



# Deep neural network enabled corrective source term approach to hybrid analysis and modeling

Sindre Stenen Blakseth<sup>a</sup>, Adil Rasheed<sup>b,d,\*</sup>, Trond Kvamsdal<sup>c,d</sup>, Omer San<sup>e</sup>

<sup>a</sup> Department of Physics, Norwegian University of Science and Technology, Norway

<sup>b</sup> Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway

<sup>c</sup> Department of Mathematical Sciences, Norwegian University of Science and Technology, Norway

<sup>d</sup> Mathematics and Cybernetics, SINTEF Digital, Norway

<sup>e</sup> School of Mechanical and Aerospace Engineering, Oklahoma State University, United States of America

## ARTICLE INFO

### Article history:

Received 24 May 2021

Received in revised form 14 November 2021

Accepted 19 November 2021

Available online 26 November 2021

### Keywords:

Deep neural networks

Digital twins

Explainable AI

Hybrid analysis and modeling

Physics-based modeling

Corrective source term approach (CoSTA)

## ABSTRACT

In this work, we introduce, justify and demonstrate the Corrective Source Term Approach (CoSTA)—a novel approach to Hybrid Analysis and Modeling (HAM). The objective of HAM is to combine physics-based modeling (PBM) and data-driven modeling (DDM) to create generalizable, trustworthy, accurate, computationally efficient and self-evolving models. CoSTA achieves this objective by augmenting the governing equation of a PBM model with a corrective source term generated using a deep neural network. In a series of numerical experiments on one-dimensional heat diffusion, CoSTA is found to outperform comparable DDM and PBM models in terms of accuracy – often reducing predictive errors by several orders of magnitude – while also generalizing better than pure DDM. Due to its flexible but solid theoretical foundation, CoSTA provides a modular framework for leveraging novel developments within both PBM and DDM. Its theoretical foundation also ensures that CoSTA can be used to model any system governed by (deterministic) partial differential equations. Moreover, CoSTA facilitates interpretation of the DNN-generated source term within the context of PBM, which results in improved explainability of the DNN. These factors make CoSTA a potential door-opener for data-driven techniques to enter high-stakes applications previously reserved for pure PBM.

© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The recent wave of digitalization has given a push to emerging technologies like digital twins. A digital twin is defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making (Rasheed et al., 2020). Paramount to digital twins' success is the level of physical realism that can be instilled into them. In this regard, as noticed by Rasheed et al. (2020), modeling plays an important role. The exact requirements of the modeling depend on the digital twin's capability level, which is generally categorized on a scale from 0 to 5 (0-standalone, 1-descriptive, 2-diagnostic, 3-predictive, 4-prescriptive, 5-autonomous) (see Fig. 1).

Although a digital twin offers huge potential in many industries, adaptation of the digital twin technology has been stagnated since its inception, mainly due to the lack of methodological

works. To leverage asset-twin technologies, Kapteyn et al. (2021) proposed a unifying mathematical foundation that draws from probabilistic graphical models and dynamical system theory. The role of surrogate models in the development of digital twin has also been emphasized by Hartmann et al. (2018) and Chakraborty et al. (2021). There is, however, more work to be done to bring physical realism into digital twins, as many industrial and scientific applications steadily migrate from sparse data to big data regimes. With this in mind, San et al. (2021) identified that there are at least four modeling characteristics of utmost importance; *generalizability*, *trustworthiness*, *computational efficiency and accuracy*, and *self-adaptation*. A model's generalizability refers to its ability to solve a wide variety of problems without any problem-specific fine-tuning. Trustworthiness refers to the extent to which a model is explainable, while computational efficiency and accuracy refers to the model's ability to make real-time predictions that match ground truth as closely as possible. Lastly, a model is self-adapting if it can learn and evolve when new situations are encountered. Until recently, most modeling approaches could be categorized as either physics-based modeling (PBM) or data-driven modeling (DDM). These categories are briefly explained below:

\* Corresponding author at: Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway.

E-mail addresses: [sindre.blakseth@sintef.no](mailto:sindre.blakseth@sintef.no) (S.S. Blakseth), [adil.rasheed@ntnu.no](mailto:adil.rasheed@ntnu.no) (A. Rasheed), [trond.kvamsdal@ntnu.no](mailto:trond.kvamsdal@ntnu.no) (T. Kvamsdal), [osan@okstate.edu](mailto:osan@okstate.edu) (O. San).

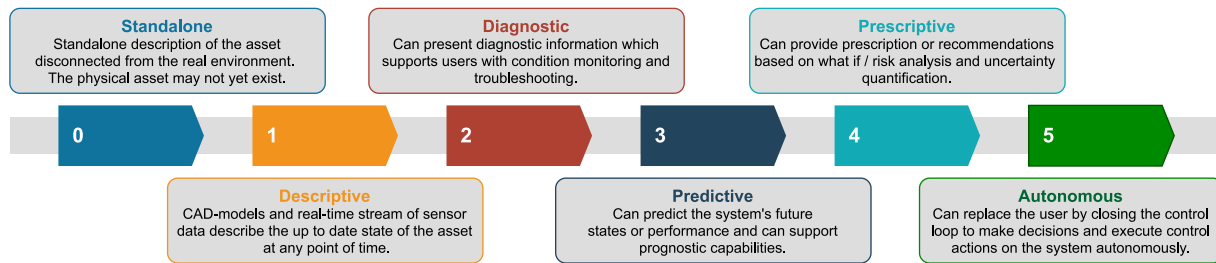


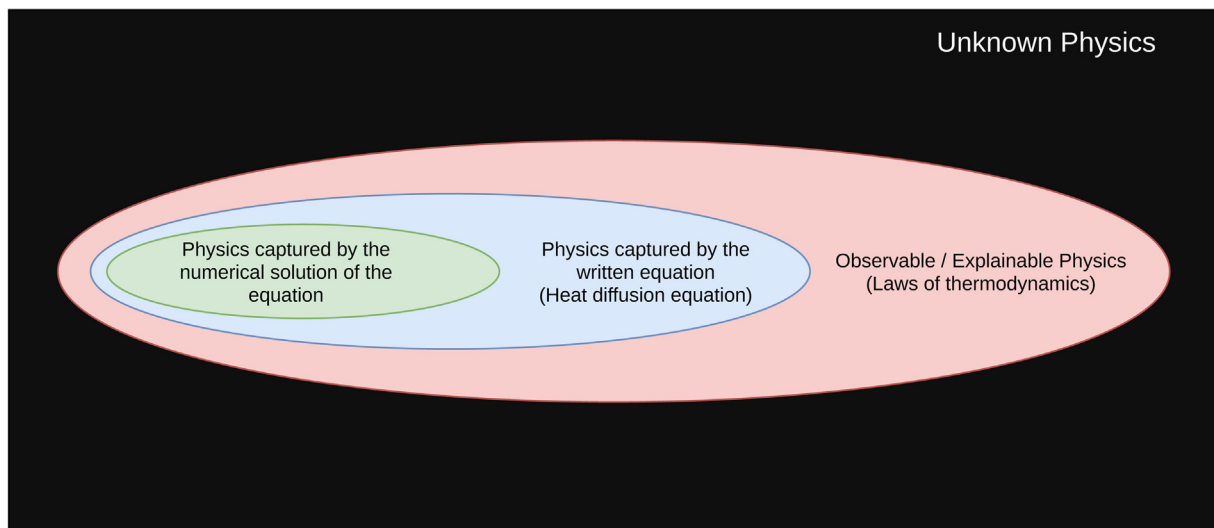
Fig. 1. The capability levels of DTs on a scale from 0 to 5.  
Source: Adapted from San et al. (2021).

**Physics-based modeling:** For any real-world system, PBM seeks to explain the system's behavior using existing knowledge of observable and explainable physics (illustrated by the pink ellipse in Fig. 2(a)). As such, PBM is ignorant of any unknown physics (illustrated by the black background) which, e.g., cannot be observed directly or is inexplicable. Given the first principles of known physics, PBM requires the derivation of one or more governing equations for the system. This derivation might involve making some assumptions, such that only partial physics (the blue ellipse in Fig. 2(a)) are accounted for by the governing equations. More often than not, these equations are difficult to solve analytically. Therefore, to solve them numerically (in a reasonable amount of time), we make further assumptions, resulting in further loss of physics. Thus, the resolved physics (the green ellipse in Fig. 2(a)) is generally only a part of the full physics governing the system. PBM has been used extensively for engineering applications like blood flow (Taylor et al., 1998), heat and mass transfer (Liu et al., 2021) and flow around wind turbines (Siddiqui et al., 2019) to name a few. A good overview of PBM in the context of digital twins can be found in Rasheed et al. (2020) and San et al. (2021). In most of such applications, PBMs tend to be computationally demanding. They are also typically static, meaning that they do not automatically adapt to new scenarios and hence can be inaccurate. Despite these limitations, PBM is attractive due to its sound first-principles foundation which yields great interpretability and generalizability.

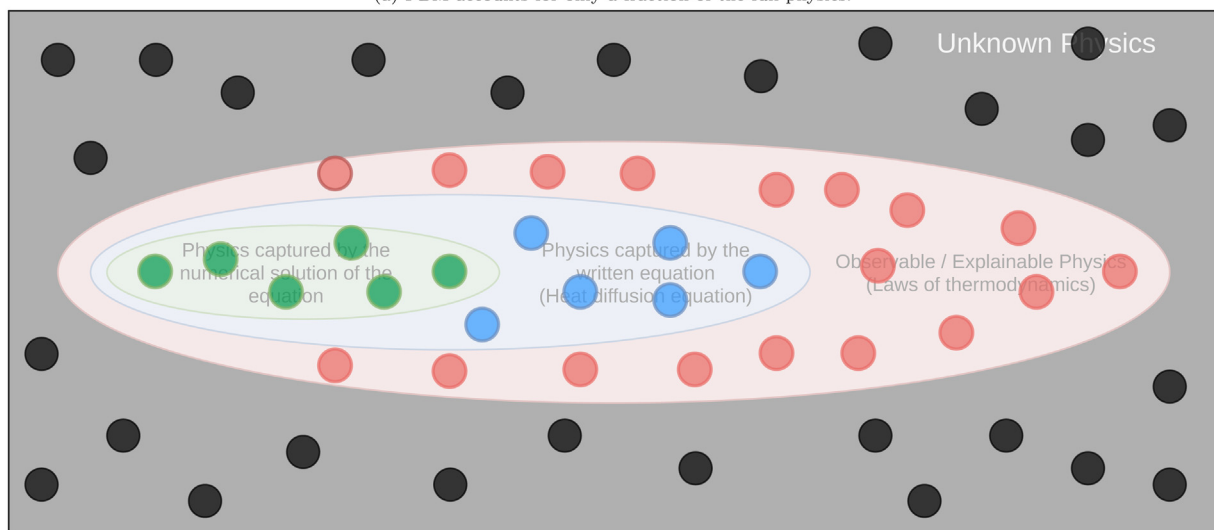
**Data-driven modeling:** With the abundant supply of big data facilitated through, for example, cost-effective sensors, open-source cutting-edge and easy-to-use machine learning libraries, cheap computational infrastructure, and high-quality, readily available training resources, DDM has become very popular. Compared to the PBM approach, DDM thrives on the assumption that data is a manifestation of both known and unknown physics, and hence, when trained with an ample amount of data, DDM will learn the full physics on their own. Data-driven techniques, in particular those involving deep neural networks (DNN), have started achieving human-level performance in several tasks that were, until recently, considered impossible for computers. Notable examples include image classification (Szegedy et al., 2016), dimensionality reduction (Hinton & Salakhutdinov, 2006), medical treatment (Liu et al., 2019) and smart agriculture (Bu & Wang, 2019). More recent applications of DDM include tropical cyclone intensity estimation (Lee et al., 2021), speech recognition (a comprehensive review can be found in Bai and Zhang (2021)), learning of complex nonlinear dynamics from data (Afebu et al., 2021) and active noise control (Zhang & Wang, 2021). Some of the advantages of DDM are their inherent online learning capability, high computational efficiency for inference, and accuracy even for very challenging problems (assuming the training, validation, and test data are prepared properly). However, acceptability of DDM in high-stake applications has been fairly limited due to their data-hungry and black-box nature, poor generalizability, inherent bias, and lack of a robust theory for model stability analysis. In

fact, the numerous vulnerabilities of DNN have been highlighted in several recent works (Akhtar & Mian, 2018; Xu et al., 2019; Yuan et al., 2017).

Due to the challenges described above, it can be concluded that neither PBM nor DDM is ideal for usage in a digital twin context as neither satisfies all the four modeling characteristics identified in San et al. (2021). Fortunately, a new paradigm in modeling called Hybrid Analysis and Modeling (HAM) – which combines the generalizability, interpretability, robust foundation, and understanding of PBM with the accuracy, computational efficiency, and automatic pattern-identification capabilities of advanced DDM, in particular DNNs – is emerging. In their recent surveys, Willard et al. (2020) and San et al. (2021) provide comprehensive overviews of techniques for integrating DDM with PBM. Most of the hybridization techniques lie in one of the categories of reduced order modeling, physics-guided machine learning (PGML), physics-informed neural network (PINN) or data-driven physics discovery using sparse or symbolic regression. Reduced order models (ROMs) have been proposed as a way of accelerating high-fidelity simulators by several orders of magnitude. In a ROM (Fonn et al., 2019), complex partial differential equations are projected onto a reduced dimensional space based on an unsupervised algorithm called the proper orthogonal decomposition (more commonly known as the principal component analysis) of the offline high fidelity simulation results resulting in a set of ordinary differential equations (ODEs). One advantage of this method is that many terms in the resulting ODEs can be computed offline using the data, and hence, what remains in an online phase is a simple forward integration of the ODEs in time, which can be very fast. These models however, do not perform well when physics is either missing or get lost during the dimensionality reduction. In the PGML approach introduced by Pawar et al. (2021), partially known physics or knowledge is injected at an intermediate layer in a DNN. Such injection has been shown to improve the accuracy and reduce the uncertainty of the trained model. However, the PGML models still do not generalize well to extrapolation scenarios. Within a PINN framework (Raissi et al., 2019), the commonly used mean squared error cost function of the DNN is regularized with the residual of the equation describing the physical laws that should be satisfied. This kind of regularization can pose a challenge during the optimization step because of the increase in the complexity of the cost function. Finally, sparse regression based on  $l_1$  regularization and symbolic regression based on gene expression programming have been shown to be very effective in discovering hidden or partially known physics directly from data (Vaddireddy et al., 2020). The data-driven discovered physics is then added to the PBM to improve their predictions. Notable work using this approach can be found in Brunton et al. (2016). One of the limitations of this class of method is that, in the case of sparse regression, additional features are required to be handcrafted, while in the case of symbolic regression, the resulting models are often unstable and hence might not be fit for interpretation.



(a) PBM accounts for only a fraction of the full physics.



(b) In DDM, data is a manifestation of all physics (known, unknown, resolved, ignored).

**Fig. 2.** Physics based modeling vs data driven modeling. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

From a careful analysis of the work cited here, it is clear that the HAM approach has the potential to fulfill the four desirable modeling characteristics described earlier in this section. However, as discussed above, earlier approaches to HAM have some limitations. In the current article, we present the *Corrective Source Term Approach* (CoSTA) to HAM. The novelty of the work from a theoretical perspective, is the development of a sound mathematical foundation of the CoSTA approach to augment the governing equation of a PBM describing partial physics with a DNN-generated corrective source term that takes into account the remaining unknown/ignored physics. By doing so, the resulting model retains the generalizability and interpretability of the PBM while exploiting DDM to make the predictions more accurate by modeling the unknown physics reflected in the data. Explainable AI is often attributed to the enhancing processes in which the results of the machine learning models and solutions can be better understood by humans. In our view, our proposed CoSTA approach can be classified as a new explainable AI approach while synthesizing a modular framework between black-box DDM and PBM. From a practical perspective, the superiority of CoSTA in terms of accuracy and generalizability is quantitatively demonstrated by comparing its results against

those of pure PBM and DDM in a series of numerical experiments concerning heat diffusion.

