A comparison of combined data assimilation and machine learning methods for offline and online model error correction

Alban Farchi, Marc Bocquet, Patrick Laloyaux, Massimo Bonavita, Quentin Malartic

Journal of Computational Science 55 (2021) 101468

Contents lists available at ScienceDirect

Journal of Computational Science

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

A comparison of combined data assimilation and machine learning methods for offline and online model error correction

Alban Farchi [a,*], Marc Bocquet [a], Patrick Laloyaux [b], Massimo Bonavita [b], Quentin Malartic [a,c]

[a] CEREA, École des Ponts and EDF R&D, Île-de-France, France
[b] ECMWF, Shinfield Park, Reading, United Kingdom
[c] LMD/IPSL École Normale Supérieure and PSL University, École Polytechnique, Université Paris-Saclay, Sorbonne Université, CNRS, Paris, France

ARTICLE INFO

Keywords:
Data assimilation
Machine learning
Model error
Surrogate model
Neural networks

ABSTRACT

Recent studies have shown that it is possible to combine machine learning methods with data assimilation to reconstruct a dynamical system using only sparse and noisy observations of that system. The same approach can be used to correct the error of a knowledge-based model. The resulting surrogate model is hybrid, with a statistical part supplementing a physical part. In practice, the correction can be added as an integrated term (*i.e.* in the model resolvent) or directly inside the tendencies of the physical model. The resolvent correction is easy to implement. The tendency correction is more technical, in particular it requires the adjoint of the physical model, but also more flexible. We use the two-scale Lorenz model to compare the two methods. The accuracy in long-range forecast experiments is somewhat similar between the surrogate models using the resolvent correction and the tendency correction. By contrast, the surrogate models using the tendency correction significantly outperform the surrogate models using the resolvent correction in data assimilation experiments. Finally, we show that the tendency correction opens the possibility to make online model error correction, *i.e.* improving the model progressively as new observations become available. The resulting algorithm can be seen as a new formulation of weak-constraint 4D-Var. We compare online and offline learning using the same framework with the two-scale Lorenz system, and show that with online learning, it is possible to extract all the information from sparse and noisy observations.

1. Introduction: machine learning for model error correction

Over the past decade, data-driven methods, and in particular machine learning (ML), have shown remarkable success in reproducing complex spatiotemporal processes, and have therefore been used in an increasing number of applications [1–3]. In the geosciences only, there is a fairly recent wealth of studies dealing with the problem of inferring the dynamics of a system from observations. Typical examples include the use of analogues, delay coordinates embedding, random forests, echo state networks and other neural networks such as residual, recurrent, or convolutional neural networks [4–12]. Most, if not all, of these examples implement a type of supervised learning where the goal is to minimise the loss function, a measure of the discrepancy between the statistical model (also called surrogate model) predictions and the observation dataset. The underlying assumption is that the system is fully observed without or with very little noise. In order to handle sparse and noisy observations, which is the case in most realistic systems in the geosciences, more and more studies consider the possibility of hybridising ML and data assimilation (DA) techniques [13–17]. In practice, DA tools are used, with the surrogate model, to estimate the state of the system from the observations while ML tools are used to estimate the surrogate model from the analysis (estimated) state. This method has been reformulated using a unifying Bayesian formalism by Bocquet et al. [16].

In the geosciences, even though models are affected by errors (*e.g.*, misrepresented physical phenomena, unresolved small-scale processes, numerical integration errors, etc.), they benefit from a long history of modelling and therefore they already provide a solid baseline. For this reason, recent studies focus on using ML techniques for model error correction instead of full model emulation [18–26]. The idea is to build a hybrid model with a physical, knowledge-based part, and a statistical part to supplement it. This means that the statistical model is trained to learn the error of the physical model. The underlying rationale is that model error correction should be an easier inference problem than full model emulation [20,21,26].

From a technical perspective, the geoscientific models are based on a set of physical laws, usually represented as ordinary or partial

[*] Corresponding author.
E-mail address: alban.farchi@enpc.fr (A. Farchi).

https://doi.org/10.1016/j.jocs.2021.101468
Received 23 July 2021; Received in revised form 7 September 2021; Accepted 28 September 2021
Available online 8 October 2021

differential equations (ODEs or PDEs). These equations define the *tendencies* of the model. A numerical scheme is used to integrate them for a small time step, and several integration steps are composed to define the *resolvent* between two forecast times. Following Farchi et al. [26], two strategies are possible for a correction term: (i) apply an integrated correction between two forecast times, *i.e.* in the resolvent, or (ii) apply a correction directly in the tendencies. The first method is by far the simplest to implement, which is why it is the most widely applied, but it faces some limitations, in particular when using the hybrid model for DA experiments. The first objective of the present paper is hence to make an exhaustive comparison of the two methods for both forecast and assimilation experiments in a simplified modelling framework.

Beyond the design of the model error correction – or more generally of any surrogate model – the question of the use of observations arise. In most cases, the statistical model is only trained once the entire observation dataset is available: this is called *offline learning*. The other option, *online*, or *sequential learning*, *i.e.* improving the surrogate model as new observations become available, is also possible in ML, even if it is less common because the methods usually require very large datasets to achieve good performance. In a context where information is only available through sparse and noisy observations, this means that we have to learn both the state of the system and the surrogate model at the same time. This is the topic of several recent studies [27,28], which emphasise the connections between this problem and classical parameter estimation in DA [29,30]. In the geosciences, online learning is more natural because observations are acquired sequentially, and improvements can be expected before having a long series of observations since the training begins from the first observation. Therefore, the second objective of the present paper is to explore the possibility to use online learning for model error correction.

The paper is organised as follows. Section 2 introduces the main methodological aspects for offline learning. We start with a brief overview of the Bayesian framework for combining DA and ML and how it can be used for model error correction. We then discuss the advantages and drawbacks of applying a correction term in the resolvent or in the tendencies, with an emphasis on the implications for forecast and assimilation applications. The two methods are compared in Section 3 using the two-scale Lorenz model [L05III, 31]. Section 4 further develops the methodology to enable online learning for model error correction. Section 5 illustrates the use of online learning with the same L05III model, and compares it to offline learning. Finally, conclusions are drawn in Section 6.

2. Offline learning of model error with resolvent or tendency correction

2.1. A Bayesian framework for data assimilation and machine learning

The starting point of the present work is a series of observations $\mathbf{y}_k \in \mathbb{R}^{N_\mathrm{y}}$ of a system at discrete times t_k for $k \in \mathbb{N}$. The state of the system is represented by a vector $\mathbf{x}_k \in \mathbb{R}^{N_\mathrm{x}}$. The observations are related to the state through the observation equation

$$\mathbf{y}_k = \mathcal{H}_k\left(\mathbf{x}_k\right) + \mathbf{v}_k, \tag{1}$$

where $\mathcal{H}_k : \mathbb{R}^{N_\mathrm{x}} \to \mathbb{R}^{N_\mathrm{y}}$ is the observation operator and $\mathbf{v}_k \in \mathbb{R}^{N_\mathrm{y}}$ the observation error at time t_k. We assume that the time evolution of the state is governed by the state equation

$$\mathbf{x}_{k+1} = \mathcal{M}^\mathrm{t}_k\left(\mathbf{x}_k\right) + \mathbf{w}_k, \tag{2}$$

where $\mathcal{M}^\mathrm{t}_k : \mathbb{R}^{N_\mathrm{x}} \to \mathbb{R}^{N_\mathrm{x}}$ is the resolvent of the (unknown) true dynamical model from t_k to t_{k+1}, and $\mathbf{w}_k \in \mathbb{R}^{N_\mathrm{x}}$ is the corresponding model error (*e.g.*, related to sub-scale processes). To simplify the presentation, we make the following assumptions:

- Observations are available at regular intervals $t_k = k\Delta t$;
- The observation operator is constant over time $\mathcal{H}_k \equiv \mathcal{H}$;
- The observation error is uncorrelated in time and normally distributed $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, where $\mathbf{R}$ is the observation error covariance matrix;
- the model error $\mathbf{w}_k$ is uncorrelated to the observation error $\mathbf{v}_k$.

In particular, the third point implies that the observations are not biased, which helps to attribute correctly the model errors. Furthermore, we also make the assumption that the true dynamical model is autonomous, in which case $\mathcal{M}^\mathrm{t}_k \equiv \mathcal{M}^\mathrm{t}_{\Delta t}$ the resolvent of the surrogate model for a Δt integration. The extension of the present work to non-autonomous dynamics is not trivial and briefly discussed in Section 5.6.

Our goal is to derive a surrogate of the true model, which can be used to predict $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$. Let $\mathbf{p}$ be the set of parameters defining the surrogate model. The discrepancy between the surrogate model predictions and the observations is measured with a cost function. A traditional ML approach to this problem is to use dense observations (*i.e.*, $\mathcal{H} = \mathbf{I}$ the identity operator) and to neglect the observation errors (*i.e.*, assuming that $\mathbf{R} = \mathbf{0}$), which yields the following cost function:

$$\mathcal{J}(\mathbf{p}) \triangleq \mathcal{L}(\mathbf{p}) + \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1} \left\| \mathbf{y}_{k+1} - \mathcal{M}_{\Delta t}\left(\mathbf{p}, \mathbf{y}_k\right) \right\|^2_{\mathbf{Q}_k^{-1}}, \tag{3}$$

where $\mathcal{L}$ is a regularisation (prior) term on $\mathbf{p}$, N_t is the number of observation batches used to define $\mathcal{J}$, and $\mathbf{x} \mapsto \mathcal{M}_{\Delta t}(\mathbf{p}, \mathbf{x})$ is the resolvent of the surrogate model for a Δt integration. The matrix norm notation $\|\mathbf{v}\|^2_\mathbf{A}$ stands for $\mathbf{v}^\top\mathbf{A}\mathbf{v}$, and $\mathbf{Q}_k$ is the model error covariance matrix at time t_k.

With sparse observations, the problem is more complex because in order to derive the surrogate model, we need to estimate the true state. A rigorous Bayesian approach to this problem consists in extending Eq. (3) to include the system trajectory $\mathbf{x}_0, \ldots, \mathbf{x}_{N_\mathrm{t}}$ in the control variables [13,14,16,32]. The joint cost function reads

$$\mathcal{J}\left(\mathbf{p}, \mathbf{x}_0, \ldots, \mathbf{x}_{N_\mathrm{t}}\right) \triangleq \mathcal{L}\left(\mathbf{p}, \mathbf{x}_0\right) + \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1} \left\| \mathbf{x}_{k+1} - \mathcal{M}_{\Delta t}\left(\mathbf{p}, \mathbf{x}_k\right) \right\|^2_{\mathbf{Q}_k^{-1}} + \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}} \left\| \mathbf{y}_k - \mathcal{H}\left(\mathbf{x}_k\right) \right\|^2_{\mathbf{R}^{-1}}, \tag{4}$$

where $\mathcal{L}$ is a regularisation term on both $\mathbf{p}$ and $\mathbf{x}_0$. The second term in Eq. (4) corresponds to the second term in Eq. (3), in which the observations have been replaced with the state, and the third term is the observation error term. Eq. (4) is overall very similar to a typical weak-constraint (WC) 4D-Var cost function [33].

Because the size of the trajectory control vector $N_\mathrm{t} \times N_\mathrm{x}$ is likely to be large, an efficient minimisation method relies on a coordinate descent technique, alternating DA steps to estimate the state with ML steps to estimate the surrogate model [15,16]. This combined DA–ML method, illustrated in Fig. 1, explicitly exploits the different nature between the arguments of $\mathcal{J}$ (state of the system and surrogate model parameters) and is highly flexible since the DA and ML steps are independent. A comprehensive description of the DA–ML method is given by Bocquet et al. [16].

The DA–ML method has first been used for full model emulation, *e.g.* by Brajard et al. [15]. In their example, the surrogate model is a neural network (NN) which represents the model tendencies. It is combined with an integration scheme to define the resolvent between two time steps. In this case, $\mathbf{p}$ corresponds to the set of weights and biases of the NN. The method has then been used to correct an imperfect physical model by Brajard et al. [23]; Farchi et al. [26]. For this problem, the formalism is simply obtained by replacing the resolvent of the surrogate model $\mathcal{M}_{\Delta t}$ with the resolvent of the corrected model, in particular in Eq. (4). In general, model error correction should be an easier inference problem than full model emulation, which means that smaller surrogate models (smaller in number of parameters) and less training data are necessary. Moreover, using a physical model is likely to be beneficial to the method, in particular during the DA steps. It also solves the issue of the initialisation: the first step of the method is to perform DA with the (non-corrected) physical model. The advantages of model error correction over full model emulation are further investigated in [26].

