Computer Methods in Applied Mechanics and Engineering 432 (2024) 117402

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Statistical variational data assimilation

Amina Benaceur [a], Barbara Verfürth [b,*]

[a] *Institut für Angewandte und Numerische Mathematik, Karlsruher Institut für Technologie, Englerstr. 2, 76131, Karlsruhe, Germany*
[b] *Institut für Numerische Simulation, Universität Bonn, Friedrich-Hirzebruch-Allee 7, 53115, Bonn, Germany*

A R T I C L E I N F O

MSC:
68T07
68T20
65K10

Keywords:
Neural networks
Deep learning
Data assimilation
PBDW

A B S T R A C T

This paper is a contribution in the context of variational data assimilation combined with statistical learning. The framework of data assimilation traditionally uses data collected at sensor locations in order to bring corrections to a numerical model designed using knowledge of the physical system of interest. However, some applications do not have available data at all times, but only during an initial training phase. Hence, we suggest to combine data assimilation with statistical learning methods; namely, deep learning. More precisely, for time steps at which data is unavailable, a surrogate deep learning model runs predictions of the 'true' data which is then assimilated by the new model. In this paper, we also derive *a priori* error estimates on this statistical variational data assimilation (SVDA) approximation. Finally, we assess the method by numerical test cases.

1. Introduction

State estimation is a task in which the quantity of interest is the 'true' state u^{true} of a physical system over a space or space–time domain of interest. In general, numerical prediction using mathematical models based on the physical knowledge of a system may be deficient due to limitations imposed by available knowledge. Data assimilation (DA) has the goal to overcome these limitations and produce more accurate predictions by incorporating experimental observations in numerical models. For this reason, data assimilation methods have been widely explored in the literature. The goal of these methods is to use *a priori* information to deduce the best mathematical model, while using available experimental data to produce the most accurate approximation of a physical system. Many data assimilation methods involve the minimization of a cost function. As opposed to statistical techniques such as Bayesian data assimilation [1] or Kalman filtering [2], variational data assimilation more explicitly uses the mathematical (variational) structure of the problem to formulate the data assimilation method. For instance, the widely studied methods 3D-VAR and 4D-VAR both usually rely on the weak formulation of the PDE to define the constrained minimization problem. Like many variational data assimilation methods, one of the drawbacks of 3D-VAR and 4D-VAR is their computational intrusivity, which means that at any stage, computational procedures need to access the model in order to perform their calculations. Intrusivity is very inconvenient in many contexts, for instance when using industrial high-fidelity black-box solvers. Hence, non-intrusive or partially-intrusive options can be valuable. The parametrized background data weak (PBDW) method introduced in [3] is closely related to and inspired by the 3D-VAR, but is non-intrusive in the above discussed sense. It has been studied in many further works, with the presence of noise [4,5], and in time-dependent contexts [6,7]. Another drawback of variational data assimilation is that it assumes that data is available at all times, which is not the case in many industrial contexts. For instance, in numerical weather prediction, data is collected using weather balloons to be sent at predefined times. Another common industrial framework is that

* Corresponding author.
E-mail address: verfuerth@ins.uni-bonn.de (B. Verfürth).

https://doi.org/10.1016/j.cma.2024.117402
Received 12 February 2024; Received in revised form 4 September 2024; Accepted 16 September 2024
Available online 3 October 2024

in which data collection campaigns are conducted during limited periods of time, whereas numerical investigation of the physical phenomena of interest may be carried out during longer periods.

The key conceptual idea in this contribution is to use machine learning (ML) in combination with data assimilation in the above described context of partially available real data. In the literature, other works bring data assimilation and statistical learning methods together. Equivalences between data assimilation and machine learning are discussed in the review paper [8], which provides a review of existing methods in the context of earth systems. In [9], a DA-ML method is suggested to constrain the output for mass conservation. The chosen data assimilation method uses Ensemble Kalman filtering, and statistical learning is performed *via* a convolutional neural network. Moreover, [10] suggests an offline DA-ML method, with iterative application of Ensemble Kalman filtering and convolutional neural networks. The goal is to estimate the state at the current step given the state at the previous one. The core idea is to derive a convolutional neural network trained using data assimilation during the offline stage. Since data assimilation is only used offline, the resulting online processing is of statistical nature. Regarding the use of variational methods for data assimilation, [11] explores two ideas. It compares the performance of DA-ML for resolvent correction and tendency correction using a modified 4D-VAR formulation. The offline training uses the DA-ML method introduced in [10]. An online training of the neural network is also explored in [11].

In this paper, we try to overcome the data-availability concern by combining data assimilation with statistical prediction. As main contribution, we introduce a novel concept called statistical variational data assimilation (SVDA). It is split into two stages: an offline stage during which we train a deep learning method, and an online stage during which we run our data assimilation model, with real observations replaced by their statistically predicted counterparts. This main idea together with the necessary notation is explained in Section 2. As a conceptual idea, SVDA offers a lot of flexibility in the choice of the data assimilation and especially the deep learning method, which should be chosen adapted to the considered use case. To illustrate how a concrete SVDA could look like in detail, we present an exemplary data assimilation and a deep learning method in Section 3. Specifically, we use the PBDW approach to circumvent the intrusivity of the classical data assimilation methods. For the deep learning, we choose a so-called LSTM-RNN which has proven good time prediction capabilities in the literature. In Section 4, we present an error analysis of the method. Finally, in Section 5, the SVDA is illustrated by some numerical results.

2. Statistical variational data assimilation

In this context, we define a so-called ‘best-knowledge’ model (bk) as the best possible model established by human expertise. The bk model is a mathematical model rendering the behavior of the system given all available knowledge about a physical system. More concretely, we will use a PDE or a set of PDEs that best models the physical problem.

In the applications we have in mind, variational data assimilation methods typically incorporate observations into so-called forecasts obtained by existing bk models. Data/Observations are collected, e.g. using sensors, and integrated into the bk model at selected so-called update times, finally building a data-driven model. Having formulated the bk model and set the sensor locations, variational data assimilation typically consists of the following two steps which are done for every new calculation (e.g., new time instance or parameter):

- Collect observations at sensor locations
- Solve the data assimilation problem using the collected observations as input

However, ready-to-use data is not always available. In fact, a common engineering scenario is that in which measurements/observations are available for a given time interval or for given parametrizations explored during a training phase. This training phase is more commonly referred to as a ‘test campaign’ in industrial and engineering scenarios. Thus, it is of great interest to develop appropriate and optimal use of the collected observations to understand the behavior of the system.

In this section, we introduce a new method to overcome the issue of unavailable observations. More precisely, we do not deal with cases where some observations have been collected inaccurately or are missing [12]. We rather deal with situations in which data has been collected for given time windows but is not available at later times. Towards this end, we suggest to use prediction-based methods in order to approximate the observations. The predicted observations will then be plugged into a data assimilation framework as a surrogate to real-time data. In this paper, we use the Parametrized Background Data-Weak approach (PBDW) as a variational data assimilation method.

As mentioned, we need to perform a statistical learning step in the SVDA. The goal of this step is to predict the alternative data needed for the data assimilation problem using chosen input variables. Many methods can be used depending on which inputs will be used to train the model. As an example, the behavior of linear phenomena can be predicted using linear regression [13], and nonlinear phenomena can be addressed using polynomial regression, or neural networks [14]. In this paper, the phenomena of interest are time-dependent and nonlinear. Hence, in the presented framework, the SVDA relies on a type of Neural Networks (NN) called Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN). Yet, the method can be run using any other deep learning or machine learning method [15]. The choice should be made in light of the intricacies of the problem of interest.

2.1. Main ideas

SVDA can be used in different contexts. This paper mainly focuses on ‘future predictions’, i.e., on generating machine learning observations from a bk model and given observations collected at previous times. The hope is to thereby improve the ‘future’ simulations where real observations are not available. Another configuration for the use of the SVDA will be considered in the numerical test cases; namely SVDA in parametric contexts. In that case, the underlying model is a parametrized PDE and observations are available for one or a few distinct parameters. The machine learning model is trained only on these data, but its 'predictions' for the model with a different parameter can be used as surrogate data in the data assimilation process for the model with this new parameter. Independent of the application case, the SVDA can algorithmically be divided into an offline and an online phase. Let us now describe the main conceptual steps of both stages in the context of 'future predictions'.