In Section 2, we start with the rationale behind the development of CoSTA. We then continue with a presentation of our chosen PBM (Section 2.2) and DDM (Section 2.3), before we explain how the PBM and a DNN can be combined using CoSTA (Section 2.4). We also provide some background on the method of manufactured solutions (Section 2.5). Section 3 is devoted to explaining our experimental setup—including the manufactured solutions considered, our DNN architecture and hyperparameter choices, and our data generation, training and testing procedures. Our experimental results are presented and discussed in Section 4 before the article is concluded in Section 5 with a brief summary and an outlook on future work.

## 2. Theory

In this section we present the rationale behind the CoSTA approach, followed by an overview of the PBM, DDM and HAM models used in our numerical experiments. The section concludes with a brief introduction to the method of manufactured solutions.

## 2.1. The rationale behind the CoSTA approach

In this subsection, we present a mathematical foundation for the proposed CoSTA approach. Assume that we are aiming to solve a problem on a domain  $\Omega$  (with boundary  $\partial\Omega$  and outward pointing unit normal  $n$ ) that can be represented by a linear partial differential equation (PDE) defined as follows:

$$\mathcal{L}u = f \quad \text{in } \Omega \quad (1)$$

$$u = g_D \quad \text{on } \partial\Omega_D \quad (2)$$

$$\mathcal{N}u = g_N \quad \text{on } \partial\Omega_N \quad (3)$$

Here  $\mathcal{L}$  is a linear<sup>1</sup> differential operator (e.g.  $\mathcal{L} = -\nabla^2$  for Poisson problems),  $u$  is the unknown (e.g. temperature for heat diffusion problems),  $f$  is the (true) source term (e.g. a heat source/sink in heat diffusion problems),  $g_D$  is the prescribed Dirichlet boundary condition along the boundary section  $\partial\Omega_D$ , and  $\mathcal{N}$  is the differential operator related to the Neumann boundary condition (e.g.  $\partial u/\partial n$  for heat diffusion problems) with prescribed value  $g_N$  (e.g. heat flux) along the boundary section  $\partial\Omega_N$ . We assume that  $\partial\Omega_D$  and  $\partial\Omega_N$  cover the whole boundary  $\partial\Omega$  without overlapping each other.

We will now address different cases of uncertainties/errors/lack of information in the abstract PDE defined above. We will, in general, let  $\tilde{u}$  denote an analytical solution to a *perturbed* version of the PDE problem defined in Eqs. (1)–(3). Furthermore, we denote *numerically* computed solutions of the original PDE and its perturbation as  $u_{\text{num}}$  and  $\tilde{u}_{\text{num}}$ , respectively. Here the subscript “num” indicates the finite resolution of the numerical method (e.g. finite difference method (FDM), finite volume method (FVM) or finite element method (FEM)).

Let the error between the two analytical solutions  $u$  and  $\tilde{u}$  be denoted  $\tilde{e}$ , i.e. we have

$$\tilde{e} = u - \tilde{u}. \quad (4)$$

We define the corresponding residual  $\tilde{r}$  as follows:

$$\tilde{r} = f - \mathcal{L}\tilde{u} \quad (5)$$

$$= \mathcal{L}u - \mathcal{L}\tilde{u} \quad (6)$$

$$= \mathcal{L}\tilde{e} \quad (7)$$

Notice that there is a unique relationship between the error in the analytical solution of the perturbed PDE  $\tilde{u}$  and the residual obtained by inserting this solution into the (true) original PDE. The CoSTA approach utilizes this relationship, as illustrated for different cases 1–4 below. Cases 1 and 2 concern possible sources of error in the governing PDE itself, while Case 3 is the case when the governing PDE is known without error but cannot be solved analytically. Combinations of Cases 1–3 are treated as Case 4.

*Case 1: Inaccurate source term or boundary conditions:* In many real-world problems, the source term  $f$  (e.g. describing internal heat generation) may not be known exactly.

Let the inaccurate source term be denoted  $\tilde{f}$  and assume that we are able to compute exactly the corresponding PDE such that:

$$\mathcal{L}\tilde{u} = \tilde{f} \quad \text{in } \Omega. \quad (8)$$

Assuming that we know  $u$  (e.g. by measurements or analytical solution), we can add a corrective source term  $\tilde{r}$  to compute an improved (analytical) solution denoted  $u_{\text{CoSTA}}$ :

$$\mathcal{L}u_{\text{CoSTA}} = \mathcal{L}\tilde{u} + (\mathcal{L}u - \mathcal{L}\tilde{u}) \quad (9)$$

$$= \tilde{f} + \mathcal{L}\tilde{e} \quad (10)$$

$$= \tilde{f} + \tilde{r} \quad (11)$$

If we are able to evaluate  $\mathcal{L}u$  for a given solution  $u$ , we get the following relationship from the equations above<sup>2</sup>:

$$\mathcal{L}u_{\text{CoSTA}} = \mathcal{L}u \quad (12)$$

Thus, by the herein developed CoSTA, we may retain the true analytical solution without any modeling error caused by the inaccurate source term  $\tilde{f}$ . The main point to be observed here is that inaccuracies in the source term can be corrected for by computing the residual from measured (or manufactured) solutions.

Suppose now that, instead of having an inaccurate source term  $\tilde{f}$ , we have an inaccurate Dirichlet condition  $\tilde{g}_D$  (e.g. inaccurate surface temperature) or an inaccurate Neumann condition  $\tilde{g}_N$  (e.g. unknown heat flux). We may correct for any error in  $\tilde{u}$  caused by  $\tilde{g}_D$  by replacing it with  $u$  along the Dirichlet boundary  $\partial\Omega_D$  and similarly any error caused by  $\tilde{g}_N$  is taken care of by replacing it with  $\mathcal{N}u$  along  $\partial\Omega_N$ .

*Case 2: Inaccurate physical parameters and differential operators:* Let the true differential operator  $\mathcal{L}$  be dependent on some physical parameter  $k$  (e.g. heat conductivity) that may be a spatial and/or temporal function. We indicate this dependency by writing  $\mathcal{L}(k)$ . Assume now that we do not know the exact value of  $k$ , but instead only an approximation  $\tilde{k}$ , such that our perturbed PDE is defined using the operator  $\tilde{\mathcal{L}} = \mathcal{L}(\tilde{k})$ . Alternatively, assume that we do not know (or simply neglect) some terms in the true operator  $\mathcal{L}$  and denote the resulting inaccurate operator  $\tilde{\mathcal{L}}$ .

For these two situations we will typically solve the following problem:

$$\tilde{\mathcal{L}}\tilde{u} = f \quad \text{in } \Omega. \quad (13)$$

Assuming again that we know  $u$  (e.g. by measurements or analytical solution) the residual, due to inaccurate differential operator where  $\tilde{\mathcal{L}} \neq \mathcal{L}$ , is given by Eq. (7), i.e.,  $\tilde{r} = \mathcal{L}\tilde{e}$ . However, if  $\mathcal{L}$  is unknown we cannot compute  $\tilde{r}$  from the relations above. Therefore, we introduce an alternative residual  $\hat{\tilde{r}}$  corresponding to using the perturbed differential operator as follows:

$$\hat{\tilde{r}} := \tilde{\mathcal{L}}\tilde{e} \quad (14)$$

We then add  $\hat{\tilde{r}}$  as a corrective source term to find the CoSTA-improved (analytical) solution  $u_{\text{CoSTA}}$ :

$$\tilde{\mathcal{L}}u_{\text{CoSTA}} = f + \hat{\tilde{r}} \quad (15)$$

$$= \tilde{\mathcal{L}}\tilde{u} + \tilde{\mathcal{L}}\tilde{e} \quad (16)$$

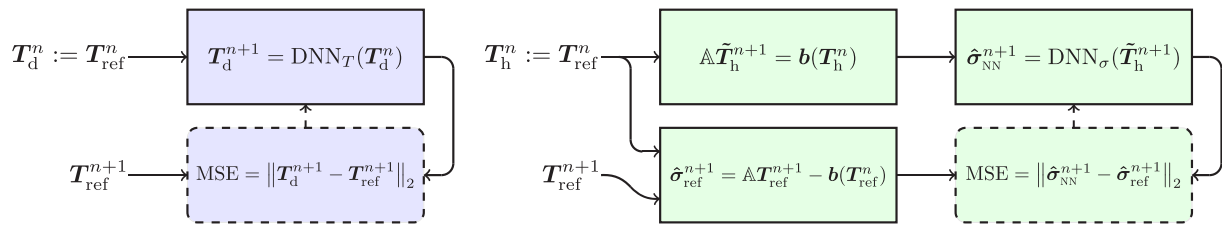
$$= \tilde{\mathcal{L}}u \quad (17)$$

Thus, CoSTA can be looked upon as either solving a “manufactured solution” defined by the true solution in Eq. (17), or as solving the problem using a perturbed (corrected) source term  $f + \hat{\tilde{r}}$  as given in Eq. (15) – in both cases using the (inaccurate) perturbed differential operator  $\tilde{\mathcal{L}}$ . Notice that, in the above, we get an analytical solution  $u_{\text{CoSTA}}$  that corresponds to a source term  $\tilde{\mathcal{L}}u$  defined by the true solution  $u$  on a perturbed PDE defined by  $\tilde{\mathcal{L}}$ . If the perturbed PDE admits a unique analytical solution, then the use of CoSTA will imply that  $u_{\text{CoSTA}} = u$ .

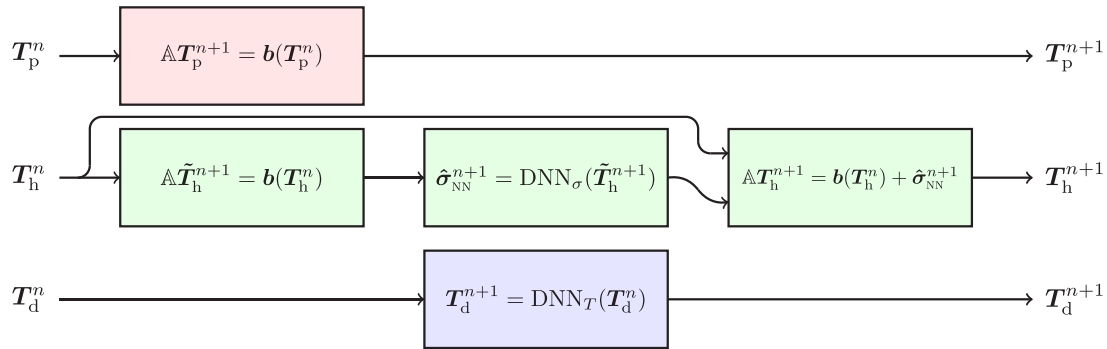
*Case 3: Inaccurate differential operator due to discretization errors:* Above we have described inaccuracy in the continuous PDE

<sup>1</sup> We may generalize the approach to handle nonlinear differential operators, but for notational simplicity we restrict ourselves first to address linear differential operators. See Blakseth (2021) for a consideration of non-linear operators.

<sup>2</sup> If  $u$  is only known in discrete points (e.g. it is measured) we may interpolate it or project it onto a polynomial basis of order  $p$  to achieve  $u_p$  which then can be differentiated and used instead of  $u$  in Eq. (12).

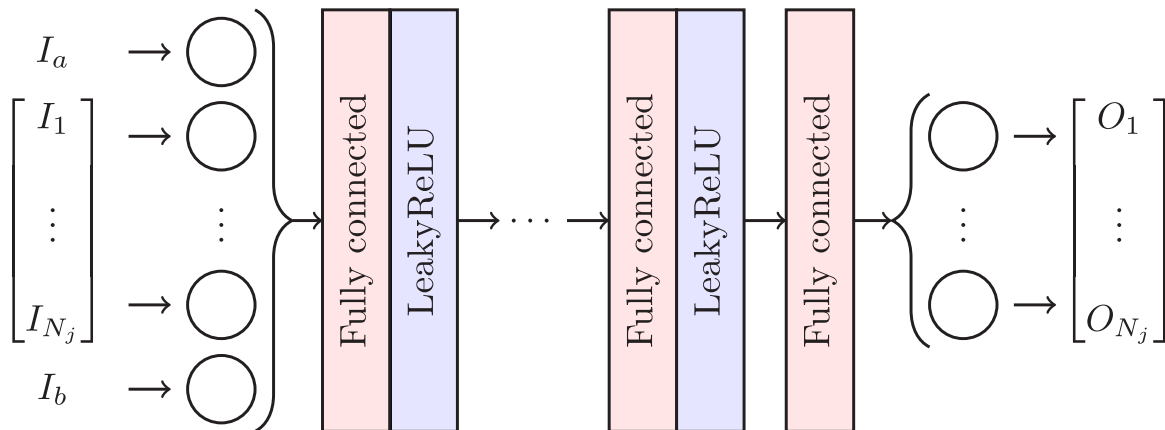


(a) Training procedures for the DNNs used in DDM (left) and HAM (right).



(b) Testing procedures for PBM (top), HAM (middle) and DDM (bottom).

**Fig. 3.** Training and testing procedures for the three modeling approaches PBM, DDM and HAM. Note that the PBM does not require any training. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**Fig. 4.** The fully connected neural network architecture used in the present work. The definitions of the input vector  $\mathbf{I}$  and the output vector  $\mathbf{O}$  depend on whether the network is used for HAM or DDM. However, note that  $\mathbf{I}$  always has two more components than  $\mathbf{O}$  due to the Dirichlet boundary conditions considered in this work.

due to modeling errors. However, when we solve a PDE with FDM/FVM/FEM we introduce discretization errors as we are solving the problem with a discrete approximation  $\mathcal{L}_{\text{num}}$  of the true differential operator  $\mathcal{L}$  (e.g. FDM using central differences). Following the approach for *Case 2* above, we get the same relationships as given in Eqs. (15)–(17) by substituting  $\tilde{\mathcal{L}}$  with  $\mathcal{L}_{\text{num}}$  for problems with only discretization errors and no modeling errors.

*Case 4: Combined modeling and discretization errors:* In our study herein, we will address problems where we have both modeling and discretization errors. Denote the corresponding differential operator  $\tilde{\mathcal{L}}_{\text{num}}$  and the inaccurate source term  $\tilde{f}$ . Our approach for retaining the true solution  $u$  of the true problem defined by  $\mathcal{L}$  and  $f$ , is outlined below.

