Data-driven parameterizations in numerical models using data assimilation and machine learning.

Julien Brajard[1,2], Alberto Carrassi[3,4], Marc Bocquet[5], Laurent Bertino[1], Arthur Filoche[2], Dominique Béréziat, Anastase Charantonis[6]

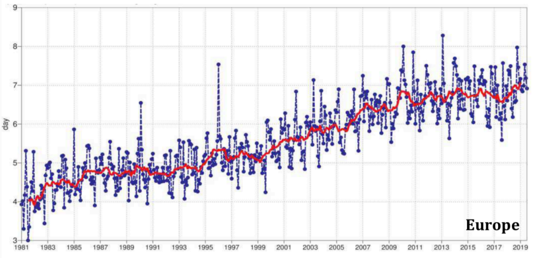
December 16, 2020

[1]NERSC, [2]Sorbonne Université, [3]University of Reading, [4]Unversity of Utrecht, [5]École des Ponts ParisTech, [6] ENSIEE

Introduction

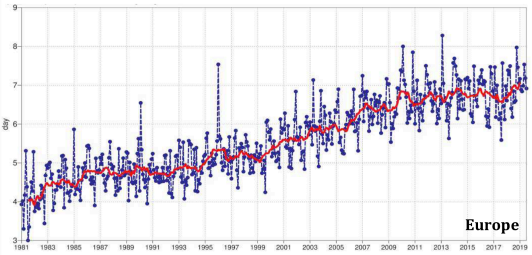
Forecast skill evolution:

source: ECMWF technicql report 2019



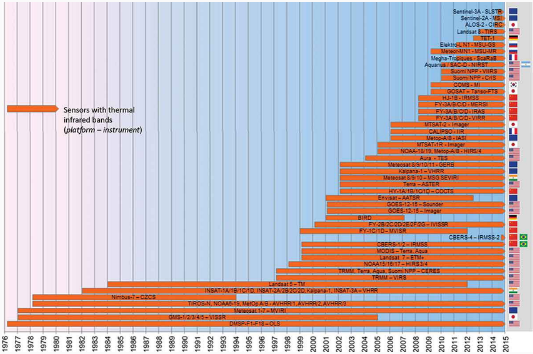
Forecast skill evolution:

source: ECMWF technicql report 2019



Data availability:

source: Kuenzer et al. (2014)



”Big” question addressed in this talk

Can data help producing better forecasts?

”Big” question addressed in this talk

Can data help producing better forecasts?

Can data help producing better models?

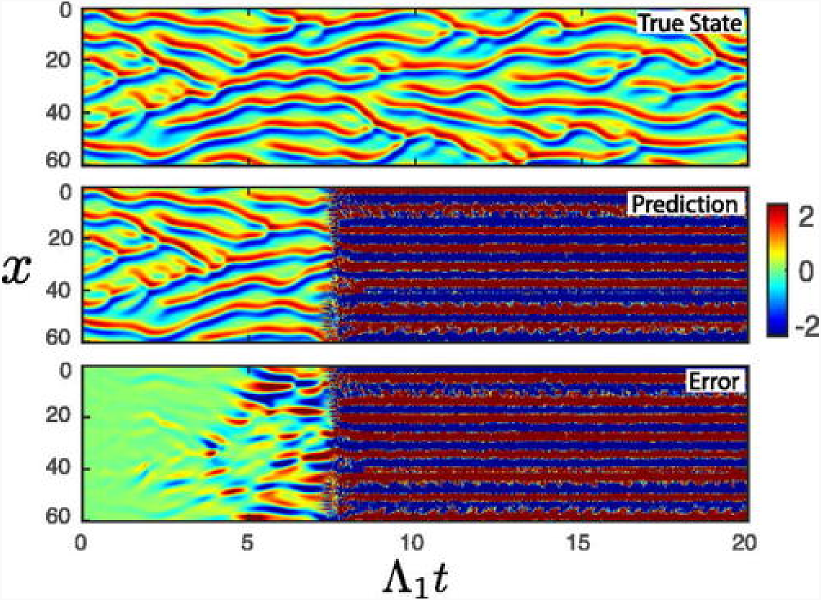
Improving the forecast

- How to improve the forecast skill?
  - Estimating accurate initial conditions
  - Improving model physics
  - Running a Higher-resolution model or having a good parametrization for unresolved scales.

Improving the forecast

- How to improve the forecast skill?
  - Estimating accurate initial conditions
  - Improving model physics
  - Running a Higher-resolution model or having a good parametrization for unresolved scales.
- Data can be used to:
  - estimation the initial conditions: can be achieved by data assimilation.
  - Emulate completely or partially dynamics of the model.

Does better forecasts implies a better model (and vice-versa)?



[Pathak et al., 2017]

Table of contents

1. Build an emulator

2. Unresolved scale parametrization

    2-scales Lorenz model

    Coupled ocean-atmosphere model MAOOAM

    The case of a never-observed variable

Build an emulator

Emulate a numerical model

We assume the existence of unknown dynamical model represented by an ordinary differential equation:

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi(\mathsf{x}) \quad \text{with resolvent} \quad \mathsf{x}_{k+1} = \mathcal{M}(\mathsf{x}_k) = \mathsf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathsf{x})\,\mathrm{d}t$$

Objectif

Emulate $\mathcal{M}$ by a data-driven model $\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k)$ (e.g. a neural network), by minimizing, e.g., $L(\mathsf{W}) = \sum_{k=0}^{K} \|\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) - \mathsf{x}_{k+1}\|^2$, where K is the number of available samples.