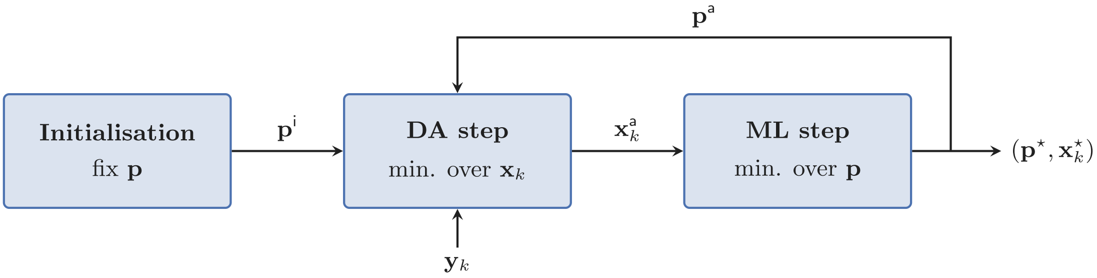
$\mathbf{p}^{\mathrm{a}}$
Initialisation
fix $\mathbf{p}$
$\mathbf{p}^{\mathrm{i}}$
DA step
min. over $\mathbf{x}_k$
$\mathbf{x}_k^{\mathrm{a}}$
ML step
min. over $\mathbf{p}$
$(\mathbf{p}^\star, \mathbf{x}_k^\star)$
$\mathbf{y}_k$

Fig. 1. Illustration of the DA–ML method for the minimisation strategy of the cost function, Eq. (4): alternate DA steps with ML steps to estimate the model parameters $\mathbf{p}$ and the state trajectory $\mathbf{x}_0, \ldots, \mathbf{x}_{N_t}$ with an increasing accuracy.

2.2. A typical geophysical model architecture

In the present work, we investigate model error correction with the DA–ML method. Before we introduce any model error correction, we need to discuss the characteristics of the model to correct. The geophysical models rely on physical laws, which most of the time take the form of ODEs or PDEs.

The core of a model is a numerical code computing the model *tendencies* ϕ, which are defined as a discretised version of the differential equations in $\mathbb{R}^{N_x}$:

$$\phi(\mathbf{x}) \triangleq \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}. \tag{5}$$

The model tendencies are integrated over time step δt using a dedicated integration scheme, for example the explicit Euler scheme:

$$\mathcal{I}(\mathbf{x}) \triangleq \mathbf{x} + \delta t \cdot \phi(\mathbf{x}), \tag{6}$$

or more elaborate schemes such as Runge–Kutta methods. Finally, several integration steps are composed to define the *resolvent*,[1] from one time step to the next:

$$\mathcal{M}_{\Delta t}(\mathbf{x}) \triangleq \mathcal{I} \circ \ldots \circ \mathcal{I}(\mathbf{x}). \tag{7}$$

Two strategies can be used to correct such physical model. The first one is to include a correction in the resolvent, Eq. (7). This is called resolvent correction (RC). The other strategy is to include the correction directly in the differential equations, in other words in the model tendencies, Eq. (5) [14]. This is called tendency correction (TC). According to Farchi et al. [26], both strategies have advantages and drawbacks. Let us illustrate the difference using a simple univariate example.

2.3. Resolvent or tendency correction in a simple univariate example

Suppose that we follow the evolution of a process $x \in \mathbb{R}$ over two δt-integration steps with the explicit Euler scheme. The true two-step resolvent is given by

$$\mathcal{M}_2^{\mathrm{t}}(x) = x + \delta t \cdot f(x) + \delta t \cdot f\{x + \delta t \cdot f(x)\}, \tag{8}$$

where f represents the true model tendencies. Our imperfect physical model has tendencies g and a two-step resolvent given by

$$\mathcal{M}_2^{\mathrm{p}}(x) = x + \delta t \cdot g(x) + \delta t \cdot g\{x + \delta t \cdot g(x)\}. \tag{9}$$

To simplify the expressions, in the following we take $\delta t = 1$.

When using a TC, we assume that the corrected model has tendencies $g + \alpha$, whereas when using RC, we assume that the two-step resolvent of the corrected model is $\mathcal{M}_2^{\mathrm{p}} + \beta$. The optimal α and β corrections are given by

$$\alpha^\star(x) = f(x) - g(x), \tag{10}$$

$$\beta^\star(x) = \mathcal{M}_2^{\mathrm{t}}(x) - \mathcal{M}_2^{\mathrm{p}}(x) \tag{11}$$

$$= f(x) - g(x) + f\{x + f(x)\} - g\{x + g(x)\}, \tag{12}$$

where the difference is highlighted in red. Obviously, the optimal β is likely to be more complex than the optimal α. The expression suggests that it will also be more nonlinear if f or g (or both) are nonlinear.

To further understand the difference, we derive the two-step resolvent with RC and TC, respectively written $\mathcal{M}_2^\beta$ and $\mathcal{M}_2^\alpha$:

$$\mathcal{M}_2^\beta(x) = x + g(x) + g\{x + g(x)\} + \beta(x) \tag{13}$$

$$\mathcal{M}_2^\alpha(x) = x + g(x) + g\{x + g(x) + \alpha(x)\} + \alpha(x) + \alpha\{x + g(x) + \alpha(x)\}, \tag{14}$$

where the difference is highlighted in red. From this perspective, it is clear that with TC, $\mathcal{M}_2^\alpha(x)$ is marked by the interaction between the physical model and the correction term α. While this interaction is beneficial because it enhances $\mathcal{M}_2^\alpha(x)$, the downside is that inferring α from data is technically more difficult than inferring β. Let us see why.

Suppose that both α and β depend on a coefficient $p \in \mathbb{R}$. Observation data usually come in the form of pairs (x_0, x_2) with $x_2 = \mathcal{M}_2^{\mathrm{t}}(x_0)$, possibly with some observation noise. Therefore, a learning step based on some kind of gradient descent would required the gradient of the *corrected* two-step resolvent with respect to p, which is given by

$$\frac{\partial \mathcal{M}_2^\beta}{\partial p}(x) = \frac{\partial \beta}{\partial p}(x), \tag{15}$$

$$\frac{\partial \mathcal{M}_2^\alpha}{\partial p}(x) = \frac{\partial \alpha}{\partial p}(x) \cdot \left[1 + g'\{x + g(x) + \alpha(x)\} + \frac{\partial \alpha}{\partial p}\{x + g(x) + \alpha(x)\}\right], \tag{16}$$

where the difference is once again highlighted in red. In particular, it depends on g', the derivative of g. The equivalent for a geophysical numerical model would be the tangent linear (TL) operator, which may be difficult to compute.

To summarise, compared to RC, TC is more difficult to program because the correction term (α in the present example) is intrusive, meaning that it requires to modify deeply the code of the physical model. It is also more difficult to train, as illustrated by the difference between Eqs. (15) and (16). On the other hand, once it is implemented, the TC has the potential to yield richer dynamics through the interaction with the physical model. Furthermore, by construction the RC can only correct the two-step resolvent, while the TC can also correct the one-step resolvent. This would make a difference when using the corrected model in a DA experiment with observations at every step, because then the one-step resolvent is explicitly needed. The simplest workaround is to assume a linear growth of errors in time [23,26]. In this case, the one-step resolvent with RC would be given by

$$\mathcal{M}_1^\beta(x) = x + g(x) + \frac{1}{2}\beta(x), \tag{17}$$

[1] The term *resolvent* is usual in the context of integral or differential equations. The same operator is often called *flow* or *flow map* in dynamical systems and *propagator* in theoretical physics.

since β is the correction term for the two-step resolvent $\mathcal{M}_2^{\beta}(x)$. However, even with the optimal β correction from Eq. (12), this one-step resolvent would still differ from the true one-step resolvent, given by

$$\mathcal{M}_1^{\mathrm{t}}(x) = x + f(x). \tag{18}$$

In their experiments, Farchi et al. [26] found that this hypothesis was the main limitation for improving the accuracy of DA experiments with the corrected model. They concluded that the best strategy to correct a model to be used in DA experiments would probably be the TC.

Finally, let us mention that the model error correction considered in this section is autonomous and additive. The autonomous hypothesis can be relaxed, for example by including time in the set of predictors. However, one must keep in mind that in this case, the training dataset should capture the time evolution of the model error. The additive hypothesis can also be relaxed. Without prior knowledge on the model error form, using an additive correction is the simpler option but other choices are possible, *e.g.* a multiplicative correction. Also note that if the physical model explicitly depends a set of parameters, the same framework can be used to calibrate these parameters.

2.4. Comparing resolvent and tendency correction

Farchi et al. [26] chose to focus on the RC because it is easier to implement. In the following section, we illustrate the difference between RC and TC. First hints in favour of the TC approach were gained from the comparison of the results of Bocquet et al. [14] and of Brajard et al. [15] on the Lorenz 40-variable model. To address the inference problem, we use the combined DA–ML method described in Section 2.1. We start by a DA step with an imperfect physical model to assimilate the observations. We then use a ML step to train a model error correction from the analysis of the DA step. Pushing the DA–ML method further, we could iterate in place: use the corrected model to get a more accurate analysis in further DA steps and learn from this more accurate analysis to get an improved model error correction in further ML steps, as illustrated by Fig. 1.

However, we choose to stop after the first DA–ML cycle for two reasons. First, DA experiments with realistic models are numerically expensive, and it may not be realistic to perform more than one DA step if the size of the trajectory N_{t} is large. It is also worth noting that operational centres usually compute *reanalyses*, which means that the first DA step is a product which is likely to be already available [34]. Second, if the physical model (without correction) is reasonably accurate, the analysis of the first DA step should be reasonably accurate and hence the improvement of the first ML step should be much larger than the improvement of further ML steps. However, even though we stop after the first DA–ML cycle, we perform a second DA step, but only for evaluation purposes.

Finally, we emphasise again the offline nature of the DA–ML method previously discussed. As is, the ML step starts only when the DA analysis is available, *i.e.* once the entire observation dataset has been assimilated in the DA step. An alternative, online approach is proposed in Section 4.

3. Numerical illustration with the two-scale Lorenz model (part I)

3.1. Models description

In our experiments, the true model is the L05III model, which describes the evolution of two sets of variables: the slow variables x_n for $n \in \{1, \dots, N_{\mathrm{x}}\}$ and the fast variables u_m for $m \in \{1, \dots, N_{\mathrm{x}} \times N_{\mathrm{u}}\}$. These two-scale dynamics are given by

$$\frac{\mathrm{d}x_n}{\mathrm{d}t} = x_{n-1}\left(x_{n+1} - x_{n-2}\right) - x_n + F - \frac{hc}{b}\sum_{m=1}^{N_{\mathrm{u}}} u_{m+(n-1)N_{\mathrm{u}}}, \tag{19a}$$

$$\frac{\mathrm{d}u_m}{\mathrm{d}t} = \frac{c}{b}\left\{b^2 u_{m+1}\left(u_{m-1} - u_{m+2}\right) - b u_m\right\} + \frac{hc}{b} x_{1+(m-1)/\!/N_{\mathrm{u}}}, \tag{19b}$$

where // is the integer division and where the indices are applied periodically: $x_{N_{\mathrm{x}}+n} = x_n$ and $u_{N_{\mathrm{x}} \times N_{\mathrm{u}}+m} = u_m$. The idea is that each slow variable x_n is coupled to the N_{u} fast variables u_m for $m \in \left\{1 + (n-1)N_{\mathrm{u}}, \dots, nN_{\mathrm{u}}\right\}$.

Table 1
Parametrisation for the true (L05III) and physical (L96) models.

Parameter	Symbol	L05III	L96
Number of slow variables	N_{x}	36	36
Number of fast variables per slow variable	N_{u}	10	
Forcing	F	10	8
Coupling	h	1	
Time-scale ratio	c	10	
Space-scale ratio	b	10	
Integration time step	δt	0.005	0.05

A first order approximation of the L05III model is the one-scale Lorenz model [L96, 35], which only describes the evolution of the slow variables x_n. The model is defined by

$$\frac{\mathrm{d}x_n}{\mathrm{d}t} = x_{n-1}\left(x_{n+1} - x_{n-2}\right) - x_n + F, \tag{20}$$

where the indices once again apply periodically: $x_{N_{\mathrm{x}}+n} = x_n$. This model is used in our experiments as the (imperfect) physical model to correct.

Both L05III and L96 models are integrated using a fourth-order Runge–Kutta scheme, and the parameter values are reported in Table 1. With this setup, the true model dynamics is chaotic, with a leading Lyapunov exponent of 1.3775 [36] and the model variability, defined as the standard deviation of the climatological distribution of the state, averaged over the slow variables, is 3.5372. When using the L96 model in place of the L05III model, two sources of model error are introduced:

1. The fast variables u_m generate unresolved processes;
2. The integration time step δt is 0.05 instead of 0.005.

Moreover, even though the forcing coefficient F differs in both models, this cannot strictly be considered as a third source of model error as $F = 10$ is chosen for the L05III model to better match the dynamics of the L96 model with $F = 8$.