Consider a finite time interval $I = [0, T]$, with $T > 0$. To discretize in time, we consider an integer $K \geq 1$, we define $0 = t^0 < \cdots < t^K = T$ as $(K + 1)$ distinct time nodes over I, and we set $\mathbb{K}^{\text{tr}} = \{1, \ldots, K\}$, $\overline{\mathbb{K}}^{\text{tr}} = \{0\} \cup \mathbb{K}^{\text{tr}}$ and $I^{\text{tr}} = \{t^k\}_{k \in \overline{\mathbb{K}}^{\text{tr}}}$. The aim is to derive a state estimate for a time-dependent solution.

We assume the existence of an initial offline phase $[0, \Delta t]$ of width $\Delta t > 0$, and an integer $k_{\text{off}} > 0$ such that $\Delta t = t^{k_{\text{off}}-1}$. In other words, k_{off} is the first index at which the observations are unavailable. During the *offline stage* of SVDA, we use the data available in the time window $[0, \Delta t]$ to predict the evolution of the observables via a deep learning model (LSTM-RNN). We emphasize that we learn the observations and not the state estimates directly with the neural network for two main reasons. First, state estimates of the true solution (and not the bk model) are typically never directly accessible, even in the offline time window, in real applications. Only the data collected at sensor locations can thus be used to train the neural network. Second, this approach of producing synthetic observations aligns better with the traditional data assimilation ansatz where observations are the input of the data assimilation problem. Therefore, this allows us to easily incorporate the SVDA concept into traditional, potentially commercial, data assimilation software. The offline stage also consists of two additional steps: building a background space $\mathcal{Z}_N$ of dimension N, and building an observable space $\mathcal{U}_M$ of dimension M. These spaces respectively model the solution (state) space for the bk model (e.g. the PDE of interest) and the space used to capture the observations. Additional details on the construction of such spaces will be given in Section 3 below.

Using the statistical (deep learning) LSTM-RNN model, data for future times (starting at time index k_{off}) is predicted for all sensor locations in the *online stage* of SVDA. Despite the absence of real data, it now becomes possible to perform a more informed simulation for the next time steps without relying solely on the bk model. In fact, the LSTM-RNN provides a surrogate prediction of the system response to circumvent the first step of collecting observations in the above traditional data assimilation procedure. The step of actually solving the data assimilation problem can then be run using a prediction of the true unavailable data observations. Hence, the final solution is obtained through a combination of a bk model and statistical learning.

3. Practical SVDA formulation

In this section, we illustrate how the SVDA concept can be put into practice, in particular by combining an LSTM-RNN with the PBDW data assimilation method. We first introduce the setting and some notation in Section 3.1. Then in Section 3.2, we review the main ideas of the PBDW approach for (traditional) variational data assimilation. The practical SVDA formulations are outlined for the offline stage in Section 3.3 and for the online stage in Section 3.4.

3.1. Setting and notation

We consider a spatial domain (open, bounded, connected subset) $\Omega \subset \mathbb{R}^d$, $d \geq 1$, with a Lipschitz boundary. We introduce a Hilbert space $\mathcal{U}$ composed of functions defined over Ω. The space $\mathcal{U}$ is endowed with an inner product $(\cdot, \cdot)$ and we denote by $\|\cdot\|$ the induced norm; $\mathcal{U}$ consists of functions $\{w : \Omega \to \mathbb{R} \mid \|w\| < \infty\}$. To fix the ideas, we assume that $H_0^1(\Omega) \subset \mathcal{U} \subset H^1(\Omega)$, and we denote the dual space of $\mathcal{U}$ by $\mathcal{U}'$. The Riesz operator $R_{\mathcal{U}} : \mathcal{U}' \to \mathcal{U}$ satisfies, for each $\ell \in \mathcal{U}'$, and for all $v \in \mathcal{U}$, the equality $\left(R_{\mathcal{U}}(\ell), v\right) = \ell(v)$. For any closed subspace $\mathcal{Q} \subset \mathcal{U}$, the orthogonal complement of $\mathcal{Q}$ is defined as $\mathcal{Q}^{\perp} := \{w \in \mathcal{U} \mid (w, v) = 0,\ \forall v \in \mathcal{Q}\}$. Finally, we introduce a parameter set $\mathcal{P} \subset \mathbb{R}^p$, $p \geq 1$, whose elements are generically denoted by $\mu \in \mathcal{P}$.

We recall that we use a 'best-knowledge' (bk) mathematical model in the form of a parametrized PDE posed over the domain Ω (or more generally, over a domain Ω^{bk} such that $\Omega \subset \Omega^{\text{bk}}$). Then, we introduce the manifold associated with the solutions of the bk model $\mathcal{M}^{\text{bk}} \subset \mathcal{U}$. In ideal situations, the true solution u^{true} is well approximated by the bk manifold, i.e., the model error

$$\epsilon_{\text{mod}}^{\text{bk}}(u^{\text{true}}) := \inf_{z \in \mathcal{M}^{\text{bk}}} \|u^{\text{true}} - z\|, \tag{1}$$

is very small.

We introduce nested background subspaces $\mathcal{Z}_1 \subset \ldots \subset \mathcal{Z}_N \subset \ldots \subset \mathcal{U}$ that are generated to approximate the bk manifold $\mathcal{M}^{\text{bk}}$ to a certain accuracy. These subspaces can be built using various model-order reduction techniques, for instance, the Reduced Basis Method [16–18]. Note that the indices of the subspaces conventionally indicate their dimensions. To measure how well the true solution is approximated by the background space $\mathcal{Z}_N$, we define the quantity $\epsilon_N^{\text{bk}}(u^{\text{true}}) := \inf_{z \in \mathcal{Z}_N} \|u^{\text{true}} - z\|$. The background space is built so that $\epsilon_N^{\text{bk}}(u^{\text{true}}) \underset{N \to +\infty}{\to} \epsilon_{\text{mod}}^{\text{bk}}(u^{\text{true}})$. Moreover, we introduce the reduction error $\epsilon_{\text{red},N}^{\text{bk}} := \sup_{u \in \mathcal{M}^{\text{bk}}} \inf_{z \in \mathcal{Z}_N} \|u - z\|$, which encodes the loss of accuracy caused by solving the bk model in the N-dimensional background space $\mathcal{Z}_N$. For later purposes, we introduce $\Pi_{\mathcal{Z}_N}(u^{\text{true}})$ as the closest point to u^{true} in $\mathcal{Z}_N$. Note that $\Pi_{\mathcal{Z}_N}$ is the $\mathcal{U}$-orthogonal projection onto $\mathcal{Z}_N$. The background space $\mathcal{Z}_N$ can be interpreted as a prior space that approximates the bk manifold which we hope approximates well the true state u^{true}. As previously alluded to, u^{true} rarely lies in $\mathcal{M}^{\text{bk}}$ in realistic engineering study cases. In the remainder of this Section, we give a brief review of the PBDW method and present the SVDA setting in the PBDW context.

3.2. PBDW formulation in the time-dependent context

Since the SVDA builds upon the PBDW, we recap the main ideas of the PBDW formulation for time-dependent problems, following [6,7].

Even when one can collect observations, full-knowledge of $u^{\rm true}$ is unrealistic. In many engineering cases, only a limited number of experimental observations of the true state $u^{\rm true}$ is affordable — interpreted as the application of prescribed observation functionals $\ell_m^{\rm obs} \in \mathcal{U}'$ for all $m \in \{1, \dots, M\}$. One can consider any observation functional that renders the behavior of some physical sensor.

In the time-dependent setting, we let these observation functionals act on time-averaged snapshots of the true solution. We also make the regularity assumption $u^{\rm true} \in L^1(I; \mathcal{U})$ on the true state. We introduce the time-integration intervals

$$\mathcal{I}^k = [t^k - \delta t^k, t^k + \delta t^k], \quad \forall k \in \mathbb{K}^{\rm tr}, \tag{2}$$

where $\delta t^k > 0$ is a parameter related to the time-precision of sensors. Then, for any function $v \in L^1(I; \mathcal{U})$, we define the time-averaged snapshots

$$v^k(x) := \frac{1}{|\mathcal{I}^k|} \int_{\mathcal{I}^k} v(t, x)\, dt \in \mathcal{U}, \quad \forall k \in \mathbb{K}^{\rm tr}. \tag{3}$$

Now we consider

$$\ell_m^{k,\rm obs}(u^{\rm true}) := \ell_m^{\rm obs}(u^{k,\rm true}), \quad \forall m \in \{1, \dots, M\},\ \forall k \in \mathbb{K}^{\rm tr}. \tag{4}$$

For instance, if the sensors act through local uniform time integration, we have

$$\ell_m^{k,\rm obs}(u^{\rm true}) = \frac{1}{|\mathcal{R}_m|} \int_{\mathcal{R}_m} u^{k,\rm true}(x)\, dx = \frac{1}{|\mathcal{R}_m|} \frac{1}{|\mathcal{I}_k|} \int_{\mathcal{R}_m} \int_{\mathcal{I}_k} u^{\rm true}(t, x)\, dx dt, \tag{5}$$

where $\mathcal{R}_m$ denotes the spatial domain of influence for the mth measurement.

