How to Calibrate a Dynamical System With Neural Network Based Physics?

Blanka Balogh, D Saint-martin, Aurélien Ribes

Geophysical Research Letters

RESEARCH LETTER
10.1029/2022GL097872

Key Points:

- Tunable parameters are included to the inputs of a neural network (NN) parameterization
- The tunable NN parameters are optimized by using a kriging method
- Long-term statistical properties of the NN-based model are tuned without any new learning procedure

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:

B. Balogh,
blanka.balogh@meteo.fr

Citation:

Balogh, B., Saint-Martin, D., & Ribes, A. (2022). How to calibrate a dynamical system with neural network based physics? *Geophysical Research Letters*, *49*, e2022GL097872. https://doi.org/10.1029/2022GL097872

Received 14 JAN 2022
Accepted 26 MAR 2022

How to Calibrate a Dynamical System With Neural Network Based Physics?

B. Balogh[1], D. Saint-Martin[1], and A. Ribes[1]

[1]CNRM, Université de Toulouse Météo-France, CNRS, Toulouse, France

Abstract Unlike the traditional subgrid scale parameterizations used in climate models, current machine learning (ML) parameterizations are only tuned *offline*, by minimizing a loss function on outputs from high-resolution models. This approach often leads to numerical instabilities and long-term biases. Here, we propose a method to design tunable ML parameterizations and calibrate them *online*. The calibration of the ML parameterization is achieved in two steps. First, some model parameters are included within the ML model input. This ML model is fitted at once for a range of values of the parameters, using an *offline* metric. Second, once the ML parameterization has been plugged into the climate model, the parameters included among the ML inputs are optimized with respect to an *online* metric quantifying errors on long-term statistics. We illustrate our method with two simple dynamical systems. Our approach significantly reduces long-term biases of the ML model.

Plain Language Summary In numerical climate models, processes occurring at scales smaller than the model resolution (e.g., convection, turbulence) need to be represented by "parameterizations." Parameterizations provide a simplified yet numerically affordable version of the modeled processes. Recently, parameterizations are also developed using machine learning (ML) by fitting to outputs from high resolution climate models. This method can lead to long-term biases when incorporating the ML parameterizations into the climate model. And, there is no possibility in the current approach to calibrate the ML parameterization to alleviate these biases. We propose here an innovative approach to calibrate ML parameterizations once they have been fitted to a learning sample. Our approach has been successfully tested on two toy models. A first set of experiments focus on the retrieval of the value of parameters used to generate a reference data set. In the second experiment, the value of some parameters not included in the neural network (NN) has been biased, resulting in errors in long-term statistics. Finding the optimal value of the NN input parameter has significantly improved the accuracy of the resulting model. Our method could be applied to improve the prediction of long-term variables in climate models.

1. Introduction

The ordinary differential equations describing a climate model, can be written

$$\dot{\mathbf{x}} = \mathcal{D}(\mathbf{x}) + \phi(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the state variable of the climate model. $\mathcal{D}$ corresponds to the discretization of the primitive equations on the model grid, whereas ϕ accounts for the subgrid scale atmospheric processes (e.g., turbulence, convection). Currently, the best compromise for representing subgrid scale processes are the "physical parameterizations," emulating a numerically affordable version of ϕ. These parameterizations are parametric functions f_ϕ. They are built heuristically on theoretical knowledge and outputs from high-resolution simulations (e.g., Jakob, 2010). In practice, f_ϕ is the sum of individually modeled processes, the parameters of which are separately estimated. The numerical efficiency of f_ϕ comes at the price of approximations. Hence, parameterizations of the atmosphere are the main source of uncertainties in climate models (e.g., Medeiros & Stevens, 2011; Medeiros et al., 2008; Stevens & Bony, 2013). Recently, interest has grown in using machine learning (ML) to develop parameterizations, noted $\widehat{f}$, promising more precise yet affordable parameterizations for climate models (e.g., Gentine et al., 2018; Rasp et al., 2018; Yuval & O'Gorman, 2020).

The development of both physical and ML-based parameterizations first involves defining a set of possible functions for the estimate of ϕ (this set is typically much larger in the ML case). Then, model parameters are optimized with respect to an *offline* metric ℓ (also called "loss function" in the ML case). This *offline* metric

evaluates the error between the estimate function (i.e., f_ϕ or $\hat{f}$) and a target function f, the best accurate approximation of ϕ—as even high-resolution model data is only an approximation of the ground truth subgrid-scale processes. This approach allows us to fit the models on a *point by point* basis. However, a climate model's performance lies in accurate predictions of long-term statistics, measured by *online* metrics. Physical parameterizations typically require an additional calibration step, once it has been plugged-in to the dynamical model. In contrast, there is usually no tunable parameter included in ML parameterizations. Nevertheless, tuning is very important for the development of climate models, to ensure model stability, to calibrate the value of long-term statistics and to reduce error due to potentially missed interactions between the dynamics and the physics. This *online* calibration can be achieved regarding an *online* metric, m. Without *online* calibration, even though a parameterization is efficient regarding the *offline* metric ℓ, there is no guarantee that the climate model will be accurate regarding the *online* evaluation metric (Brenowitz, Henn, et al., 2020).

The development of current ML parameterizations is still hampered by issues addressed by *online* calibration in the case of physical parameterizations, such as numerical instabilities (e.g., Brenowitz, Beucler, et al., 2020; Rasp, 2020). These issues are currently handled by enforcing physical conservation laws (Beucler et al., 2021) or using ML to replace part of a physical parameterization (Yuval et al., 2021). Additionally, the target function f can also be imperfect. In this case, even though the fit of the ML model is excellent, errors in the imperfect training data set will also be learned and result in a high *online* error m, also called long-term biases. This is a well known issue with physical parameterizations. The "art of tuning" model parameters consists in finding a compromise between *offline* and *online* metrics, going back and forth in optimizing the parameters with respect to ℓ and m (Couvreux et al., 2021; Hourdin et al., 2017; Schmidt et al., 2017).

