Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 415 (2023) 116219
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Geometric learning for computational mechanics, Part III: Physics-constrained response surface of geometrically nonlinear shells

Mian Xiao, Ran Ma, WaiChing Sun*

Department of Civil Engineering and Engineering Mechanics, Columbia University, 614 SW Mudd, Mail Code: 4709, New York, NY 10027, United States of America

Received 19 March 2023; received in revised form 25 June 2023; accepted 27 June 2023
Available online 3 August 2023

Abstract

This paper presents a graph-manifold iterative algorithm to predict the configurations of geometrically exact shells subjected to external loading. The finite element solutions are first stored in a weighted graph where each graph node stores the nodal displacement and nodal director. This collection of solutions is embedded onto a low-dimensional latent space through a graph isomorphism encoder. This graph embedding step reduces the dimensionality of the nonlinear data and makes it easier for the response surface to be constructed. The decoder, in return, converts an element in the latent space back to a weighted graph that represents a finite element solution. As such, the deformed configuration of the shell can be obtained by decoding the predictions in the latent space without running extra finite element simulations. For engineering applications where the shell is often subjected to concentrated loads or a local portion of the shell structure is of particular interest, we use the solutions stored in a graph to reconstruct a smooth manifold where the balance laws are enforced to control the curvature of the shell. The resultant computer algorithm enjoys both the speed of the nonlinear dimensional reduced solver and the fidelity of the solutions at locations where it matters.

Keywords: Machine learning; Graph neural network; Shell; Reduced order modeling

1. Introduction

Shells are structural elements commonly found in numerous applications, such as the roof of a building [1–4], the wings of airplanes and space structures [5], wind turbine blades [6], as well as deployable structures like parachutes [7] and papers [8]. In fact, shell structures all share one geometric feature — the kinematics of the mid-surface of the structure is sufficient to represent the kinematics of the deformed configurations.

To enforce the kinematics constraint in a finite element shell model, the *degenerated solid approach* [9–12] was widely adopted in the early 80 s. Meanwhile, capturing the geometrical nonlinearity becomes necessary for shells that undergo significant deformation to maintain the accuracy of predictions. In those cases, geometrically consistent shell models, such as [13,14], are often used such that the balance laws are parametrized in a way that avoids the

* Corresponding author.
E-mail address: wsun@columbia.edu (W. Sun).

https://doi.org/10.1016/j.cma.2023.116219

explicit appearance of the Riemannian connection of the mid-surface but captures the geometrical effects (cf. l Simo and Fox [15], Simo et al. [16], Ibrahimbegović [17], Roh and Cho [18]).

A key technical barrier in the computational modeling of the Reissner–Mindlin shell is to effectively represent the shell configuration in the geometrically nonlinear regime with a singularity-free parametrization expressed relative to an intrinsic frame for the surface displacement and director. Simo and Fox [15] resolved this issue by idealizing a shell as a Riemannian manifold, a subclass of differentiable manifolds with Euclidean tangential spaces [19]. Apart from shell structures, the Riemannian manifold has been used to represent a large variety of objects and defects of materials commonly encountered in engineering applications, including but not limited to films [20,21], crack surfaces [22,23], and lower-dimensional substructures in metamaterials or composites [24–27].

While geometrically exact shell finite elements may provide an improvement in computational efficiency over conventional 3D continuum finite element in the finite deformation regime, many applications of the shell, such as those used in generating vehicle crashing tests or response surface, often requires a proper dimensional reduction strategy to generate a large number of numerical solutions within limited time. The existing reduced-order models (ROM) for shell theory are classified into linear methods and nonlinear methods [28]. The linear model reduction method typically uses a set of orthogonal basis obtained via linear embedding techniques, such as the principal component analysis, or singular value decomposition [29,30], to reduce the dimensionality of the system of equations [31]. The hyper-reduction approaches, for example, the discrete empirical interpolation method (DEIM) [32] and the Gappy-POD scheme [33], further reduce the computation cost of nonlinear problems by selecting a subset of finite elements to perform stress update and assembly. The major disadvantage of linear model reduction methods is that the quality of the predictions often depends on the choices of the orthogonal basis used to perform the projection [34]. As such, a successful reduced order model often requires a good sampling strategy [31] as well as the inclusion of additional vectors in the basis, such as modal derivatives [35] and dual modes [36]. The nonlinear methods, for example, the invariant manifold theory [28], construct a curved manifold instead of a linear space in the phase space to approximate the trajectories of the dynamic system.

On the other hand, the dataset used for generating the orthogonal bases (through, for instance, the method of snapshot [37–39]) could sufficiently populate the parametric space. In this sense, this dataset may contain sufficient information to generate a *solution manifold* so that without any further full-scale or reduced-order simulations, we may obtain any element within a class of solution with respect to a parametric domain (say different tractions applied at the same area of the shell). This solution manifold can be regarded as a hypersurface in a $(N + P)$-dimensional Euclidean space where N is the dimension of the finite element space, and P is the total number of parameters necessary to parametrize the external loading. Such a solution manifold can then serve as digital twins where the solution manifold may directly predict the responses of the shell structures from a specific class of external loadings.

This paper is the Part III of the geometric learning for computational mechanics series in which we continue to leverage geometric learning techniques showcased in Vlassis et al. [40], Vlassis and Sun [41] for solid mechanics problems. The purpose of this paper is to build a response surface model that forecasts behaviors of geometrical nonlinear shells. To simplify the problem, we restrict our attention to the problems where bifurcations or instability, such as buckling, do not occur [42]. This assumption eliminates the branching of the solution and ensures that each deformed configuration corresponds to a particular external loading represented as a point in the parametric space. We then use a predictor–corrector algorithm to generate the solution manifold/hypersurface in a $(N+P)$-dimensional Euclidean space. In the predictor step, we first reduce the complexity of the learning problems by translating the finite element solution of the Simo-Fox-Rafai shell as a weighted graph. This graph representation provides us another opportunity to perform nonlinear dimensional reduction through graph embedding. As the graph embedding maps each solution graph onto a point in the latent space, we then use neural networks to create a hypersurface in a $(N + P)$-dimensional Euclidean space where all the solution data is supposed to be lied on. In the corrector step, we introduce corrections to ensure the admissibility of the solution predicted by the neural network. This correction is done by applying a physics-informed correction that re-parametrizes a small domain of interest such that enforcing the balance principles would not lead to a difficult high-dimensional non-convex optimization problem that is often attributed to the failures of some physics-informed neural network in the literature (cf. Krishnapriyan et al. [43]).

The rest of the paper is organized as follows: Section 2 presents a detailed graph representation of the shell manifold and the graph-manifold learning formulation. Section 3 introduces the enforcement of physical governing equations with implementation perspectives. Section 4 shows two numerical examples with problem configurations to demonstrate how the dataset is collected from finite element simulation results. Section 5 lists prediction results for the two examples of interest that justify the correctness and applicability of the proposed geometric learning model. Section 6 summarizes our findings.

2. Deep geometric learning with graph neural networks

In this section, we present the predictor part of the predictor–corrector algorithm, where the deformed shell configuration is predicted directly from the graph autoencoder architecture without any online simulation. In Section 2.1, we explain the interpretation of the deformed shell fields as an undirected node-weighted graph and the setting of the neural network for multi-fidelity data augmentation. In Section 2.2, we demonstrate how the graph autoencoder embeds the deformed shell configurations onto a lower-dimensional latent space for reduced-order predictions through the graph isomorphism layers. The node embedding is performed by considering nodes within the same finite element as a one-hop neighborhood. In summary, we obtain the solution manifold by performing the following three tasks, i.e.,

1. Generate an augmented high fidelity set ${}^h\tilde{\mathbb{G}} = \{{}^h\tilde{\mathcal{G}}'_1, \ldots, {}^h\tilde{\mathcal{G}}'_{T_l}\}$ with data consisting of fine-mesh data ${}^h\tilde{\mathcal{G}}'_j$ and interpolated data from the coarse mesh ${}^l\mathcal{G}'_j$, such that ${}^h\tilde{\mathcal{G}}'_j \sim {}^l\mathcal{G}'_j$. (see Section 2.2.2)
2. Perform graph embedding based on the combined high fidelity set ${}^h\tilde{\mathbb{G}}$. (see Section 2.2.2)
3. Learn the governing dynamics that maps $\boldsymbol{p}$ to the lower-ordered latent space (See Section 2.3).

where $\sim$ indicates that ${}^h\tilde{\mathcal{G}}'_j$ is the high fidelity result on the finer mesh corresponding to the low fidelity result ${}^l\mathcal{G}'_j$ of the same loading condition. For simplicity, we assume that all snapshots in ${}^h\mathbb{G}$ come from results on the same fine mesh, while all snapshots in ${}^l\mathbb{G}$ come from results on the same mesh that is much coarser than their fine counterpart.

2.1. Graph representation for shells discretized by different meshes

To ensure the robustness of the response surface without exhausting our resources for high-fidelity simulations, we introduce a data augmentation technique in which we may run a limited amount of high-fidelity simulations (or experimental tests) for the important parametric domains while utilizing more cost-efficient low-fidelity simulations (in our cases, the same finite element simulations with a coarser mesh) to populate the less critical regions of the parametric domain.

We consider each node of the finite elements as a graph vertex. Graph edges are assigned for each pair of vertices that are nodes of a finite element edge [41]. accordingly, each nodal solution can be stored as the weight of the corresponding vertex such that the finite element solution can be stored as an undirected node-weighted graph. By assuming that we only use the same set of bases for the testing and interpolating functions of all finite elements, we eliminate the need to introduce edge weights doe the edge set. This setting simplifies the graph representation of the finite element solutions.

In the following part of this section, we will introduce the mathematical expression of finite element node graph as a foundation for our machine learning model. A finite element node graph is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i | \, i = 1, \ldots, N\}$ is the node set of the graph as v_i corresponding to individual finite element nodes, and $\mathcal{E} = \{(v_{1j}, v_{2j}) | \, j = 1, \ldots, M; \;\; v_{1j}, v_{2j} \in \mathcal{V}\}$ is the edge set where the existence of each individual edge indicates that node v_{1j} and v_{2j} belong to the same element, as shown in Fig. 1. N and M indicate the size of the node set and edge set. In order to deal with geometrical features to be incorporated in the machine learning model, we enrich the graph representation as a node-weighted graph: $\mathcal{G}' = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ where $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^D | i = 1, \ldots, N\}$ is the nodal feature set as $\boldsymbol{x}_i$ indicates the geometrical feature vector at node v_i with a dimension of D. We may enforce $D \geq 3$ as the manifold reconstruction task requires at least the three spatial coordinates of the finite element point cloud.

With the finite element node graph defined, the fundamental geometric learning task for the reduced order modeling on the shell can be stated as follows: given a series of snapshots of the deformed shell manifold of the same initial configuration and path-independent material properties $\mathbb{G} = \{\mathcal{G}'_1, \ldots, \mathcal{G}'_T\}$ sampled by a subset of the parametrized load $\overline{\boldsymbol{p}} \in \mathbb{R}^P$, we would like to predict the shell manifold geometry $\mathcal{G}'$ deformed by any element of $\boldsymbol{p} \in \mathbb{R}^P$ via the reduced-order latent space of $\mathbb{G}$.

In the geometric learning literature, the embedding that performs the feature engineering or nonlinear dimensional reduction is often referred to as the upstream task, whereas the predictions that leverage the feature to make predictions is called the downstream application [44,45]. In our case, we follow the same workflow. We first embed

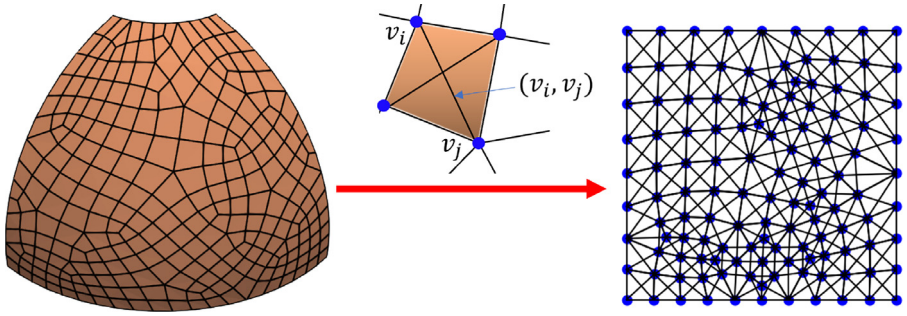


Fig. 1. Node graph interpretation of a finite element mesh.

these shell manifold snapshots represented by node-weighted graphs onto a low-dimensional latent space $\mathbb{R}^{D_{\text{enc}}}$ (upstream task), then we learn how $\boldsymbol{p}$ governs the shell configuration not via the graph or the original finite element space, but in the low-dimensional latent space $\mathbb{R}^{D_{\text{enc}}}$, where $D_{\text{enc}} \ll N$ is the dimension of the latent space.

To establish a trade-off between fidelity and cost efficiency, the collection of snapshots in $\mathbb{G}$ that constitute the database for training the digital twins may come from finite element meshes **of different resolutions**. In our numerical experiments, we collect data from two finite element meshes, i.e.,

– High-fidelity set: ${}^{h}\mathbb{G} = \{{}^{h}\mathcal{G}'_1, \ldots, {}^{h}\mathcal{G}'_{T_h}\}$.
– Low-fidelity set ${}^{l}\mathbb{G} = \{{}^{l}\mathcal{G}'_1, \ldots, {}^{l}\mathcal{G}'_{T_l}\}$

where $T_h \ll T_l$ and ${}^{h}\mathbb{G} \cap {}^{l}\mathbb{G} = \emptyset$ and ${}^{h}\mathbb{G} \cup {}^{l}\mathbb{G} = \mathbb{G}$.

Here we leverage the fact that message-passing GIN may enable embedding graphs of different sizes to design an algorithm capable of inferring the unseen fine-mesh finite element solutions from a multi-fidelity database. To achieve this objective, we introduce a neural network augmentation mechanism to pre-process the coarse mesh data before embedded in the graph autoencoder architecture such that the reconstructed graph is corresponding to the fine mesh used to generate ${}^{h}\mathbb{G}$. (see Fig. 2). In analog to a prolongation of a multigrid solver, we train a neural network to perform a nonlinear mapping that maps the node features of the coarse-mesh graphs onto those of the fine-mesh graphs. This step then enables us to combine the actual fine-mesh data and the fine-mesh data interpolated from the coarse mesh to perform the graph embedding such that a single unified latent space can be used to represent both sets of data. Provided that the mapping is successful, this step enables us to reduce the cost of data generation by only administrating fine-mesh simulations in the domain of interest.

Remark 1. Other feasible choices of representing finite element solutions in a mesh may include node graph, dual graph, communication graph [46] as well as tree [47].

2.2. The geometric learning model with graph neural networks

This section demonstrates how we resolve the aforementioned geometric learning subtasks. We first introduce the neural network architecture used in Section 2.2.1. Then, we present the formulation of these learning tasks: Section 2.2.2 shows how we learn the mapping for building ${}^{h}\tilde{\mathbb{G}}$ and then find the reduced ordered latent space; Section 2.3 shows how to predict the shell geometry with $\boldsymbol{p}$ based on the latent space. A sketch of the modeling procedures in this section is summarized in Fig. 2.

2.2.1. The graph autoencoder architecture

We adopt the graph isomorphism network (GIN) (cf. Xu et al. [48]) to perform the embedding task. We use GIN because it is a message-passing model capable of discriminating different graph structures identified by the Weisfeiler–Leman isomorphism test [49]. Providing that we use a proper graph pooling layer, the embedding of GIN is inherently permutation invariance, which means that the ordering of the nodes will not affect the predictions. More importantly, the fact that GIN passes the isomorphism test enables us to distinguish non-isomorphic subgraphs

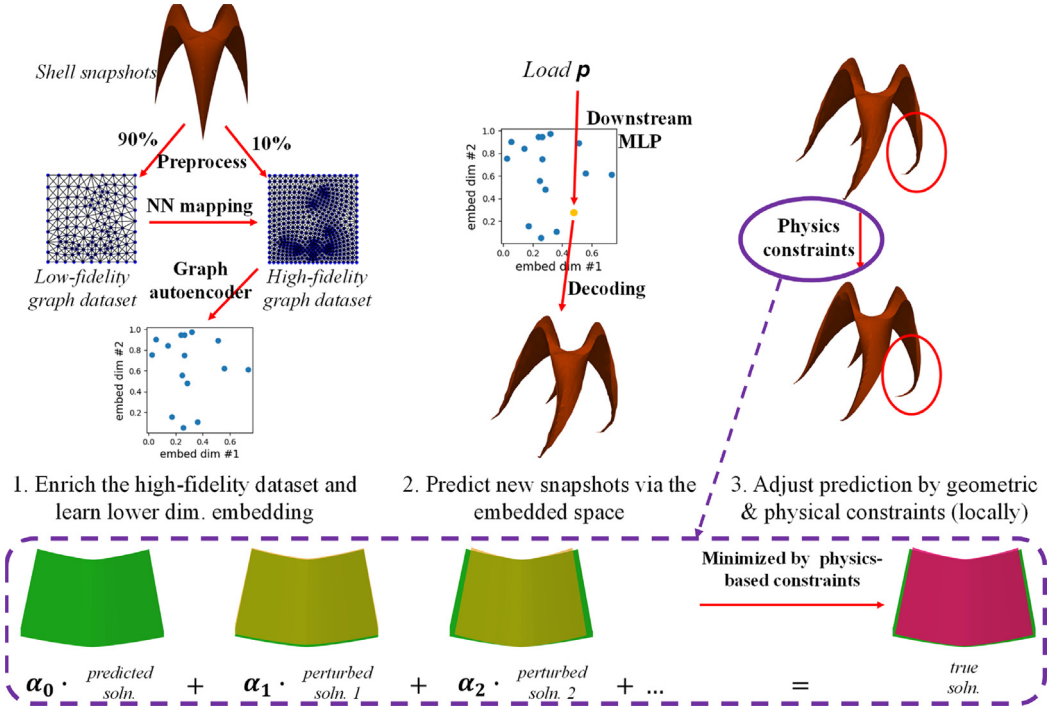


Fig. 2. Procedure sketch of the proposed geometric model. Notice that the portion number (90% low-fidelity data and 10% high-fidelity data) in Step 1 is not strict but related to the capacity to acquire high-fidelity simulation results.