The most convenient configuration to collect observations in industrial contexts is to measure the quantities at user-defined space–time locations. In actual practice, sensors do not take pointwise measures but localized ones. A sensor collects the data that is enclosed in a small area centered at the sensor location. Hence, Eq. (5) means that the sensor returns a measurement that is equal to the space–time averaged quantity we are collecting. Moreover, the observation functionals $\ell_m^{k,\rm obs}$ are Riesz representations of any type of physical data available for the user, and not necessarily pointwise or pointwise-like (integrations over small patches) observations.

Generally, we introduce the time-independent observable space $\mathcal{U}_M \subset \mathcal{U}$ such that

$$\mathcal{U}_M = {\rm Span}\{q_1, \dots, q_M\}, \tag{6}$$

where $q_m := R_{\mathcal{U}}(\ell_m^{\rm obs})$ is the Riesz representation of $\ell_m^{\rm obs} \in \mathcal{U}'$, for all $m \in \{1, \dots, M\}$, i.e.,

$$\ell_m^{\rm obs}(u^{k,\rm true}) = (u^{k,\rm true}, q_m), \quad \forall m \in \{1, \dots, M\}, \quad \forall k \in \mathbb{K}^{\rm tr}. \tag{7}$$

Note that, for fixed sensor locations, the computational effort to compute the Riesz representations of the observation functionals is time-independent and is incurred only once so that, for all $m \in \{1, \dots, M\}$ and $k \in \mathbb{K}^{\rm tr}$, the experimental observations of the true state satisfy:

$$\ell_m^{k,\rm obs}(u^{\rm true}) = \left(u^{k,\rm true}, q_m\right) = \frac{1}{|\mathcal{I}_k|} \int_{\mathcal{I}^k} \ell_m^{\rm obs}(u^{\rm true}(t, \cdot))dt. \tag{8}$$

Hence, for all $q \in \mathcal{U}_M$ such that,

$$q = \sum_{m=1}^{M} \alpha_m q_m, \tag{9}$$

the inner product $\left(u^{k,\rm true}, q\right)$ is deduced from the experimental observations as follows:

$$\begin{aligned} \left(u^{k,\rm true}, q\right) &= \frac{1}{|\mathcal{I}_k|} \int_{\mathcal{I}^k} \sum_{m=1}^{M} \alpha_m \left(u^{\rm true}(t, \cdot), q_m\right) dt \\ &= \frac{1}{|\mathcal{I}_k|} \sum_{m=1}^{M} \alpha_m \int_{\mathcal{I}^k} \ell_m^{\rm obs}(u^{\rm true}(t, \cdot))dt. \end{aligned} \tag{10}$$

Henceforth, we make the crucial assumption that

$$\mathcal{Z}_N \cap \mathcal{U}_M^{\perp} = \{0\}, \tag{11}$$

which is also equivalent to

$$\beta_{N,M} := \inf_{w \in \mathcal{Z}_N} \sup_{v \in \mathcal{U}_M} \frac{(w, v)}{\|w\|\ \|v\|} \in (0, 1], \tag{12}$$

where $\beta_{N,M}$ is the so-called stability constant (the reader can refer to [19] for a proof). Assumption (11) can be viewed as a requirement to have enough sensors (note that $\mathcal{Z}_N \cap \mathcal{U}^{\perp} = \{0\}$). Under this assumption, the limited-observations PBDW statement reads: for each $k \in \mathbb{K}^{\text{tr}}$, find $(u^{k,*}_{N,M}, z^{k,*}_{N,M}, \eta^{k,*}_{N,M}) \in \mathcal{U} \times \mathcal{Z}_N \times \mathcal{U}_M$ such that

$$(u^{k,*}_{N,M}, z^{k,*}_{N,M}, \eta^{k,*}_{N,M}) = \underset{\substack{u_{N,M}\in\mathcal{U}\\ z_{N,M}\in\mathcal{Z}_N\\ \eta_{N,M}\in\mathcal{U}_M}}{\operatorname{arginf}} \|\eta_{N,M}\|^2, \tag{13a}$$

$$\text{subject to } (u_{N,M}, v) = (\eta_{N,M}, v) + (z_{N,M}, v), \qquad \forall v \in \mathcal{U}, \tag{13b}$$

$$(u_{N,M}, \phi) = (u^{k,\text{true}}, \phi), \qquad \forall \phi \in \mathcal{U}_M. \tag{13c}$$

The limited-observations saddle-point problem associated with (13) reads: for each $k \in \mathbb{K}^{\text{tr}}$, find $(z^{k,*}_{N,M}, \eta^{k,*}_{N,M}) \in \mathcal{Z}_N \times \mathcal{U}_M$ such that

$$(\eta^{k,*}_{N,M}, q) + (z^{k,*}_{N,M}, q) = (u^{k,\text{true}}, q), \quad \forall q \in \mathcal{U}_M, \tag{14a}$$

$$(\eta^{k,*}_{N,M}, p) = 0, \qquad \forall p \in \mathcal{Z}_N, \tag{14b}$$

Hence, the limited-observations state estimate is

$$u^{k,*}_{N,M} = z^{k,*}_{N,M} + \eta^{k,*}_{N,M}, \quad \forall k \in \mathbb{K}^{\text{tr}}. \tag{15}$$

In algebraic form, the limited-observations PBDW statement reads: for each $k \in \mathbb{K}^{\text{tr}}$, find $(\boldsymbol{z}^{k,*}_{N,M}, \boldsymbol{\eta}^{k,*}_{N,M}) \in \mathbb{R}^N \times \mathbb{R}^M$ such that

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\eta}^{k,*}_{N,M} \\ \boldsymbol{z}^{k,*}_{N,M} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\ell}^{k,\text{obs}}_M \\ \mathbf{0} \end{pmatrix}, \tag{16}$$

with the matrices

$$\mathbf{A} = \Big((q_{m'}, q_m) \Big)_{1\le m,m'\le M} \in \mathbb{R}^{M\times M}, \ \mathbf{B} = \Big((\zeta_n, q_m) \Big)_{1\le m\le M, 1\le n\le N} \in \mathbb{R}^{M\times N}, \tag{17}$$

where $\mathcal{Z}_N = \text{span}\{\zeta_1, \dots, \zeta_N\}$, and the vector of observations

$$\boldsymbol{\ell}^{k,\text{obs}}_M = \left(\ell^{\text{obs}}_m(u^{k,\text{true}}) \right)_{1\le m\le M} \in \mathbb{R}^M. \tag{18}$$

Note that the matrices **A** and **B** are time-independent; only the right-hand side in (16) depends on k.