The accuracy of the physical (L96) model in reproducing the dynamics of the true (L05III) model is measured using the forecast skill (FS) defined as the average root-mean-squared error (RMSE) of the forecast after a given lead time:

$$\mathrm{FS}(k\delta t) \triangleq \frac{1}{N_{\mathrm{e}}}\sum_{i=1}^{N_{\mathrm{e}}} \mathrm{RMSE}\left[\boldsymbol{\mathit{\Pi}} \circ \mathcal{M}_{k\delta t}^{\mathrm{t}}\left(\mathbf{x}_i, \mathbf{u}_i\right), \mathcal{M}_{k\delta t}^{\mathrm{p}}\left(\mathbf{x}_i\right)\right]. \tag{21}$$

In this equation, $\mathcal{M}_{k\delta t}^{\mathrm{t}}$ and $\mathcal{M}_{k\delta t}^{\mathrm{p}}$ are the resolvents of the true and physical models for a $k\delta t$ integration, respectively, $\boldsymbol{\mathit{\Pi}}$ is the projection operator onto the set of slow variables $\boldsymbol{\mathit{\Pi}}(\mathbf{x}, \mathbf{u}) = \mathbf{x}$, and $\left(\mathbf{x}_i, \mathbf{u}_i\right)$ for $i \in \left\{1, \dots, N_{\mathrm{e}}\right\}$ is a set of N_{e} initial conditions representative of the true model climatology. The FS, normalised by the model variability, is shown in Fig. 2a and illustrates the poor accuracy of the physical model. In the following sections, we will see how model error corrections can be used to improve the FS, but one must keep in mind that there is an intrinsic limit to potential improvements, because it is presumably impossible to exactly reproduce the dynamics of the true model with only $N_{\mathrm{x}} = 36$ variables.

3.2. Data assimilation with the physical model

The first step of the DA–ML method is to perform DA with the physical model. The truth $\left(\mathbf{x}_k^{\mathrm{t}}, \mathbf{u}_k^{\mathrm{t}}\right)$ is generated using the true model. Observations are taken every $\Delta t = 0.05$ from the slow variables only, using

$$\mathbf{y}_k = \mathbf{x}_k^{\mathrm{t}} + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{22}$$

In other words, the observation operator is $\mathcal{H} = \mathbf{I}$, the observations are not biased, and the observation error covariance matrix is $\mathbf{R} = \mathbf{I}$.

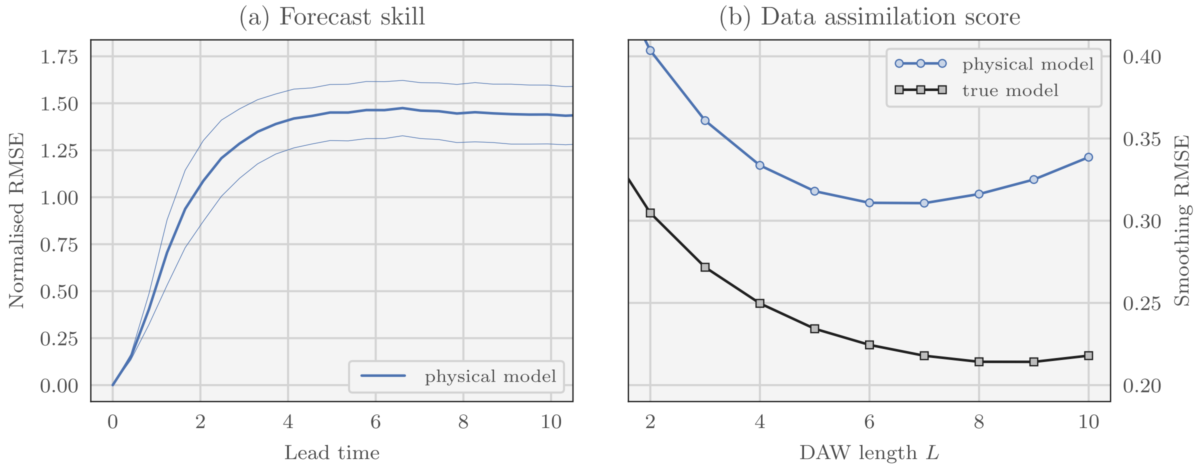

Fig. 2. Left panel (a): forecast skill of the physical model (in units of the model variability) as a function of the lead time (in units of the Lyapunov time). The thick line shows the average over the $N_e = 1024$ initial conditions and the thin lines indicate plus or minus one standard deviation. Right panel (b): accuracy of the DA step as a function of the length of the DAW L with the physical model (in blue) and with the true model (in black). The sRMSE is averaged over at least $8192/\!/L$ cycles after a spin-up period of at least $1024/\!/L$ cycles, and over 16 repetitions of each experiment. For each value of L, b is optimally tuned to yield the lowest sRMSE. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Numerical illustrations with sparse observation operators are provided, *e.g.*, by Brajard et al. [15]; Bocquet et al. [16]; Brajard et al. [23]; Farchi et al. [26]. Except in the case of very sparse observations, the results are qualitatively similar over a wide range of observation density, which is why, for the present study, we have chosen to use dense observations for simplicity.

As explained in Section 2.1, the goal of the DA step is to minimise Eq. (4) with respect to the state trajectory $\mathbf{x}_0, \ldots, \mathbf{x}_{N_t}$, which is a WC 4D-Var problem. However, solving a WC minimisation is probably unaffordable for large trajectories, as discussed by Bocquet et al. [16]. To overcome this issue, we choose to assimilate the observations using the cycled strong-constraint (SC) 4D-Var algorithm, with consecutive DA windows (DAWs) of L batches of observations. This will provide an approximate solution to the WC 4D-Var problem. More specifically, each 4D-Var problem consists in minimising the cost function

$$\mathcal{J}\left(\mathbf{x}_k\right) = \frac{1}{2}\left\|\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\sum_{l=0}^{L-1}\left\|\mathbf{y}_{k+l} - \mathcal{M}^{\mathrm{p}}_{l\Delta t}\left(\mathbf{x}_k\right)\right\|^2_{\mathbf{R}^{-1}}, \tag{23}$$

where $\mathbf{x}_k^{\mathrm{b}}$ is the background, $\mathbf{B}$ is the background error covariance matrix, $\left\{\mathbf{y}_k, \ldots, \mathbf{y}_{k+L-1}\right\}$ is the set of assimilated observations, and $\mathcal{M}^{\mathrm{p}}_{l\Delta t}$ is the resolvent of the physical model for an integration of $l\Delta t$. The analysis is performed at time t_k (the time of the first batch of assimilated observations) and, in this cycled context, it is used to obtain the background for the next analysis which is performed at time t_{k+L}, using

$$\mathbf{x}^{\mathrm{b}}_{k+L} = \mathcal{M}^{\mathrm{p}}_{L\Delta t}\left(\mathbf{x}^{\mathrm{a}}_k\right). \tag{24}$$

For the first cycle, the background state is obtained by perturbing the truth:

$$\mathbf{x}^{\mathrm{b}}_0 = \mathbf{x}^{\mathrm{t}}_0 + \mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right). \tag{25}$$

Finally, the background error covariance matrix $\mathbf{B}$ is set to $b^2\mathbf{I}$, where b is an algorithmic parameter to specify.

At each cycle, the cost function $\mathcal{J}$, Eq. (23), is minimised using the L-BFGS algorithm [37], a quasi-Newton minimisation algorithm. The gradient of $\mathcal{J}$ is computed exactly using automatic differentiation, and the starting point of the minimisation is $\mathbf{x}^{\mathrm{b}}_k$. The accuracy of the DA step is measured using the RMSE of the analysis (analysis minus truth) at the start of the DAW, hereafter called the *smoothing* RMSE (sRMSE), averaged over a sufficiently large number of cycles to ensure the convergence of the statistical indicators.

In order to choose an appropriate value for the length of the DAW L, we first study the evolution of the sRMSE as a function of L. The results are shown in Fig. 2b. As expected, the sRMSE starts by decreasing with L. It reaches an optimum for $L = 6$, and then increases with L as the impact of model error grows. For comparison, Fig. 2 also shows the results when using the true model in place of the physical model. Note that in this case the 4D-Var cost function $\mathcal{J}$, Eq. (23), depends on both the slow and the fast variables $\mathbf{x}_k$ and $\mathbf{u}_k$. The evolution of the sRMSE as a function of L is very similar, with the exception that the scores are overall much lower, and that the sRMSE increase for large values of L does not come from model error but from optimisation issues. Indeed, for long DAWs, the cost function $\mathcal{J}$ is likely to have several local minima, which would make the L-BFGS algorithm not suited for the minimisation. Using a quasi-static formulation of 4D-Var could mitigate this issue [38,39].

3.3. Model error correction with a univariate polynomial regression

The present model error setup, as described in Section 3.1, has already been addressed outside the scope of ML, for example by Wilks [40]. The idea is to replace the physical model tendencies, Eq. (20), by

$$\frac{\mathrm{d}x_n}{\mathrm{d}t} = x_{n-1}\left(x_{n+1} - x_{n-2}\right) - x_n + F + g\left(x_n\right), \tag{26}$$

where g is a univariate fourth-order polynomial correction, shared between all $N_{\mathrm{x}} = 36$ slow variables. The five coefficients of g are computed using a least-square regression of the difference between Eq. (20) and the empirical tendencies

$$\frac{x^{\mathrm{t}}_n\left(t + \delta t\right) - x^{\mathrm{t}}_n\left(t\right)}{\delta t} \tag{27}$$

computed from a trajectory $\mathbf{x}^{\mathrm{t}}(t)$ of the true model.

Offering a baseline score for later comparison, Fig. 3 shows the FS and the DA score for the model with the polynomial regression g. For this illustration, following the approach of Wilks [40], the coefficients of g are computed using 2000 pairs of snapshots $\left(\mathbf{x}^{\mathrm{t}}(t), \mathbf{x}^{\mathrm{t}}(t+\delta t)\right)$ with $\delta t = 0.005$, the integration time step of the true model. The time interval between two consecutive pairs of snapshots is set to 1000 integration steps. The results show that this simple correction is effective, both in forecast and DA experiments. In particular, the DA score is very close to the one obtained with the true model. It is even better for $L \geq 10$. This probably comes from the fact that a small amount of model error regularises the cost function Eq. (23) and mitigates the numerical issues discussed at the end of Section 3.2.

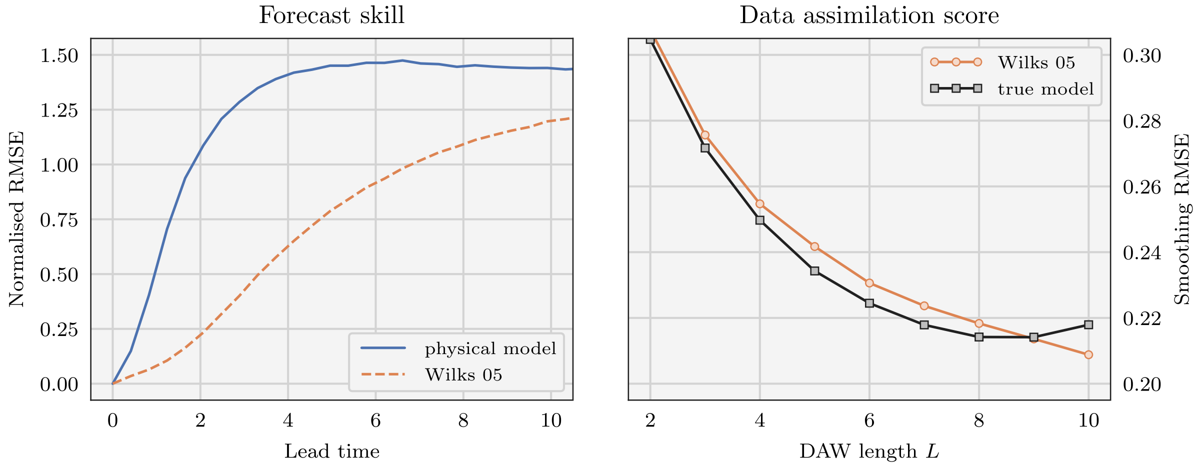

Fig. 3. Same as Fig. 2 for the physical model (in blue), the model with polynomial regression (in orange) and the true model (in black). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Of course, this method cannot be applied to realistic models because the regression requires the truth, at a very high frequency ($\delta t = 0.005$). We have checked that using $\delta t = 0.05$ (the integration time step of the physical model) yields an ineffective correction. Nevertheless, it shows that the model error structure in this setup can be effectively represented with a small number of parameters. The TC framework presented in Section 2 can be seen as a generalisation of the method of Wilks [40] to (i) any kind of correction g, in particular multivariate ones, and (ii) sparse and noisy observations for the training. A more complex but less scalable model error correction scheme has also been proposed for the same model by Pulido et al. [30].