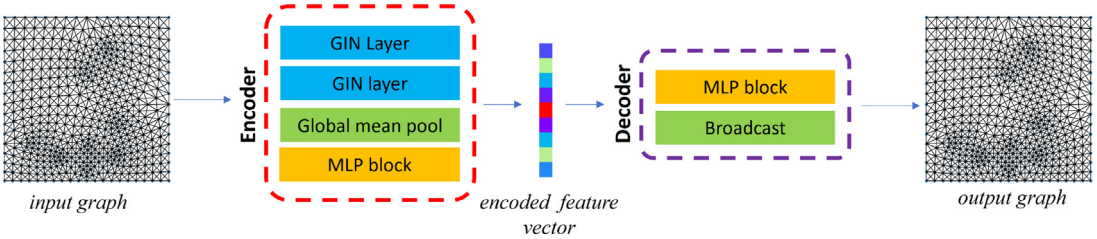


Fig. 3. Graph autoencoder architecture. Notice that the input and output graphs are not necessarily sharing the same adjacency matrix.

by mapping them onto different encoded latent vectors and vice versa – a feat that cannot be achieved by the conventional graph convolutional network and GraphSAGE [48]. These two features combined improve the expressive power of the GIN such that the relationships among finite element nodes can be captured by the neural network. Fig. 3 shows the architecture of the graph autoencoder designed for the shell problems. The encoder part of this architecture takes in the adjacency matrix and feature matrix of the input graph $\mathcal{G}'_{\text{in}} : (\boldsymbol{A}_{\text{in}}, \boldsymbol{X}_{\text{in}})$ and produces an encoded feature vector $\boldsymbol{h}_{\text{enc}}$, which is denoted as a functional expression: $\boldsymbol{h}_{\text{enc}} = \text{Enc}(\boldsymbol{X}_{\text{in}}, \boldsymbol{A}_{\text{in}})$. The decoder part of this architecture then utilizes the encoder output to produce a decoded feature matrix $\tilde{\boldsymbol{X}}$. The adjacency matrix of the output graph $\boldsymbol{A}_{\text{out}}$ can be assumed as a prior (since each dataset shares one same $\boldsymbol{A}$) in order to complete an output graph $\mathcal{G}'_{\text{out}} : (\boldsymbol{A}_{\text{out}}, \tilde{\boldsymbol{X}})$, which could be written as $\tilde{\boldsymbol{X}} = \text{Dec}(\boldsymbol{h}_{\text{enc}})$. Problem formulation is generally based on supervision of the decoded output $\tilde{\boldsymbol{X}}$, which will be the focus of the following sections. The rest of this section will present details about how the layer components shown in Fig. 3 operate.

We first introduce one of the most widely-used architectures called multi-layer perceptron (MLP), which is included as a substructure in our graph autoencoder architecture. MLP is a functional approximation expressed as follows:

$$\text{MLP}(\boldsymbol{X}) = \boldsymbol{W}^{(K)} \cdot \text{act}(\boldsymbol{W}^{(K-1)} \cdot \text{act}(\ldots \text{act}(\boldsymbol{W}^{(1)} \cdot \boldsymbol{X} + \boldsymbol{b}^{(1)}) \ldots) + \boldsymbol{b}^{(K-1)}) + \boldsymbol{b}^{(K)} \tag{1}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are called the weight matrix and the bias vector of the MLP substructure, respectively. The superscript (K) indicates the Kth layer of the MLP substructure. Meanwhile, act$(\cdot)$ denotes the activation function of individual layers. In this work, we adopt the rectified linear unit (ReLU) for act$(\cdot)$ such that $\mathrm{ReLU}(x) = x$ if $x > 0$; otherwise, $\mathrm{ReLU}(x) = 0$.

We then focus on the GIN convolution layers, which take in both the adjacency matrix and feature matrix and output an embedded feature matrix. The matrix formulation of a GIN layer is:

$$\boldsymbol{H}^{(k)} = \mathrm{MLP}^{(k)}\left((\boldsymbol{A} + (1+\epsilon)\boldsymbol{I}) \cdot \boldsymbol{H}^{(k-1)}\right) \tag{2}$$

where the superscript (k) indicates the kth layer in the architecture; $\boldsymbol{H}$ is the embedded nodal feature matrix coming from the output of the previous layer with $\boldsymbol{H}^{(1)} = \boldsymbol{X}$ at the input layer. ϵ is a learnable parameter. For consecutive GIN convolution layers, the following layer accepts the same $\boldsymbol{A}$ as the previous layer. For the beginning GIN layer in both the encoder and the decoder, $\boldsymbol{A}$ should be prescribed as either $\boldsymbol{A}_{\text{in}}$ or $\boldsymbol{A}_{\text{out}}$.

Our architecture also includes global operations on the graph. The graph global mean pooling operation in the encoder performs the following computation:

$$\boldsymbol{h}_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{h}_i \tag{3}$$

where $\boldsymbol{h}_i$ is the embedded nodal feature of v_i corresponding to the ith row of $\boldsymbol{H}$, and $\boldsymbol{h}_{\text{avg}}$ is the resultant graph feature vector from global mean pooling.

The broadcasting operation in the decoder is a matrix reshape operation that converts a row vector $\boldsymbol{h}_{\text{dec}}$ of size ND_{enc} obtained from the output of the MLP substructure in the decoder to the corresponding embedded nodal feature matrix of size $N \times D_{\text{enc}}$ such that:

$$h_{ij} = (\boldsymbol{h}_{\text{dec}})_{j+(i-1)D_{\text{enc}}} \tag{4}$$

where h_{ij} indicates the jth component of the embedded nodal feature vector $\boldsymbol{h}_i$ after broadcasting.

2.2.2. The learning problem for the high-fidelity shell responses

In the previous section, we discuss the strategy to construct the mapping $\mathcal{F}({}^{l}\mathcal{G}'_j) = {}^{h}\tilde{\mathcal{G}}'_j$ such that ${}^{h}\tilde{\mathcal{G}}'_j \sim {}^{l}\mathcal{G}'_j$ to establish an augmented data set ${}^{h}\tilde{\mathbb{G}}$. We assume that $\mathcal{F}$ is learned in a supervised manner: we aim to minimize the discrepancy between the nodal feature matrix of training labels and the approximated nodal feature matrix output by the neural network. We intuitively construct the training labels with ${}^{h}\mathbb{G}$, and thus we required that for each snapshot in ${}^{h}\mathbb{G}$ there exist some snapshot in ${}^{l}\mathbb{G}$ corresponding to the results with the same loading condition, which is summarized as follows:

$$\exists\, {}^{lh}\mathbb{G} \subset {}^{l}\mathbb{G} \quad \text{s.t.} \quad {}^{h}\mathbb{G} = \left\{\mathcal{F}({}^{lh}\mathcal{G}'_j) \middle| {}^{lh}\mathcal{G}'_j \in {}^{lh}\mathbb{G}\right\} \tag{5}$$

The subset ${}^{lh}\mathbb{G}$ then constructs the training input set. We next approximate $\mathcal{F}$ with the graph autoencoder (proposed in Section 2.2.1) by computing the decoded nodal features as $\tilde{\boldsymbol{X}} = \hat{\boldsymbol{F}}(\boldsymbol{X})$. The loss function is adopted as the node-wise mean square error of the nodal feature matrix, which leads to the following training objective:

$$\min_{\boldsymbol{\Theta}^F} \frac{1}{N_s^{lh}} \sum_{i=1}^{N_s^{lh}} \left\| \hat{\boldsymbol{F}}(\boldsymbol{X}^{lh}_{(i)}) - \boldsymbol{X}^{h}_{(i)} \right\|^2_{\text{fro}}, \quad \hat{\boldsymbol{F}}(\boldsymbol{X}) = \mathrm{Dec}_F(\mathrm{Enc}_F(\boldsymbol{X}, \boldsymbol{A}^l)) \tag{6}$$

where $\boldsymbol{\Theta}^F$ is the collection of all trainable network parameters of $\hat{\boldsymbol{F}}(\cdot)$. N_s^{lh} is the size of ${}^{lh}\mathbb{G}$ as well as ${}^{h}\mathbb{G}$. The subscript (i) indicates the ith sample in ${}^{h}\mathbb{G}$ or ${}^{lh}\mathbb{G}$, while the sample sequence satisfies ${}^{h}\mathcal{G}'_i \sim {}^{lh}\mathcal{G}'_i$ with ${}^{h}\mathcal{G}'_i : (\boldsymbol{A}^h, \boldsymbol{X}^h_{(i)})$ and ${}^{lh}\mathcal{G}'_i : (\boldsymbol{A}^l, \boldsymbol{X}^{lh}_{(i)})$. The operator $\|\cdot\|_{\text{fro}}$ indicates the Frobenius norm of a matrix. The subscript F of Dec and Enc indicates the decoder and encoder function for the graph autoencoder approximating $\mathcal{F}$, in order to differentiate from the reconstruction autoencoder mentioned in the following part. The approximated mapping $\hat{\boldsymbol{F}}(\cdot)$ helps us to enrich the high-fidelity dataset as follows:

$${}^{h}\tilde{\mathbb{G}} = \left\{ {}^{h}\tilde{\mathcal{G}}'_j : (\boldsymbol{A}^h, \tilde{\boldsymbol{X}}^h_{(j)}) \mid \tilde{\boldsymbol{X}}^h_{(j)} = \begin{cases} \hat{\boldsymbol{F}}(\boldsymbol{X}^l_{(j)}), & {}^{l}\mathcal{G}'_j : (\boldsymbol{A}^l, \boldsymbol{X}^l_{(j)}) \in {}^{l}\mathbb{G} \setminus {}^{lh}\mathbb{G}, \\ \boldsymbol{X}^l_{(j)}, & {}^{l}\mathcal{G}'_j : (\boldsymbol{A}^l, \boldsymbol{X}^l_{(j)}) \in {}^{lh}\mathbb{G}. \end{cases} \right\} \tag{7}$$

Here we realize the significance of having a low-fidelity dataset ${}^{l}\mathbb{G}$ and the mapping $\mathcal{F}$: we are now able to construct a relatively large high-fidelity dataset to improve the reduced ordered model without spending too much effort performing experiments on the high-fidelity scales.

With ${}^{h}\tilde{\mathbb{G}}$ populated from ${}^{l}\mathbb{G}$, we are ready to formulate the graph embedding problem that determines the reduced ordered latent space. The general idea is to construct a graph autoencoder function $\hat{\boldsymbol{R}}(\cdot)$ whose output approximates its input itself. We still adopt the loss function as the node-wise mean square error for the feature matrix between the labels in ${}^{h}\tilde{\mathbb{G}}$ and the output from $\hat{\boldsymbol{R}}(\cdot)$, which yields the following training objective:

$$\min_{\boldsymbol{\Theta}^R} \frac{1}{N_s^{hh}} \sum_{i=1}^{N_s^{hh}} \left\| \hat{\boldsymbol{R}}(\tilde{\boldsymbol{X}}_{(i)}^h) - \tilde{\boldsymbol{X}}_{(i)}^h \right\|_{\text{fro}}^2, \quad \hat{\boldsymbol{R}}(\boldsymbol{X}) = \text{Dec}_R(\text{Enc}_R(\boldsymbol{X}, \boldsymbol{A}^h)) \tag{8}$$

where $\boldsymbol{\Theta}^R$ is the collection of all trainable network parameters of $\hat{\boldsymbol{R}}(\cdot)$. N_s^{hh} is the size of ${}^{h}\tilde{\mathbb{G}}$ as well as ${}^{l}\mathbb{G}$. The subscript R indicates the encoder and decoder function of the reconstruction autoencoder. The reduced ordered latent space $\mathbb{L}$ is then defined as the space spanned by $\boldsymbol{h}_{\text{enc}} = \text{Enc}_R(\boldsymbol{X}, \boldsymbol{A}^h)$ for arbitrary $\boldsymbol{X}$ coming from an admissible deformed shell configuration $\mathcal{G}'$, where $\mathbb{L} \subset \mathbb{R}^{D_{\text{enc}}}$.

2.3. Predicting the high-fidelity results without full-scale simulations

This section presents how we utilize the parametrized load $\boldsymbol{p}$ to predict the actual deformed shell configuration $\mathcal{G}'$. As we find the reduced ordered latent space $\mathbb{L}$ in the previous section, we may notice that $\mathbb{L}$ is generally entangled, which does not ideally captures the reduced ordered dynamics in the maximum sense. We here propose to construct a response controlling law on $\mathbb{L}$ based on $\boldsymbol{p}$ corresponding to the shell configuration of interest to disentangle the reduced ordered latent space, denoted as a functional expression $\boldsymbol{h}_{\text{enc}} = f(\boldsymbol{p})$. We then approximately parametrize the response controlling law function as $\hat{\boldsymbol{f}}(\cdot)$ by a simple feed-forward neural network with MLP architecture. We fit $\hat{\boldsymbol{f}}(\cdot)$ with the mean square error loss between the encoded feature vector obtained from the graph autoencoder and that computed by $\hat{\boldsymbol{f}}(\cdot)$, which yields the following learning objective:

$$\min_{\boldsymbol{\Theta}^f} \frac{1}{N_s^{hh}} \sum_{i=1}^{N_s^{hh}} \left\| \hat{\boldsymbol{f}}(\boldsymbol{p}_{(i)}) - \text{Enc}_R(\tilde{\boldsymbol{X}}_{(i)}^h, \boldsymbol{A}^h) \right\|_2^2 \tag{9}$$

where $\boldsymbol{\Theta}^f$ is the collection of all trainable network parameters of $\hat{\boldsymbol{f}}(\cdot)$. $\boldsymbol{p}_{(i)}$ is the loading condition corresponding to the ith snapshot ${}^{h}\tilde{\mathcal{G}}_i' : (\boldsymbol{A}^h, \tilde{\boldsymbol{X}}_{(i)}^h)$. The operator $\|\cdot\|_2$ is the vector Euclidean norm.

After we learn the neural network approximation of $f(\cdot)$, the prediction of high-fidelity results is to determine the nodal feature matrix $\bar{\boldsymbol{X}}$ given some loading condition $\bar{\boldsymbol{p}}$ as follows:

$$\bar{\boldsymbol{X}} = \text{Dec}_R(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})) \tag{10}$$

In essence, we introduce a graph neural network approach to construct a response surface. Since (1) the data are obtained from the simulations that obey the balance principles and (2) a successful embedding should be capable of preserving the relationships among nodes, we hypothesize that this will give us more accurate and robust predictions than other alternatives that employ basis functions that directly interpolate the hypersurface in the ambient space $\mathbb{R}^{N+P}$. This hypothesis will be tested in Section 5. For the domain of interest where the balance principle must be precisely enforced, we introduce a locally-enhancement technique that can improve the fidelity via the tangent space of the nonlinear solution manifold.

3. Local enrichment with geometrical and physical constraints on the machine learning prediction

The previous section shows how the graph representation of a deformed shell manifold is predicted. For applications where the mechanical response at a specific location is of interest (potential singularity, extreme deformation), the validity of the machine learning predictions for mechanics problems should be verified by the physical laws. They can be enforced with a regularization term in the training objective [50–52], where a notable example is the *physics-informed learning* [53], or with prior assumptions in the solution functions [54,55], where a notable example is the *neural operator* [56].

In this work, the physics-based corrector is implemented in a reduced-order fashion instead of traditional loss regularization. We reparametrize a local region of interest on the shell configuration as a linear combination of functional basis derived from the direct neural network prediction. We then determine the correction of neural network predictions from Section 2 by computing physical properties (e.g., value of the residual equation) and formulate the optimization problem accordingly. Note that directly imposing this additional constraint into the loss function of the autoencoder may significantly increase the time required to train the autoencoder. As such, we decide to introduce this extra postprocessing step, which improves the fidelity at the cost of longer execution time in the online simulation. An alternative where the constraint is directly incorporated into the training of the autoencoder will be considered in the future but is out of the scope of this study.

3.1. Physical constraints for geometrically nonlinear shell

This section briefly summarizes the geometrically exact shell model we adopt [15], for the convenience of introducing the governing equations as physical constraints. In this model, we consider the shell configuration as a 2D Riemann manifold with finite extent in the actual 3D space, where the positional information of the shell is indicated by the mid-surface location $\boldsymbol{\varphi}$. The geometrically exact feature is achieved by introducing a vector field $\boldsymbol{t}$, also called director, to account for the bending and shear effect of the shell. These vector fields $\boldsymbol{\varphi}$ and $\boldsymbol{t}$ formulate the basic representation of the shell geometry and facilitate the definition of shell strain measures and the evaluation of shell stress measures. Due to this two-field mixed formulation, there exist two residual equations as the target of the physical constraint evaluation: one is associated with linear momentum balance, denoted as $\boldsymbol{\mathcal{R}}^n$; the other is associated with the director momentum balance, denoted as $\boldsymbol{\mathcal{R}}^m$. The necessary ingredients for evaluating physical constraints are summarized in Table 1. Meanwhile, the shell response is assumed quasi-static.