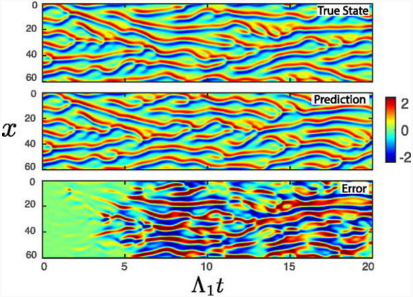
Emulate a numerical model

We assume the existence of unknown dynamical model represented by an ordinary differential equation:

$$\frac{\mathrm{d}\mathrm{x}}{\mathrm{d}t} = \phi(\mathrm{x}) \quad \text{with resolvent} \quad \mathrm{x}_{k+1} = \mathcal{M}(\mathrm{x}_k) = \mathrm{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathrm{x})\,\mathrm{d}t$$

Objectif

Emulate $\mathcal{M}$ by a data-driven model $\mathcal{G}_{\mathrm{W}}(\mathrm{x}_k)$ (e.g. a neural network), by minimizing, e.g., $L(\mathrm{W}) = \sum_{k=0}^{K} \|\mathcal{G}_{\mathrm{W}}(\mathrm{x}_k) - \mathrm{x}_{k+1}\|^2$, where K is the number of available samples.



from [Pathak et al., 2017], Fig. 4.

With sparse and noisy data

Let us consider the case where x_k is not known perfectly. We only have access to noisy and sparse observations $\mathsf{y}_k^{\mathrm{obs}}$:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{o}} \qquad \epsilon_k^{\mathrm{o}} \in \mathcal{N}(0, \mathbb{R})$$

Both the emulator $\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k)$ and the state x_k has to be determined.

With sparse and noisy data

Let us consider the case where x_k is not known perfectly. We only have access to noisy and sparse observations $\mathsf{y}_k^{\mathrm{obs}}$:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{o}} \qquad \epsilon_k^{\mathrm{o}} \in \mathcal{N}(0, \mathbb{R})$$

Both the emulator $\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k)$ and the state x_k has to be determined.

Idea

Combining DA and ML to develop accurate numerical emulator from imperfect data.

With sparse and noisy data

Let us consider the case where x_k is not known perfectly. We only have access to noisy and sparse observations $\mathsf{y}_k^{\mathrm{obs}}$:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{o}} \qquad \epsilon_k^{\mathrm{o}} \in \mathcal{N}(0, \mathbb{R})$$

Both the emulator $\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k)$ and the state x_k has to be determined.

Idea

Combining DA and ML to develop accurate numerical emulator from imperfect data.

What is Data Assimilation good at?

Given a numerical model, some observations and assumptions on uncertainties:

- Estimate the state of a system in an objective way,
- Estimate the uncertainty of the state.

With sparse and noisy data

Let us consider the case where x_k is not known perfectly. We only have access to noisy and sparse observations $\mathsf{y}_k^{\mathrm{obs}}$:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{o}} \qquad \epsilon_k^{\mathrm{o}} \in \mathcal{N}(0, \mathbb{R})$$

Both the emulator $\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k)$ and the state x_k has to be determined.

Idea

Combining DA and ML to develop accurate numerical emulator from imperfect data.

What is Data Assimilation good at?

Given a numerical model, some observations and assumptions on uncertainties:

- Estimate the state of a system in an objective way,
- Estimate the uncertainty of the state.

What is Machine Learning good at?

Given a “good enough” dataset:

- Retrieve some hidden relationships in the dataset.

With sparse and noisy data

Let us consider the case where $\mathbf{x}_k$ is not known perfectly. We only have access to noisy and sparse observations $\mathbf{y}_k^{\text{obs}}$:

$$\mathbf{y}_k^{\text{obs}} = \mathcal{H}_k(\mathbf{x}_k) + \epsilon_k^{\text{o}} \qquad \epsilon_k^{\text{o}} \in \mathcal{N}(0, \mathbb{R})$$

Both the emulator $\mathcal{G}_{\mathbf{W}}(\mathbf{x}_k)$ and the state $\mathbf{x}_k$ has to be determined.

Idea

Combining DA and ML to develop accurate numerical emulator from imperfect data.

What is Data Assimilation good at?

Given a numerical model, some observations and assumptions on uncertainties:

- Estimate the state of a system in an objective way,
- Estimate the uncertainty of the state.

What is Machine Learning good at?

Given a "good enough" dataset:

- Retrieve some hidden relationships in the dataset.

[Abarbanel et al., 2018, Bocquet et al., 2019, Bocquet et al., 2020, Brajard et al., 2020a, Nguyen et al., 2020]

Combining DA and ML

More details in [Brajard et al., 2020a]

Emulation of the resolvent **combining** DA and ML:

$$\mathsf{x}_{k+1} = \mathcal{M}(\mathsf{x}_k) \approx \mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) + \epsilon_k^{\mathrm{m}},$$

where $\mathcal{G}_{\mathsf{W}}$ is a neural network parameterised by W and ϵ_k^{m} is a stochastic noise.