Remark 1 (*3D-VAR*)**.** We highlight that the PBDW is a special case of 3D-VAR [20] variational data assimilation. We introduce the bilinear forms $a : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ and $b : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, and the linear form $f : \mathcal{U} \to \mathbb{R}$. In a noise-free context, the 3D-VAR method consists in solving

$$(z^{k,*}, \eta^{k,*}) = \underset{\substack{z\in\mathcal{U}\\ \eta\in\mathcal{U}_M}}{\operatorname{arginf}} \frac{1}{2}\|\eta\|^2 + \frac{\lambda}{2}\|\Pi_{\mathcal{U}_M} u^{k,\text{true}} - \Pi_{\mathcal{U}_M}(z+\eta)\|^2, \tag{19a}$$

$$\text{subject to} \quad a(z, v) = f(v) + b(\eta, v), \quad \forall v \in \mathcal{U}, \tag{19b}$$

which is equivalent to solving

$$(z^{k,*}, \eta^{k,*}) = \underset{\substack{u\in\mathcal{U}\\ z\in\mathcal{Z}\\ \eta\in\mathcal{U}}}{\operatorname{arginf}} \|\eta\|^2, \tag{20a}$$

$$\text{subject to} \quad a(z, v) = f(v) + b(\eta, v), \quad \forall v \in \mathcal{U}, \tag{20b}$$

$$(z + \eta, \phi) = (u^{k,\text{true}}, \phi), \qquad \forall v \in \mathcal{U}_M. \tag{20c}$$

If $b \equiv 0$, the constraint (20b) can be replaced by a requirement that z belongs to the manifold of solutions $\mathcal{Z} = \{u \in \mathcal{U} \mid a(u, v) = f(v)\}$. In this case, the 3D-VAR problem reads:

$$(u^*, z^*, \eta^*) = \underset{\substack{u\in\mathcal{U}\\ z\in\mathcal{Z}\\ \eta\in\mathcal{U}}}{\operatorname{arginf}} \|\eta\|^2, \tag{21a}$$

$$\text{subject to} \quad (u, v) = (z, v) + (\eta, v), \qquad \forall v \in \mathcal{U}, \tag{21b}$$

$$(u, \phi) = (u^{\text{true}}, \phi), \qquad \forall v \in \mathcal{U}_M, \tag{21c}$$

which is the same idea as the PBDW formulation.

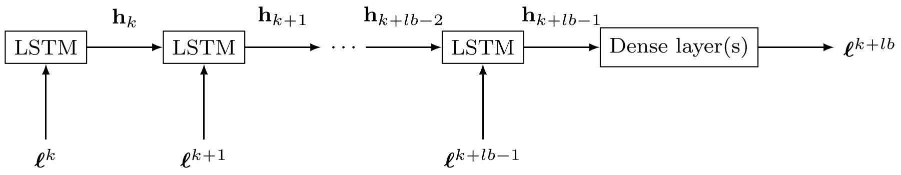

Fig. 1. Sketch of the LSTM-RNN setting in the numerical experiments. For better visibility, only the propagation of the recurrent state is depicted.

3.3. SVDA offline phase

As described in Section 2, the SVDA consists of an offline and an online phase. During the offline stage (cf. Algorithm 1 below), one precomputes the functions $(\zeta_n)_{1\le n\le N}$ as basis of $\mathcal{Z}_N$ and the Riesz representations $(q_m)_{1\le m\le M}$, cf. (7). Recall that the basis $(\zeta_n)_{1\le n\le N}$ is computed in our case using the Reduced Basis Method [16–18] based on solution snapshots of the bk model. With ζ_n and q_m at hand, one precomputes and stores the matrices $\mathbf{A}\in\mathbb{R}^{M\times M}$ and $\mathbf{B}\in\mathbb{R}^{M\times N}$ defined in (17) once and for all.

One also runs the deep learning module on the training data in order to produce an LSTM-RNN prediction function. Concretely, we now describe the set-up of our LSTM-RNN model for the SVDA. Recall that the specific choice of the machine learning model can be exchanged in the SVDA formulation and should be adapted to the problem context. We therefore focus our description on the concepts used in the numerical experiments below.

For completeness, we briefly describe how a single LSTM cell works. As LSTMs are recurrent networks, there is a recurrent state denoted by $\mathbf{h}$. Additionally, an LSTM unit also includes a cell state $\mathbf{c}$. At unit j, $\mathbf{c}_j$ and $\mathbf{h}_j$ are computed from the previous steps $\mathbf{c}_{j-1}$ and $\mathbf{h}_{j-1}$ as well as from the input value $\mathbf{x}_j$, which reflects the input variable at time instance t_j. $\mathbf{h}_{j-1}$ and $\mathbf{x}_j$ are concatenated to $\hat{\mathbf{x}}_j$, from which the values between 0 and 1 of the *forget, update and output gates* $\mathbf{f}_j,\mathbf{u}_j,\mathbf{o}_j$ are computed via

$$\mathbf{f}_j=\sigma(\mathbf{W}_f\hat{\mathbf{x}}_j+\mathbf{b}_f),\quad \mathbf{u}_j=\sigma(\mathbf{W}_u\hat{\mathbf{x}}_j+\mathbf{b}_u),\quad \mathbf{o}_j=\sigma(\mathbf{W}_o\hat{\mathbf{x}}_j+\mathbf{b}_o).$$

Here, $\mathbf{W}_f,\mathbf{W}_u,\mathbf{W}_o$ and $\mathbf{b}_f,\mathbf{b}_u,\mathbf{b}_o$ are the trainable weights and biases, respectively, and σ denotes the activation function. The gate mechanism is the cornerstone of every LSTM unit and determines how the recurrent and cell state are updated. First a new candidate cell state $\tilde{\mathbf{c}}_j$ is computed as

$$\tilde{\mathbf{c}}_j=\tanh(\mathbf{W}_c\hat{\mathbf{x}}_j+\mathbf{b}_c),$$

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are yet another trainable weight and bias, respectively. Then, the updated quantities are calculated via

$$\mathbf{c}_j=\mathbf{f}_j\odot\mathbf{c}_{j-1}+\mathbf{u}_j\odot\tilde{\mathbf{c}}_j,\qquad \mathbf{h}_j=\mathbf{o}_j\odot\tanh\mathbf{c}_j,$$

where $\odot$ denotes the component-wise product. While $\mathbf{c}$ is an internal variable, the recurrent state $\mathbf{h}$ also serves as output, especially at the final LSTM unit. We refer the reader to [21] as well as the original articles [22,23] for more details on LSTMs.

For our LSTM-RNN model, we use lb LSTM units with the so-called lookback variable $lb\le k_{\text{off}}-1$. In other words, this variable determines how many previous time steps are used for the prediction of the observables at the current time. Consequently, the input variable has the shape $\mathbb{R}^{lb\times M}$. Besides the LSTM units, our neural network consists of dense layer(s), see Fig. 1 for a sketch. Details on the used architecture and the training are provided in Section 5. The described neural network construction is used to emulate the map from $(\ell^k,\dots\ell^{k+lb-1})$ to ℓ^{k+lb} for each training time step t^k, where we recall that ℓ^k denotes the vector of observations, cf. (18). For fixed k^{off}, lb also influences the number of available input/output training data pairs. To better illustrate this, let us consider two extreme cases. On the one hand, for $lb=1$, we have $k_{\text{off}}-1$ training data pairs. On the other hand, for $lb=k_{\text{off}}-1$, we have only a single training data pair.

Algorithm 1 Offline stage of the SVDA

```
INPUT : $\{q_1,\dots,q_M\}$: Riesz representations of the observations; bk model and set of training parameters; lookback variable $lb$ and hyperparameters for LSTM-RNN
1: Compute $\mathcal{Z}_N=\mathrm{span}\{\zeta_1,\dots,,\zeta_N\}$ using the Reduced Basis Method based on bk model and training parameters.
2: Set $\mathcal{U}_M := \mathrm{Span}\{q_1,\dots,q_M\}$.
3: Compute the matrices $\mathbf{A}$ and $\mathbf{B}$ according to (17) using $\mathcal{Z}_N$ and $\mathcal{U}_M$.
4: Compute the LSTM-RNN prediction function $\ell_M^{\mathrm{DL}}$ using $lb$ LSTM units and additional dense layers, cf. Fig. 1, training based on Adams optimizer, cf. Section 5.
OUTPUT : $\mathcal{Z}_N$, $\mathcal{U}_M$, $\mathbf{A}$ and $\mathbf{B}$, $\ell_M^{\mathrm{DL}}$.
```

3.4. SVDA online phase

During the *online stage* (cf. Algorithm 2), we would like to solve the PBDW problem (16) at each time step t_k in $[\Delta t, T]$. However, recall that in the SVDA context, the observations are not available at the time steps of interest. Hence, one cannot evaluate the observations $\ell_m^{k,\mathrm{obs}}(u^{\mathrm{true}})$ in (4), which will be replaced by their LSTM-RNN prediction. Therefore, we apply the trained LSTM-RNN prediction function to the input data in order to generate a vector of predicted observations $\boldsymbol{\ell}_M^{k,\mathrm{DL}}$ to replace $\boldsymbol{\ell}_M^{k,\mathrm{obs}}$ in (16). Precisely, we get the predictions $\boldsymbol{\ell}^{k,DL}$ for any time step t_k very cheaply by providing observations at lb previous time steps. We, hence, have to sequentially run the neural network several times to get all surrogate observations in $[\Delta t, T]$. Since the evaluation of a trained neural network is rather cheap, this sequential run comes at extremely low cost. The computation of $\boldsymbol{\ell}^{k,DL}$ for times t^k in $[\Delta t, T]$ can also be viewed as a machine learning based time stepping.