We first solve the following problem to find a predictor  $\tilde{u}_{\text{num}}$ :

$$\tilde{\mathcal{L}}_{\text{num}} \tilde{u}_{\text{num}} = \tilde{f} \quad \text{in } \Omega. \quad (18)$$

Then we compute the residual, i.e. the corrective source term, corresponding to the error  $\tilde{e}_{\text{num}} = u - \tilde{u}_{\text{num}}$  in the predictor<sup>3</sup>:

$$\hat{r}_{\text{num}} := \tilde{\mathcal{L}}_{\text{num}} \tilde{e}_{\text{num}} \quad (19)$$

Finally, we do the following corrector step to compute the CoSTA-improved numerical solution:

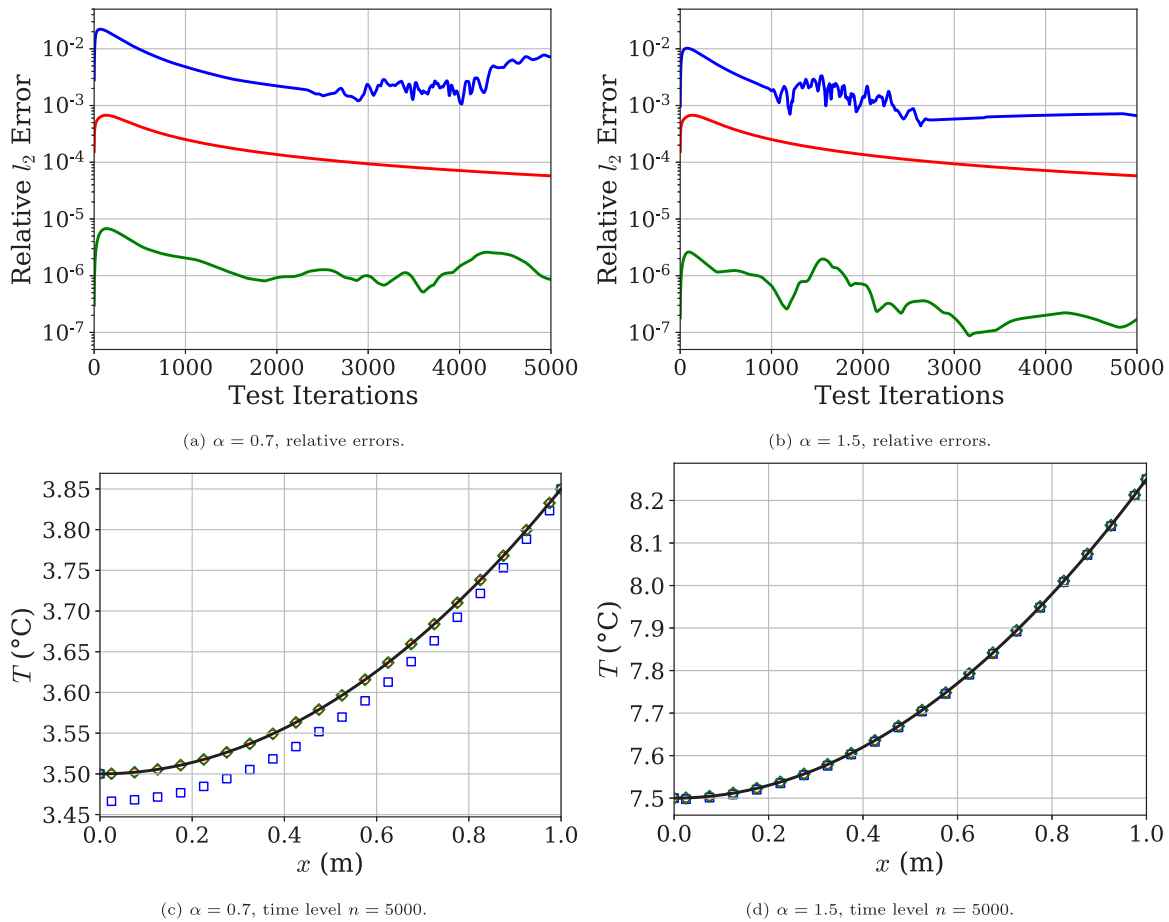
$$\tilde{\mathcal{L}}_{\text{num}} u_{\text{CoSTA}} = \tilde{f} + \hat{r}_{\text{num}} \quad (20)$$

$$= \tilde{\mathcal{L}}_{\text{num}} \tilde{u}_{\text{num}} + \tilde{\mathcal{L}}_{\text{num}} \tilde{e}_{\text{num}} \quad (21)$$

$$= \tilde{\mathcal{L}}_{\text{num}} u \quad (22)$$

Notice that if we knew the true solution  $u(x, t; \mu)$  at any node, at every time step for any choice of the parameter vector  $\mu$  *a priori*, we would not need to do the predictor step or compute

<sup>3</sup> See Section 2.4 for an example of how to compute the corrective source term in practice.



**Fig. 5.** Solution 0, interpolation: Comparison of relative errors and final temperature profiles for  $\alpha = 0.7, 1.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). HAM's predictions are by far the most accurate, followed by PBM, while DDM is least accurate.

the related corrective source term, because we could have solved Eq. (22) directly. However, in practice we do not know  $u$  for all choices of  $\mu$ , but we may train a neural network to return a quite accurate corrective source term (formally defined by Eq. (19)) given a predictor  $\tilde{u}_{\text{num}}$  computed by Eq. (18). Thus, the corrector step in CoSTA corresponds then to solving Eq. (20).

*Applications:* To test and demonstrate the value of the proposed CoSTA approach, we choose the problem of one-dimensional heat conduction described by a PDE derived from the first principles using the first law of thermodynamics. Accurate modeling of heat conduction is vital for a wide array of problems ranging from the modeling of heat transfer from the Earth to its atmosphere, modeling heat transfer characteristics of the built environment, and modeling the accumulation of thermal stresses in heat storage infrastructures. However, the accuracy in most of these applications is compromised due to geometric simplifications, uncertainty associated with the values of thermophysical properties used in the calculation, neglect of unknown (and even known) phenomena, and numerical approximations. The CoSTA approach, if successful, has the potential to solve these kinds of issues – not only for heat conduction modeling, but also for the modeling of any other steady-state or dynamical system that can be described by a (system of) PDE(s).

## 2.2. Physics-based modeling

In PBM, PDEs are widely used as governing equations, describing various physical phenomena by relating partial derivatives of relevant physical quantities. In this paper, we consider the

one-dimensional (1D) unsteady heat diffusion equation, which describes 1D transient heat transfer in a system of volume  $V$  and cross-sectional area  $A$ . The equation, which can be derived by applying the principle of energy conservation to the 1st law of thermodynamics, reads

$$\int_V \rho c_V \frac{\partial T}{\partial t} dV = \left( kA \frac{\partial T}{\partial x} \right)_e - \left( kA \frac{\partial T}{\partial x} \right)_w + \int_V \hat{q} dV, \quad (23)$$

where  $T$ ,  $\rho$ ,  $c_V$ , and  $k$  denote temperature, density, heat capacity, and thermal diffusivity, respectively. The term on the equation's left-hand side represents the momentary change in the system's internal energy. Furthermore, the first two terms on the right-hand side represent the heat flux across the system's right (eastern, denoted by subscript  $e$ ) boundary and left (western, denoted by subscript  $w$ ) boundary, respectively, while the last term on the right-hand side ( $\hat{q}$ ) is a source term which accounts for heat generated within the system. Under certain smoothness requirements, the 1D unsteady heat equation can also be written on the so-called differential form:

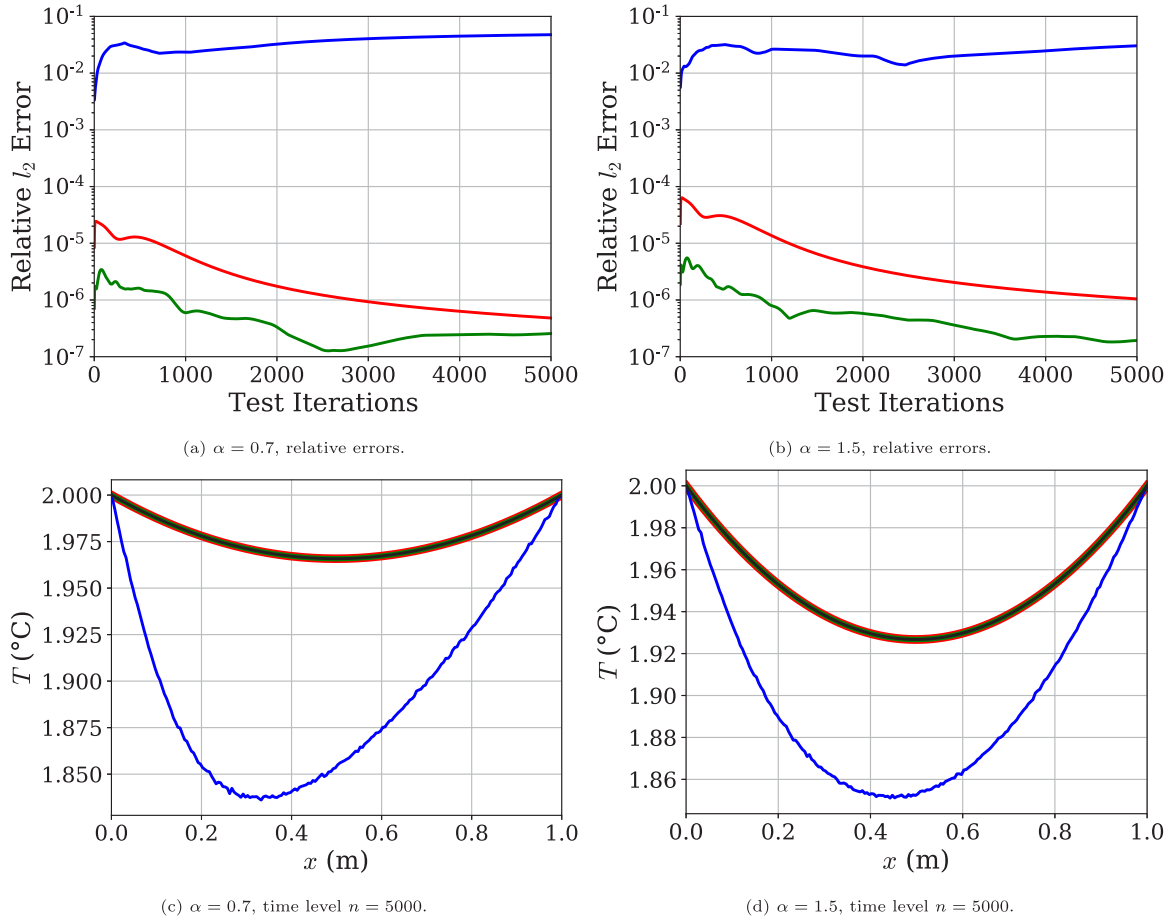
$$\rho c_V \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left( k \frac{\partial T}{\partial x} \right) + \hat{q}. \quad (24)$$

Comparing to Eq. (1), the differential operator of the heat equation is given by

$$\mathcal{L}T = \rho c_V \frac{\partial T}{\partial t} - \frac{\partial}{\partial x} \left( k \frac{\partial T}{\partial x} \right) \quad (25)$$

while the source term is  $f = \hat{q}$ .

In the cases where the solution of Eq. (23) (or Eq. (24)) cannot be expressed analytically, approximate solutions can be obtained



**Fig. 6.** Solution 3 with fine grid, interpolation: Comparison of relative errors and final temperature profiles for  $\alpha = 0.7, 1.5$  (— Exact, — PBM, — DDM, — HAM). HAM improves on an already highly accurate PBM, while DDM is inaccurate.

using numerical methods such as FDMs, FVMs, and FEMs. For Dirichlet boundary conditions (BCs), Eq. (23) can be written in the form

$$\mathbb{A}\mathbf{T}^{n+1} = \mathbf{b}(\mathbf{T}^n; T_a, T_b, \hat{q}), \quad (26)$$

when discretized using the Implicit Euler FVM. Here,  $\mathbf{T} = [T_1, \dots, T_{N_j}]^T$  denotes the temperature at all  $N_j$  interior grid nodes  $x_1, \dots, x_{N_j}$ . Furthermore, the superscripts  $n$  and  $n+1$  denote two subsequent time levels,  $\mathbb{A}$  is a tri-diagonal matrix and  $\mathbf{b}$  is a vector depending on  $\mathbf{T}^n$ , the BCs ( $T_a$  and  $T_b$ ) and  $\hat{q}$ . Notice that  $\mathbb{A}$  is the algebraic matrix representation of the discrete differential operator  $\mathcal{L}_{\text{num}}$ , and  $\mathbf{b}$  is the vector representation of the source term  $f$  which also includes the effects of the boundary conditions.

Since Eq. (26) is an approximation of Eq. (23), a solution of one of the equations is generally not a solution of the other. Note also that, in cases where the governing equation (23) is not fully known, Eq. (26) has to be based on an approximation of Eq. (23), which causes further discrepancies between the solutions of Eq. (23) and Eq. (26), as discussed in Section 1. To distinguish the two classes of solutions, we use the notation  $T_{\text{ref}}$  to denote a solution of the true governing equation (23) (i.e., similar to  $u$  given by Eqs. (1)–(3) in the general case) and  $\mathbf{T}_p$  to denote a solution of the discrete system (26). In the context of a prediction problem,  $T_{\text{ref}}$  is then the ideal prediction, while  $\mathbf{T}_p$  is the prediction made by the PBM (i.e., similar to  $\hat{u}_{\text{num}}$  given by Eq. (18) in the general case). Thus the equation that we solve to generate the PBM solution is given by

$$\mathbb{A}\mathbf{T}_p^{n+1} = \mathbf{b}(\mathbf{T}_p^n) \quad (27)$$

with the prescribed boundary conditions implicitly included in  $\mathbf{b}$ . It should be stressed that there is no learning involved in PBM and hence, in Fig. 3 – where we illustrate the training and testing processes for PBM, DDM and HAM – there is no mention of PBM in the part concerning training (Fig. 3(a)).

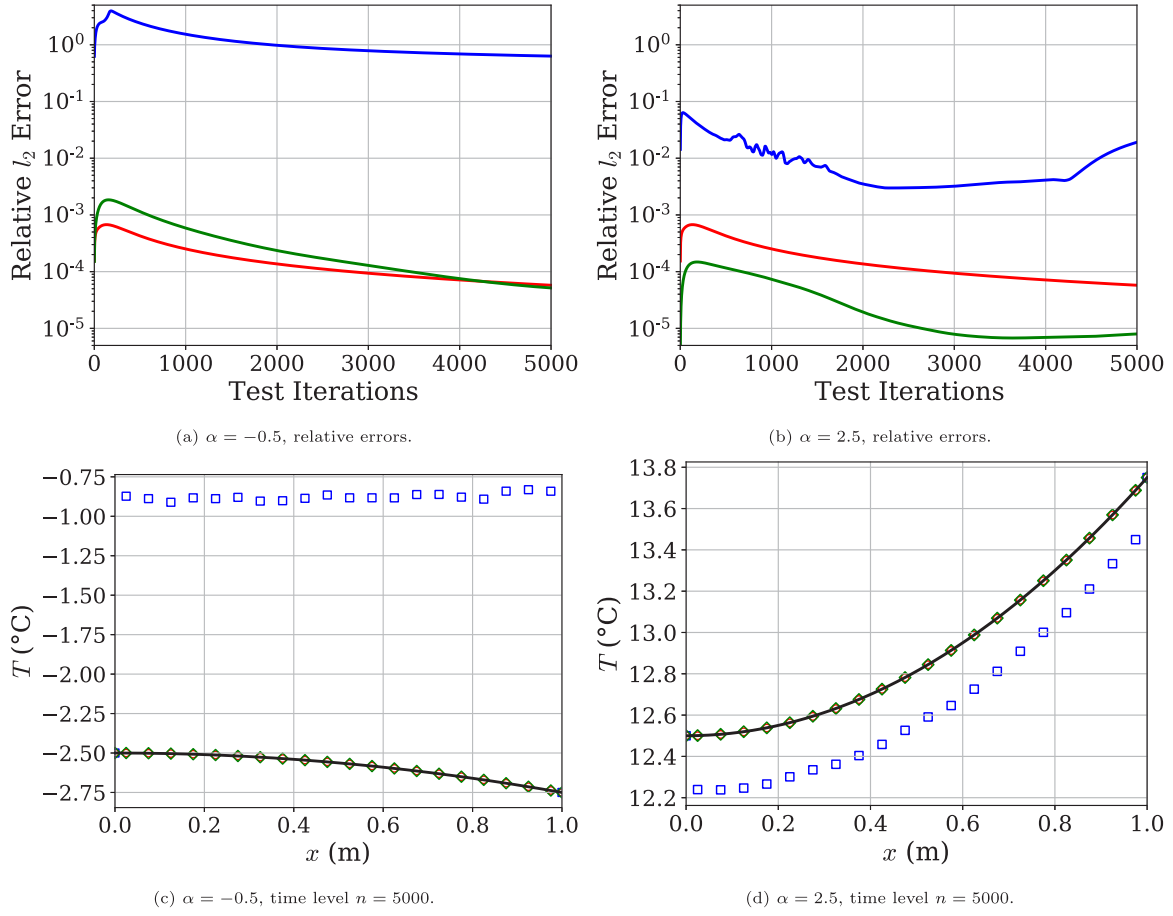
### 2.3. Data-driven modeling

In DDM, physics is learned directly from the observation data. For transient systems, one common DDM approach is to define a mapping from the observed state at one time level to the observed state at the subsequent time. A DNN is then trained to approximate this mapping. In the context of 1D heat diffusion problems with known Dirichlet BCs, the mapping to be learned by the DNN is

$$\text{DNN}_T : \mathbb{R}^{N_j+2} \rightarrow \mathbb{R}^{N_j} \quad \text{such that} \quad \mathbf{T}_d^{n+1} = \mathbf{T}_{\text{ref}}^{n+1}, \quad (28)$$

$$\mathbf{T}_d^n \mapsto \mathbf{T}_d^{n+1}$$

where  $\mathbf{T}_d^n$  refers to the temperature profile predicted by the DDM at time level  $n$ , and  $\mathbf{T}_{\text{ref}}^n$  is the solution  $T_{\text{ref}}$  of the true governing equation (Eq. (23)) sampled at the grid nodes  $x_1, \dots, x_{N_j}$  and time level  $n$ . Note that the dimensionality discrepancy between the DNN's input and output is due to the input containing the boundary temperatures, which the output does not include; since the boundary temperatures are assumed known, there is no reason to have the DNN predict them. However, we do want to include the boundary temperature in the DNN input, since they represent relevant physical information. To avoid notational complexity, we



**Fig. 7.** Solution 0, extrapolation: Comparison of relative errors and final temperature profiles for  $\alpha = -0.5, 2.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). Both HAM and PBM outperform DDM. HAM improves on PBM for  $\alpha = 2.5$ , which is the scenario most similar to the training scenarios, but diminishes accuracy for  $\alpha = -0.5$ .

use  $\mathbf{T}_d^n$  to denote both vectors containing and not containing the boundary temperatures.