Remark 2. $\boldsymbol{\varphi}$ and $\boldsymbol{t}$ are deformable mappings from a static 2D parametric space $\mathcal{A} \subset \mathbb{R}^2$ to the actual 3D space $\mathbb{R}^3$ and the unit sphere $\boldsymbol{S}^2$. For notational convenience, we use (ξ, η) to denote the parametric coordinates in $\mathcal{A}$ from now on. This enables us to express the shell configuration mappings in functional forms:

$$\boldsymbol{\varphi}(\xi, \eta) = [\varphi_1(\xi, \eta)\ \ \varphi_2(\xi, \eta)\ \ \varphi_3(\xi, \eta)]^T \tag{11}$$

and

$$\boldsymbol{t}(\xi, \eta) = [t_1(\xi, \eta)\ \ t_2(\xi, \eta)\ \ t_3(\xi, \eta)]^T, \tag{12}$$

where φ_i and t_i $(i = 1, 2, 3)$ indicate the componentwise location and director fields, respectively. All super and subscripts α, β, μ listed in Table 1 are associated with these parametric coordinates implicitly, where the number 1 corresponds to ξ and the number 2 corresponds to η.

3.2. Reduced-order modeling via tangential space of the solution manifold

In the previous section, we argue that the shell configuration vector fields $\boldsymbol{\varphi}$ and $\boldsymbol{t}$ provide sufficient information for evaluating the geometric and physical constraint equations. One possible remedy to improve the compatibility of the balance principles without sufficiently slowing down the prediction is to introduce Galerkin projection via a collection of orthogonal bases. In theory, these orthogonal bases can be generated through principal component analysis performed on the entire collection of solutions, and the fidelity can be improved by incorporating more orthogonal bases at the expense of computational speed. However, this global linear embedding strategy is known to yield an unsatisfactory fidelity-speed tradeoff for highly nonlinear problems. As such, we introduce an empirical approach in which we establish a locally linear embedding where we identify the tangential space at the parametric load $\boldsymbol{p}$ where we sought the solution. Then we use the basis of this tangential space to establish the Galerkin projection such that the resultant Gakerlin projection may vary according to the types of loadings $\boldsymbol{p}$, a technique commonly used to model high-dimensional data (see, for instance, Donoho and Grimes [57]). A similar strategy has also been used in He et al. [58], He and Chen [59] where a locally convex space is established via points in a constitutive manifold for data-driven simulations.

Table 1
Summary of the governing equations of the quasi-static geometrically exact shell model adopted for verifying the physical constraints locally. All super and subscripts $\alpha, \beta, \mu = 1, 2$.

Governing equations		Nomenclature or explanation	
Surface frame and Jacobian	$\boldsymbol{a}_{0\alpha} = \boldsymbol{\varphi}_{0,\alpha}, \ \boldsymbol{a}_\alpha = \boldsymbol{\varphi}_{,\alpha}$	$\boldsymbol{a}_{0\alpha}, \boldsymbol{a}_\alpha$	initial and convected surface frame vector (tangential basis)
	$J = \|\boldsymbol{a}_1 \times \boldsymbol{a}_2\|_2$	J	surface Jacobian
Kinematic relationships	$a_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta \ \gamma_\alpha = \boldsymbol{a}_\alpha \cdot \boldsymbol{t}$ $\kappa_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{t}_{,\beta}$	$a_{\alpha\beta}$ $\gamma_{\alpha\beta}$ $\kappa_{\alpha\beta}$	kinematic variables for computing the effective strain measures in the following equations
Effective strains	$\varepsilon_{\alpha\beta} = \frac{1}{2}(a_{\alpha\beta} - a_{0\alpha\beta})$ $\delta_\alpha = \gamma_\alpha - \gamma_{0\alpha}$ $\rho_{\alpha\beta} = \kappa_{\alpha\beta} - \kappa_{0\alpha\beta}$	$\varepsilon_{\alpha\beta}$ δ_α $\rho_{\alpha\beta}$	effective membrane strain effective shear strain effective coupled strain
Frame vector constraint	$\boldsymbol{t}_{,\alpha} = \lambda^\mu_\alpha \boldsymbol{a}_\mu + \lambda^3_\alpha \boldsymbol{t}$ $\boldsymbol{t} \cdot \boldsymbol{t}_{,\alpha} = 0$ $\lambda^3_\alpha = -\lambda^\mu_\alpha \gamma_\mu$	λ^μ_α	coefficients of the frame vectors $\boldsymbol{a}_\mu$, $\boldsymbol{t}$ after the vector decomposition of $\boldsymbol{t}_{,\alpha}$
Linear momentum balance the director momentum balance	$\boldsymbol{\mathcal{R}}^n = \frac{1}{J}(J\boldsymbol{n}_\alpha)_{,\alpha} = 0$ $\boldsymbol{\mathcal{R}}^m = \frac{1}{J}(J\tilde{\boldsymbol{m}}_\alpha)_{,\alpha} - \bar{\boldsymbol{l}} = 0$	$\boldsymbol{n}_\alpha$ $\tilde{\boldsymbol{m}}_\alpha, \bar{\boldsymbol{l}}$	stress resultant stress couple, across-the-thickness stress resultant
Decomposition of stress resultants and stress couples	$\boldsymbol{n}_\alpha = n_{\beta\alpha}\boldsymbol{a}_\beta + q_\alpha \boldsymbol{t}$ $\tilde{\boldsymbol{m}}_\alpha = \tilde{m}_{\beta\alpha}\boldsymbol{a}_\beta + \tilde{m}_{3\alpha}\boldsymbol{t}$ $\bar{\boldsymbol{l}} = \bar{\lambda}\boldsymbol{t} + (\tilde{q}_\alpha + \lambda^\alpha_\mu \tilde{m}_{3\mu})\boldsymbol{a}_\alpha$	$n_{\beta\alpha}, \ q_\alpha$ $\tilde{m}_{\beta\alpha}, \ \tilde{m}_{3\alpha}$ $\bar{\lambda}$	stress resultant components effective stress couple components Lagrange multiplier of $\boldsymbol{t}$
Effective stress resultants	$\tilde{n}_{\beta\alpha} = n_{\beta\alpha} - \lambda^\beta_\mu \tilde{m}_{\alpha\mu}$ $\tilde{q}_\alpha = q_\alpha + \lambda^\beta_\mu \gamma_\beta \tilde{m}_{\alpha\mu}$	$\tilde{n}_{\beta\alpha}$ $\tilde{q}_\alpha$	effective membrane stress effective shear stress
Constitutive laws	$\psi(\varepsilon_{\beta\alpha}, \delta_\alpha, \rho_{\beta\alpha})$ $\tilde{n}_{\beta\alpha} = \partial\psi/\partial\varepsilon_{\beta\alpha}$ $\tilde{q}_\alpha = \partial\psi/\partial\delta_\alpha$ $\tilde{m}_{\beta\alpha} = \partial\psi/\partial\rho_{\beta\alpha}$	ψ	assumed elastic energy functional

As a result, we may formulate the problem for enforcing physical constraints as an optimization problem to minimize the norm of residual vectors $\boldsymbol{\mathcal{R}}^n$, $\boldsymbol{\mathcal{R}}^m$ with $\boldsymbol{\varphi}$ and $\boldsymbol{t}$ as governing variables. Functional variables $\boldsymbol{\varphi}$ and $\boldsymbol{t}$ are parametrized as a linear combination of functional basis that is perturbed from the predicted solution:

$$\begin{cases} \boldsymbol{\varphi} & = c_0 \hat{\boldsymbol{\varphi}}|_{\boldsymbol{p}} + \sum_k^{\tilde{K}} c_k \hat{\boldsymbol{\varphi}}|_{\boldsymbol{p}+\delta\boldsymbol{p}_k} \\ \boldsymbol{t} & = c_0 \hat{\boldsymbol{t}}|_{\boldsymbol{p}} + \sum_k^{\tilde{K}} c_k \hat{\boldsymbol{t}}|_{\boldsymbol{p}+\delta\boldsymbol{p}_k} \\ 1 & = c_0 + \sum_k^{\tilde{K}} c_k \end{cases} \tag{13}$$

where $\tilde{K}$ is the number of functional basis. $\hat{\boldsymbol{\varphi}}$ and $\hat{\boldsymbol{t}}$ are trained neural network functions for the local solution predicted at the load specified by the load vector in the subscript ($\boldsymbol{p} + \delta\boldsymbol{p}_k$), and non-degenerated basis is preferred such that: $\text{Rank}(\text{span}\{\hat{\boldsymbol{\varphi}}|_{\boldsymbol{p}+\delta\boldsymbol{p}_1}, \ldots \hat{\boldsymbol{\varphi}}|_{\boldsymbol{p}+\delta\boldsymbol{p}_{\tilde{K}}}\}) = \tilde{K}$, $\text{Rank}(\text{span}\{\hat{\boldsymbol{\varphi}}|_{\boldsymbol{t}+\delta\boldsymbol{t}_1}, \ldots \hat{\boldsymbol{\varphi}}|_{\boldsymbol{t}+\delta\boldsymbol{t}_{\tilde{K}}}\}) = \tilde{K}$. With this parametrization of the target function, we can formulate the physics-based solution adjustment as an optimization problem with coefficients c_k as governing variables instead of the infinite-dimensional functional space:

$$\min_{c_0, c_1, \ldots, c_{\tilde{K}}} \|\boldsymbol{\mathcal{R}}^n\|_2 + \tilde{w}\|[\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m]\|_2 \quad \text{s.t. } c_0 + \sum_k^{\tilde{K}} c_k = 1 \tag{14}$$

where $\tilde{w}$ is a tunable factor that controls the relative importance of the director momentum residual in the minimization progress with respect to the linear momentum residual. Instead of adopting $\boldsymbol{\mathcal{R}}^m$ in the optimization objective, we use a 2D vector $[\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m](\beta = 1, 2)$ for the consideration of director momentum equation, which will be explained in the rest of this section.

We next focus on deriving the balance laws listed in Table 1 and implementing them based on the locally approximated manifold expressions. We notice that the variables $\tilde{m}^{3\alpha}, \bar{\lambda}$ cannot be derived from either kinematic relationships or constitutive relationships. In this sense, we aim to clear out the terms with $\tilde{m}^{3\alpha}, \bar{\lambda}$ in the expanded

Algorithm 1 Improve model predictions with physical constraints around a neighborhood of one finite element node $\bar{v}_0$

1: Prescribe $\delta \boldsymbol{p}_1, \cdots, \delta \boldsymbol{p}_{\tilde{K}}$ and predict the deformed shell configurations at the five different loading conditions $\boldsymbol{p}$, $\boldsymbol{p} + \delta \boldsymbol{p}_1, \cdots, \boldsymbol{p} + \delta \boldsymbol{p}_{\tilde{K}}$ with Eq. (10). $\tilde{K}$ is generally selected according to the dimension of the loading parametric space.
2: Find a set of neighborhood nodes $\bar{v}_0$, and extract the local $\hat{\boldsymbol{\varphi}}$ and $\hat{\boldsymbol{t}}$ field based on the neighborhood node set for the $\tilde{K}+1$ loading conditions in step 1.
3: Train five different MLP functions to approximate the perturbed local solutions. Notice that fitting five components are sufficient: φ_1, φ_2, φ_3, t_1, t_2, since we can determine t_3 by $t_3 = \pm\sqrt{1 - t_1^2 - t_2^2}$ where the sign is up to the orientation of the shell configuration.
4: Establish a nonlinear optimizer to solve the problem described by Eq. (14).

expression of $\boldsymbol{\mathcal{R}}^m$. We accomplish it by taking the dot product of $\boldsymbol{\mathcal{R}}^m$ with respect to $\boldsymbol{t}_{,\beta}$, where we first expand $\boldsymbol{\mathcal{R}}^m$ as follows:

$$\boldsymbol{\mathcal{R}}^m = (\tilde{m}_{\mu\alpha,\alpha}\boldsymbol{a}_\mu + \tilde{m}_{3\alpha,\alpha}\boldsymbol{t} + \tilde{m}_{\mu\alpha}\boldsymbol{a}_{\mu,\alpha} + \tilde{m}_{3\alpha}\boldsymbol{t}_{,\alpha}) + \frac{J_{,\alpha}}{J}(\tilde{m}_{\mu\alpha}\boldsymbol{a}_\mu + \tilde{m}_{3\alpha}\boldsymbol{t}) - \bar{\lambda}\boldsymbol{t} - (\tilde{q}_\alpha + \lambda^\alpha_\mu \tilde{m}_{3\mu})\boldsymbol{a}_\alpha \tag{15}$$

We then compute $\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m$ and simplify it with $\boldsymbol{t}_{,\beta} \cdot \boldsymbol{t} = 0$ and $\boldsymbol{t}_{,\beta} \cdot \boldsymbol{a}_\alpha = \kappa_{\alpha\beta}$:

$$\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m = (\tilde{m}_{\mu\alpha,\alpha}\kappa_{\mu\beta} + \tilde{m}_{\alpha\mu}\boldsymbol{t}_{,\beta} \cdot \boldsymbol{a}_{\mu,\alpha} + \tilde{m}_{3\alpha}\boldsymbol{t}_{,\beta} \cdot \boldsymbol{t}_{,\alpha}) + \frac{J_{,\alpha}}{J}\tilde{m}_{\mu\alpha}\kappa_{\mu\beta} - (\tilde{q}_\alpha + \lambda^\alpha_\mu \tilde{m}_{3\mu})\kappa_{\alpha\beta} \tag{16}$$

The frame vector decomposition expression in Table 1 shows us $\boldsymbol{t}_{,\alpha} \cdot \boldsymbol{t}_{,\beta} = \lambda^\mu_\alpha \boldsymbol{a}_\mu \cdot \boldsymbol{t}_{,\beta} + \lambda^3_\alpha \boldsymbol{t} \cdot \boldsymbol{t}_{,\beta} = \lambda^\mu_\alpha \kappa_{\mu\beta}$. This concludes the equivalence between two terms in (16): $\tilde{m}_{3\alpha}\boldsymbol{t}_{,\beta} \cdot \boldsymbol{t}_{,\alpha} = \lambda^\alpha_\mu \tilde{m}_{3\mu}\kappa_{\alpha\beta}$. As a result, the director-gradient-weighted director momentum residual in (16) is finally computed as follows:

$$\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m = \tilde{m}_{\mu\alpha,\alpha}\kappa_{\mu\beta} + \tilde{m}_{\mu\alpha}\boldsymbol{t}_{,\beta} \cdot \boldsymbol{a}_{\mu,\alpha} + \frac{J_{,\alpha}}{J}\tilde{m}_{\mu\alpha}\kappa_{\mu\beta} - \tilde{q}_\alpha\kappa_{\alpha\beta} \tag{17}$$

In contrast to $\boldsymbol{\mathcal{R}}^m$, $\boldsymbol{\mathcal{R}}^n$ possesses an expanded expression consisting of $\tilde{n}_{\alpha\beta}, \tilde{q}_\alpha, \tilde{m}_{\alpha\beta}, \lambda^\beta_\mu, \gamma_\beta, \boldsymbol{a}_\alpha, \boldsymbol{t}$ and some of their gradient or divergence with respect to the parametric coordinates. These variables can be computed via the constitutive/kinematic equations or other geometric constraints listed in Table 1. Hence we no longer perform further derivation on $\boldsymbol{\mathcal{R}}^n$ for numerical implementation purposes. The procedures for computing the residuals of the linear and director momentum balance are then summarized in Algorithm 2:

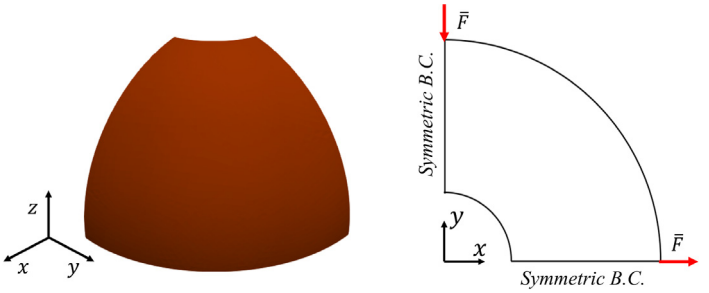
Algorithm 2 Compute the residual around a neighborhood of one finite element node $\bar{v}_0$

Given: $\boldsymbol{\varphi}$ and $\boldsymbol{t}$ around $\bar{v}_0$ in the form of Eq. (13) with coefficients $c_0 \cdots c_4$ determined.