To resolve these issues, we propose to include some tunable parameters into the ML parameterization inputs. This allows the calibration of new ML-based parameterization with respect to an *online* metric in an efficient way. Our approach to find the best value of tunable parameters is less empirical than the methods used to tune physical parameterizations. It relies on the minimization of the *online* metric m on long-term statistics, akin to the methodology described in Schneider et al. (2017) and in Cleary et al. (2021). The method will be demonstrated using Lorenz'63 and Lorenz'96 toy models (Lorenz, 1963, 1996). These toy models offer a simple framework to perform proof-of-concept atmospheric modeling experiments using ML tools (e.g., Chattopadhyay et al., 2020; Scher & Messori, 2019). Lorenz'96 model has also the advantage of implementing a "parameterization." The large scale variable is indeed interacting with the non-linear subgrid-scale variable.

The paper is organized as follows. Methodology is described in Section 2. Section 3 demonstrates our calibration method on two basic examples using Lorenz'63 and Lorenz'96 models. Conclusions are drawn in Section 4, discussing more broadly the results we have obtained.

2. Methodology

2.1. Step 1: Building a Tunable Neural Network Parameterization

The first step toward designing and optimizing tunable neural network (NN) parameterizations is to take into consideration some uncertain but tunable parameters, when fitting the NN. The target function f is often fitted to outputs from high-resolution simulations. This target function depends on uncertain parameters, noted $\boldsymbol{\theta} \in \mathbb{R}^p$, used to generate the high-resolution simulations: $f \equiv f(\boldsymbol{x}; \boldsymbol{\theta})$. In traditional approaches, the value of model parameters are fixed to the "best estimate" value of these parameters, $\boldsymbol{\theta}_0$. Thus, $f \equiv f(\boldsymbol{x}; \boldsymbol{\theta}_0)$ does not depend on any parameter and $\boldsymbol{\theta}$ is not included in the NN model input (e.g., Gentine et al., 2018; Yuval & O'Gorman, 2020). The novelty of our approach is to keep part of the uncertain parameters $\boldsymbol{\theta}$ among the input variables of the NN, and thus to retain not only the dependency of f from $\boldsymbol{x}$ but also from some model parameters $\boldsymbol{\theta}$.

The NN learning sample, of size N, is $\{(\mathbf{x}, \boldsymbol{\theta})_i, f(\mathbf{x}_i; \boldsymbol{\theta}_i)\}_{1 \le i \le N}$. The NN is fitted by optimizing an *offline* metric ℓ, also called "loss function." Typically, this loss function is Mean Squared Error (MSE), computed for each predicted time step individually:

$$\ell\left(f(\mathbf{x}; \boldsymbol{\theta}), \hat{f}(\mathbf{x}; \boldsymbol{\theta})\right) = \frac{1}{N} \sum_{i=1}^{N} \left\| f(\mathbf{x}_i; \boldsymbol{\theta}_i) - \hat{f}(\mathbf{x}_i; \boldsymbol{\theta}_i) \right\|^2. \quad (2)$$

The parameterization obtained after training the NN is noted, $\widehat{f}(\mathbf{x};\boldsymbol{\theta})$. It can be incorporated into the dynamical model so as to replace $\boldsymbol{\phi}$ in Equation 1. The resulting dynamical system can be used to generate a validation time series of the dynamical model, noted $[\mathbf{x}]^{\hat{f}}$.

2.2. Step 2: Optimizing the Tunable Neural Network Parameterization

The goal of this second step is to tune the optimal value of the parameters $\boldsymbol{\theta}$. Whereas the NN model was trained *offline* on a learning sample, calibration relies on an *online* validation metric, m, given by:

$$m(\boldsymbol{\theta}) = \frac{1}{M}\sum_{k=1}^{M} \left\|\mathcal{M}_k^{\text{ref}} - \mathcal{M}_k(\boldsymbol{\theta})\right\|^2, \tag{3}$$

where $\mathcal{M}$ are a set of M long-term statistics computed over $[\mathbf{x}]^{\hat{f}}$, and $\mathcal{M}^{\text{ref}}$ a set of reference statistics. The long-term statistics involved in the computation of m are typically the average and standard deviation values estimated over time series. The minimal value of m is reached at the optimal value of $\boldsymbol{\theta}$, noted $\boldsymbol{\theta}^*$.

Theoretically, the statistics $\mathcal{M}$ are computed over a time series of infinite length. In reality, we only compute an estimate of the long-term metric over a finite length simulation. The length of these simulation is chosen so that the *online* metric m no longer depends on the $\boldsymbol{x}$ initial condition. To simplify notations, the estimate of the long-term statistics over validation time series $[\mathbf{x}]$ is noted, $\widehat{\mathcal{M}}$. Even though this time series is sufficiently long, the resulting metric $\widehat{m}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{k=1}^{M}\left\|\mathcal{M}_k^{\text{ref}} - \widehat{\mathcal{M}}_k(\boldsymbol{\theta})\right\|^2$ will be noised, which can lead the optimizer to a local minimum of $\widehat{m}$, instead of the global minimum. To address this issue, minimization will be performed over a smoothed version of $\widehat{m}$, obtained by kriging (or Gaussian Process Regression, Cressie, 1992). The kriging metamodel will be fitted to a sample of $\left\{\boldsymbol{\theta}_i, \widehat{m}(\boldsymbol{\theta}_i)\right\}$. The obtained kriging metamodel is noted $\tilde{m}$. The optimal value of model parameters, $\boldsymbol{\theta}^*$, is obtained when $\tilde{m}$ reaches its (absolute) minimum value.

In summary, we train a NN parameterization depending on some tunable parameters $\boldsymbol{\theta}$, using an *offline* metric, ℓ. Since the main purpose is to reduce long-term prediction errors of the dynamical model, the value of $\boldsymbol{\theta}$ is subsequently optimized regarding an *online* metric, m. In practice, optimization is done over a kriging metamodel emulating m as a function of $\boldsymbol{\theta}$. In the following, our method will be demonstrated using the Lorenz'63 (hereafter L63, Lorenz, 1963) and the Lorenz'96 (hereafter L96, Lorenz, 1996) models.

