\overline{\boldsymbol{v}}_{\mathrm{h}i}, i = 1, \ldots, \overline{N}$ such that $\mathrm{span}\{\overline{\boldsymbol{\phi}}_{\mathrm{h}i}\}_{i=1}^{\overline{N}} = \overline{\mathcal{V}}_{\mathrm{h}}$, $\mathrm{span}\{\overline{\boldsymbol{v}}_{\mathrm{h}i}\}_{i=1}^{\overline{N}} = \overline{\mathcal{W}}_{\mathrm{h}}$. The system (5) simplifies to a force balance at each node in the case of Lagrange finite elements. We observe that the system (5) is unclosed as it requires specification of the boundary term $\left(\boldsymbol{P}\left(\boldsymbol{\psi}'\right) \cdot \mathbf{n}_{\mathcal{I}}\right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}}$, i.e., we need to specify the forcing exhibited by the inner domain on the outer domain. We employ a machine-learning approach for this purpose.

Before proceeding, we comment on the assumption $\Gamma_{\mathcal{I}} \cup \Gamma = \emptyset$. Without this term, Eq. (5) would have an additional unclosed term corresponding to prescribed tractions on $\Gamma_{\mathcal{I}} \cap \Gamma_{\boldsymbol{P}}$. While it is possible to construct a data-driven approach that handles this case, it requires considerations such as how to parameterize the tractions. We leave this as future work.

4 Machine-learning framework

Developing a method that only requires solving the coarse-scale system requires a model for the impact of the fine scales on the coarse scales. To this end, we propose a coupled FEM-ML formulation wherein the ML model is used to describe the impact of the fine scales on the coarse scales at the interface boundary. This section outlines our coupled FEM-ML formulation, describes the offline-online approach employed to construct the ML model, and lastly outlines several candidate machine-learning models.

4.1 FEM-ML coupled formulation

Closing the coarse-scale equation requires specification of the traction applied by the fine scales on the domain boundary, i.e., we require a model for the term $\int_{\Gamma_{\mathcal{I}}} \left(\boldsymbol{P}' \cdot \mathbf{n}_{\mathcal{I}} \right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}} dS$. We propose a machine learning approach for this purpose, i.e., we propose to develop a machine learning model $\boldsymbol{\mathcal{M}}(\cdot) = \left[\mathcal{M}_1(\cdot)^T, \ldots, \mathcal{M}_{\overline{N}}(\cdot)^T \right]^T \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}}$ such that

$$\mathcal{M}_i(\boldsymbol{\theta}) \approx \int_{\Gamma_{\mathcal{I}}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}' \right) \cdot \mathbf{n}_{\mathcal{I}} \right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}i} dS \in \mathbb{R}^3,$$

for $i = 1, \ldots, \overline{N}$. In the above, $\mathcal{M}_i(\boldsymbol{\theta})$ denotes the ML model output for the ith test function for the 3 DOFs (i.e., DOFs in the x_1, x_2, and x_3 directions) and $\boldsymbol{\theta}$ are the to-be-defined features fed into the ML model. The coarse-scale equation then becomes: find $\overline{\boldsymbol{\psi}}_{\mathrm{h}} \in \overline{\mathcal{V}}_{\mathrm{h}}$ such that for $i = 1, \ldots, \overline{N}$,

$$\int_{\overline{\Omega}} \boldsymbol{P}\left(\overline{\boldsymbol{\psi}}_{\mathrm{h}} \right) : \nabla \overline{\boldsymbol{v}}_{\mathrm{h}i} dV - \int_{\Gamma_{\boldsymbol{P}}} \left(\boldsymbol{P}\left(\overline{\boldsymbol{\psi}}_{\mathrm{h}} \right) \cdot \mathbf{n} \right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}i} dS + \mathcal{M}_i(\boldsymbol{\theta}) = \int_{\overline{\Omega}} \boldsymbol{b} \cdot \overline{\boldsymbol{v}}_{\mathrm{h}i} dV. \quad (6)$$

4.2 Offline-online workflow

To develop the data-driven model, we employ an *offline–online* workflow. In the offline phase, we perform a computationally intensive process that involves solving the *full* finite element model (3) for N_s different configurations (e.g., different boundary conditions) to obtain N_s deformation maps $\{\boldsymbol{\psi}_{\mathrm{h}}^i\}_{i=1}^{N_s}$, where N_s are the number of training samples. These sample deformation maps are then processed to compute the target training matrix $\mathbf{F} \in \mathbb{R}^{\overline{N}_{\mathrm{dof}} \times N_s}$ comprising the terms $\int_{\Gamma_{\mathcal{I}}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}_{\mathrm{h}}^j \right) \cdot \mathbf{n} \right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}i} dS$ for $i = 1, \ldots, \overline{N}$, $j = 1, \ldots, N_s$. Analogously, we collect the feature training matrix $\mathbf{U} \in \mathbb{R}^{N_{\boldsymbol{\theta}} \times N_s}$, where $N_{\boldsymbol{\theta}}$ are the number of features. In this work we consider the case where the features are the nodal values of the displacement fields, i.e.,

$$\boldsymbol{\theta} = \overline{\mathbf{u}}_{\mathrm{h}} \equiv \left[\left[\overline{\boldsymbol{\psi}}_{\mathrm{h}}(\boldsymbol{X}_1^{\overline{\Omega}}) - \boldsymbol{X}_1^{\overline{\Omega}} \right]^T \cdots \left[\overline{\boldsymbol{\psi}}_{\mathrm{h}}(\boldsymbol{X}_{\overline{N}}^{\overline{\Omega}}) - \boldsymbol{X}_{\overline{N}}^{\overline{\Omega}} \right]^T \right]^T \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}},$$

where $\boldsymbol{X}_i^{\overline{\Omega}}$, $i = 1, \ldots, \overline{N}$ denotes coordinates of the ith coarse node.

4.2.1 Simplification for Lagrange finite elements

In the present work we exclusively consider application to Lagrange finite elements. In this case the term $\int_{\Gamma_\mathcal{I}} \left(\boldsymbol{P}\left(\boldsymbol{\psi}'\right) \cdot \mathbf{n} \right) \cdot \overline{\boldsymbol{v}}_{\mathrm{h}i} dS$ is exactly zero for basis functions not on the interface $\Gamma_\mathcal{I}$ and we only need to construct a model for basis functions on the interface such that $\boldsymbol{\mathcal{M}}(\cdot) \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}}}$, where $\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} = 3\overline{N}^{\Gamma_\mathcal{I}}$. In this case we need only collect the target training matrix for the $\overline{N}^{\Gamma_\mathcal{I}}$ basis functions on the interface. This results in a target training matrix of size $\mathbf{F} \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} \times N_s}$. Analogously, for features we employ only the nodal values of the displacement fields on the interface, i.e., $\boldsymbol{\theta} = \overline{\mathbf{u}}_{\mathrm{h}}^{\Gamma_\mathcal{I}}$ such that $\mathbf{U} \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} \times N_s}$. This approach will be adapted for the remainder of the manuscript.

4.3 Machine-learning model structure

Having defined the feature matrix $\mathbf{U}$ and the response matrix $\mathbf{F}$ we now outline two candidate approaches for the ML model $\boldsymbol{\mathcal{M}}$.

Approach 1: Direct mapping with proper orthogonal decomposition. The first approach we consider comprises learning a direct feature-to-response mapping with the addition of POD to reduce the dimensionality of the input feature space and the response space. This approach can be described as

$$\boldsymbol{\mathcal{M}} : \overline{\mathbf{u}}_{\mathrm{h}}^{\Gamma_\mathcal{I}} \mapsto \boldsymbol{\Phi}_{\boldsymbol{f}} \hat{\boldsymbol{\mathcal{M}}} \left(\boldsymbol{\Phi}_{\boldsymbol{u}}^T \overline{\mathbf{u}}_{\mathrm{h}}^{\Gamma_\mathcal{I}} ; \mathbf{w} \right) + \mathbf{f}_0,$$

where $\mathbf{f}_0 \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}}}$ is a constant vector (accounting, e.g., for body forces, preload), $\boldsymbol{\Phi}_{\boldsymbol{f}} \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} \times K_{\boldsymbol{f}}}$ and $\boldsymbol{\Phi}_{\boldsymbol{u}} \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} \times K_{\boldsymbol{u}}}$ are *basis* matrices for the interface forces and displacements, respectively, obtained through POD, $K_{\boldsymbol{u}}$ and $K_{\boldsymbol{f}}$ are the reduced dimension for the displacements and forces, both of which are model hyperparameters, $\hat{\boldsymbol{\mathcal{M}}} : \mathbb{R}^{K_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\mathbf{w}}} \to \mathbb{R}^{K_{\boldsymbol{f}}}$ comprises a machine-learned model that maps from reduced displacement states to reduced force states, and $\mathbf{w} \in \mathbb{R}^{N_{\mathbf{w}}}$ are model parameters (obtained from training). The basis matrices for forces and displacements are obtained by solving the optimization problems

$$\boldsymbol{\Phi}_{\boldsymbol{f}} = \underset{\boldsymbol{\Phi}_{\boldsymbol{f}}^* \in \{\mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} \times K_{\boldsymbol{f}}} | [\boldsymbol{\Phi}_{\boldsymbol{f}}^*]^T \boldsymbol{\Phi}_{\boldsymbol{f}}^* = \mathbf{I}\}}{\arg\min} \ \|\boldsymbol{\Phi}_{\boldsymbol{f}}^* [\boldsymbol{\Phi}_{\boldsymbol{f}}^*]^T \mathbf{F}^* - \mathbf{F}^*\|_2^2, \tag{7}$$

$$\boldsymbol{\Phi}_{\boldsymbol{u}} = \underset{\boldsymbol{\Phi}_{\boldsymbol{u}}^* \in \{\mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_\mathcal{I}} \times K_{\boldsymbol{u}}} | [\boldsymbol{\Phi}_{\boldsymbol{u}}^*]^T \boldsymbol{\Phi}_{\boldsymbol{u}}^* = \mathbf{I}\}}{\arg\min} \ \|\boldsymbol{\Phi}_{\boldsymbol{u}}^* [\boldsymbol{\Phi}_{\boldsymbol{u}}^*]^T \mathbf{U} - \mathbf{U}\|_2^2, \tag{8}$$

where $\mathbf{F}^* = \mathbf{F} - [\mathbf{f}_0 \ \cdots \ \mathbf{f}_0]$. These optimization problems can be solved efficiently with a (thin) singular value decomposition (SVD) [36]. The algorithm is described in Appendix A. We emphasize that by employing POD, the training of the model scales with the reduced dimensions $K_{\boldsymbol{u}}$ and $K_{\boldsymbol{f}}$ rather than the number of interface nodes.

A variety of regression models can be used to learn the reduced model $\hat{\boldsymbol{\mathcal{M}}}$. Here we investigate two approaches spanning different regimes of model complexity: linear regression and neural networks. We investigate linear regression due to its simplicity

and interpretability, while we investigate neural networks due to their high capacity to learn complex functions. These two strategies are now described.

- **Linear least squares (LLS)**. In a linear regression approach the reduced ML model is defined by a matrix $\hat{\mathbf{A}} \in \mathbb{R}^{K_{\boldsymbol{f}} \times K_{\boldsymbol{u}}}$ such that the full model can be written as

$$\mathcal{M} : \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} \mapsto \boldsymbol{\Phi}_{\boldsymbol{f}} \hat{\mathbf{A}} \boldsymbol{\Phi}_{\boldsymbol{u}}^{T} \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} + \mathbf{f}_0.$$

The reduced system matrix is defined by the solution to the linear least-squares problem

$$\hat{\mathbf{A}} = \underset{\hat{\mathbf{A}}^* \in \mathbb{R}^{K_{\boldsymbol{f}} \times K_{\boldsymbol{u}}}}{\arg\min} \|\boldsymbol{\Phi}_{\boldsymbol{f}}^{T} \mathbf{F}^* - \hat{\mathbf{A}}^* \boldsymbol{\Phi}_{\boldsymbol{u}}^{T} \mathbf{U}\|_2^2. \tag{9}$$

The least-squares problem can be efficiently solved as it can be broken down into $K_{\boldsymbol{f}}$ least-squares problems for each row of $\hat{\mathbf{A}}$.

- **Neural network (NN)** This method learns a feed forward neural network for the mapping from the (reduced) displacement field to the (reduced) force field. In this case the ML model is given by

$$\mathcal{M} : \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} \mapsto \boldsymbol{\Phi}_{\boldsymbol{f}} \mathcal{N}\left(\boldsymbol{\Phi}_{\boldsymbol{u}}^{T} \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right) + \mathbf{f}_0$$

where $\mathcal{N} : \mathbb{R}^{K_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\mathbf{w}}} \to \mathbb{R}^{K_{\boldsymbol{f}}}$ comprises a neural network model that maps from the reduced displacement states to the reduced force states and $\mathbf{w} \in \mathbb{R}^{N_{\mathbf{w}}}$ are the network weights. In the present work, these network weights are obtained from approximate minimization of the loss function,

$$\underset{\mathbf{w} \in \mathbb{R}^{N_{\mathbf{w}}}}{\text{minimize}} \sum_{i=1}^{N_s} \|\mathcal{N}\left(\boldsymbol{\Phi}_{\boldsymbol{u}}^{T} \mathbf{U}_i; \mathbf{w}\right) - \boldsymbol{\Phi}_{\boldsymbol{f}}^{T} \mathbf{F}_i^*\|_2^2.$$

Additional details on model training are given in Section 6.2.