Our reason for choosing DNN-based DDM over other applicable DDMs is that DNNs have the ability to approximate any nonlinear function, as guaranteed by the universal approximation theorem. DNNs (Goodfellow et al., 2016), which are inspired by the biological neural networks found in e.g. human brains, typically consist of multiple layers with one layer's output being passed through a non-linear activation function before being used as the input of the next layer. Each layer typically consists of a number of processing units with their own tunable parameters commonly called weights and biases. The nature of, and relations between, the processing units vary across different layer types. We refer to the specific composition of different layers used to define a DNN as that DNN's 'architecture'.

We say that a neural network is 'trained' when the individual parameters are tuned in an effort to make the network a better approximator of the desired function. In this paper, we train the DNNs using the framework of supervised learning, which requires the preparation of sample DNN inputs and corresponding target outputs. During training, for any sample input, the DNN's output is compared to the corresponding target output using a chosen cost function. Then, the backpropagation algorithm (Goodfellow et al., 2016) is used to calculate the gradients of the computed cost with respect to the individual network parameters. Finally, the network parameters are updated, typically using a gradient descent algorithm, such as to minimize the cost function. The cost function used in this work is the commonly used mean squared error. An overview of the training and testing approach for DDM is illustrated in Fig. 3 with the color blue.

#### 2.4. Hybrid analysis and modeling with CoSTA

Given a PBM, the principal goal of the corrective source term approach (CoSTA) is to modify the governing equation solved by the PBM using a corrective source term, such as to recover the true solution of the problem at hand. In this section, we demonstrate how CoSTA can be used in practice to correct the Implicit Euler FVM for unsteady heat transfer (Eq. (26)).

The first step of applying CoSTA to the Implicit Euler FVM is to add the corrective source term  $\hat{\sigma}^{n+1}$  to the right hand side of Eq. (26), such as to obtain the modified system

$$\Delta \mathbf{T}_h^{n+1} = \mathbf{b}(\mathbf{T}_h^n; T_a, T_b, \hat{q}) + \hat{\sigma}^{n+1}, \quad (29)$$

whose solutions we denote  $\mathbf{T}_h$  (the subscript  $h$  corresponds to HAM). Our goal is now to obtain an explicit expression for  $\hat{\sigma}^{n+1}$  using the framework from Section 2.1. To this end, notice that Eq. (29) is analogous to Eq. (20) with the following relations:

$$\tilde{\mathcal{L}}_{\text{num}} = \Delta, \quad u_{\text{CoSTA}} = \mathbf{T}_h^{n+1}, \quad \tilde{\mathbf{f}} = \mathbf{b}, \quad \hat{\mathbf{r}} = \hat{\sigma}^{n+1}. \quad (30)$$

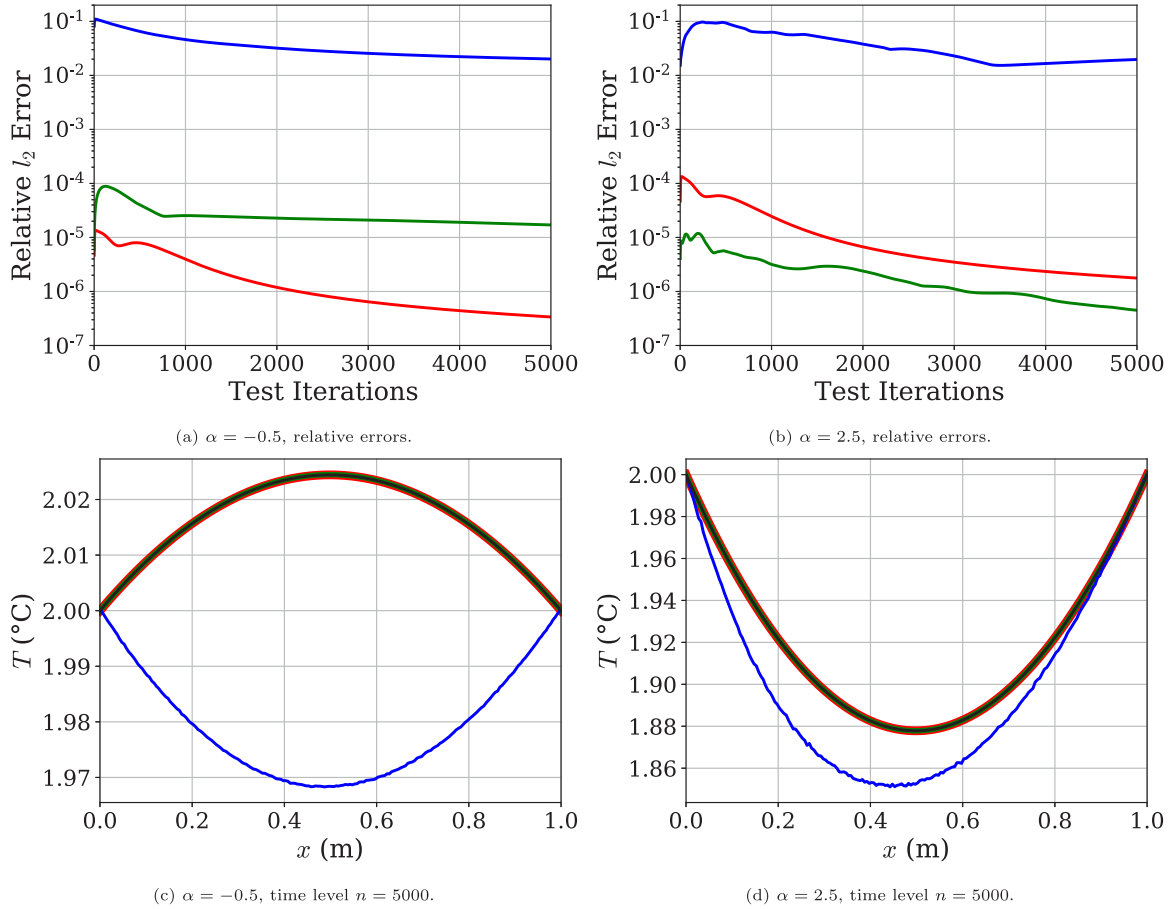
From the definition of  $\hat{\mathbf{r}}$  (cf. Eq. (19)), we thus have

$$\hat{\sigma}^{n+1} = \hat{\mathbf{r}} := \tilde{\mathcal{L}}_{\text{num}} \tilde{\mathbf{e}}_{\text{num}}, \quad (31)$$

$$= \tilde{\mathcal{L}}_{\text{num}} u - \tilde{\mathcal{L}}_{\text{num}} \tilde{u}_{\text{num}} \quad (32)$$

where we utilized the definition of  $\tilde{\mathbf{e}}_{\text{num}}$  to transition from the first to the second line. As in Section 2.3, we let  $\mathbf{T}_{\text{ref}}^{n+1}$  denote the true solution which we aim to find, i.e., we have  $u = \mathbf{T}_{\text{ref}}^{n+1}$ . Moreover, as the analogue of the predictor  $\tilde{u}_{\text{num}}$ , we choose  $\tilde{\mathbf{T}}_h^{n+1}$





**Fig. 8.** Solution 3 with fine grid, extrapolation: Comparison of relative errors and final temperature profiles for  $\alpha = -0.5, 2.5$  (— Exact, — PBM, — DDM, — HAM). For both  $\alpha$ , PBM exhibits high accuracy while DDM exhibits comparatively low accuracy. HAM is able to improve on the accuracy of the PBM for the  $\alpha = 2.5$ , but yields reduced accuracy for  $\alpha = -0.5$ .

given by

$$\mathbb{A}\tilde{\mathbf{T}}_h^{n+1} = \mathbf{b}(\mathbf{T}_h^n), \quad (33)$$

where  $\mathbb{A}$  and  $\mathbf{b}$  are as defined in Eq. (29). By inserting for  $\tilde{\mathcal{L}}_{\text{num}}$ ,  $u$  and  $\tilde{u}_{\text{num}}$  in Eq. (32), we thus obtain

$$\hat{\sigma}^{n+1} = \mathbb{A}\mathbf{T}_{\text{ref}}^{n+1} - \mathbb{A}\tilde{\mathbf{T}}_h^{n+1} \quad (34)$$

$$= \mathbb{A}\mathbf{T}_{\text{ref}}^{n+1} - \mathbf{b}(\mathbf{T}_h^n). \quad (35)$$

If we now insert  $\mathbf{T}_h^n = \mathbf{T}_{\text{ref}}^n$  into the equation above,<sup>4</sup> we get

$$\hat{\sigma}^{n+1} = \mathbb{A}\mathbf{T}_{\text{ref}}^{n+1} - \mathbf{b}(\mathbf{T}_{\text{ref}}^n), \quad (36)$$

which is the definition of the corrective source term that we will use to generate data for our numerical experiments (cf. Section 3.3). Note that  $\hat{\sigma}^{n+1}$  corrects the error of the Implicit Euler FVM over a single time step. Starting from some known initial temperature profile  $\mathbf{T}^0 = \mathbf{T}_h^0 = \mathbf{T}_{\text{ref}}^0$ , the combined use of Eqs. (29) and (36), guarantees  $\mathbf{T}_h^n = \mathbf{T}_{\text{ref}}^n$  also for all  $n > 0$ .<sup>5</sup>

Since  $\mathbf{T}_{\text{ref}}^{n+1}$  is not known *a priori*, Eq. (36) – and hence also Eq. (29) – cannot be evaluated using pure PBM. On the other hand, pure DDM can (implicitly) take  $\hat{\sigma}^{n+1}$  into account, but also completely discards what is already known about heat diffusion

<sup>4</sup> Our primary motivation for doing this is that it produces the desirable result  $\mathbf{T}_h^n = \mathbf{T}_{\text{ref}}^n \implies \mathbf{T}_h^{n+1} = \mathbf{T}_{\text{ref}}^{n+1} \forall n \geq 0$ .

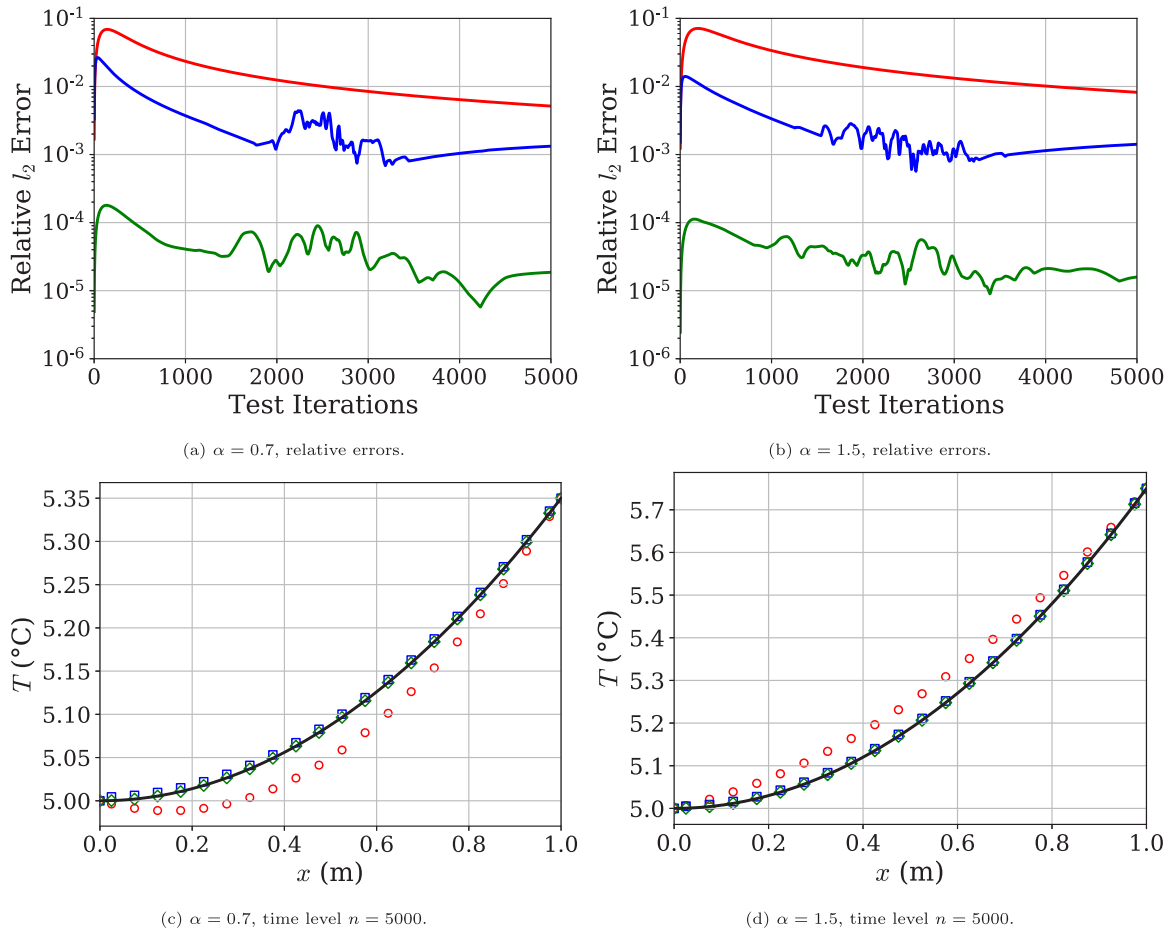
<sup>5</sup> This can be proven by induction:  $\mathbf{T}_h^n = \mathbf{T}_{\text{ref}}^n \implies \mathbb{A}\mathbf{T}_h^{n+1} = \mathbf{b}(\mathbf{T}_h^n) + \hat{\sigma}^{n+1} = \mathbf{b}(\mathbf{T}_{\text{ref}}^n) + \mathbb{A}\mathbf{T}_{\text{ref}}^{n+1} - \mathbf{b}(\mathbf{T}_{\text{ref}}^n) = \mathbb{A}\mathbf{T}_{\text{ref}}^{n+1}$ , which implies  $\mathbf{T}_h^{n+1} = \mathbf{T}_{\text{ref}}^{n+1}$  since  $\mathbb{A}$  has full rank.

problems. Instead of going to either of these extremes, we choose a middle ground by training a deep neural network  $\text{DNN}_\sigma$  to approximate Eq. (36) given the predictor  $\tilde{\mathbf{T}}_h^{n+1}$  defined in Eq. (33). The DNN approximation is then inserted into the modified PBM (Eq. (29)). That is, we insert  $\hat{\sigma}_{\text{NN}}^{n+1} = \text{DNN}_\sigma(\tilde{\mathbf{T}}_h^{n+1})$  in the place of  $\hat{\sigma}^{n+1}$  in Eq. (29) to obtain the HAM prediction  $\mathbf{T}_h^{n+1}$ .