2.3. The Lorenz'63 Model

The L63 system consists of a set of ordinary differential equations that can be expressed:

$$\begin{aligned}
\dot{x}_1 &= \sigma\,(x_2 - x_1)\,,\\
\dot{x}_2 &= x_1\,(\rho - x_3) - x_2,\\
\dot{x}_3 &= x_1 x_2 - \beta x_3.
\end{aligned} \tag{4}$$

Temporal evolution of the L63 state variable, $\boldsymbol{x} = (x_1, x_2, x_3)$, is governed by Equation 4, which involves a set of three model parameters, (σ, ρ, β). Equation 4 admits a chaotic solution in the vicinity of $(\sigma_0, \rho_0, \beta_0) = (10, 28, 8/3)$.

The *online* minimization metric m (Equation 3) depends strongly on the identification of variables quantifying the long-term statistical behavior of the L63 system. The first candidate of such a long-term variable is the mean value for each one of the three L63 state variables. However, the average value of x_1 and x_2 is independent of L63 parameters (see also Figure S1 in Supporting Information S1). Thus, only the average value of x_3, noted $\mu^{(3)}$, can be tuned. This quantity is useful to retrieve the optimal value of at most one L63 parameter. In addition to $\mu^{(3)}$, standard deviations over the time series can also be computed. Standard deviations will be noted $\sigma^{(i)}$ for each one of the L63 parameters, with $i \in \{1, 2, 3\}$. However, as highlighted by Figure S1 in Supporting Information S1, a strong correlation appears between the three standard deviations. Hence, it is possible to retrieve the optimal value of at most two L63 parameters by optimization. Thus, the following L63 idealized case study will use $\mathcal{M} = \left(\mu^{(3)}, \sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)}\right)$.

2.4. The Lorenz'96 Model

The 2-level L96 dynamical system depending on parameters (h, c, F, b) is given by:

$$\frac{\mathrm{d}x_k}{\mathrm{d}t} = -x_{k-1}\left(x_{k-2} - x_{k+1}\right) - x_k + F - \frac{hc}{b}\sum_{j=1}^{J} y_{k,j}, \tag{5}$$

$$\frac{1}{c}\frac{\mathrm{d}y_{k,j}}{\mathrm{d}t} = -by_{k,j+1}\left(y_{k,j+2} - y_{k,j-1}\right) - y_{k,j} + \frac{hc}{b}x_k, \tag{6}$$

where $\{x_k\}_{1\le k\le K}$ are K large-scale variables coupled to J small-scale variables $\left\{y_{k_0,j}\right\}_{1\le j\le J}$ for each $1 \le k_0 \le K$. The coupling term $B_k = -\frac{hc}{b}\sum_{k=1}^{K} y_{k,j}$ can be seen as a "subgrid-scale parameterization." Thus, Equation 5 can be analogous to a (very) simple climate model, as described by Equation 1. In this comparison, $\mathcal{D}(x_k) = -x_{k-1}\left(x_{k-2} - x_{k+1}\right) - x_k + F$ and $\phi(x_k) = B_k$. B_k can be approximated as a function of the large-scale state variables, x_k, and is often modeled with polynoms (e.g., Arnold et al., 2013). This subgrid-scale parameterization will be the target variable of the NN model. Given the symmetry of the L96 model, it is common to fit the NN to predict B_k as a function of data from only one spatial variable (e.g., Gagne et al., 2020; Rasp, 2020; Watson, 2019). In our case, the NN learns to predict a single B_k as a function of (x_k, c). This implies that the NN input is of size 2 and the output of size 1. To simplify notations, in the following, the NN will be noted, $\widehat{B}(x, c)$.

As in the L63 case, the mean value of the state variables does not depend on input parameters (h, c, F, b), and is therefore not used in the *online* metric m. Thus, the metric is based on standard deviations only, estimated on time series obtained with the NN parameterization.

3. Case Studies

3.1. Perfect Model Calibration

3.1.1. The Lorenz'63 Model

The objective is to fit an NN model, noted $\widehat{f}$, to approximate the L63 time derivative as a function of the state variable $\boldsymbol{x}$ and parameters $\boldsymbol{\theta} = (\rho, \beta)$:

$$\dot{\widehat{\mathbf{x}}} = \widehat{f}(\mathbf{x}, \boldsymbol{\theta}). \tag{7}$$

The learning sample is generated by an optimal sampling method, used to select relevant $(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}^3 \times \mathbb{R}^2$ values to build the NN learning sample. Latin Hypercube Sampling (hereafter LHS, McKay, 1992) is such a sampling method. The interest of using a specific sampling method to train stable and accurate NN based parameterization has been shown in Balogh et al. (2021). The boundaries of the LHS are set around plausible values for both $\boldsymbol{x}$ and $\boldsymbol{\theta}$. The resulting learning sample of size N_{LHS} is $[(\mathbf{x}, \boldsymbol{\theta})]^{\mathrm{LHS}} = \{(\mathbf{x}, \beta)_i, f\left(\mathbf{x}_i; \boldsymbol{\theta}_i\right)\}_{1\le i\le N_{\mathrm{LHS}}}$.

The target variable $\dot{\mathbf{x}}$ is computed with L63 model equations, parameterized with $\sigma = \sigma_0$. The learning sample is of size $N_{\mathrm{LHS}} = 10^7$. Values for $\boldsymbol{\theta} = (\rho, \beta)$ are sampled in $[26.5, 32] \times [1.5, 3.2]$. The NN model consists of $n_l = 7$ hidden layers of type "Dense." More specific details about the NN architecture are available in the Table S1 in Supporting Information S1. R^2 score over an independent subset of 20% of the learning sample is monitored during training. The best weights regarding the R^2 score are loaded after 30 epochs. The final model has $R^2 = 1.00$, which is not surprising given the low complexity of L63 model, and is consistent with case studies focusing on emulating L63 dynamics with NNs (Rasp, 2020).

