Combining Data Assimilation and Machine Learning to emulate a numerical model

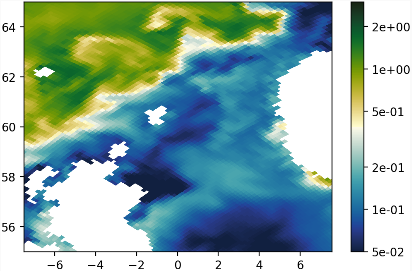
Julien Brajard, Alberto Carrassi, Marc Bocquet, Laurent Bertino

05 June 2019

NERSC, LOCEAN-IPSL-Sorbonne Université, CEREA

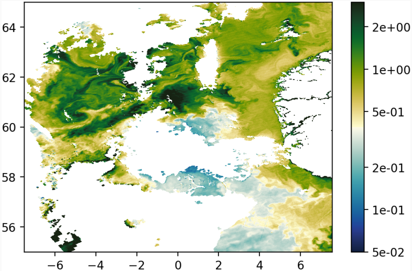
Motivation

Chloropyhll-a (Model)
July 26, 2018



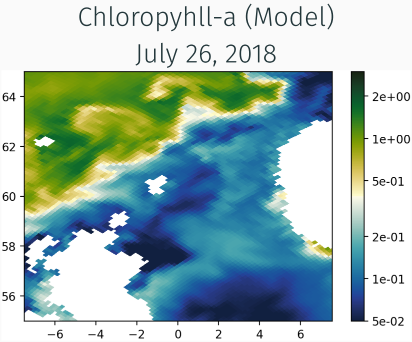
TOPAZ4-ECOSMO forecast

Chloropyhll-a (Observation)
July 26, 2018

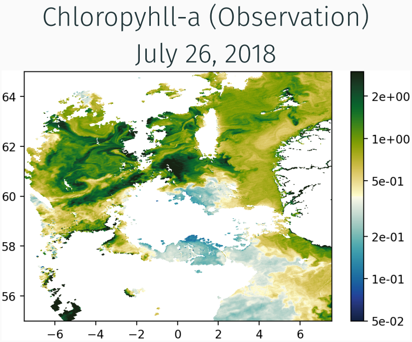


MODIS Aqua

Motivation



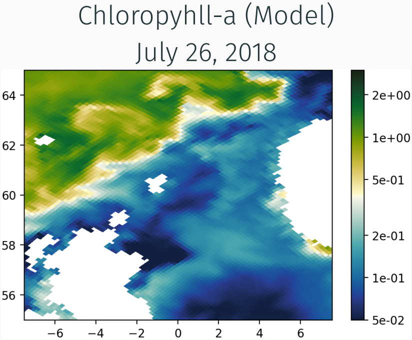
TOPAZ4-ECOSMO forecast



MODIS Aqua

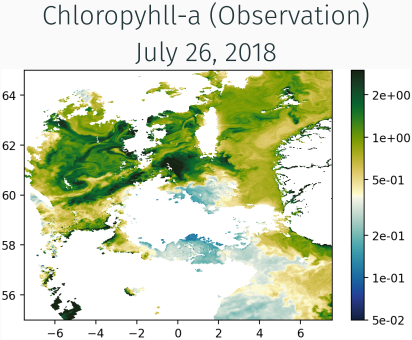
- Unresolved process
- Unknown parameters