Let us now present the SVDA formulation of the PBDW problem. We directly start from the algebraic formulation (16) and replace $\boldsymbol{\ell}_M^{k,\mathrm{obs}}$ by $\boldsymbol{\ell}_M^{k,\mathrm{DL}}$. The SVDA-PBDW statement reads: for each $k \in \mathbb{K}^{\mathrm{tr}}$, find $(\boldsymbol{z}_{N,M}^{k,\mathrm{svda}}, \boldsymbol{\eta}_{N,M}^{k,\mathrm{svda}}) \in \mathbb{R}^N \times \mathbb{R}^M$ such that

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\eta}_{N,M}^{k,\mathrm{svda}} \\ \boldsymbol{z}_{N,M}^{k,\mathrm{svda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\ell}_M^{k,\mathrm{DL}} \\ \mathbf{0} \end{pmatrix}, \tag{22}$$

with the matrices $\mathbf{A}, \mathbf{B}$ as in the original PBDW formulation above (cf. (17)). Having solved this linear $(N+M)$-dimensional problem, for each $k \in \{k_{\mathrm{off}}, \dots, K\}$, the SVDA state estimate $\boldsymbol{u}_{N,M}^{k,\mathrm{svda}}$ is deduced as follows:

$$\boldsymbol{u}_{N,M}^{k,\mathrm{svda}} = \mathbf{Z}_N \boldsymbol{z}_{N,M}^{k,\mathrm{svda}} + \mathbf{U}_M \boldsymbol{\eta}_{N,M}^{k,\mathrm{svda}}, \tag{23}$$

where $\mathbf{Z}_N$ and $\mathbf{U}_M$ are the algebraic counterparts of $\mathcal{Z}_N$ and $\mathcal{U}_M$ respectively. Here, we emphasize that predicting $\boldsymbol{\ell}_M^{k,\mathrm{DL}}$ with the LSTM-RNN instead of the 'state' $\boldsymbol{z}_{N,M}, \boldsymbol{\eta}_{N,M}$ allows us to use the traditional data assimilation framework of PBDW, as we already explained in Section 2.1. This ensures 'consistency' in the solver between time steps in the sense that model (16) for time steps with real data available and model (22) for time steps without available data are of similar nature. Additionally, we can in this way easily guarantee that indeed $\eta_{N,M}^{k,\mathrm{svda}} \in \mathcal{Z}_N^{\perp}$. Finally, note that predicting ℓ instead of $[z, \eta]$ implies less computational costs as the observation vector has only M components instead of $N+M$.

Algorithm 2 Online stage of the SVDA

```
INPUT : $k_{\mathrm{off}}$, output of the offline stage: $\mathcal{Z}_N, \mathcal{U}_M$, matrices $\boldsymbol{A}, \boldsymbol{B}$, trained LSTM-RNN prediction function $\boldsymbol{\ell}_M^{\mathrm{DL}}$.
1: for $k \in \{k_{\mathrm{off}}, \dots, K\}$ do
2:     Compute $\boldsymbol{\ell}_M^{k,\mathrm{DL}}$ by applying the LSTM-RNN prediction function $\boldsymbol{\ell}_M^{\mathrm{DL}}$ at time $t^k$ to input data from previous $lb$ time steps.
3:     Solve the SVDA online system (22) at time $t^k$.
4:     Update the SVDA state estimate $\boldsymbol{u}_{N,M}^{k,\mathrm{svda}}$ according to (23).
5: end for
OUTPUT : SVDA future trajectory $\{\boldsymbol{u}_{N,M}^{k,\mathrm{svda}}\}_{k_{\mathrm{off}} \le k \le K}$.
```

Note that we can reformulate (22) into variational form similar to the PBDW, which is useful for the error analysis, but not used in the practical implementation. For this, we use (7) to define LSTM-RNN predictions of the state out of $\boldsymbol{\ell}_M^{k,\mathrm{DL}}$. Precisely, we define the field of LSTM-RNN state predictions $u^{k,\mathrm{DL}} \in \mathcal{U}_M$ via

$$u^{k,\mathrm{DL}} = \sum_{m=1}^{M} u_m^{k,\mathrm{DL}} q_m \tag{24}$$

and

$$\boldsymbol{\ell}_M^{k,\mathrm{DL}} = \left(\ell_m^{\mathrm{obs}}(u^{k,\mathrm{DL}})\right)_{1 \le m \le M} = \left((u^{k,\mathrm{DL}}, q_m)\right)_{1 \le m \le M} \tag{25}$$

The variational form of (22) of the SVDA problem now reads: for each $k \in \mathbb{K}^{\mathrm{tr}}$, find $(z_{N,M}^{k,\mathrm{svda}}, \eta_{N,M}^{k,\mathrm{svda}}) \in \mathcal{Z}_N \times \mathcal{U}_M$ such that

$$(\eta_{N,M}^{k,\mathrm{svda}}, q) + (z_{N,M}^{k,\mathrm{svda}}, q) = (u^{k,\mathrm{DL}}, q), \quad \forall q \in \mathcal{U}_M, \tag{26a}$$

$$(\eta_{N,M}^{k,\mathrm{svda}}, p) = 0, \quad \forall p \in \mathcal{Z}_N, \tag{26b}$$

Hence, the SVDA state estimate is

$$u_{N,M}^{k,\mathrm{svda}} = z_{N,M}^{k,\mathrm{svda}} + \eta_{N,M}^{k,\mathrm{svda}}, \quad \forall k \in \mathbb{K}^{\mathrm{tr}}. \tag{27}$$

Note that $u_{N,M}^{k,\mathrm{svda}}$ is, hence, the function associated with the vector $\boldsymbol{u}_{N,M}^{k,\mathrm{svda}}$.

4. Error estimation

In this section, we establish an *a priori* error analysis of the SVDA approximation. We first recall an important result proved in [3,24].

Proposition 1. *At each time step $k \in \mathbb{K}^{\mathrm{tr}}$, the PBDW error estimation satisfies*

$$\|u^{k,\mathrm{true}} - u^{k,*}_{N,M}\| \leq \frac{1}{\beta_{N,M}} \inf_{q \in \mathcal{U}_M \cap \mathcal{Z}_N^{\perp}} \|\Pi_{\mathcal{Z}_N} u^{k,\mathrm{true}} - q\|, \tag{28}$$

where $\beta_{N,M}$ is the stability constant defined as

$$\beta_{N,M} := \inf_{z \in \mathcal{Z}_N} \sup_{q \in \mathcal{U}_M} \frac{(z,q)}{\|z\|\|q\|} \in (0,1].$$

Proof. See [3], Proposition 2, and for the improved constant of $\frac{1}{\beta_{N,M}}$ see [24].

Let us now estimate the SVDA approximation error.

Proposition 2. *In the present context, the following upper bounds on the SVDA approximation hold true*

$$\|u^{k,*}_{N,M} - u^{k,\mathrm{svda}}_{N,M}\| \leq \left(1 + \frac{2}{\beta_{N,M}}\right) \|\Pi_{\mathcal{U}_M} u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}\|. \tag{29}$$

$$\begin{aligned} \|u^{k,\mathrm{true}} - u^{k,\mathrm{svda}}_{N,M}\| \leq{}& \frac{1}{\beta_{N,M}} \inf_{q \in \mathcal{U}_M \cap \mathcal{Z}_N^{\perp}} \|\Pi_{\mathcal{Z}_N} u^{k,\mathrm{true}} - q\| \\ &+ \left(1 + \frac{2}{\beta_{N,M}}\right) \|\Pi_{\mathcal{U}_M} u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}\|. \end{aligned} \tag{30}$$

Proof.

