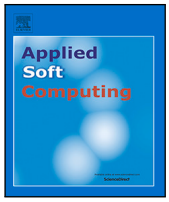
Applied Soft Computing Journal 153 (2024) 111312

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc



Enhancing elasticity models with deep learning: A novel corrective source term approach for accurate predictions

Sondre Sørbø [a], Sindre Stenen Blakseth [a,e], Adil Rasheed [b,c,*], Trond Kvamsdal [a,c], Omer San [d]

[a] Department of Mathematical Sciences, Norwegian University of Science and Technology, Norway
[b] Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway
[c] Mathematics and Cybernetics, SINTEF Digital, Norway
[d] Department of Mechanical, Aerospace and Biomedical Engineering, University of Tennessee, Knoxville, United States of America
[e] Department of Gas Technology, SINTEF Energy Research, Norway

ARTICLE INFO

Keywords:
Deep neural networks
Hybrid analysis and modeling
Corrective source term approach
Predictive modeling
Partial differential equations

ABSTRACT

With the recent wave of digitalization, specifically in the context of safety–critical applications, there has been a growing need for computationally efficient, accurate, generalizable, and trustworthy models. Physics-based models have traditionally been used extensively for simulating and understanding complex phenomena. However, these models though trustworthy and generalizable to a wide array of problems, are not ideal for real-time. To address this issue, the physics-based models are simplified. Unfortunately, these simplifications, like reducing the dimension of the problem (3D to 2D) or linearizing the highly non-linear characteristics of the problem, can degrade model accuracy. Data-driven models, on the other hand, can exhibit better computational efficiency and accuracy. However, they fail to generalize and operate as blackbox, limiting their acceptability in safety–critical applications. In the current article, we demonstrate how we can use a data-driven approach to correct for the two kinds of simplifications in a physics-based model. To demonstrate the methodology's effectiveness, we apply the method to model several elasticity problems. The results show that the hybrid approach, which we call the corrective source term approach, can make erroneous physics-based models more accurate and certain. The hybrid model also exhibits superior performance in terms of accuracy, model uncertainty, and generalizability when compared to its end-to-end data-driven modeling counterpart.

1. Introduction

Predictive modeling and simulation has traditionally been dominated by physics-based modeling (PBM). With the rise of machine learning (ML) in recent times, data-driven modeling (DDM)[1] has shown its ability to outperform PBM in many situations [1–4]. However, DDM comes with its own disadvantages, limiting these methods' overall usefulness. In [5], the authors describe the ideal model in the context of digital twins as generalizable, trustworthy, computationally efficient, accurate and self-evolving. A model's generalizability is its ability to solve various problems without problem-specific fine-tuning. Trustworthiness refers to the extent to which a model is explainable or interpretable, while computational efficiency and accuracy refer to the model's ability to make real-time predictions that match ground truth as closely as possible. Lastly, a model is self-evolving if it can learn and evolve when new situations are encountered. PBMs, when based on the correct physics, can be accurate and generalizable but are usually computationally demanding and do not adapt to new scenarios automatically. DDMs, on the other hand, after training, are very efficient, possibly very accurate and can be self-evolving. However, they typically lack in trustworthiness and generalizability.

For both PBM and DDM, a model's accuracy depends on the knowledge used to build the model. To achieve high predictive accuracy, PBMs require a proper understanding of all relevant physical phenomena, as well as mathematical methods for solving the equations used to represent these phenomena. If our physics knowledge is inaccurate and/or we lack methods for efficient computation, PBMs' accuracy will be significantly reduced. Contrary to PBMs, DDMs do not rely on physics knowledge in order to achieve high accuracy. Instead, they rely on (possibly large amounts of) data that is representative for the scenarios in which the model is to be used. With good data,

* Corresponding author at: Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway.
E-mail addresses: sondre.sorbo@sintef.no (S. Sørbø), sindre.blakseth@sintef.no (S.S. Blakseth), adil.rasheed@ntnu.no (A. Rasheed), trond.kvamsdal@ntnu.no (T. Kvamsdal), osan@utk.edu (O. San).

[1] Throughout the text, we will use the acronym DDM to refer to both data-driven modeling and data-driven models. Similarly, PBM may refer to physics-based modeling or physics-based models.

https://doi.org/10.1016/j.asoc.2024.111312
Received 6 February 2023; Received in revised form 25 January 2024; Accepted 27 January 2024
Available online 30 January 2024

Nomenclature

PBM	Physics-Based Modeling
ML	Machine Learning
DDM	Data-Driven Modeling
HAM	Hybrid Analysis and Modeling
NN	Neural Network
RNN	Recurrent Neural Network
ROM	Reduced-Order Modeling
ODE	Ordinary Differential Equation
DNN	Deep Neural Network
PINN	Physics-Informed Neural Network
PDE	Partial Differential Equation
PGML	Physics-Guided Machine Learning
CoSTA	Corrective Source Term Approach
$\boldsymbol{u}$	Displacement
$\boldsymbol{\sigma}$	Cauchy stress tensor
n	Number of spatial dimensions
$\boldsymbol{f}$	Imposed structural load
t	Time
$\boldsymbol{\epsilon}$	Strain
$\boldsymbol{C}$	Constitutive tensor relating stress and strain
E	Young's modulus
ν	Poisson ratio
$\mathbf{D}$	Strain operator
γ	Engineering shear strain
Ω	Spatial domain
T	Final time
IBVP	Initial Boundary Value Problem
$\boldsymbol{x}$	Position
$\partial\Omega$	Domain boundary
$\partial\Omega_{\mathrm{D}}$	Domain boundary segment with Dirichlet boundary conditions
$\partial\Omega_{\mathrm{N}}$	Domain boundary segment with Neumann boundary conditions
$\boldsymbol{n}$	Boundary normal vector
g_{D}	Dirichlet boundary data
g_{N}	Neumann boundary data
$\boldsymbol{\mu}_0$	Initial condition for displacement
$\boldsymbol{\rho}_0$	Initial condition for the time derivative of displacement
VP	Variational Problem
X	n-dimensional Hilbert space on the spacial domain (Ω)
X_D	Subset of X satisfying boundary conditions
X_0	Subspace of X vanishing at the boundary
$Y(\cdot)$	Solution search space with spacial search space $\cdot$
$\boldsymbol{u}_{\mathrm{D}}$	Lifting function for non-homogeneous boundary conditions
$\boldsymbol{u}_0$	Unknown part of displacement with homogeneous boundary conditions
$\boldsymbol{v}$	Test function
$\boldsymbol{w}$	Test function
$a(\cdot,\cdot)$	Bilinear form from the variational problem formulation
$(\cdot,\cdot)$	Bilinear form from the variational problem formulation
$l(\cdot)$	Linear form from the variational problem formulation
$\cdot_h$	Discrete approximation of $\cdot$
N_{dof}	Number of degree of freedom in finite element approximation
N_{el}	Number of elements in the domain triangulation
ϕ_i	Nodal basis function
$(u_{h,0})_j(t)$	Coefficient j of the nodal basis representation of the trial function
v_i	Coefficient i of the nodal basis representation of the test function
$P_1(\cdot)$	The space of linear polynomials on $\cdot$
A	Stiffness matrix
F	Load vector
M	Mass matrix
U	Finite element approximation of displacement
$\cdot^{(i)}$	Quantity $\cdot$ evaluated at time level i
k	Time step
N	Number of neural network layers
D	Mapping defined by a neural network
$\mathcal{T}_i$	Affine transformation in layer i of a neural network
s_i	Activation function in layer i of a neural network
d_i	Number of nodes in layer i of a neural network
L_Ω	General differential operator
$L_{\partial\Omega}$	General differential operator
ω	Unknown of interest in general differential equation
θ	General source term
ψ	General boundary condition
$\tilde{\cdot}$	Perturbed instance of $\cdot$ (perturbation caused by e.g. some error or noise)
r	Residual between perturbed and unperturbed differential equation
$\tilde{\tilde{\cdot}}$	Corrected instance of perturbed quantity $\tilde{\cdot}$
α	Parametrization of system state
$\mathcal{A}_{\mathrm{train}}$	Set of α-values used for training
$\mathcal{A}_{\mathrm{val}}$	Set of α-values used for validation
$\mathcal{A}_{\mathrm{test}}$	Set of α-values used for testing
K	Number of time steps
$\vec{\cdot}$	Vector form of exact solution
$\bar{\cdot}$	Prediction based on previous exact step
$\hat{\cdot}$	Prediction based on previous predicted step
x	First spacial coordinate
y	Second spacial coordinate
z	Third spacial coordinate
RRMSE	Relative Root Mean Square Error

DDMs can deliver highly accurate results at only a fraction of the computational cost of PBMs with comparable accuracy, if such a PBM is even available. However, given unrepresentative data, DDMs perform poorly due to their lack of generalizability. Moreover, since DDMs rely on data rather than explainable and verifiable physics knowledge, their interpretability is lacking. This combination of poor generalizability and interpretability results in poor trustworthiness and has prevented widespread utilization of DDM in high-stakes applications where even a single poor prediction can be detrimental.

The steadily increasing industrial adoption of digital twins [6] entails that models possessing all the characteristics identified by [5] are as relevant as ever. However, as we have seen, neither PBM nor DDM possess all four of these characteristics. As such, their respective deficiencies imply that neither paradigm is suitable for reaping the full benefits of the real-time monitoring and control enabled by digital twins.

To counter these deficiencies, hybrid analysis and modeling (HAM) is emerging as a new eclectic paradigm that combines techniques from both PBM and DDM. The HAM approach unites the advantages of PBM, such as generalizability, interpretability, solid foundation, and comprehension, with the accuracy, computational efficiency, and automatic pattern-recognition abilities of DDM. In their recent reviews, [7] and [5] offer a comprehensive look at techniques for combining DDM with PBM. A lot of the hybridization techniques can be classified into the following categories: (i) Embedding PBMs inside neural networks (NNs), (ii) Model order reduction, (iii) Physics-based regularization terms, (iv) Data-driven equation discovery, (v) Error correction approaches, (vi) Sanity check mechanisms using PBMs.

In the following sections, we present related work and discuss the advantages and disadvantages of the approaches.

1.1. Methods for embedding PBMs directly into NNs

This approach to hybridization is the most straightforward. More advanced techniques usually create a differentiable PBM that can be used as a layer in a neural network. For instance, OptNet [8] is a differentiable convex optimization solver that can be used as a layer in a network. In [9] the authors proposed the differentiable physics engine, a rigid body simulator that can be embedded in a NN. They showed that it is possible to learn a mapping from visual data to the positions and velocities of objects, which are then updated using the simulator. Authors in [10] simulated a structural dynamics problem by designing a hybrid recurrent NN (RNN) that contains an implicit numerical integrator. These approaches are usually quite data-efficient, but they can make both inference and training more expensive.

1.2. Model order reduction methods

The reduced-order modeling (ROM) approach has been widely used to project complex partial differential equations onto a reduced dimensional space based on the singular value decomposition of the offline high fidelity simulation data, resulting in a set of ordinary differential equations (ODEs) which are much faster to solve [11,12]. This has enabled high-fidelity numerical solvers to be accelerated by several orders of magnitude. However, ROMs can become unstable in the presence of unknown/unresolved complex physics. To address this issue, recent research has demonstrated how unknown and hidden physics within a ROM framework can be accounted for using deep neural networks (DNNs) [13,14]. Nevertheless, ROMs require the exact form of the original equation before they can be applied.

1.3. Physics-based regularization terms

By incorporating a physics-based model (PBM) into the objective function, deep learning models (DDMs) can be guided to adhere to known physical laws during training. An example of this is the physics-informed neural network (PINN) proposed by [15], which uses a NN to represent the solution to a partial differential equation (PDE) and adds an additional loss term to penalize any deviations from the equation at a sample of points. [16] applied the PINN approach to solve heat transfer problems in manufacturing processes, while [17] extended it to enable control in a state-space setting. [18] created a model to classify the health of the bearings by training a NN on physics-based features and regularizing the model using the output from a physics-based threshold model. More applications of PINN can be found in heat transfer modeling [19], multicomponent reactor modeling [20] or in predicting the lifetime under multiaxial loading [21]. However, these approaches require precise knowledge of the loss term and can be difficult to train due to the increased complexity of the cost function, particularly if the regularization term necessitates the evaluation of a complex model.

1.4. Data-driven equation discovery

Sparse regression, which is based on l_1 regularization, and symbolic regression, which is based on gene expression programming, have been demonstrated to be highly successful in uncovering hidden or partially known physical laws from data. Examples of this type of approach can be found in [22] and [23]. On a more applied side, the authors in [24] discovered the equation describing the relationship between features and material characteristics while Meyer et al. [25] utilized thermodynamically consistent neural network to model plasticity and discover the associated evolution laws. However, there are some drawbacks to this class of methods. For instance, with sparse regression, extra features must be manually created, while with symbolic regression, the resulting models can be unstable and prone to overfitting.