Approach 2: Stiffness matrix mapping with proper orthogonal decomposition. The approach outlined above is the simplest learning framework one can employ. This approach, however, does not directly enforce any physical or mathematical properties of the underlying FEM. As will be shown later in this manuscript, we found that embedding the above models into an FEM code often led to unstable solutions. To address this challenge, we propose a second approach that embeds structure into the model by learning an SPSD stiffness matrix. We again use POD to reduce the dimensionality of the input feature and response spaces. This approach can be described as

$$\mathcal{M} : \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} \mapsto \boldsymbol{\Phi}_* \left[\hat{\mathcal{L}}\left(\boldsymbol{\Phi}_*^{T} \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right)\right] \left[\hat{\mathcal{L}}\left(\boldsymbol{\Phi}_*^{T} \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right)\right]^{T} \boldsymbol{\Phi}_*^{T} \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} + \mathbf{f}_0 \tag{10}$$

where $\hat{\mathcal{L}} : \mathbb{R}^{K_*} \times \mathbb{R}^{N_{\mathbf{w}}} \to \mathbb{R}^{K_* \times K_*}$ comprises a machine-learned model that maps from the reduced displacement states to the lower triangular part of a reduced stiffness matrix, and $\boldsymbol{\Phi}_* \in \mathbb{R}^{\overline{N}_{\text{dof}}^{\Gamma_{\mathcal{I}}} \times (K_{\boldsymbol{f}} + K_{\boldsymbol{u}})} = \textit{orthogonalize}\left([\boldsymbol{\Phi}_{\boldsymbol{f}}, \boldsymbol{\Phi}_{\boldsymbol{u}}]\right)$ is a basis matrix combining the (orthogonalized) union of the force basis and displacement basis. We note the orthogonalization can be performed efficiently with a QR decomposition.

Critically, Eq. (10) can be written in matrix form as

$$\mathcal{M}\left(\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}\right) = \mathbf{K}_{\mathrm{ML}}\left(\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}\right)\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} + \mathbf{f}_0,$$

where $\mathbf{K}_{\mathrm{ML}}(\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}) = [\mathbf{K}_{\mathrm{ML}}(\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}})]^T \in \mathbb{R}^{\overline{N}_{\mathrm{dof}}^{\Gamma_{\mathcal{I}}} \times \overline{N}_{\mathrm{dof}}^{\Gamma_{\mathcal{I}}}}$ is an SPSD stiffness matrix. This structure is directly enforced in the model by learning the lower triangular matrix $\hat{\mathcal{L}}$. Enforcing our model to be SPSD makes the coupled FEM-ML system more amenable for conjugate gradient solvers, which are commonly employed in solid mechanics. Lastly, we note the use of the combined basis $\mathbf{\Phi}_*$ for both the force and displacement field, opposed to the individual bases employed in Approach 1. The reason for employing the combined basis is that both the force field and displacement field must employ the same basis for the model (10) to be SPSD[2]. In the present work we achieve this by combining both the bases for the force field and displacement fields. For simplicity we consider the case where $K_* = K_{\boldsymbol{u}} + K_{\boldsymbol{f}}$; it is possible that a more compact representation can be obtained by considering a single POD problem where the displacements and forces are stacked.

A variety of regression models can again be used to learn the reduced stiffness matrix $\hat{\mathcal{L}}$. We again consider a linear regression approach and a neural-network-based approach.

- **Symmetric positive semi-definite linear least-squares (SPSD-LLS)**. This approach learns a linear model for the stiffness matrix. This method is described by

$$\mathcal{M} : \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} \mapsto \mathbf{\Phi}_*\hat{\mathbf{L}}\hat{\mathbf{L}}^T\mathbf{\Phi}_*^T\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} + \mathbf{f}_0$$

where $\hat{\mathbf{L}} \in \mathbb{R}^{K_* \times K_*}$ is defined by the optimization problem

$$\hat{\mathbf{L}} = \underset{\hat{\mathbf{L}}^* \in \mathbb{R}^{K_* \times K_*}}{\arg\min} \|\mathbf{\Phi}_*^T\mathbf{F}^* - \hat{\mathbf{L}}^*\left[\hat{\mathbf{L}}^*\right]^T\mathbf{\Phi}_*^T\mathbf{U}\|_2^2. \tag{11}$$

By definition, SPSD-LLS learns a system matrix that is SPSD. To the best of our knowledge, no analytic solution exists to the optimization problem (11), but it can be solved in a straightforward manner by casting it as a constrained optimization problem. In the present work we solve the optimization problem (11) with the optimization package cvxpy [2, 16].

- **Symmetric positive semi-definite neural network regression (SPSD-NN)** This approach learns a feed forward neural network for the mapping from the (reduced) displacement field to the (reduced) lower triangular stiffness matrix. In this case the ML model is given by

$$\mathcal{M} : \overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} \mapsto \mathbf{\Phi}_*\left[\mathcal{L}_{\mathcal{N}\mathcal{N}}\left(\mathbf{\Phi}_*^T\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right)\right]\left[\mathcal{L}_{\mathcal{N}\mathcal{N}}\left(\mathbf{\Phi}_*^T\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right)\right]^T\mathbf{\Phi}_*^T\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}} + \mathbf{f}_0 \tag{12}$$

---

[2]For example, if we employ different bases for the force and displacement, then stiffness matrix associated with (10) would take the form $\mathbf{\Phi}_{\boldsymbol{f}}\left[\hat{\mathcal{L}}\left(\mathbf{\Phi}_{\boldsymbol{u}}^T\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right)\right]\left[\hat{\mathcal{L}}\left(\mathbf{\Phi}_{\boldsymbol{u}}^T\overline{\mathbf{u}}_{\mathsf{h}}^{\Gamma_{\mathcal{I}}}; \mathbf{w}\right)\right]^T\mathbf{\Phi}_{\boldsymbol{u}}^T$ which is not symmetric.

where $\mathcal{L}_{\mathcal{NN}} : \mathbb{R}^{K_*} \times \mathbb{R}^{N_{\mathbf{w}}} \to \mathbb{R}^{K_* \times K_*}$ is the neural network model for the reduced lower stiffness matrix. The network weights are obtained from approximate minimization of the loss function,

$$\underset{\mathbf{w} \in \mathbb{R}^{N_{\mathbf{w}}}}{\text{minimize}} \sum_{i=1}^{N_s} \| \left[\mathcal{L}_{\mathcal{NN}} \left(\mathbf{\Phi}_*^T \mathbf{U}_i; \mathbf{w} \right) \right] \left[\mathcal{L}_{\mathcal{NN}} \left(\mathbf{\Phi}_*^T \mathbf{U}_i; \mathbf{w} \right) \right]^T \mathbf{\Phi}_*^T \mathbf{U}_i - \mathbf{\Phi}_*^T \mathbf{F}_i^* \|_2^2.$$

In summary, we considered four model forms: a linear model (LLS) and nonlinear (NN) model that directly map from the interface displacements to the interface forces, as well as a linear model (SPSD-LLS) and nonlinear model (SPSD-NN) that directly map from interface displacements to an SPSD stiffness matrix.

Remark 4.1. *We emphasize that the above approaches learn the mapping between displacements and forces with the purpose of removing a subdomain from the computational domain. A similar approach could be to learn, e.g., the strain–stress mapping on the domain boundary. This type of approach was employed in Refs. [4, 37], which leverage neural networks to learn the constitutive relationship defining the stress-strain relationship at every point a computational domain. In the present work we directly pursue a displacement–force mapping for ease of implementation into our finite element solver. Specifically, the contribution of our ML model simply appears as an additional (state-dependent) source term in the residual of the governing equations (as opposed to, e.g., a stress field that needs to be integrated).*

Remark 4.2. *The above formulations are designed for path-independent static problems and will be incomplete for dynamic path-dependent problems (e.g., plasticity) where history effects can be important.*

5 Positive definiteness of coarse-scale model in a simplified setting

In this section we demonstrate that, in the simplified setting of linear elasticity in one-dimension with homogeneous boundary conditions, the SPSD stiffness-based models result in a global coarse scale system that is SPD and, as a result, have invertible system matrices with unique solutions. We further show that, for the direct displacement-to-force model, we cannot guarantee an invertible system with a unique solution.

The governing equations of linear elasticity in one dimension are

$$-AE\frac{\partial^2 u}{\partial x^2} = b \tag{13}$$

for $x \in [0, L]$ where $u : [0, L] \to \mathbb{R}$ is the displacement with boundary conditions $u(0) = u(L) = 0$. In the above, $A \in \mathbb{R}^+$ is the cross-sectional area, $E \in \mathbb{R}^+$ is the modulus of elasticity, and $b \in \mathbb{R}$ is a constant body forcing term. Discretization of (13) into N degrees of freedom with $N + 1$ uniform first-order Lagrange finite elements

results in the system for the interior DOFs,

$$\mathbf{K}\mathbf{u}_h = \mathbf{b} \tag{14}$$

where

$$\mathbf{K} = \frac{AE}{\Delta x} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \qquad \mathbf{b} = \Delta x \begin{bmatrix} b \\ b \\ \vdots \\ b \\ b \end{bmatrix}$$

with $\Delta x = \frac{L}{N+1}$. We note that the system is SPD, i.e., $\mathbf{u}_h^T \mathbf{K}\mathbf{u}_h > 0$, and as such is invertible such that the system (14) has a unique solution.

We now consider the domain decomposition formulation such that the first two and last two nodes are the "coarse-scale" nodes and the remaining inner nodes are the "fine-scale" nodes, i.e., $\overline{\mathbf{u}}_h = [u_{h1}, u_{h2}, u_{hN-1}, u_{hN}]^T \in \mathbb{R}^4$ and $\mathbf{u}_h' = [u_{h3}, \cdots, u_{hN-2}]^T \in \mathbb{R}^{N-4}$ with boundary degrees of freedom $\overline{\mathbf{u}}_h^{\Gamma_{\mathcal{I}}} = [u_{h2}, u_{hN-1}]^T \in \mathbb{R}^2$. The resulting coarse-scale problem is

$$\overline{\mathbf{K}}\overline{\mathbf{u}}_h + \overline{\mathbf{s}} = \overline{\mathbf{b}} \tag{15}$$

where

$$\overline{\mathbf{K}} = \frac{AE}{\Delta x} \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \qquad \overline{\mathbf{b}} = \Delta x \begin{bmatrix} b \\ b \\ b \\ b \end{bmatrix},$$

are the coarse-scale stiffness matrices and forcing vectors and

$$\overline{\mathbf{s}} = \frac{AE}{\Delta x} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \overline{u}_{h2} \\ u'_{h1} \\ u'_{hN-4} \\ \overline{u}_{h3} \end{bmatrix}$$

is the contribution of the interior degrees of freedom on the exterior degrees of freedom at the interface. We emphasize that this term depends on the (unknown) interior degrees of freedom. We further emphasize that the coarse-scale stiffness matrix is SPD such that $\overline{\mathbf{u}}_h^T \overline{\mathbf{K}}\overline{\mathbf{u}}_h > 0$.

Replacing $\overline{\mathbf{s}}$ from Eq. (15) with one of the machine-learned models $\mathcal{M}$ outlined in Section 4 results in the coarse-scale problem

$$\overline{\mathbf{K}}\overline{\mathbf{u}}_h + \mathcal{M}(\overline{\mathbf{u}}_h^{\Gamma_{\mathcal{I}}}) = \overline{\mathbf{b}}. \tag{16}$$

We now provide a brief analysis of the models outlined in Section 4.

- **Linear least-squares.** For this model the reduced system becomes

$$\left[\overline{\mathbf{K}} + \mathbf{K}_{\mathsf{ML}}\right]\overline{\mathbf{u}}_h = \overline{\mathbf{b}} - \mathbf{f}_0 \tag{17}$$

13

where
$$\mathbf{K}_{\mathsf{ML}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \left[\boldsymbol{\Phi_f}\hat{\mathbf{A}}\boldsymbol{\Phi_u^T}\right]_{11} & \left[\boldsymbol{\Phi_f}\hat{\mathbf{A}}\boldsymbol{\Phi_u^T}\right]_{12} & 0 \\ 0 & \left[\boldsymbol{\Phi_f}\hat{\mathbf{A}}\boldsymbol{\Phi_u^T}\right]_{21} & \left[\boldsymbol{\Phi_f}\hat{\mathbf{A}}\boldsymbol{\Phi_u^T}\right]_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

As we place no constraints on the construction of $\hat{\mathbf{A}}$, we cannot guarantee that $\mathbf{K}_{\mathsf{ML}}$ is positive (semi-)definite, nor can we guarantee a unique solution to the system (17) without knowledge of the operator $\boldsymbol{\Phi_f}\hat{\mathbf{A}}\boldsymbol{\Phi_u^T}$, which in general is not full rank. As a canonical example, we can consider the case where

$$\boldsymbol{\Phi_f}\hat{\mathbf{A}}\boldsymbol{\Phi_u^T} = \begin{bmatrix} -0.5 & 0 \\ 0 & -0.5 \end{bmatrix},$$

which is an admissible solution to the optimization problem (9). In this case the coarse-scale equation stiffness matrix becomes

$$\overline{\mathbf{K}} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

which is singular. We further observe that, in general, the ML contribution will *not* be symmetric. Thus, even if invertible, the system (17) cannot in general be solved with the popular conjugate gradient method.

- **Neural network.** The neural network model results in the coarse-scale equation

$$\overline{\mathbf{K}}\overline{\mathbf{u}}_\mathsf{h} + \boldsymbol{\Phi_f}\mathcal{N}\left(\boldsymbol{\Phi_u^T}\overline{\mathbf{u}}_\mathsf{h}^{\Gamma_{\mathcal{I}}};\mathbf{w}\right) = \overline{\mathbf{b}} - \mathbf{f}_0. \tag{18}$$

Similar to the linear least-squares model, we cannot guarantee that the Newton iteration associated with (18) will involve full-rank matrices. In general the matrices will not be SPD and solution via the conjugate gradient method is inappropriate.