The fitted NN model is then used to generate time derivatives, which are integrated with a Runge-Kutta 4 time stepping scheme with a temporal increment $\Delta t = 0.05$. The resulting validation time series or "orbit" $[\mathbf{x}]^{\widehat{f}}$ is of length 1000 Model Time Units (hereafter, MTU, where 1 MTU $= 20\Delta t$) not including a spin-up of 200 MTU. The validation orbits are used to compute long-term metrics and thus to assess the *online* performance of the model, measured by $\widehat{m}$:

$$\widehat{m}(\boldsymbol{\theta}) = \left(\widehat{\mu}^{(3)}(\boldsymbol{\theta}) - \mu_{\mathrm{ref}}^{(3)}\right)^2 + \sum_{n=1}^{3}\left(\widehat{\sigma}^{(n)}(\boldsymbol{\theta}) - \sigma_{\mathrm{ref}}^{(n)}\right)^2, \tag{8}$$

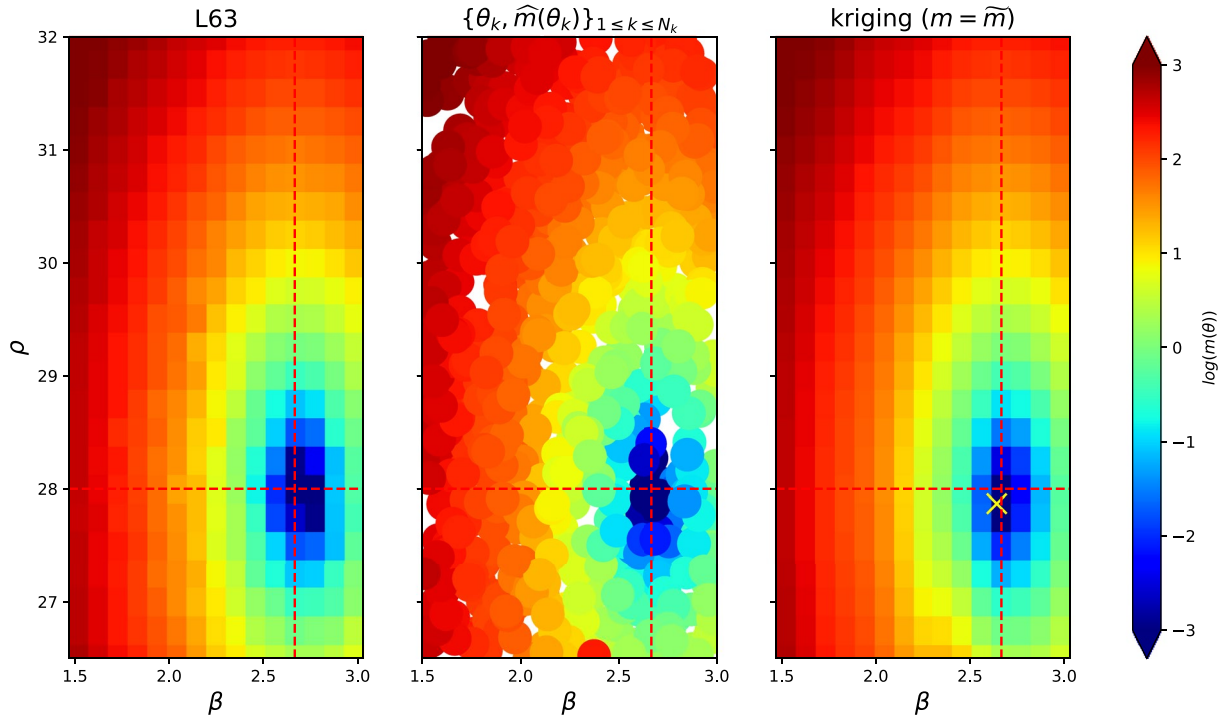

Figure 1. (left) The *online* metric m is computed over orbits of length 1000 Model Time Units (MTUs) generated with real L63 equations (Equation 4) for $\sigma = \sigma_0$ and for different regularly-spaced values of $\boldsymbol{\theta} = (\rho, \beta)$. (middle) Values of $\widehat{m}$ in the kriging learning sample, that is, $\left\{\boldsymbol{\theta}_i, \widehat{m}(\boldsymbol{\theta}_i)\right\}_{1 \le i \le N_k}$. The long-term statistics, $\widehat{m}(\boldsymbol{\theta}_i)$ are computed on validation orbits $[\mathbf{x}]^{\hat{f}}$ of length 1000 MTUs. (right) The fitted kriging metamodel $\tilde{m}$ used to find the optimal value of the tunable parameters, $\boldsymbol{\theta}^*$ (yellow cross). The optimal value of parameters is $\boldsymbol{\theta}^* = (27.9, 2.64)$, which is close to the values used to generate the reference data set: $(\rho, \sigma) = (\sigma_0, \rho_0)$. Color shades represent the logarithm of the *online* evaluation metric m (Equation 8) or its estimates.

where $\mu^{(3)}_{\text{ref}}$ and $\boldsymbol{\sigma}_{\text{ref}}$ (resp. $\widehat{\mu}^{(3)}(\beta)$ and $\widehat{\boldsymbol{\sigma}}(\beta)$) are parameters computed over the reference orbit (resp. validation orbit $[\mathbf{x}]^{\hat{f}}$). In this example, the "reference" data set consist in a long time-series of length 3000 MTU (considered as "infinite" lenght), generated by integrating L63 equations (Equation 4) with parameters $(\sigma_0, \rho_0, \beta_0)$.