1.5. Physics guided machine learning

On many occasions part of the physics governing a process is known, but the actual form describing is not known. To exploit such partial knowledge, Pawar et al. [26] proposed a physics-guided machine learning (PGML) approach. The basic idea behind the PGML approach is to inject partial knowledge into one of the layers within a DNN to guide the training process. Partially knowledge can, for example, come from a simplistic model as has been shown in [26,27]. More recently, the Theseus library [28] provides a framework for conducting guided training of neural networks. However, this approach does not take advantage of the partially known form of the equation.

1.6. Proposal

From the previous discussion, it is clear that almost all HAM approaches have pros and cons. One limitation of all the methods is that they are more tilted towards the data-driven modeling approach. To this end, Corrective Source Term Approach (CoSTA) has been proposed recently. CoSTA is a method proposed by [29] that explicitly addresses the problem of unknown physics. This is done by augmenting the governing equations of a PBM describing partial physics with a DNN-generated corrective source term that takes into account the remaining unknown/ignored physics. One added benefit of the CoSTA approach is that the physical laws can be used to keep a sanity check on the predictions of the DNN used, i.e. checking conservation laws. A similar approach has also been used to model unresolved physics in turbulent flows [30,31]. The method has also been shown to work well for modeling complex aluminium extraction process [32]. In this regard, the main contributions distinguishing the present work from previous publications on CoSTA [29,32–34], are summarized below.

- We investigate whether CoSTA can be used to correct modeling errors incurred due to **dimensionality reduction** in PBMs. Reducing the dimensionality of a model is commonly seen in engineering applications in order to reduce computational complexity, thereby achieving real-time performance. However, this comes at the cost of reduced predictive accuracy. CoSTA has not previously been used to correct for this kind of modeling error.
- We investigate whether CoSTA can be used to correct modeling errors incurred due to **linearization** of non-linear governing equations. Linearization is another technique that is commonly used to speed up models, and which also reduces predictive accuracy. The use of CoSTA for correcting linearization error has been touched upon briefly in [33], but is considered in much greater detail in the present work.

- We study the effect of randomness in DNN initialization and training procedures on CoSTA's performance, relative to stand-alone PBM and DDM.
- We demonstrate how to combine CoSTA with PBMs based on the **finite element method**. Previously, CoSTA has only been used in conjunction with finite volume methods.
- We apply CoSTA to a new class of problems: **elasticity modeling**. Previously, CoSTA has only been used to model heat diffusion, which is a fundamentally different phenomenon.

The article is structured as follows: Section 2 presents the relevant theory for the work. Section 3 presents the method applied in the paper and a description of the cases considered. The results are presented and discussed in Section 4. Finally, in Section 5, conclusions are given, and potential future work is presented.

2. Theory

In this section, we discuss the theory underlying the models used in our numerical experiments. We begin with general considerations regarding PBM in Section 2.1, while Sections 2.2 and 2.3 are devoted to the physics-based elasticity models used in the present work. The latter sections are largely based on the textbooks by [35] and [36]. DDM (Section 2.4), CoSTA for elasticity problems (Section 2.5) and the method of manufactured solutions (Section 2.6) are the remaining topics covered in this section.

2.1. Physics based modeling

PBM involves careful observation of a physical phenomenon of interest (elasticity in the current work), development of its partial understanding, expression of the understanding in the form of mathematical equations and ultimately solution of these equations. Due to the partial understanding and numerous assumptions along the steps from observation to solution of the equations, a large portion of the important governing physics gets ignored. Due to high computational costs, high fidelity simulators with minimal assumptions have so far been limited to the offline design phase only. Despite this major drawback, what makes these models attractive are sound foundations from first principles, interpretability, generalizability and existence of robust theories for the analysis of stability and uncertainty. The PBMs used in this paper are based on partial differential equations (PDEs) describing linear elasticity in solid materials (cf. Section 2.2). These PDEs are discretized using the finite element method[2] along the spatial dimension, and the backward Euler method along the temporal dimension. The numerical methods will, in the vast majority of interesting scenarios, introduce some discretization error into these models. Moreover, modeling error may be present as well due to the governing equations being simplified, incomplete or even incorrect. Sometimes, modeling error may be purposefully introduced to simplify the model, thereby increasing its computational efficiency. Other times, modeling error could stem from a lack of knowledge about the system to be modeled. In this work, we explore the impact of modeling error through PDE linearization, dimensionality reduction and unknown load terms, as described in Section 3.

2.2. Linear elasticity modeling

The purpose of elastic modeling is to predict the response of a solid material to external, typically mechanical loads. This response is quantified in terms of the displacement $\boldsymbol{u}$ and stresses $\boldsymbol{\sigma}$. For an n-dimensional system, $\boldsymbol{u}$ is a n-dimensional vector field whose elements describe a point's actual position in relation to where that point would have been if no forces were applied to the system. Moreover, $\boldsymbol{\sigma}$, known as the Cauchy stress tensor, is a second order tensor with dimension $n\times n$, describing the system's internal forces. The displacement and stresses are related through Newton's 2nd law written for elastic continua. For isotropic systems with unit mass density, this form of Newton's 2nd law reads

$$\nabla\cdot\boldsymbol{\sigma}-\ddot{\boldsymbol{u}}=-\boldsymbol{f}. \tag{1}$$

Here, $\boldsymbol{f}$ denotes the forces (loads) acting on the system and $\ddot{\boldsymbol{u}}=\partial^2\boldsymbol{u}/\partial t^2$ is the acceleration field where t is time. Since we have two unknown fields ($\boldsymbol{\sigma}$ and $\boldsymbol{u}$), but only one equation, we need another constraint to have a well-defined problem. To achieve this we introduce a kinematic relationship between the strain field $\boldsymbol{\varepsilon}$ (that describes the relative stretching (deformation) of an infinitesimal element of the system) and the displacement $\boldsymbol{u}$, see Eq. (2), and then a constitutive relation between the stresses $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$, see Eq. (3), where the $\boldsymbol{C}$ is the fourth order constitutive tensor.

$$\varepsilon=\frac{1}{2}(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^T) \tag{2}$$

$$\boldsymbol{\sigma}=\boldsymbol{C}\varepsilon \tag{3}$$

In computational mechanics, it is common to introduce the Voigt notation to represent the symmetric stress and strain tensors as vectors, $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ respectively, and the fourth order constitutive tensor $\boldsymbol{C}$ as a two-dimensional matrix $\mathbf{C}$.

For a system with constant Young's modulus, E, and constant Poisson ratio, ν, the Eqs. (1)–(2) can be written more conveniently as

$$\boldsymbol{\varepsilon}=\mathbf{D}\boldsymbol{u} \tag{4}$$

$$\boldsymbol{\sigma}=\mathbf{C}\boldsymbol{\varepsilon} \tag{5}$$

$$\mathbf{D}^{\mathrm{T}}\boldsymbol{\sigma}-\ddot{\boldsymbol{u}}=-\boldsymbol{f}, \tag{6}$$

where the precise definitions of $\mathbf{C}$, $\boldsymbol{\sigma}$, $\boldsymbol{\varepsilon}$, and the strain giving differential operator $\mathbf{D}$ depend on the number of space dimensions n. We will refer to Eqs. (4)–(6) as the linear elasticity equations.

In 2D, we have

$$\mathbf{D}=\begin{bmatrix}\frac{\partial}{\partial x}&0\\0&\frac{\partial}{\partial y}\\\frac{\partial}{\partial y}&\frac{\partial}{\partial x}\end{bmatrix},\quad \boldsymbol{\varepsilon}=\begin{bmatrix}\varepsilon_{xx}\\\varepsilon_{yy}\\\gamma_{xy}\end{bmatrix},\quad \boldsymbol{\sigma}=\begin{bmatrix}\sigma_{xx}\\\sigma_{yy}\\\sigma_{xy}\end{bmatrix}, \tag{7}$$

and

$$\mathbf{C}=\frac{E}{1-\nu^2}\begin{bmatrix}1&\nu&0\\\nu&1&0\\0&0&\frac{1-\nu}{2}\end{bmatrix}, \tag{8}$$

while the 3D definitions read

$$\mathbf{D}=\begin{bmatrix}\frac{\partial}{\partial x}&0&0\\0&\frac{\partial}{\partial y}&0\\0&0&\frac{\partial}{\partial z}\\0&\frac{\partial}{\partial z}&\frac{\partial}{\partial y}\\\frac{\partial}{\partial z}&0&\frac{\partial}{\partial x}\\\frac{\partial}{\partial y}&\frac{\partial}{\partial x}&0\end{bmatrix},\quad \boldsymbol{\varepsilon}=\begin{bmatrix}\varepsilon_{xx}\\\varepsilon_{yy}\\\varepsilon_{zz}\\\gamma_{yz}\\\gamma_{zx}\\\gamma_{xy}\end{bmatrix},\quad \boldsymbol{\sigma}=\begin{bmatrix}\sigma_{xx}\\\sigma_{yy}\\\sigma_{zz}\\\sigma_{yz}\\\sigma_{zx}\\\sigma_{xy}\end{bmatrix}. \tag{9}$$

and

$$\mathbf{C}=\frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu&\nu&\nu&0&0&0\\\nu&1-\nu&\nu&0&0&0\\\nu&\nu&1-\nu&0&0&0\\0&0&0&\frac{1-2\nu}{2}&0&0\\0&0&0&0&\frac{1-2\nu}{2}&0\\0&0&0&0&0&\frac{1-2\nu}{2}\end{bmatrix}. \tag{10}$$

Notice that, $\gamma_{yz}=2\varepsilon_{yz}$, $\gamma_{zx}=2\varepsilon_{zx}$, and $\gamma_{xy}=2\varepsilon_{xy}$, are the engineering shear strains.

[2] See e.g. [37] or [38] for excellent introductions to finite element method.

2.2.1. Initial boundary value problem

We will consider Eqs. (4) and (5) on a bounded spatial domain Ω, for a time interval $[0, T]$. For these equations to have a unique solution, initial and boundary conditions must be prescribed. The resulting initial boundary value problem (IBVP) reads

$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\varepsilon}(\boldsymbol{u}) \qquad \forall \boldsymbol{x} \in \Omega \tag{11}$$

$$\mathbf{D}^{\mathrm{T}}\boldsymbol{\sigma} - \ddot{\boldsymbol{u}} = -\boldsymbol{f} \qquad \forall \boldsymbol{x} \in \Omega \tag{12}$$

$$\boldsymbol{u}(t, \boldsymbol{x}) = \boldsymbol{g}_{\mathrm{D}}(t, \boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \partial\Omega_{\mathrm{D}} \tag{13}$$

$$\boldsymbol{\sigma}(t, \boldsymbol{x}) \cdot \boldsymbol{n} = \boldsymbol{g}_{\mathrm{N}}(t, \boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \partial\Omega_{\mathrm{N}} \tag{14}$$

$$\boldsymbol{u}(0, \boldsymbol{x}) = \boldsymbol{\mu}_0(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \Omega \tag{15}$$

$$\dot{\boldsymbol{u}}(0, \boldsymbol{x}) = \boldsymbol{\rho}_0(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \Omega, \tag{16}$$

where $\partial\Omega_{\mathrm{D}}$ and $\partial\Omega_{\mathrm{N}}$ are the parts of the boundary with respectively Dirichlet boundary condition $\boldsymbol{g}_{\mathrm{D}}$ and Neumann boundary condition $\boldsymbol{g}_{\mathrm{N}}$, $\boldsymbol{n}$ is the unit vector normal to the boundary, and $\boldsymbol{\mu}_0$ and $\boldsymbol{\rho}_0$ are the initial value conditions. In the present work, we use Dirichlet boundary conditions on all of the boundaries, i.e., $\partial\Omega_{\mathrm{D}} = \partial\Omega$.