1. By subtracting (26a) from (14a), we obtain

$$(\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}, q) + (z^{k,*}_{N,M} - z^{k,\mathrm{svda}}_{N,M}, q) = (u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}, q), \qquad \forall q \in \mathcal{U}_M.$$

Let us choose the test function $q = \eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M} \in \mathcal{U}_M$ as a test function. Using the fact that $\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M} \in \mathcal{Z}_N^{\perp}$ along with the Cauchy–Schwarz inequality, we get

$$\begin{aligned} \|\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}\|^2 &= \left(\Pi_{\mathcal{U}_M}(u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}), \eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}\right) \\ &\leq \|\Pi_{\mathcal{U}_M}(u^{k,\mathrm{true}} - u^{k,\mathrm{DL}})\| \; \|\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}\|. \end{aligned}$$

Hence,

$$\|\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}\| \leq \|\Pi_{\mathcal{U}_M}(u^{k,\mathrm{true}} - u^{k,\mathrm{DL}})\|. \tag{31}$$

Moreover, $(z^{k,*}_{N,M} - z^{k,\mathrm{svda}}_{N,M}) \in \mathcal{Z}_N$ leads to

$$\begin{aligned} \beta_{N,M}\|z^{k,*}_{N,M} - z^{k,\mathrm{svda}}_{N,M}\| &\leq \sup_{q \in \mathcal{U}_M} \frac{(z^{k,*}_{N,M} - z^{k,\mathrm{svda}}_{N,M}, q)}{\|q\|} \\ &= \sup_{q \in \mathcal{U}_M} \frac{(u^{k,\mathrm{true}} - u^{k,\mathrm{DL}} - (\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}), q)}{\|q\|} \\ &\leq \|\Pi_{\mathcal{U}_M}(u^{k,\mathrm{true}} - u^{k,\mathrm{DL}})\| + \|\eta^{k,*}_{N,M} - \eta^{k,\mathrm{svda}}_{N,M}\| \\ &\leq 2\|\Pi_{\mathcal{U}_M} u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}\|, \end{aligned}$$

where the last inequality follows from (31) and the fact that $u^{k,\mathrm{DL}} \in \mathcal{U}_M$. Hence,

$$\|z^{k,*}_{N,M} - z^{k,\mathrm{svda}}_{N,M}\| \leq \frac{2}{\beta_{N,M}} \|\Pi_{\mathcal{U}_M} u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}\|. \tag{32}$$

Combining (31) and (32), we obtain

$$\|u^{k,*}_{N,M} - u^{k,\mathrm{svda}}_{N,M}\| \leq \left(1 + \frac{2}{\beta_{N,M}}\right) \|\Pi_{\mathcal{U}_M} u^{k,\mathrm{true}} - u^{k,\mathrm{DL}}\|. \tag{33}$$

The spirit of this proof is similar to that of noisy observations [4].

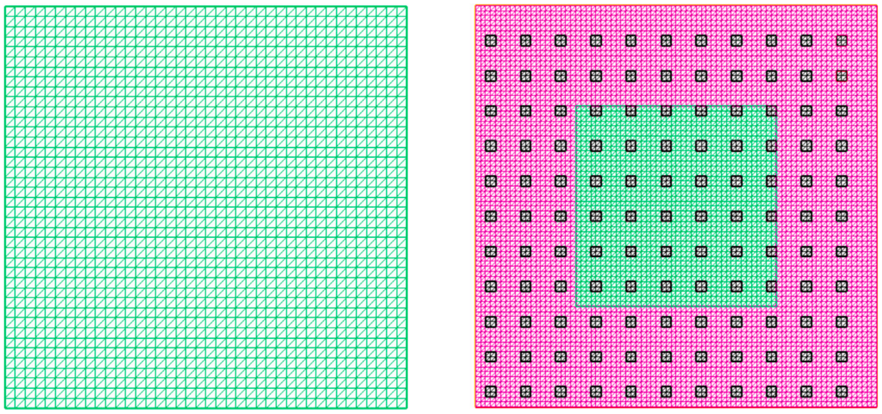

Fig. 2. Computational domain and mesh with $\mathcal{N} = 6561$. The little black squares are observation subsets $\{\mathcal{R}_m\}_{m=1}^{121}$. Left: Mono-material plate. Right: Bi-material plate.

2. Using the triangle inequality, we have

$$\|u^{k,\text{true}} - u^{k,\text{svda}}_{N,M}\| \leq \|u^{k,\text{true}} - u^{k,*}_{N,M}\| + \|u^{k,*}_{N,M} - u^{k,\text{svda}}_{N,M}\|. \tag{34}$$

   It follows from (34), (28), and (33) that

$$\|u^{k,\text{true}} - u^{k,\text{svda}}_{N,M}\| \leq \frac{1}{\beta_{N,M}} \inf_{q \in \mathcal{U}_M \cap \mathcal{Z}_N^{\perp}} \|\Pi_{\mathcal{Z}_N} u^{k,\text{true}} - q\| \tag{35}$$

$$+ \left(1 + \frac{2}{\beta_{N,M}}\right) \|\Pi_{\mathcal{U}_M} u^{k,\text{true}} - u^{k,\text{DL}}\|. \tag{36}$$

The result in (30) shows that the quality of the SVDA approximation has two contributions. The first contribution depends on the quality of the PBDW spaces. The better the quality of the background space $\mathcal{Z}_N$ and the observable space $\mathcal{U}_M$, the smaller the error $\|u^{k,\text{true}} - u^{k,\text{svda}}_{N,M}\|$. The second contribution is related to the quality of the statistical prediction. The more accurate $u^{k,\text{DL}}$, the smaller the error $\|u^{k,\text{true}} - u^{k,\text{svda}}_{N,M}\|$. In concrete applications, both error contributions may be estimated further or assessed with an indicator to yield an overall bound for the SVDA quality. While there is a rich literature on error estimation for variational data assimilation, see e.g. [3] in the PBDW context, the performance of machine learning methods is typically assessed by heuristics and experimental studies for use cases. For instance, the class of LSTM-RNN networks is widely used in the literature and has shown excellent performance for time series prediction tasks, see [21–23] and references therein. While the good general approximation properties of neural networks are well established, see e.g. [25] for parabolic problems and [26] for parametrized PDEs, error estimates for concrete trained neural networks are much more rare. Two interesting approaches that might allow to estimate $\|\Pi_{\mathcal{U}_M} u^{k,\text{true}} - u^{k,\text{DL}}\|$ for appropriate machine learning methods within SVDA in the future are the following: [27] proposes an error analysis framework for physics informed neural networks. [28] combines machine learning with certified reduced basis approaches for parametric PDEs.

5. Numerical results

In this section, we implement the above developments. The goal is to illustrate the computational performance of the SVDA method as a proof of concept. We resort to the following numerical strategy:

1. Synthesize two different models out of the same PDE. Towards that end, an efficient strategy is to change a (physical) parameter. The first model will be considered as the 'true' model and the second will be the 'best-knowledge' model.
2. Using the 'true' model, create training data for the initial time interval $[0, \Delta t]$.
3. Train the statistical model using LSTM-RNN.
4. Run the online SVDA.

We consider a two-dimensional setting based on the plate illustrated in the left panel of Fig. 2 with $\Omega = (-2, 2)^2 \subset \mathbb{R}^2$. We use a finite element (FE) [29] subspace $\mathcal{U}^{\mathcal{N}} \subset \mathcal{U} = H^1(\Omega)$, where $H^1(\Omega)$ is the linear space of square integrable and differentiable functions defined on Ω. The subspace $\mathcal{U}$ consists of continuous, piecewise affine functions in order to generate FE solutions. The FE subspace $\mathcal{U}^{\mathcal{N}}$ is based on a mesh that contains $\mathcal{N} = 6561$ nodes. The experimental data is generated synthetically and the observation subsets $\{\mathcal{R}_m\}_{1 \leq m \leq M}$ are uniformly selected over the plate as illustrated in the right panel of Fig. 2. Regarding the implementation, the FE computations use the software `FreeFem++` [30], the SVDA algorithm has been developed in `Python`. The deep learning subroutines of the SVDA use the `Python` library `Tensorflow.keras` [31].