- **Symmetric positive semi-definite linear-least-squares** The SPSD-LLS model results in the reduced system

$$\left[\overline{\mathbf{K}} + \mathbf{K}_{\mathsf{ML}}\right]\overline{\mathbf{u}}_\mathsf{h} = \overline{\mathbf{b}} - \mathbf{f}_0 \tag{19}$$

where
$$\mathbf{K}_{\mathsf{ML}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \left[\boldsymbol{\Phi_*}\hat{\mathbf{L}}\hat{\mathbf{L}}^T\boldsymbol{\Phi_*^T}\right]_{11} & \left[\boldsymbol{\Phi_*}\hat{\mathbf{L}}\hat{\mathbf{L}}^T\boldsymbol{\Phi_*^T}\right]_{12} & 0 \\ 0 & \left[\boldsymbol{\Phi_*}\hat{\mathbf{L}}\hat{\mathbf{L}}^T\boldsymbol{\Phi_*^T}\right]_{21} & \left[\boldsymbol{\Phi_*}\hat{\mathbf{L}}\hat{\mathbf{L}}^T\boldsymbol{\Phi_*^T}\right]_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

By construction, $\overline{\mathbf{u}}_\mathsf{h}^T\mathbf{K}_{\mathsf{ML}}\overline{\mathbf{u}}_\mathsf{h} \geq 0$ and, as a result, $\overline{\mathbf{u}}_\mathsf{h}^T\left[\overline{\mathbf{K}} + \mathbf{K}_{\mathsf{ML}}\right]\overline{\mathbf{u}}_\mathsf{h} > 0$. Thus, the coarse-scale system is SPD. By definition of SPD matrices, the system is invertible with a unique solution and may be solved with the conjugate gradient method.

14

- **Symmetric positive semi-definite neural network**. The SPSD-NN model results in the reduced system

$$\left[\overline{\mathbf{K}} + \mathbf{K}_{\mathsf{ML}}(\overline{\mathbf{u}}_{\mathsf{h}}) \right] \overline{\mathbf{u}}_{\mathsf{h}} = \overline{\mathbf{b}} - \mathbf{f}_0 \tag{20}$$

where

$$\mathbf{K}_{\mathsf{ML}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \left[\boldsymbol{\Phi}_* \boldsymbol{\mathcal{L}}_{\mathcal{NN}} \boldsymbol{\mathcal{L}}_{\mathcal{NN}}^T \boldsymbol{\Phi}_*^T \right]_{11} & \left[\boldsymbol{\Phi}_* \boldsymbol{\mathcal{L}}_{\mathcal{NN}} \boldsymbol{\mathcal{L}}_{\mathcal{NN}}^T \boldsymbol{\Phi}_*^T \right]_{12} & 0 \\ 0 & \left[\boldsymbol{\Phi}_* \boldsymbol{\mathcal{L}}_{\mathcal{NN}} \boldsymbol{\mathcal{L}}_{\mathcal{NN}}^T \boldsymbol{\Phi}_*^T \right]_{21} & \left[\boldsymbol{\Phi}_* \boldsymbol{\mathcal{L}}_{\mathcal{NN}} \boldsymbol{\mathcal{L}}_{\mathcal{NN}}^T \boldsymbol{\Phi}_*^T \right]_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Again, by construction, $\overline{\mathbf{u}}_{\mathsf{h}}^T \mathbf{K}_{\mathsf{ML}} \overline{\mathbf{u}}_{\mathsf{h}} \geq 0$ and, as a result, the coarse-scale system is symmetric positive definite for any $\overline{\mathbf{u}}_{\mathsf{h}}$. The system is invertible with a unique solution and may be solved with the conjugate gradient method.

In summary, this section demonstrated that, in the simplified setting of linear elasticity in one-dimension, the SPSD stiffness-based models result in coarse-scale systems that are SPD. These systems have invertible system matrices with unique solutions. We further demonstrated that we cannot guarantee the systems emerging from the direct displacement-to-force mappings have unique solutions.

6 Numerical examples

We now investigate the performance of the proposed models on several exemplars, starting from a canonical cube undergoing finite deformations to a more complex fastener-bushing geometry undergoing finite deformations with contact.

6.1 Implementation in Sierra Solid Mechanics

The machine-learned models outlined in Section 4 are implemented in the Sierra/Solid-Mechanics code base, which is a part of the Sierra simulation code suite developed at Sandia National Laboratories. The ML implementation is built on top of the pocket-tensor library [35], which provides a C++ interface for TensorFlow and Keras models. For the direct displacement-to-force models, the stiffness matrix associated with the ML element is computed via finite differences, while for the displacement-to-stiffness models the stiffness matrix is directly output from the neural network. For all cases, the nonlinear system of equations resulting from the FEM and FEM–ML models is solved via the nonlinear conjugate gradient method with a full tangent preconditioner. We employ hypoelastic material models for all examples and solve the governing equations with finite deformations under a quasi-static approximation. Contact is handled using the Augmented Lagrange algorithm. We refer the interested reader to the Sierra/-SolidMechanics theory manual for more details [7]. We note that, while it would be interesting to assess the performance of more generic solvers for the non-SPSD models (e.g., generalized minimum residual), these are not available in Sierra/SolidMechanics given the superior performance of conjugate gradient for SM problems.

6.2 Training of ML models

All machine-learned models are trained in Python. For LLS, we directly solve the least-squares problem using scipy.optimize.lsq_linear. Given the analytic solution to the optimization problem we do not hold out any data in training. For SPSD-LLS, we solve the optimization problem with the splitting conic solver as implemented in the software cvxpy [2, 16]. We again do not hold out any data for training SPSD-LLS.

Both the NN and SPSD-NN models are implemented and trained in PyTorch. For training the networks we employ random shuffling and a standard 80/20 training/-validation split of the data. For each model configuration, we train 10 models with PyTorch's default random initialization (Kaiming Uniform for dense linear layers). We select a final model based on the loss on the (holdout) validation set. The neural networks are trained with the ADAM optimizer for 35000 epochs. We employ an early stopping criteria if the 2000-epoch running mean of the validation loss has not decreased. Unless otherwise noted, we employ a batch size of 50 with a learning rate schedule of $\mathsf{lr} = \{1 \times 10^{-3}, 2 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}, 2 \times 10^{-5}\}$, where the learning rate is lowered after $500, 1000, 2000, 5000$, and 15000 epochs. For network architecture, we employ fully connected neural networks with three hidden layers. The number of nodes in each layer is set to be equivalent to the size of the network input vector (e.g., the reduced basis dimension, K). The loss function for all models is the ℓ^2 loss between the predicted and truth reduced forces. We employ ReLU for our nonlinear activation function. To attempt to improve the robustness, we additionally consider ℓ^2 weight regularization (weight_decay in PyTorch) for NN. We train models for penalty parameters of $\lambda = \{0, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\}$, and the "optimal" model is again chosen via loss on the validation set. We do not employ regularization for our SPSD-NN models.

Lastly, we comment on data normalization. We employ a max-abs normalization strategy for both the NN and SPSD-NN models. Here, the reduced coordinates associated with both the displacements and forces are normalized by their maximum absolute value across the training set. To describe this mathematically, let $\hat{\mathbf{U}}_{\mathrm{train}} = \mathbf{\Phi}_{\boldsymbol{u}}^T \mathbf{U}_{\mathrm{train}} \in \mathbb{R}^{K \times n_{\mathrm{train}}}$ denote the reduced coordinates associated with the training set for the displacement field; we have used $\mathbf{U}_{\mathrm{train}}$ to denote the training data matrix obtained after the training/validation split on $\mathbf{U}$ such that $n_{\mathrm{train}} < N_s$. We scale these reduced coordinates by

$$\hat{\mathbf{U}}_{\mathrm{train}}^* = \frac{\hat{\mathbf{U}}_{\mathrm{train}}}{\max\left(\mathrm{abs}\left(\hat{\mathbf{U}}_{\mathrm{train}}\right)\right)}.$$

We note that $\max\left(\mathrm{abs}\left(\hat{\mathbf{U}}_{\mathrm{train}}\right)\right) \in \mathbb{R}^+$ provides a scalar scaling used for all reduced coordinates. We found that this approach out-performed normalization strategies that used an individual scaling for each reduced coordinate. The same scaling is employed for the forces. We note that we do not center our data, which would complicate maintaining SPD structure in, e.g., Eq. (12).

The training stage for our examples involves solving a full FEM that directly resolves the fastener. The coarse and fine scales are then taken to be a subset of this full FEM discretization and the training data are extracted accordingly. In theory,
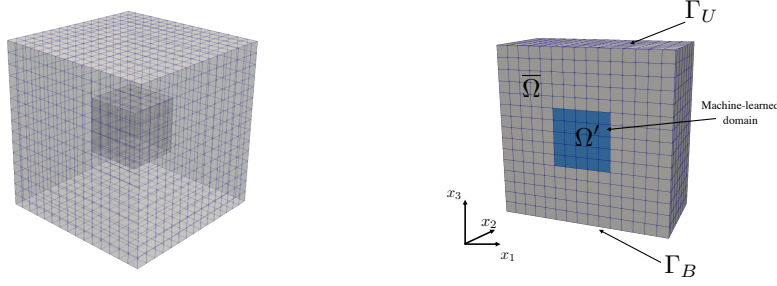
Fig. 3: Example 1, multi-element cube. Depiction of the setup of the cube problem. The ML model replaces the interior cube, which comprises a $5 \times 5 \times 5$ element block of the full FEM model.

one could employ non-conforming decompositions (e.g., perform the training on one discretization and learn an ML model that is applicable for other meshes via, e.g., interpolation). For simplicity, we do not consider this here. In practice, our training process consists of two phases. First, we solve the full FEM that directly resolves the coarse and fine scales. After this, we extract the fine-scale domain from the model and execute a post-processing step to compute the internal forces.

6.3 Multi-element deformed cube

We begin by considering quasi-static deformation of a cube on the domain $\Omega \equiv [-1.5\mathsf{in}, 1.5\mathsf{in}]^3$. The "inner" fine-scale domain is defined as $\Omega' = (-0.5\mathsf{in}, 0.5\mathsf{in})^3$ while the outer domain is defined such that $\Omega = \overline{\Omega} \oplus \Omega'$. For discretization, the domain Ω is discretized into $15 \times 15 \times 15$ uniformly sized cubes in each direction resulting in $\overline{N}^{\Gamma_\mathcal{I}} = 152$ interface nodes. We employ standard bilinear Lagrange finite elements. Figure 3 depicts a setup of the problem. We employ homogeneous Dirichlet boundary conditions on the bottom boundary and Dirichlet boundary conditions on the top boundary, i.e.,

$$
\begin{aligned}
\boldsymbol{u}_\mathrm{h} &= \boldsymbol{0} \text{ on } \Gamma_B, \\
\boldsymbol{u}_\mathrm{h} &= [u_1, u_2, u_3] \text{ on } \Gamma_U,
\end{aligned}
\tag{21}
$$

where $\Gamma_B = \{(x_1, x_2, x_3) \in \Omega | x_3 = -1.5\mathsf{in}\}$ is the lower boundary and $\Gamma_U = \{(x_1, x_2, x_3) \in \Omega | x_3 = 1.5\mathsf{in}\}$ is the upper boundary. We employ a hypoelastic material model across the entire domain. For $\boldsymbol{x} \in \overline{\Omega}$ the material model is characterized by a Young's modulus of 28.5e6 psi with a Poisson's ratio of $\nu = 0.3$. For $\boldsymbol{x} \in \Omega'$ the material model is characterized by a Young's modulus of 29×10^6 psi with a Poisson's ratio of $\nu = 0.3$. As quantities of interest (QoIs), we consider integrated reaction forces on the bottom of the cube, Γ_B, as well as the error of the displacement field $\overline{\mathbf{u}}_\mathrm{h}$ in the outer domain. We denote these errors as e_{x_i} and e_{u_i}, $i = 1, \ldots, 3$, respectively. We note that all reported errors are relative errrors that are measured in the ℓ^2 norm, i.e., for an ML-FEM solution $\mathbf{x}$ and a full FEM solution $\mathbf{y}$, we report $e = \|\mathbf{x} - \mathbf{y}\|_2 / \|\mathbf{y}\|_2$.

17
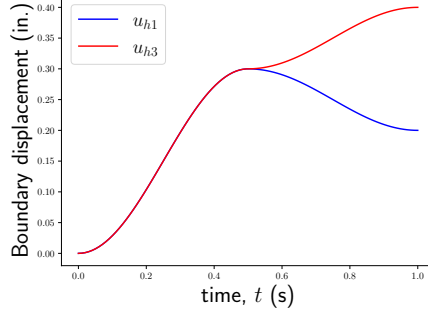
Fig. 4: Example 1, multi-element cube. Depiction of loading profile for $b_{x1} = 0.3$, $b_{x2} = -0.1$, $b_{z1} = 0.3$, $b_{z2} = 0.1$.

Testing run number	b_{x1}	b_{x2}	b_{z1}	b_{z2}
1	0.05	-0.01	-0.02	0.01
2	0.15	-0.10	-0.20	0.15
3	0.40	-0.10	-0.30	0.10
4	-0.30	-0.30	0.50	-0.05

Table 1: Example 1, multi-element cube. Parameter values for testing configurations.

For training we solve the cube problem for 16 different non-proportional quasi-static loading trajectories for $t \in [0\mathsf{s}, 1\mathsf{s}]$, where t is pseudo-time and a single loading trajectory comprises 100 time steps with the top boundary displaced by a fixed profile. This loading profile is prescribed (in inches) for $\boldsymbol{x} \in \Gamma_U$ as

$$
\begin{aligned}
\boldsymbol{u}_{\mathrm{h}1}(t, \boldsymbol{x}) &= \begin{cases} \frac{1}{2}\left(b_{x1} + b_{x1}\cos\left(\left(\frac{t-0}{T_{x1}-0}\right)\pi - \pi\right)\right) & 0 \le t \le T_{x1} \\ b_{x1} + \frac{1}{2}\left(b_{x2} + b_{x2}\cos\left(\left(\frac{t-T_{x1}}{T-T_{x1}}\right)\pi - \pi\right)\right) & T_{x1} < t < T \end{cases} \\
\boldsymbol{u}_{\mathrm{h}3}(t, \boldsymbol{x}) &= \begin{cases} \frac{1}{2}\left(b_{z1} + b_{z1}\cos\left(\left(\frac{t-0}{T_{z1}-0}\right)\pi - \pi\right)\right) & 0 \le t \le T_{z1} \\ b_{z1} + \frac{1}{2}\left(b_{z2} + b_{z2}\cos\left(\left(\frac{t-T_{z1}}{T-T_{z1}}\right)\pi - \pi\right)\right) & T_{z1} < t < T \end{cases}
\end{aligned} \tag{22}
$$

where $T_{x1} = T_{z1} = 0.5\mathsf{s}$. Figure 4 shows an example loading profile for $b_{x1} = 0.3$, $b_{x2} = -0.1$, $b_{z1} = 0.3$, $b_{z2} = 0.1$.