2.2.2. Variational formulation

Using the Galerkin approach we can transform the IBVP above to a Variational Problem (VP). To facilitate this transformation we introduce the following function spaces:

$$X = \boldsymbol{H}^1(\Omega) = [H^1(\Omega)]^n \tag{17}$$

$$X_{\mathrm{D}} = \{\boldsymbol{v} \in X \,:\, \boldsymbol{v} = \boldsymbol{g}_{\mathrm{D}} \text{ on } \partial\Omega_{\mathrm{D}}\} \tag{18}$$

$$X_0 = \{\boldsymbol{v} \in X \,:\, \boldsymbol{v} = 0 \text{ on } \partial\Omega_{\mathrm{D}}\} \tag{19}$$

$$Y(X) = \{\boldsymbol{v} \,:\, \forall t \in [0, T], \boldsymbol{v}(\boldsymbol{x}, t) \in X, \int_0^T \|\boldsymbol{v}\|^2_{\boldsymbol{H}^2(\Omega)}\, dt < \infty\} \tag{20}$$

Here, $\boldsymbol{H}^1(\Omega) = [H^1(\Omega)]^n$ and $\boldsymbol{H}^2(\Omega) = [H^2(\Omega)]^n$ are the Hilbert spaces for functions with first and second order weak derivatives, respectively, for problems with $\Omega \in \mathbb{R}^n$, and we define the spaces $Y(X_{\mathrm{D}})$ and $Y(X_0)$ by substituting X with X_{D} and X_0 in Eq. (20). Furthermore, we split the unknown displacement into two parts: $\boldsymbol{u} = \boldsymbol{u}_0 + \boldsymbol{u}_{\mathrm{D}}$, where $\boldsymbol{u}, \boldsymbol{u}_{\mathrm{D}} \in X_{\mathrm{D}}$ and $\boldsymbol{u}_0 \in X_0$. The so-called *lifting* function $\boldsymbol{u}_{\mathrm{D}}$ is introduced in order to handle non-homogeneous Dirichlet conditions, see e.g., [37] for details. The variational formulations then read

$$\text{Find } \boldsymbol{u}_0 \in Y(X_0) : \; a(\boldsymbol{u}_0, \boldsymbol{v}) + \frac{d^2}{dt^2}(\boldsymbol{u}_0, \boldsymbol{v}) = l(\boldsymbol{v}) \;\; \forall \boldsymbol{v} \in X_0, \tag{21}$$

where the bilinear forms are defined as

$$a(\boldsymbol{w}, \boldsymbol{v}) = \int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{v})^{\mathrm{T}} \mathbf{C} \boldsymbol{\varepsilon}(\boldsymbol{w})\, d\Omega \;\; \forall \boldsymbol{w}, \boldsymbol{v} \in X, \tag{22}$$

$$(\boldsymbol{w}, \boldsymbol{v}) = \int_\Omega \boldsymbol{v}^{\mathrm{T}} \boldsymbol{w}\, d\Omega \;\; \forall \boldsymbol{w}, \boldsymbol{v} \in X, \tag{23}$$

and the linear form reads

$$l(\boldsymbol{v}) = \int_\Omega \boldsymbol{f}\boldsymbol{v}\, d\Omega + \int_{\partial\Omega_{\mathrm{N}}} \boldsymbol{g}_{\mathrm{N}} \boldsymbol{v}\, d\partial\Omega - a(\boldsymbol{u}_{\mathrm{D}}, \boldsymbol{v}) - \frac{d^2}{dt^2}(\boldsymbol{u}_{\mathrm{D}}, \boldsymbol{v}) \;\; \forall \boldsymbol{v} \in X_0, \tag{24}$$

2.2.3. Semi-discretization with finite elements

By discretizing in space we obtain the semi-discrete formulation:

$$\text{Find } \boldsymbol{u}_0 \in Y(X_{h,0}) : \; a(\boldsymbol{u}_{h,0}, \boldsymbol{v}) + \frac{d^2}{dt^2}(\boldsymbol{u}_{h,0}, \boldsymbol{v}) = l(\boldsymbol{v}) \;\; \forall \boldsymbol{v} \in X_{h,0}, \tag{25}$$

Here, $X_{h,0} \subset X_0$, and $\dim(X_{h,0}) = N_{\mathrm{dof}} < \infty$. For linear finite elements we can express $X_{h,0}$ as

$$X_{h,0} = \left\{ v \in X_0 \,|\, v_{|\Omega_e} \in P_1(\Omega_e),\; e = 1, \ldots, N_{\mathrm{el}} \right\} \tag{26}$$

$$= \mathrm{span}\{\phi_1, \phi_2, \ldots, \phi_{N_{\mathrm{dof}}}\} \tag{27}$$

where the domain Ω are properly triangulated into simplex elements (lines in 1D, triangles in 2D, and tetrahedrons in 3D) with domain Ω_e, and N_{dof} and N_{el} are the number of degrees of freedom and number of finite elements, respectively. Using the nodal basis functions $\{\phi_i\}$, $i = 1, \ldots, N_{\mathrm{dof}}$ we can represent any test function in $X_{h,0}$ and trial function in $Y(X_{h,0})$ as follows:

$$\forall \boldsymbol{v} \in X_{h,0} \qquad \boldsymbol{v}(\boldsymbol{x}) = \sum_{i=1}^{N_{\mathrm{dof}}} v_i\, \boldsymbol{\phi}_i(\boldsymbol{x}) \tag{28}$$

$$\forall \boldsymbol{u}_{h,0}(\boldsymbol{x}) \in Y(X_{h,0}) \qquad \boldsymbol{u}_{h,0} = \sum_{j=1}^{N_{\mathrm{dof}}} (u_{h,0})_j(t)\, \boldsymbol{\phi}_i(\boldsymbol{x}) \tag{29}$$

Notice that, the coefficients for the trial functions, $(u_{h,0})_j(t)$, are time-dependent, whereas that is not the case for the coefficients for the test functions, v_i.

By insertion of the nodal basis into Eq. (25) we get the following system of ordinary differential equations (ODEs):

$$\boldsymbol{A}\boldsymbol{U}(t) + \boldsymbol{M}\ddot{\boldsymbol{U}}(t) = \boldsymbol{F}(t) \tag{30}$$

where an element of the system *stiffness matrix* $\boldsymbol{A}$ is defined by $A_{ij} = a(\phi_i, \phi_j)$, an element of the system *mass matrix* $\boldsymbol{M}$ is defined by $M_{ij} = (\phi_i, \phi_j)$, an element of the system *load vector*, $\boldsymbol{F}$ is defined by $F_i = l(\phi_i)$, and an element of the unknown finite element solution $\boldsymbol{U}$ is $U_i(t) = (u_{h,0})_i(t)$, where the range of the indexes are: $i, j = 1, \ldots, N_{\mathrm{dof}}$.

2.2.4. System of discrete equations

To obtain a fully discrete system of equations we need to discretize Eq. (30) in time. We will here use a second order accurate implicit Euler finite-difference approximations:

$$\boldsymbol{A}\boldsymbol{U}^{(i+1)} + \boldsymbol{M}\frac{1}{k^2}\left[\boldsymbol{U}^{(i-1)} - 2\boldsymbol{U}^{(i)} + \boldsymbol{U}^{(i+1)}\right] = \boldsymbol{F}^{(i+1)} \tag{31}$$

Here, $\boldsymbol{U}^{(i)}$ denotes the approximate solution $\boldsymbol{U}(t)$ at the time $t^i = ik$, where k is the time step. After rearranging the term we get the following system of algebraic equations:

$$\left(\boldsymbol{A} + \frac{1}{k^2}\boldsymbol{M}\right)\boldsymbol{U}^{(i+1)} = \boldsymbol{F}^{(i+1)} + \frac{1}{k^2}\boldsymbol{M}\left[2\boldsymbol{U}^{(i)} - \boldsymbol{U}^{(i-1)}\right] \tag{32}$$

This system can be solved by any appropriate solver, preferably a sparse solver for 2D and small 3D problems. For large problems (e.g., in 3D) iterative solvers as Conjugate Gradient Method will typically be a better choice. Starting from U_0, the known initial condition $\boldsymbol{\mu}_0$ projected onto X_h, repeatedly solving Eq. (32) allows us to march forward in time, eventually obtaining approximations of $\boldsymbol{u}(\boldsymbol{x}, t)$ up to the specified final time T. For the spatial discretization, we use piecewise linear Lagrange simplex elements [39] on an equidistant grid. As can be seen in Section 4.1, the relative error of the resulting model's predictions is roughly 1% or smaller in the scenarios where no modeling error is synthesized. This is a reasonable result for what must be considered as a fairly simple PBM, and serves as a good benchmark for DDM and CoSTA to beat.

2.3. Nonlinear elasticity modeling

As previously mentioned, the linear elasticity model introduced in the previous section is well-suited for modeling small strains and stresses.[3] More flexible elasticity modeling can be achieved by relaxing the assumptions that material properties like Young's modulus E and the Poisson ratio ν be constants. For the nonlinear elasticity cases considered herein, we consider E as a function of strain, i.e. $E = E(\boldsymbol{\varepsilon})$. To accommodate for this, a slight alteration to the system (4)–(5) is required, such that we obtain

$$\boldsymbol{\sigma} = \mathbf{C}(\boldsymbol{\varepsilon})\boldsymbol{\varepsilon} \tag{33}$$

$$\mathbf{D}^{\mathrm{T}}\boldsymbol{\sigma} - \ddot{\boldsymbol{u}} = -\boldsymbol{f}. \tag{34}$$

Here, we highlight that Eq. (33) is nonlinear in $\boldsymbol{u}$ via the Young's modulus E, and hence the constitutive matrix $\mathbf{C}$, dependence on the strain vector $\boldsymbol{\varepsilon}$.

[3] The precise meaning of "small" strains and stresses is highly material-dependent.

Table 1
Description of experiments and modeling errors (in addition to discretization error).

Exp. #	Modeled physics	PBM modeling error
1	2D Linear elasticity with load term	None
2	2D Linear elasticity with load term	Load term replaced with zero
3	3D Linear elasticity with reduced dimensionality	A dimension is ignored
4	2D Nonlinear elasticity	PDE is linearized

2.4. Data driven modeling using neural networks

We use deep neural networks (DNN) to create the purely data-driven models considered in herein. This is motivated by the observation that DNNs are universal approximators [40–42] and hence have the ability to model highly complex nonlinear phenomena. A DNN with N layers is a function $D : \mathbb{R}^{d_0} \to \mathbb{R}^{d_N}$ defined by

$$D(x) = \mathcal{T}_N \circ s_{N-1} \circ \mathcal{T}_{N-1} \circ s_{N-2} \cdots \circ s_1 \circ \mathcal{T}_1(x) \tag{35}$$

where $\mathcal{T}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ are affine transformations and $s_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$ are some preferably nonlinear activation functions.[4] Each transformation $\mathcal{T}_i$ is determined by a weight matrix and a bias vector, a total of $d_{i-1} \times d_i + d_i$ values. These values are tuned to minimize the DNN's predictive error on the training data. DNNs are widely used, due to their simplicity and yet astonishing performance in many situations. They can be applied to a large variety of problems, and have delivered impressive achievements across numerous fields of research and applications. Meanwhile, they also have some inherent weaknesses. They are prone to overfitting on the training data. While there are many ways of preventing the network from being too specialized [43], they will not be good at extrapolation cases where the prediction task is somehow qualitatively different from the training tasks. In addition, they are not easily explainable or predictable, meaning it is hard to explain what kind of patterns the network will look for, and unexpected predictions can occur. Compared to PBMs, DDMs usually make predictions much faster, but may need long training times for optimal performance.

2.5. Corrective source term approach

In this section we present a brief justification of the use of CoSTA. It is based on the more elaborate argument that can be found in [29].

Consider the differential equations

$$\begin{aligned} L_\Omega \omega &= \theta, \quad \forall \boldsymbol{x} \in \Omega \\ L_{\partial\Omega} \omega &= \psi, \quad \forall \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{36}$$

where $L_\Omega, L_{\partial\Omega}$ are differential operators, θ is a source term, ψ a function specifying the boundary condition, and ω is the unknown of interest. For notational simplicity we assume ω to be a scalar. Now let $\tilde{\omega}$ be the solution to the perturbed problem

$$\begin{aligned} \tilde{L}_\Omega \tilde{\omega} &= \tilde{\theta}, \quad \forall \boldsymbol{x} \in \Omega \\ \tilde{L}_{\partial\Omega} \tilde{\omega} &= \tilde{\psi}, \quad \forall \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{37}$$

where the perturbations $\tilde{\cdot}$ are due to imperfections such as unknown physics, modeling errors, discretization error, or noise. For example, we will later in this paper simplify a nonlinear operator L_Ω with a linear (and discretized) $\tilde{L}_\Omega$. Assume we can calculate the residuals defined as

$$\begin{aligned} r_\Omega &= \tilde{L}_\Omega(\omega - \tilde{\omega}) \\ r_{\partial\Omega} &= \tilde{L}_{\partial\Omega}(\omega - \tilde{\omega}), \end{aligned} \tag{38}$$

and let $\tilde{\tilde{\omega}}$ be the solution of the corrected, perturbed problem

$$\begin{aligned} \tilde{L}_\Omega \tilde{\tilde{\omega}} &= \tilde{\theta} + r_\Omega, \quad \forall \boldsymbol{x} \in \Omega \\ \tilde{L}_{\partial\Omega} \tilde{\tilde{\omega}} &= \tilde{\psi} + r_{\partial\Omega} \quad \forall \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{39}$$

Using the definition of the residuals, as well as the perturbed Eq. (37), we see that

$$\begin{aligned} \tilde{L}_\Omega \tilde{\tilde{\omega}} &= \tilde{\theta} + \tilde{L}_\Omega(\omega - \tilde{\omega}) = \tilde{L}_\Omega \omega, \quad \forall \boldsymbol{x} \in \Omega \\ \tilde{L}_{\partial\Omega} \tilde{\tilde{\omega}} &= \tilde{\psi} + \tilde{L}_{\partial\Omega}(\omega - \tilde{\omega}) = \tilde{L}_{\partial\Omega} \omega \quad \forall \boldsymbol{x} \in \partial\Omega, \end{aligned}$$

which reduces to $\tilde{\tilde{\omega}} = \omega$ if the corrected, perturbed problem (39) yields a unique solution. From this argument, we see that the source term corrections are able to compensate for perturbations in the differential operators as well as the source terms.