1: Compute the surface frame vectors $\boldsymbol{a}_{0\alpha}, \boldsymbol{a}_\alpha$, and the surface Jacobian J.
2: Compute the $\boldsymbol{t}_{,\alpha}$ with the director field obtained in Step 2.
3: Compute the kinematic variables $a_{\alpha\beta}, \gamma_\alpha, \kappa_{\alpha\beta}$ in both the current and the initial configuration.
4: Compute effective strain measures $\varepsilon_{\alpha\beta}, \delta_\alpha, \rho_{\alpha\beta}$.
5: Find the coefficients λ^μ_α from vector decomposition of $\boldsymbol{t}_{,\alpha}$.
6: Evaluate the effective stress measures $\tilde{n}_{\alpha\beta}, \tilde{q}_\alpha, \tilde{m}_{\alpha\beta}$ with the constitutive equations (ref constitutive eqn)
7: Compute $\boldsymbol{\mathcal{R}}^n = \frac{1}{J}(J\boldsymbol{n}_\alpha)_{,\alpha}$ and $\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m$ with Eq. (17).

4. Training of the geometric learning model

In this section, we present numerical examples that demonstrate the correctness and applicability of our model, and the procedures to generate the dataset with different levels of fidelity via finite element simulation. In Section 4.1, we introduce three shell problems of interest. We first show the shell geometries and the corresponding mesh discretizations with multiple resolutions. Then we demonstrate the simulation setup to demonstrate how data are sampled within load parametric space. We additionally list the hyperparameters assumed at the model training phase in Section 4.2.



(a) shell domain **(b)** boundary conditions (projected to xy plane)

Fig. 4. Computational domain and boundary conditions of the initial configuration of the hemispherical shell holed at the pole.

4.1. Generation of the graph database for deformed shell fields

4.1.1. A hemispherical shell holed at the pole

The shell problem configuration presented in this subsection is in a hemispherical shape but with a circular hole punched at the pole. The edge of the hole coincides with a latitude line at a zenith angle of 18°. According to the symmetry of this configuration, we take one-quarter of the holed hemisphere as the computation domain. The initial shell configuration can be expressed as follows, assuming that the initial director field matches the outward normal vector field of the shell surface:

$$\begin{cases} \varphi_1(\xi,\eta) = \bar{r}\cos(\frac{\pi}{2}\xi)\cos(\frac{2\pi}{5}\eta), \\ \varphi_2(\xi,\eta) = \bar{r}\sin(\frac{\pi}{2}\xi)\cos(\frac{2\pi}{5}\eta), \\ \varphi_3(\xi,\eta) = \bar{r}\sin(\frac{2\pi}{5}\eta). \end{cases} \quad \begin{cases} t_1(\xi,\eta) = \varphi_1(\xi,\eta)/\bar{r}, \\ t_2(\xi,\eta) = \varphi_2(\xi,\eta)/\bar{r}, \\ t_3(\xi,\eta) = \varphi_3(\xi,\eta)/\bar{r}. \end{cases} \quad 0 \le \xi,\eta \le 1 \tag{18}$$

where $\bar{r}$ is the spherical radius chosen to be 10 m in our case.

The domain symmetry requires symmetric boundary conditions on the two lateral bounds located at $\xi = 0$ and $\xi = 1$. The driving loads are prescribed as the concentrated force at two locations: at $\xi = 0, \eta = 0$ it points toward the positive spatial x-axis, while at $\xi = 1, \eta = 0$ it points toward the negative spatial y-axis. The two forces possess the same magnitude, indicating that there is just one parameter $\bar{F}$ controlling the shell dynamics. The material properties are as follows: Young's modulus is 68.25 MPa, and the Poisson ratio is 0.3. The thickness of this structure is assumed to be 0.04 m. We plot the shell configuration and the loading conditions (projected to xy plane) in Fig. 4.

In the following part of this section, we focus on how simulations are performed to collect snapshots of deformed shell configurations. For the sake of dataset enrichment mentioned in Section 2.1, we discretize the configurations of both numerical examples into finite element meshes with different resolutions. In this sense, the low-fidelity dataset ${}^{l}\mathbb{G}$ is established by collecting simulation snapshots in the coarser mesh, while the high-fidelity dataset ${}^{h}\mathbb{G}$ is established with simulation snapshots in the finer mesh. We usually simulate fewer steps (snapshots) in the finer mesh than that in the coarser mesh.

During mesh discretization, we first mesh the parametric domain to obtain a set of node coordinates (in 2D) and a set of element connections. We then perform a coordinate transformation that maps the parametric coordinate to its spatial counterpart on the shell manifold. We finally define the 3D mesh configuration by combining the transformed node coordinates and the same element connections as those in the 2D parametric mesh. Fig. 5 shows the finite element meshes in different resolutions for the first example: a hemispherical shell holed at the pole. Fig. 5 (a,c) demonstrates the 3D shell mesh configuration from some perspective, while Fig. 5 (b,d) shows the mesh discretization in the 2D parametric space before coordinate transformation. The coarser mesh (Fig. 5 (a,b)) has 125 nodes and 107 elements, while the finer mesh (Fig. 5 (c,d)) has 1781 nodes and 1712 elements.

We simulate 200 uniform-increment steps with the coarser mesh to build ${}^{l}\mathbb{G}$, while the loading force value $\bar{F}$ grows linearly to 1. To generate ${}^{h}\mathbb{G}$ with the finer mesh (Fig. 5(c)), we maintain the $\bar{F}$ value in the last step as the same ultimate load magnitude in the simulation setup with the coarser mesh (equal to 1) and load it for 20

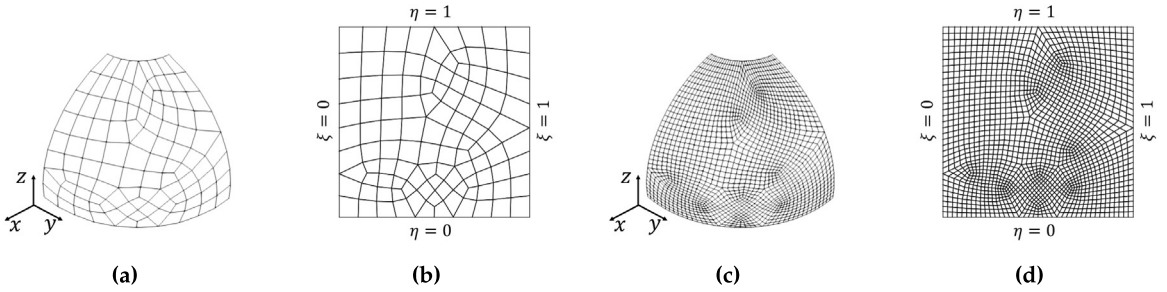


Fig. 5. Mesh discretization of the holed hemispherical shell. (a), (b) are the finite element mesh with **coarse** resolution viewed in the actual 3D space and the parametric space, respectively. (c), (d) are the finite element mesh with **fine** resolution viewed in the actual 3D space and the parametric space, respectively.

uniform-increment steps. In this sense, for the 10th, 20th, ..., 200th snapshots in ${}^{l}\mathbb{G}$ if counted sequentially in time, there is a snapshot in ${}^{h}\mathbb{G}$ corresponding to the same prescribed load $\bar{F}$. ${}^{lh}\mathbb{G}$ is then established by extracting the last snapshot of every ten steps ${}^{l}\mathbb{G}$ sorted by time. We also simulate additional steps with the finer mesh to build a dataset for testing prediction accuracy, where we prescribe the external force as 0.6275 and 0.9425 such that there are no snapshots in the training dataset that correspond to these loads.

4.1.2. A doubly-curved shell anchored at four corners

The second problem simulates a doubly-curved shell structure anchored at four corners for shelter purposes. The geometry of this shell configuration can be generalized with the following equation: $z-\omega(x, y) = 0$, where $\omega(x, y)$ is expressed as follows (unit is meter for x, y, z):

$$\omega(x, y) = 2.4(1-\omega_1(x, y)^2) + \omega_3(x, y)^2\left[3(1-\omega_2(x, y)^2) - 2.4(1-\omega_1(x, y)^2)\right], \quad -1 \le x, y \le 1 \tag{19}$$

where

$$\omega_1(x, y) = \frac{|x|+|y|}{2}, \quad \omega_2(x, y) = \frac{|x+y|-|x-y|}{2}, \quad \omega_3(x, y) = \frac{|x+y|+|x-y|}{2} \tag{20}$$

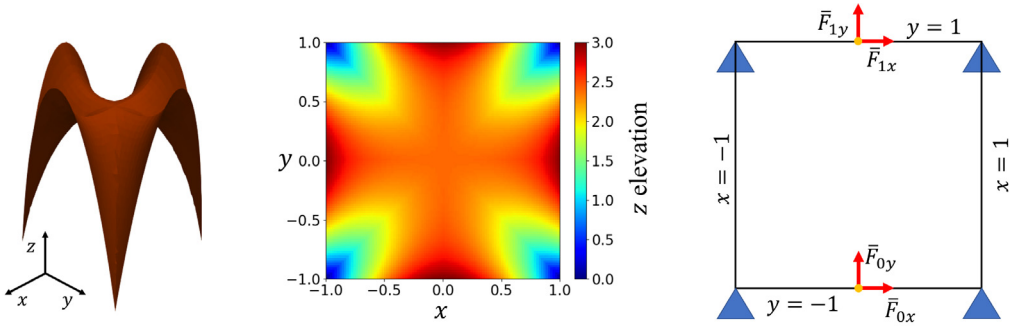
We present the spatial view of the initial configuration from one perspective and plot the z elevation as a function of x, y in Fig. 6. We still assume that the initial director field corresponds to the outward normal vector field of the shell surface. As this shell is anchored at four corners, we enforce fix boundary conditions at the four corners. The driven force is concentrated and located at $(x, y, z) = (0, -1, 3)$ and $(x, y, z) = (0, 1, 3)$, where the forces have only x and y components. In total, the external load can be represented with four parameters. The material properties are as follows: Young's modulus is 29 MPa, and the Poisson ratio is 0.3. The thickness of this structure is assumed to be 0.04 m.

Fig. 7 presents the finite element meshes in different resolutions for the second example: a doubly curved shell anchored at four corners. The coarser mesh (Fig. 7 (a,b)) has 397 nodes and 356 elements, while the finer mesh (Fig. 7 (c,d)) has 1521 nodes and 1424 elements. To prescribe the load, we randomly sample 400 ultimate load values and prescribe a monotonic straight-line loading path to each ultimate load point. On each loading path, we simulate 20 steps. Fig. 8 demonstrates how the 400 loading paths populate the load parameter space:

To construct ${}^{l}\mathbb{G}$ for this case, we simulate all steps on each loading path with the coarser mesh (Fig. 7 (a,b)), which yields in total 8,000 snapshots. When establishing ${}^{h}\mathbb{G}$, we randomly sample 40 load paths within the paths provided in Fig. and simulate them with the finer mesh (Fig. 7 (c,d)), which yields in total 800 snapshots. For testing purposes, we prescribed three test loading paths: one is seen in ${}^{l}\mathbb{G}\backslash{}^{lh}\mathbb{G}$ while the other two is unseen in the training set as Fig. 9 shows.

4.1.3. A cylindrical patch clamped at one boundary

As previous examples create the parametric loading space by applying concentrated point forces, we would like to establish a example demonstrating the response surface modeling for shells under distributed loading. So we propose the third problem as cylindrical patch clamped at one boundary, and loaded at the opposite boundary with

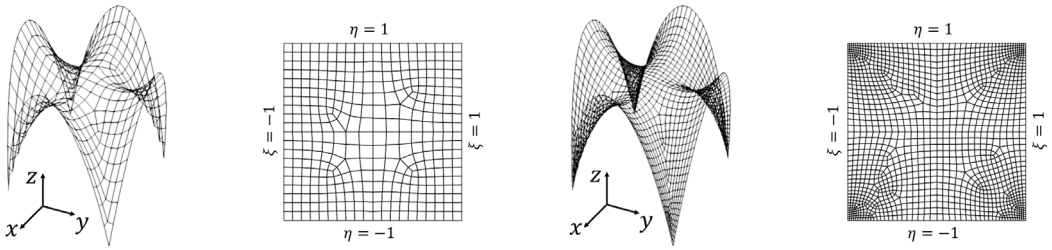


(a) shell configuration in one perspective

(b) z-direction elevation plot in xy plane

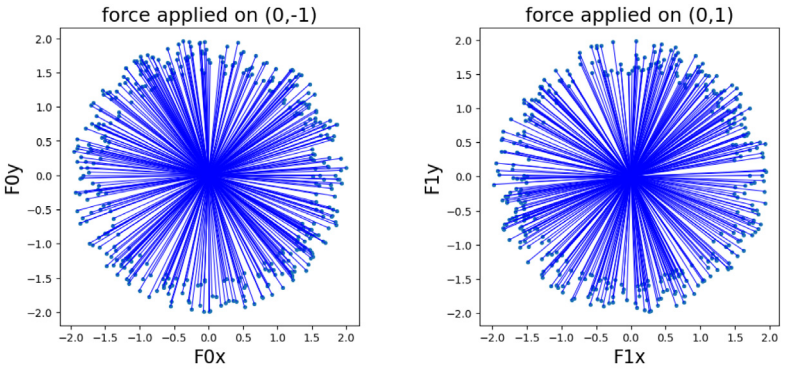
(c) boundary conditions (projected to xy plane)

Fig. 6. Geometry of the initial configuration of the doubly-curved shell anchored at four corners together with the boundary conditions. x, y, z have the same length unit (m). Triangulars in (c) indicate fix boundary conditions (in all directions) at a specific point.



(a) **(b)** **(c)** **(d)**

Fig. 7. Mesh discretization of the doubly curved shell anchored at four corners. (a), (b) are the finite element mesh with **coarse** resolution viewed in the actual 3D space and the parametric space, respectively. (c), (d) are the finite element mesh with **fine** resolution viewed in the actual 3D space and the parametric space, respectively.



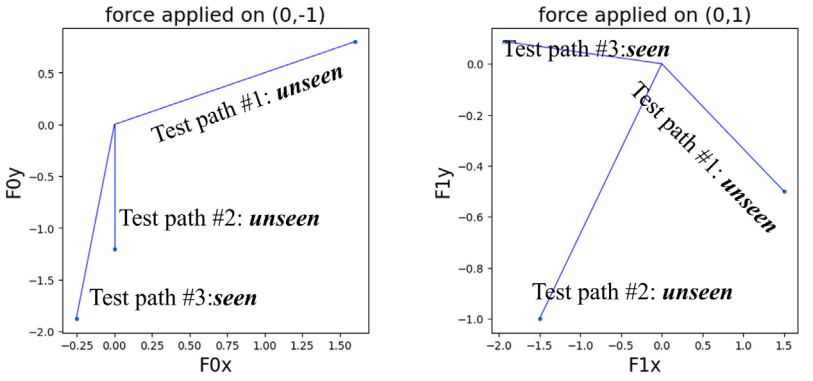
(a) Loading paths projected to $\bar{F}_{0x}\bar{F}_{0y}$ plane.

(b) Loading paths projected to $\bar{F}_{1x}\bar{F}_{1y}$ plane.

Fig. 8. Prescribed loading paths to generate the training dataset for the doubly-curved shell example. All 400 paths are projected to either $\bar{F}_{0x}\bar{F}_{0y}$ plane or $\bar{F}_{1x}\bar{F}_{1y}$ plane as straight lines, and the ultimate load at the end of each path is marked by a circle. (as Fig. 6(c) shows, $\bar{F}_{0x}, \bar{F}_{0y}$ is the force applied at $(0, -1)$ and $\bar{F}_{1x}, \bar{F}_{1y}$ is the force applied at $(0, 1)$) The overbar is omitted in the axes label texts for simplicity.

uniform line loads. The geometry of this shell configuration is shown in Fig. 10, and its parametric equation is written as follows:

$$x = 2\xi, \quad y = 1.5\eta, \quad z = \sqrt{4^2 - x^2} \quad \text{for } 0 \le \xi, \eta \le 1 \tag{21}$$



(a) Test loading paths projected to $\bar{F}_{0x}\bar{F}_{0y}$ plane.

(b) Test loading paths projected to $\bar{F}_{1x}\bar{F}_{1y}$ plane.

Fig. 9. Loading paths for modeling testing for the doubly-curved shell an example. The overbar is omitted in the axes label texts for simplicity.

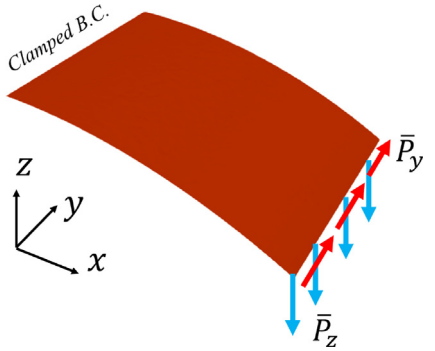


Fig. 10. Geometry of the initial configuration of the cylindrical patch clamped at the left boundary ($\xi = 0$). x, y, z have the same length unit (m). On the right boundary ($\xi = 1$), it is loaded by two uniformly distributed line forces: one pointing toward the positive y direction at a magnitude of $\bar{P}_y$, the other pointing toward the negative z direction at a magnitude of $\bar{P}_z$.