3.4. Resolvent and tendency correction with the DA–ML method

3.4.1. The data assimilation step

Given the results of Section 3.2, we start the DA–ML method by a long DA experiment with the physical model and with $L = 6$. At each cycle, we only keep the analysis at the start of the DAW. The result is a time series of analysis snapshots $\mathbf{x}^{\mathrm{a}}_{kL}$, where the time interval between two snapshots is $L\Delta t = 0.3$. This trajectory is used to build the training dataset for the ML step. In other words, the surrogate models are trained to reproduce the map

$$\mathbf{x}^{\mathrm{a}}_{kL} \mapsto \mathbf{x}^{\mathrm{a}}_{(k+1)L}. \tag{28}$$

Another trajectory, resulting from a distinct long DA experiment, is used to build the validation dataset. Finally, since the ultimate goal is to predict the true dynamics and not the dynamics of the analysis snapshots, we compute an additional trajectory, this time with the true model. This third trajectory $\mathbf{x}^{\mathrm{t}}_{kL}$ is used to build the test dataset, and hence to evaluate the ability of the surrogate models to reproduce the map

$$\mathbf{x}^{\mathrm{t}}_{kL} \mapsto \mathbf{x}^{\mathrm{t}}_{(k+1)L}. \tag{29}$$

3.4.2. Designing the surrogate models

The second step of the DA–ML consists in defining and training a surrogate model with the analysis of the first DA step. In this section, three different surrogate models are tested to correct the physical model. All three of them are autonomous and use NNs. For the first surrogate, the correction is computed using a NN called CNN-a and then *added to the resolvent* of the physical model, following the RC approach, which yields

$$\mathcal{M}^{\mathrm{a}}_{L\Delta t}(\mathbf{p}, \mathbf{x}) \triangleq \mathcal{M}^{\mathrm{p}}_{L\Delta t}(\mathbf{x}) + \mathcal{F}^{\mathrm{a}}(\mathbf{p}, \mathbf{x}). \tag{30}$$

In this equation, $\mathcal{M}^{\mathrm{p}}_{L\Delta t}$ is the resolvent of the physical model for an integration of $L\Delta t$ (one DAW), $\mathcal{F}^{\mathrm{a}}$ is the map encoding CNN-a, $\mathbf{p}$ is the set of parameters of CNN-a (the weights and biases of the NN), and $\mathcal{M}^{\mathrm{a}}_{L\Delta t}$ is the resolvent of the resulting surrogate model, called RC-CNN-a. For the second surrogate, the correction is computed using a NN called CNN-b and then *added to the tendencies* of the physical model, following the TC approach:

$$\phi^{\mathrm{b}}(\mathbf{p}, \mathbf{x}) \triangleq \phi^{\mathrm{p}}(\mathbf{x}) + \mathcal{F}^{\mathrm{b}}(\mathbf{p}, \mathbf{x}). \tag{31}$$

In this equation, ϕ^{p} represents the physical model tendencies, given by Eq. (20), $\mathcal{F}^{\mathrm{b}}$ is the function encoding CNN-b, $\mathbf{p}$ is the set of parameters of CNN-b, and ϕ^{b} represents the tendencies of the resulting surrogate model, called TC-CNN-b. To compute the resolvent of this model, $\mathcal{M}^{\mathrm{b}}_{L\Delta t}$, we keep the integration scheme and time step of the physical model. Finally, the third surrogate model, called TC-CNN-c, is similar to TC-CNN-b with CNN-b replaced with another NN called CNN-c, which uses a different activation function.

As explained in Section 3.3, the model error structure is not overly complex. For this reason, we want to keep the NNs as simple as possible. We have experimented with several NNs configurations and have selected the following sequential (or feed-forward) architecture with:

1. The input layer;
2. A sequence of convolutional layers;
3. A final convolutional layer as output layer (without activation).

All intermediate convolutional layers share the same number of filters, the same convolutional window, and the same activation function. They also use periodic padding to preserve the input and output shape of the layers. The last convolutional layer uses only one filter, a convolution window of only one variable, and no activation function. The purpose of this layer is not to actually perform a convolution, but to project the output of the previous layer to the output variables. The settings of the intermediate convolutional layers are reported in Table 2 for CNN-a, CNN-b, and CNN-c, alongside the total number of parameters.

3.4.3. Neural networks initialisation

When working with NNs, the parameter initialisation step is important. A common method is to use random values for the initial weights and to set the initial biases to zero. The underlying idea is that there is no reason for the optimal weights to display any specific symmetry. Because such symmetries are preserved during the training, even with stochastic gradient descent, they need to be broken during the initialisation, hence the use of random initial values [2].

In our case however, the situation is different because the surrogate models are hybrid. Since the corrections are additive, all three

Table 2
Settings of the convolutional layers of the NNs used in the DA–ML method. The absence of an activation function is tantamount to a linear activation function.

Setting	CNN-a	CNN-b	CNN-c
Number of layers	4	1	1
Number of filters per layer	16	16	16
Size of the convolutional window	5	5	5
Activation function	tanh		tanh
Total number of parameters	4001	113	113

surrogate models are equivalent to the physical model when $\mathbf{p} = \mathbf{0}$, and it is highly probable that a random $\mathbf{p}$ would make the model predictions worse. Initialising the NNs with $\mathbf{p} = \mathbf{0}$ would hence make sense, but for the reasons aforementioned, this could yield suboptimal surrogate models after the training. Therefore, we initialise the NNs using the following approach, which we found to be a good compromise. The intermediate convolutional layers are initialised using the classical method in ML (random weights and zero biases) and the last convolutional layer is initialised to zero (both zero weights and zero biases). This approach is very similar to the ReZero method developed by Bachlechner et al. [41].

3.4.4. Training the surrogate models

We now start the ML step. The surrogate model parameters $\mathbf{p}$ are optimised using the Adam algorithm, a variant of the stochastic gradient descent [42]. The loss function is the mean-squared error (MSE) over the training dataset, made of analysis snapshots. The training consists of 1024 epochs with a learning rate of 1×10^{-3} and a batch size of 32. After the entire training step, we keep the model which yields the lowest MSE over the validation dataset, also made of analysis snapshots. This is necessary since the cost functions of the NNs do not include any internal mechanism to mitigate overfitting (*e.g.* regularisation). Finally, we evaluate the trained model by computing the MSE over the test dataset, made of truth snapshots, hereafter called test MSE (tMSE). For comparison, we also train and evaluate the surrogate models using the exact same method but with snapshots from the truth (instead of the analysis) in the training and validation datasets. This is equivalent to using dense and noiseless observations.

Fig. 4 shows the training results for datasets of increasing size. Note that, in order to build a dataset of size N_t, the total number of cycles (or DAWs) required in the preliminary DA step is $2 \times \left(N_\mathrm{t} + 1\right)$:

- $N_\mathrm{t} + 1$ cycles to produce the N_t pairs of analysis snapshots $\left(\mathbf{x}^\mathrm{a}_{kL}, \mathbf{x}^\mathrm{a}_{(k+1)L}\right)$ for the training dataset;
- and the same for the validation dataset.

Let us first discuss the training with the truth, because it shows the full potential of each model. First, all three models do improve over the physical model and yield very low tMSEs, but the scores with TC are significantly better than with RC. As explained in Section 2.3, the models with TC benefit from the interaction between the physical model and the correction term (the NN). This is even more important here than in the univariate example of Section 2.3, because here the number of interactions for a prediction is 24: 4 interactions per integration step (through the fourth-order Runge–Kutta scheme) times 6 integration steps between two DAWs. Furthermore, the training dataset needs to be much larger to get an accurate model with RC than with TC. This is consistent with the total number of trainable parameters for each model, reported in Table 1: 4001 for RC-CNN-a and only 113 for TC-CNN-b and TC-CNN-c. It is remarkable that with a training dataset of only one pair of truth snapshots (the smallest possible training dataset), the models with TC are already very accurate, more than RC-CNN-a trained with the largest dataset considered in the present study! Obviously, this is only possible because the correction is autonomous. The overall accuracy of TC-CNN-b is also remarkable, given the fact that the correction provided by CNN-b is linear. The only difference between CNN-b and CNN-c is their activation function for the intermediate layers. This means that the difference between TC-CNN-b and TC-CNN-c illustrates the nonlinearity of the error in the model tendencies: weak but non-negligible. For comparison, we have checked that with RC, the accuracy of the surrogate models built using linear NNs is not satisfactory. This shows that the nonlinearity of the dynamics over one DAW, Eq. (29), is significant and that estimating and correcting model error as a tendency forcing is a good first-order strategy to mitigate the effects of nonlinearities. For completeness, we mention that better scores could have been obtained with RC, for example with larger or deeper NNs. However, this would increase the number of trainable parameter, which means that the training dataset should be even larger.

When training with the analysis, the accuracy of the surrogate models is much lower, which was expected, but all three models are still able to improve over the physical model. Most of the conclusions hold: the scores with TC are better than with RC and require much smaller datasets. However this time, the linear TC-CNN-b outperforms the nonlinear TC-CNN-c. This is probably a sign that nonlinear NNs are harder to train.

3.4.5. Forecast skill of the surrogate models

The tMSE is a measure of the accuracy of a model for an integration of one DAW of $L\Delta t = 0.3$. In this section, we measure the accuracy of the surrogate models for longer forecast, by computing the FS as defined in Section 3.1. In order not to penalise RC-CNN-a over TC-CNN-b and TC-CNN-c, we evaluate the surrogate models which have been trained with the largest dataset.

The results are shown in Fig. 5 and demonstrate that the models, trained for an integration of one DAW, remain effective for much longer integrations. When training with the truth, the ranking of the three surrogate models is clear and consistent with the tMSE results: TC-CNN-c is the most accurate, followed by TC-CNN-b, and the least accurate is RC-CNN-a. When training with the analysis, the ranking of the models for long integrations (*e.g.* longer that 6 DAWs) is the opposite of the ranking for the tMSE results: RC-CNN-a is the most accurate, very closely followed by TC-CNN-c, and the least accurate is TC-CNN-b, the only model built using a linear NN. There is a crossover in the forecast error curves after 2 or 3 DAWs, with the errors of the linear NN (TC-CNN-b) becoming larger than those of the nonlinear NNs (RC-CNN-a and TC-CNN-c). This issue is not specific to the L05III model nor to the use of nonlinear NNs since Farchi et al. [26] also faced the same kind of issues with a quasi-geostrophic model and with linear NNs. We do not have yet a convincing explanation for this behaviour, which is specific to the use of the analysis for the training, and understanding it better will require further work. In any case, we conclude that, when the surrogate models are trained with the analysis, the accuracy for long forecasts is somewhat similar with RC and with TC and requires nonlinear corrections.

3.4.6. Data assimilation experiments with the surrogate models

In this section, the goal is to measure the accuracy of the surrogate models in DA experiments. To do that, we reimplement the 4D-Var setup described in Section 3.2 replacing the physical model by one of the surrogate models. In particular, in the cost function Eq. (23), $\mathcal{M}^\mathrm{p}_{l\Delta t}$, the resolvent of the physical model for an integration of $l\Delta t$, is replaced with $\mathcal{M}^\mathrm{a}_{l\Delta t}$, $\mathcal{M}^\mathrm{b}_{l\Delta t}$, or $\mathcal{M}^\mathrm{c}_{l\Delta t}$, the resolvents of RC-CNN-a, TC-CNN-b, or TC-CNN-c for an integration of $l\Delta t$. While the construction of $\mathcal{M}^\mathrm{b}_{l\Delta t}$ and $\mathcal{M}^\mathrm{c}_{l\Delta t}$ is similar to that of $\mathcal{M}^\mathrm{p}_{l\Delta t}$ because TC-CNN-b and TC-CNN-c use the TC method, the construction of $\mathcal{M}^\mathrm{a}_{l\Delta t}$ requires an additional assumption because RC-CNN-a uses the RC method. As discussed in Section 2.3, the simplest assumption is the linear growth of errors in time, for which

$$\mathcal{M}^\mathrm{a}_{\Delta t} \triangleq \mathcal{M}^\mathrm{p}_{\Delta t} + \frac{1}{L}\mathcal{F}^\mathrm{a}, \tag{32}$$