Training data for the DNN can be generated from a known time series describing the system’s past, or from time series describing the temporal development of similar systems, using Eq. (36). The whole process of training and testing the complete CoSTA-based HAM model is illustrated in Fig. 3 using the color green.

### 2.5. Method of manufactured solutions

A central part of the present study is the method of manufactured solutions (MMS), which has long been a popular tool for verifying the numerical PDE solvers used in PBMs (see e.g. Roache (2002) for an extensive introduction). The key concept of MMS is to choose some explicitly expressible function, and then calculate the source term required for this function to be a solution of the PDE in question. For the 1D unsteady heat diffusion equation, this amounts to deciding upon some temperature function  $T(x, t)$  and calculating the source term  $\hat{q}(x, t)$  required for the differential form of the equation (Eq. (24)) to be satisfied. Thus, the use of MMS allows us to obtain exact reference solutions  $T_{\text{ref}}$  of the 1D unsteady heat equation without running expensive high-fidelity simulations. We can then use these reference solutions to evaluate the accuracy of the temperature profiles predicted using



**Fig. 9.** Solution 1, interpolation: Comparison of relative errors and final temperature profiles for  $\alpha = 0.7, 1.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). All predictions are qualitatively correct, but HAM yields superior accuracy.

PBM, DDM and HAM, as well as for generating DNN training and validation data.

### 3. Experimental setup and procedures

To evaluate the performance of the PBM, DDM and HAM models described in the previous section, we have designed a series of numerical experiments (the results of which are presented and discussed in Section 4), where each experiment is based on a manufactured solution of the 1D unsteady heat equation (Eq. (23)). The experimental setup and procedures used to conduct these experiments are described in the following section.

#### 3.1. Choice of manufactured solutions

All manufactured solutions  $T(x, t; \alpha)$  used to conduct the present study are listed in Table 1. They include both polynomials and trigonometric functions, so as to cover a wide variety of different functional behaviors. Each solution is parametrized by a quantity  $\alpha$ , which allows us to investigate the generalizability of the PBM, DDM and HAM methods across different  $\alpha$ -values (corresponding to generalizability across different operating conditions in an application context). Along with each manufactured solution, we have included the corresponding source term  $\hat{q}$  required for the listed functions  $T(x, t; \alpha)$  to satisfy the 1D heat equation (Eq. (23) or, equivalently, Eq. (24)). From a real world application perspective, the problem corresponds to a one dimensional solid body initialized (at  $t = 0$ ) with a temperature profile using the functions given in Table 1. Then the evaluation

of the temperature across the solid body is predicted under the influence of differential heating across the length of the solid body given by the source term  $\hat{q}$ . This heat transfer phenomenon is encountered in a wide variety of real life applications like heat loss/gain through the building walls, transfer of heat from the center of the earth to the atmosphere, and cooling/heating of electric chip on electrical equipment.

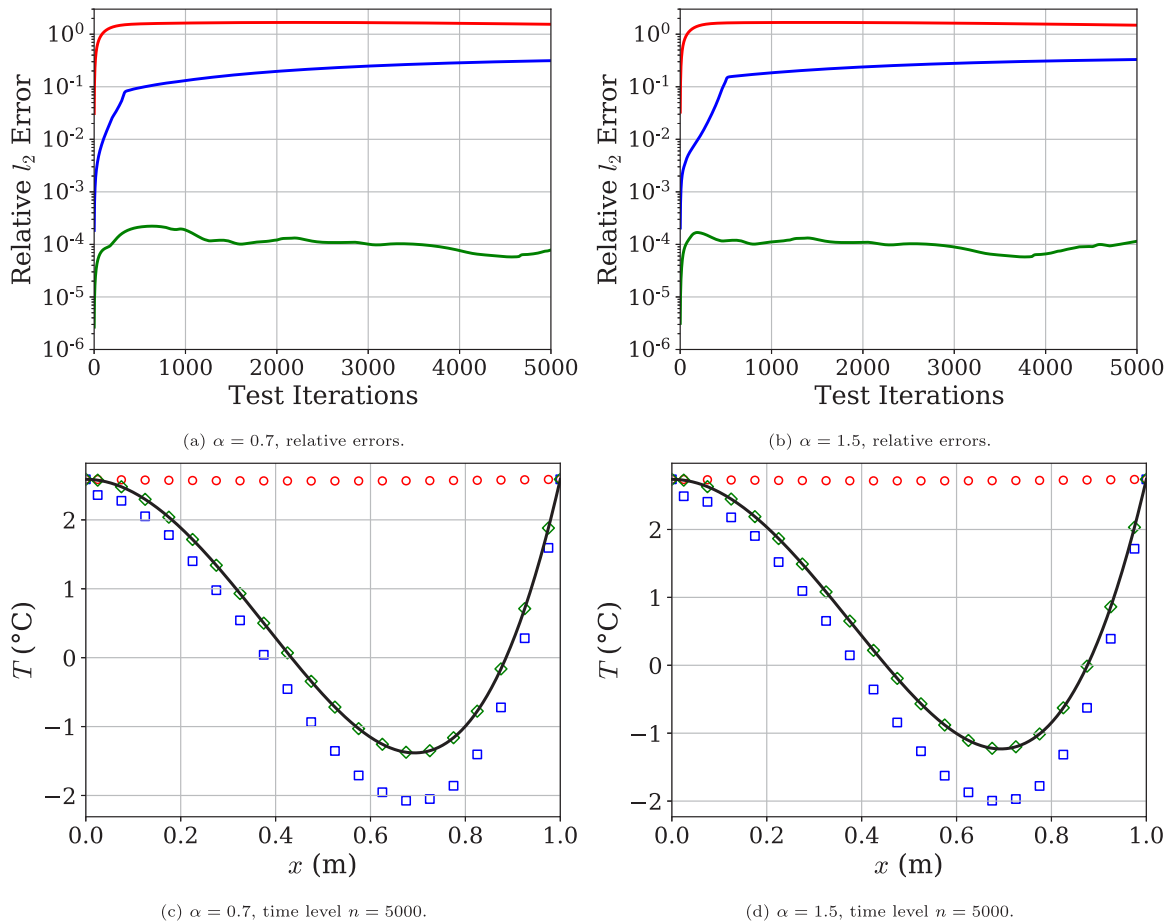
#### 3.2. Parametrization

For each manufactured solution in Table 1, the 22 different  $\alpha$ -values listed in Table 2 were used to generate the training, validation and testing data. Of the 22  $\alpha$ -values, 16 were used to generate training data, 2 were used to generate validation data and 4 were used to generate testing data, as indicated in the table. Of the  $\alpha$ -values used for testing, two lie within the interval  $[0.1, 2.0]$  covered by  $\mathcal{A}_{\text{train}}$  (defined in Table 2) while the other two do not. This allows us to evaluate the generalizability of PBM, DDM and HAM in both interpolation and extrapolation scenarios.

#### 3.3. Training and validation data generation

Given an  $\alpha$  and a manufactured solution  $T(x, t; \alpha)$  of the 1D unsteady heat equation, we sample  $T(x, t; \alpha)$  at the center of  $N_j = 20$  equally sized grid cells<sup>6</sup> on the spatial domain  $[0 \text{ m}, 1 \text{ m}]$  and at  $N_t = 5001$  equally spaced time levels on the temporal

<sup>6</sup> With the exception of one experiment in Section 4.1, where we used  $N_j = 200$  equally sized grid cells.



**Fig. 10.** Solution 2, interpolation: Comparison of relative errors and final temperature profiles for  $\alpha = 0.7, 1.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). Only HAM yields predictions that are qualitatively correct and of reasonable accuracy.

**Table 1**

Manufactured solutions: Functions  $T(x, t; \alpha)$  used for performance evaluation of the studied PBM, DDM and HAM methods (top five rows) and for DNN hyperparameter tuning (bottom two rows). All  $T(x, t; \alpha)$  were defined on the spatial domain  $x \in [0 \text{ m}, 1 \text{ m}]$  and the temporal domain  $t \in [0 \text{ s}, 5 \text{ s}]$ .

Solutions	$T(x, t; \alpha)$	$\hat{q}(x, t; \alpha)$
0	$\alpha(t + \frac{1}{2}x^2)$	0
1	$t + \frac{1}{2}\alpha x^2$	$1 - \alpha$
2	$\sqrt{t + \alpha + 1} + 10x^2(x - 1)(x + 2)$	$\frac{1}{2\sqrt{t + \alpha + 1}} - 120x^2 - 60x + 40$
3	$2 + \alpha(x - 1) \tanh(\frac{x}{t+0.1})$	$\frac{\alpha}{(t+0.1)^2} (x(1-x) + 2((x-1) \tanh(\frac{x}{t+0.1}) - t - 0.1)) \operatorname{sech}^2(\frac{x}{t+0.1})$
4	$1 + \sin(2\pi t + \alpha) \cos(2\pi x)$	$2\pi (\cos(2\pi t + \alpha) - 2\pi \sin(2\pi t + \alpha)) \cos(2\pi x)$
A	$\sqrt{t + \alpha + 1} + 7x^2(x - 1)(x + 2)$	$\frac{1}{2\sqrt{t + \alpha + 1}} - 84x^2 - 42x + 28$
B	$-\frac{x^3(x-\alpha)}{t+0.1}$	$\frac{x^4 - \alpha x^3}{(t+0.1)^2} + \frac{12x^2 - 6\alpha x}{t+0.1}$

**Table 2**

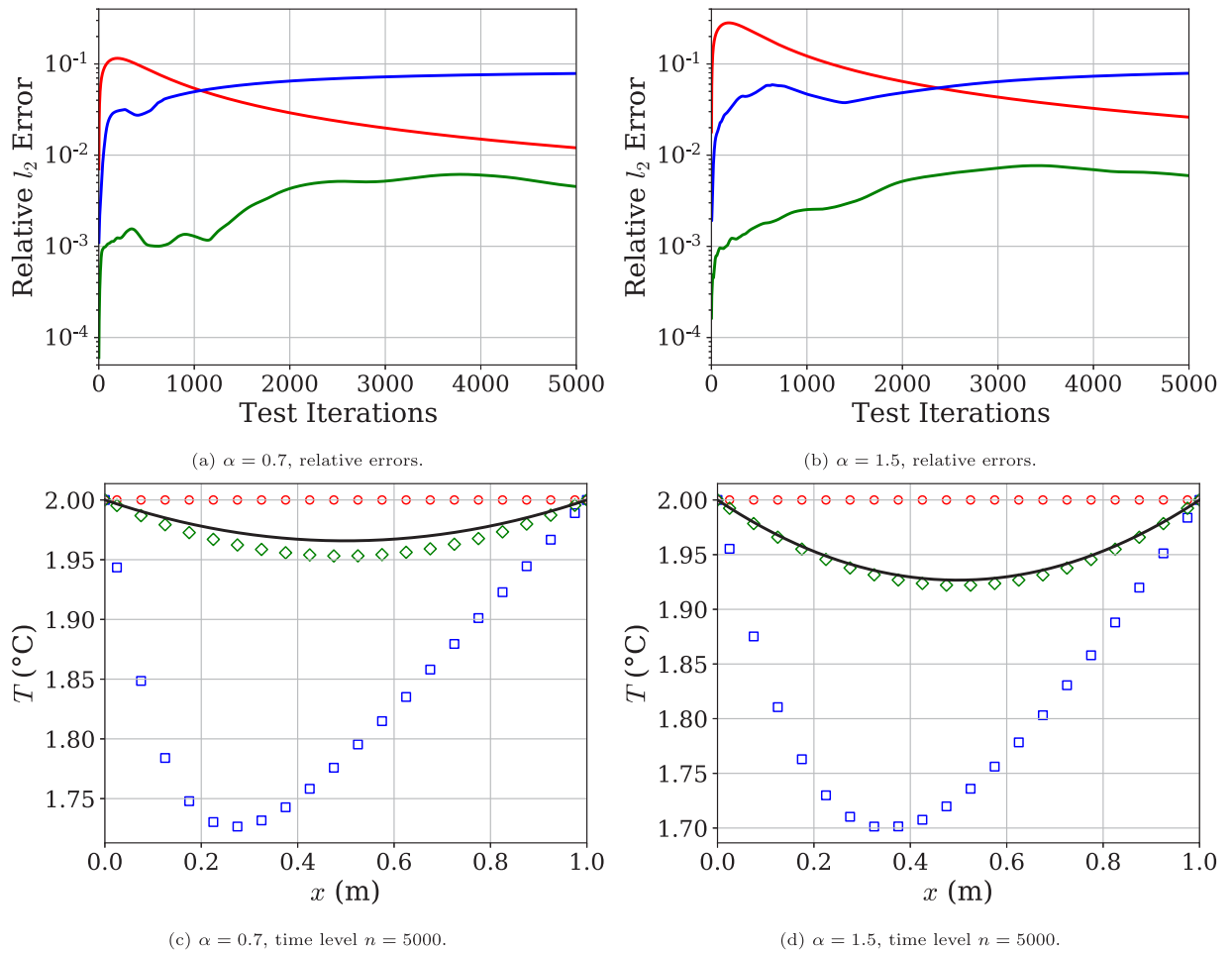
Parametrization: Selection of  $\alpha$ -values used for generating training, validation and testing time series.

Purpose	Set of $\alpha$ -values	Symbol
Training	$\{0.1, 0.2, \dots, 2.0\} \setminus \{0.7, 0.8, 1.1, 1.5\}$	$\mathcal{A}_{\text{train}}$
Validation	$\{0.8, 1.1\}$	$\mathcal{A}_{\text{val}}$
Testing	$\{-0.5, 0.7, 1.5, 2.5\}$	$\mathcal{A}_{\text{test}}$

domain  $[0 \text{ s}, 5 \text{ s}]$  to generate a reference time series  $\{\mathbf{T}_{\text{ref}}^n\}_{n=0}^{N_t-1}$ . Training and validation datasets are then generated using the reference time series as described in Algorithm 1. This results in a total of  $5000 \cdot 16 = 80000$  training examples and  $5000 \cdot 2 = 10000$  validation examples (per model) for each experiment, since we use 16  $\alpha$ -values to generate training data and 2  $\alpha$ -values to generate validation data (cf. Table 2). Moreover, each data

example contains two vectors, an  $N_j + 2$ -dimensional DNN input vector and an  $N_j$ -dimensional DNN target output vector.<sup>7</sup> The precise definitions of these vectors are given in Algorithm 1. Note that, for the experiments described in Section 4.2, we set  $\hat{q} = 0$  in Eq. (26) and Eq. (29) to synthesize modeling error. Note also that, for each time level  $n$ , all data in the datasets are generated based on  $\mathbf{T}_{\text{ref}}^{n-1}$ . This means that no error can accumulate across multiple time steps, so DNNs trained using these datasets will necessarily be trained to make corrections across individual time steps. That is, they correct only local, not global, time stepping errors. This choice was made based on three observations: (1) A successful, generalizable reduction in local error will necessarily

<sup>7</sup> As explained for DDM in Section 2.3, the dimensionality reduction occurs because we take the BCs to be known without error, which means the DNN need not make predictions for the boundaries. This also applies to HAM.