5.1. Physical model problem

We apply the above methodology to the following parabolic PDE: For any value of the parameter $\mu \in \mathcal{P}$, find $u(\mu) : I \times \Omega \to \mathbb{R}$ such that

$$\begin{cases} \dfrac{\partial u(\mu)}{\partial t} - \nabla \cdot (D(\mu)\nabla u(\mu)) = 0, & \text{in } I \times \Omega, \\ u(\mu)(t=0,\cdot) = u_0, & \text{in } \Omega, \\ \text{Boundary conditions}, & \text{on } I \times \partial\Omega, \end{cases} \tag{37}$$

where $u_0 = 293.15$ K (20 °C). We will supplement (37) with Stefan–Boltzmann boundary conditions on $\partial\Omega$, i.e.,

$$-D(\mu)\frac{\partial u}{\partial n} = \sigma\varepsilon(u^4 - u_r^4), \qquad \text{on } I \times \partial\Omega, \tag{38}$$

with an enclosure temperature $u_r = 303.15$ K (30 °C), the Stefan–Boltzmann constant $\sigma = 5.67 \times 10^{-8}$ W m^{-2} K^{-4}, and an emissivity $\varepsilon = 3.10^{-3}$. The Stefan–Boltzmann boundary condition is nonlinear. Hence, the resulting problem (37)–(38) is nonlinear. For both subsequent test cases, the background spaces $\mathcal{Z}_N$ will be generated by solving the nonlinear PDE (37)–(38) with a uniform diffusivity function $D(\mu)$ such that for all $x \in \Omega$, $D(\mu)(x) = D_{\text{uni}}(\mu)(x) := \mu \mathbf{1}_\Omega(x)$ (mono-material plate, cf. left panel of Fig. 2).

5.2. Synthetic data generation

We generate the data by first synthesizing a true solution and then applying to it the linear functionals by means of their Riesz representations in the observable space $\mathcal{U}_M$. In order to synthesize the true solution, we consider a 'true model' based on the bi-material plate (cf. right panel of Fig. 2) where we choose a fixed internal diffusivity $D_{\text{int}} = 1$ and define, for each $\mu \in \mathcal{P}$, the diffusivity function $D(\mu)$ as $D(\mu)(x) = D_{\text{syn}}(\mu)(x) := \mu D_{\text{int}} \mathbf{1}_{\Omega_{\text{ext}}}(x) + D_{\text{int}} \mathbf{1}_{\Omega_{int}}(x)$, for all $x \in \Omega$, where $\Omega_{\text{int}} = (-1,1)^2$ and $\Omega_{\text{ext}} = (-2,2)^2 \setminus (-1,1)^2$, so that $\overline{\Omega} = \overline{\Omega}_{\text{int}} \cup \overline{\Omega}_{\text{ext}}$ and $\Omega_{\text{int}} \cap \Omega_{\text{ext}} = \emptyset$. The synthetic true solutions are then defined as the solutions of (37)–(38) for all $\mu \in \mathcal{P}$.

5.3. Test case (a) : Future forecast

For time discretization, we consider the time interval $I = [0, 2.5]$ s, a constant time step $\tau^k = 1.25 \times 10^{-2}$ s for all $k \in \mathbb{K}^{\text{tr}}$, and the set of discrete time nodes $\mathbb{K}^{\text{tr}} = \{1, \dots, 200\}$. The parameter of the true model is fixed to $\mu = 15$, both for training as well as testing. We first build the bk space. Using a Proper Orthogonal decomposition (POD), we obtain $N = 4$ basis functions. In this section, the bk model corresponds to the solution of (37)–(38) for the mono-material plate without incorporating any data. We train our LSTM-RNN (cf. Section 2) with 2 additional dense hidden layers and 1 dense output layer using the data on the first 50 time steps. Using a mean squared error loss function and the Adam optimizer with a learning rate 10^{-2}, we perform the SVDA prediction for the remaining future time steps. We refer the reader to [14] for more details on how to fit a neural network (number of hidden layers, optimizer choice, etc.). We recall that the non-deterministic nature of the LSTM-RNN propagates to the SVDA. Hence, the output results are tendencies of the solution behavior and not exactly reproducible solutions. For the values of the lookback parameter $lb \in \{1, 7\}$ (cf. Section 2), Fig. 3 displays the relative L^2 errors to the true solution.

We clearly see that the machine learning surrogates improve the accuracy of the future prediction in comparison to using only the bk model (in green). Comparable results – errors of about one percent – are also achieved if we increase the training time window to the first 100 time steps (data not shown). Note that the orange line, which depicts the (unrealistic) situation that true observations are available for data assimilation also at future time steps, is *not* a lower bound for the SVDA error. The reason is that the data assimilation problem is constrained, i.e., it is a problem of minimization under constraint. This means that the solution to the problem is non-unique and only a local optimum can be recovered. The local optimum recovered by PBDW and SVDA may be different and may or may not be the global optimum over the discrete space. In particular, the local optimum recovered by SVDA may be better than the local optimum recovered by PBDW. As expected, errors grow over time as we depart from the time window with true observations available at close previous steps, but the error growth is rather moderate overall. Fig. 4 shows the error of a pure LSTM-RNN prediction for ℓ at all 'future' time steps. We observe that the error shows the same qualitative trend as the SVDA error in Fig. 3, but the values for pure LSTM-RNN prediction are about a factor 3 to 4 larger at the final time. We, therefore, concentrate on the SVDA error in the following. Since all results are quite similar for the different lookback values, we will focus on $lb = 1$ in the following.

5.4. Test case (b) : Parametric forecast

For a parametric forecast, we train the LSTM-RNN using the true model with $\mu = 15$ over the whole time interval $[0, 2.5]s$. The time steps, the spatial discretization and the network architecture are the same as in Test case (a). In particular, the PBDW background space $\mathcal{Z}_n$ is based on the mono-material plate with $\mu = 15$, cf. Section 5.1. We use the trained LSTM-RNN to create surrogate data at all time steps for the value $\mu = 17$ and integrate those surrogate observations into the SVDA. Fig. 5 shows the relative L^2 errors to the true solution. As announced, we focus on the case $lb = 1$. As in the previous section, the bk model corresponds to the mono-material plate with $\mu = 15$. Although the parameter change from $\mu = 15$ for the bk model to $\mu = 17$ for the true model is very moderate, the green line for the bk error grows immediately and the average bk error is about $4.5 \cdot 10^{-2}$. Again, the SVDA is able to improve the

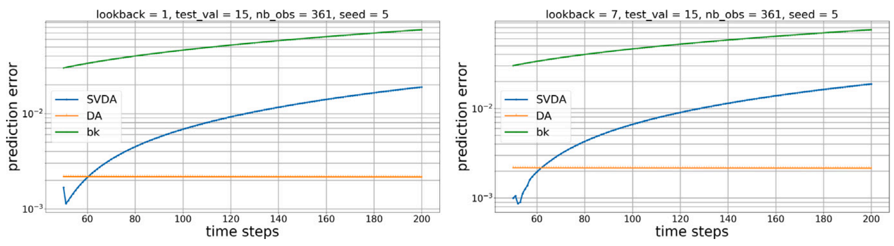

Fig. 3. Error estimation for $lb = 1$ (left) and $lb = 7$ (right).

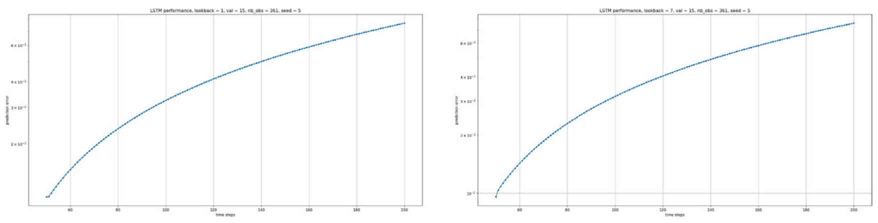

Fig. 4. Error estimation for a pure LSTM-RNN prediction of ℓ for the two lookback values.

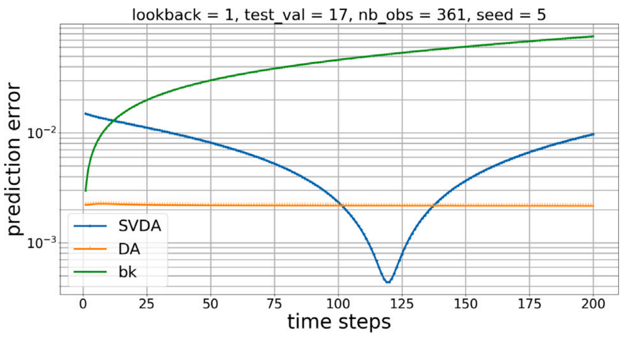

Fig. 5. Error estimation in the parametric case.

accuracy and its average error is about $6 \cdot 10^{-3}$. We emphasize that the LSTM-RNN predictions within SVDA are parameter-agnostic, i.e., LSTM-RNN predicts the solution at the next time step just from the input of the solution at the previous time step, but without any parameter input. Only the PBDW segment of the SVDA handles parameter variation. Therefore, considered test case with a rather limited training set for the LSTM-RNN is a challenge and underlines the promising properties of SVDA. Interestingly, the SVDA error even drops quite drastically somewhere in the middle of time simulation. We believe that the nonlinearity of the model as well as the statistical training allow for such an error behavior, which is rather uncommon for traditional numerical methods. Finally, we mention that comparable results for the SVDA were even obtained when testing for other parameters — even if they were rather different from the bk parameter $\mu = 15$.