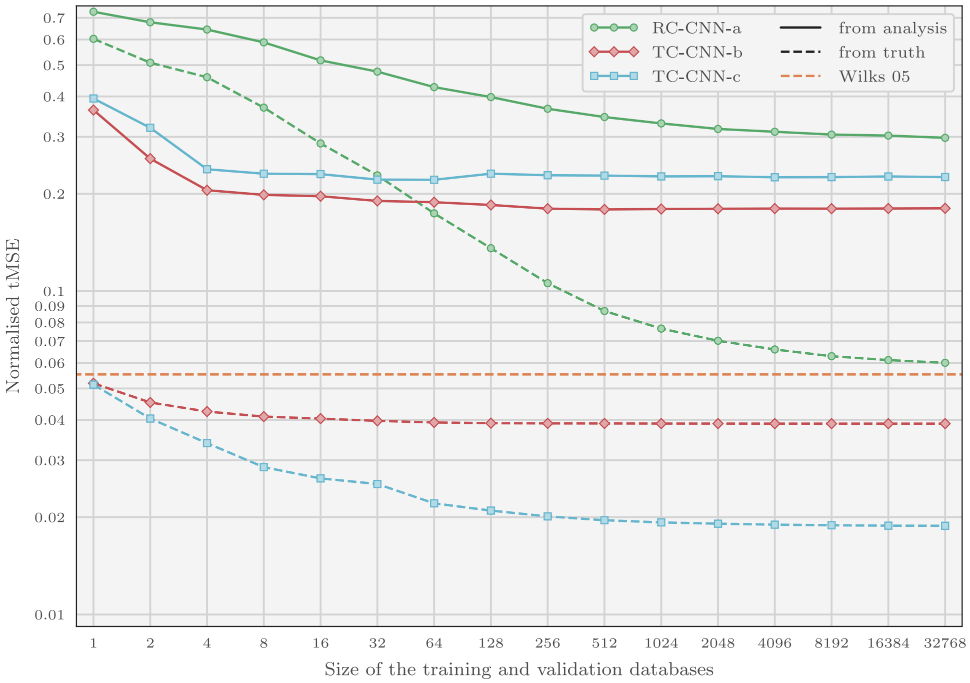

Fig. 4. Evolution of the tMSE of the trained surrogate model as a function of the size of the training and validation datasets for RC-CNN-a (in green), TC-CNN-b (in red), and TC-CNN-c (in cyan). The tMSE is computed using a test dataset of size $N_t = 8192$ and then normalised by the tMSE of the physical model (0.277 85). Each experiment (initialisation, training and evaluation) is repeated 16 times with different training, validation, and test datasets and the final score is averaged over the 16 repetitions. The surrogate models are trained either with the analysis (continuous lines) or with the truth (dashed lines). For comparison, the horizontal dashed orange line indicates the score for the model with the polynomial regression of Wilks [40]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

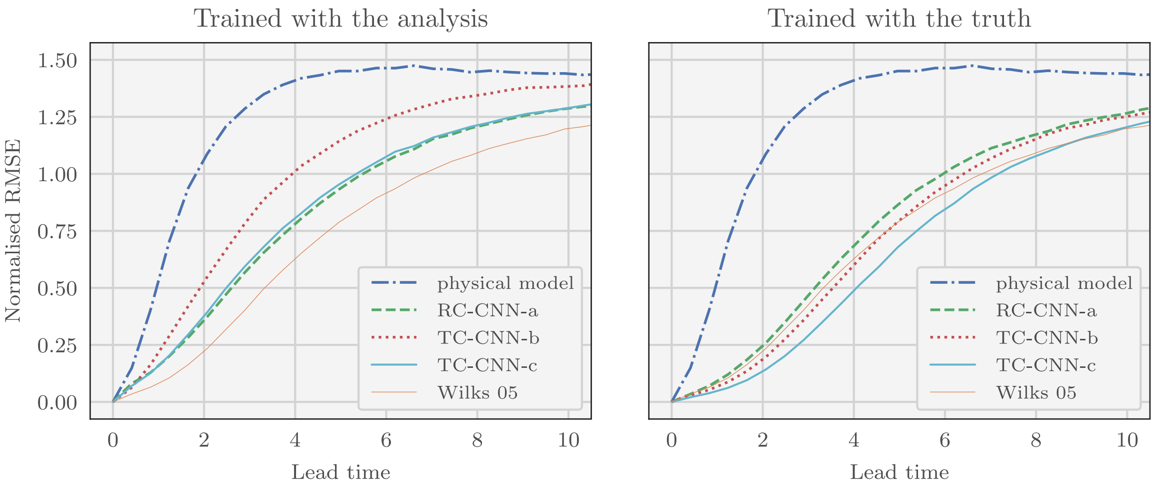

Fig. 5. Same as Fig. 2, left panel, for the physical model (in blue) and the trained surrogate models: RC-CNN-a (in green), TC-CNN-b (in red), and TC-CNN-c (in cyan). The surrogate models are trained either with the analysis (left panel) or with the truth (right panel). For comparison, the thin orange line indicates the scores for the model with the polynomial regression of Wilks [40]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where $\mathcal{F}^a$ is defined in Section 3.4.2 as the function encoding CNN-a. This assumption is the standard assumption in the current implementation of WC 4D-Var [43,44] and it is the assumption we make for our DA experiments. Furthermore, for the same reason as above, we evaluate the models which have been trained with the largest dataset.

Following the approach of Section 3.2, we study the evolution of the sRMSE as a function of the length of the DAW L. The results are shown in Fig. 6 and demonstrate that the improved accuracy of the models also yields more accurate analyses. From these results, two elements could be highlighted. First, the sRMSE is much lower with TC-CNN-b and TC-CNN-c, which both use the TC method, than with RC-CNN-a, which uses the RC method. Additionally, the sRMSE is lower when RC-CNN-a is trained with the analysis than with the truth, even though the model error predictions are more accurate, as indicated by the lower tMSE. These results confirm the hypothesis of Farchi et al. [26] that the main obstacle to more accurate analyses with the RC method is the assumption of linear growth of errors in time and that the TC method is more appropriate for DA experiments where the impact of nonlinearities is significant. Second, when trained with the truth, the sRMSE obtained with the TC method is of the same order as that obtained with the true model, it is even better for TC-CNN-c! For large windows, typically $L \geq 8$, a fraction of the improvement comes from the numerical issues with the true model discussed at the end of Section 3.2. For smaller windows however, we believe that the remaining improvement shows that TC-CNN-c may capture other deficiencies than model error.

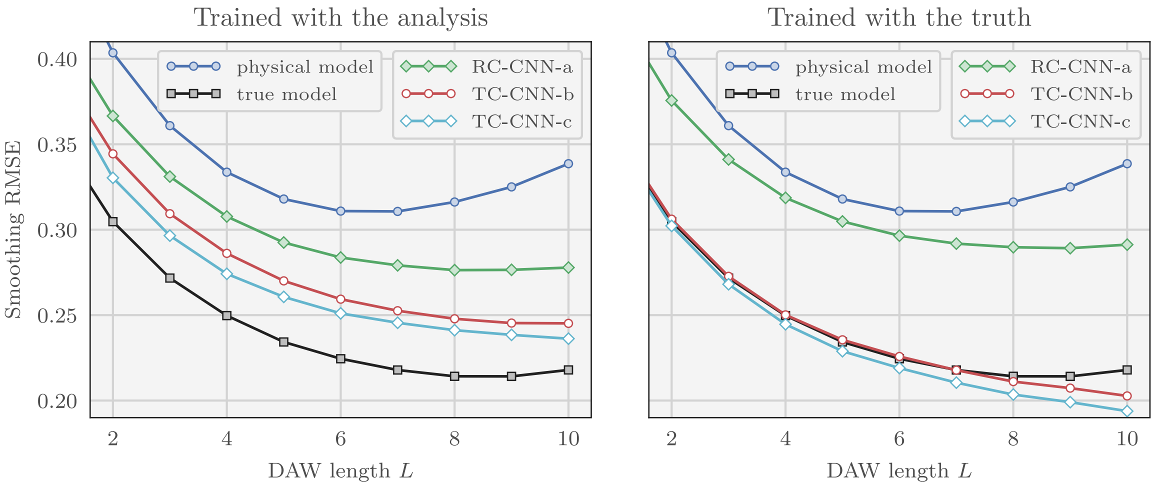

Fig. 6. Same as Fig. 2, right panel, for the physical model (in blue), the true model (in black), and the trained surrogate models: RC-CNN-a (in green), TC-CNN-b (in red), and TC-CNN-c (in cyan). The surrogate models are trained either with the analysis (left panel) or with the truth (right panel). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.4.7. *Learning from less frequent snapshots*

Because the true model is chaotic, forecasts are highly sensitive to the initial condition. In addition, the accuracy of forecasts at longer lead times is generally more sensitive to incorrect model parameters, while the accuracy of the training dataset is unchanged. This is beneficial when training the surrogate model with the analysis since it reduces the impact of the analysis error in the training dataset. On the other hand, longer forecast are inherently harder to predict. Therefore, we expect to see a trade-off between both effects when increasing the length of the forecasts.

By construction, the surrogate models are trained to emulate the dynamics over one DAW, Eq. (28). Changing the DAW length L is not optimal, because $L = 6$ minimises the sRMSE and hence it ensures the most accurate analysis on average. Another possibility is to keep $L = 6$ and train the surrogate models to emulate the dynamics over multiple DAWs, *i.e.* to reproduce the map

$$\mathbf{x}^{\mathrm{a}}_{kL} \mapsto \mathbf{x}^{\mathrm{a}}_{(k+N)L}, \tag{33}$$

where N is the number of DAWs over which the models should emulate the dynamics. This technique has been used by Farchi et al. [26] in the context of the RC method, with only partial success. Indeed, using $N > 1$ systematically worsen the DA scores, which is not a surprise because the assumption of linear growth of errors in time is less and less valid for longer forecasts. In this section, we evaluate this technique with the TC method, which does not suffer from the same limitations as the RC method (in particular it does not require additional assumption for DA experiments).

Fig. 7 shows the forecast skill and the DA score for TC-CNN-c trained with the analysis using $N = 2$. For comparison, the scores for TC-CNN-c trained with the analysis and with the truth, both using $N = 1$, are taken from Figs. 5 and 6 and reproduced here. These results confirm that increasing the length of the forecasts to predict yields more accurate surrogate models, both for forecast and DA experiments. Moreover, we have checked that the scores get worse when further increasing the length of the forecasts to $N = 4$ (not shown here). This indicates that the trade-off aforementioned reaches an optimum for $N = 2$ or $N = 3$.

3.4.8. *Iterating the DA–ML method*

Throughout the previous experiments, we have seen different performances depending on whether the surrogate models are trained with the analysis or with the truth, the latter case being equivalent to using dense and noiseless observations. Iterating the DA–ML method, as originally proposed by Brajard et al. [15] and formalised by Bocquet et al. [16] as a coordinate descent optimisation, is a way to bring the performances of the surrogate models trained with the analysis closer to that of the surrogate models trained with the truth.

Given the results of the DA experiment with the RC method in Section 3.4.6, it is possible that an additional iteration of the DA–ML method will not improve the DA score by much. This is not the case with the TC method because in this case, the analysis with the surrogate model is much more accurate than that of the physical (non-corrected) model. However, for the TC method we would like to show an alternative approach in the next section.

4. Online learning of model error with tendency correction

4.1. *From offline to online learning*

As already mentioned, the DA–ML method presented in Section 2 and illustrated in Section 3 is an offline learning method, meaning that the training starts only once all observations have been assimilated. This makes the method simple and flexible because the ML and DA steps are independent from each other. As a consequence, it is probably easier to extract all the information from the analysis during the ML step.

On the other hand, using an offline approach also has drawbacks. As suggested in Section 3.4.8, the DA–ML method needs to be iterated to give the best performance. At each iteration, the entire set of observations has to be re-assimilated, which can be problematic when the DA step is numerically expensive (for example with a realistic model). This is particularly concerning in the case of an operational model for which a new correction must be trained each time the model gets updated. Such issues does not affect online approaches, since the training could start as soon as the first observation arrives.

In our context, online learning means that, each time a new batch of observation becomes available, we have to estimate both the state of the system and the surrogate model at the same time. To address this problem, the most natural approach is to use the formalism of DA with an augmented state containing the current state of the system $\mathbf{x}$ and the parameters of the surrogate model $\mathbf{p}$, following the principle formulated by Jazwinski [45]. The resulting inference problem shares many aspects with classical parameter estimation in DA [29].