In real scenarios, we obviously cannot calculate the residual exactly, as that would require the solution we are trying to estimate. The idea of the CoSTA method is to use a DDM to estimate the residual. For the input of the DDM we use the uncorrected solution $\tilde{\omega}$. This means the PBM is used twice per time level, first to solve Eq. (37) for $\tilde{\omega}$, then Eq. (39) for $\tilde{\tilde{\omega}}$. As mentioned in [29], other DDM input choices are possible. For example, one could use the corrected prediction $\tilde{\tilde{\omega}}$ from the previous time level, such that the PBM is only used once per time level. Investigating different DDM input choices is outside the scope of the present work, so we stick with the DDM input used in [29].

In the case of linear elasticity, the variational formulation of Eq. (37) corresponds to Eq. (21). The corrected form of Eq. (32), corresponding to the general corrected perturbed Eq. (39), reads

$$\left(\boldsymbol{A} + \frac{1}{k^2}\boldsymbol{M}\right)\boldsymbol{U}^{(i+1)} = \boldsymbol{F}^{(i+1)} + \frac{1}{k^2}\boldsymbol{M}\left[2\boldsymbol{U}^{(i)} - \boldsymbol{U}^{(i-1)}\right] + \boldsymbol{r} \tag{40}$$

where $\boldsymbol{r}$ is the residual vector corresponding to Eq. (32) with an appropriate projection of the true solution inserted in the place of U. In the present work, we use a DNN to approximate the residual vector $\boldsymbol{r}$. As the boundary values are known (as we here assume $\partial\Omega_{\mathrm{D}} = \partial\Omega$), we do not need any correction for these elements. Therefore the DNN only needs to output values for the interior nodes.

2.6. Method of manufactured solutions

We seek to evaluate the performance of the models described above on a selection of different elasticity modeling problems. To this end, we use the method of manufactured solutions [44] to create exact reference data. For a governing equation written on the general form (36), the method involves choosing a solution ω, and calculating the source θ that admits the chosen ω as a solution to the governing equation. In order to evaluate the accuracy of a model, the model is used to approximate the chosen solution ω given the calculated θ as well as the correct boundary data $\omega|_{t=0}$ and $\omega|_\Omega$.

The alternative to using the method of manufactured solutions is to choose the conditions θ, $\omega|_{t=0}$ and $\omega|_\Omega$ to use for approximating the solution $\tilde{\omega}$. But then the correct solution is unknown, so the error $\omega - \tilde{\omega}$ must be approximated by using a more precise method.[5] Usually, this is very computationally demanding, and it only yields an approximate error. In comparison, the method of manufactured solutions is computationally inexpensive and introduces no uncertainty in the assessment of model accuracy.

[4] With linear activation functions the method reduces to multivariate linear regression. For the universal approximation property it should also be nonpolynomial.

[5] Usually high fidelity numerical solutions of the equation, with a fine grid.

Table 2
Values of the parameter α used for training, validation and testing.

Set usage	Notation	Values
Testing	$\mathcal{A}_{\text{test}}$	$\{-0.5, 0.7, 1.5, 2.5\}$
Validation	$\mathcal{A}_{\text{val}}$	$\{0.8, 1.1\}$
Training	$\mathcal{A}_{\text{train}}$	$\{0.1, 0.2, \dots 2.0\} \setminus (\mathcal{A}_{\text{test}} \cup \mathcal{A}_{\text{val}})$

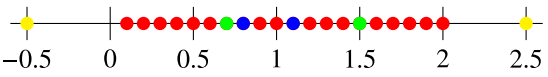


Fig. 1. Values of α used for training, validation, interpolation testing and extrapolation testing.

3. Methodology

In the present work, we consider four different numerical experiments, simply enumerated as Experiment 1–4. The goal of these experiments is to test the predictive accuracy and generalizability of CoSTA, in comparison to its constituent PBM and DDM components. This section is dedicated to describing the methodology of the experiments. Elements of the methodology that are common among all experiments are described in Section 3.1. What is unique for each experiments is then described in Sections 3.2–3.4. This is summarized in Table 1. All four experiments concern elasticity modeling using the PBM, DDM and CoSTA models introduced earlier.

3.1. General setup

3.1.1. Data generation

To conduct our numerical experiments, we need data for two purposes. First of all, we need reference data to which we can compare the predictions of our models. Moreover, we need training data for the purely data-driven model and the data-driven component of CoSTA. To obtain this data, we use the method of manufactured solutions (cf. Section 2.6).[6] For a lack of established benchmark solutions used to evaluate the accuracy of linear elasticity equation solvers, we have created our own manufactured solutions. In order to cover a broad class of solutions, the manufactured solutions include a variety of polynomial, exponential and harmonic functions. These functions are described in the sections covering experiment-specific information. What they all have in common is that they are parametrized by some variable α. Consequently, it is useful to view each manufactured solution as a family of functions, one for each value of α, corresponding to different system states. In the present work, we consider a total of 22 α-values, as listed in Table 2 and visualized in Fig. 1. 16 of these values are used for training the DNNs used in the DDM and CoSTA models,[7] while two more are used for DNN validation. The remaining four values are used for testing the accuracy of the models. Two of these lie within the range used to generate training data, while the other values are outside this range. We refer to the former as interpolation scenarios and the latter as extrapolation scenarios.

As mentioned in Section 2.2, we have two quantities of main interest in elasticity modeling: $\boldsymbol{\sigma}$ and $\boldsymbol{u}$.[8] These are related through the constitutive relation (4), which depends on the material properties E and ν. Unless otherwise stated, we always use $E = 1$ and $\nu = 0.25$. Using the constitutive relation, if we know $\boldsymbol{u}$, we can compute $\boldsymbol{\sigma}$, and vice versa. For our manufactured solutions, we elect to fix $\boldsymbol{u}$, E and ν. $\boldsymbol{\sigma}$ is then computed using Eq. (4), before $\boldsymbol{f}$ is computed using Eq. (5).

[6] In a real-world use-case, one might instead use sensor measurements or simulation data from a high-fidelity offline model.

[7] We highlight that PBM does not require any training.

[8] Recall that σ and $\boldsymbol{\sigma}$ are equivalent representations of the same variables.

Table 3
Overview of spatial and temporal resolution.

Exp. #	No. Elements	No. Time Steps, K
1	15×15	1000
2	15×15	1000
3	15×15	1000
4	10×10	500

3.1.2. Model summary

In our numerical experiments, we use the PBM, DDM and CoSTA models for elasticity problems that were introduced in Section 2. These are briefly summarized below.

PBM: Eq. (32) is solved using a linear finite element model with first order (i.e. piecewise linear) simplex Lagrange elements [39] on an equidistant grid. Unless otherwise specified, the spatial domain considered is the unit square $[0, 1] \times [0, 1]$, and the time domain is the unit interval $[0, 1]$. The time interval is divided into time steps of constant length $k = K^{-1}$, where K is the number of time steps. The spatial and temporal resolution used varies across experiments, as shown in Table 3.

DDM: For the DDM, we use a DNN with four dense hidden layers of 80 nodes each. The length of the input and output layers depend on the PDE and discretization. The input vector contains every basis function, while the output does not contain the functions on the boundary. Recall that the two-dimensional (2D) elasticity equation has a vector field solution with two basis functions per element. For the experiments with 15 elements in each dimension, this gives $(2 \times 16)^2$ input nodes and $(2 \times 14)^2$ output nodes, as shown in Fig. 2.

The neural networks are implemented using TensorFlow [45]. As activation function we use leaky ReLU [46], with coefficient 0.01 for negative inputs. Training parameters are presented in Table 4. A patience of 20 means the training stops when the score on the *validation* set has not improved for the last 20 optimizer steps. All the data is normalized before it is inputted in the DNN, and the output is unnormalized, based on the statistical properties of the training data.

CoSTA: Eq. (40) is solved using the exact same finite element method discretization used in the PBM with exactly the same spatial and temporal resolution. Moreover, the residual r is approximated by a DNN with exactly the same architecture and hyperparameter values as the DNN used for pure DDM.

3.1.3. Experimental procedure

The experimental procedure used is presented in Algorithm 1. K denotes the total number of time steps. Note the difference between $\check{\boldsymbol{U}}$, $\tilde{\boldsymbol{U}}$ and $\hat{\boldsymbol{U}}$: we denote by $\check{\boldsymbol{U}}$ the vector form of the exact solution.[9] The predictions $\tilde{\boldsymbol{U}}$, used for training, are based on the previous *exact* step, while $\hat{\boldsymbol{U}}$, used for testing, are based on the previous *predicted* step (the first predicted step is based on the exact initial values). As in Section 2, we use the superscript to denote the time step, e.g. $\hat{\boldsymbol{U}}^{(i)}$ is at time $t_i = ik$.

The PBM, DDM and CoSTA models map their predictions $\hat{\boldsymbol{U}}^{(i-1)}$ to $\hat{\boldsymbol{U}}^{(i)}$. This map is used iteratively to calculate $\hat{\boldsymbol{U}}^{(K)}$ at the final time step, from the initial (exact) input $\check{\boldsymbol{U}}^{(0)}$. We should therefore expect that any error in one step propagates into the next. It is not feasible to train DDM or CoSTA to counteract this by fixing the global error, because (1) The models would depend on the arbitrary starting time, and (2) the required amount of data and training time would increase drastically since the number of training examples per time series would decrease from K to 1. Therefore, the exact solution $\check{\boldsymbol{U}}$ is used as both the input and output training data for every time step.

For testing scenarios, the exact solution at time $t = 0$ is used as the input for the first step, and then each model predict each step based

[9] That is, the piecewise linear function characterized by $\check{\boldsymbol{U}}$ equals the exact solution on the grid points.

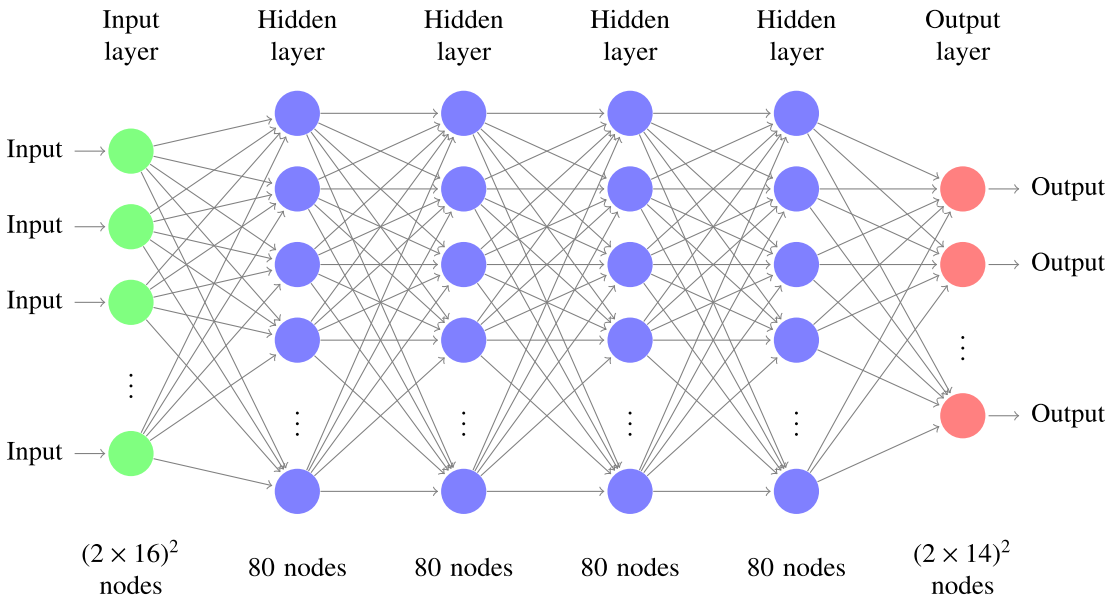


Fig. 2. Visualization of the DNN architecture for the experiments with 15 × 15 elements. The nodes represent input, output and intermediate values, while the arrows going between them represent dependencies. Generally there is one input node for each basis function, and one output node for each basis function not on the edge.

Algorithm 1: Pseudocode showing how the experiments were performed.