6. Conclusion

We introduced a statistical variational data assimilation (SVDA) method providing a new concept that combines machine learning methods with traditional data assimilation. While we explained the details of the framework based on long-short term memory (LSTM) networks and the parametric background data weak (PBDW) approach for data assimilation, the key idea is very flexible and versatile and can be easily adapted to other neural networks and data assimilation schemes. The core idea is to train a neural network based on available observations and use the network's predictions as surrogate data in situations where no observations are available, but data assimilation is needed or wished. We rigorously proved that the overall SVDA error is bounded above by the quality of discrete (approximation) spaces used in data assimilation and the quality of the machine learning model. To illustrate the performance and applicability of the SVDA in practice, we considered a parametric heat equation with nonlinear boundary conditions. Both for future and parametric forecasts, the SVDA showed promising results. The flexibility of the framework entails that we cannot test the plethora of potential applications. Further investigations for more realistic scenarios (e.g. weather prediction) and on the influence of the various method parameters (lookback, neural network choice, etc.) are interesting future research directions. Further, the influence of noisy data should be considered, where one could apply a noise correction filter to the observations prior to their use as an input to the LSTM-RNN or one could use the PBDW variant in [32] and additionally control the noise propagation in this time-dependent case by a weighting factor varying with the time step of the PBDW correction.

CRediT authorship contribution statement

Amina Benaceur: Writing – original draft, Visualization, Validation, Software, Investigation, Formal analysis, Conceptualization. **Barbara Verfürth:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Funding acquisition.

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 work is funded by the Federal Ministry of Education and Research (BMBF) and the Baden-Württemberg Ministry of Science as part of the Excellence Strategy of the German Federal and State Governments. The authors acknowledges TEEMLEAP members Peter Knippertz, Sebastian Lerch, Uwe Ehret, Jörg Meyer, Julian Quinting, and Jannik Wilhelm for stimulating discussions. The help of the student assistant Klara Becker in running additional simulations is gratefully acknowledged. This work was conducted while both authors where affiliated with Karlsruhe Institute of Technology.

We thank the anonymous referees for their valuable comments, which helped to improve the paper.

References

[1] C.K. Wikle, L.M. Berliner, A Bayesian tutorial for data assimilation, Phys. D 230 (1–2) (2007) 1–16, http://dx.doi.org/10.1016/j.physd.2006.09.017.
[2] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME Ser. D. J. Basic Eng. 82 (1) (1960) 35–45.
[3] Y. Maday, A.T. Patera, J.D. Penn, M. Yano, A parameterized-background data-weak approach to variational data assimilation: formulation, analysis, and application to acoustics, Internat. J. Numer. Methods Engrg. 102 (5) (2015) 933–965, http://dx.doi.org/10.1002/nme.4747.
[4] Y. Maday, A.T. Patera, J.D. Penn, M. Yano, PBDW state estimation: noisy observations; configuration-adaptive background spaces; physical interpretations, in: CANUM 2014—42e Congrès National d'Analyse Numérique, in: ESAIM Proc. Surveys, vol. 50, EDP Sci., Les Ulis, 2015, pp. 144–168, http://dx.doi.org/10.1051/proc/201550008.
[5] J. Hammond, R. Chakir, F. Bourquin, Y. Maday, PBDW: A non-intrusive reduced basis data assimilation method and its application to an urban dispersion modeling framework, Appl. Math. Model. 76 (2019) 1–25, http://dx.doi.org/10.1016/j.apm.2019.05.012.
[6] A. Benaceur, Reducing sensors for transient heat transfer problems by means of variational data assimilation, SMAI J. Comput. Math. 7 (2021) 1–25.
[7] A. Benaceur, A time-dependent parametrized background data-weak approach, in: Numerical Mathematics and Advanced Applications—ENUMATH 2019, in: Lect. Notes Comput. Sci. Eng., vol. 139, Springer, 2021, pp. 125–133, http://dx.doi.org/10.1007/978-3-030-55874-1_11.
[8] A. Geer, Learning earth system models from observations: machine learning or data assimilation? Phil. Trans. R. Soc. A 379 (2194) (2021) 20200089.
[9] Y. Ruckstuhl, T. Janjić, S. Rasp, Training a convolutional neural network to conserve mass in data assimilation, Nonlinear Process. Geophys. 28 (1) (2021) 111–119.
[10] J. Brajard, A. Carrassi, M. Bocquet, L. Bertino, Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model, Geosci. Model Dev. Discuss. (2019) 1–21.
[11] A. Farchi, M. Bocquet, P. Laloyaux, M. Bonavita, Q. Malartic, A comparison of combined data assimilation and machine learning methods for offline and online model error correction, J. Comput. Sci. 55 (2021) 101468.
[12] S.O. Ba, T. Corpetti, B. Chapron, R. Fablet, Variational data assimilation for missing data interpolation in SST images, in: 2010 IEEE International Geoscience and Remote Sensing Symposium, IEEE, 2010, pp. 264–267.
[13] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, vol. 112, Springer, 2013.
[14] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2, Springer, 2009.
[15] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[16] P. Benner, M. Ohlberger, A. Cohen, K. Willcox, Model Reduction and Approximation: Theory and Algorithms, SIAM, 2017.
[17] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, vol. 590, Springer, 2016.
[18] A. Quarteroni, A. Manzoni, F. Negri, Reduced Basis Methods for Partial Differential Equations: an Introduction, vol. 92, Springer, 2015.
[19] A. Benaceur, Reduced Order Modeling in Thermo-Mechanics (Ph.D. thesis), Université Paris-Est, 2018.
[20] A. Lorenc, A global three-dimensional multivariate statistical interpolation scheme, Mon. Weather Rev. 109 (4) (1981) 701–721.
[21] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Comput. 31 (7) (2019) 1235–1270.
[22] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with LSTM, Neural Comput. 12 (10) (2000) 2451–2471, http://dx.doi.org/10.1162/089976600300015015.
[23] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[24] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, P. Wojtaszczyk, Data assimilation in reduced modeling, SIAM/ASA J. Uncertain. Quantif. 5 (1) (2017) 1–29.
[25] M. Hutzenthaler, A. Jentzen, T. Kruse, Overcoming the curse of dimensionality in the numerical approximation of parabolic partial differential equations with gradient-dependent nonlinearities, Found. Comput. Math. 22 (4) (2022) 905–966, http://dx.doi.org/10.1007/s10208-021-09514-y.
[26] G. Kutyniok, P. Petersen, M. Raslan, R. Schneider, A theoretical analysis of deep neural networks and parametric PDEs, Constr. Approx. 55 (1) (2022) 73–125, http://dx.doi.org/10.1007/s00365-021-09551-4.
[27] M. Zeinhofer, R. Masri, K.-A. Mardal, A unified framework for the error analysis of physics-informed neural networks, 2023, arXiv preprint 2311.00529.

[28] B. Haasdonk, H. Kleikamp, M. Ohlberger, F. Schindler, T. Wenzel, A new certified hierarchical and adaptive RB-ML-ROM surrogate model for parametrized PDEs, SIAM J. Sci. Comput. 45 (3) (2023) A1039–A1065, http://dx.doi.org/10.1137/22M1493318.

[29] A. Ern, J.-L. Guermond, Theory and practice of finite elements, in: Applied Mathematical Sciences, vol. 159, Springer-Verlag, New York, 2004, p. xiv+524, http://dx.doi.org/10.1007/978-1-4757-4355-5.

[30] F. Hecht, New developments in FreeFem++, J. Numer. Math. 20 (3–4) (2012) 251–265.

[31] F. Chollet, et al., Keras, 2015, https://keras.io.

[32] Y. Maday, T. Taddei, Adaptive PBDW approach to state estimation: noisy observations; user-defined update spaces, SIAM J. Sci. Comput. 41 (4) (2019) B669–B693, http://dx.doi.org/10.1137/18M116544X.