4.2. *A new formulation of weak-constraint 4D-Var*

Following the approach described in Section 3.2, the observations are assimilated using the 4D-Var algorithm, with consecutive windows

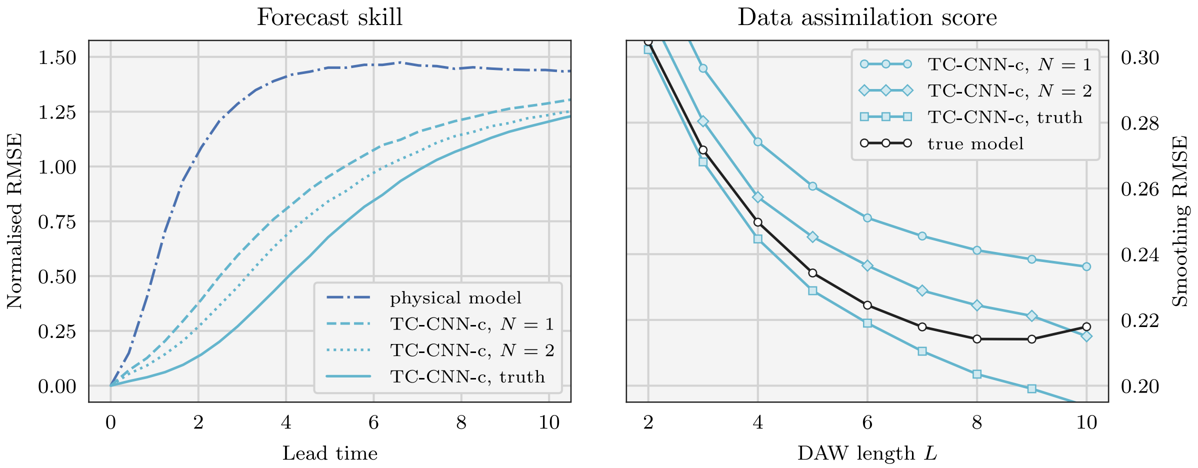

Fig. 7. Same as Fig. 2 for the physical model (in blue), the true model (in black), and the trained surrogate model TC-CNN-c (in cyan). The surrogate model is trained either with the analysis over one DAW ($N = 1$), over two DAWs ($N = 2$), or with the truth over one DAW ($N = 1$). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of L batches of observations. Since the algorithm now has to estimate the model parameters in addition to the model state, each 4D-Var problem consists in minimising the cost function

$$\mathcal{J}\left(\mathbf{p}_k,\mathbf{x}_k\right) = \frac{1}{2}\left\|\mathbf{p}_k-\mathbf{p}_k^{\mathrm{b}}\right\|^2_{\mathbf{B}_{\mathrm{p}}^{-1}} + \frac{1}{2}\left\|\mathbf{x}_k-\mathbf{x}_k^{\mathrm{b}}\right\|^2_{\mathbf{B}_{\mathrm{x}}^{-1}} + \frac{1}{2}\sum_{l=0}^{L-1}\left\|\mathbf{y}_{k+l}-\mathcal{H}\circ\mathcal{M}_{l\Delta t}\left(\mathbf{p}_k,\mathbf{x}_k\right)\right\|^2_{\mathbf{R}^{-1}}, \tag{34}$$

where $\mathbf{p}_k^{\mathrm{b}}$ and $\mathbf{B}_{\mathrm{p}}$ are the background and background error covariance matrix for model parameters, respectively. Comparing Eq. (34) to the 4D-Var cost function without model parameters, Eq. (23), two differences can be highlighted. First, the resolvent of the physical model $\mathcal{M}^{\mathrm{p}}_{l\Delta t}$ is replaced with the resolvent of the surrogate model $\mathcal{M}_{l\Delta t}$, which now depends on the model parameters. Second, a background (or prior) term on model parameters has been added. This background term is very important because, in this cycled context, it carries out the information on model parameters from one DAW to the next. Indeed, the background on model parameters for the next DAW is equal to the analysis on model parameters of the current DAW:

$$\mathbf{p}^{\mathrm{b}}_{k+L} = \mathbf{p}^{\mathrm{a}}_k. \tag{35}$$

In other words, the forecast model for model parameters is the persistence, which is consistent with the fact that the model is autonomous.

Like WC 4D-Var, Eq. (34) can be inferred from Bayes' rule, but Eq. (34) additionally neglects the cross-covariances between the background errors for model state and for model parameters. Including cross-covariances between model state and model parameters is possible but requires either a prior knowledge or the use of an ensemble to estimate them, which is beyond the scope of the present work.

Our assimilation method is summarised in Algorithm 1 in a cycled context. Rigorously speaking, this could be seen as a SC method because it includes only one state vector in the control variables. However with our method, by contrast with SC 4D-Var, the model can be updated during the analysis (when the model parameters change). Therefore, considering a broader definition of WC methods, Algorithm 1 can be seen as a specific formulation of WC 4D-Var.

Optionally, the model parameters can be pre-trained. For completeness, we would like to mention the similarities with the algorithms developed by Bocquet et al. [27]; Malartic et al. [46] which solve the same kind of problems but in a filtering context (*i.e.* with $L = 1$) with several variants of the ensemble Kalman filter. Finally note that, just as the DA–ML method of Section 2 can be used for model error correction instead of full model emulation, the present formulation of WC 4D-Var can be used for model error correction as well.

5. Numerical illustration with the two-scale Lorenz model (part II)

In this section, we illustrate the online learning method developed in Section 4 using the same model error setup as in Section 3. The true model is the L05III model and the physical model is the L96 model. However, because the online learning approach is based on the sole formalism of DA, we only use the surrogate models TC-CNN-b and TC-CNN-c, built with the TC method, since we have shown in Section 3.4.6 that it is the most consistent and the most efficient for DA experiments.

5.1. Data assimilation setup

For this illustration, we take the same DA problem as in Section 3.2. The truth is generated using the true model, and observations are taken every $\Delta t = 0.05$ from the slow variables only, using Eq. (22).

We start the experiments by assimilating the observations using the SC 4D-Var algorithm with the exact same setup as in Section 3.2. In particular, we use the physical model, and we choose $L = 6$ as it yields the best results with this model. After a total of 1024 DA cycles, which is enough to ensure the convergence of the analysis error statistics, we shift to the NN formulation of WC 4D-Var (Algorithm 1). In other words, we switch on model error correction. For the following cycles, we keep $L = 6$ and $\mathbf{B}_{\mathrm{x}} = b_{\mathrm{x}}^2\mathbf{I}$, although b_{x} can be different than in the 1024 preliminary cycles. In addition, we set $\mathbf{B}_{\mathrm{p}} = b_{\mathrm{p}}^2\mathbf{I}$, where b_{p} is another algorithmic parameter to specify. Finally for consistency, the initial background for model parameters $\mathbf{p}_0^{\mathrm{b}}$ is constructed using the NN initialisation method described in Section 3.4.3.

With WC 4D-Var, at each cycle the cost function $\mathcal{J}$, Eq. (34), is minimised using the same L-BFGS algorithm as for the SC 4D-Var. The gradient of $\mathcal{J}$, with respect to both model state and model parameters, is computed exactly using automatic differentiation, and the starting point of the minimisation is $\left(\mathbf{p}_k^{\mathrm{b}},\mathbf{x}_k^{\mathrm{b}}\right)$. The accuracy of the analysis is measured using the RMSE *on model state* (analysis minus truth) at the start of the DAW, which corresponds to the sRMSE defined in Section 3.2. In this cycled context, the sRMSE only improves if the model is getting more accurate. Nevertheless, we also measure the accuracy of the model by computing the tMSE defined in Section 3.4.4.

Finally note that the 1024 preliminary cycles with the physical model are not mandatory. We have checked that the results are qualitatively similar without them. In fact, adding these preliminary cycle is a way to get in real condition, where we need to correct a physical model which is already running since a while. In the following section, we do not discuss the preliminary cycles.

Algorithm 1 NN formulation of WC 4D-Var in a cycled context

```
Parameters: Observation operator $\mathcal{H}$, surrogate model $\mathcal{M}$ with NN, background error covariance matrices for model state and model parameters $\mathbf{B}_{\mathrm{x}}$ and $\mathbf{B}_{\mathrm{p}}$, observation error covariance matrix $\mathbf{R}$, DAW length $L$
Input: Initial background for model state and model parameters $\mathbf{x}_0^{\mathrm{b}}$ and $\mathbf{p}_0^{\mathrm{b}}$, observations $\left\{\mathbf{y}_0, \dots, \mathbf{y}_{(N_{\mathrm{t}}+1)L}\right\}$
for $k = 0$ to $N_{\mathrm{t}}$ do
    Compute the analysis $\mathbf{p}_{kL}^{\mathrm{a}}$ and $\mathbf{x}_{kL}^{\mathrm{a}}$ by minimising $\mathcal{J}\left(\mathbf{p}_{kL}, \mathbf{x}_{kL}\right)$, Eq. (34)    ▷ analysis at time $t_{kL}$
    Forecast the model parameters $\mathbf{p}_{(k+1)L}^{\mathrm{b}} = \mathbf{p}_{kL}^{\mathrm{a}}$    ▷ using persistence
    Forecast the model state $\mathbf{x}_{(k+1)L}^{\mathrm{b}} = \mathcal{M}_{L\Delta t}\left(\mathbf{p}_{kL}^{\mathrm{a}}, \mathbf{x}_{kL}^{\mathrm{a}}\right)$    ▷ using the surrogate model
end for
return $\left(\mathbf{x}_{kL}^{\mathrm{a}}, \mathbf{p}_{kL}^{\mathrm{a}}\right)$ for $k \in \left\{0, \dots, N_{\mathrm{t}}\right\}$    ▷ analysis trajectory
```

5.2. Results with TC-CNN-b as surrogate model

Let us start by applying the method to TC-CNN-b. In other words, in this case $\mathcal{M}_{l\Delta t}$ in Eq. (34) is the resolvent of TC-CNN-b for an integration of $l\Delta t$. For this experiment, we use the following algorithmic parameters, which we have empirically found to yield good performances:

$$b_{\mathrm{x}} = 0.28 + 0.15 \times \exp\left(-t/256\right), \tag{36a}$$

$$\widehat{b}_{\mathrm{p}} = 0.001 + 0.1 \times \exp\left(-t/1024\right), \tag{36b}$$

$$b_{\mathrm{p}} = \min\left[0.05, \widehat{b}_{\mathrm{p}}\right], \tag{36c}$$

where t is the time measured in number of DAWs after the 1024 preliminary cycles. The rationale behind this choice is that the optimal values of b_{x} and b_{p} should be larger at the start of the experiment than at the end (when the model is more accurate). To come up with Eq. (36), we first chose the shape of b_{x} and b_{p} (exponential decay) and we then tuned the coefficients until we got satisfying results. Also note that we have checked that the results are qualitatively similar when replacing the exponential decay of b_{x} and b_{p} with a linear decay.

Fig. 8 shows the time series of sRMSE and tMSE throughout the experiment, after the 1024 preliminary cycles. For comparison, the scores obtained when TC-CNN-b is trained using the offline DA–ML approach, both with the analysis and with the truth, are taken from Figs. 4 and 6 and reproduced here. First of all, the experiment is a success: the algorithm is working as expected and steadily improves the model, which can be seen both in the sRMSE (DA score) and in the tMSE (forecast score). After 128 to 256 cycles the model becomes more accurate than if trained offline with the analysis. Finally, the accuracy converges after 2048 to 4096 cycles. In the end, the model is almost as accurate as if trained offline with the truth! This is a strong result because, as mentioned in Section 3.4.4, training with the truth illustrates the full potential of a surrogate model. This shows that our online learning method has been able to extract all the information from the observations, and that we cannot expect better results with this surrogate model.

5.3. Results with TC-CNN-c as surrogate model

We now apply the method to TC-CNN-c. For this experiment, the algorithmic parameters, once again chosen on empirical grounds, are slightly different:

$$b_{\mathrm{x}} = 0.26 + 0.20 \times \exp\left(-t/256\right), \tag{37a}$$

$$\widehat{b}_{\mathrm{p}} = 0.01 + 0.05 \times \exp\left(-t/3072\right), \tag{37b}$$

$$b_{\mathrm{p}} = \min\left[0.05, \widehat{b}_{\mathrm{p}}\right], \tag{37c}$$

with t being once again the time measured in number of DAWs after the 1024 preliminary cycles.

Fig. 9 shows the time series of sRMSE and tMSE throughout the experiment, after the 1024 preliminary cycles. The success of the experiment is as clear as with TC-CNN-b: the algorithm steadily improves the model, which can be seen both in the sRMSE and in the tMSE. After 128 to 256 cycles the model becomes more accurate than if trained offline with the analysis. However, even though the total number of DA cycles increases, the accuracy has not yet converged at the end of the experiment. One must keep in mind that the correction provided by CNN-c is here nonlinear, contrary to the previous experiment with TC-CNN-b where the correction provided by CNN-b is linear. Therefore, we think that the present experiment is a good illustration of the increased complexity of training nonlinear NNs. Nevertheless, the accuracy of the model after 8192 is remarkable and in particular the sRMSE is lower than when using the true model! We also think that, if we were to extend the experiment with an appropriate tuning for b_{p}, the model would in the end be almost as accurate as if trained offline with the truth, just as in the previous experiment with TC-CNN-b.

5.4. Additional remarks on the online experiments

While preparing the online experiments, we have found that tuning the algorithmic parameter b_{p} is very important. If b_{p} is too small, the algorithm gives too much weight to the background on model parameters $\mathbf{p}_k^{\mathrm{b}}$. Even if this does not stop the learning process, it tapers the model parameter update, which makes the convergence slower[2]. On the other hand if b_{p} is too large, the algorithm overfits the model parameters to the observation window, which can yield divergence. As mentioned in Section 5.2, what makes the tuning of b_{p} really complex here is that, as the model steadily improves, the optimal value of b_{p} decrease. The values selected for the experiments, Eqs. (36) and (37), have been chosen by *trial and error*. Even though they yield good performances, they have not been optimally tuned. This means that a faster convergence could have been most likely obtained with other values. Finally, note that the algorithmic parameter b_{p} here is very similar to the tapering parameter introduced by Bocquet et al. [27]; Malartic et al. [46] in their variants of the ensemble Kalman filter.