Pick a solution $\boldsymbol{u}_{\text{exact}}(t, x, y, z, \alpha)$;
Use governing equations (the PDE, before any simplification) to calculate $\boldsymbol{f}$ from $\boldsymbol{u}_{\text{exact}}$;
for $\alpha \in \mathcal{A}_{train}$ *and* $\alpha \in \mathcal{A}_{val}$ **do**
  for $i = 0, 1, 2 \ldots K-1$ **do**
    Use (simplified) PBM to calculate $\bar{\boldsymbol{U}}^{(i+1)}_{\text{PBM}} = \text{PBM}(\check{\boldsymbol{U}}^{(i)})$;
    Calculate the residual $\boldsymbol{r}^{(i+1)} = \hat{L}(\check{\boldsymbol{U}}^{(i+1)} - \bar{\boldsymbol{U}}^{(i+1)}_{\text{PBM}})$;
Train DDM to map $\check{\boldsymbol{U}}^{(i)}$ to $\check{\boldsymbol{U}}^{(i+1)}$, i.e. minimize
  $\sum_{\alpha \in \mathcal{A}_{\text{train}}} \sum_{i=0}^{K-1} \left| \check{\boldsymbol{U}}^{(i+1)} - \text{DDM}(\check{\boldsymbol{U}}^{(i)}) \right|^2$;
Train CoSTA network to map $\bar{\boldsymbol{U}}^{(i+1)}_{\text{PBM}}$ to $\boldsymbol{r}^{(i+1)}$, i.e. minimize
  $\sum_{\alpha \in \mathcal{A}_{\text{train}}} \sum_{i=0}^{K-1} \left| \boldsymbol{r}^{(i+1)} - \text{DDM}_{\text{CoSTA}}(\text{PBM}(\check{\boldsymbol{U}}^{(i)})) \right|^2$;
$\hat{\boldsymbol{U}}^{(0)}_{\text{DDM}} = \check{\boldsymbol{U}}^{(0)}$;
$\hat{\boldsymbol{U}}^{(0)}_{\text{PBM}} = \check{\boldsymbol{U}}^{(0)}$;
$\hat{\boldsymbol{U}}^{(0)}_{\text{CoSTA}} = \check{\boldsymbol{U}}^{(0)}$;
for $\alpha \in \mathcal{A}_{test}$ **do**
  for $i = 0, 1, 2 \ldots K-1$ **do**
    calculate $\hat{\boldsymbol{U}}^{(i+1)}_{\text{DDM}} = \text{DDM}(\hat{\boldsymbol{U}}^{(i)}_{\text{DDM}})$;
    calculate $\hat{\boldsymbol{U}}^{(i+1)}_{\text{PBM}} = \text{PBM}(\hat{\boldsymbol{U}}^{(i)}_{\text{PBM}})$;
    calculate $\hat{\boldsymbol{U}}^{(i+1)}_{\text{CoSTA}} = \text{PBM}(\hat{\boldsymbol{U}}^{(i)}_{\text{CoSTA}}, \text{DDM}_{\text{CoSTA}}(\text{PBM}(\hat{\boldsymbol{U}}^{(i)}_{\text{CoSTA}})))$;
Evaluate by comparing to exact solution $\boldsymbol{u}_{\text{exact}}$

on the previous. Boundary conditions are also used as input for each step, along with the source function values for the PBM. The error is measured at each step, and presented in the result sections.

For actual usage of this method for prediction, the first two lines and the last line of Algorithm 1 is skipped. Instead, $\boldsymbol{f}$ must be known (or approximated), along with $\boldsymbol{u}_{\text{exact}}$ in training scenarios, and the initial conditions and boundary conditions in the testing scenarios.

3.1.4. Evaluation and visualization

To evaluate the performance of the different methods, relative root mean square error (RRMSE) is measured at each time step. This value is defined

$$\text{RRMSE}(\boldsymbol{U}^{(i)}, \check{\boldsymbol{U}}^{(i)}) = \frac{\|\boldsymbol{U}^{(i)} - \check{\boldsymbol{U}}^{(i)}\|_2}{\|\check{\boldsymbol{U}}^{(i)}\|_2} \tag{41}$$

for a prediction $\boldsymbol{U}$ and correct solution $\check{\boldsymbol{U}}$. For models with stochastic results (i.e. DDM and CoSTA, that includes a DNN with random initialization), 10 models are trained and used. In a real world application, the objective would determine if we are interested in the least amount of error at all time steps, or only at the last one. In this work we are interested in both. In the result plots presented in Section 4, the mean of the RRMSE of the 10 models is plotted as a line on a logarithmic scale, as a function of time steps. The uncertainty is also quantified and visualized in the plots by shading the area between the mean RRMSE, and the mean RRMSE plus one standard deviation.[10] Mean RRMSE *minus* one standard deviation is not plotted since it might be negative, which does not work well on logarithmic scales.

Table 4
Parameters used for the training procedures for all of the neural networks.

Loss function	MSE
Optimizer	Adam [47]
Learning rate	$1e-5$
Patience	20

3.2. Modeling linear elasticity with known and unknown load term

In Experiments 1 and 2, we consider a system governed by the 2D transient linear elasticity Eqs. (4) and (5) with a non-zero load term $\boldsymbol{f}$. In Experiment 1, $\boldsymbol{f}$ is known to the PBM and CoSTA models. Consequently, discretization error will be the only source of error in the PBM, and the task of the DNN used in the CoSTA model will be to reduce this error. In Experiment 2, $\boldsymbol{f}$ is assumed unknown, so we set $\boldsymbol{f} = \boldsymbol{0}$ in both the PBM and CoSTA models, thereby synthesizing modeling error.[11] Since the DDM does not have explicit knowledge about $\boldsymbol{f}$, there

[10] The standard deviation is calculated using one reduced degree of freedom due to the estimation of the mean.

[11] In real-world use-cases, one should use the best available *a priori* estimation of $\boldsymbol{f}$.

Table 5
Manufactured solutions for the elastic problems.

Label	$\boldsymbol{u}(t, x, y, \alpha)$
e1	$\begin{bmatrix} \sin(\pi(x+\alpha y))\cos(\alpha t) \\ \cos(\pi(x+\alpha y))\sin(\alpha t) \end{bmatrix}$
e2	$\begin{bmatrix} \exp(\frac{(-tx^2+y^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2-y^2)}{(1+\alpha+t^2)}) \end{bmatrix}$
e3	$\begin{bmatrix} x^3+y^2(t+0.5)^{1.5}+xy\alpha \\ x^2+y^3(t+0.5)^{1.1}+xy\alpha \end{bmatrix}$

Table 6
Manufactured solutions for the dimensionally reduced elasticity problems.

Label	$u(t, x, y, \alpha)$
ed1	$\begin{bmatrix} \sin(\pi(x+\alpha y+\frac{1+\alpha}{2}z))\cos(\alpha t) \\ \cos(\pi(x+\alpha y+\frac{1+\alpha}{2}z))\sin(\alpha t) \\ -\cos(\pi(x+\alpha y+\frac{1+\alpha}{2}z))\sin(\alpha t) \end{bmatrix}$
ed2	$\begin{bmatrix} \exp(\frac{(-tx^2+y^2+z^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2-y^2+z^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2+y^2-z^2)}{(1+\alpha+t^2)}) \end{bmatrix}$
ed3	$\begin{bmatrix} x^3+y^2(t+\frac{1}{2})^{1.5}+xy\alpha+\sqrt{t+\frac{1}{2}}z^2+z(x+y)\alpha \\ x^2+y^3(t+\frac{1}{2})^{1.1}-xy\alpha+\sqrt{t+\frac{1}{2}}z^2+z(x-y)\alpha \\ x^2+y^2(t+\frac{1}{2})^{1.1}+xy\alpha+\sqrt{t+\frac{1}{2}}z^3+z(y-x)\alpha \end{bmatrix}$

is no difference between Experiments 1 and 2 from the perspective of the DDM. The manufactured solutions used in Experiments 1 and 2 are listed in Table 5. These are all solutions of the *2D* linear elasticity equations.

3.3. Modeling linear elasticity with reduced dimensionality

In general, a model's computational complexity and expense increase greatly with the number of dimensions being modeled. Conversely, if the situations allow it, reducing the dimensionality of a model can greatly reduce the model's computational cost. However, the model's accuracy will generally also be reduced. Experiment 3 is designed to investigate whether CoSTA can be used to correct errors introduced by dimensionality reduction. To this end, we use our 2D linear elasticity PBM, DDM and CoSTA models to model the displacement of a 2D plane in a 3D object, as illustrated in Fig. 3. The manufactured solutions, which are solutions of the *3D* linear elasticity equations, are listed in Table 6. Note that, since the displacement $\boldsymbol{u}$ is a vector field with as many components as there are dimensions, our 2D models cannot predict the z-component of $\boldsymbol{u}$.[12] Therefore, we consider only the x- and y-components of $\boldsymbol{u}$ when evaluating the predictive accuracy of our models. This approach is most relevant when displacement in the reduced direction is either small and/or of little interest compared to displacement in the other directions.

[12] An alternative approach to dimensional reduction for the linear elasticity equation (and other vector field PDEs), is to base the PBM on the three-dimensional model, but replace the derivatives in the ignored z-direction with zero (or some other appropriate value). The resulting PBM would produce predictions for all components of $\boldsymbol{u}$. Although the PBM prediction of the third component likely would be quite inaccurate, the CoSTA term could help. Due to the inter-dependencies of the components, this could potentially give a useful prediction of the third component of the displacement, that in turn could make the other components more accurate. This idea is not pursued further in this paper.

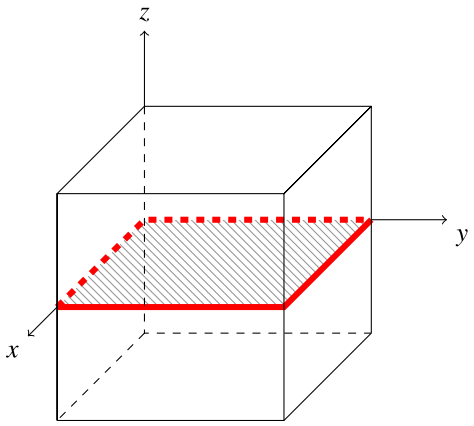


Fig. 3. Example of an object to model using the dimensional reduction method. While the full model is a three-dimensional cube, we only predict the shaded plane. Boundary conditions needed are those at the thick red edges at $z = 0$.

Table 7
Manufactured solutions for nonlinear elastic problems.

Label	$\boldsymbol{u}(t, x, y, \alpha)$	E
n1	$\begin{bmatrix} \sin(\pi(x+\alpha y))\cos(\alpha t) \\ \cos(\pi(x+\alpha y))\sin(\alpha t) \end{bmatrix}$	$\frac{5}{\sqrt{20+\|\boldsymbol{\varepsilon}\|_F}}$
n2	$\begin{bmatrix} \exp(\frac{(-tx^2+y^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2-y^2)}{(1+\alpha+t^2)}) \end{bmatrix}$	$\frac{5}{\sqrt{20+\|\boldsymbol{\varepsilon}\|_F}}$
n3	$\begin{bmatrix} x^3+y^2(t+0.5)^{1.5}+xy\alpha \\ x^2+y^3(t+0.5)^{1.1}+xy\alpha \end{bmatrix}$	$\frac{5}{\sqrt{20+\|\boldsymbol{\varepsilon}\|_F}}$

3.4. Modeling nonlinear elasticity

In Experiment 4, we consider a system with non-constant Young's modulus $E = E(\boldsymbol{\varepsilon})$. The stiffness of a material usually decreases with applied strain, so we choose a decreasing function

$$E(\boldsymbol{\varepsilon}) = \frac{5}{\sqrt{20 + \|\boldsymbol{\varepsilon}\|_F}}, \tag{42}$$

where $\|\cdot\|_F$ is the Frobenius norm. With this choice, the assumption of $E = 1$ in the PBM is obviously not true, but still "in the ball park" for the manufactured solutions we consider here. These manufactured solutions, which have the same displacement $\boldsymbol{u}$ as the solutions discussed in Section 3.2, are presented in Table 7.[13]

The non-linearity greatly increases the cost of generating reference data. Therefore. only 500 time steps, 10×10 elements and 5 initializations of DDM and CoSTA were used in this experiment. The DNN learning rate was also increased to 8×10^{-5}.

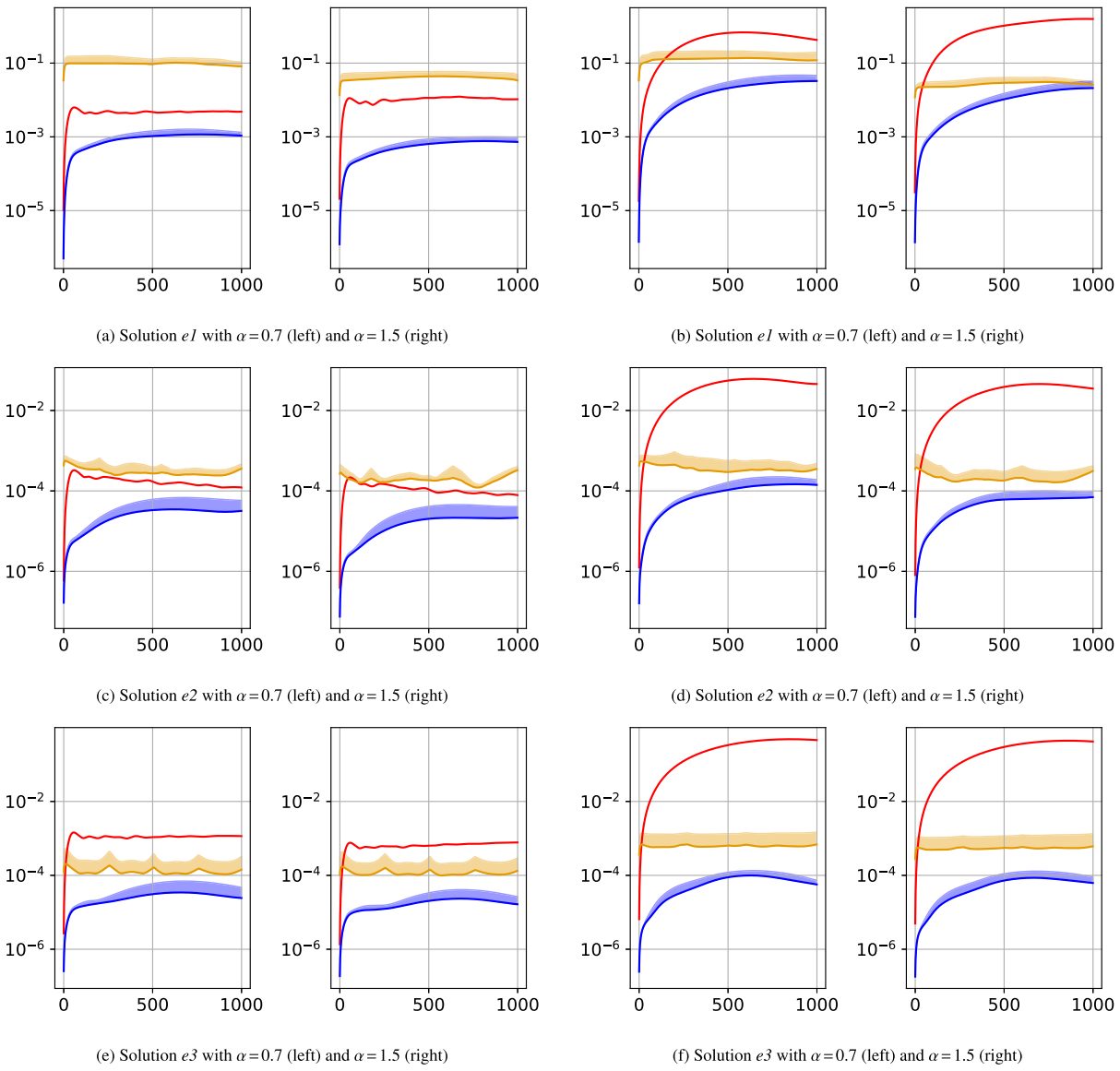
4. Results and discussion

For the exact implementation used for the experiments and figures in this work, see the first author's GitHub repository [48].