A kriging metamodel learns to approximate $\widehat{m}$ as a function of $\boldsymbol{\theta}$, over a learning sample of size $N_k = 750$. To build the kriging learning data set, $\boldsymbol{\theta}$ values are sampled in the interval $[26.5, 32] \times [1.5, 3]$. The sampling interval for $\boldsymbol{\theta}$ has been slightly reduced compared with those used to generate the NN learning sample to avoid potential out-of-sample issues. In practice, we use a LHS to generate a sample of $(\boldsymbol{x}_0, \boldsymbol{\theta})$ to compute the validation orbits of length 1000 MTU on which the long-term metrics are estimated. Including the initial state variable $\boldsymbol{x}_0$ in the LHS sample reduces the noise related to the (finite) length of the validation orbits from the kriging learning sample. For the reasons explained in the Methodology section, dependency of $\widehat{m}$ on $\boldsymbol{x}_0$ is ignored.

The kriging model $\tilde{m}$ is fitted to $\left\{\boldsymbol{\theta}_i, \widehat{m}(\boldsymbol{\theta}_i)\right\}_{1 \le i \le N_k}$. The minimum of $\tilde{m}$ is retrieved with BFGS optimizer (Fletcher, 2013). The optimal value found for the tunable parameters is $\boldsymbol{\theta}^* = (27.9, 2.64)$ (Figure 1). The reference orbit was generated using $(\rho_0, \beta_0) = (28, 8/3)$: $\boldsymbol{\theta}^*$ is close to the values of the parameters used to generate the reference orbit.

3.1.2. The Lorenz'96 Model

The NN model is trained to predict $B = -\frac{hc}{b} \sum_{j=1}^{J} y_j$ as a function of x and $\boldsymbol{\theta} = c$. The learning sample is built by sequentially integrating the L96 equations (see Equations 5 and 6, with $K = 8$ and $J = 32$), using a fourth order Runge-Kutta time stepping scheme with an increment $\Delta t = 0.005$. The length of training integrations is 3,5 MTU (where 1 MTU = 200 Δt), not including 1,5 MTU of model spin-up. We perform $N_i = 500$ integrations, the initial conditions and $\boldsymbol{\theta}$ values $(\boldsymbol{x}, \boldsymbol{y}, c)$ of which are sampled using LHS, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$ and $\boldsymbol{y} = (y_{1,1}, y_{1,2}, \ldots, y_{K,J})$. The sampling interval for c values is $[6, 14]$.

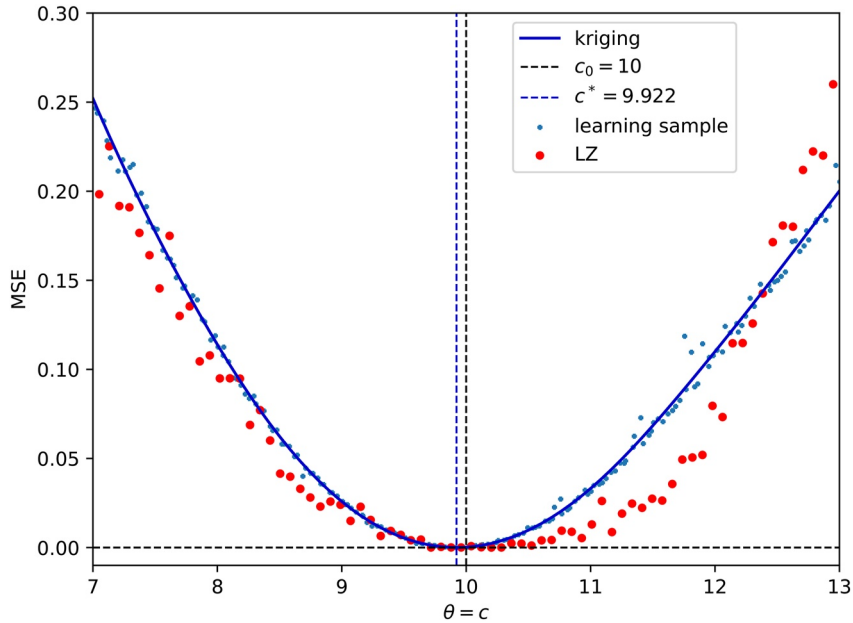

Figure 2. A Latin Hypercube Sampling (LHS) sample of $(\boldsymbol{x}, b)$ values, $\{\mathbf{x}_i, c_i\}_{1\leq i\leq N_l}$, is generated using LHS with c sampled in [7, 13]. The corresponding values of the long-term metrics, $\widehat{m}(c_i)$, $1 \leq c_i \leq N_l$, are computed on $[\mathbf{x}]^{\widehat{B}}$ (length: 3,5 Model Time Units (MTU)) from the LHS initial conditions. The resulting sample $\left\{c_i, \widehat{m}(c_i)\right\}_{1\leq i\leq N_k}$ (light blue scatter points) is used to train the kriging model (solid blue line). For comparison, red dots represent the metric m computed over time series (length: 15 MTU) obtained by applying L96 equations (Equation 5), for discrete values of c. The fitted kriging metamodel, $\tilde{m}$ (solid blue line), is a smoothed version of the metric computed on time series generated with the neural network parameterization. The optimal value of c, c*, is computed by minimizing $\tilde{m}$. $c^* = 9.922$ (dashed blue line) can be compared with $c_0 = 10$ (dashed black line), which was used to generate the reference orbit.

The NN is two layers deep, and has 32 nodes on each. More specific details about the NN architecture are available in Table S2 in Supporting Information S1. The model is trained over 30 epochs using the MSE loss function (Equation 2). The validation data set is made of 15% randomly chosen samples from the learning data set. R^2 score is monitored over this data set during fitting. Best weights regarding the validation R^2 score are saved and loaded after 30 epochs. The final model has $R^2 = 0.89$ and is noted, $\widehat{B}(x, c)$.