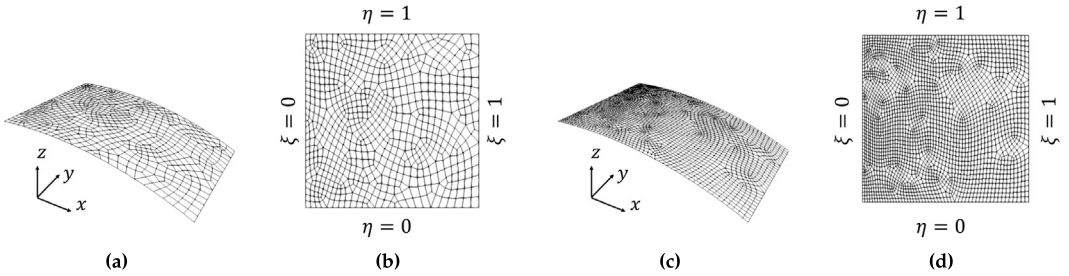


Fig. 11. Mesh discretization of the cylindrical patch clamp at one boundary. (a), (b) are the finite element mesh with **coarse** resolution viewed in the actual 3D space and the parametric space, respectively. (c), (d) are the finite element mesh with **fine** resolution viewed in the actual 3D space and the parametric space, respectively.

The material properties are as follows: Young's modulus is 28 MPa, and the Poisson ratio is 0.3. The thickness of this structure is assumed to be 0.04 m. Fig. 11 presents the finite element meshes in different resolutions for the second example: a doubly curved shell anchored at four corners. The coarser mesh (Fig. 11 (a,b)) has 397 nodes and 356 elements, while the finer mesh (Fig. 11 (c,d)) has 1521 nodes and 1424 elements.

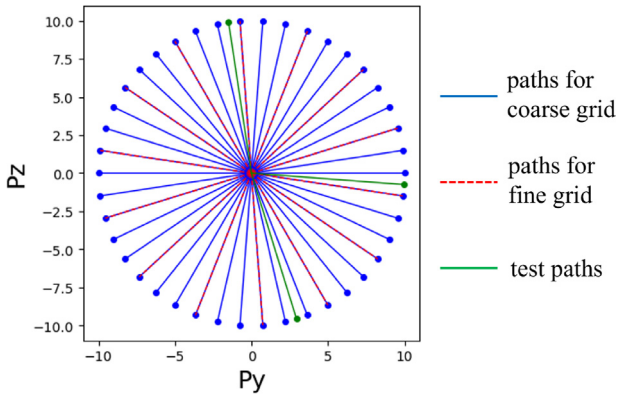


Fig. 12. Prescribed loading paths to generate training dataset for the clamped cylindrical patch example. Paths are shown as straight lines in the $\bar{P}_y \bar{P}_z$ space, and the ultimate load at the end of each path is marked by a circle.

Table 2
Simulation time of the three examples at different resolutions when generating the training dataset. Neural network evaluation takes almost the same time for all problems: 0.15 s, so it is not listed as items in Table 2. This is also true for physically constrained optimization, where it usually costs 28 s to get the corrected local solutions. Computational times are evaluated on an NVIDIA DGX A100 workstation.

	Resolution	Steps per path	Avg. time per path	Total sim. time
Hemispherical shell	coarse	200	23 s	23 s
	fine	20	59 s	59 s
Doubly curved shell	coarse	20	11 s	4400 s
	fine	20	68 s	2720 s
Clamped shell	coarse	18	19 s	798 s
	fine	6	47 s	658 s

The external load can be fully represented with two parameters ($\bar{P}_y$, $\bar{P}_z$). To populate the load parameter space, 42 load paths are prescribed radially such that the entire range of polar angle is equally divided into 42 segments. Each monotonic straight-line loading path is simulated to some ultimate load point for 18 steps. We select one-third of these paths to generate the high-fidelity simulation, but the steps on each path are limited to 6 (to control the ratio between the low-fidelity data and the high-fidelity data). Three paths not existing in the training dataset are generated randomly for test purposes. All of the prescribed paths are shown in Fig. 12:

Table 2 compares the performance of the machine learning predictions again the benchmark finite element simulations. Note that a key advantage of the neural network predictions against the nonlinear finite element simulations is that the neural network predictions does not require multiple loading step to reach the target load whereas a conventional nonlinear finite element simulations may requires multiple incremental load steps to remain close to the equilibrium state. As such, our approach, which requires 0.15 s neural network predictions + 28 s for equilibrium correction, can be an attractive option when compared with the total simulation time of the FEM (cf. Kim and James [60]).

4.2. Training configuration of the geometric models

This section aims to provide the detailed setting we employed for training the geometric learning model of the numerical examples. The set of hyperparameters is provided to ensure reproducibility of the results. In each example, we introduce the training setup for the mapping $\hat{\boldsymbol{F}}$, the graph autoencoder $\hat{\boldsymbol{R}}$, and the shell response controlling law $\hat{\boldsymbol{f}}$ sequentially. Training setups usually cover the dimensions of neural network layers or substructures, the choice of optimizer, the training iterations, and the learning rate. The neural networking is trained using the PyTorch open-source library [61]. The training configuration for all examples is summarized in Tables 3–5. The success of this training is evidenced in the loss vs. epochs shown in Fig. 13 where the loss histories against the training and testing dataset of the doubly-curved shell example are provided.

Table 3
Training setup of the **first** example. The Adam optimizer [62] is adopted for all tasks.

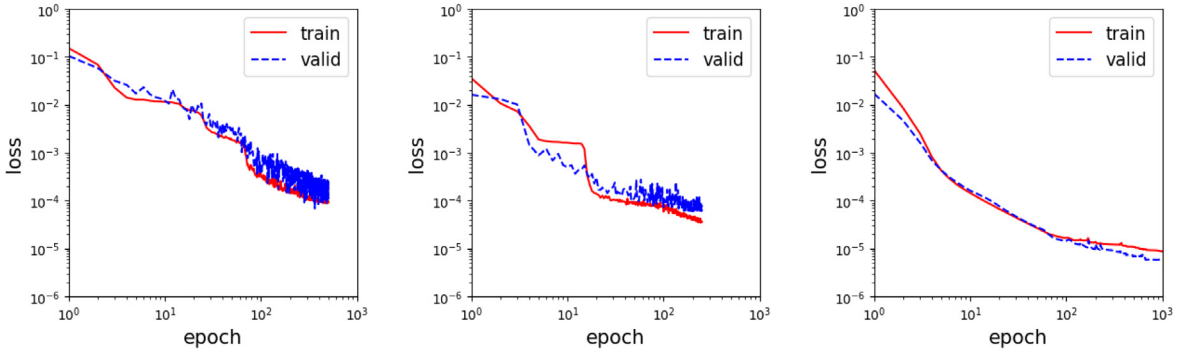
$\hat{\boldsymbol{F}}$ training setup	features of the GIN layers	encoded dimensions	8
		latent channels	64
	MLP substructure of GIN layers	num. of layers	3
		hidden neurons	100
	features of the optimizer	batch size	1
		training iterations	400
		learning rate	5×10^{-4}
$\hat{\boldsymbol{R}}$ training setup	all setups are the same as those while training $\hat{\boldsymbol{F}}$ except that the batch size is 16		
$\hat{f}$ training setup	MLP architecture	num. of layers	3
		hidden neurons	100
	features of the optimizer	batch size	16
		training iterations	500
		learning rate	5×10^{-4}

Table 4
Training setup of the **second** example. The Adam optimizer is adopted for all tasks.

$\hat{\boldsymbol{F}}$ training setup	features of the GIN layers	encoded dimensions	16
		latent channels	64
	MLP substructure of GIN layers	same as Table 3	
	features of the optimizer	batch size	32
		training iterations	500
		learning rate	2.5×10^{-4}
$\hat{\boldsymbol{R}}$ training setup	features of the GIN layers	same as those in $\hat{\boldsymbol{F}}$ training setup	
	MLP substructure of GIN layers	same as those in $\hat{\boldsymbol{F}}$ training setup	
	features of the optimizer	same as those in $\hat{\boldsymbol{F}}$ training setup except that the training iteration is 250	
$\hat{f}$ training setup	MLP architecture	same as Table 3	
	features of the optimizer	batch size	128
		training iterations	1000
		learning rate	5×10^{-4}

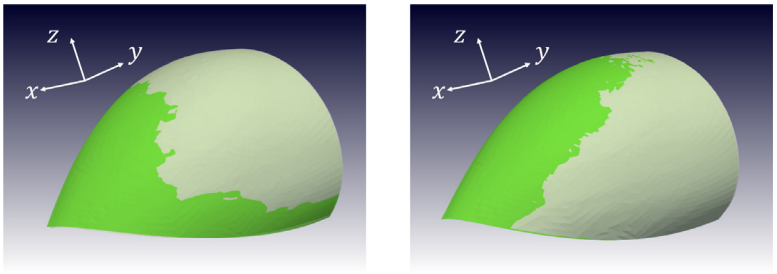
Table 5
Training setup of the **third** example. The Adam optimizer is adopted for all tasks.

$\hat{\boldsymbol{F}}$ training setup	features of the GIN layers	same as Table 4	
	MLP substructure of GIN layers	same as Table 4	
	features of the optimizer	batch size	16
		training iterations	750
		learning rate	2.5×10^{-4}
$\hat{\boldsymbol{R}}$ training setup	features of the GIN layers	same as those in $\hat{\boldsymbol{F}}$ training setup	
	MLP substructure of GIN layers	same as those in $\hat{\boldsymbol{F}}$ training setup	
	features of the optimizer	same as those in $\hat{\boldsymbol{F}}$ training setup except that the training iteration is 250	
$\hat{f}$ training setup	MLP architecture	same as Table 4	
	features of the optimizer	batch size	64
		training iterations	1200
		learning rate	5×10^{-4}



(a) $\hat{\boldsymbol{F}}$ training history (b) $\hat{\boldsymbol{R}}$ training history (c) $\hat{\boldsymbol{f}}$ training history

Fig. 13. Example training and validation loss function values for different neural network training tasks (for the doubly-curved shell example).



(a) Configuration at $\bar{F} = 0.6275$ (b) Configuration at $\bar{F} = 0.9425$

Fig. 14. Predicted deformed shell configurations at two test snapshots with prescribed loads not included in the training samples. The green surface indicates the reference configuration (from DNS) while the gray surface indicates the configuration predicted from our model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

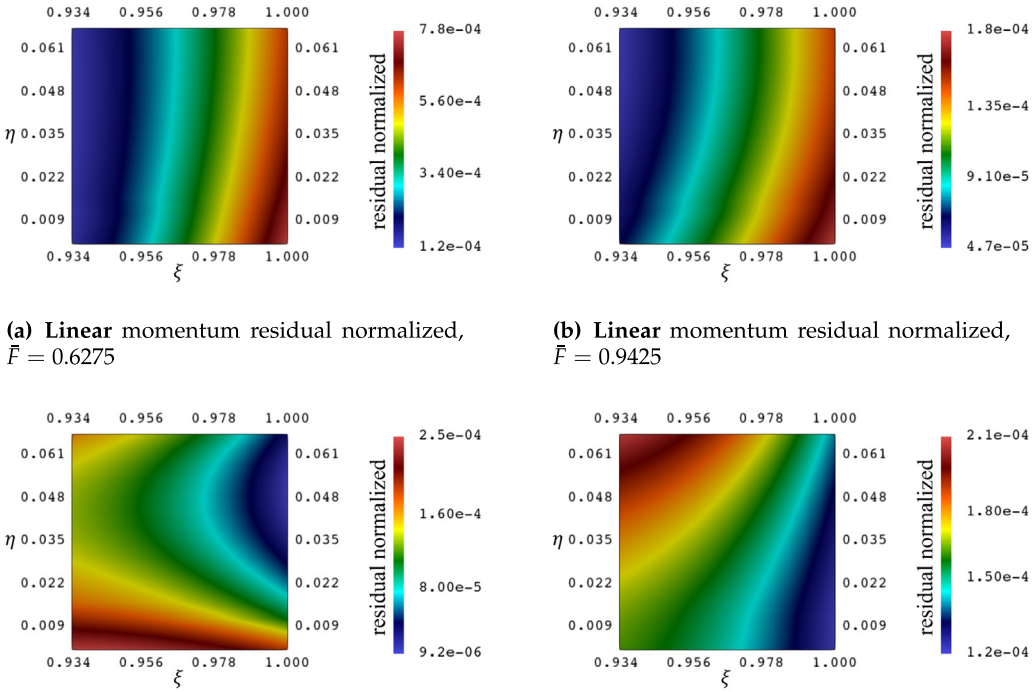
5. Numerical results and discussion

This section follows the demonstration in Section 4 and presents modeling results of the two examples of interest. Section 5.1 shows verification results on the holed spherical shell to justify the usage of curvature filters and the correctness of the resulting prediction. Sections 5.2 and 5.3 shows miscellaneous results that demonstrate the capacity of our model in reconstructing a realistic response surface in higher-dimensional parametric space.

5.1. Spherical shell verification

The first example mainly serves as a verification example showing that the response surface can be successfully reconstructed in a lower dimensional space. As the parametric loading space is only one-dimensional, we select two loading states whose simulated shell configuration is not included in the training dataset: $\bar{F} = 0.6275, 0.9425$. We plot the predicted shell manifold against the configuration obtained from direct numerical simulations (DNS) in Fig. 14, where we can easily observe the predicted configuration is very close to the reference one (from DNS). Physical residual equation values are also evaluated within a neighborhood of one of the loading points ($(\xi, \eta) = (1, 0)$), as presented in Fig. 15, which suggests that the predicted shell configuration is in fact well compatible with the physical constraints equation within the local area around where the concentrated load applied. Notice that the value of the residuals are normalized to a dimensionless state by: $|\boldsymbol{\mathcal{R}}^n|_2 \times \Delta h/\tilde{n}_m$ and $|[\boldsymbol{t}_{,\beta} \cdot \boldsymbol{\mathcal{R}}^m]|_2 \times \Delta h/\tilde{n}_m$, where Δh is the mesh size and $\tilde{n}_m$ is the maximum principal component of the effective membrane stress tensor $\tilde{n}_{\beta\alpha}$ at the loaded point of interest.

We here want to highlight the capability of our model for capturing the nonlinearity in the response surface in the parametric space. To this end, we propose to compare the prediction with our model and the linear interpolation



(a) Linear momentum residual normalized, $\bar{F} = 0.6275$

(b) Linear momentum residual normalized, $\bar{F} = 0.9425$

(c) Weighted **director** momentum residual normalized, $\bar{F} = 0.6275$

(d) Weighted **director** momentum residual normalized, $\bar{F} = 0.9425$

Fig. 15. Normalized residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (1, 0)$, where concentrated force is applied.

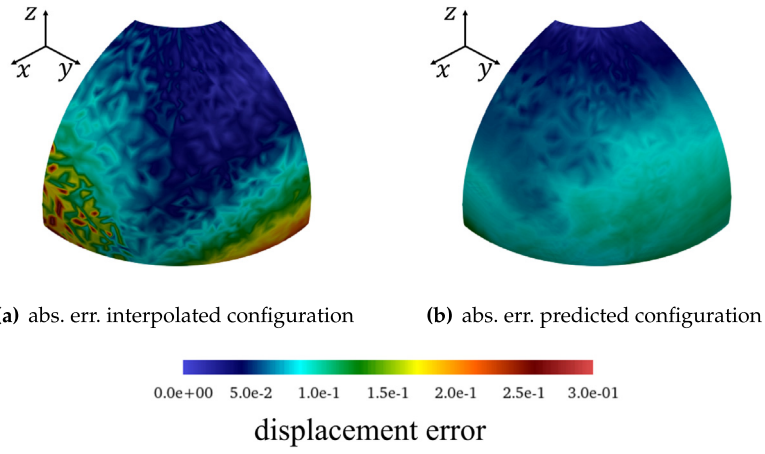
results, with a focus on the loading case where $\bar{F} = 0.9425$. We fetch two loading conditions that formulate the closest possible upper and lower bound for the targeted case ($\bar{F} = 0.9425$), which could be determined as: $\bar{F} = 0.945, 0.94$. We extract deformed configurations corresponding to the two fetched conditions from the database, and generate the interpolated configuration by computing the arithmetic average of the displacement across the entire domain, as the load targeted load is the arithmetic average of the fetched loads. For visualization purposes, we plot the displacement prediction the absolute error of both the interpolated configuration and the predicted configuration onto the initial shell configuration in Fig. 16.

5.2. A doubly-curved shell structure

This section provides numerical results regarding the shell configuration prediction and the responses in a local area near the concentrated force after the resulting adjustment with physical constraints for the doubly-curved shell example introduced in 4.1.2. We first present the deformed shell configuration for the three testing paths we prescribed in 4.1, where each configuration is extracted after simulated to the end of the loading path. As Fig. 17 shows, the predicted shell configurations mostly overlap with the reference shell configurations, which are from the high-fidelity numerical simulation results on the fine mesh shown in Fig. 7(c).

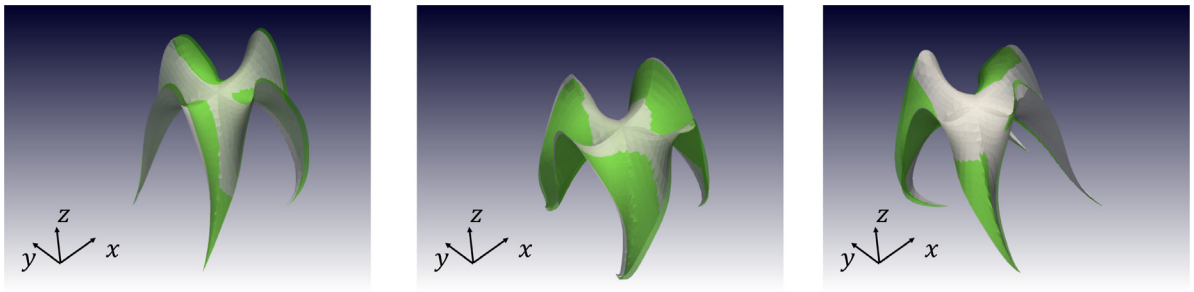
Again, we here want to show the privilege of our model w.r.t. the linear interpolation method in higher dimensional parametric space. The test configuration is generated with a load of $\boldsymbol{p} = (-0.94, -0.14, 0.03, 0.77)$. The resulting absolute error plot of the initial configuration is provided in Fig. 20. We find that the general magnitude and the RMSE response of the displacement absolute error of our model prediction are much lower than their interpolated counterparts, unlike the responses in Section 5.1. This further demonstrates the capability of recovering the significant nonlinear behavior of the response surface in higher dimensional cases (see Fig. 18).