As a training set we examine a hyper-cube of the parameters $b_{x1} = (-0.3, 0.3)$, $b_{x2} = (-0.1, 0.1)$, $b_{z1} = (-0.3, 0.3)$, and $b_{z2} = (-0.1, 0.1)$. We sample the corners of the hyper-cube, which results in 1600 total quasi-static solves (16 parameter configurations with 100 time steps per configuration). For testing we examine four loading configurations as described in Table 1. We note that this setup comprises training runs that deform the body by up to 10% and testing configurations that deform the body up to 16%, both of which are beyond the small-strain region and result in non-linearities.

18
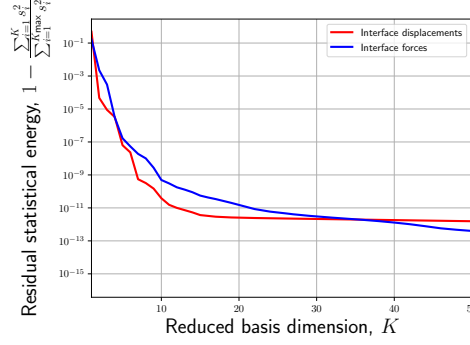
Fig. 5: Example 1, multi-element cube. Residual statistical energy of the reduced basis approximation for the interface displacement and force fields.

6.3.1 Offline training results

We first assess the reduced basis approximations as described in optimization problems (7) and (8). As there is no body forcing term, we take $\mathbf{f}_0 = \mathbf{0}$. Figure 5 depicts the residual statistical energy (i.e., relative energy contained in the truncated singular values) associated with the reduced basis approximation[3]. We observe that both the interface force field and interface displacement field are well-represented with relatively few basis vectors. With just 10 basis vectors the residual statistical energy of both the interface displacement and force fields is less than 10^{-9}. This quick decay demonstrates that the interface can be characterized with relatively few degrees of freedom, and justifies the ansatz of performing the learning process in the reduced space.

Next, Figure 6 shows the training error for the various ML models considered as a function of the reduced basis dimension, K, and the number of model parameters, N_w. We observe that LLS yields significantly more accurate results than SPSD-LLS. SPSD-LLS saturates in accuracy around $K_* = 10$ while LLS continues to converge, reaching a training error of 10^{-3}. We next observe that LLS results in models that are approximately as accurate as NN and SPSD-NN. We believe that, for this example, LLS performs as well as NN because the data are almost linear. Since LLS has an analytic solution, we are more easily able to optimize the model. Lastly, we observe that SPSD-NN performs as well as NN and LLS, and is significantly more accurate than SPSD-LLS.

6.3.2 Coupled FEM-ML results

We now examine *a posteriori* results where the ML model is coupled to the FEM solver. Figure 7 shows QoI errors for the various models as a function of the reduced dimension. LLS results in accurate models for small reduced basis dimensions but quickly goes unstable as the basis dimension grows past $K = 7$. We emphasize that this instability is not observed in the offline training results and is likely a result of the interplay between the LLS-POD model and the solver. Despite being the least accurate

---

[3]Note that the residual statistical energy bounds the error of the reduced basis approximation.
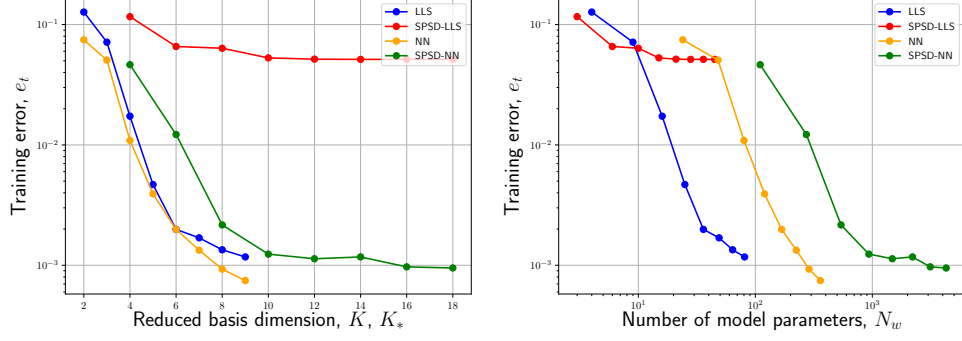
Fig. 6: Example 1, multi-element cube. Offline training results of machine learned models. On the left we show the training error as a function of the reduced basis dimension K while on the right we show results as a function of model parameters, N_w. Note that the SPSD models employ the combined basis of dimension $K^* = 2K$.

model in training, SPSD-LLS results in stable and accurate models with relative errors for both QoIs of less than 1% for all reduced basis dimensions. NN results in accurate models, but becomes unstable for reduced basis dimensions $K > 7$. SPSD-NN is the most accurate and robust model. It is stable for all reduced basis dimensions and results in relative errors of less than 0.01% for almost all reduced basis dimensions. Lastly, we comment on the observation that both SPSD-LLS and SPSD-NN display little convergence with reduced basis dimension. In this example, relatively few basis vectors capture the majority of the statistical energy. Using just a 6 dimensional basis, SPSD-NN results in displacement fields with relative errors of approximately 0.01%, and there is little room for improvement by adding more bases. Improved convergence will be observed in the following examples where more basis vectors are required to represent the interface displacements and forces.

Next, Figure 8 shows QoI results for two of the testing configurations. We plot results for the optimal configurations (i.e., the configuration leading to the lowest error) for each model. The optimal configurations are $K = 7$, $K^* = 14$, $K = 5$, and $K^* = 18$ for LLS, SPSD-LLS, NN, and SPSD-NN, respectively. We observe that all predictions lie on top of the truth data. Overall, we observe that all models are accurate when they converge. Lastly, Figure 9 shows the von Mises stress predicted by the SPSD-NN model (left) and full model (right) for testing configuration #4. With the contours shown in the figure, we observe no noticeable difference between the FEM–ML and FEM-only solutions.

The results of this elementary example demonstrate the capability of the ML models to replace a subdomain, even when the behavior of the material is in a nonlinear large strain regime. We additionally observed that, while LLS and NN models were accurate in the offline phase, they rarely led to converged solutions when coupled with the FEM solver. The SPSD-LLS and SPSD-NN models, both of which enforced an SPSD stiffness matrix, were more robust when coupled to the FEM model.
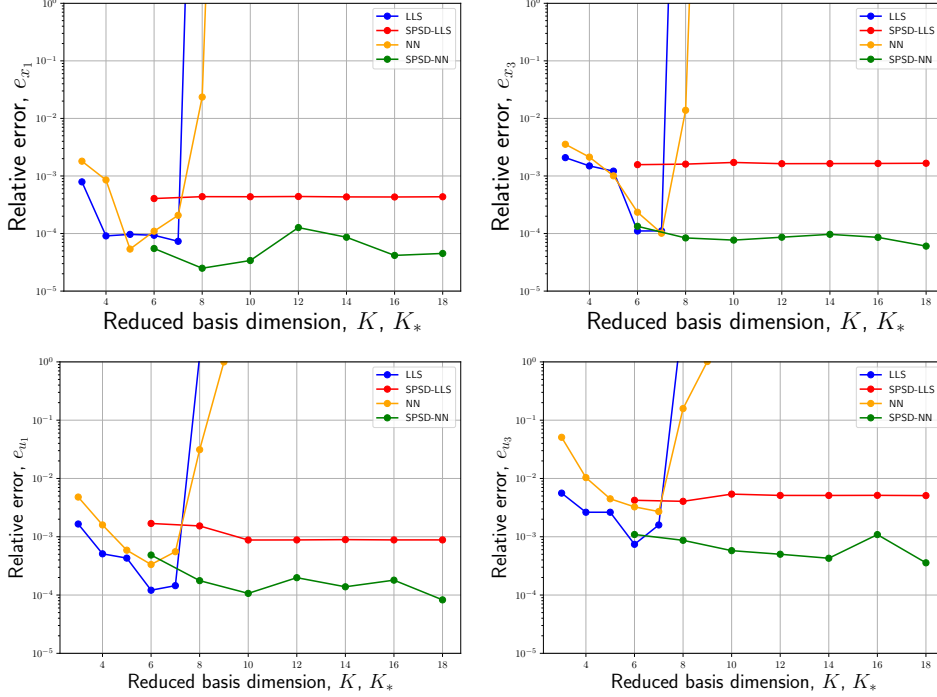
Fig. 7: Example 1, multi-element cube. Relative errors for coupled FEM-ML models as a function of reduced basis dimension. The top row shows relative errors for the integrated force QoIs, while the bottom row shows relative errors for the displacement field QoIs. Note that the SPSD models employ the combined basis of dimension $K^* = 2K$.

6.4 Fastener undergoing loading with contact

We now consider a more complex example comprising a fastener-bushing geometry undergoing quasi-static radial loading. This geometry is derived from a test setup which can be found in [1]. The NAS1351-3-20P fastener is modeled as a "plug" (fastener head and shank with no modeling of threads) with a 0.187 inch diameter and sits in the middle of the top bushing through hole, initially not in contact. The fastener is connected to the bottom bushing with a contiguous mesh. As the joint is loaded laterally, the 0.003 inch gap between the top bushing and fastener will close and the two volumes will be in contact. The problem is symmetric about the x_1-x_3 plane, and symmetry is enforced with a symmetric boundary condition. There is no preload in this case; preload will be considered in the next section. Figure 10 depicts the problem. The full FEM mesh has $N = 115,292$ degrees of freedom. We remove both the fastener and the immediate domain around the fastener, which results in an interface with $\overline{N}^{\Gamma_{\mathcal{I}}} = 3348$ nodes. We emphasize that contact occurs exclusively within the removed domain, and as a result our ML model must be able to accurately
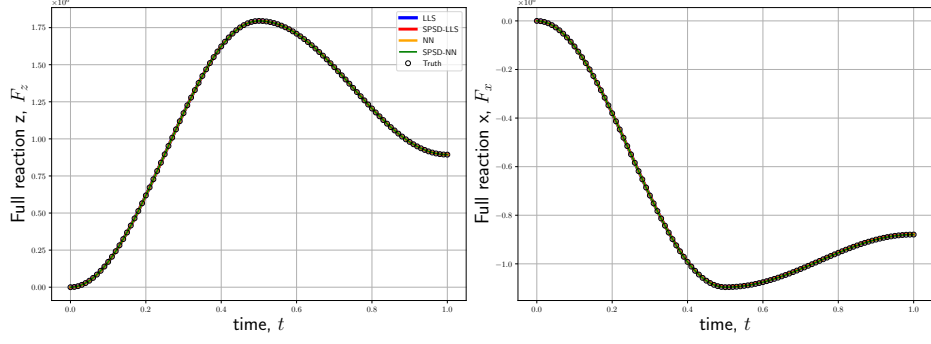
(a) Testing run number 1.
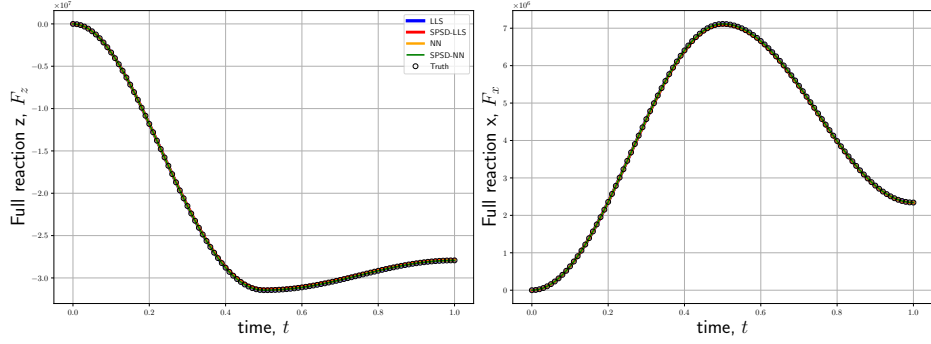
(b) Testing run number 4.

Fig. 8: Example 1, multi-element cube. Results of ML–FEM and FEM-only model on select testing configurations. Results are shown for the optimal configuration for each model, which corresponds to $K = 7$, $K^* = 14$, $K = 5$, and $K^* = 18$ for LLS, SPSD-LLS, NN, and SPSD-NN, respectively.

characterize contact. We employ homogeneous Dirichlet conditions on the exterior of the bottom bushing and Dirichlet boundary conditions on the top boundary. The quasi-static loading profile is given in inches for $\boldsymbol{x}$ on Γ_U, $t \in [0, 1]$ by $\boldsymbol{u}_{h1}(t, \boldsymbol{x}) = \beta t$, $\boldsymbol{u}_{h3}(t, \boldsymbol{x}) = \frac{1}{400}t$; this loading configuration corresponds to radial loading (i.e., a pull at a constant angle) whose angle is a function of the parameter β. We again employ a hypoelastic material model across the entire domain. For $\boldsymbol{x} \in \overline{\Omega}$ the material model is characterized by a Young's modulus of 28.5e6 psi with Poisson's ratio of $\nu = 0.3$. For $\boldsymbol{x} \in \Omega'$ the material model is characterized by a Young's modulus of 29×10^6 psi with a Poisson's ratio of $\nu = 0.3$. As QoIs, we consider intergrated reaction forces on the exterior of the bottom bushing in the x_1 and x_3 directions as well as the error of the displacement field $\overline{\boldsymbol{u}}_h$ in the outer domain.