Combining DA and ML

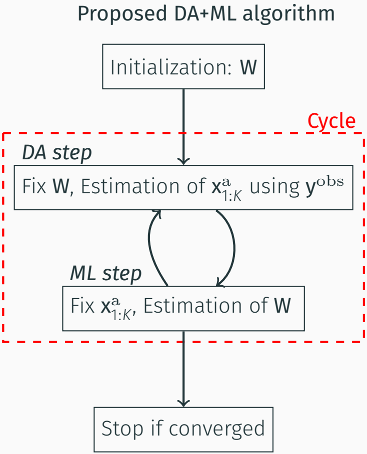
More details in [Brajard et al., 2020a]

Emulation of the resolvent **combining** DA and ML:

$$\mathbf{x}_{k+1} = \mathcal{M}(\mathbf{x}_k) \approx \mathcal{G}_{\mathbf{W}}(\mathbf{x}_k) + \epsilon_k^{\mathrm{m}},$$

where $\mathcal{G}_{\mathbf{W}}$ is a neural network parameterised by $\mathbf{W}$ and ϵ_k^{m} is a stochastic noise.

- For the DA step we use the Finite-Size Ensemble Kalman Filter (EnKF-N) [Bocquet 2011].
- For the ML step we train a neural net



Proposed DA+ML algorithm

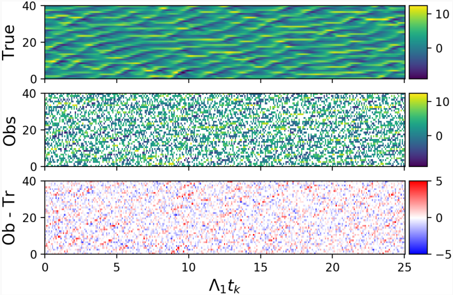
- Step 1 - Data Assimilation: estimate the state field $\mathsf{x}_{1:K}$ (the analysis) and associated (analysis) error covariance, P_k, based on the fixed model parameters W and using sparse and noisy data, y.
- Step 2 - Machine learning: using $\mathsf{x}_{1:K}$ and P_k from DA estimate W
  - The neural network can be expressed as a parametric function $\mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) = \mathsf{x}_k + f_{\mathrm{nn}}(\mathsf{x}_k, \mathsf{W})$ where f_{nn} is a neural network and W its weights; f_{nn} is composed of convolutive layers.
  - The determination of the optimal W is done in the *training phase* by minimising the loss function:

$$L(\mathsf{W}) = \sum_{k=0}^{K-N_{\mathrm{f}}-1} \sum_{i=1}^{N_{\mathrm{f}}} \left\| \mathcal{G}_{\mathsf{W}}^{(i)}(\mathsf{x}_k) - \mathsf{x}_{k+i} \right\|^2_{\mathsf{P}_k^{-1}},$$

    where N_{f} is the number of time steps corresponding to the forecast lead time on which the error between the simulation and the target is minimised (with "coordinate descent" [Bocquet et al., 2020]).
  - P_k is a symmetric, semi-definite positive matrix estimated using the *analysis error covariances* from the DA step.
  - This time-dependent matrix, P_k, plays the role of the surrogate model error covariance matrix and gives different weights to each state during the optimisation process.

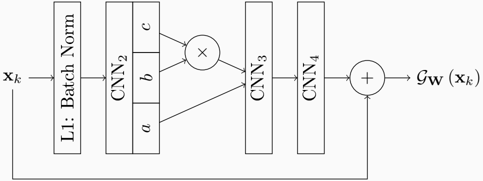
Numerical experiment: Lorenz 96 model

- A simulation is performed over $K = 40,000$ time steps: $\mathrm{x}_{0:K}^{\mathrm{ref}}$
- $\mathrm{y}_k^{\mathrm{obs}} = \mathcal{H}_t(\mathrm{x}_k^{\mathrm{ref}}) + \epsilon_k^{\mathrm{obs}}$; $\mathrm{y}_t^{\mathrm{obs}} \in \mathbb{R}^p$



- $\mathcal{H}_k$ is defined at each time step by randomly sample p=20 observations (50% of the state space).
- $\epsilon_k^{\mathrm{obs}}$ is generated using a Gaussian law of mean 0 and standard deviation 1.

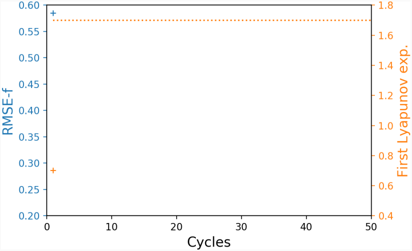
Neural Network setup

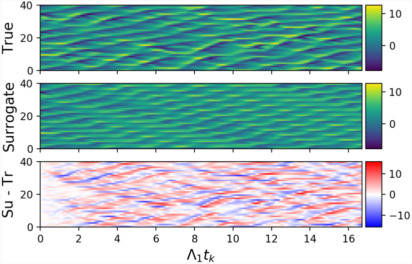


Layer	number of unit	filter size	number of weights
1 (batchnorm)			2
2 (bilinear)	24×3	5	144×3
3 (convolutive)	37	5	8917
4 (linear)	1	1	38

Residual bi-linear convolutive neural network (9391 weights)

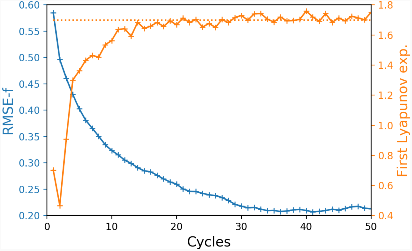
Convergence of the algorithm

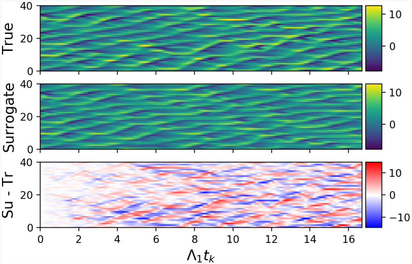




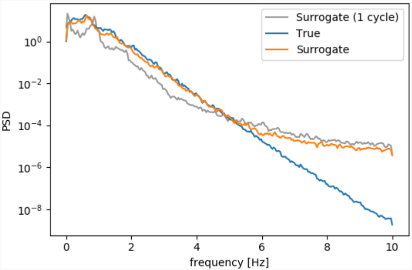
- The true 1st Lyapunov exponent is $\Lambda_1 = 1.67$.
- RMSE-f after 1 time step is shown.