**Fig. 11.** Solution 3, interpolation: Comparison of relative errors and final temperature profiles for  $\alpha = 0.7, 1.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). Only HAM is able to provide qualitatively correct predictions at the final time level.

also reduce global error. (2) Corrections of global errors are hard to interpret due to error accumulation. (3) Data is more efficiently obtained when one data example corresponds to a single time step rather than a complete time series comprising multiple time steps.

### 3.4. DNN training

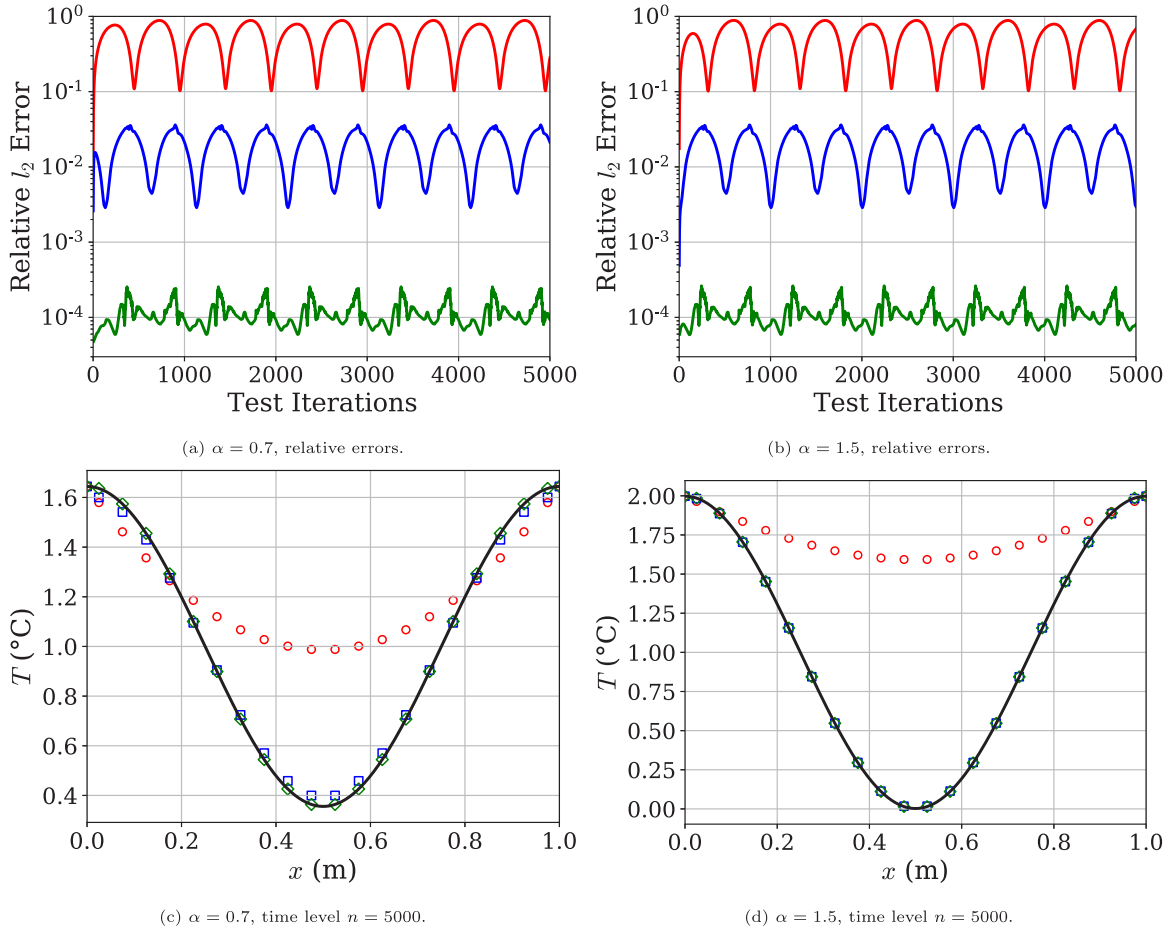
Both DDM and CoSTA-based HAM require training of DNNs using the datasets described above. For each training iteration, one batch of data examples  $(\mathbf{T}_{\text{ref}}^{n-1}, \tilde{\mathbf{T}}_h^n, \mathbf{T}_{\text{ref}}^n, \hat{\sigma}^n)$  is used to update the network's parameters. However, the components of each data example are used differently for the two methods. For HAM,  $\tilde{\mathbf{T}}_h^n$  is given as input to the DNN, and the corresponding DNN output is then compared to  $\hat{\sigma}^n$  using a cost function. On the other hand, for DDM,  $\mathbf{T}_{\text{ref}}^{n-1}$  is used as DNN input, and the corresponding output is compared to  $\mathbf{T}_{\text{ref}}^n$ , again using the cost function. We use the mean squared error (MSE) cost function, as implemented in the PyTorch ML framework (Paszke et al., 2019). Given the computed cost, the DNN parameters are updated using backpropagation and the chosen optimization algorithm. We use the Adam optimizer introduced in Kingma and Ba (2014). The full DNN training process is illustrated in Fig. 3(a), where we have also illustrated how  $\hat{\sigma}^n$  and  $\tilde{\mathbf{T}}_h^n$  were generated from the reference profiles  $\mathbf{T}_{\text{ref}}^{n-1}$  and  $\mathbf{T}_{\text{ref}}^n$ , as described in the previous section.

At regular intervals (validation periods), the total MSE cost for all data examples in the *validation* set is computed. The validation cost is computed analogously to the training cost, as described

above. We utilize the early stopping regularization technique by stopping the DNN training if a new lowest validation cost has not been recorded for a certain number of consecutive validation periods (this number is denoted “overfit limit” in Table 3).

### 3.5. Model evaluation

When deploying a predictive model for use on a practical application, it is imperative that the model remains accurate across multiple time levels without relying on the reference data. Thus, when evaluating the performance of PBM, DDM and HAM, we need to look at the *global* error of their predictions, even though, for DDM and HAM, the DNNs were trained to make local corrections only. For DDM, this means that  $\mathbf{T}_d^{n-1}$  – not  $\mathbf{T}_{\text{ref}}^{n-1}$  – must be used as DNN input to generate  $\mathbf{T}_d^n$ . Similarly, for HAM,  $\mathbf{T}_h^{n-1}$  – not  $\mathbf{T}_{\text{ref}}^{n-1}$  – must be inserted into Eq. (33) (with  $\hat{q} = 0$  for the experiments of Section 4.2) to generate the DNN input  $\tilde{\mathbf{T}}_h^n$ . For the sake of completeness, we also state that for PBM,  $\mathbf{T}_p^n$  is calculated by inserting  $\mathbf{T}_p^{n-1}$  – not  $\mathbf{T}_{\text{ref}}^{n-1}$  – into Eq. (27) (again with  $\hat{q} = 0$  for the experiments described in Section 4.2). The testing procedure is described in its entirety in Algorithm 2 and is illustrated in Fig. 3(b) for a single time step. Since we perform testing using 4 distinct  $\alpha$ -values (cf. Table 2), this procedure yields  $5000 \cdot 4 = 20000$  test predictions (per model) in any given experiment. We use the  $\ell_2$ -norm to quantify the accuracy of these predictions, as described in the beginning of Section 4. For a general  $N$ -dimensional vector  $\mathbf{v} \in \mathbb{R}^N$ , the definition of the



**Fig. 12.** Solution 4, interpolation: Comparison of relative errors and final temperature profiles for  $\alpha = 0.7, 1.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). Both DDM and HAM provide reasonable predictions, while PBM does not. HAM is significantly more accurate than DDM.

$\ell_2$ -norm reads

$$\|v\|_2 = \frac{1}{N} \left( \sum_{i=1}^N v_i^2 \right)^{1/2}, \quad (37)$$

where  $v_i$  are the components of  $v$ .

### 3.6. Neural network architectures and hyperparameters

For both DDM and CoSTA-based HAM, we use DNNs with the fully connected neural network (FCNN) architecture illustrated in Fig. 4, and the hyperparameters listed in Table 3. The optimal set of hyperparameters were found using a grid search for the DDM and then applied to HAM as well. The number of fully connected (FC) layers, the width of each FC layer and the learning rate were chosen such as to minimize the total validation loss obtained for the two manufactured solutions A and B at the bottom of Table 3 in a parameter grid search. The other hyperparameters values were chosen based on prior experience with DNN tuning. We make no claim that our architecture choices are optimal. To the contrary, we expect that using DNN architectures specifically constructed for time series problems (e.g. LSTM networks) would improve DDM and HAM performance. However, in the present work, we have chosen to use a simple DNN such that its technical details would not obscure our main contribution, which is the formal introduction and justification of CoSTA. Therefore, comparisons of different architectures and hyperparameters are deferred to future research, as will be touched upon in Section 5.

---

**Algorithm 1:** Data generation procedure for the training and validation datasets. All training data is normalized to have a component-wise mean of 0 and standard deviation of 1. The validation data is normalized using the same normalization coefficients as the training set. For the experiments where modeling error is synthesized, we set  $\hat{q} = 0$  in Eqs. (33) and (29).

---

Define grid nodes  $\mathbf{x}$  and time levels  $t^0, t^1, \dots, t^{N_t-1}$ .

```

for  $T(\mathbf{x}, t; \alpha)$  in Table 1 do
  for  $\alpha$  in  $\mathcal{A}_{\text{train}}$  or  $\mathcal{A}_{\text{val}}$  do
    Initialize:  $\mathbf{T}_{\text{ref}}^0 = T(\mathbf{x}, t^0; \alpha)$ 
    for  $n = 1, 2, \dots, N_t - 1$  do
      Get DDM input by sampling:  $\mathbf{T}_{\text{ref}}^{n-1} \leftarrow T(\mathbf{x}, t^{n-1}; \alpha)$ .
      Get HAM input  $\tilde{\mathbf{T}}_h^n$  by inserting  $\mathbf{T}_{\text{ref}}^{n-1}$  into Eq. (33).
      Get DDM target output by sampling:
         $\mathbf{T}_{\text{ref}}^n \leftarrow T(\mathbf{x}, t^n; \alpha)$ .
      Get HAM target output  $\hat{\sigma}^n$  using  $\mathbf{T}_{\text{ref}}^n$  and  $\mathbf{T}_{\text{ref}}^{n-1}$  in
        Eq. (36).
      Store  $(\mathbf{T}_{\text{ref}}^{n-1}, \tilde{\mathbf{T}}_h^n, \mathbf{T}_{\text{ref}}^n, \hat{\sigma}^n)$  as an individual data
        example in the appropriate dataset.
    end
  end
  Normalize all data examples in both datasets.
end

```

---

**Algorithm 2:** Testing algorithm used in our numerical experiments. Note that, to match the operating conditions seen during training, all DNN inputs/outputs are normalized/unnormlized using the same normalization coefficients as were used to normalize the training and validation data (cf. Algorithm 1). For the experiments where modeling error is synthesized, we set  $\hat{q} = 0$  in Eqs. (27), (33) and (29).

Define grid nodes  $\mathbf{x}$  and time levels  $t^0, t^1, \dots, t^{N_t-1}$ .

for  $T(x, t; \alpha)$  in top part of Table 1 do

for  $\alpha$  in  $\mathcal{A}_{\text{test}}$  do

Initialize:  $\mathbf{T}_{\text{ref}}^0 = \mathbf{T}_p^0 = \mathbf{T}_d^0 = \mathbf{T}_h^0 = T(\mathbf{x}, t^0; \alpha)$

for  $n = 1, 2, \dots, N_t - 1$  do

Sample manufactured solution:  $\mathbf{T}_{\text{ref}}^n = T(\mathbf{x}, t^n; \alpha)$ .

Get PBM prediction  $\mathbf{T}_p^n$  by inserting  $\mathbf{T}_p^{n-1}$  into Eq. (27).

Get HAM DNN input  $\tilde{\mathbf{T}}_h^n$  by inserting  $\mathbf{T}_h^{n-1}$  into Eq. (33).

Get DDM prediction:  $\mathbf{T}_d^n \leftarrow \text{DNN}_T(\mathbf{T}_d^{n-1})$

$\hat{\sigma}_{\text{NN}}^n \leftarrow \text{DNN}_\sigma(\tilde{\mathbf{T}}_h^n)$

Get HAM prediction  $\mathbf{T}_h^n$  by inserting  $\mathbf{T}_h^{n-1}$  and  $\hat{\sigma}_{\text{NN}}^n$  into Eq. (29).

Relative  $l_2$  error of PBM prediction  $\leftarrow$

$$\frac{\|\mathbf{T}_p^n - \mathbf{T}_{\text{ref}}^n\|_2}{\|\mathbf{T}_{\text{ref}}^n\|_2}$$

Relative  $l_2$  error of DDM prediction  $\leftarrow$

$$\frac{\|\mathbf{T}_d^n - \mathbf{T}_{\text{ref}}^n\|_2}{\|\mathbf{T}_{\text{ref}}^n\|_2}$$

Relative  $l_2$  error of HAM solution  $\leftarrow$

$$\frac{\|\mathbf{T}_h^n - \mathbf{T}_{\text{ref}}^n\|_2}{\|\mathbf{T}_{\text{ref}}^n\|_2}$$

end

end

end

**Table 3**  
Hyperparameters of the DNN used in the present work.

Parameter	Value
Loss function	MSE
Learning rate	1e−5
Optimizer	Adam
Batch size	32
Number of hidden FC layers	4
Hidden FC layer width	80
LeakyReLU slope	0.01
Validation period	1e2
Overfit limit	20

## 4. Results and discussion

In each numerical experiment, we consider one of the manufactured solutions listed in the top section of Table 1. In any given experiment, we aim to reproduce the time series  $\{\mathbf{T}_{\text{ref}}^n\}_{n=0}^{N_t-1}$  for each  $\alpha \in \mathcal{A}_{\text{test}}$  using our PBM, DDM and HAM models. To quantify the models' performance, we present the temporal development of the relative  $l_2$ -errors,  $\frac{\|\mathbf{T}_p^n - \mathbf{T}_{\text{ref}}^n\|_2}{\|\mathbf{T}_{\text{ref}}^n\|_2}$ ,  $\frac{\|\mathbf{T}_d^n - \mathbf{T}_{\text{ref}}^n\|_2}{\|\mathbf{T}_{\text{ref}}^n\|_2}$ , and  $\frac{\|\mathbf{T}_h^n - \mathbf{T}_{\text{ref}}^n\|_2}{\|\mathbf{T}_{\text{ref}}^n\|_2}$ , of the PBM-, DDM- and HAM predictions with respect to the sampled manufactured solution  $\mathbf{T}_{\text{ref}}^n$ . We also present the temperature profiles predicted by PBM, DDM and HAM at the final time level  $n = N_t - 1$  (corresponding to  $t = 5.0$  s) alongside the manufactured solution  $T(\mathbf{x}, t^{N_t-1}; \alpha)$ .

The boundary conditions and initial conditions corresponding to the manufactured solutions are assumed known and utilized as applicable for all three model types in all experiments. However, in order to synthesize scenarios where some relevant physics are unknown, for the experiments discussed in Section 4.2, it is

assumed that we do not know the source terms  $\hat{q}$  required for the manufactured solutions to satisfy Eq. (23). No such limitation is imposed on the models discussed in Section 4.1.