We consider 21 training configurations for $\beta = \{-0.005 + 0.0005i\}_{i=0}^{20}$ and $t \in \{0.01i\}_{i=1}^{101}$. This dataset comprises quasi-static trajectories with 101 time steps per

22
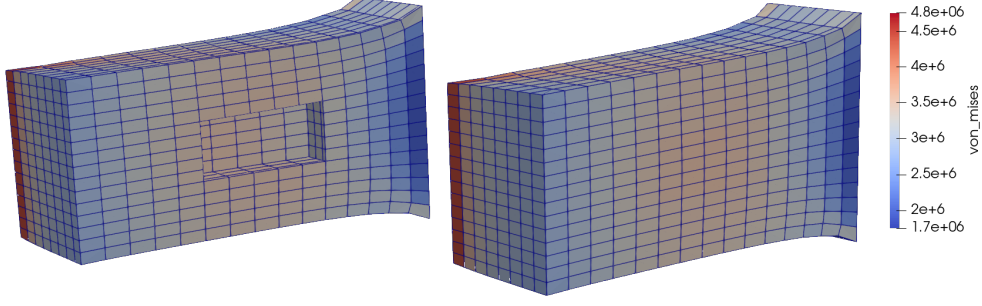
Fig. 9: Example 1, multi-element cube. Results of FEM-ML coupled simulation with the SPSD-NN model (left) and FEM-only simulation (right) for prediction of the Von-Mises stress. Note that displacements are amplified for visualization.

trajectory for pulls ranging from approximately $-65°$ to $65°$, which results in 2121 total quasi-static solves.

6.4.1 Offline training results

Figure 11 demonstrates that the interface degrees of freedom are amenable to dimension reduction by displaying the residual statistical energy associated with the reduced basis approximation. As there is again no body forcing term, we take $\mathbf{f}_0 = \mathbf{0}$. As compared to the previous exemplar, the residual statistical energy in the force approximation decays slower. Overall, however, we again observe that both the interface force field and interface displacement field are well-represented with relatively few basis vectors. With just 10 basis vectors the residual statistical energy of the interface displacement field is less than 10^{-10} and the interface force field is less than 10^{-6}. This quick decay again demonstrates that the interface can be characterized with relatively few dimensions.

Figure 12 shows the training error for the various ML models considered as a function of the reduced basis dimension, K, and the number of model parameters, N_w. We observe that SPSD-LLS results in the highest training errors and that increasing the basis dimension no longer results in lower errors after around $K_* = 8$. In training, the standard LLS model is much more accurate than SPSD-LLS and results in accurate models that achieve a relative error around 10^{-3}. We observe that NN is more accurate than LLS for low reduced basis dimensions, but is surprisingly *less accurate* for larger basis dimensions. Given that the data are inherently nonlinear due to contact, we theorize that this result is due to LLS over-fitting to the data when there are many model parameters; note that LLS is analytically solved and is not subject to any optimization process like NN. For small reduced basis dimensions, where the models have fewer parameters, NN presumably outperforms LLS due to its ability to model nonlinear responses. We observe that SPSD-NN is significantly more accurate than SPSD-LLS and is competitive with both LLS and NN. This result demonstrates that we are able to generate a structure-preserving SPSD stiffness matrix model with a

23
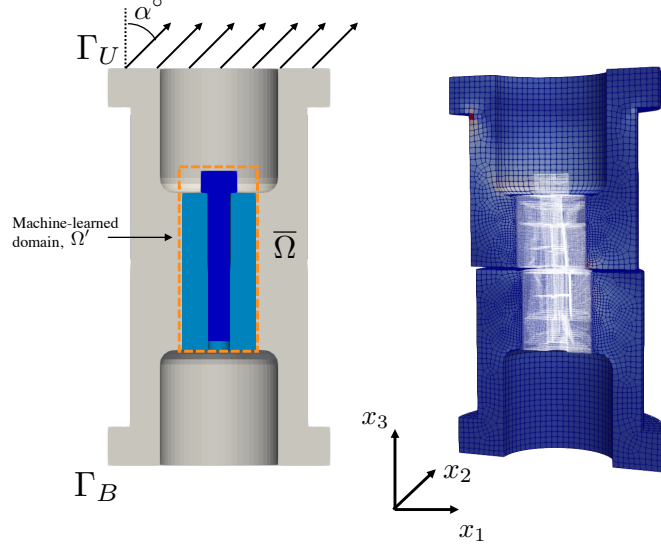
Fig. 10: Example 2, fastener contact. Problem schematic for fastener example. On the left we show the decomposition between the resolved FEM domain and the ML domain along with the loading configuration, while on the right we show a sample training solution at $\alpha \approx 65°$. We show contours of max Von Mises stress with the fastener subdomain colored in white. Displacements are amplified for visualization.

comparable accuracy to LLS and NN. It is interesting to observe that the improvement between SPSD-NN and SPSD-LLS is far greater than between NN and LLS.

Figure 13 reports the CPU time required for performing the FEM training simulations and training (one realization of) the different ML models. We observe that the training time of all the ML models is small compared to the time required to run the high-fidelity training simulations; the cost of running the high-fidelity models is over 500x more than the most expensive ML model (SPSD-LLS) and is approximately 5000x more than the neural-network-based ML models. Comparing the different ML models, we find that the training cost of LLS and SPSD-LLS tends to increase with reduced basis dimension, while the cost of NN and SPSD-NN is relatively constant.
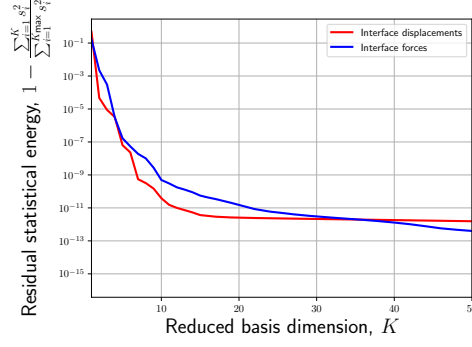
Fig. 11: Example 2, fastener contact. Residual statistical energy of the reduced basis approximation for the interface displacement and force fields.
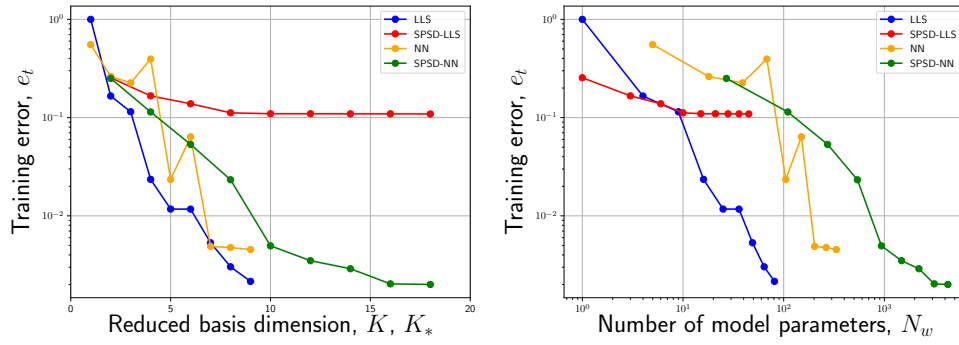
Fig. 12: Example 2, fastener contact. Offline training results of machine learned models trained on the fastener dataset. On the left we show the training error as a function of the reduced basis dimension K while on the right we show results as a function of model parameters, N_w. Note that the SPSD models employ the combined basis of dimension $K^* = 2K$.

6.5 Coupled FEM-ML results

We now examine results for when the ML model is coupled to the FEM solver. We consider 9 runs for

$$\beta = \{-0.005, -0.00475, -0.0045, -0.00425, -0.0025, -0.00125, -0.001, -0.00025, 0.000\}.$$

This set of runs contains configurations that were in the training set as well as novel testing configurations. Figure 14 shows the convergence of the relative error summed over all cases as a function of reduced basis dimension. The top row shows errors in the predicted x_1-reaction and x_3-reaction forces while the bottom row shows errors in the predicted x_1 and x_3 displacement fields. We observe that SPSD-NN is by far the best

25
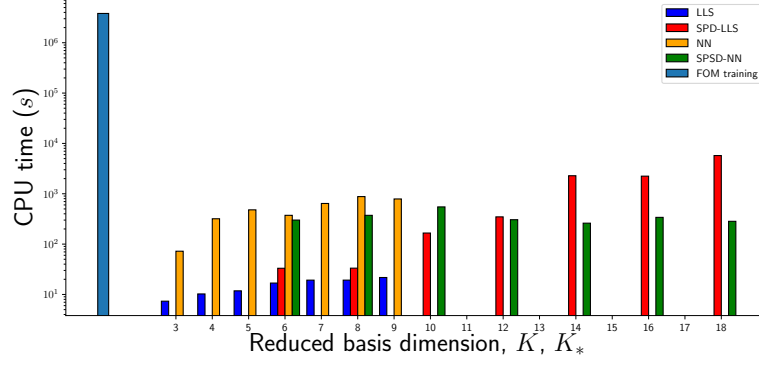
Fig. 13: Example 2, fastener contact. CPU times required for conducting the high-fidelity training simulations and ML model training. Note that the SPSD models employ the combined basis of dimension $K^* = 2K$.

performing method. We observe that it remains stable and becomes more accurate as the model complexity grows; the most complex model tested led to relative errors of around 0.2%. We next observe that LLS is inaccurate and goes unstable as the model complexity increases. The performance of NN is mixed. While the $K = 9$ configuration led to good results, most other configurations performed poorly. In particular, we highlight the discrepancy between, e.g., the performance of the $K = 8$ model when coupled to the model (Figure 14) as opposed to in the offline training phase (Figure 12). We additionally note that convergence of NN with reduced basis dimension is poor. Lastly, we observe that SPSD-LLS remains stable as the reduced basis dimension grows, but similarly to Figure 12, the model is unable to improve as we allow it to have more parameters. This lack of improvement can be attributed to the fact that the data are nonlinear and the SPSD-LLS model cannot be overfit to the data in the same way that LLS can due to the structure we impose on the model. The SPSD-NN model does yield improved results as the reduced basis dimension grows.

Figure 15 shows the relative error of the best performing model of each method as a function of loading angle for the various cases; the best performing models correspond to $K = 3$, $K^* = 6$, $K = 9$, and $K^* = 18$ for the LLS, SPSD-LLS, NN, and SPSD-NN models, respectively. We note that contact occurs in the FEM for all cases when, approximately, $|\alpha| > 40$. As expected SPSD-NN is by far the best performing method and leads to relative errors of under 0.2% in all cases. NN is the next best performing method, while SPSD-LLS and LLS both associate with relatively high errors. This is particularly true when there is contact in which case all methods associate with $> 10\%$ error when it comes to predicting the x_1-reaction force.

Figure 16 shows the reaction forces predicted by the coupled FEM-ML models for several testing configurations. We observe that SPSD-NN is the best performing method and leads to predictions that lie on top of the truth values. The optimal configuration of NN additionally performs well and results in predictions that almost lie

26
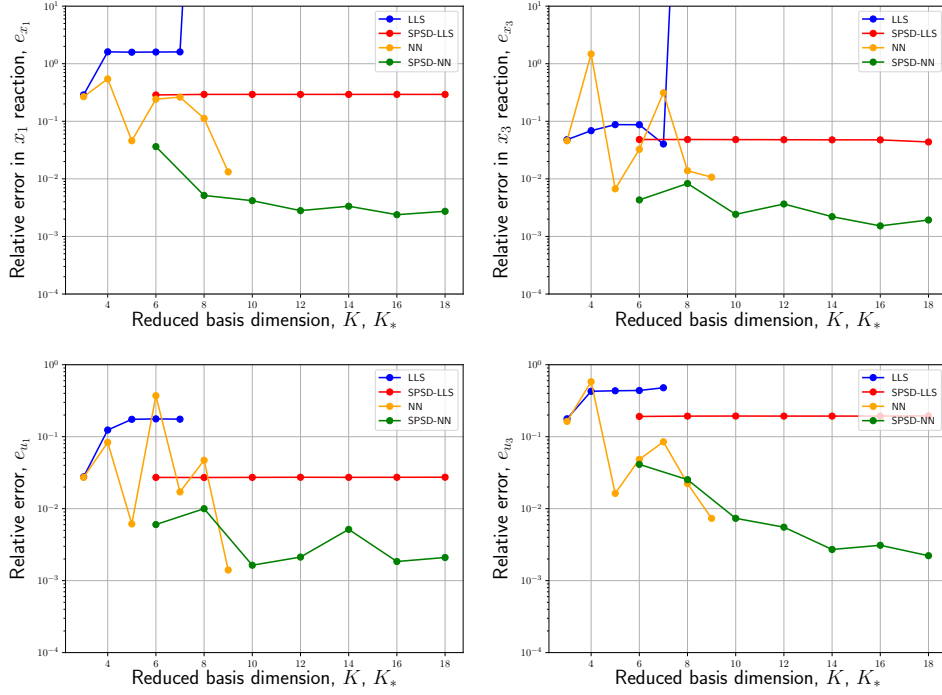
Fig. 14: Example 2, fastener contact. Coupled FEM-ML results for various ML models as a function of reduced basis dimension. The top row shows relative errors for the integrated force QoIs, while the bottom row shows relative errors for the displacement field QoIs. Note that the SPSD models employ the combined basis of dimension $K^* = 2K$.
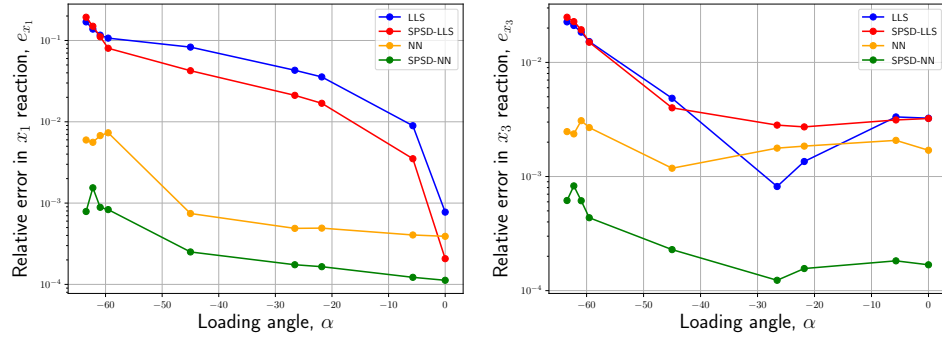
Fig. 15: Example 2, fastener contact. Coupled FEM-ML results for various ML models as a function of loading angle. Results are shown for the optimal configuration for each model, which corresponds to $K = 3$, $K^* = 6$, $K = 3$, and $K^* = 18$ for the LLS, SPSD-LLS, NN, and SPSD-NN models, respectively.

on top of the truth data. Both models are able to correctly capture contact and correctly predict the change in the stiffness. For larger reaction forces, which correspond to larger (absolute) loading angles, both models are able to capture the ramp-like behavior of the reaction force. For lower absolute loading angles, in which case contact does not occur and the response is mostly linear, SPSD-NN and NN are able to still predict the correct stiffness without being "contaminated" by its ability to capture the increased stiffness once contact occurs. Both LLS and SPSD-LLS, which are linear models, are unable to model the contact non-linearity and simply bisect its behavior. This leads to an under-prediction of the stiffness in the post-contact regime and an over-prediction of the stiffness in the pre-contact regime.