We compare the performance of the proposed model prediction against the method of snapshots (see Fig. 20). Instead of classical POD with the method of snapshots, we first perform a singular value decomposition to extract



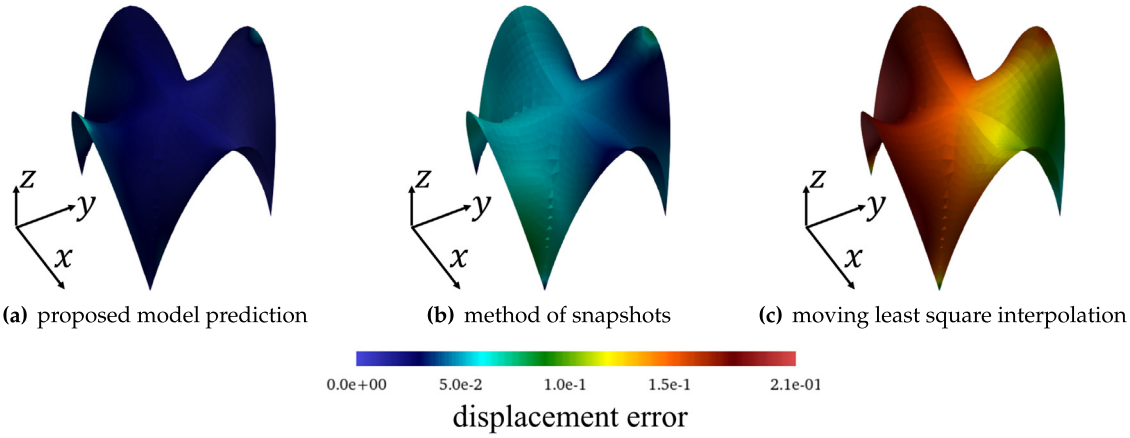
(a) abs. err. interpolated configuration **(b)** abs. err. predicted configuration

Fig. 16. Absolute error of the displacement plotted on the initial shell configuration, with $\bar{F} = 0.9425$. (a) absolute error of the **interpolated** configuration; (a) absolute error of the **predicted** configuration. In general, the prediction of our model outperforms the result obtained by the linear interpolation method. Displacement absolute error unit: m.



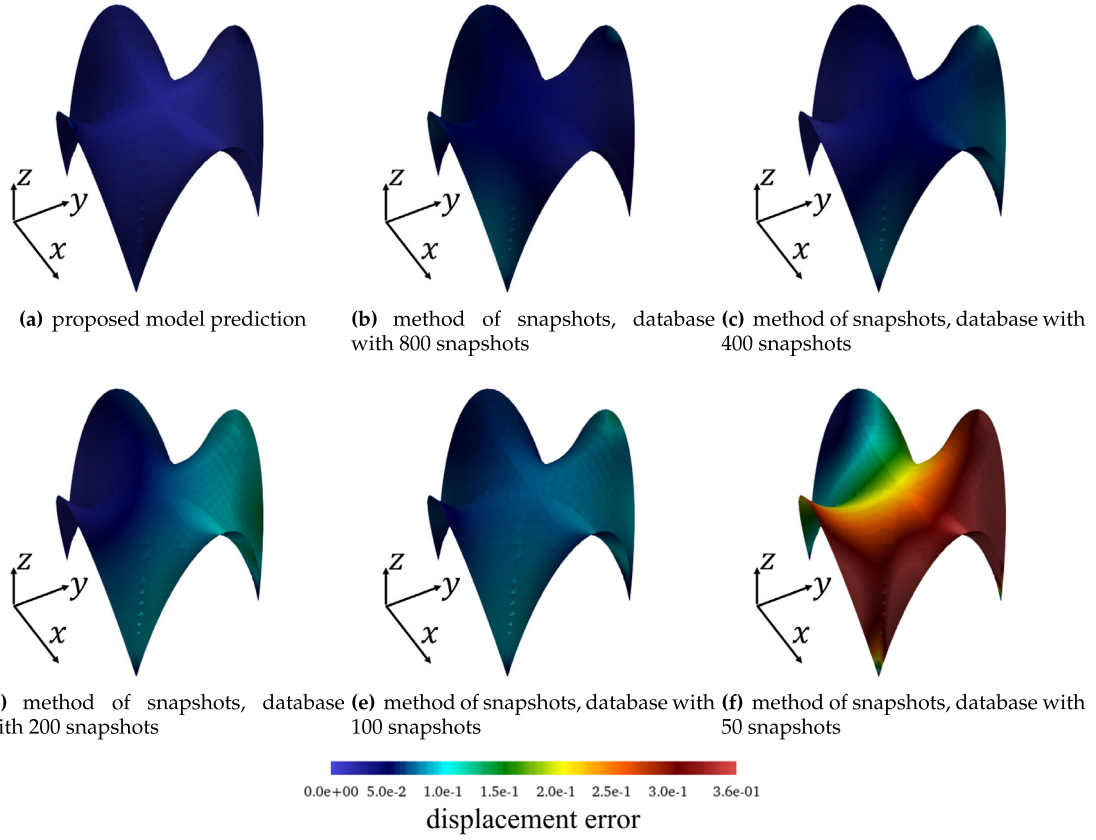
(a) test path # 1 **(b)** test path # 2 **(c)** test path # 3

Fig. 17. Predicted deformed shell configurations (doubly-curved) at the last step simulation results of the three prescribed test paths demonstrated in Section 4.1. Displacements are scaled up by 7.5 times. The green surface indicates the ground-truth configuration while the gray surface indicates the configuration predicted from our model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) proposed model prediction **(b)** method of snapshots **(c)** moving least square interpolation

Fig. 18. Comparison of the absolute error of the displacement field for the test case with load $\boldsymbol{p} = (-0.94, -0.14, 0.03, 0.77)$ between different prediction approaches. Displacement absolute error unit: m.



(a) proposed model prediction **(b)** method of snapshots, database with 800 snapshots **(c)** method of snapshots, database with 400 snapshots

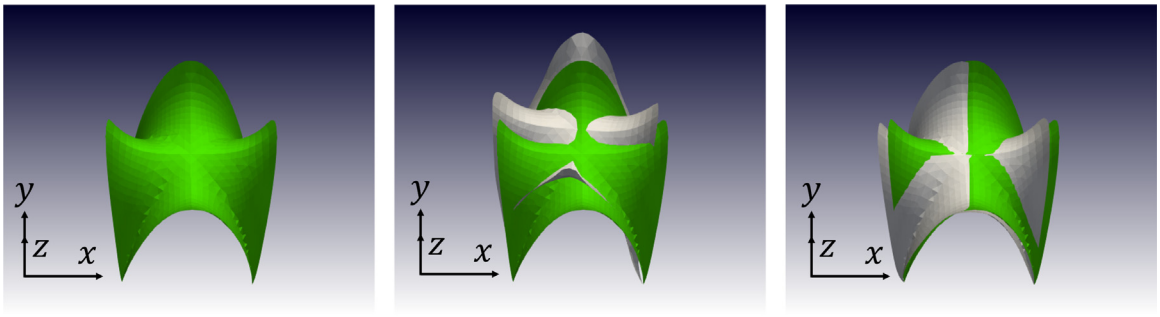
(d) method of snapshots, database with 200 snapshots **(e)** method of snapshots, database with 100 snapshots **(f)** method of snapshots, database with 50 snapshots

Fig. 19. Absolute error of the displacement field for the test case with load $\boldsymbol{p} = (-0.94, -0.14, 0.03, 0.77)$, obtained by the method of snapshots with databases with different sizes. Displacement absolute error unit: m.

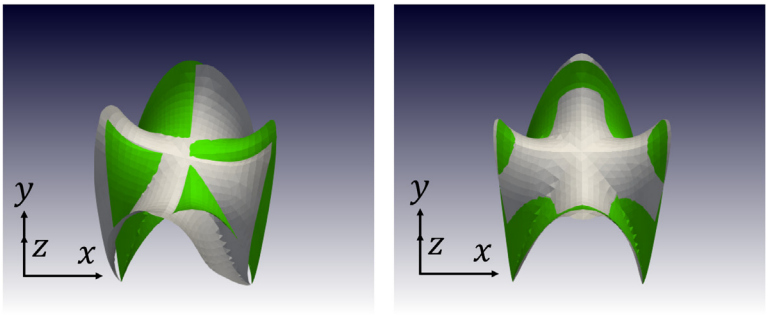
an orthogonal basis solution representation (with 10 orthogonal solutions), and then fit the coefficients associated with each basis with the reference solution. (b)∼(f) shows the results obtained from the method of snapshots using databases with different numbers of snapshots, where initially in (b) we use all 800 snapshots in ${}^{h}\mathbb{G}$, and then we gradually reduce the number of snapshots in the database by random selection ((c)∼(f)), so that we can get a sense of when the method snapshot is no longer reliable (see Fig. 19).

Next, we would like to show the progress with solution adjustment with physical constraints. Fig. 21 shows one example set of perturbed modes as the basis for adjusting the shell configuration locally around the loaded points $(\xi, \eta) = (0, -1)$ at the end of test path # 1. These perturbed fields are generated by random prescribe perturbation $\delta \boldsymbol{p}_i$ $(i = 1, 2, 3, 4)$ in the parametric space, while the linear independency is verified for the perturbed fields. Adjusted local shell configurations for three different loading paths are shown in Fig. 22. We also provide the computed linear momentum residual and director momentum residual fields within the neighborhood of the two loaded points for the three different loading cases in Figs. 23–28.

We further provide additional figures that demonstrate the learned response surface from a different perspective. Fig. 29 presents the load response curve at loading points $(\xi, \eta) = (0, -1)$ and $(0, 1)$ for different loading paths, where the magnitude of horizontal deflection (the displacement vector without z component) is plotted as a function of the magnitude of concentrated load applied at corresponding points. In most cases, the deflection prediction is quite accurate, except that a small discrepancy is observed toward the end of the loading paths. This discrepancy could be attributed to the sparsity of the training data near the edge of the sampling domain that negatively impacts the neural network performance. Furthermore, the prediction accuracy also highly depends on the expressive power of the graph neural network, which can be further improved. In fact, there remain open questions on how to distinguish basic graph structures, such as differences in graph cycles, a task yet to be tackled by even the most

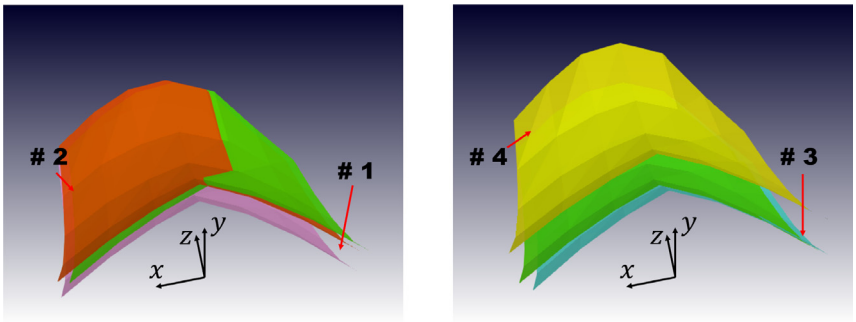


(a) 1st solution **(b)** 2nd solution **(c)** 3rd solution



(d) 4th solution **(e)** 5th solution

Fig. 20. The first five shell configurations of the orthogonal solution basis obtained from the method of snapshots. (a) is the undeformed shell configuration (the green surface), which also serves as the 1st solution in the orthogonal basis. In (b)∼(e), we plot the solutions as deformed shell configurations (the gray surface) against the undeformed shell configuration. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

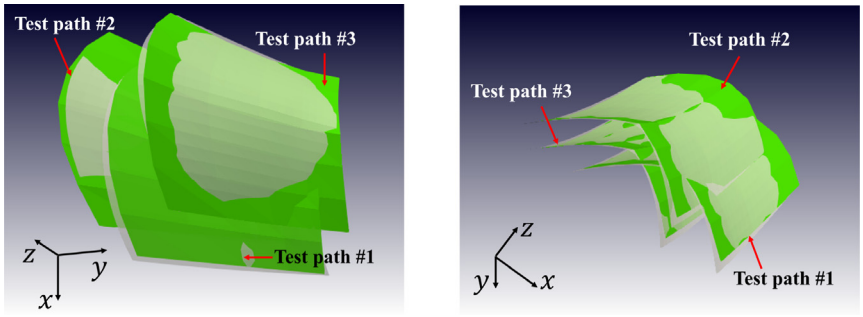


(a) basis # 1 and # 2 **(b)** basis # 3 and # 4

Fig. 21. Examples of the perturbed functional basis for optimizing the predicted shell configuration locally with physical constraints, for the case of test path # 1 around $(\xi, \eta) = (0, -1)$. The green surface indicates the non-perturbed reference local patch simulated shell configuration. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

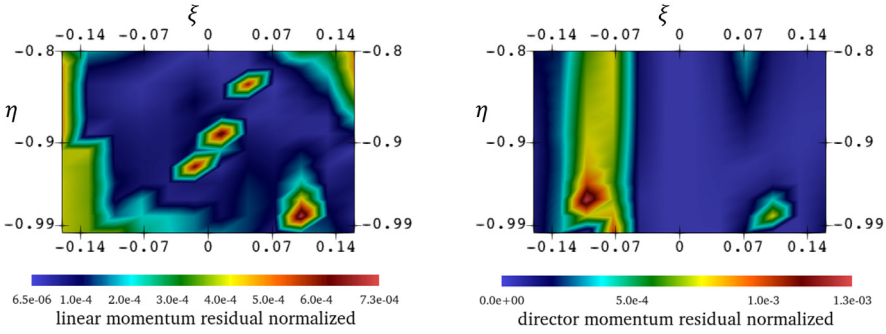
state-of-the-art graph neural network [63]. Further improvement in the graph autoencoder design will be considered in the future but is out of the scope of this study.

Next, we provide a global view of the response surface in the parametric space. It is plausible to assume that the ground truth response surface of our physical problem is a smooth nonlinear manifold, both in the non-reduced space and reduced latent space $\mathcal{L}$. As a result, we propose to check whether the reconstructed solution manifold is smooth and relatively close to the reference solution manifold from the simulation. To this end, we utilize the encoded feature vector in the reduced ordered latent space that is learned by (8). Fig. 30 shows L2-norm of the difference (vector) between the embedded EFV and the predicted EFV. The resultant field is a scalar function of



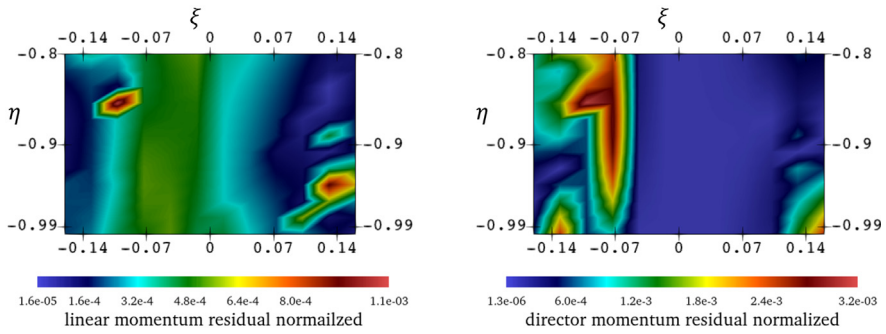
(a) configurations aound $(\xi, \eta) = (0, -1)$ **(b)** configurations aound $(\xi, \eta) = (0, 1)$

Fig. 22. Resulting local patch on the shell configuration after correction based on physical constraints for specific points of interest (e.g., near concentrated load). The green surface indicates the reference local patch simulated shell configuration; the gray surface indicates the corrected local patch. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) linear momentum residual **(b)** director momentum residual

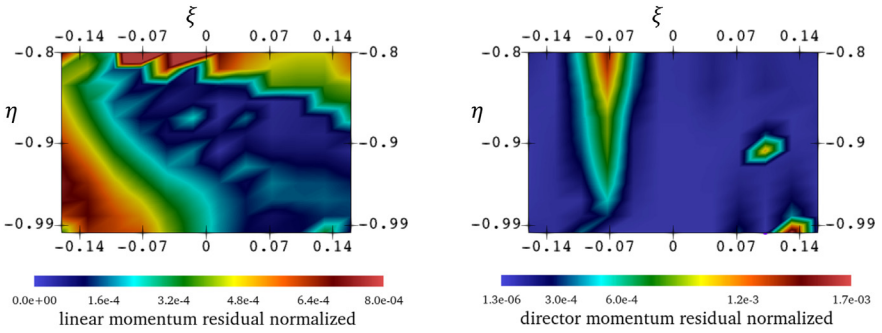
Fig. 23. Normalized momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (0, -1)$ at the last step simulation results of test paths # 1.