The fitted NN is then used to generate validation time series $[\mathbf{x}]^{\widehat{B}}$ of length 15 MTU, using $\widehat{B}$ instead of B in the L96 equations (Equation 5). Long-term metric $\widehat{m}$ is computed over $[\mathbf{x}]^{\widehat{B}}$, using standard deviation values only:

$$\widehat{m}(c) = \left(\widehat{\sigma}(c) - \sigma_{\text{ref}}\right)^2, \tag{9}$$

with σ_{ref} (resp. $\widehat{\sigma}(c)$) the standard deviation computed over the reference time series (resp. the validation time series). A time series of length 15 MTU, generated by integrating L96 equations (Equations 5 and 6) with $(h_0, F_0, b_0, c_0) = (1, 10, 10, 10)$, is considered as the "reference" data set.

As described in Section 2.3, a kriging metamodel, $\tilde{m}$, is fitted to approximate $\widehat{m}$ on a sample of $\left\{c_i, \widehat{m}(c_i)\right\}_{1\leq i\leq N_K}$, of size $N_k = 200$. The initial conditions for the orbits used to compute $\widehat{m}$ in the kriging learning sample are generated by LHS, with $c \in [7., 13.]$ (Figure 2). The value of c minimizing the *online* metric is retrieved on $\tilde{m}$, by using BFGS minimization from SciPy python package. The optimal value of c is $c^* = 9.922$. This value is very close to the value used to generate the reference data set, that is, $c_0 = 10$.

3.2. Imperfect Model Calibration: The Lorenz'96 Model

We now investigate the case where one of the L96 model parameters is carrying biases. To reproduce this situation, the L96 NN parameterization has been trained on output from the L96 model using the reference value of the model parameters, that is, $(h, F, b) = (h_0, F_0, b_0)$. However, the NN model will be implemented in an L96 system

where one of the model parameters is set to a value different from its reference value. We will show that optimizing the value of the tunable parameter included in the NN still allows us to obtain the wished long-term statistics.

Namely, we set the value of F to a range of biased values, $F_b \neq F_0$. For each F_b, we generate validation time serie using the NN parameterization, $\widehat{B}(x;c)$, and compute the corresponding long-term statistics. To underline the dependence of the long-term statistics on F_b, the *online* metric will be noted:

$$\widehat{m}_{F_b}(c) = \left(\widehat{\sigma}_{F_b}(c) - \sigma_{\text{ref}}\right)^2, \tag{10}$$

where $\widehat{\sigma}_{F_b}$ is the standard deviation computed on a validation time series where $F = F_b$.

To predict the estimated metric, $\widehat{m}_{F_b}(c)$, again, a kriging metamodel is trained over a data set $\left\{c_i, \widehat{m}_{F_b}(c_i)\right\}_{1\leq i\leq N_k}$ where c_i are $N_k = 200$ values sampled in the interval $[7, 13]$. Hence, a kriging metamodel is obtained for each F_b and the corresponding functions are noted $\tilde{m}_{F_b}(c)$.

We now optimize the value of c with respect to the *online* metric $\widehat{m}_{F_b}$. For each F_b, we find the optimal value of c, noted $c^*(F_b)$, by minimizing the corresponding kriging metamodel, $\tilde{m}_{F_b}$ (Figure 3, bottom panel). Bias compensation can be evaluated by comparing the value of $\tilde{m}_{F_b}$ computed on validation time series generated with $\widehat{B}(x;c_0)$ and $\widehat{B}(x;c^*)$, for each value of F_b. Whereas the loss values remain high for $c = c_0$ when F is strongly biased, the use of $c^*(F_b)$ significantly reduces the *online* loss value (Figure 3). As soon as $F_b \neq F_0$, $c^*(F_b) \neq c_0$. This means that the choice of $c = c_0$ is not optimal. It also means that at least part of the bias induced by the wrong parameter F_b can be actually compensated by tuning another parameter of the model, c. This result suggest that, at least to some extent, even an imperfect model can be tuned to adjust long-term statistics.

Figure 3 clearly shows that, as soon as $F_b \neq F_0$, the proposed method is a statistically significant improvement to the baseline (linear regression) physical parameterizations and the NN parameterization with $c = c_0$. Confidence intervals associated with our estimates of the metric have been constructed by bootstrap through the sampling of two main sources of uncertainty. First, we build 9 different NN parameterizations, each of which has been fitted to a learning sample generated with different initial conditions. Second, each one of the 9 parameterizations are used to generate 3 validation orbits of length 10, 15 and 25 MTUs. Thus, a sample of $n = 27$ validation orbits is obtained and used to compute 95% confidence intervals for both the "standard" ($c = c_0$) and the "optimal" ($c = c^*$) cases, and for each value of F_b.

4. Conclusion - Discussion

To improve current ML based physics in climate models, the effort is usually only focused on improving both the learning sample and the *offline* fit of the ML model. But these ML models can and need to be further improved by using an *online* metric to their calibration (Schneider et al., 2017). Thus, we propose a method to apply *online* calibration to ML parameterizations.

The key novelty of our approach is to include some of the physical parameters $\boldsymbol{\theta}$ among the input variables of the NN. The NN model is fitted *offline* to a learning sample of outputs from an high-resolution climate model. In this way, the NN parameterization is able to emulate the physics not only for one single $\boldsymbol{\theta}_0$, but for a range of values of $\boldsymbol{\theta}$. When the fitted NN model is plugged-in to replace the physical parameterization, the value of parameters $\boldsymbol{\theta}$ is calibrated *online*, as to reduce errors on long-term statistics of the climate model. As a proof-of-concept experiment, our methodology is demonstrated using L63 and L96 models. We show that our method can be used to optimize the value of some parameters to compensate long-term errors due to biases carried by another parameter which cannot be calibrated.

In addition to the reduction of long-term model errors, including some physical parameters among the ML model inputs can also increase the confidence we have in ML parameterizations, the interpretability of which is often questioned. Supplementary parameters can also be used to estimate uncertainties related to some processes. However, although satisfying results were obtained using toy models, further research is needed to test our methodology into real climate models. The generation of a learning sample for subgrid scale parameterizations is already a challenging issue; this task is even more difficult in our new method as a sampling of $\boldsymbol{\theta}$ is needed in addition to the sampling of $\boldsymbol{x}$. The numerical cost of the generation of the learning sample is thus increased. If our approach was to be applied to a climate model, the methodology described for the L96 model could be applied.