Figure 17 shows contour plots of the max von Mises stress throughout the outer bushing as predicted by the coupled FEM-ML model with the best performing SPSD-NN model (left) and FEM-only model (right). For this range of von Mises stress contours, the coupled FEM-ML model leads to results that are visually identical to the FEM-only model. Figure 18 reports the relative CPU times of the FEM–ML reduced-order models as compared to the FEM–only full-order model. We note that we only show CPU times for the SPSD-LLS and SPSD-NN models given that LLS and NN did not converge for many configurations. We observe that the SPSD-LLS model results in speedups between 3-20x, depending on the model configuration, while SPSD-NN results in speedups between 5-25x, again depending on the model configuration. Interestingly, we observe that the SPSD-NN model is, on average, faster than the SPSD-LLS model. An investigation into this reveals that, in the conjugate gradient solver, converging the FEM–ML model with the SPSD-NN model requires, on average, significantly fewer iterations than the SPSD-LLS model. This result suggests that the SPSD-NN model is producing better conditioned stiffness matrices, and highlights adding penalties on the conditioning of the stiffness matrix as an interesting path for future work. Lastly, we highlight that we observe a slight increase in CPU time for SPSD-LLS as the reduced basis dimension is increased, suggesting that conditioning of the resulting stiffness matrix degrades as the basis dimension is increased. No such pattern is apparent for the SPSD-NN model.

Lastly, we comment on the combined offline and online cost of the FEM-ML models. This example employed 11 samples for training the coupled FEM-ML model. As shown in Figure 13, running the high-fidelity model for these 11 samples is the dominant cost of the offline training phase. The gain realized from using the coupled FEM-ML models depends on the number of online samples that are required. Figure 19 reports the theoretical cost ratio of employing an FEM-ML model for an analysis opposed to the FEM-only model as a function of the number of online samples. We report results for FEM-ML models associating with $2.5\%, 5\%, 10\%$, and 20% relative run times, which covers the cost ratios associated with the SPSD-NN models reported in Figure 18. The theoretical cost ratio is computed by
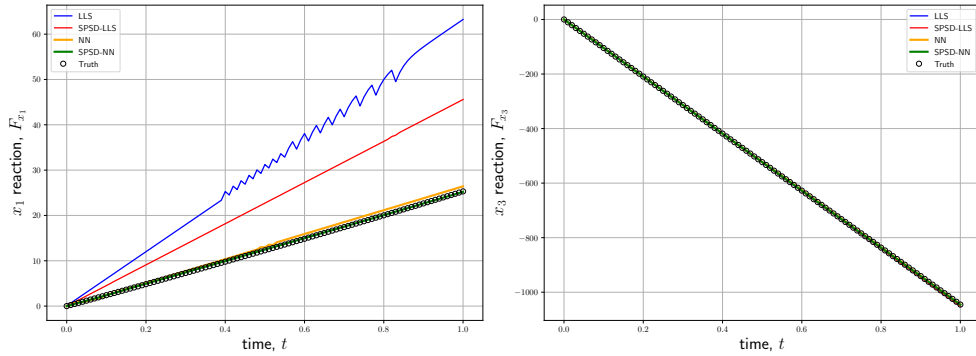
$$\text{CostRatio} = \frac{N_s + N_s^{\text{online}} \frac{\text{FEM-ML-cpu-time}}{\text{FEM-only-cpu-time}}}{N_s^{\text{online}}}. \tag{23}$$

28
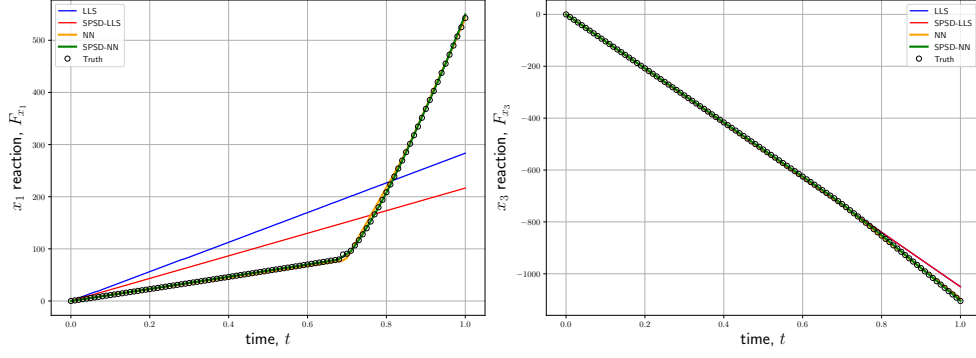
(a) Sample 1 ($\beta = -0.00475$)
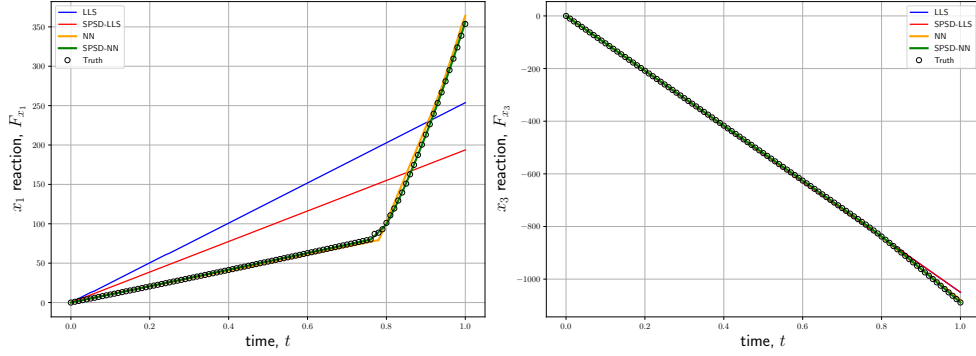
(b) Sample 4 ($\beta = -0.00425$)

(c) Sample 7 ($\beta = -0.00125$)

Fig. 16: Example 2, fastener contact. Coupled FEM-ML results for select testing configurations. We show the measured reaction force in the x_1 (shear) direction on the left and the measured reaction force in the x_3 (tensile) direction on the right. We note that contact is more predominant in the shear direction. Values with small absolute loading angles correspond to the cases with low reaction forces in the shear direction, while values with large absolute loading angles correspond to cases with large loading angles in the shear direction.

29
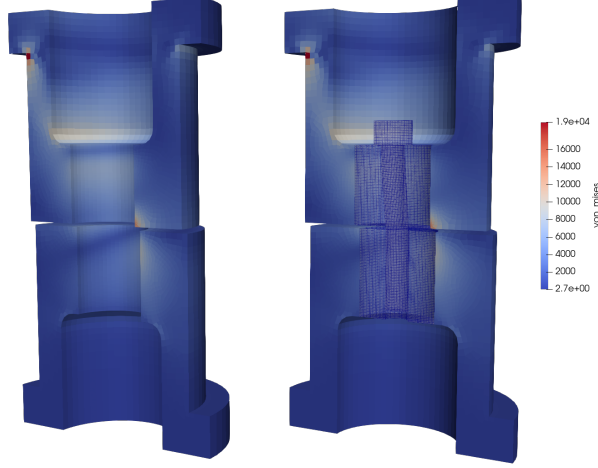
Fig. 17: Example 2, fastener contact. Predictions for the max von Mises stresses by the FEM-ML (left) and FEM-only (right) models. Note that the displacements are enlarged by a factor of 10 for visualization.
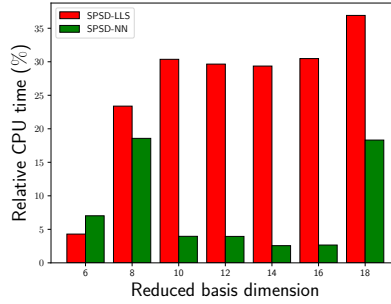
Fig. 18: Example 2, fastener contact. Relative CPU times for the FEM–ML coupled reduced-order models (i.e., CPU time of the FEM-ML model divided by CPU time of the FEM-only full-order model which resolves the fastener). The reported CPU times are averaged across the nine testing configurations.

We observe that an overall speedup is obtained after 11-15 online samples, and by 100 online samples all models are giving over a 3x gain in efficiency.

6.6 Fastener undergoing three-dimensional radial loading with contact and preload

The final problem we consider is again a fastener-bushing geometry, but this time we consider a three-dimensional loading profile with an initial preloaded state. The problem configuration is shown in Figure 20. The full FEM mesh comprises $N =$
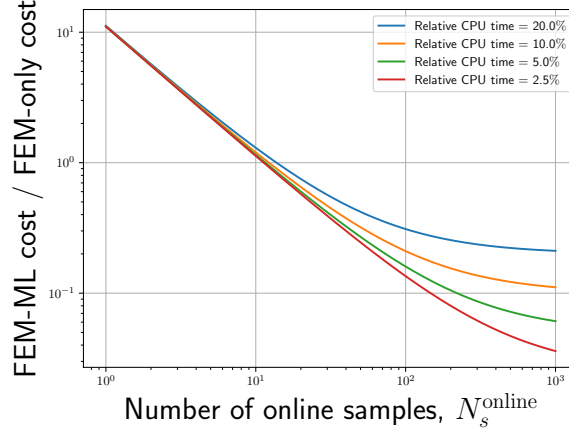
Fig. 19: Example 2, fastener contact. Cost ratio associated with using the FEM-ML model for an analysis opposed to the FEM-only model as a function of the number of samples. The cost of the FEM-ML model includes the offline training time required for generating the 11 training simulations.

$234{,}750$ degrees of freedom. We remove the immediate domain around the fastener, which results in an interface with $\overline{N}^{\Gamma_{\mathcal{I}}} = 6840$ nodes. We again emphasize that contact occurs exclusively within the removed domain, and as a result our ML model must be able to accurately characterize contact. We again employ homogeneous Dirichlet conditions on the bottom boundary. The quasi-static loading profile for $\boldsymbol{x}$ on Γ_U is given, in inches, by

$$
\begin{aligned}
\boldsymbol{u}_{\mathrm{h}1}(t, \boldsymbol{x}) &= \beta t \\
\boldsymbol{u}_{\mathrm{h}2}(t, \boldsymbol{x}) &= \alpha t \\
\boldsymbol{u}_{\mathrm{h}3}(t, \boldsymbol{x}) &= \frac{1}{400} t.
\end{aligned}
\tag{24}
$$

We consider 121 training configurations for $\beta = \{-0.005 + 0.001i\}_{i=0}^{10}$, $\alpha = \{-0.005 + 0.001i\}_{i=0}^{10}$, and $t \in \{0.01i\}_{i=1}^{101}$ s. This dataset comprises quasi-static trajectories with 101 quasi-static time steps per trajectory for pulls ranging from approximately $-65°$ to $65°$ in both directions. In addition, for this example we consider a preloaded state such that, at time $t = 0$, the initial force on the interface is $\mathbf{f}_{\mathsf{interface}} = \mathbf{f}_{\mathsf{preload}} \neq 0$. This preload is applied through the use of an artificial strain in the axial direction of the fastener.

We employ the same material configuration as the previous two exemplars. For $\boldsymbol{x} \in \overline{\Omega}$ the material model is characterized by a Young's modulus of 28.5e6 psi with Poisson's ratio of $\nu = 0.3$. For $\boldsymbol{x} \in \Omega'$ the material model is characterized by a Young's modulus of 29×10^6 psi with a Poisson's ratio of $\nu = 0.3$. As a QoI, we again consider the integrated reaction forces on the exterior of the bottom bushing and displacement fields in the outer domain.
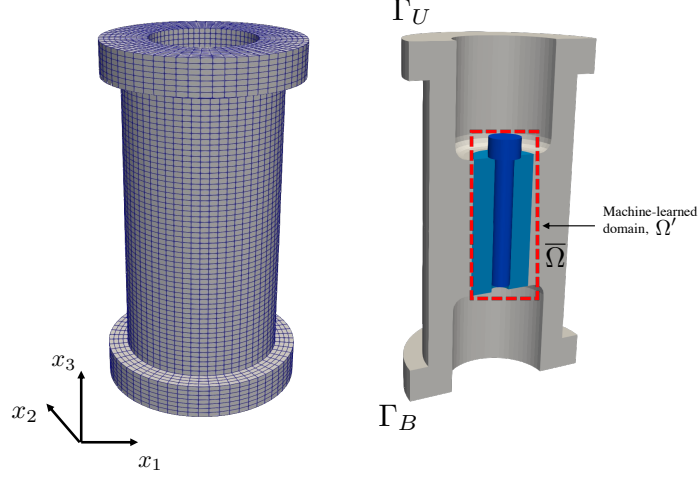
Fig. 20: Example 3, fastener with preload and contact. Schematic of the problem configuration. On the left we show the outer computational mesh and on the right we show the geometry clipped down the middle. The middle region highlighted by the red box comprises the fastener-bushing domain that will be replaced with the ML model.