(a) linear momentum residual **(b)** director momentum residual

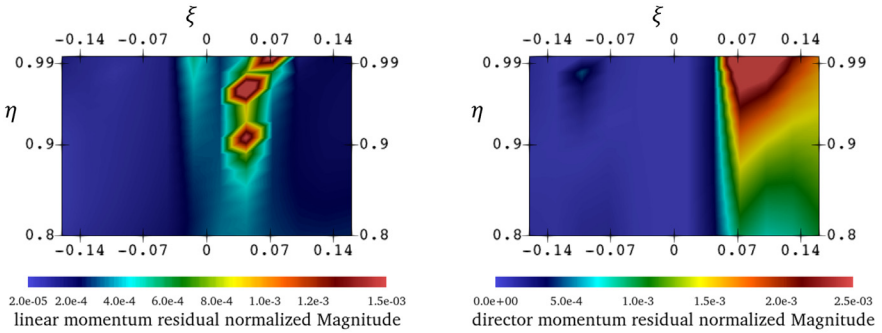
Fig. 24. Normalized momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (0, -1)$ at the last step simulation results of test paths # 2.

the parametric load (4D Euclidean space), and we take cross-section views on three representative planes to verify the continuity of the learned manifold of the response surface in the reduced ordered space. As we can observe, the difference between the EFV from simulated solutions and the predicted solutions is smoothly distributed and limited in magnitude (as a Euclidean norm in 16D space). This further justifies the validity of our solution manifold reconstruction process.



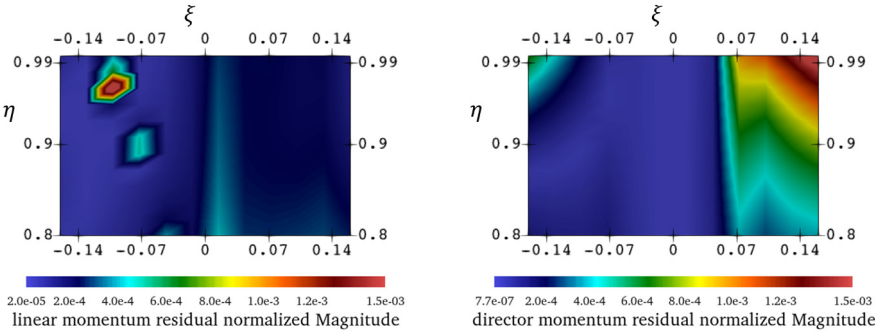
(a) linear momentum residual **(b)** director momentum residual

Fig. 25. Normalized momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (0, -1)$ at the last step simulation results of test paths # 3.



(a) linear momentum residual **(b)** director momentum residual

Fig. 26. Normalized momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (0, 1)$ at the last step simulation results of test paths # 1.

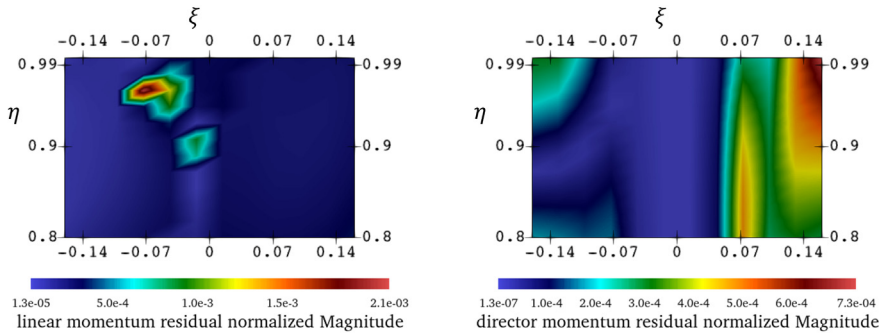


(a) linear momentum residual **(b)** director momentum residual

Fig. 27. Normalized momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (0, 1)$ at the last step simulation results of test paths # 2.

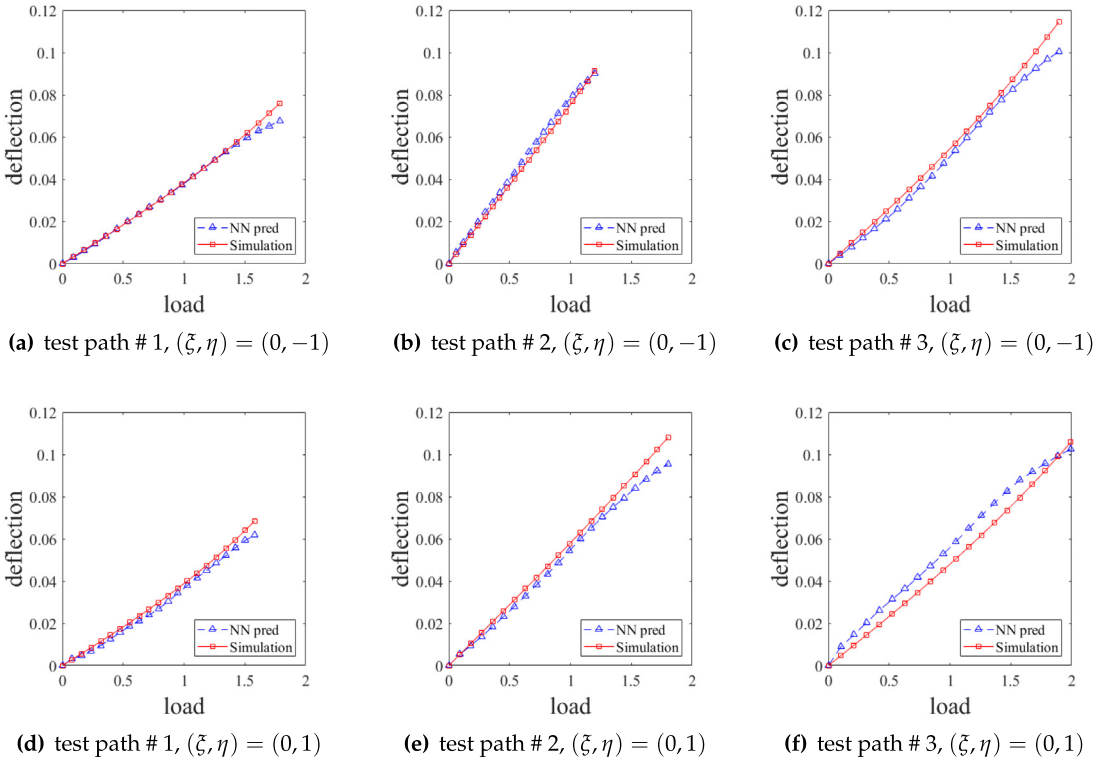
5.3. Cylindrical shell patch clamped at one boundary

This section provides numerical results for the cylindrical shell patch example introduced in 4.1.3, so that we further demonstrate how the response surface is established for non-concentrated loads. As we have shown in Section 5.2, we present the quantitative prediction results with the deformed shell configuration for the last snapshot



(a) linear momentum residual **(b)** director momentum residual

Fig. 28. Normalized momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (0, 1)$ at the last step simulation results of test paths # 3.



(a) test path # 1, $(\xi, \eta) = (0, -1)$ **(b)** test path # 2, $(\xi, \eta) = (0, -1)$ **(c)** test path # 3, $(\xi, \eta) = (0, -1)$

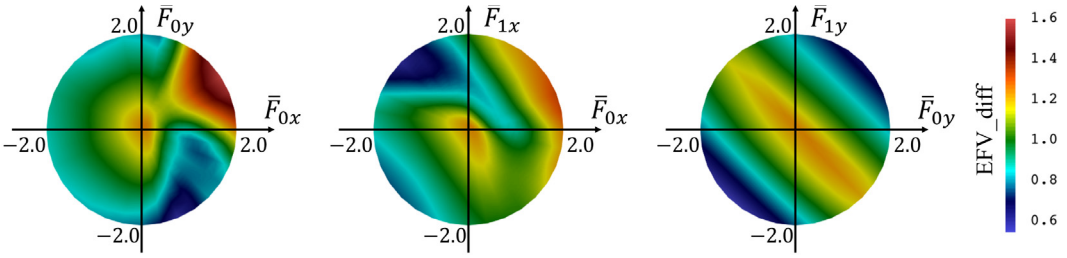
(d) test path # 1, $(\xi, \eta) = (0, 1)$ **(e)** test path # 2, $(\xi, \eta) = (0, 1)$ **(f)** test path # 3, $(\xi, \eta) = (0, 1)$

Fig. 29. Load response curve over the three prescribed test paths. x axis value indicates the magnitude of the prescribed force at either point $(\xi, \eta) = (0, -1)$ or $(\xi, \eta) = (0, 1)$ (unit: kN); y axis value indicates the magnitude of horizontal deflection at the same point (unit: m). The simulated load curve is well recovered by the NN prediction in all cases.

of the three testing paths we prescribed in 4.1.3. The deformed configurations are plotted in Fig. 31, where the test paths are ordered by the polar angle value on the $\bar{P}_y \bar{P}_z$ plane.

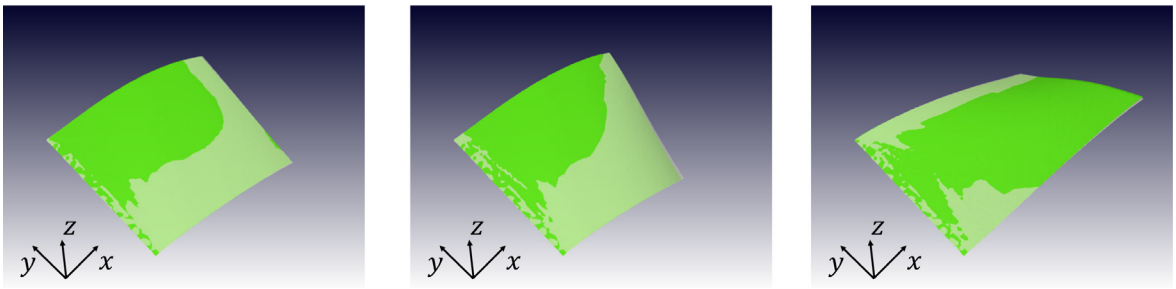
Next, we focus on the local response at the bottom-right corner ($(\xi, \eta) = (1, 0)$), as this location seems to exhibit the largest displacement over the computational domain. We first extract the load–displacement response at this point for the test paths in Fig. 32.

Then we make the physics-based adjustment to the local solutions at $(\xi, \eta) = (1, 0)$. For the 2-dimensional $\bar{P}_y \bar{P}_z$ load parametric space, two perturbed local solutions are sufficient to create an orthogonal basis to optimize the shell prediction locally. The adjusted local shell configurations for three different loading paths are shown in Fig. 33.



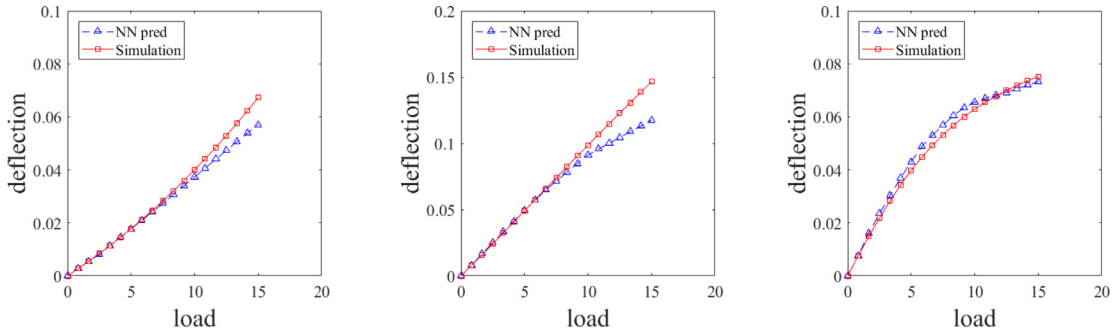
(a) on $\bar{F}_{0x}\bar{F}_{0y}$ plane **(b)** on $\bar{F}_{0x}\bar{F}_{1x}$ plane **(c)** on $\bar{F}_{0y}\bar{F}_{1y}$ plane

Fig. 30. L2-norm of the difference (vector) between the embedded EFV and the predicted EFV. The resultant scalar field as a function of the parametric load (in 4D Euclidean space) is visualized by taking the cross-section view on three planes: (a) $\bar{F}_{0x}\bar{F}_{0y}$ plane, (b) $\bar{F}_{0x}\bar{F}_{1x}$ plane, and (c) $\bar{F}_{0y}\bar{F}_{1y}$ plane.



(a) test path # 1 **(b)** test path # 2 **(c)** test path # 3

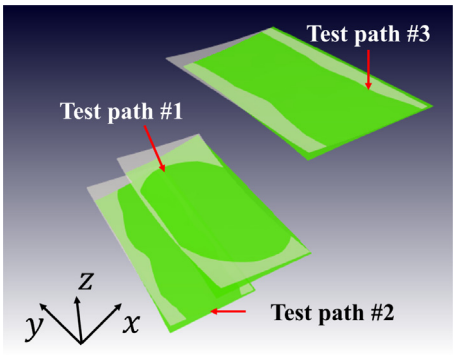
Fig. 31. Predicted deformed shell configurations (cylindrical patch) at the last step simulation results of the three prescribed test paths demonstrated in Section 4.1.3. Displacements are scaled up by 2.0 times. The green surface indicates the ground-truth configuration while the gray surface indicates the configuration predicted from our model. The prediction agrees well with the reference one. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) test path # 1 **(b)** test path # 2 **(c)** test path # 3

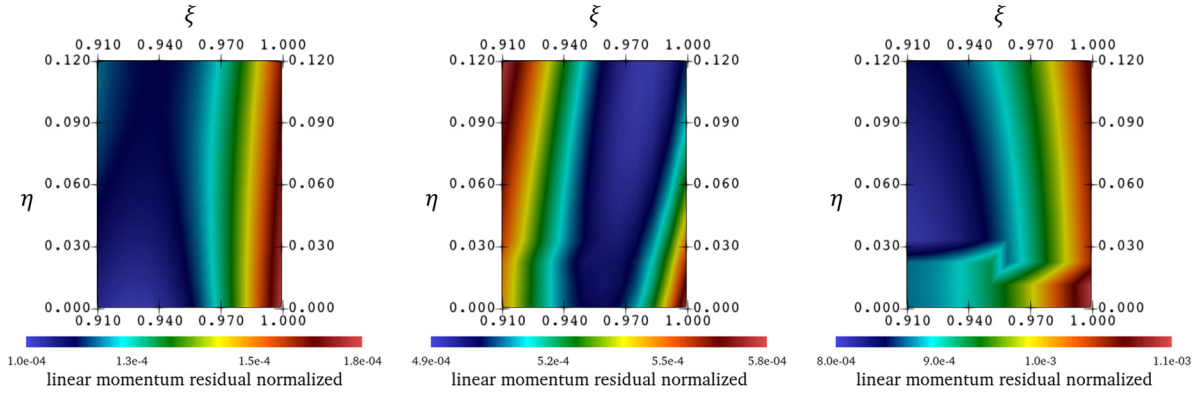
Fig. 32. Load response curve over the three prescribed test paths. x-axis value is equivalent to $\sqrt{\bar{P}_y^2 + \bar{P}_z^2}$ (unit: kN/m); y-axis value indicates the magnitude of displacement of the bottom-right corner at $(\xi, \eta) = (1, 0)$ (unit: m). The simulated load curve is well recovered by the NN prediction in all cases.

We also compute linear momentum residual and director momentum residual fields within the neighborhood of this location in Figs. 34 and 35. The small residual values verify that our model works fine for a shell model with distributed loads. And interestingly, the residual fields in Figs. 34 and 35 are quite smooth rather than those in Figs. 23–28, where the authors speculate that it is caused by the fact that the ground-truth solution does not exhibit a singularity as the load is not concentrated.