4.1. Experiments 1 & 2 – known and unknown load term

The temporal development of the models RRMSE is presented in Fig. 4 for interpolation scenarios and Fig. 5 for extrapolation scenarios. Results for the same manufactured solution with known and unknown

[13] Of course, since we use the same $\boldsymbol{u}$ but a different E, $\boldsymbol{\sigma}$ and $\boldsymbol{f}$ are different here than for Experiments 1 and 2. For the sake of brevity, we have not written out $\boldsymbol{\sigma}$ or $\boldsymbol{f}$ for any of out manufactured solutions here. However, they can be found in [48].



(a) Solution *e1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(b) Solution *e1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(c) Solution *e2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(d) Solution *e2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(e) Solution *e3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(f) Solution *e3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

Fig. 4. Temporal development of relative l_2 error for solutions with correct source term (left) and zero source term (right) in interpolation scenarios. CoSTA is the most accurate method in all the cases. (PBM, DDM, CoSTA).

load term are shown next to each other for easy comparison. We observe that CoSTA is more accurate than both PBM and DDM in all interpolation cases, and also in a significant majority of extrapolation cases. The only two cases where CoSTA is clearly not the most accurate model is $\alpha = -0.5$ for Solution *e1* with known source term and for Solution *e2* with unknown source term. In accordance with the discussion in Section 1, PBM performs at its best in the cases with no unknown physics, and DDM performs at its best in the interpolation scenarios.

The observation that CoSTA is generally more accurate than the PBM in Experiment 1 (with known load term) shows that CoSTA can be used to correct discretization error. This is in line with the findings in [29], and suggests that CoSTA can be used to speed up expensive PBMs by permitting coarse discretizations without loss of accuracy. As for Experiment 2 (with unknown load term), since CoSTA outperforms DDM in most cases, it is evident that even a PBM severely affected by modeling error can be valuable in combination with a DNN.

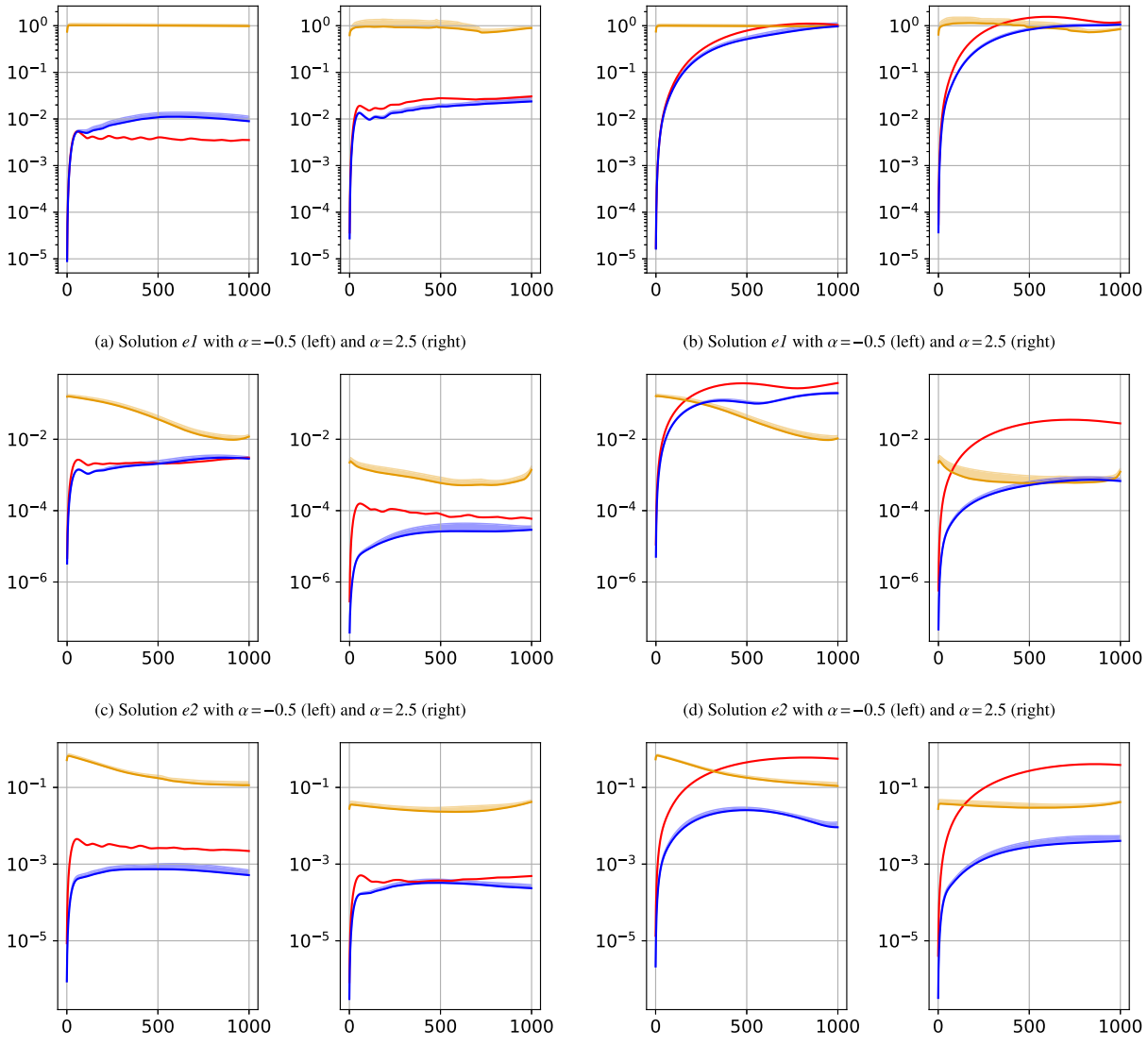
From the top row of Fig. 5, it is clear that the purely data-driven model struggles in both extrapolation scenarios for Solution *e1*. Indeed, the DDM predictions have relative errors of roughly 100%. Consequently, it is perhaps not so surprising that the data-driven component of CoSTA also struggles. Although the observed increase in error caused by CoSTA's DDM-generated correction term is undesirable, it is promising that CoSTA did not fail completely in a way similar to the DDM. This suggests that CoSTA is more robust than DDM.

For Solution *e2*, we observe that DDM performs quite well in all cases. This can perhaps be attributed to the simplicity of the solution, both in terms of spatial variability and its dependence on α.[14] Our results suggest that, in such simple cases, if the PBM is insufficiently accurate, it does not ease the learning task of the DDM, so pure DDM is more accurate. This is what we observe for Solution *e2* with $\alpha = -0.5$ and unknown load term.

Finally, we note that the error of CoSTA often fluctuates less and has a smaller standard deviation[15] than the DDM errors. This increased

[14] Solution *e2* at $t = 0$ is largely dependent on α, while its value at $t = 1$ is much less so. Apart from the average slope, the general shape of the solution is also largely independent of α.

[15] Beware that, due to the logarithmic scaling of the y-axes, a larger shaded area does not necessarily imply a larger standard deviation (cf. e.g. the middle row, left half of Fig. 4).



(a) Solution *e1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(b) Solution *e1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(c) Solution *e2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(d) Solution *e2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(e) Solution *e3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(f) Solution *e3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Fig. 5. Temporal development of relative l_2 error for solutions with correct source term (left) and zero source term (right) in extrapolation scenarios. CoSTA is the best method, or among the best methods, in most cases, only being beaten once by each of the other methods. (PBM, DDM, CoSTA).

consistency contributes positively to the trustworthiness of CoSTA, as compared to DDM.

4.2. Experiment 3 – dimensionality reduction

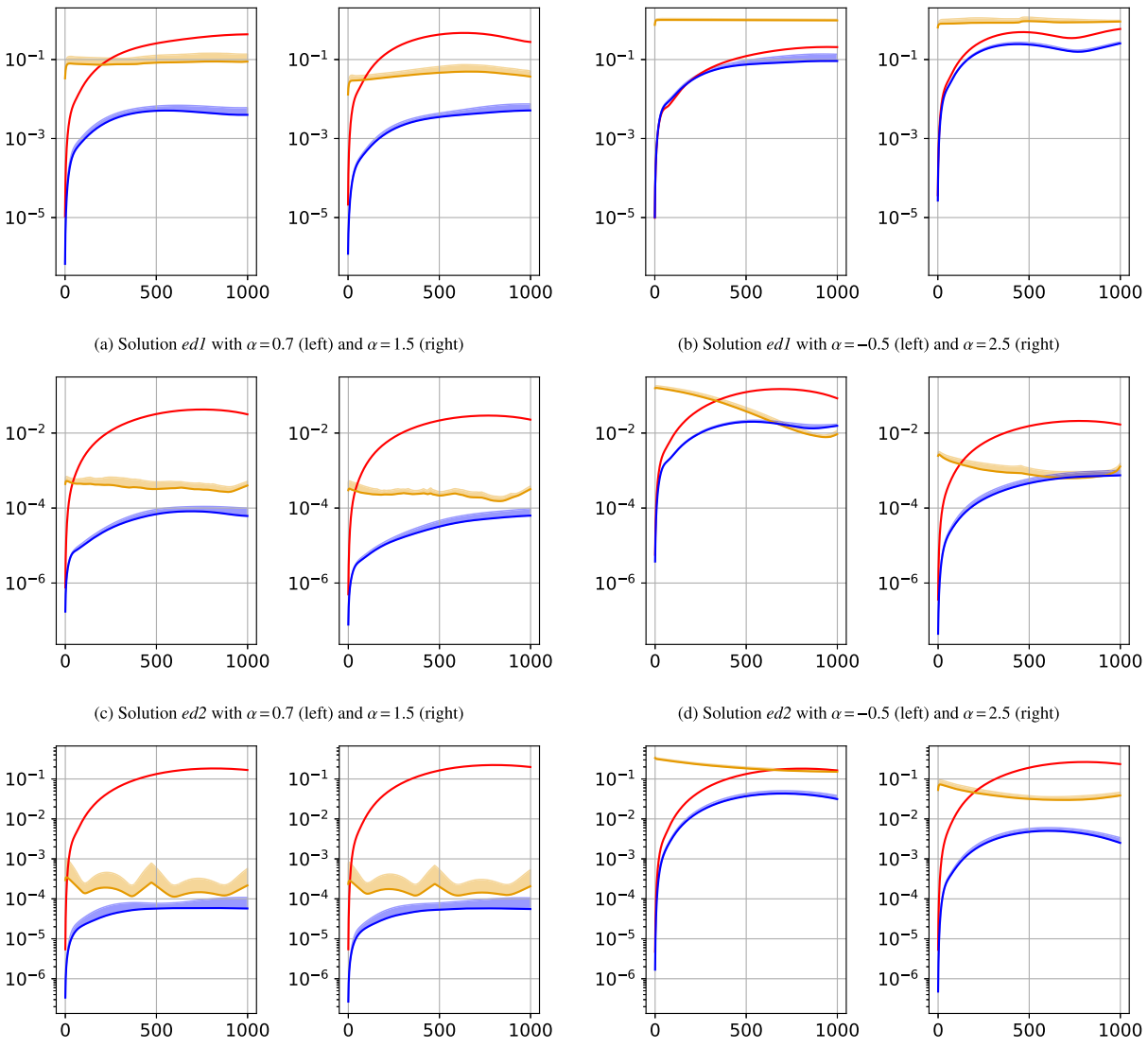
The models' RRMSE for Experiment 3 are presented in Fig. 6. For interpolation cases, CoSTA is consistently more accurate than the other two methods. DDM is in turn much better than PBM in these cases. For extrapolation, the DDM and PBM are alternately more accurate than each other, while CoSTA is clearly the most accurate model for Solutions *ed1* and *ed3*. For Solution *ed2* DDM is most accurate in the last time steps for $\alpha = -0.5$, while DDM and CoSTA are roughly equally accurate for $\alpha = 2.5$. Overall, the results indicate that CoSTA is well-suited for correcting modeling error stemming from dimensionality reduction in PBMs.