For this complex exemplar we only investigate the performance of the SPSD-NN model given its superior performance over other formulations. We additionally note that we employ a slightly different training configuration for this example: the batch size is 60 and our initial learning rate is $lr = 2.5 \times 10^{-4}$. The rest is the same as is described in Section 6.2.

6.7 Offline training results

We again first assess the reduced basis approximations. To capture preload, the force offset vector is set to be the force state resulting from preload, i.e., $\mathbf{f}_0 = \mathbf{f}_{\mathsf{preload}}$[4]. Figure 21 depicts the residual statistical energy associated with the reduced basis approximation. As compared to the previous examples, we observe a much slower decay in the residual statistical energy of the interface force field: 40 basis vectors are required to reach a residual statistical energy of 10^{-6}, while in the previous two exemplars this tolerance was achieved with only 10 basis vectors. This result demonstrates the more sophisticated physics present in this example. Despite the slower decay in singular values associated with the interface forces, we are still able to identify a subspace of a much reduced dimension capable of representing the majority of the statistical energy of the force and displacement interface fields.

Figure 22 reports the CPU time required for performing the training simulations and training the ML models. We again observe that the training time of the ML models is small compared to the time required to run the high-fidelity training simulations;

---

[4]The preload force vector is non-uniform and is extracted from the FEM simulation in the training phase. In future work, we plan to parameterize the preload force to handle different types of preload.
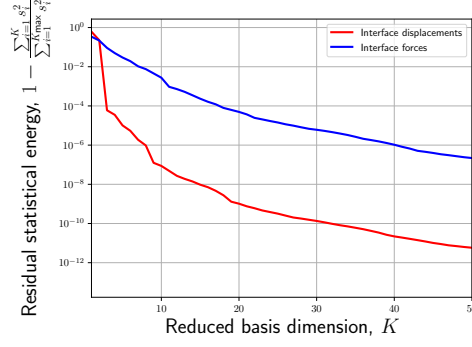
Fig. 21: Example 3, fastener with preload and contact. Residual statistical energy of the reduced basis approximation for the interface displacement and force fields.
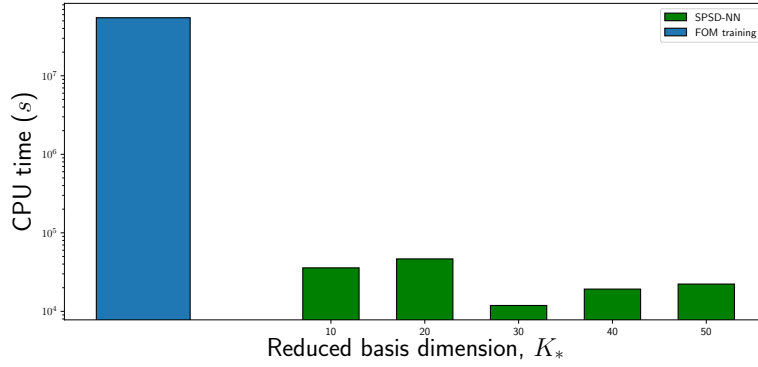

Fig. 22: Example 2, fastener contact. CPU times required for conducting the high-fidelity training simulations and ML model training.

running the high-fidelity models costs 5000-10000x more than training the ML models. Comparing the different ML models, we find that the training cost is relatively insensitive to the reduced basis dimension.

6.8 Coupled FEM-ML results

We move directly to the coupled FEM–ML results. We consider a set of out-of-sample testing runs for $\beta = \{0.0005 + 0.001i\}_{i=0}^{4}$, $\alpha = \{0.0005 + 0.001i\}_{i=0}^{4}$. Figure 23 shows QoI predictions for $K^* = 50$ for select testing samples. Focusing on the reaction force in the x_3 direction, we observe that the ML model is able to replicate the initial preloaded state by virtue of the constant offset vector. We further observe that the ML model is able to simulate the change in stiffness that occurs due to the initial preload around $t \approx 0.05$. Next, examining the reaction forces in the x_1 and x_2 directions, we observe the ML model is able to accurately characterize the change in stiffness due

to contact. The model correctly predicts both when contact occurs, as well as the magnitude of the resulting reaction force. We emphasize that all results are for out-of-sample configurations. Next, Figure 24 shows the convergence of the QoI errors as a function of the reduced basis dimension. We observe almost monotonic convergence for all QoIs. Figure 25 shows contour plots for the max von Mises stress as predicted by the FEM–ML, with the SPSD-NN model at a reduced basis dimension of $K_* = 40$, and FEM-only models. No differences in results are distinguishable.

Figure 26 presents the relative wall-times of the various FEM–ML coupled models as compared to the FEM-only model. We observe that, for all basis dimensions, the FEM–ML coupled systems result in over 100x speedups. We again do not observe a noticeable change in cost with increasing the reduced basis dimension. Next, we assess the combined offline and online cost of the FEM-ML models. This example employed 121 samples for training the coupled FEM-ML model. Running the high-fidelity model for these 121 samples is the dominant cost of the offline training phase (Figure 22). Figure 27 reports the theoretical cost ratio of employing the FEM-ML model for an analysis opposed to the FEM-only model as a function of the number of online samples. We report results for FEM-ML models associating with $0.5\%, 0.75\%$, and 1% relative run times, which covers the cost ratios associated with the SPSD-NN models reported in Figure 26. The theoretical cost ratio is computed as in Eq. (23). As expected we observe that an overall speedup is obtained after 122 online samples. By 1000 online samples, all models are giving over a 5x gain in efficiency.

The present example employed 121 configurations in the training set; as such the model will not be useful if less than 121 online samples are required. The training cost can be reduced by using a smaller training set size. To this end, we consider ML models with a reduced basis dimension of $K_* = 20$ trained using three different training sets,

1. $\beta = \{-0.005 + 0.0005i\}_{i=0}^{10}$, $\alpha = \{-0.005 + 0.005i\}_{i=0}^{10}$,
2. $\beta = \{-0.005 + 0.0005i\}_{i=0,2,4,6,8,10}$, $\alpha = \{-0.005 + 0.005i\}_{i=0,2,4,6,8,10}$,
3. $\beta = \{-0.005 + 0.0005i\}_{i=0,3,6,9}$, $\alpha = \{-0.005 + 0.005i\}_{i=0,3,6,9}$.

All three data sets employ $t \in \{0.01i\}_{i=1}^{101}$ s. For each case we test on the same out-of-sample testing runs as before. Figure 28 shows the convergence of the reaction QoI errors as a function of the training dataset size, denoted by $|\mathcal{D}_{\text{train}}|$. We observe monotonic convergence in all QoIs as the number of training data increase. For the coarsest training dataset, we observe relatively large QoI errors around 3%, while for the finest dataset we observe sub 1% QoI errors. As expected, these results indicate that decreasing the training set size reduces the total offline cost, but also decreases model accuracy. The trade-off between offline cost and model accuracy will depend on the analysis in which the FEM-ML model is being used.

7 Conclusion

This work introduced a machine-learning strategy for finite element analysis of solid mechanics wherein we replaced hard-to-resolve portions of a computational domain with a data-driven surrogate. We proposed two types of data-driven surrogates: one that maps directly from the interface displacements to the interface forces, and one
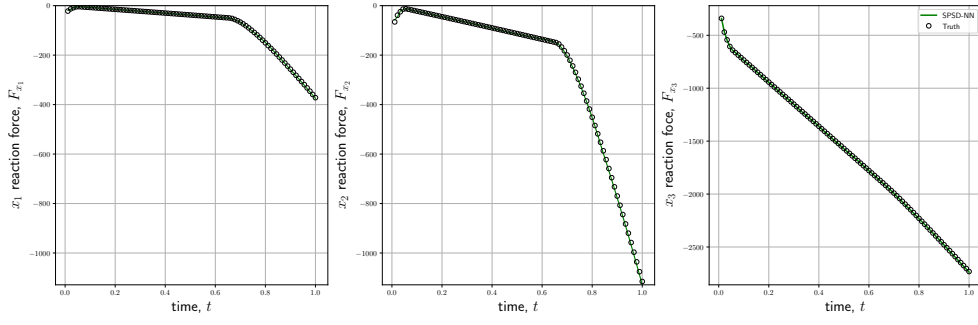
(a) Sample 8 ($\beta = 0.0015$, $\alpha = 0.0025$)
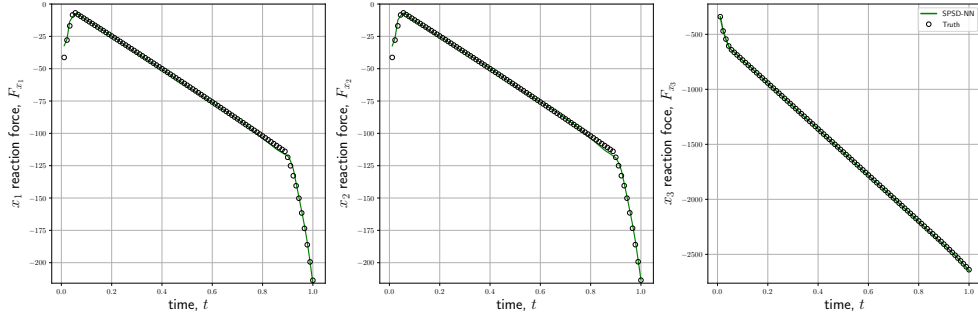
(b) Sample 13 ($\beta = 0.0025$, $\alpha = 0.0025$)
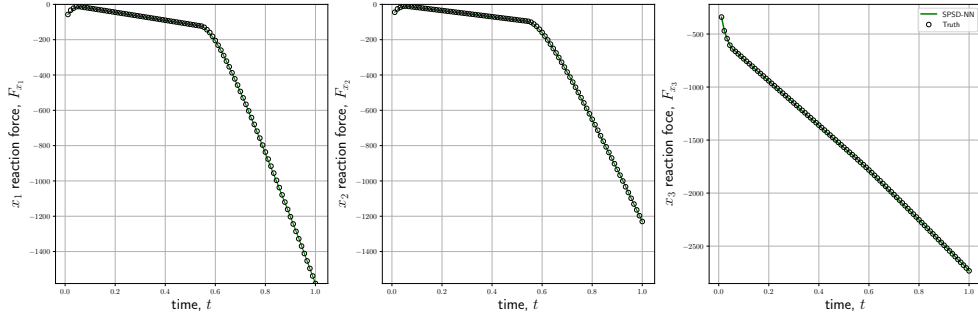
(c) Sample 24 ($\beta = 0.0045$, $\alpha = 0.0035$)

Fig. 23: Example 3, fastener with preload and contact. Coupled FEM-ML results for select testing configurations. From left to right, we show the predicted reaction force in the x_1 (shear), x_2 (shear), and x_3 (tension) directions, respectively. Each line corresponds to a different quasi-static trajectory obtained from a different value of β and α.
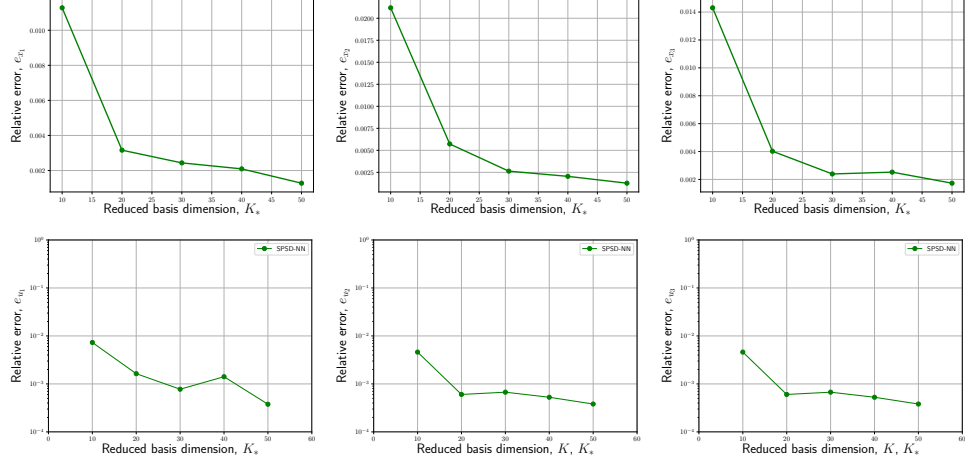
Fig. 24: Example 3, fastener with preload and contact. Relative errors of coupled FEM-ML results with the SPSD-NN ML model as a function of reduced basis dimension. The top row shows relative errors for the integrated force QoIs, while the bottom row shows relative errors for the displacement fields. From left to right, we show the predictions in the x_1 (shear), x_2 (shear), and x_3 (tension) directions, respectively.
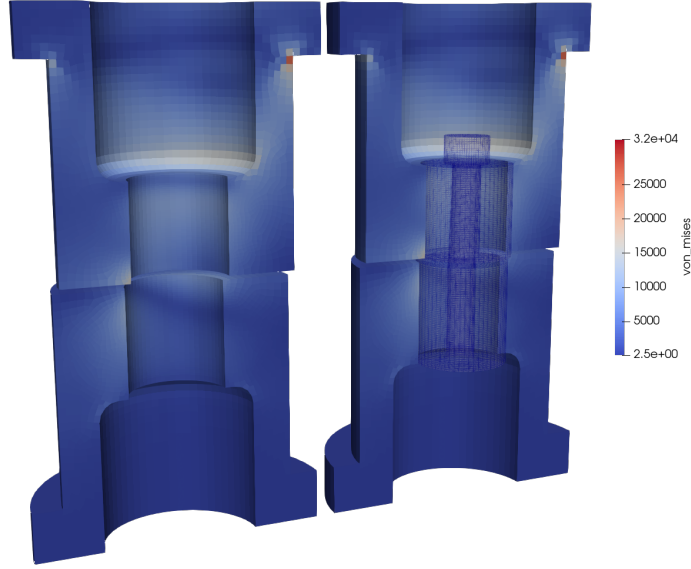
Fig. 25: Example 3, fastener with preload and contact. Predictions for the max Von Mises stresses by the FEM-ML (left) and FEM-only (right) models for $\beta = 0.0045$, $\alpha = 0.0035$. Note that the displacements are enlarged by a factor of 10 for visualization. The FEM–ML solution was obtained approximately 130x faster than the FEM-only solution.
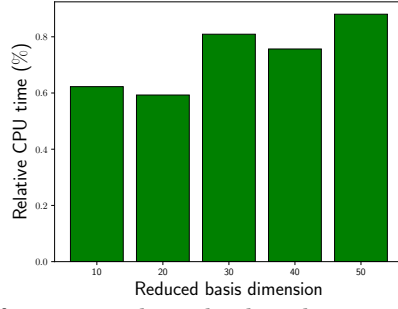
Fig. 26: Example 3, fastener with preload and contact. Relative CPU times for the FEM–ML coupled reduced-order models (i.e., CPU time of the FEM-ML model divided by CPU time of the FEM-only full-order model which resolves the fastener). The reported CPU times are averaged across all testing configurations.
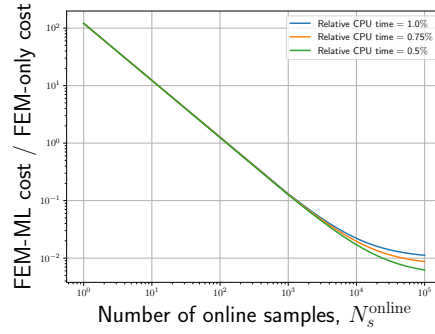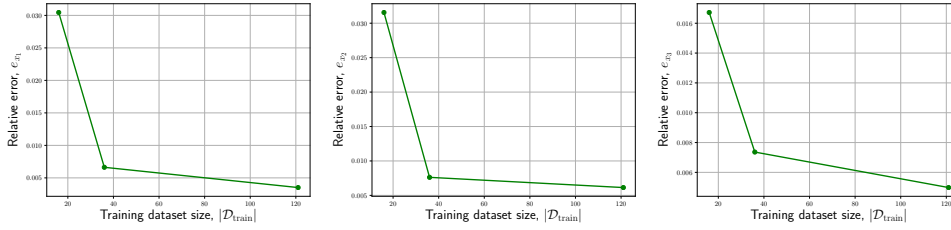