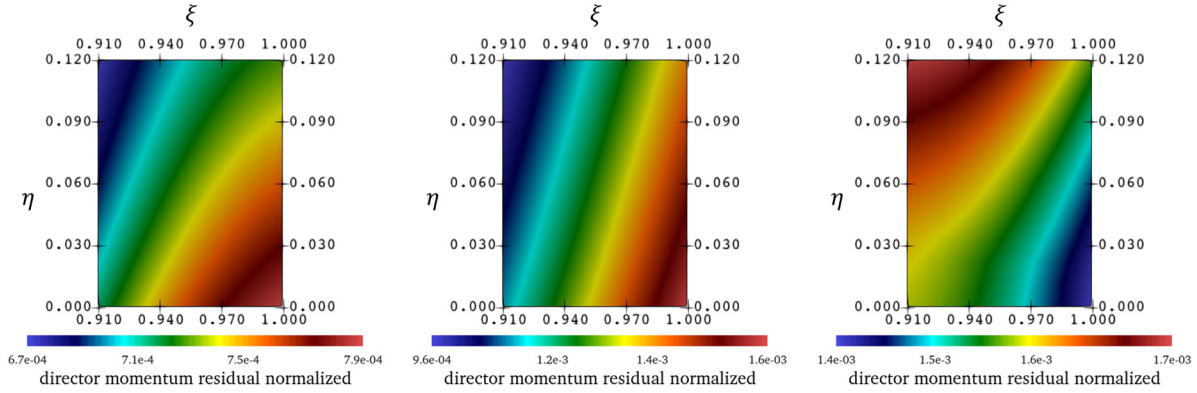
(a) configurations around $(\xi, \eta) = (1, 0)$

Fig. 33. Resulting local patch on the shell configuration (cylindrical patch) after correction based on physical constraints for $(\xi, \eta) = (1, 0)$ (bottom-right corner). The green surface indicates the reference local patch simulated shell configuration; the gray surface indicates the corrected local patch. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



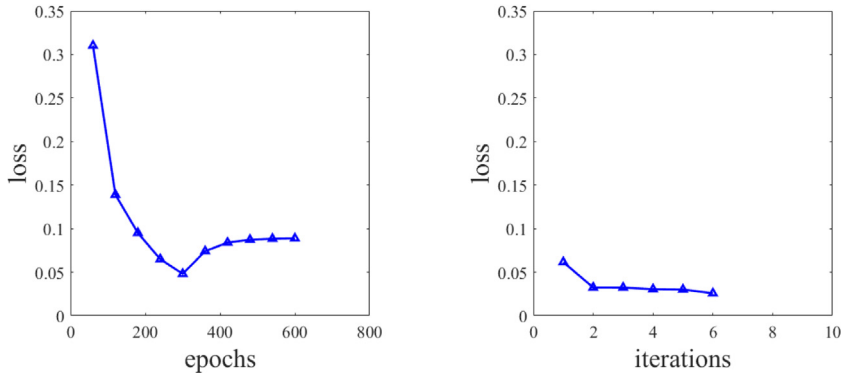
(a) test path # 1 **(b)** test path # 2 **(c)** test path # 3

Fig. 34. Normalized linear momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (1, 0)$ at the last step simulation results of the three test paths (the cylindrical patch example).



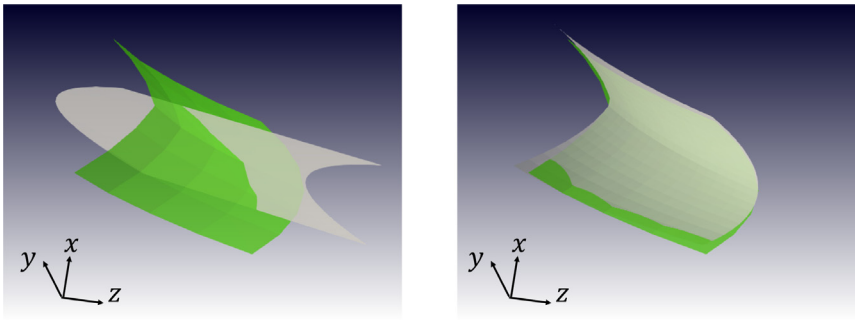
(a) test path # 1 **(b)** test path # 2 **(c)** test path # 3

Fig. 35. Normalized weighted director momentum residual vector fields in the neighborhood of parametric coordinates $(\xi, \eta) = (1, 0)$ at the last step simulation results of the three test paths (the cylindrical patch example).



(a) direct physics-informed training **(b)** Algorithm 1

Fig. 36. Example loss history recorded during the physics-informed adjustment of the shell configuration locally using the following approach: (a) direct physics-informed training; (b) Algorithm 1. The physical residual in case (a) first decrease and then increase to a relatively large value, indicating a potential failure of training.



(a) direct physics-informed training **(b)** Algorithm 1

Fig. 37. Example of the local shell configuration after the adjustment with physical constraints. The green surface indicates the reference local shell configuration obtained from a numerical simulation; the gray surface indicates the shell configuration obtained using the following approach: (a) direct physics-informed training; (b) Algorithm 1. The adjusted configuration in case (a) cannot match the reference one at all, while it is the opposite in case (b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Intuitively, we may propose direct functional mappings for $\hat{\boldsymbol{\varphi}}$ and $\hat{\boldsymbol{t}}$ parametrized by neural networks and train it with the loss derived from residual equations in (14) via physics-informed learning [50,53]. We call this methodology as *direct physics-informed training* herein. However, fine-tuning the neural network to suppress this physical loss function is not trivial. Due to the non-convexity of the optimization problem, it is difficult to guarantee the convergence rate of the supervised learning problem as they are highly sensitive to hyperparameters, including the neural network architecture and optimization strategies [50].

Figs. 36 and 37 compare the training performance and the deformed configurations of the shell obtained via the physics-constrained neural network and those obtained from the proposed correction algorithm. These results clearly indicate the issue of non-convexity of the physics-informed training in the sense that a local loss minimizer which is far from the equilibrium (loss ≈ 0.1) is encountered. It should be indicated a more sophisticated hyperparameter tuning, a more efficient optimization algorithm and a longer exploration (with more epochs) might all increase the probability of getting a better result. However, these interventions are not feasible for online or time-sensitive simulations where maintaining the consistency of the solver speed is necessary.

6. Conclusions

This paper presents a predictor–corrector geometric learning digital twin that combines the expressive power of graph isomorphism neural network and nonlinear graph embedding to adpatively generate orthogonal bases locally

that predict the admissible deformed configuration of a geometrically exact shell undergoing stable deformation. Our numerical experiments indicate that the reduced-order model is capable of delivering robust simulation-free predictions that are several orders more accurate than those obtained from the interpolated response surface. Another salient feature of the current approach is that it only requires training of the neural network at the training phase of the model, but not during the deploying of the models when it is used to make predictions. This feature helps us deliver more robust and predictable performance, both in terms of accuracy and execution speed. Our numerical results show that the proposed framework is capable of delivering simulation-free or reduced-order simulation results with reasonable accuracy and robustness at affordable training and deployment costs. On the other hand, the major limitation of the proposed method is that it is not yet able to handle predictions for unstable problems where bifurcated solutions may exist. Future works in this line of research may include the development of such a remedy to handle bifurcation and for systems very sensitive to perturbations. Another two major directions may include techniques that can handle transient and dynamic solutions, and the usage of the proposed model to deduce constitutive models for composites, such as woven composite and laminated composite shells. Researches in these directions are currently in progress.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors thank the two anonymous reviewers for their constructive feedback and suggestions. The authors is grateful for Professor Jure Leskovec who allow us to audit the course CS224W in the winter quarter of 2023 at Stanford and the support provided by the UPS Foundation and the Department of Civil and Environmental engineering of Stanford University that enables the authors to complete this manuscript at the Stanford campus. The authors are supported by the National Science Foundation under grant contracts CMMI-1846875, and the Dynamic Materials and Interactions Program from the Air Force Office of Scientific Research under grant contracts FA9550-21-1-0391 and FA9550-21-1-0027, and the MURI Grant No. FA9550-19-1-0318. These supports are gratefully acknowledged. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors, including the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors, including the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Stephen Timoshenko, Sergius Woinowsky-Krieger, et al., Theory of Plates and Shells, Vol. 2, McGraw-hill New York, 1959.
[2] Raymond David Mindlin, H.H. Bleich, Response of an Elastic Cylindrical Shell to a Transverse, Step Shock Wave, American Society of Mechanical Engineers, 1953.
[3] Chris R. Calladine, Theory of Shell Structures, Cambridge University Press, 1983.
[4] Petr Krysl, Jiun-Shyan Chen, Benchmarking computational shell models, Arch. Comput. Methods Eng. 30 (1) (2023) 301–315.
[5] Alejandro Arteaga Mota, A Class of Geometrically Exact Membrane and Cable Finite Elements Based on the Hu-Washizu Functional, Cornell University, 2000.
[6] M.E. Bechly, P.D. Clausen, Structural design of a composite wind turbine blade using finite element analysis, Comput. Struct. 63 (3) (1997) 639–646.
[7] Philip Avery, Daniel Z. Huang, Wanli He, Johanna Ehlers, Armen Derkevorkian, Charbel Farhat, A computationally tractable framework for nonlinear dynamic multiscale modeling of membrane woven fabrics, Internat. J. Numer. Methods Engrg. 122 (10) (2021) 2598–2625.
[8] Liliana Beldie, Göran Sandberg, Lars Sandberg, Paperboard packages exposed to static loads–finite element modelling and experiments, Packag. Technol. Sci. Int. J. 14 (4) (2001) 171–178.

[9] Thomas J.R. Hughes, Wing Kam Liu, Nonlinear finite element analysis of shells: Part I. Three-dimensional shells, Comput. Methods Appl. Mech. Engrg. 26 (3) (1981) 331–362.
[10] Eduardo N. Dvorkin, Klaus-Jürgen Bathe, A continuum mechanics based four-node shell element for general non-linear analysis, Eng. Comput. (1984).
[11] Gary Mitchel Stanley, Continuum-Based Shell Elements, Stanford University, 1985.
[12] Wing K. Liu, E.S. Law, D. Lam, T. Belytschko, Resultant-stress degenerated-shell element, Comput. Methods Appl. Mech. Engrg. 55 (3) (1986) 259–300.
[13] J.L. Ericksen, C. Truesdell, Exact theory of stress and strain in rods and shells, Arch. Ration. Mech. Anal. 1 (1) (1957) 295–323.
[14] Paul Mansour Naghdi, The theory of shells and plates, in: Linear Theories of Elasticity and Thermoelasticity, Springer, 1973, pp. 425–640.
[15] Juan C. Simo, David D. Fox, On a stress resultant geometrically exact shell model. Part I: Formulation and optimal parametrization, Comput. Methods Appl. Mech. Engrg. 72 (3) (1989) 267–304.
[16] J.C. Simo, D.D. Fox, M.S. Rifai, On a stress resultant geometrically exact shell model. Part II: The linear theory; computational aspects, Comput. Methods Appl. Mech. Engrg. 73 (1) (1989) 53–92.
[17] Adnan Ibrahimbegović, Stress resultant geometrically nonlinear shell theory with drilling rotations—Part I. A consistent formulation, Comput. Methods Appl. Mech. Engrg. 118 (3–4) (1994) 265–284.
[18] Hee Yuel Roh, Maenghyo Cho, The application of geometrically exact shell elements to B-spline surfaces, Comput. Methods Appl. Mech. Engrg. 193 (23–26) (2004) 2261–2299.
[19] Erwin Kreyszig, Differential Geometry, Courier Corporation, 2013.
[20] Richard P. Vinci, Joost J. Vlassak, Mechanical behavior of thin films, Annu. Rev. Mater. Sci. 26 (1) (1996) 431–462.
[21] Emma Lejeune, Ali Javili, Christian Linder, Understanding geometric instabilities in thin films via a multi-layer model, Soft Matter 12 (3) (2016) 806–816.
[22] Jon E. Olson, Joint pattern development: Effects of subcritical crack growth and mechanical crack interaction, J. Geophys. Res. Solid Earth 98 (B7) (1993) 12251–12265.
[23] Christian Linder, Francisco Armero, Finite elements with embedded strong discontinuities for the modeling of failure in solids, Internat. J. Numer. Methods Engrg. 72 (12) (2007) 1391–1433.
[24] Isabelle Staude, Jörg Schilling, Metamaterial-inspired silicon nanophotonics, Nat. Photonics 11 (5) (2017) 274–284.
[25] Jacob Fish, Gregory J. Wagner, Sinan Keten, Mesoscopic and multiscale modelling in materials, Nature Mater. 20 (6) (2021) 774–786.
[26] Lihua Jin, Alex Chortos, Feifei Lian, Eric Pop, Christian Linder, Zhenan Bao, Wei Cai, Microstructural origin of resistance–strain hysteresis in carbon nanotube thin film conductors, Proc. Natl. Acad. Sci. 115 (9) (2018) 1986–1991.
[27] Siddhant Kumar, Stephanie Tan, Li Zheng, Dennis M. Kochmann, Inverse-designed spinodoid metamaterials, npj Comput. Mater. 6 (1) (2020) 1–10.
[28] Cyril Touzé, Alessandra Vizzaccaro, Olivier Thomas, Model order reduction methods for geometrically nonlinear structures: a review of nonlinear techniques, Nonlinear Dynam. 105 (2) (2021) 1141–1190.
[29] Sanjay Lall, Petr Krysl, Jerrold E. Marsden, Structure-preserving model reduction for mechanical systems, Physica D 184 (1–4) (2003) 304–318.
[30] Charbel Farhat, Philip Avery, Todd Chapman, Julien Cortial, Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency, Internat. J. Numer. Methods Engrg. 98 (9) (2014) 625–662.
[31] Shobhit Jain, Paolo Tiso, Simulation-free hyper-reduction for geometrically nonlinear structural dynamics: a quadratic manifold lifting approach, J. Comput. Nonlinear Dyn. 13 (7) (2018).
[32] Patricia Astrid, Siep Weiland, Karen Willcox, Ton Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Trans. Automat. Control 53 (10) (2008) 2237–2251.
[33] Alejandro Cosimo, Alberto Cardona, Sergio Idelsohn, Improving the k-compressibility of hyper reduced order models with moving sources: applications to welding and phase change problems, Comput. Methods Appl. Mech. Engrg. 274 (2014) 237–263.
[34] Petr Krysl, Sanjay Lall, Jerrold E. Marsden, Dimensional model reduction in non-linear finite element dynamics of solids and structures, Internat. J. Numer. Methods Engrg. 51 (4) (2001) 479–504.
[35] Sergio R. Idelsohn, Alberto Cardona, A load-dependent basis for reduced nonlinear structural dynamics, Comput. Struct. 20 (1–3) (1985) 203–210.
[36] Kwangkeun Kim, Adrian G. Radu, X.Q. Wang, Marc P. Mignolet, Nonlinear reduced order modeling of isotropic and functionally graded plates, Int. J. Non-Linear Mech. 49 (2013) 100–110.
[37] Tiangang Cui, Youssef M. Marzouk, Karen E. Willcox, Data-driven model reduction for the Bayesian solution of inverse problems, Internat. J. Numer. Methods Engrg. 102 (5) (2015) 966–990.
[38] Xinran Zhong, WaiChing Sun, An adaptive reduced-dimensional discrete element model for dynamic responses of granular materials with high-frequency noises, Int. J. Multiscale Comput. Eng. 16 (4) (2018).
[39] Xinran Zhong, WaiChing Sun, Ying Dai, A reduced-dimensional explicit discrete element solver for simulating granular mixing problems, Granul. Matter 23 (2021) 1–13.
[40] Nikolaos N. Vlassis, Ran Ma, WaiChing Sun, Geometric deep learning for computational mechanics part I: anisotropic hyperelasticity, Comput. Methods Appl. Mech. Engrg. 371 (2020) 113299.
[41] Nikolaos N. Vlassis, WaiChing Sun, Geometric learning for computational mechanics part II: Graph embedding for interpretable multiscale plasticity, Comput. Methods Appl. Mech. Engrg. 404 (2023) 115768.
[42] Aleksandr Mikhailovich Lyapunov, The general problem of the stability of motion, Internat. J. Control 55 (3) (1992) 531–534.

[43] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, Michael W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.
[44] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, Kevin Murphy, Machine learning on graphs: A model and comprehensive taxonomy, J. Mach. Learn. Res. 23 (89) (2022) 1–64.
[45] Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021, arXiv preprint arXiv:2104.13478.
[46] Glaucio H. Paulino, Ivan F.M. Menezes, Marcelo Gattass, Subrata Mukherjee, Node and element resequencing using the laplacian of a finite element graph: part I—general concepts and algorithm, Internat. J. Numer. Methods Engrg. 37 (9) (1994) 1511–1530.
[47] Alejandro Mota, Jaroslaw Knap, Michael Ortiz, Fracture and fragmentation of simplicial finite element meshes using graphs, Internat. J. Numer. Methods Engrg. 73 (11) (2008) 1547–1570.
[48] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka, How powerful are graph neural networks? 2018, arXiv preprint arXiv:1810.00826.
[49] Boris Weisfeiler, Andrei Leman, The reduction of a graph to canonical form and the algebra which appears therein, NTI, Ser. 2 (9) (1968) 12–16.
[50] Jan-Hendrik Bastek, Dennis M. Kochmann, Physics-informed neural networks for shell structures, Eur. J. Mech. A Solids 97 (2023) 104849.
[51] Nikolaos N. Vlassis, WaiChing Sun, Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening, Comput. Methods Appl. Mech. Engrg. 377 (2021) 113695.
[52] Bahador Bahmani, WaiChing Sun, Training multi-objective/multi-task collocation physics-informed neural network with student/teachers transfer learnings, 2021, arXiv preprint arXiv:2107.11496.
[53] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[54] Minglang Yin, Enrui Zhang, Yue Yu, George Em Karniadakis, Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems, Comput. Methods Appl. Mech. Engrg. (2022) 115027.
[55] Tapas Tripura, Souvik Chakraborty, Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems, Comput. Methods Appl. Mech. Engrg. 404 (2023) 115783.
[56] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.
[57] David L. Donoho, Carrie Grimes, Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. 100 (10) (2003) 5591–5596.
[58] Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, Shantanu Sinha, Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids, Data-Cent. Eng. 1 (2020) e19.
[59] Qizhi He, Jiun-Shyan Chen, A physics-constrained data-driven approach based on locally convex reconstruction for noisy database, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112791.
[60] Theodore Kim, Doug L. James, Skipping steps in deformable simulation with online model reduction, in: ACM SIGGRAPH Asia 2009 Papers, 2009, pp. 1–9.
[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).
[62] Adam - PyTorch 1.13 documentation, URL https://pytorch.org/docs/stable/generated/torch.optim.Adam.html.
[63] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, Jure Leskovec, Identity-aware graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 12, 2021, pp. 10737–10745.