Convergence of the algorithm



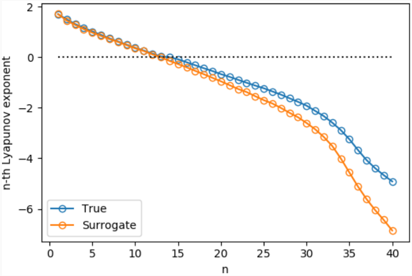


- The true 1st Lyapunov exponent is $\Lambda_1 = 1.67$.
- RMSE-f after 1 time step is shown.



- After one cycle, some frequencies are favoured (see the peak at ~ 0.8Hz) and indicate that the **periodic signals are learnt first**.
- At convergence, the surrogate model reproduces the spectrum up to 5 Hz but then **adds high-frequency noise**.
- **Low frequencies are better observed** and better reproduced after the DA step.
- The PSD has been computed using a long simulation ($16,000$ time steps), which means that **the surrogate model is very stable**.

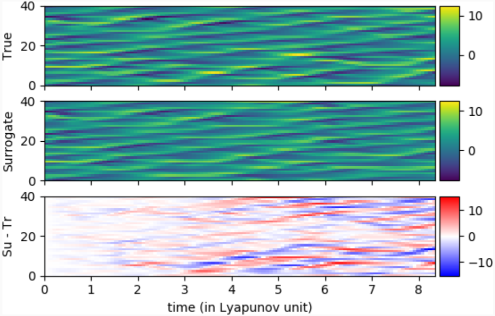
Emulating the underlying dynamics: *Lyapunov spectrum*



- **Accurate unstable spectrum** ⇒ Same error growth rate and Kolmogorov entropy, as the true model.
- Less for the neutral and quasi-neutral part of the spectrum.
- This is similar to what found in[Pathak et al., 2017]. Maybe due to the slower convergence (linear vs exponential) of the neutral exps.
- The stable part of the spectrum is shifted toward smaller values ⇒ PDFs contracts faster than in the true model, *i.e.* **surrogate model more dissipative than the truth**.

Forecast skill

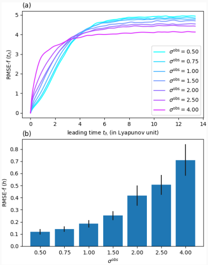
Hovmøller plot of the true and surrogate models (in Lyapunov time, LT)



- The simulations start from the same initial conditions.
- Good prediction until 2 LTs. Error saturation at 4-5 LTs.

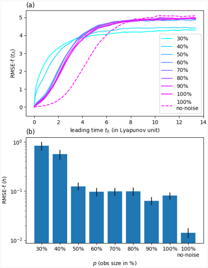
Sensitivity to observation scenario

vs Obs error



- RMSE-f deteriorates as σ^{o} increases.
- Asymptotic RMSE-f inverse proportional to σ^{o}.
- Training on (very) noisy data leads to underestimate the forecast variance (also less extreme values)

vs # Obs



- The perfect ($\sigma^{\mathrm{obs}} = 0$) and complete case ($p = 100\%$) is shown for reference.
- Strong degradation for $p < 50\%$ but little sensitivity for $p \geq 50\%$.
- This suggests DA is successful in providing accurate analysis as soon as half of the domain is observed.

Unresolved scale parametrization

Using ML for parametrization

We now assume that we know partially the system dynamics:

$$\mathsf{x}(t+\delta t) = \mathcal{M}^{\varphi}[\mathsf{x}(t)] + \mathcal{M}^{\mathrm{UN}}[(t)],$$

where:

- $\mathsf{x}(t)$ is the state of the dynamical system
- $\mathcal{M}^{\varphi}$ is the physical model (assumed to be known a priori)
- $\mathcal{M}^{\mathrm{UN}}$ is the unresolved component of the dynamics to be inferred from data
- δt is the integration time step

Using ML for parametrization

We now assume that we know partially the system dynamics:

$$\mathsf{x}(t+\delta t) = \mathcal{M}^{\varphi}[\mathsf{x}(t)] + \mathcal{M}^{\mathrm{UN}}[(t)],$$

where:

- $\mathsf{x}(t)$ is the state of the dynamical system
- $\mathcal{M}^{\varphi}$ is the physical model (assumed to be known a priori)
- $\mathcal{M}^{\mathrm{UN}}$ is the unresolved component of the dynamics to be inferred from data
- δt is the integration time step

Objective

Approximate $\mathcal{M}^{\mathrm{UN}}$ is approximated by a data-driven model represented under the form of a neural network whose parameters are $\boldsymbol{\theta}$: $\mathcal{M}_{\boldsymbol{\theta}}[\mathsf{x}(t)]$

Using ML for parametrization

We now assume that we know partially the system dynamics:

$$\mathrm{x}(t+\delta t) = \mathcal{M}^{\varphi}[\mathrm{x}(t)] + \mathcal{M}^{\mathrm{UN}}[(t)],$$

where:

- $\mathrm{x}(t)$ is the state of the dynamical system
- $\mathcal{M}^{\varphi}$ is the physical model (assumed to be known a priori)
- $\mathcal{M}^{\mathrm{UN}}$ is the unresolved component of the dynamics to be inferred from data
- δt is the integration time step