The online experiments use the zero/random NN initialisation method described in Section 3.4.3. Therefore, at the start of the WC 4D-Var cycles the surrogate model is equivalent to the physical model. Other initialisation methods are possible. For example, one can use the set of parameters obtained after the offline learning method of Section 3.4. We have implemented this method (note illustrated here) and checked that, with an appropriate tuning of b_{p}, the final scores are the same than with the zero/random initialisation but the convergence is somewhat faster.

Finally, the online experiments use the same DAW length as with the physical model, $L = 6$. However, since the accuracy of the model increases during the experiment, increasing L is a reasonable option. We have performed the online experiments with $L = 10$ (not illustrated here). The evolution of the tMSE (forecast score) is very similar to that with $L = 6$, but, as expected from Fig. 6, the sRMSE (DA score) is close to 0.2, significantly lower than with $L = 6$.

[2] The convergence speed is measured here in number of DA cycles before convergence and not in wall-clock execution time.

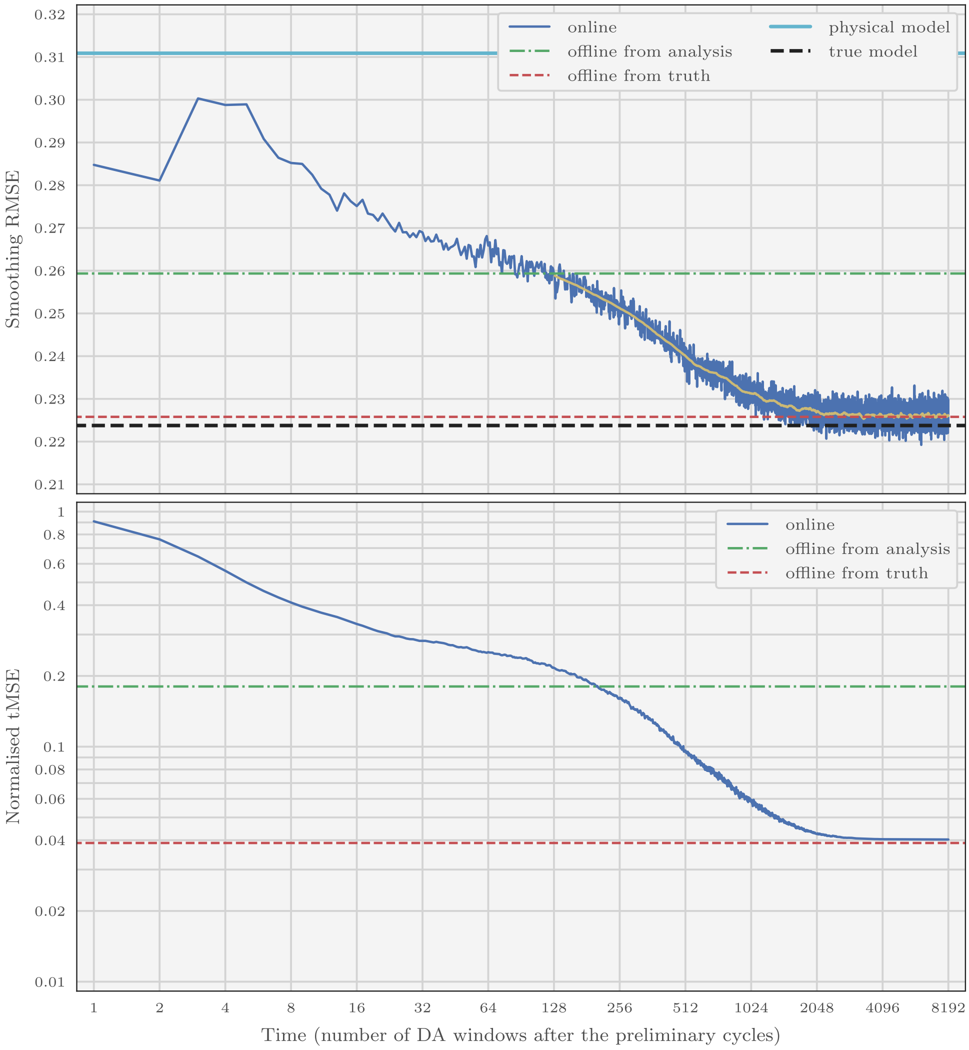

Fig. 8. Time series of sRMSE (top panel) and tMSE (bottom panel) for the online experiment with TC-CNN-b (in blue). The tMSE is computed using a test dataset of size $N_t = 1024$ and then normalised by the tMSE of the physical model. Both sRMSE and tMSE are averaged over 512 repetitions of the experiment. In addition, the yellow line shows the moving-average of the sRMSE over 128 DAWs. For comparison, the horizontal lines show the scores for the physical model (in cyan), the true model (in black), TC-CNN-b trained offline with the analysis (in green) and trained offline with the truth (in red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.5. Tendency correction in a realistic model

Both the offline and online learning experiments illustrate the efficiency of the TC method. Including a TC into a complex realistic model is not immediate, depending on the structure of the code, because the correction term must be added to the code computing the tendencies. In order to use the variational methods illustrated in the present work to train a TC, both offline and online, we must be able to compute the gradient of the resolvent of the total model (physical model with correction) with respect to the model state and to the model parameters. As we have shown in Section 2.3, these gradients depend on the gradients of the correction term, which can be easily obtained with a ML library, but also on the TL operator of the physical model. The present work, with a simple model, has been largely facilitated by the fact that the code of the physical model has been written entirely with a ML library, which implies that we have benefited from automatic differentiation. For realistic models, which are inherently more complex, it is essential to have efficient differentiation methods.

Beyond these technical aspects, the number of trainable model parameters is a potential source of concern. In the present work, we have tried to keep it as small as possible, and ended up with 113 parameters (for both CNN-b and CNN-c). In preliminary experiments, a much larger NN (a fully-connected NN with a total of $36 \times 64 + 64 + 64 \times 36 + 36 = 4708$ parameters) has been tested and we found similar performances, although with more training data. Even though 113 parameters (or even 4708) will most likely not be enough to correct a complex, realistic model, we have the hope that with smart ML models, we will be able to keep the number of trainable parameters under control [22].

5.6. Generalisation to non-autonomous dynamics

We conclude this test series by briefly discussing the possibility to extend the present work to non-autonomous dynamics. With an offline learning method, a non-autonomous dynamics can only be learnt if (i) the time-dependency of the model is parametrised (which implies that time should be among the set of predictors) and (ii) the training dataset is large enough to infer the parametrisation. With an online learning method, parametrising the time-dependency is also an option, but such parametrisation is not mandatory if the time evolution of the dynamics is slow. For example, the online experiments of Sections 5.2 and 5.3 would probably also work if the dynamics evolution is no faster than a few hundred DA cycles.

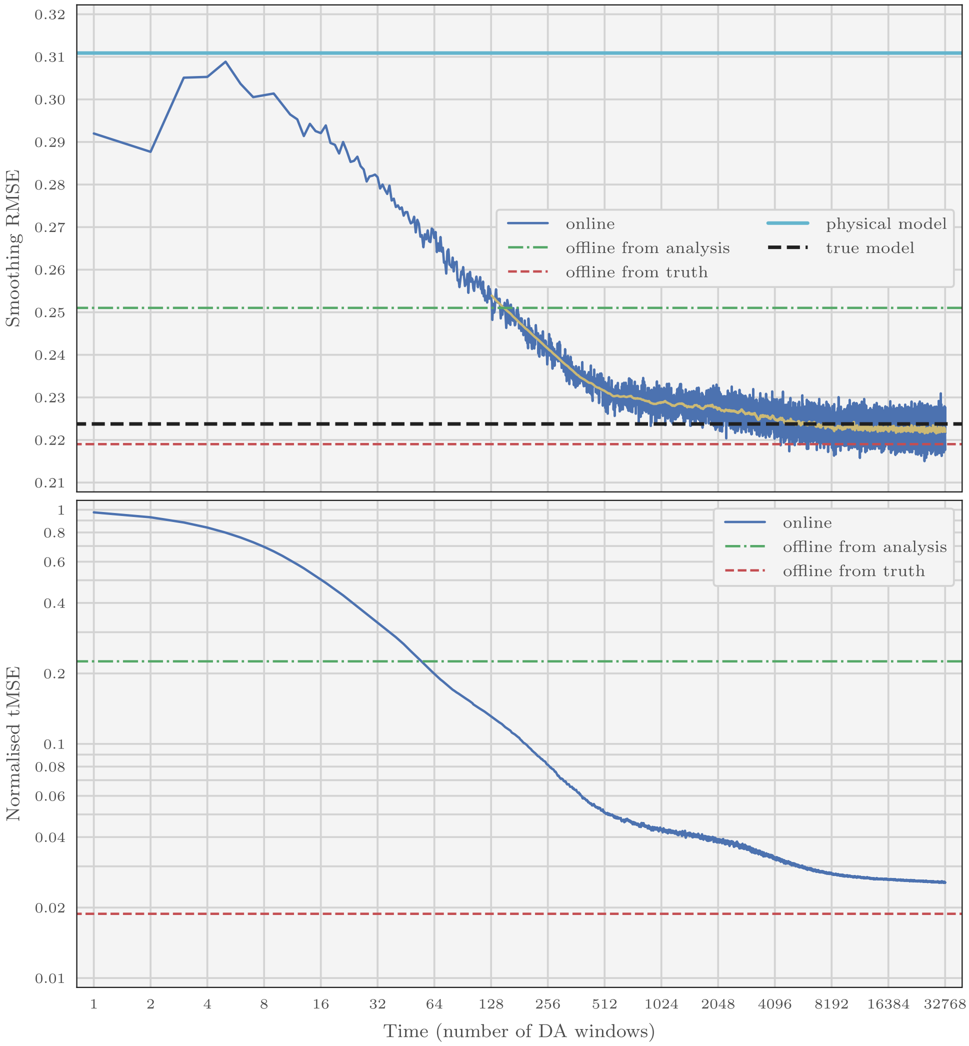

Fig. 9. Same as Fig. 8 for the online experiment with TC-CNN-c.

6. Conclusions

Combining DA and ML to emulate a dynamical model has been originally proposed by Brajard et al. [15]. The use of DA is essential here to assimilate sparse and noisy observations, which cannot be rigorously treated with ML methods alone. The same strategy can be used for model error correction instead of full model emulation. This has many advantages as it makes the inference problem easier [20,21]. In practice, a correction term can be included either in the model resolvent (*i.e.* as an integrated term between two forecast times) or directly in the model tendencies [14,26].

In the present article, we have compared the two methods. The first method, which we have called RC (resolvent correction), is easy to implement and has already been illustrated using low-order models by Brajard et al. [23]; Farchi et al. [26]. The second method, which we have called TC (tendency correction), is more technical to implement, in particular because it requires the adjoint of the physical model to correct, but it has the advantage of being more flexible, in particular for short-term forecasts. Both methods have been tested using the two-scale Lorenz system. In this case, model error primarily comes from unresolved small-scales processes (the fast variables). We have used noisy observations of the slow variables to build three different surrogate models. All three surrogate models are hybrid, with a physical part, the non-corrected model, and a statistical part, the correction term, built using simple NNs. The first surrogate model uses the RC method, while the other two use the TC method. The surrogate models are trained using the offline DA–ML method of Brajard et al. [15] and then evaluated in forecast and DA experiments. The results show that with the TC method, the surrogate models benefit from the interaction between the physical model and the NN, in such a way that it is possible to use much smaller NNs and much fewer training data to get similar results. The accuracy in forecast experiments is somewhat similar between the models using the RC and the TC method. By contrast, the models using the TC method significantly outperform the models using the RC method in DA experiments. This can be explained by the violation of the assumption of linear error growth in time, necessary with the RC method for the short-term forecasts within each DA experiment.

In the second part of this paper, we explored the possibility to train the surrogate models in an online fashion. With sparse and noisy observations, this means that we would have to learn both model state and model parameters at the same time. To address this problem, we introduce a new DA algorithm, which can be seen as a new formulation of WC 4D-Var. The algorithm has been implemented and tested using the same model error setup with the two-scale Lorenz system. The results show that online learning works as expected: the surrogate model is steadily improved, which can be seen both in the DA score and the forecast score; it quickly becomes more accurate than if trained offline with the analysis. At the end of the experiment, the model is almost as accurate as if trained offline with the truth. These results show that with online learning, it is possible to extract all the information from sparse and noisy observations.

Even though the results of the online experiments are promising, the NN formulation of WC 4D-Var derived in the present work could still benefit from methodological developments. In our experiments, we have found that tuning the algorithmic parameter b_p is a difficult but critical task. Ideally, the tuning method should be adaptive, with the objective of making the convergence faster and more accurate [27]. Furthermore, the method implicitly assumes that background errors for model state and model parameters are uncorrelated. Including cross-correlations between model state and model parameters is possible; it would be interesting to check whether this could make a difference in the accuracy of the analysis. More generally, the conclusions of the present work, and in particular the advantages of the TC method over the RC method, must be confirmed using higher-dimensional models.