Fig. 27: Example 3, fastener with preload and contact. Cost ratio associated with using the FEM-ML model for an analysis opposed to the FEM-only model as a function of the number of samples. The cost of the FEM-ML model includes the offline training time required for generating the 121 training simulations.

Fig. 28: Example 3, fastener with preload and contact. Relative errors of coupled FEM-ML results with the SPSD-NN ML model as a function of training data set size. From left to right, we show the predicted reaction force in the x_1 (shear), x_2 (shear), and x_3 (tension) directions, respectively.

37

that maps from the interface displacements to the interface forces by virtue of a stiffness matrix that is enforced to be SPSD. We demonstrated, in a simplified setting, that this latter formulation results in a global coarse-scale problem that is symmetric positive-definite (SPD) which guarantees a unique solution and makes the form amenable to conjugate gradient solvers.

We presented numerical experiments across three exemplars spanning a range of physics. These exemplars demonstrated that, in general, direct force-to-displacement models are not robust when combined with a conjugate gradient solver. At this time it is not clear if this lack of robustness is due to interplay between the model form and the conjugate gradient solver, or if rather it is due to the resulting problem being ill-posed. Even if model robustness can be attributed to the conjugate gradient solver, in which case improved performance could be obtained, e.g., by switching to a general minimized residual-based solver, this is still a significant issue given the wide use and effectiveness of conjugate gradient-based solvers in SM.

Our numerical experiments demonstrated that the displacement-to-force via an SPSD stiffness matrix approach resulted in robust and efficient models. In all of our testing cases, the FEM-ML model with the SPSD-NN approach was able to predict QoIs with a relative error of less than 0.5%. On the final exemplar, which was the most complex test case, this formulation led to sub 0.25% errors with speed-ups of over 100x as compared to the standard FEM simulation.

The results of this work demonstrate the promise of using machine-learning for alleviating the computational burden associated with hard-to-resolve portions of a computational domain, like threaded fasteners, for traditional finite element methods. Follow on work will focus on aspects including (1) extension to more sophisticated material models, e.g., elastic-plastic, where history behavior may become important, (2) extension to component-level exemplars with fasteners, where the same fastener model is repeated throughout the domain, (3) adding additional physical constraints to our models, e.g., rigid body rotation, and (4) prediction of QoIs within the substituted subdomain, such as fastener stress and fastener failure.

8 Acknowledgements

References

[1] *Correlation of Reduced-Order Models of a Threaded Fastener in Multi-Axial Loading*, vol. Volume 9: 17th International Conference on Multibody Systems,

Nonlinear Dynamics, and Control (MSNDC) of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 08 2021. V009T09A004.

[2] A. AGRAWAL, R. VERSCHUEREN, S. DIAMOND, AND S. BOYD, *A rewriting system for convex optimization problems*, Journal of Control and Decision, 5 (2018), pp. 42–60.

[3] S. E. AHMED, O. SAN, K. KARA, R. YOUNIS, AND A. RASHEED, *Interface learning of multiphysics and multiscale systems*, Phys. Rev. E, 102 (2020), p. 053304.

[4] F. AS'AD, P. AVERY, AND C. FARHAT, *A mechanics-informed artificial neural network approach in data-driven constitutive modeling*, International Journal for Numerical Methods in Engineering, 123 (2022), pp. 2738–2759.

[5] S. BALKRISHNA TANDALE, B. MARKERT, AND M. STOFFEL, *Intelligent stiffness computation for plate and beam structures by neural network enhanced finite element analysis*, International Journal for Numerical Methods in Engineering, 123 (2022), pp. 4001–4031.

[6] J. BARNETT, I. TEZAUR, AND A. MOTA, *The Schwarz alternating method for the seamless coupling of nonlinear reduced order models and full order models*, 2022.

[7] F. BECKWITH, G. BERGEL, G. DE FRIAS, M. MEREWETHER, S. MILLER, M. MOSBY, K. PARMAR, J. PLEWS, T. SHELTON, J. THOMAS, B. TREWEEK, M. VEILLEUX, E. WAGMAN, K. MANKTELOW, AND J. TRAGESER, *Sierra/SolidMechanics 5.10 Theory Manual*, tech. rep., Sandia National Laboratories, 2022.

[8] G. BERGEL, D. M. DE OCA ZAPIAIN, AND V. ROMERO, *Neural Network Ensembles and Uncertainty Estimation for Predictions of Inelastic Mechanical Deformation Using FEM-NN Approach*, tech. rep., Sandia National Laboratories, 2023.

[9] G. BERKOOZ, P. HOLMES, AND J. L. LUMLEY, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annu. Rev. Fluid Mech., 25 (1993), pp. 539–575.

[10] C. PRUD'HOMME AND D.V. ROVAS AND K. VEROY AND L. MACHIELS AND Y. MADAY AND A. PATERA AND G. TURINICI, *Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations*, Proceedings SMA Symposium, (2002).

[11] G. CAPUANO AND J. J. RIMOLI, *Smart finite elements: A novel machine learning application*, Computer Methods in Applied Mechanics and Engineering, 345 (2019), pp. 363–381.

[12] J. H. CHAUDHRY, L. N. OLSON, AND P. SENTZ, *A least-squares finite element reduced basis method*, 2020.

[13] R. R. CRAIG AND M. C. C. BAMPTON, *Coupling of substructures for dynamic analyses.*, AIAA Journal, 6 (1968), pp. 1313–1319.

[14] A. DE CASTRO, P. BOCHEV, P. KUBERRY, AND I. TEZAUR, *Explicit synchronous partitioned scheme for coupled reduced order models based on composite reduced bases*, 2023.

[15] A. DE CASTRO, P. KUBERRY, I. TEZAUR, AND P. BOCHEV, *A Novel Partitioned Approach for Reduced Order Model – Finite Element Model (ROM-FEM) and ROM-ROM Coupling*, 2022.

[16] S. DIAMOND AND S. BOYD, *CVXPY: A Python-embedded modeling language for convex optimization*, Journal of Machine Learning Research, 17 (2016), pp. 1–5.

[17] T.-P. FRIES AND T. BELYTSCHKO, *The extended/generalized finite element method: An overview of the method and its applications*, International Journal for Numerical Methods in Engineering, 84 (2010), pp. 253–304.

[18] R. J. GUYAN, *Reduction of stiffness and mass matrices*, AIAA Journal, 3 (1965), pp. 380–380.

[19] C. HOANG, Y. CHOI, AND K. CARLBERG, *Domain-decomposition least-squares Petrov–Galerkin (DD-LSPG) nonlinear model reduction*, Computer Methods in Applied Mechanics and Engineering, 384 (2021), p. 113997.

[20] T. Y. HOU AND X.-H. WU, *A multiscale finite element method for elliptic problems in composite materials and porous media*, Journal of Computational Physics, 134 (1997), pp. 169–189.

[21] T. J. HUGHES, G. R. FEIJÓO, L. MAZZEI, AND J.-B. QUINCY, *The variational multiscale method—a paradigm for computational mechanics*, Computer Methods in Applied Mechanics and Engineering, 166 (1998), pp. 3–24. Advances in Stabilized Methods in Computational Mechanics.

[22] D. HUYNH, D. KNEZEVIC, AND A. PATERA, *A static condensation reduced basis element method: Complex problems*, Computer Methods in Applied Mechanics and Engineering, 259 (2013), pp. 197–216.

[23] L. IAPICHINO, A. QUARTERONI, AND G. ROZZA, *Reduced basis method and domain decomposition for elliptic problems in networks and complex parametrized geometries*, Computers & Mathematics with Applications, 71 (2016), pp. 408–430.

[24] A. IOLLO, G. SAMBATARO, AND T. TADDEI, *A one-shot overlapping Schwarz method for component-based model reduction: application to nonlinear elasticity,*

Computer Methods in Applied Mechanics and Engineering, 404 (2023), p. 115786.

[25] A. KOEPPE, F. BAMER, AND B. MARKERT, *An intelligent nonlinear meta element for elastoplastic continua: deep learning using a new Time-distributed Residual U-Net architecture*, Computer Methods in Applied Mechanics and Engineering, 366 (2020), p. 113088.

[26] B. LAKSHMINARAYANAN, A. PRITZEL, AND C. BLUNDELL, *Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles*, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 6405–6416.

[27] K. LI, K. TANG, T. WU, AND Q. LIAO, *D3m: A deep domain decomposition method for partial differential equations*, IEEE Access, 8 (2020), pp. 5283–5294.

[28] W. LI, X. XIANG, AND Y. XU, *Deep domain decomposition method: Elliptic problems*, 2020.

[29] P. LINDSAY, J. FIKE, I. TEZAUR, AND K. CARLBERG, *Preconditioned least-squares Petrov–Galerkin reduced order models*, International Journal for Numerical Methods in Engineering, 123 (2022), pp. 4809–4843.

[30] Y. MADAY AND E. M. RØNQUIST, *A reduced-basis element method*, Journal of Scientific Computing, 17 (2002), pp. 447–459.

[31] A. MOTA, I. TEZAUR, AND C. ALLEMAN, *The schwarz alternating method in solid mechanics*, Computer Methods in Applied Mechanics and Engineering, 319 (2017), pp. 19–51.

[32] A. MOTA, I. TEZAUR, AND G. PHLIPOT, *The Schwarz alternating method for transient solid dynamics*, International Journal for Numerical Methods in Engineering, 123 (2022), pp. 5036–5071.

[33] C. PRUD'HOMME, D. ROVAS, K. VEROY, L. MACHIELS, Y. MADAY, A. PATERA, AND G. TURINICI, *Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods*, Journal of Fluids Engineering, 124 (2001), pp. 70–80.

[34] S. B. TANDALE, B. MARKERT, AND M. STOFFEL, *Smart stiffness computation of one-dimensional finite elements*, Mechanics Research Communications, 119 (2022), p. 103817.

[35] G. VALIENTE, *pocket-tensor*. https://github.com/GValiente/pocket-tensor, 2018.

[36] S. VOLKWEIN, *Model reduction using proper orthogonal decomposition*. Lecture Notes.

[37] K. XU, D. Z. HUANG, AND E. DARVE, *Learning constitutive relations using symmetric positive definite neural networks*, Journal of Computational Physics,

428 (2021), p. 110072.

A Proper orthogonal decomposition

Algorithm 1 provides the algorithm for computing the POD basis used in this work. We note that the basis dimension K can be determined from the decay of the singular values; for simplicity, we treat it as an algorithm input. We emphasize that the first

---

Algorithm 1 Proper orthogonal decomposition (POD)

Input: Snapshot matrix, $\mathbf{S} \in \mathbb{R}^{N \times N_s}$; desired basis dimension K.
Output: POD basis, $\boldsymbol{\Phi} \in \mathbb{R}^{N \times K}$, $K \leq N_s$.
Steps:

1. Compute the (thin) singular value decomposition

$$\mathbf{S} = \boldsymbol{U \Sigma V}^T, \tag{25}$$

 where $\boldsymbol{U} \equiv [\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_{N_s}]$.
2. Truncate the left singular vectors such that $\boldsymbol{\Phi}_i = \boldsymbol{u}_i$, $i = 1, \ldots, K$.

K singular vectors $\boldsymbol{u}_i$, $i = 1, \ldots, K$ form a basis which comprises the solution to, e.g., the optimization problem (7) with the input snapshot matrix being $\mathbf{F}^*$ or (8) with the input snapshot matrix being $\mathbf{U}$. The accuracy of the POD approximation can be bounded by energy contained in the truncated singular values, $\sum_{k=K}^{N_s} \boldsymbol{\Sigma}_{kk}^2$.