Objective

Approximate $\mathcal{M}^{\mathrm{UN}}$ is approximated by a data-driven model represented under the form of a neural network whose parameters are $\boldsymbol{\theta}$: $\mathcal{M}_{\boldsymbol{\theta}}[\mathrm{x}(t)]$

It has been done using high-resolution model as the ”truth” [Rasp et al., 2018, Brenowitz and Bretherton, 2018, O’Gorman and Dwyer, 2018, Bolton and Zanna, 2019]:

Using ML for parametrization

We now assume that we know partially the system dynamics:

$$\mathsf{x}(t+\delta t) = \mathcal{M}^{\varphi}[\mathsf{x}(t)] + \mathcal{M}^{\mathrm{UN}}[(t)],$$

where:

- $\mathsf{x}(t)$ is the state of the dynamical system
- $\mathcal{M}^{\varphi}$ is the physical model (assumed to be known a priori)
- $\mathcal{M}^{\mathrm{UN}}$ is the unresolved component of the dynamics to be inferred from data
- δt is the integration time step

Objective

Approximate $\mathcal{M}^{\mathrm{UN}}$ is approximated by a data-driven model represented under the form of a neural network whose parameters are $\boldsymbol{\theta}$: $\mathcal{M}_{\boldsymbol{\theta}}[\mathsf{x}(t)]$

It has been done using high-resolution model as the "truth" [Rasp et al., 2018, Brenowitz and Bretherton, 2018, O'Gorman and Dwyer, 2018, Bolton and Zanna, 2019]:

- High-resolution model are very expensive (especially in the case of coupled fast/slow dynamics)
- There is no guaranty that high-resolution models will converge toward the observed state.

Few comments:

- If the physical model is hard-coded within a neural network, the problem is equivalent to what has been presented in the first part, but it is very costly in general. The physical model, which can be complex, has to be coded in a neural network framework.

Determining a parametrization from sparse and noisy data

Few comments:

- If the physical model is hard-coded within a neural network, the problem is equivalent to what has been presented in the first part, but it is very costly in general. The physical model, which can be complex, has to be coded in a neural network framework.
- The time step of the observation is usually larger than the integration time step.

Determining a parametrization from sparse and noisy data

Few comments:

- If the physical model is hard-coded within a neural network, the problem is equivalent to what has been presented in the first part, but it is very costly in general. The physical model, which can be complex, has to be coded in a neural network framework.
- The time step of the observation is usually larger than the integration time step.
- We generally cannot compute the gradient (or adjoint) of the physical model.

Determining a parametrization from sparse and noisy data

Few comments:

- If the physical model is hard-coded within a neural network, the problem is equivalent to what has been presented in the first part, but it is very costly in general. The physical model, which can be complex, has to be coded in a neural network framework.
- The time step of the observation is usually larger than the integration time step.
- We generally cannot compute the gradient (or adjoint) of the physical model.

Position of our approach:

- We do not rely on an adjoint of the physical model
- the training of the neural network is made separately to the integration of the physical model.

Related works in the context of model error estimations in data assimilation [Bonavita and Laloyaux, 2020].

Proposed approach

More details in [Brajard et al., 2020b]

Observation Setup

Observations $\mathbf{y}_k$ are assumed to be made at each Δt time step such as $\Delta t = N_c \delta t$ (N_c is a positive integer and δt is the integration time step).

Simplified description of the algorithm:

1. Estimating the state $\mathbf{x}^{\mathrm{a}}_{1:K}$. At each time t_k, we calculate a forecast $\mathbf{x}^{\mathrm{f}}$:

$$\mathbf{x}^{\mathrm{f}}_{k+1} = \mathbf{x}^{\mathrm{f}}(t_k + \Delta t) = (\mathcal{M}^{\varphi})^{N_c}(\mathbf{x}^{\mathrm{a}}_k)$$

   An observation $\mathbf{y}_{k+1}$ is assimilated to produce an analysis state $\mathbf{x}^{\mathrm{a}}_{k+1}$
2. Determining an estimation of the unknown part of the model. We assume that:
   - $\mathbf{x}(t + \Delta t) \approx (\mathcal{M}^{\varphi})^{N_c}(\mathbf{x}(t)) + N_c \cdot \mathcal{M}^{\mathrm{UN}}[\mathbf{x}(t)]$
   - $\mathbf{x}(t) \approx \mathbf{x}^{\mathrm{a}}(t)$

   We consider that $\mathcal{M}^{\mathrm{UN}}(\mathbf{x}_k) \approx \mathbf{z}_{k+1} = 1/N_c \cdot \left(\mathbf{x}^{\mathrm{a}}_{k+1} - \mathbf{x}^{\mathrm{f}}_{k+1}\right)$
3. Training a neural network $\mathcal{M}_{\boldsymbol{\theta}}$ by minimising the loss $L(\boldsymbol{\theta}) = \sum_{k=0}^{K-1} ||\mathcal{M}_{\boldsymbol{\theta}}(\mathbf{x}^{\mathrm{a}}_k) - \mathbf{z}_{k+1}||^2$
4. Using the hybrid model $\mathcal{M}^{\varphi} + \mathcal{M}_{\boldsymbol{\theta}}$ to produce new simulations (*e.g.* to make forecasts).