CRediT authorship contribution statement

Alban Farchi: Conceptualisation, Formal analysis, Investigation, Methodology, Software, Validation, Visualisation, Writing – original draft, Writing – review editing. **Marc Bocquet:** Conceptualisation, Methodology, Supervision, Writing – review & editing. **Patrick Laloyaux:** Conceptualisation, Methodology, Supervision, Writing – review & editing. **Massimo Bonavita:** Conceptualisation, Methodology, Supervision, Writing – review & editing. **Quentin Malartic:** Conceptualisation, Methodology, Writing – review & editing.

Acknowledgements

The authors are grateful to an anonymous reviewer for insightful comments which helped improve the manuscript. CEREA is a member of Institut Pierre–Simon Laplace.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, http://dx.doi.org/10.1038/nature14539.

[2] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, in: Adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts, 2016.

[3] F. Chollet, Deep Learning with Python, Manning Publications Co, Shelter Island, New York, 2018.

[4] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937, http://dx.doi.org/10.1073/pnas.1517384113.

[5] F. Hamilton, T. Berry, T. Sauer, Ensemble Kalman filtering without a model, Phys. Rev. X 6 (1) (2016) 011021, http://dx.doi.org/10.1103/PhysRevX.6.011021.

[6] R. Lguensat, P. Tandeo, P. Ailliot, M. Pulido, R. Fablet, The analog data assimilation, Mon. Weather Rev. 145 (10) (2017) 4093–4107, http://dx.doi.org/10.1175/MWR-D-16-0441.1.

[7] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Phys. Rev. Lett. 120 (2) (2018) 024102, http://dx.doi.org/10.1103/PhysRevLett.120.024102.

[8] P.D. Dueben, P. Bauer, Challenges and design choices for global weather and climate models based on machine learning, Geosci. Model Dev. 11 (10) (2018) 3999–4009, http://dx.doi.org/10.5194/gmd-11-3999-2018.

[9] R. Fablet, S. Ouala, C. Herzet, Bilinear residual neural network for the identification and forecasting of geophysical dynamics, in: 2018 26th European Signal Processing Conference, EUSIPCO, IEEE, Rome, 2018, pp. 1477–1481, http://dx.doi.org/10.23919/EUSIPCO.2018.8553492.

[10] S. Scher, G. Messori, Generalization properties of feed-forward neural networks trained on Lorenz systems, Nonlinear Process. Geophys. 26 (4) (2019) 381–399, http://dx.doi.org/10.5194/npg-26-381-2019.

[11] J.A. Weyn, D.R. Durran, R. Caruana, Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data, J. Adv. Modelling Earth Syst. 11 (8) (2019) 2680–2693, http://dx.doi.org/10.1029/2019MS001705.

[12] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B.R. Hunt, E. Ott, A machine learning-based global atmospheric forecast model, Geophys. Res. Lett. 47 (9) (2020) http://dx.doi.org/10.1029/2020GL087776.

[13] H.D. Abarbanel, P.J. Rozdeba, S. Shirman, Machine learning: Deepest learning as statistical data assimilation problems, Neural Comput. 30 (8) (2018) 2025–2055, http://dx.doi.org/10.1162/neco_a_01094.

[14] M. Bocquet, J. Brajard, A. Carrassi, L. Bertino, Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models, Nonlinear Process. Geophys. 26 (3) (2019) 143–162, http://dx.doi.org/10.5194/npg-26-143-2019.

[15] 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, J. Comput. Sci. 44 (2020) 101171, http://dx.doi.org/10.1016/j.jocs.2020.101171.

[16] M. Bocquet, J. Brajard, A. Carrassi, L. Bertino, BayesIan inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization, Found. Data Sci. 2 (1) (2020) 55–80, http://dx.doi.org/10.3934/fods.2020004.

[17] R. Arcucci, J. Zhu, S. Hu, Y.-K. Guo, Deep data assimilation: Integrating deep learning with data assimilation, Appl. Sci. 11 (3) (2021) 1114, http://dx.doi.org/10.3390/app11031114.

[18] S. Rasp, M.S. Pritchard, P. Gentine, Deep learning to represent subgrid processes in climate models, Proc. Natl. Acad. Sci. 115 (39) (2018) 9684–9689, http://dx.doi.org/10.1073/pnas.1810286115.

[19] T. Bolton, L. Zanna, Applications of deep learning to ocean data inference and subgrid parameterization, J. Adv. Modelling Earth Syst. 11 (1) (2019) 376–399, http://dx.doi.org/10.1029/2018MS001472.

[20] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, V. Kumar, Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles, in: Proceedings of the 2019 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019, p. 558, 566, http://dx.doi.org/10.1137/1.9781611975673.

[21] P.A. Watson, Applying machine learning to improve simulations of a chaotic dynamical system using empirical error correction, J. Adv. Modelling Earth Syst. 11 (5) (2019) 1402–1417, http://dx.doi.org/10.1029/2018MS001597.

[22] M. Bonavita, P. Laloyaux, Machine learning for model error inference and correction, J. Adv. Modelling Earth Syst. 12 (12) (2020) http://dx.doi.org/10.1029/2020MS002232.

[23] J. Brajard, A. Carrassi, M. Bocquet, L. Bertino, Combining data assimilation and machine learning to infer unresolved scale parametrization, Phil. Trans. R. Soc. A 379 (2194) (2021) 20200086, http://dx.doi.org/10.1098/rsta.2020.0086.

[24] D.J. Gagne, H.M. Christensen, A.C. Subramanian, A.H. Monahan, Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz '96 model, J. Adv. Modelling Earth Syst. 12 (3) (2020) http://dx.doi.org/10.1029/2019MS001896.

[25] A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, E. Ott, Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems, Chaos 30 (5) (2020) 053111, http://dx.doi.org/10.1063/5.0005541.

[26] A. Farchi, P. Laloyaux, M. Bonavita, M. Bocquet, Using machine learning to correct model error in data assimilation and forecast applications, Q. J. R. Meteorol. Soc. 147 (739) (2021) 3067–3084, http://dx.doi.org/10.1002/qj.4116.

[27] M. Bocquet, A. Farchi, Q. Malartic, Online learning of both state and dynamics using ensemble Kalman filters, Found. Data Sci. (2639-8001_2019_0_24) (2020) http://dx.doi.org/10.3934/fods.2020015.

[28] G.A. Gottwald, S. Reich, Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation, Physica D 423 (2021) 132911, http://dx.doi.org/10.1016/j.physd.2021.132911.

[29] J.J. Ruiz, M. Pulido, T. Miyoshi, Estimating model parameters with ensemble-based data assimilation: A review, J. Meteorol. Soc. Jpn. II 91 (2) (2013) 79–99, http://dx.doi.org/10.2151/jmsj.2013-201.

[30] M. Pulido, P. Tandeo, M. Bocquet, A. Carrassi, M. Lucini, Stochastic parameterization identification using ensemble Kalman filtering combined with maximum likelihood methods, Tellus A: Dyn. Meteorol. Oceanogr. 70 (1) (2018) 1–17, http://dx.doi.org/10.1080/16000870.2018.1442099.

[31] E.N. Lorenz, Designing chaotic models, J. Atmos. Sci. 62 (5) (2005) 1574–1587, http://dx.doi.org/10.1175/JAS3430.1.

[32] W.W. Hsieh, B. Tang, Applying neural network models to prediction and data analysis in meteorology and oceanography, Bull. Am. Meteorol. Soc. 79 (9) (1998) 1855–1870, http://dx.doi.org/10.1175/1520-0477(1998)079<1855:ANNMTP>2.0.CO;2.

[33] Y. Trémolet, Accounting for an imperfect model in 4D-Var, Q. J. R. Meteorol. Soc. 132 (621) (2006) 2483–2504, http://dx.doi.org/10.1256/qj.05.224.

[34] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, A. Simmons, C. Soci, S. Abdalla, X. Abellan, G. Balsamo, P. Bechtold, G. Biavati, J. Bidlot, M. Bonavita, G. Chiara, P. Dahlgren, D. Dee, M. Diamantakis, R. Dragani, J. Flemming, R. Forbes, M. Fuentes, A. Geer, L. Haimberger, S. Healy, R.J. Hogan, E. Hólm, M. Janisková, S. Keeley, P. Laloyaux, P. Lopez, C. Lupu, G. Radnoti, P. Rosnay, I. Rozum, F. Vamborg, S. Villaume, J.-N. Thépaut, The ERA5 global reanalysis, Q. J. R. Meteorol. Soc. 146 (730) (2020) 1999–2049, http://dx.doi.org/10.1002/qj.3803.

[35] E.N. Lorenz, K.A. Emanuel, Optimal sites for supplementary weather observations: Simulation with a small model, J. Atmos. Sci. 55 (3) (1998) 399–414, http://dx.doi.org/10.1175/1520-0469(1998)055<0399:OSFSWO>2.0.CO;2.

[36] L. Mitchell, A. Carrassi, Accounting for model error due to unresolved scales within ensemble Kalman filtering, Q. J. R. Meteorol. Soc. 141 (689) (2015) 1417–1428, http://dx.doi.org/10.1002/qj.2451.

[37] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (5) (1995) 1190–1208, http://dx.doi.org/10.1137/0916069.

[38] C. Pires, R. Vautard, O. Talagrand, On extending the limits of variational assimilation in nonlinear chaotic systems, Tellus A 48 (1996) 96–121, http://dx.doi.org/10.1034/j.1600-0870.1996.00006.x.

[39] A. Fillion, M. Bocquet, S. Gratton, Quasi static ensemble variational data assimilation: a theoretical and numerical study with the iterative ensemble Kalman smoother, Nonlinear Processes Geophys. 25 (2018) 315–334, http://dx.doi.org/10.5194/npg-25-315-2018.

[40] D.S. Wilks, Effects of stochastic parametrizations in the Lorenz '96 system, Q. J. R. Meteorol. Soc. 131 (606) (2005) 389–407, http://dx.doi.org/10.1256/qj.04.03.

[41] T. Bachlechner, B.P. Majumder, H.H. Mao, G.W. Cottrell, J. McAuley, ReZero is all you need: Fast convergence at large depth, 2020, arXiv:2003.04887, arXiv:2003.04887 [Cs, Stat].

[42] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, San Diego, CA, USA, 2015.

[43] P. Laloyaux, M. Bonavita, M. Chrust, S. Gürol, Exploring the potential and limitations of weak-constraint 4D-Var, Q. J. R. Meteorol. Soc. 146 (733) (2020) 4067–4082, http://dx.doi.org/10.1002/qj.3891.

[44] P. Laloyaux, M. Bonavita, M. Dahoui, J. Farnan, S. Healy, E. Hólm, S. Lang, Towards an unbiased stratospheric analysis, Q. J. R. Meteorol. Soc. 146 (730) (2020) 2392–2409, http://dx.doi.org/10.1002/qj.3798.

[45] A.H. Jazwinski, Stochastic Processes and Filtering Theory, in: Mathematics in science and engineering, (64) Acad. Press, San Diego, 1970.

[46] Q. Malartic, A. Farchi, M. Bocquet, State, global and local parameter estimation using local ensemble Kalman filters: applications to online machine learning of chaotic dynamics, SIAM/ASA J. Uncertain. Quantif. (2021) submitted for publication.

Alban Farchi is a recently hired permanent researcher at CEREA. He works in the field of data assimilation for the geosciences with application to atmospheric chemistry. In 2020, he has been a visitor scientist at ECMWF, working on the use of machine learning techniques to correct model error in data assimilation and forecast applications.

Marc Bocquet is Professor at École des Ponts (France) and deputy director of CEREA laboratory. He works on the methods of data assimilation, environmental statistics and machine learning, with applications to dynamical systems, atmospheric chemistry and transport, and meteorology. He has published 107 peer-reviewed papers.

Patrick Laloyaux Trained as an applied mathematician, Patrick Laloyaux has expertise in optimisation methods for variational problems and in ensemble Kalman filters. He obtained his Ph.D. from the University of Namur (Belgium) conducted in partnership with the European Centre for Research and Advanced Training in Scientific Computation (CERFACS, France). During his Ph.D., he developed new preconditioning and gradient-free techniques for solving variational problems. He was also a teaching assistant giving tutorials (220 hours/year) to undergraduate and graduate students in programming, scientific computation and numerical optimisation.

Massimo Bonavita has a background in Physics. He has worked for 15 years in Italian Weather Service, first as a forecaster and then in data assimilation, where he developed ensemble and variational analysis systems for local area model initialisation. He joined ECMWF in March 2009 to work on the development of the variational data assimilation system and its ensemble extension. Since February 2016 Massimo leads the Data Assimilation Methodology Team at ECMWF, which works on the continuous improvement of the data assimilation algorithms used for the initialisation of ECMWF forecasts.

Quentin Malartic is a Ph.D. student at École des Ponts ParisTech and École Normale Supérieure. His research interest is data assimilation and machine learning in the context of chaotic dynamics. He holds a master's degree in both geosciences and civil engineering from Université Paris Saclay.