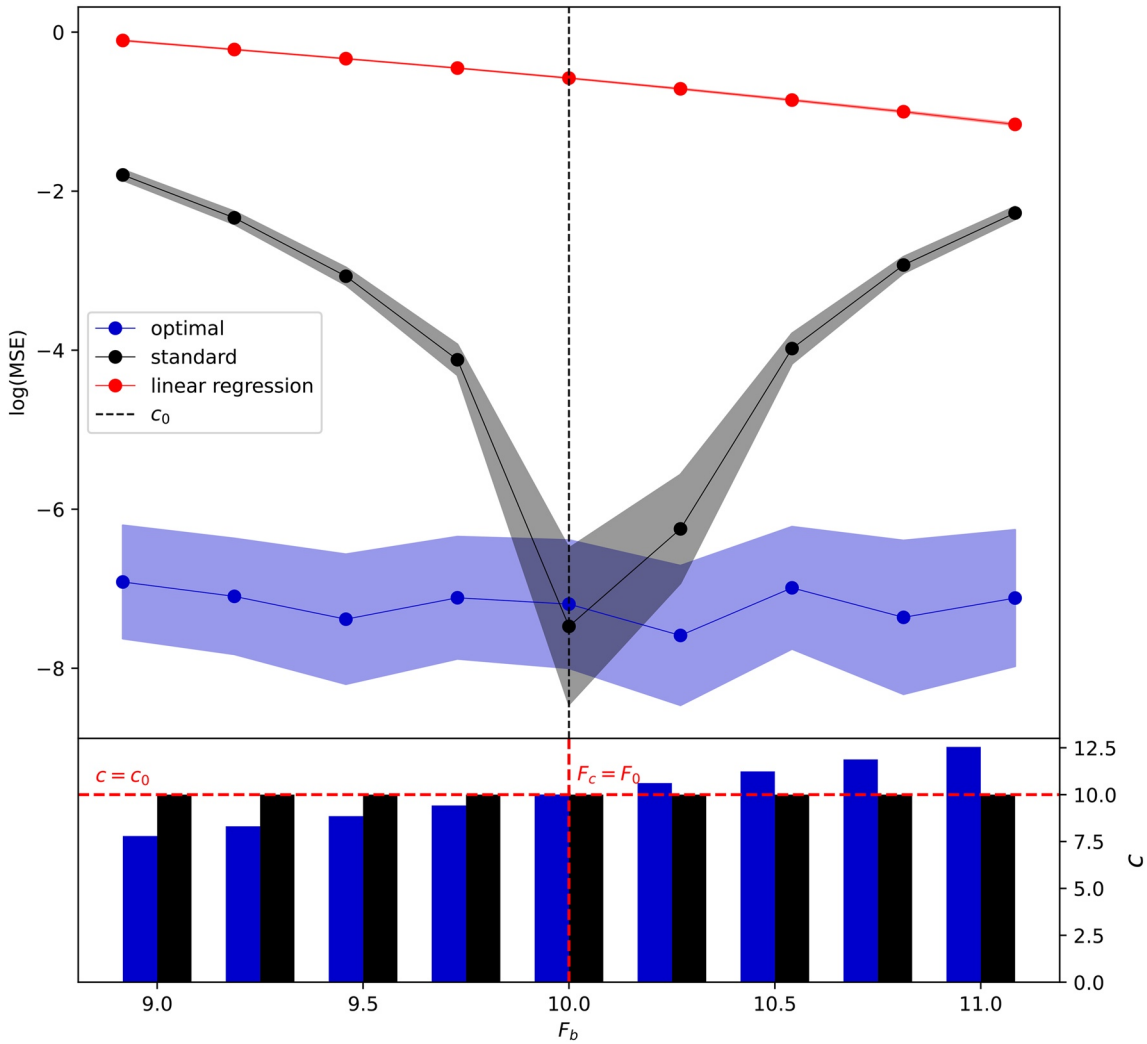

Figure 3. (black) Orbits computed with neural network parameterization $\widehat{B}(x; c)$ with $c = c_0$ for each F_b in the interval $[9, 11]$ (black bars, bottom panel). The logarithm of kriged values of the metric corresponding to these time series, $\tilde{m}_{F_b}(c_0)$, are plotted in the top panel (dotted solid black line). The 95% confidence intervals are represented in gray shading. This method can be compared with output using a linear regression approach, fitted to approximate B (red, top panel). (blue) For each F_b, we also compute the optimal value c^* of c to minimize the *online* metric $\tilde{m}_{F_b}$ (blue bars, bottom panel). The 95% confidence intervals are represented in blue shading. The minimal value of each F_b, $\tilde{m}_{F_b}(c^*)$, associated with orbits obtained with machine learning parameterization $\widehat{B}(x; c^*)$ remains close to zero (dotted solid blue line, top panel).

This would require a set of short high-resolution integrations for several values of $\boldsymbol{\theta}$, for example, taken from a LHS on $\boldsymbol{\theta}$ only. In this way the number of model integrations would be kept small, while preserving a large learning sample on $(\boldsymbol{x}, \boldsymbol{\theta})$ (as each integration provides a large sample of $\boldsymbol{x}$ values). In the L96 example, we did not discuss which minimum value of the number of integrations N_i is sufficient to efficiently emulate the system. Finding such a minimum value will require careful examination in the case of a climate model. Once a satisfying learning sample is built, the next hurdle to overcome is the fit of the NN model. For more complex models, a simple feed-forward NN may be insufficient and the choice of MSE as the *offline* metric may not be relevant. Finally, finding the optimal *online* metric can also be a strenuous issue.