Algorithm setup

Data assimilation

EnKF-N with a ensemble of size 50 an additive noise at each time step δt
https://github.com/nansencenter/DAPPER

Neural network

- The neural network is composed of 3 convolutional *or* 3 multi-layer perceptrons layers. Hyperparameters (size of each layer, batchsize, optimizer, regularization, ...) are determined via Bayesian optimisation (***hypertopt*** package).
- An upper bound hybrid model is trained with "true data" ($\mathbf{x}_k^{\mathrm{a}} = \mathbf{x}_k$).
- The "target" (*i.e.* the model error) is estimated using the *analysis increments*, $\left(\mathbf{x}_{k+1}^{\mathrm{a}} - \mathbf{x}_{k+1}^{\mathrm{f}}\right)$.
- $\left(\mathbf{x}_{k+1}^{\mathrm{a}} - \mathbf{x}_{k+1}^{\mathrm{f}}\right)$ contains both i.c. and model error. The former is assumed to have high frequencies.
- Therefore the time series $\mathbf{x}_{0:K}^{\mathrm{a}}$ estimated by DA is filtered using a simple low-pass filter (a rolling mean) producing a smoothed time series $\mathbf{x}_{0:K}^{\mathrm{s}}$

Numerical experiments: Lorenz 2-scales

- The "unknown" model
  (to be represented by a neural network)
- The physical model
  (assumed to be known)

$$
\begin{aligned}
\frac{\mathrm{d}x_n}{\mathrm{d}t} &= \boxed{\psi_n^+(\mathbf{x}) + F} - \boxed{h\frac{c}{b}\sum_{m=0}^{9} u_{m+10n}}\,, \\
\frac{\mathrm{d}u_m}{\mathrm{d}t} &= \frac{c}{b}\psi_m^-(b\mathbf{u}) + h\frac{c}{b}x_{m/10}, \\
\psi_n^\pm(\mathbf{x}) &= x_{n\mp1}(x_{n\pm1} - x_{n\mp2}) - u_n,
\end{aligned}
$$

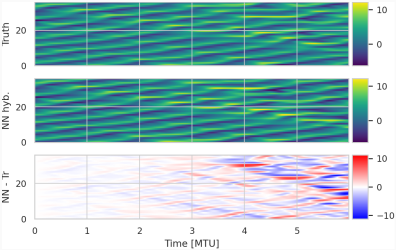
$n = 0, \cdots, N_x - 1\,(N_x = 36)$, $m = 0, \cdots, N_u - 1\,(N_u = 360)$, $(c, b, h, F) = (10, 10, 1, 10)$.

Data generation

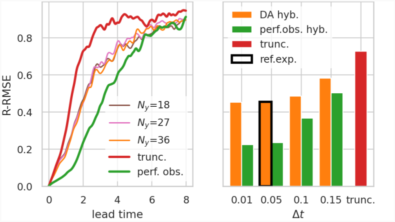
The full model (+) is integrated using RK4 scheme with an integration time step $\delta t = 0.005$ to generate the true field $\mathbf{x}_{0:K}$. The observations are produced at each $\Delta t = 3 \cdot \delta t$ time steps by perturbing the true field with a centred Gaussian of standard deviation $\sigma^{\mathrm{o}} = 1$.

Numerical experiments: Lorenz 2-scales

Hovmøller plot of the true and hybrid models



RMSE-f *vs* time RMSE-f *vs* obs interval Δt



- The hybrid model has **predictive skill until 3-4 MTU**, *i.e.* approximately until 3 Lyapunov times.

- The skill is **not much sensitive** to changing the number of observations, N_y.

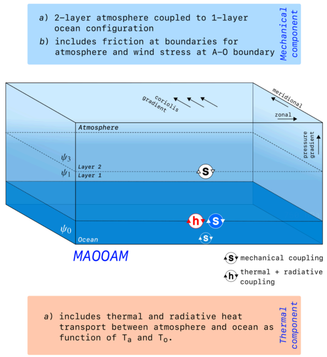
- **More sensitive** to observation frequency: for large Δt the target analysis increment contains coupled signal between resolved and unresolved scales.

Numerical experiments: atmosphere-ocean model MAOOAM

▶ **MAOOAM**: Modular arbitrary-order ocean-atmosphere model [De Cruz et al., 2016]

▶ A two-layer QG atmosphere coupled, thermally and mechanically, to a QG shallow-water ocean layer in the β-plane.

▶ MAOOAM is resolved in spectral space, for streamfunction and potential temperature, with adjustable resolution.



Configurations

1. **Truth**: $n_a = 20$ and $n_o = 8$ modes for atmosphere and ocean. **Total dimension** $N_x = 56$.
2. **Truncated**: $n_a = 10$ and $n_o = 8$ modes for atmosphere and ocean. **Total dimension** $N_x = 36$.

▶ The truncated model is **missing 20 high-order atmospheric variables**

Numerical experiments: atmosphere-ocean model MAOOAM

▶ Both ocean and atmosphere are observed (**coupled DA**) every 27 hours (see [Tondeur et al., 2020])

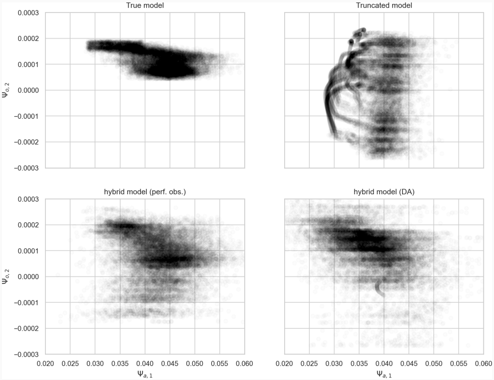
▶ There is not locality in spectral space so the NN is made of 3 layers multi-layer perceptrons