4.3. Experiment 4 – linearization of nonlinear elasticity

The models' RRMSE for Experiment 4 are presented in Fig. 7. CoSTA is clearly more accurate than PBM and DDM in all the interpolation cases. DDM is more accurate than PBM for all interpolation cases except Solution *n1*, where PBM is more accurate. In the extrapolation scenarios, CoSTA is significantly more accurate than PBM for Solution *n2* with $\alpha = 2.5$ and Solution *n3*, while in the other cases the two model's are roughly equally accurate. DDM fails completely (100% RRMSE) for Solution *n1*, and is also the least accurate model for Solution *n3*. However, for Solution *n2*, DDM is more accurate than PBM in both cases and also more accurate than CoSTA for $\alpha = -0.5$. The latter case is the only one in this experiment where CoSTA is not the most accurate model.

4.4. Result comparison and further discussion

In this section, we summarize and provide further comments on the results of our numerical experiments. We have discussed four experiments (concerning discretization error, unknown load term, dimensionality reduction and linearization) with 3 parametrized manufactured solutions and 4 α-values combining to a total of 48 test scenarios. In most scenarios, and in all interpolation scenarios, CoSTA has been the most accurate model. Still, there has been some variability in the relative performance of the three models.



(a) Solution *ed1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(b) Solution *ed1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(c) Solution *ed2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(d) Solution *ed2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(e) Solution *ed3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(f) Solution *ed3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Fig. 6. Temporal development of relative l_2 error for dimensional reduced linear elasticity in interpolation scenarios (left) and extrapolation scenarios (right). Observe that CoSTA is more accurate than PBM in all cases, and better than DDM in all except $\alpha = -0.5$ for solution *ed2*. (PBM, DDM, CoSTA).

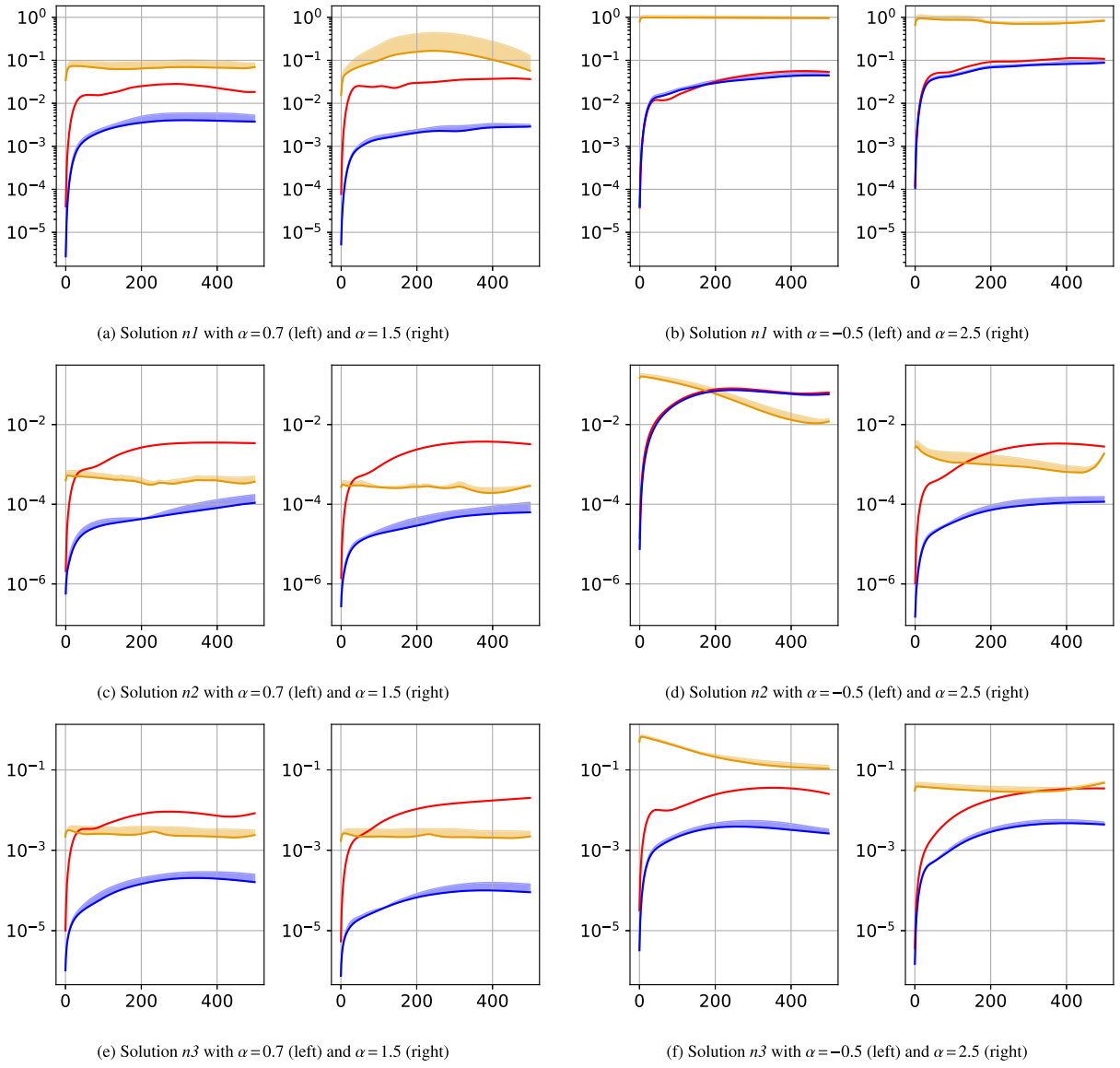
Tables 8, 9 and 10 show the number of times each model was the most accurate (i.e. had the lowest (mean) RRMSE at the final time step), categorized by α-value, experiment and manufactured solution, respectively. Moreover, Fig. 8 shows the number of times a model was more accurate than the other models by at least a factor δ, as a function of significance threshold δ. Similarly, Fig. 9 shows the number of times each model was the least accurate, again as a function of the significance threshold. These tables and figures will form the basis for further discussion below.

First of all, Tables 8, 9 and 10 and Figs. 8 and 9 all support the statement that CoSTA is generally more accurate than PBM and DDM in our experiments.

In Table 8, we observe that $\alpha = -0.5$ is the only α-value for which CoSTA has not been the consistent winner. Given the discussion in Section 1, where PBM was labeled as more generalizable than DDM, it may seem strange that, in the extrapolation scenarios, the DDM is the most accurate model more frequently than the PBM. However, we believe that this can be explained by considering three factors, as explained in the following.

The first factor is that, out of the four scenarios with $\alpha = -0.5$ where the DDM is the most accurate model, three concerns the exponential manufactured solution. As touched upon briefly in Section 4.1, this solution has a qualitatively simple dependence on α which may be well suited for approximation by DNNs, although the underlying mechanism for this is unknown. The fourth case with $\alpha = -0.5$ where DDM "wins", is for Solution *e1* in Experiment 2. Here, although DDM is the most accurate, its RRMSE at the final time step is still close to 100%, and its predictions consequently worthless from a practical standpoint.

Secondly, across all 48 scenarios, the accuracy of DDM is more often than not higher than that of PBM (cf. Figs. 8 and 9). That is, on average, our DDM is more accurate than our PBM for the modeling problems considered in our experiments. Thus, DDM can afford to lose some accuracy due to poor generalizability and still be more accurate than PBM. Thirdly, the accuracy of DDM is worse in all extrapolation scenarios than in the corresponding interpolation scenarios. PBM does not exhibit a similar trend. Considering these factors, we believe that our results do not contradict the statement that PBM generalizes better than DDM, but rather support it.



(a) Solution *n1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(b) Solution *n1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(c) Solution *n2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(d) Solution *n2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(e) Solution *n3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

(f) Solution *n3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Fig. 7. Temporal development of relative l_2 error for nonlinear elasticity in interpolation scenarios (left) and extrapolation scenarios (right). While CoSTA is most accurate in all interpolation cases, the extrapolation results vary more. (PBM, DDM, CoSTA).

Table 8
Overview of number times each method was the most accurate at final time step, for each value of α and in total.

α	PBM	DDM	CoSTA	Total
−0.5	1	4	7	12
0.7	0	0	12	12
1.5	0	0	12	12
2.5	0	1	11	12
Total	1	5	42	48

Table 9
Overview of number times each method was the most accurate at final time step, for each type of extra modeling error.

Modeling error	PBM	DDM	CoSTA	Total
None	1	0	11	12
Zero source term	0	3	9	12
Dimensionality reduction	0	1	11	12
Linearization	0	1	11	12

Table 10
Overview of number times each method was the most accurate at final time step, for each manufactured solution.

Solution	PBM	DDM	CoSTA	Total
e1/ed1/n1 (sinusoidal)	1	2	13	16
e2/ed2/n2 (exponential)	0	3	13	16
e3/ed3/n3 (polynomial)	0	0	16	16

Both DDM and CoSTA have, at times, exhibited noticeable variability in their predictions. (DDM more so than CoSTA.) Consistency is important for trustworthiness, and to avoid having to run several instances of the same model, and should therefore be taken into account when comparing the methods. One way of penalizing the variance when summarizing the results is to compare final mean error plus a standard deviation. The dotted lines in Figs. 8 and 9 show the results of such a penalization. We see that CoSTA and DDM both perform a bit worse, and PBM a bit better, with this method of evaluation. But the difference between the dotted and solid lines is quite insignificant. In

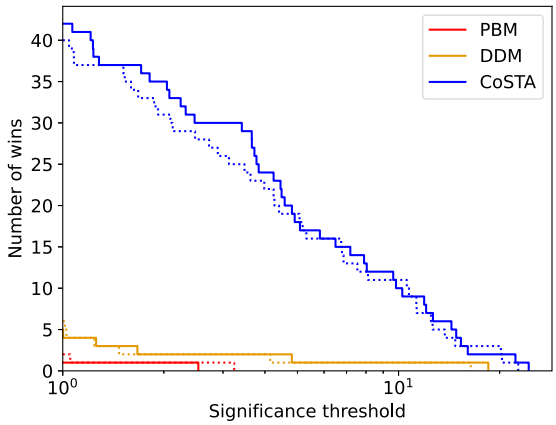


Fig. 8. Overview of number of times each method was the most accurate at the final time step, at various degrees of significance. For a value x_1 on the x-axis, the height of a curve is the number of times the method was better (had lower error) than both the other methods by a factor of at least x_1. E.g. by observing the blue line at 10^1, we see that in about 10 of the 48 tests, CoSTA won by a factor of at least 10^1 (meaning both PBM and DDM had an error of more than 10 times that of CoSTA). The solid line is the results of evaluating the means (of the CoSTA and DDM initializations), while the dotted line is from evaluating the mean + one standard deviation, as a way of penalizing inconsistency.

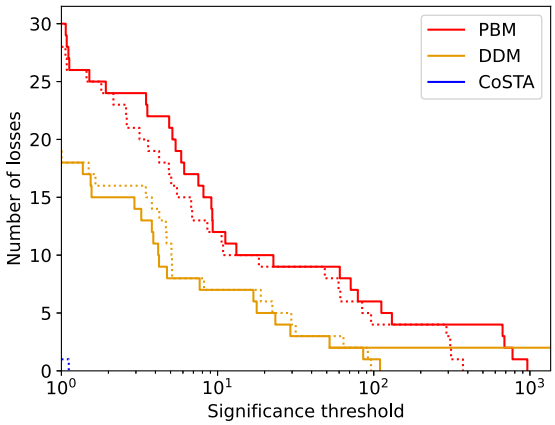


Fig. 9. Overview of number of times each method was the *least* accurate at the final time step, at various degrees of significance. For a value x_1 on the x-axis, the height of a curve is the number of times a method was worse than both the other methods by a factor of at least x_1. The solid line is the results of evaluating the means (of the CoSTA and DDM initializations), while the dotted line is from evaluating the mean + one standard deviation, as a way of penalizing inconsistency. We see that CoSTA was never the most inaccurate method, and barely the worst once when adding the standard deviation.

other words, the difference in accuracy between the methods, are generally much larger than the difference in accuracy between differently initialized instances of the same model.[16]

The above discussion on the scenarios where DDM is the most accurate model can shed light on the question of when to use CoSTA. It seems that, if the variable to predict is in some sense mathematically well-behaved in space and time, predicting a corrective source term is not necessarily simpler than predicting the solution directly. This appears to be especially true if (1) the qualitative behavior of the system is similar across the relevant scenarios, (2) pure PBM offers inaccurate (or worse, qualitatively incorrect) predictions. In such scenarios, and for simpler solutions, DDM might outperform CoSTA.