Data Availability Statement

Code is made available at: https://zenodo.org/record/6141165.

Acknowledgments
We are grateful for the insightful comments and suggestions made by Olivier Geoffroy.

References

Arnold, H. M., Moroz, I. M., & Palmer, T. N. (2013). Stochastic parametrizations and model uncertainty in the Lorenz '96 system. *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, *371*(1991), 20110479. https://doi.org/10.1098/rsta.2011.0479

Balogh, B., Saint-Martin, D., & Ribes, A. (2021). A toy model to investigate stability of Ai-based dynamical systems. *Geophysical Research Letters*, *48*(8), e2020GL092133. https://doi.org/10.1029/2020gl092133

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., & Gentine, P. (2021). Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, *126*(9), 098302. https://doi.org/10.1103/physrevlett.126.098302

Brenowitz, N. D., Beucler, T., Pritchard, M., & Bretherton, C. S. (2020). Interpreting and stabilizing machine-learning parametrizations of convection. *Journal of the Atmospheric Sciences*, *77*(12), 4357–4375. https://doi.org/10.1175/jas-d-20-0082.1

Brenowitz, N. D., Henn, B., McGibbon, J., Clark, S. K., Kwa, A., Perkins, W. A., et al. (2020). *Machine learning climate model dynamics: Offline versus online performance*. arXiv:2011.03081 [physics]. (arXiv: 2011.03081).

Chattopadhyay, A., Hassanzadeh, P., & Subramanian, D. (2020). Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, *27*(3), 373–389. https://doi.org/10.5194/npg-27-373-2020

Cleary, E., Garbuno-Inigo, A., Lan, S., Schneider, T., & Stuart, A. M. (2021). Calibrate, emulate, sample. *Journal of Computational Physics*, *424*, 109716. https://doi.org/10.1016/j.jcp.2020.109716

Couvreux, F., Hourdin, F., Williamson, D., Roehrig, R., Volodina, V., Villefranque, N., et al. (2021). Process-based climate model development harnessing machine learning: I. A calibration tool for parameterization improvement. *Journal of Advances in Modeling Earth Systems*, *13*(3), e2020MS002217. https://doi.org/10.1029/2020ms002217

Cressie, N. (1992). *Statistics for spatial data*. Wiley Online Library.

Fletcher, R. (2013). *Practical methods of optimization*. Wiley Online Library.

Gagne, D. J., Christensen, H. M., Subramanian, A. C., & Monahan, A. H. (2020). Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz '96 model. *Journal of Advances in Modeling Earth Systems*, *12*(3), e2019MS001896. https://doi.org/10.1029/2019MS001896

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., & Yacalis, G. (2018). Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, *45*(11), 5742–5751. https://doi.org/10.1029/2018gl078202

Hourdin, F., Mauritsen, T., Gettelman, A., Golaz, J.-C., Balaji, V., Duan, Q., et al. (2017). The art and science of climate model tuning. *Bulletin of the American Meteorological Society*, *98*(3), 589–602. https://doi.org/10.1175/bams-d-15-00135.1

Jakob, C. (2010). Accelerating progress in global atmospheric model development through improved parameterizations: Challenges, opportunities, and strategies. *Bulletin of the American Meteorological Society*, *91*(7), 869–876. https://doi.org/10.1175/2009bams2898.1

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *202*, 130–141. https://doi.org/10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2

Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proc. ECMWF Seminar on Predictability*. (Vol. I, pp. 1–18).

McKay, M. D. (1992). Latin Hypercube Sampling as a tool in uncertainty analysis of computer models. *Proceedings of the 24th Conference on Winter Simulation*, 557–564. https://doi.org/10.1145/167293.167637

Medeiros, B., & Stevens, B. (2011). Revealing differences in GCM representations of low clouds. *Climate Dynamics*, *36*(1–2), 385–399. https://doi.org/10.1007/s00382-009-0694-5

Medeiros, B., Stevens, B., Held, I. M., Zhao, M., Williamson, D. L., Olson, J. G., & Bretherton, C. S. (2008). Aquaplanets, climate sensitivity, and low clouds. *Journal of Climate*, *21*(19), 4974–4991. https://doi.org/10.1175/2008jcli1995.1

Rasp, S. (2020). Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: General algorithms and Lorenz 96 case study (v1.0). *Geoscientific Model Development*, *13*(5), 2185–2196. https://doi.org/10.5194/gmd-13-2185-2020

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684–9689. https://doi.org/10.1073/pnas.1810286115

Scher, S., & Messori, G. (2019). Generalization properties of feed-forward neural networks trained on Lorenz systems. *Nonlinear Processes in Geophysics*, *26*(4), 381–399. https://doi.org/10.5194/npg-26-381-2019

Schmidt, G. A., Bader, D., Donner, L. J., Elsaesser, G. S., Golaz, J.-C., Hannay, C., et al. (2017). Practice and philosophy of climate model tuning across six US modeling centers. *Geoscientific Model Development*, *10*(9), 3207–3223. https://doi.org/10.5194/gmd-10-3207-2017

Schneider, T., Lan, S., Stuart, A., & Teixeira, J. (2017). Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations: Earth system modeling 2.0. *Geophysical Research Letters*, *44*(24), 12396–12417. https://doi.org/10.1002/2017GL076101

Stevens, B., & Bony, S. (2013). What are climate models missing? *Science*, *340*(6136), 1053–1054. https://doi.org/10.1126/science.1237554

Watson, P. A. G. (2019). Applying machine learning to improve simulations of a chaotic dynamical system using empirical error correction. *Journal of Advances in Modeling Earth Systems*, *11*(5), 1402–1417. https://doi.org/10.1029/2018ms001597

Yuval, J., & O'Gorman, P. A. (2020). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Communications*, *11*(1), 3295. https://doi.org/10.1038/s41467-020-17142-3

Yuval, J., O'Gorman, P. A., & Hill, C. N. (2021). Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, *48*(6), e2020GL091363. https://doi.org/10.1029/2020gl091363