RMSE-f of hybrid and truncated MAOOAM models

RMSE-f(lead time τ)	$\psi_{o,2}$(2 years)	$\theta_{o,2}$(2 years)	$\psi_{a,1}$(1 day)
Truncated	0.23	0.21	0.36
Perfect obs. hybrid	0.07	0.07	0.23
DA hybrid	**0.10**	**0.06**	**0.28**

- The hybrid models have superior skill to the truncated model.
- The improvement is larger for the ocean that is fully resolved: it is thus fully due to an enhanced representation of the atmosphere-ocean coupling processes.
- The atmosphere is improved less: the hybrid is not very good in representing the fast processes.
- But they are also poorly observed with $\Delta t = 27$ hrs

Reconstructing the model attractor

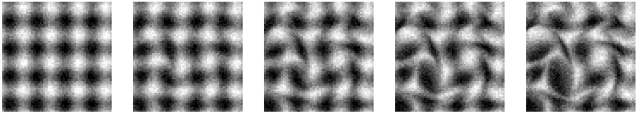


- Cross-section of the attractor for two key variables $\psi_{a,1}$ and $\psi_{o,2}$.
- The truncated model visits areas of the phase space that are not admitted in the real dynamics.
- Discrepancies are reduced by the hybrid models; some states remain out of the true model attractor, tough, but much fewer.

[Filoche et al., 2020]

Available information :

Noisy Observations: $Y_t = I_t + \varepsilon_R$



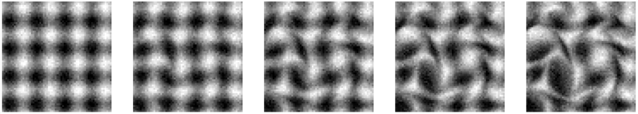
Incomplete dynamics: $\frac{\partial w}{\partial t} = ?$

Numerical experiments: unobserved variable

[Filoche et al., 2020]

Available information :

Noisy Observations: $Y_t = I_t + \varepsilon_R$



Incomplete dynamics: $\dfrac{\partial \mathrm{w}}{\partial t} = ?$

Hybrid model: Combining a **numerical scheme** of the known physics with a **CNN**

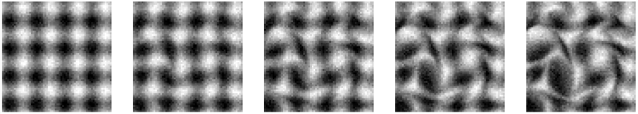
$\mathbb{M}_P + \mathbb{M}_L(\boldsymbol{\theta}) = \mathbb{M}_\theta$ the resolvent of the following PDE-system
$$\begin{cases} \dfrac{\partial I}{\partial t} + \mathrm{w}.\nabla I = 0 \\ \dfrac{\partial \mathrm{w}}{\partial t} + f_\theta(\mathrm{w}) = 0 \\ \nabla I = 0, \quad \partial\Omega \\ \mathrm{w} = 0, \quad \partial\Omega \end{cases}$$

Numerical experiments: unobserved variable

[Filoche et al., 2020]

Available information :

Noisy Observations: $Y_t = I_t + \varepsilon_R$



Incomplete dynamics: $\frac{\partial \mathbf{w}}{\partial t} = ?$

Hybrid model: Combining a **numerical scheme** of the known physics with a **CNN**

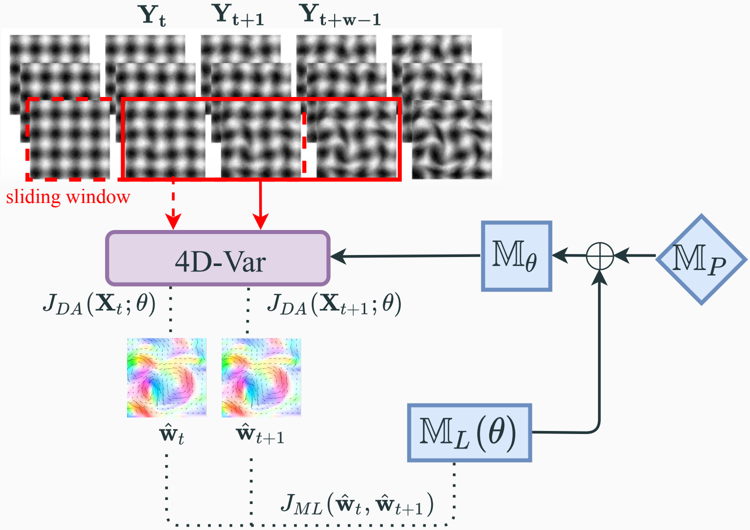
$\mathbb{M}_P + \mathbb{M}_L(\boldsymbol{\theta}) = \mathbb{M}_\theta$ the resolvent of the following PDE-system

$$\begin{cases} \frac{\partial I}{\partial t} + \mathbf{w}.\nabla I = 0 \\ \frac{\partial \mathbf{w}}{\partial t} + f_\theta(\mathbf{w}) = 0 \\ \nabla I = 0, \quad \partial\Omega \\ \mathbf{w} = 0, \quad \partial\Omega \end{cases}$$

Goal : Training the CNN i.e. Learning a dynamics on a never-observed variable

Numerical experiments: unobserved variable

All optimizations are done with deep learning tools based on automatic differentiation



Numerical experiments: unobserved variable

Model evaluation

We produce forecasts over multiple initial conditions (not used during the training) and compare them with ground truth trajectories calculating RMSE on I. We compare the following dynamics:

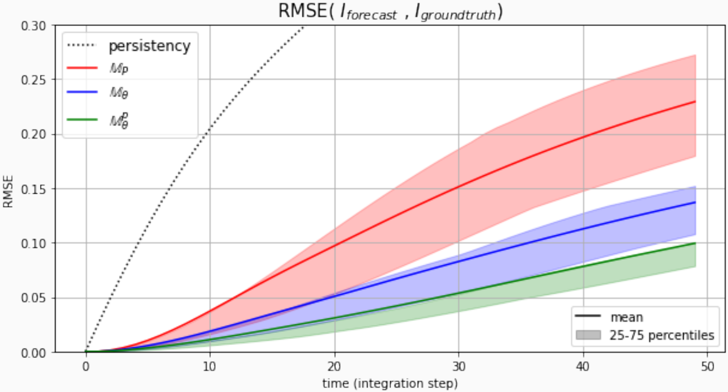
- ▷ Incomplete physics-based model
- ▷ Hybrid model trained on perfect data
- ▷ Hybrid model trained with our method

Numerical experiments: unobserved variable

Model evaluation

We produce forecasts over multiple initial conditions (not used during the training) and compare them with ground truth trajectories calculating RMSE on *I*. We compare the following dynamics:

- ▷ Incomplete physics-based model
- ▷ Hybrid model trained on perfect data
- ▷ Hybrid model trained with our method



Conclusion

- Part I - Build a surrogate model from partial and noisy data:
  - Accurate reproduction of the more energetic (slower) scale and of the unstable Lyapunov exponents
  - Method sensitive to data noise, less on data spatial density as long as $> 50\%$ of domain observed.

Conclusion

- Part I - Build a surrogate model from partial and noisy data:
  - Accurate reproduction of the more energetic (slower) scale and of the unstable Lyapunov exponents
  - Method sensitive to data noise, less on data spatial density as long as $> 50\%$ of domain observed.

- Part II - Build a hybrid model made of a physics-based + data-driven surrogate of the unresolved scales
  - Little sensitive to data noise and spatial density BUT more on data temporal frequency.
  - **No need for the adjoint of the truncated model**.

- **Caveat**: The hybrid model can be expensive to compute (different computing requirement, CPU multiprocessors *vs* GPU).

Conclusion

- Part I - Build a surrogate model from partial and noisy data:
  - Accurate reproduction of the more energetic (slower) scale and of the unstable Lyapunov exponents
  - Method sensitive to data noise, less on data spatial density as long as $> 50\%$ of domain observed.

- Part II - Build a hybrid model made of a physics-based + data-driven surrogate of the unresolved scales
  - Little sensitive to data noise and spatial density BUT more on data temporal frequency.
  - **No need for the adjoint of the truncated model**.

- **Caveat**: The hybrid model can be expensive to compute (different computing requirement, CPU multiprocessors *vs* GPU).

- Future directions:

1. Application to non-autonomous systems
2. Accommodate the estimation of the hybrid model error.

Reference i

Abarbanel, H. D., Rozdeba, P. J., and Shirman, S. (2018).
Machine learning: Deepest learning as statistical data assimilation problems.
Neural computation, 30(8):2025–2055.

Bocquet, M., Brajard, J., Carrassi, A., and Bertino, L. (2019).
Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models.
Nonlinear Processes in Geophysics, 26(3):143–162.

Bocquet, M., Brajard, J., Carrassi, A., and Bertino, L. (2020).
Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization.
Foundations of Data Science, 2(1):55.

Bolton, T. and Zanna, L. (2019).
Applications of deep learning to ocean data inference and subgrid parameterization.
Journal of Advances in Modeling Earth Systems, 11(1):376–399.

Bonavita, M. and Laloyaux, P. (2020).
Machine learning for model error inference and correction.
Journal of Advances in Modeling Earth Systems, page e2020MS002232.

Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L. (2020a).
Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model.
Journal of Computational Science, 44:101171.

Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L. (2020b).
Combining data assimilation and machine learning to infer unresolved scale parametrisation.
in press in Philosophical Transaction A.

Brenowitz, N. D. and Bretherton, C. S. (2018).
Prognostic validation of a neural network unified physics parameterization.
Geophysical Research Letters, 45(12):6289–6298.

De Cruz, L., Demaeyer, J., and Vannitsem, S. (2016).
The modular arbitrary-order ocean-atmosphere model: Maooam v1. 0.
Geoscientific Model Development, 9(8).

Filoche, A., Brajard, J., Charantonis, A., and Béréziat, D. (2020).
Completing physics-based models by learning hidden dynamics through data assimilation.
In *NeurIPS 2020, workshop AI4Earth*, Vancouver (virtual), Canada.

Nguyen, D., Ouala, S., Drumetz, L., and Fablet, R. (2020).
Assimilation-based learning of chaotic dynamical systems from noisy and partial data.
In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3862–3866. IEEE.

O'Gorman, P. A. and Dwyer, J. G. (2018).
Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events.
Journal of Advances in Modeling Earth Systems, 10(10):2548–2563.

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., and Ott, E. (2017).
Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data.
Chaos: An Interdisciplinary Journal of Nonlinear Science, 27(12):121102.

Rasp, S., Pritchard, M. S., and Gentine, P. (2018).
Deep learning to represent subgrid processes in climate models.
Proceedings of the National Academy of Sciences, 115(39):9684–9689.

Reference iii

Tondeur, M., Carrassi, A., Vannitsem, S., and Bocquet, M. (2020).
On temporal scale separation in coupled data assimilation with the ensemble kalman filter.
Journal of Statistical Physics, pages 1–25.

julien.brajard@nersc.no