On the opposite end of the spectrum, if a PBM of sufficient accuracy and acceptable computational cost is available, there is no need to use CoSTA. If the relevant scenarios are very different from those used to train CoSTA's DDM-component, CoSTA might even perform worse than pure PBM, as seen for Solution *e1* with $\alpha = -0.5$ in Experiment 1 (cf. Fig. 5).

We highlight that the possible limitations described above are rather weak. Indeed, in the vast majority of cases we have considered, the CoSTA model is significantly more accurate than the PBM and DDM models — sometimes by more than one order of magnitude. Moreover, CoSTA's superior accuracy persists across a selection of different, commonly encountered error sources: discretization error, unknown physics, dimensionality reduction and linearization. Overall, we conclude that CoSTA is the superior model in our numerical experiments.

5. Conclusion and future work

In our research endeavors focused on elasticity modeling, we embarked on a series of numerical experiments aimed at showcasing the effectiveness of CoSTA in amalgamating physics-based modeling (PBM) and data-driven modeling (DDM) techniques into a comprehensive unified model. The results of these experiments, as detailed in this study, have revealed the considerable potential of CoSTA in bridging the gap between traditional PBM and DDM methodologies. More precisely:

- When it comes to evaluating the accuracy of our CoSTA model, our findings present a compelling case. Across a total of 48 distinct experimental scenarios, our CoSTA model consistently outperformed its individual PBM and DDM counterparts in 42 of these cases. In particular, in numerous instances, the superiority of CoSTA was not marginal but rather substantial, surpassing its constituent models by more than one order of magnitude. This outstanding performance underscores the robustness and adaptability of CoSTA to address a wide range of engineering applications and error sources.
- The experiments carried out in this study encompassed a diverse set of error sources frequently encountered in engineering contexts. These included the discretization error, the error stemming from unknown physics, the dimensionality reduction error, and the linearization error. Our investigations reaffirmed earlier observations from related research [29,34], demonstrating that CoSTA excels at mitigating modeling errors attributed to discretization and errors related to the incorporation of unknown physics. Importantly, our work stands out as the first comprehensive demonstration of CoSTA's capabilities in correcting modeling errors arising from dimensionality reduction and linearization within PBMs.
- Crucially, our findings also highlight the robustness of CoSTA. Even when accounting for randomness introduced by deep neural network (DNN) initialization and the optimization process through stochastic gradient descent, CoSTA's superiority remained evident and persistent.
- Furthermore, it is worth noting a significant milestone achieved in this study. This work marks the first successful application of CoSTA in the context of PBMs based on the finite element method, showcasing the versatility of CoSTA across diverse modeling techniques. Furthermore, to foster transparency and promote further research in this area, we have made the implementation of our CoSTA model open source [48], thus allowing the broader scientific community to benefit from and build on our findings.

[16] This is easy to see in the error plots — while the blue shaded area showing the standard deviation of CoSTA error at times is relatively thick, the distance to the other lines is usually considerably larger. Remember also that the thickness of the shaded area is also logarithmically scaled, so when the mean prediction is very accurate, a thick line does not indicate a big variance. For example, in Fig. 7(e), despite the variance, CoSTA is consistently much more accurate than the other methods.

In summary, our research underscores the compelling potential of CoSTA as a unifying framework to combine PBM and DDM techniques, demonstrating its exceptional accuracy, adaptability, and robustness across a variety of engineering applications and error sources. However, we still foresee improvements to the CoSTA method in the following directions:

- Addressing the black-box nature through the use of symbolic regression: While CoSTA has shown promising results, addressing its black-box nature is essential for a deeper understanding and broader applicability. Symbolic regression techniques, such as genetic programming or neural-symbolic methods, can be explored to extract interpretable mathematical expressions from the DDM model. This would help uncover the underlying physics and relationships captured by CoSTA.
- Exploiting the correlation in time: To enhance CoSTA's ability to model transient systems and temporal dependencies, recurrent neural networks (RNNs) like Long Short-Term Memory (LSTM) or transformer-based architectures can be incorporated. These models excel at capturing sequential patterns and could be especially useful for systems with complex temporal dynamics.
- Dealing with noise: To make CoSTA more robust in practical scenarios, incorporating a denoising step into the pipeline is crucial. This step could involve training CoSTA on data with synthetic noise to improve its resilience to real-world noise. Additionally, exploring advanced denoising techniques, such as autoencoders can help enhance the model's performance in noisy environments.
- Using real-time measurements: To bridge the gap between laboratory experiments and real-world applications, it is essential to consider the use of real-time measurements. Building a latent space Reduced Order Model (ROM) that incorporates CoSTA as a component can facilitate real-time predictions and control. This would involve integrating CoSTA with online sensor data and dynamic system simulations to make it practical for real-world applications.

By addressing these four points, CoSTA's capabilities, interpretability, robustness, and real-world applicability can be further enhanced, ultimately advancing the field of computational modeling and simulation for various physical phenomena.

CRediT authorship contribution statement

Sondre Sørbø: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Sindre Stenen Blakseth:** Writing – review & editing, Writing – original draft. **Adil Rasheed:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Funding acquisition, Conceptualization. **Trond Kvamsdal:** Writing – original draft, Supervision, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Omer San:** Writing – review & editing, Investigation, Funding acquisition, Formal analysis, Conceptualization.

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

This publication has been prepared as part of NorthWind (Norwegian Research Centre on Wind Energy) co-financed by the Research Council of Norway, industry and research partners. Read more at www.northwindresearch.no (grant no. 321954). O.S. gratefully acknowledges the National Science Foundation support (DMS-2012255).

References

[1] H. Wei, S. Zhao, Q. Rong, H. Bao, Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods, Int. J. Heat Mass Transfer 127 (2018) 908–916.
[2] P. Baldi, K.T. Bauer, C. Eng, P. Sadowski, D. Whiteson, Jet substructure classification in high-energy physics with deep neural networks, Phys. Rev. D 93 (2016) 094034.
[3] G. Ibarra-Berastegi, J. Saénz, G. Esnaola, A. Ezcurra, A. Ulazia, Short-term forecasting of the wave energy flux: Analogues, random forests, and physics-based models, Ocean Eng. 104 (2015) 530–539.
[4] X. Jia, J.D. Willard, A. Karpatne, J.S. Read, J.A. Zwart, M.S. Steinbach, V. Kumar, Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles, ACM/IMS Trans. Data Sci. 2 (2021) 1–26.
[5] O. San, A. Rasheed, T. Kvamsdal, Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution, GAMM-Mitt. 44 (2) (2021) e202100007.
[6] A. Rasheed, O. San, T. Kvamsdal, Digital twin: Values, challenges and enablers from a modeling perspective, IEEE Access 8 (2020) 21980–22012.
[7] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating physics-based modeling with machine learning: A survey, 2020, arXiv preprint arXiv:2003.04919.
[8] B. Amos, J.Z. Kolter, OptNet: Differentiable Optimization as a Layer in Neural Networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 136–145.
[9] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, J.Z. Kolter, End-to-end differentiable physics for learning and control, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018, p. 1.
[10] Y. Yu, H. Yao, Y. Liu, Structural dynamics simulation using a novel physics-guided machine learning method, Eng. Appl. Artif. Intell. 96 (2020) 103947.
[11] A. Quarteroni, G. Rozza, Reduced Order Methods for Modeling and Computational Reduction, vol. 9, Springer, New York, 2014.
[12] E. Fonn, H.v. Brummelen, T. Kvamsdal, A. Rasheed, Fast divergence-conforming reduced basis methods for steady Navier–Stokes flow, Comput. Methods Appl. Mech. Engrg. 346 (2019) 486–512.
[13] S. Pawar, S.E. Ahmed, O. San, A. Rasheed, An evolve-then-correct reduced order model for hidden fluid dynamics, Mathematics 8 (4) (2020) 570.
[14] S. Pawar, S.E. Ahmed, O. San, A. Rasheed, Data-driven recovery of hidden physics in reduced order modeling of fluid flows, Phys. Fluids 32 (2020) 036602.
[15] M. Raissi, P. Perdikaris, G.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.
[16] N. Zobeiry, K.D. Humfeld, A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications, Eng. Appl. Artif. Intell. 101 (2021) 104232.
[17] F. Arnold, R. King, State-space modeling for control based on physics-informed neural networks, Eng. Appl. Artif. Intell. 101 (2021) 104195.
[18] S. Shen, H. Lu, M. Sadoughi, C. Hu, V. Nemani, A. Thelen, K. Webster, M. Darr, J. Sidon, S. Kenny, A physics-informed deep learning approach for bearing fault detection, Eng. Appl. Artif. Intell. 103 (2021) 104295.
[19] M.M. Billah, A.I. Khan, J. Liu, P. Dutta, Physics-informed deep neural network for inverse heat transfer problems in materials, Mater. Today Commun. 35 (2023) 106336.
[20] Z. Sun, H. Du, C. Miao, Q. Hou, A physics-informed neural network based simulation tool for reacting flow with multicomponent reactants, Adv. Eng. Softw. 185 (2023) 103525.
[21] J. Halamka, M. Bartošák, M. Španiel, Using hybrid physics-informed neural networks to predict lifetime under multiaxial fatigue loading, Eng. Fract. Mech. 289 (2023) 109351.
[22] K. Champion, B. Lusch, J.N. Kutz, S.L. Brunton, Data-driven discovery of coordinates and governing equations, Proc. Natl. Acad. Sci. 116 (45) (2019) 22445–22451.
[23] H. Vaddireddy, A. Rasheed, A.E. Staples, O. San, Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensors, Phys. Fluids Editor's pick 32 (2020) 015113.
[24] H. Zhang, X. Liu, G. Zhang, Y. Zhu, S. Li, Q. Qian, D. Dai, R. Che, T. Xu, Deriving equation from data via knowledge discovery and machine learning: A study of Young's modulus of Ti-Nb alloys, Comput. Mater. Sci. 228 (2023) 112349.

[25] K.A. Meyer, F. Ekre, Thermodynamically consistent neural network plasticity modeling and discovery of evolution laws, J. Mech. Phys. Solids 180 (2023) 105416.
[26] S. Pawar, O. San, B. Aksoylu, A. Rasheed, T. Kvamsdal, Physics guided machine learning using simplified theories, Phys. Fluids 33 (1) (2021) 011701.
[27] H. Robinson, S. Pawar, A. Rasheed, O. San, Physics guided neural networks for modelling of non-linear dynamics, Neural Netw. 154 (2022) 333–345.
[28] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R.T. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, J. Dong, B. Amos, M. Mukadam, Theseus: A library for differentiable nonlinear optimization, Adv. Neural Inf. Process. Syst. (2022).
[29] S.S. Blakseth, A. Rasheed, T. Kvamsdal, O. San, Deep neural network enabled corrective source term approach to hybrid analysis and modeling, Neural Netw. 146 (2022) 181–199.
[30] R. Maulik, O. San, A. Rasheed, P. Vedula, Sub-grid modelling for two-dimensional turbulence using neural networks, J. Fluid Mech. 858 (2019) 122–144.
[31] S. Pawar, O. San, A. Rasheed, P. Vedula, A priori analysis on deep learning of subgrid-scale parameterizations for Kraichnan turbulence, Theor. Comput. Fluid Dyn. 34 (2020) 429–455.
[32] H. Robinson, E. Lundby, A. Rasheed, J.T. Gravdahl, Deep learning assisted physics-based modeling of aluminum extraction process, Eng. Appl. Artif. Intell. 125 (2023) 106623.
[33] S.S. Blakseth, Introducing CoSTA: A Deep Neural Network Enabled Approach to Improving Physics-Based Numerical Simulations, NTNU, 2021.
[34] S.S. Blakseth, A. Rasheed, T. Kvamsdal, O. San, Combining physics-based and data-driven techniques for reliable hybrid analysis and modeling using the corrective source term approach, Appl. Soft Comput. 128 (2022) 109533.
[35] W.S. Slaughter, The Linearized Theory of Elasticity, Birkhäuser, Boston, MA, 2002.
[36] F. Irgens, Continuum Mechanics, Springer Berlin Heidelberg, 2008.
[37] A. Quarteroni, Numerical Models for Differential Problems, third ed., Springer International Publishing, 2017.
[38] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, third ed., Springer, New York, 2008.
[39] R. Courant, Variational methods for the solution of problems of equilibrium and vibrations, Bull. Amer. Math. Soc. 49 (1943) 1–23.
[40] G.V. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989) 303–314.
[41] K. Hornik, M.B. Stinchcombe, H.L. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[42] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867.
[43] I. Nusrat, S.-B. Jang, A comparison of regularization techniques in deep neural networks, Symmetry 10 (11) (2018).
[44] P.J. Roache, Code verification by the method of manufactured solutions, J. Fluids Eng. 124 (1) (2001) 4–10.
[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org.
[46] A.L. Maas, A.Y. Hannum, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proceedings of the 30th International Conference on Machine Learning, Vol. 28, 2013, p. 3.
[47] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[48] S. Sørbø, FEM_CoSTA, 2022, GitHub Repository https://github.com/sondsorb/FEM_CoSTA.