### 4.1. Experiments without modeling error

In this section, we compare the performance of the three approaches PBM, DDM and HAM in situations where we have full knowledge of the physics. This implies that the structure of the governing equation, including the source term and the exact values of the PBM parameters, are fully known. The following sections present the results for interpolation and extrapolation scenarios for Solutions 0 and 3 (cf. Table 1). For the current experiment using Solution 3, we used a finer spatial discretization than elsewhere in this work ( $N_j = 200$ , rather than  $N_j = 20$ ) in order to demonstrate the full power PBM when all physics is known.

#### 4.1.1. Interpolation scenarios

Figs. 5 and 6 present the results for the two interpolation cases corresponding to Solutions 0 and 3. From the figures, it is clear that the predictions of the PBM are better than those of the DDM in both the cases. The errors associated with the PBM can be fully attributed to the discretization error, which is especially small for Solution 3 where a finer spatial grid was used. For both solutions, HAM improves on the accuracy of PBM by compensating for the discretization error, even when the discretization error is small.

#### 4.1.2. Extrapolation scenarios

Figs. 7 and 8 present the results for the two extrapolation cases corresponding to Solutions 0 and 3. As expected, DDM fails to generalize to both the extrapolation cases, while PBM exhibits similar level of accuracy for both interpolation and extrapolation. HAM generalizes better than DDM, and yields high accuracy in both extrapolation cases. However, HAM does not significantly outperform PBM here.

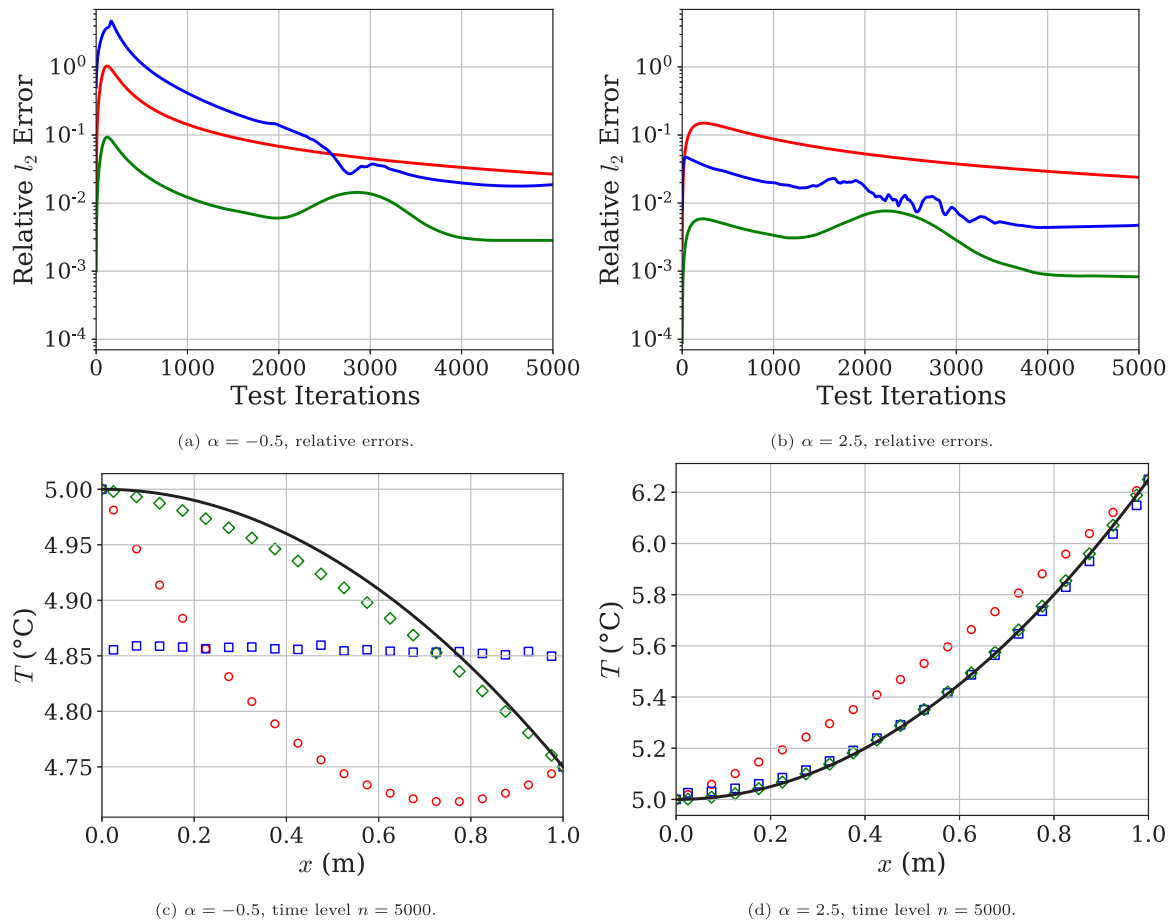
In conclusion, PBM generalizes better than DDM, while HAM generalizes approximately as well as PBM in the presence of numerical error. This ability of the HAM to correct for the numerical error can be exploited to relax the mesh requirement needed for solving equations at a fixed error tolerance. That is, equations can be solved on a coarse mesh in a computationally efficient manner, and CoSTA-based HAM can be used to correct for the numerical errors resulting from the coarse discretization.

### 4.2. Experiments with modeling error

In this section we present the results corresponding to manufactured solutions 1–4, given in Table 1. The results corresponding to interpolation are presented first, followed by the results corresponding to extrapolation.

#### 4.2.1. Interpolation scenarios

As can be seen from the bottom halves of Figs. 9–12, the predictions of CoSTA-based HAM are qualitatively correct for all interpolation scenarios. DDM also provides qualitatively correct predictions for Solutions 1, 2 and 4 (cf. Figs. 9, 10 and 12), but its final-time-level predictions for Solution 3 deviate significantly from the reference profiles (cf. Fig. 11). Furthermore, the relative errors of the HAM predictions are consistently more than one order of magnitude lower than those of the DDM predictions, as shown in the top halves of Figs. 9–12; the difference in accuracy is particularly striking for Solution 2 (cf. Fig. 10). Since HAM and DDM both utilize the same DNN setup and training routine, the significant performance difference can only be explained by the utilization of PBM inside HAM. As discussed previously, the



**Fig. 13.** Solution 1, extrapolation: Comparison of relative errors and final temperature profiles for  $\alpha = -0.5, 2.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). All models perform reasonably well for  $\alpha = 2.5$ , but only HAM provides qualitatively correct predictions for  $\alpha = -0.5$ .

combined use of a PBM and a DNN in CoSTA-based HAM allows some relevant physics to be accounted for by the PBM, thereby making the learning task of the DNN easier than in the DDM case where *all* relevant physics must be learnt by the DNN. It is interesting to note that HAM clearly benefits from the PBM in spite of the PBM yielding low quality predictions on its own; the PBM predictions are qualitatively incorrect for Solutions 2 and 3 (cf. Figs. 10 and 11), and their errors are roughly one order of magnitude larger than the corresponding DDM errors for Solutions 1 and 4 (cf. Figs. 9 and 12). Another interesting observation is that the DDM and HAM error curves are less smooth than those of PBM, suggesting that the DNNs inject some pseudo-random noise into the DDM and HAM predictions. This noise could possibly be reduced by informing the DNNs of the temporal correlation between data at subsequent time levels through input (e.g. use two subsequent profiles as input, rather than just a single profile), using a different DNN architecture (e.g. LSTM architecture) and/or a different training regime (e.g. perform adversarial training with a temporal discriminator).

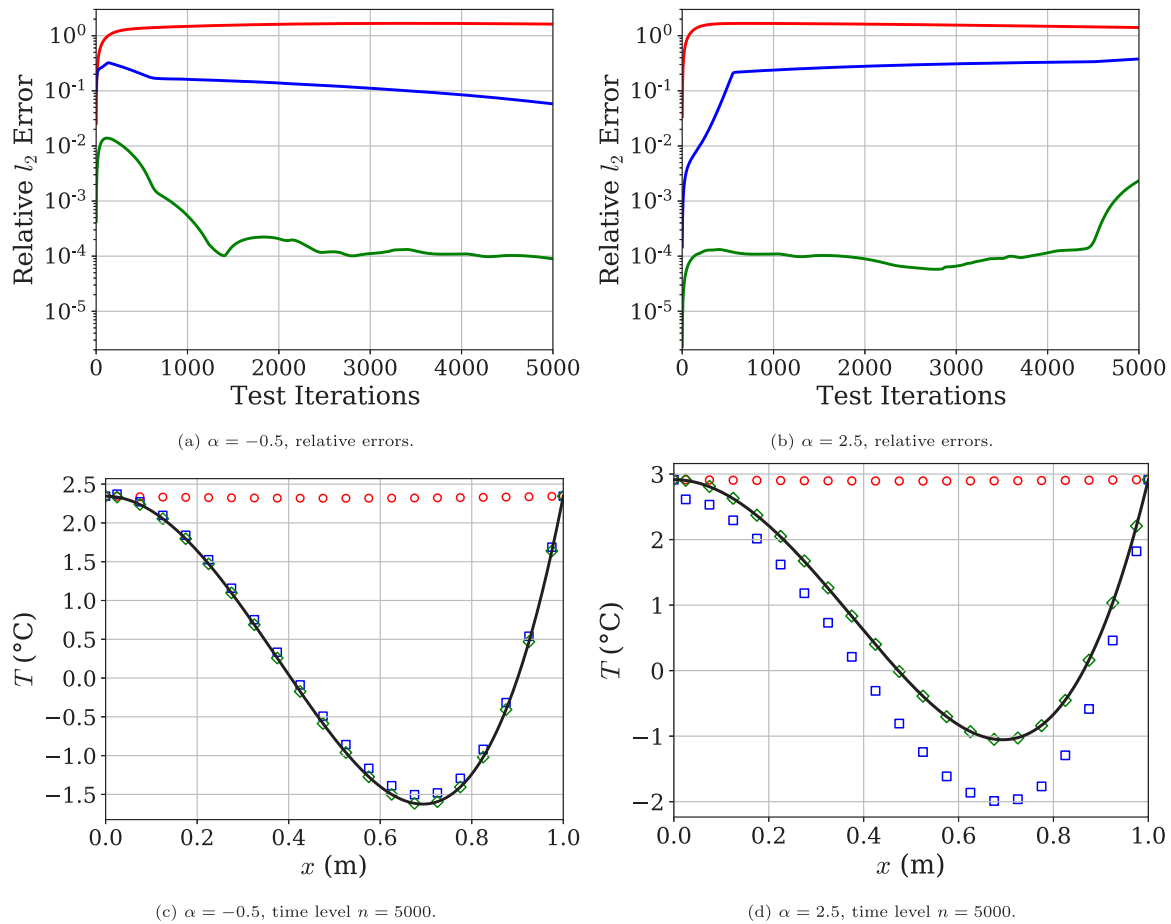
#### 4.2.2. Extrapolation scenarios

Owing to its well-known generalizability, the PBM performs similarly in the extrapolation scenarios as in the interpolation scenarios. However, as the accuracy of PBM was poor in the interpolation scenarios due to significant modeling error, the PBM results shown in Figs. 13–16 are also poor in general. DDM performs better than PBM, providing qualitatively correct predictions for Solutions 2 and 4 (cf. Figs. 14 and 16). The better performance

of DDM in comparison to the PBM can be attributed to the fact that DDM, though generally poor in generalization, still learns about the nature of the unknown source term from the data. However, in terms of accuracy, HAM significantly outperforms DDM for both these manufactured solutions. HAM also outperforms DDM for Solution 1, where the DDM prediction for  $\alpha = -0.5$  is qualitatively incorrect while the HAM prediction is not (cf. Fig. 13(c)). Furthermore, HAM also outperforms both PBM and DDM for Solution 3 with  $\alpha = 2.5$  (cf. Fig. 15(b)). The remaining scenario – Solution 3 with  $\alpha = -0.5$  – proves to be the most difficult scenario considered, as all three models fail to provide qualitatively correct predictions for this scenario (cf. Fig. 15(c)). Thus, while CoSTA-based HAM generally performs well in our experiments, the results for Solution 3 with  $\alpha = -0.5$  show that significant improvements are still possible and desirable. In addition to ensuring temporal coherence, as previously discussed, utilizing problem-specific data augmentation techniques might be one way of boosting HAM’s performance. Another possibility is to use a more accurate PBM as a basis for the full CoSTA model which will result in more relevant physics being accounted for by the PBM, thereby simplifying the learning task of the DNN.

### 5. Conclusion and future work

This paper has introduced the Corrective Source Term Approach (CoSTA) to hybrid analysis and modeling (HAM). The method utilizes a deep neural network (DNN) to generate a corrective source term that augments the discretized governing



**Fig. 14.** Solution 2, extrapolation: Comparison of relative errors and final temperature profiles for  $\alpha = -0.5, 2.5$  (— Exact,  $\circ$  PBM,  $\diamond$  DDM,  $\diamond$  HAM). HAM's predictions are (mostly) three orders of magnitude more accurate than those of DDM. PBM's predictions are qualitatively incorrect.

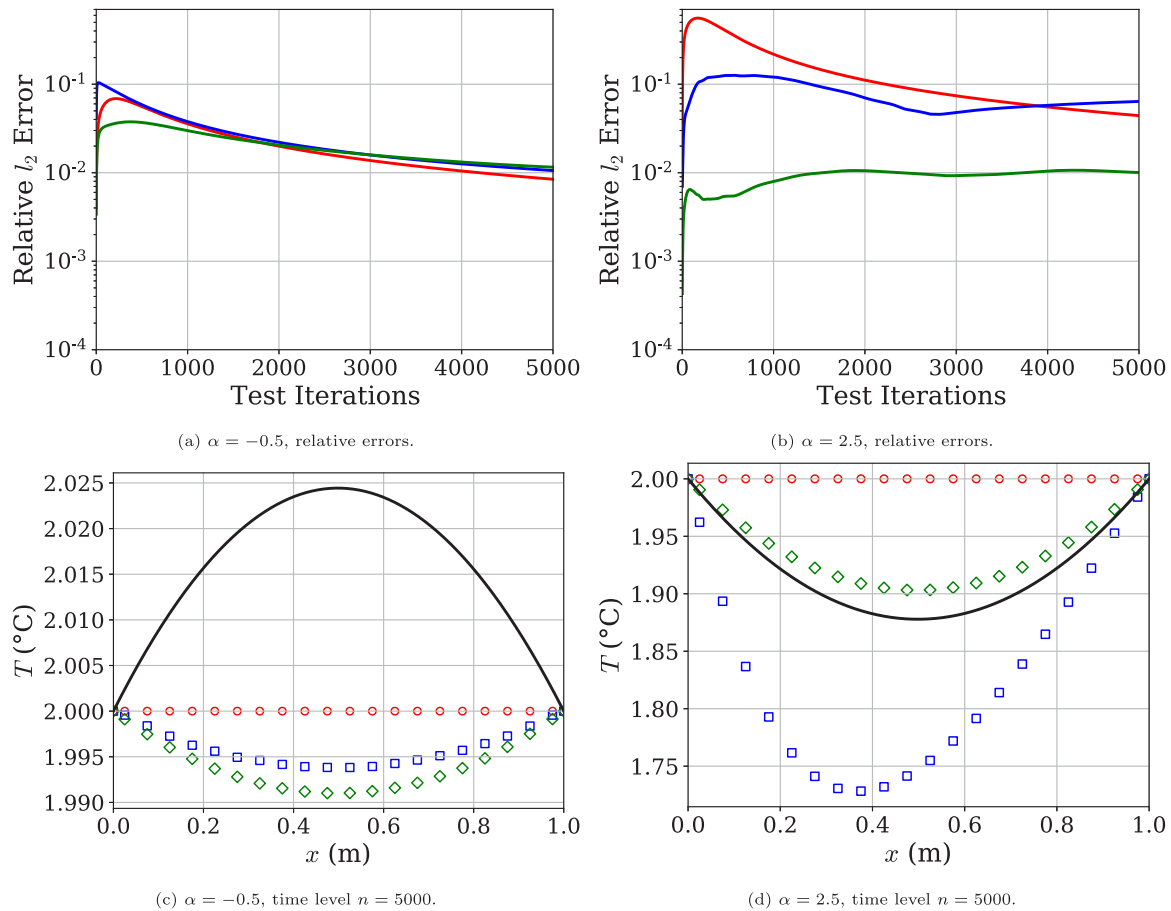
equation of a physics-based model (PBM). The purpose of the corrective source term is to account for physics that is unresolved by the PBM. In a series of numerical experiments on one-dimensional heat diffusion problems, we have compared the performance of CoSTA-based HAM with that of the uncorrected PBM and that of a comparable data-driven model (DDM). The most important conclusions from the study are as follows:

- In the absence of any modeling error, i.e., when the full physics were known, the PBM was more accurate than the DDM. The PBM also generalized well to the extrapolation scenarios, while the DDM did not. On the other hand, HAM improved on the accuracy of the PBM in the interpolation scenarios, while still performing better or very close to the PBM in the extrapolation scenarios. The accuracy improvements of HAM can be attributed to its ability to compensate for the numerical discretization error. This ability can be exploited to accelerate numerical solvers by enabling the use of coarse meshes while retaining accuracy, as unresolved subgrid-scale physics can be accounted for by the corrective source term. This feature of HAM will be extremely useful in digital twin applications as accurate and real-time performance of the models is of utmost importance.
- In the presence of modeling error i.e., when some relevant physics was unknown, the PBM failed to make reasonable predictions. The DDM generally outperformed the PBM for both interpolation and extrapolation scenarios, since the DDM could learn about the unknown physics from data

observations. However, HAM consistently performed better than both PBM and DDM, often decreasing the relative error of the predictions by several orders of magnitude. The improvement in predictions compared to DDM was even more pronounced for extrapolation scenarios; in other words, HAM was found to be more generalizable to new, unseen scenarios. In the context of digital twin applications, it is foreseen that new scenarios never witnessed before could occur from time to time, and HAM will be able to deal with such situation better than PBM and DDM.

The work presented herein clearly illustrates the great potential that can be unlocked by combining PBM and DDM in a HAM framework like CoSTA. While we have only considered a single PBM and a single DNN architecture for modeling a single class of physical systems, CoSTA can easily be used with other PBMs and DNNs to model other systems as well – including multidimensional systems and systems with multiple governing equations – due its solid but flexible theoretical foundation. Thus, the numerous benefits of employing a HAM framework like CoSTA, which we identify below, can greatly increase the applicability of data-driven techniques in a variety of applications, including digital twins and also high-stakes engineering applications currently reserved for pure PBM. Notable examples include improving the dynamic model of a system (like a vehicle or a process), improving the accuracy of building energy simulation models using regular measurements, modeling complex physical phenomena like turbulence whose full physical understanding is still limited,





**Fig. 15.** Solution 3, extrapolation: Comparison of relative errors and final temperature profiles for  $\alpha = -0.5, 2.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). HAM is most accurate for  $\alpha = 2.5$ , while all methods fail to provide qualitatively correct predictions for  $\alpha = -0.5$ .

accelerating high fidelity numerical solvers, and stabilizing ROMs based on significant dimensionality reduction.

One important benefit is that HAM aims to utilize existing knowledge to the greatest extent possible, such that, for any given problem, HAM requires less complex data-driven techniques than pure DDM. This can improve the predictability and interpretability of the DDM, thereby making the model more trustworthy. Another related benefit is that the DNN-generated corrective source term of CoSTA can be explained and analyzed within the framework of a PBM based on known first principles. This allows for an inbuilt sanity check for the data-driven part of the HAM model. For example, in the case of heat diffusion modeling, if there exist known bounds for the energy transferred to a system, then the DNN-generated source term must also be bounded, and any violation of these bounds can be automatically detected. The lack of such automatic DNN-misbehavior detection in pure DDM is one of the reasons why the black-box nature of DNNs has inhibited the acceptance of data-driven techniques in high-stakes applications. As for digital twins, CoSTA facilitates the use of coarse meshes in numerical solvers through its ability to correct both modeling errors and discretization errors, thereby improving computational efficiency while retaining accuracy. Furthermore, since the learning of the source term can continue even after model deployment, CoSTA allows for any new phenomena encountered by the digital twin to be learned automatically. Thus, CoSTA has the potential to develop models which will be generalizable, trustworthy, accurate and computationally efficient, and self-evolving. However, while our results are promising, more research is required with regard to the accuracy, stability and

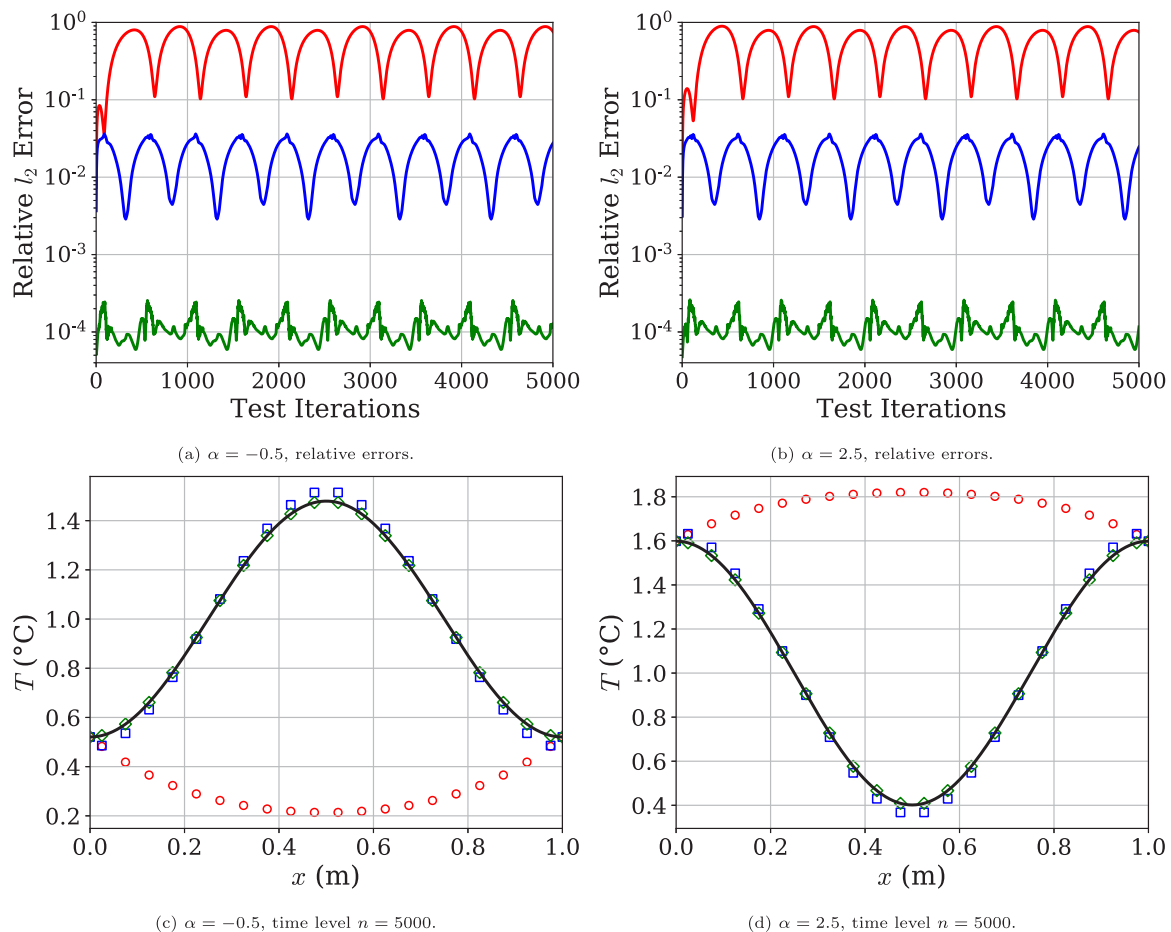
interpretability of the learned source term, such as to ensure that the findings of this paper generalize to other physical systems and processes. Making optimal choices for PBMs, as well as for DNN architectures and training regimes, are other interesting research directions. Some of the outlined improvements and extensions will be the topic of future research by the authors, but we hope that this paper will inspire other scientists to pursue the outlined research directions as well.

#### 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.

#### Acknowledgments

The second and third authors are grateful for the support received by the Research Council of Norway and the industrial partners of the following projects: EXAIGON, Norway–*Explainable AI systems for gradual industry adoption* (Grant No. 304843), *Hole cleaning monitoring in drilling with distributed sensors and hybrid methods*, Norway (Grant No. 308823), and RaPiD, Norway–*Reciprocal Physics and Data-driven models* (Grant No. 313909). The fourth author gratefully acknowledges the Early Career Research Program (ECRP) support of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC0019290.



**Fig. 16.** System 4, extrapolation: Comparison of relative errors and final temperature profiles for  $\alpha = -0.5, 2.5$  (— Exact,  $\circ$  PBM,  $\square$  DDM,  $\diamond$  HAM). Both DDM and HAM provide qualitatively correct predictions, while PBM does not. HAM is significantly more accurate than DDM.

## References

- Afebu, K. O., Liu, Y., Papatheou, E., & Guo, B. (2021). LSTM-based approach for predicting periodic motions of an impacting system via transient dynamics. *Neural Networks*, 140, 49–64. <http://dx.doi.org/10.1016/j.neunet.2021.02.027>, URL <https://www.sciencedirect.com/science/article/pii/S0893608021000770>.
- Akhtar, N., & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *CoRR*, arXiv:1801.00553, arXiv:1801.00553, URL <http://arxiv.org/abs/1801.00553>.
- Bai, Z., & Zhang, X.-L. (2021). Speaker recognition based on deep learning: An overview. *Neural Networks*, 140, 65–99. <http://dx.doi.org/10.1016/j.neunet.2021.03.004>, URL <https://www.sciencedirect.com/science/article/pii/S0893608021000848>.
- Blakseth, S. S. (2021). *Introducing CoSTA: A Deep Neural Network Enabled Approach to Improving Physics-Based Numerical Simulations* (Master's thesis), NTNU.
- Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15), 3932–3937. <http://dx.doi.org/10.1073/pnas.1517384113>, arXiv:https://www.pnas.org/content/113/15/3932.full.pdf, URL <https://www.pnas.org/content/113/15/3932>.
- Bu, F., & Wang, X. (2019). A smart agriculture IoT system based on deep reinforcement learning. *Future Generation Computer Systems*, 99, 500–507.
- Chakraborty, S., Adhikari, S., & Ganguli, R. (2021). The role of surrogate models in the development of digital twins of dynamic systems. *Applied Mathematical Modelling*, 90, 662–681.
- Fonn, E., Brummelen, H. v., Kvamsdal, T., & Rasheed, A. (2019). Fast divergence-conforming reduced basis methods for steady Navier–Stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 346, 486–512. <http://dx.doi.org/10.1016/j.cma.2018.11.038>.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press, <http://www.deeplearningbook.org>.
- Hartmann, D., Herz, M., & Wever, U. (2018). Model order reduction a key technology for digital twins. In *Reduced-order modeling (ROM) for simulation and optimization* (pp. 167–179). Cham, Switzerland: Springer.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507. <http://dx.doi.org/10.1126/science.1127647>.
- Kapteyn, M. G., Pretorius, J. V., & Willcox, K. E. (2021). A probabilistic graphical model foundation for enabling predictive digital twins at scale. *Nature Computational Science*, 1(5), 337–347.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- Lee, Y.-J., Hall, D., Liu, Q., Liao, W.-W., & Huang, M.-C. (2021). Interpretable tropical cyclone intensity estimation using dvorak-inspired machine learning techniques. *Engineering Applications of Artificial Intelligence*, 101, Article 104233. <http://dx.doi.org/10.1016/j.engappai.2021.104233>, URL <https://www.sciencedirect.com/science/article/pii/S0952197621000804>.
- Liu, L., Cao, Z., Shen, T., Zhang, L., & Zhang, L. (2021). Experimental and numerical investigation on flow and heat transfer characteristics of a multi-waves internally spiral finned tube. *International Journal of Heat and Mass Transfer*, 172, Article 121104. <http://dx.doi.org/10.1016/j.ijheatmasstransfer.2021.121104>, URL <https://www.sciencedirect.com/science/article/pii/S0017931021002076>.
- Liu, Z., Yao, C., Yu, H., & Wu, T. (2019). Deep reinforcement learning with its application for lung cancer detection in medical internet of things. *Future Generation Computer Systems*, 97, 1–9.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ..., Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. arXiv:1912.01703 [Cs, Stat], URL <http://arxiv.org/abs/1912.01703>, arXiv:1912.01703.
- Pawar, S., San, O., Aksoylu, B., Rasheed, A., & Kvamsdal, T. (2021). Physics guided machine learning using simplified theories. *Physics of Fluids*, 33(1), Article 011701.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707.

- Rasheed, A., San, O., & Kvamsdal, T. (2020). Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8, 21980–22012. <http://dx.doi.org/10.1109/ACCESS.2020.2970143>.
- Roache, P. J. (2002). Code verification by the method of manufactured solutions. *Journal of Fluids Engineering*, 124(1), 4–10.
- San, O., Rasheed, A., & Kvamsdal, T. (2021). Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. *GAMM-Mitteilungen*, 44, Article e202100007.
- Siddiqui, M. S., Rasheed, A., Tabib, M., & Kvamsdal, T. (2019). Numerical investigation of modeling frameworks and geometric approximations on NREL 5 MW wind turbine. *Renewable Energy*, 132, 1058–1075.
- Szegedy, C., Ioffe, S., & Vanhoucke, V. (2016). Inception-v4, inception-ResNet and the impact of residual connections on learning. *CoRR*, arXiv:1602.07261, arXiv:1602.07261, URL <http://arxiv.org/abs/1602.07261>.
- Taylor, C. A., Hughes, T. J., & Zarins, C. K. (1998). Finite element modeling of blood flow in arteries. *Computer Methods in Applied Mechanics and Engineering*, 158(1), 155–196. [http://dx.doi.org/10.1016/S0045-7825\(98\)80008-X](http://dx.doi.org/10.1016/S0045-7825(98)80008-X), URL <https://www.sciencedirect.com/science/article/pii/S004578259880008X>.
- Vaddireddy, H., Rasheed, A., Staples, A. E., & San, O. (2020). Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensors. *Physics of Fluids, Editor's Pick*, 32, Article 015113. <http://dx.doi.org/10.1063/1.5136351>.
- Willard, J., Jia, X., Xu, S., Steinbach, M., & Kumar, V. (2020). Integrating physics-based modeling with machine learning: A survey. arXiv preprint arXiv:2003.04919.
- Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., & Jain, A. (2019). Adversarial attacks and defenses in images, graphs and text: A review. arXiv:1909.08072.
- Yuan, X., He, P., Zhu, Q., Bhat, R. R., & Li, X. (2017). Adversarial examples: Attacks and defenses for deep learning. *CoRR*, arXiv:1712.07107, arXiv:1712.07107, URL <http://arxiv.org/abs/1712.07107>.
- Zhang, H., & Wang, D. (2021). Deep ANC: A deep learning approach to active noise control. *Neural Networks*, 141, 1–10. <http://dx.doi.org/10.1016/j.neunet.2021.03.037>, URL <https://www.sciencedirect.com/science/article/pii/S0893608021001258>.