Motivation



TOPAZ4-ECOSMO forecast

- Unresolved process
- Unknown parameters

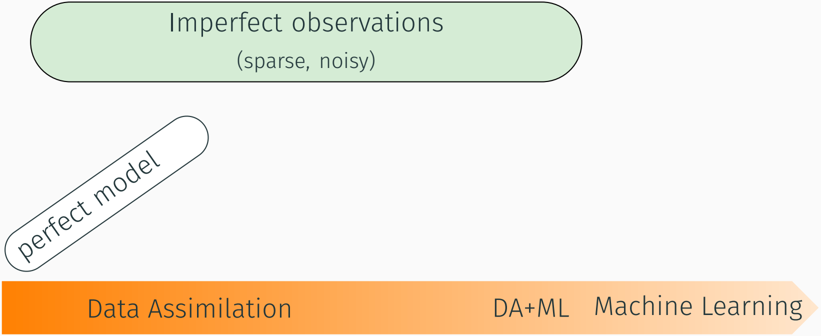


MODIS Aqua

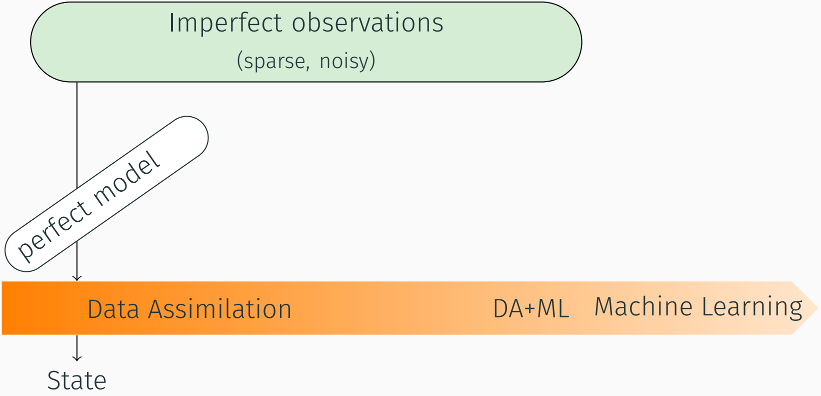
- Sparse
- Noisy

Typology of problems and approaches

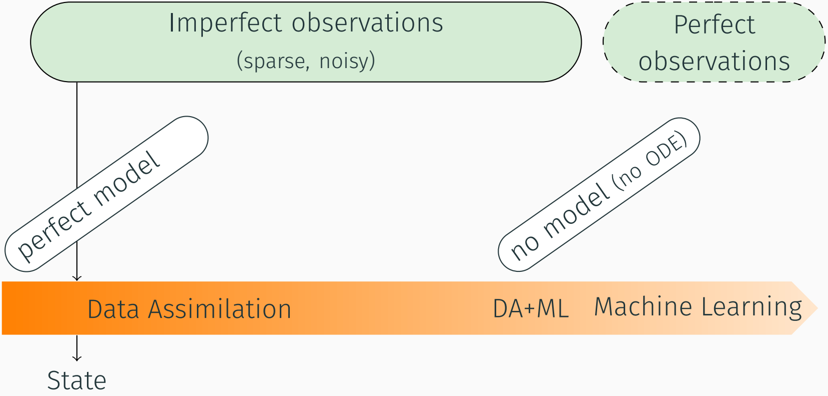
Typology of problems and approaches



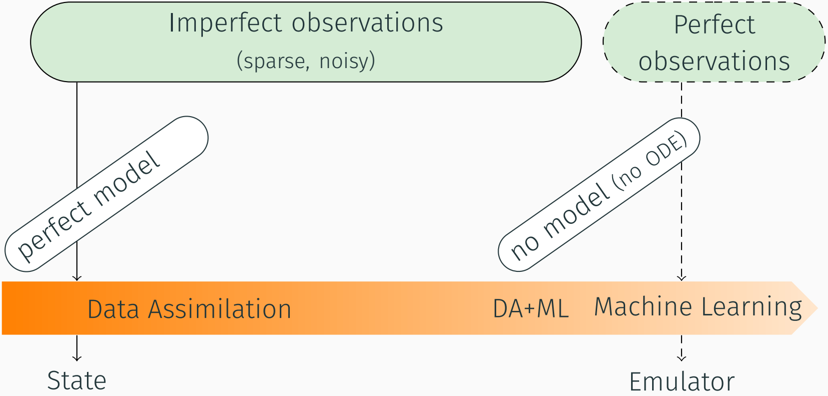
Typology of problems and approaches



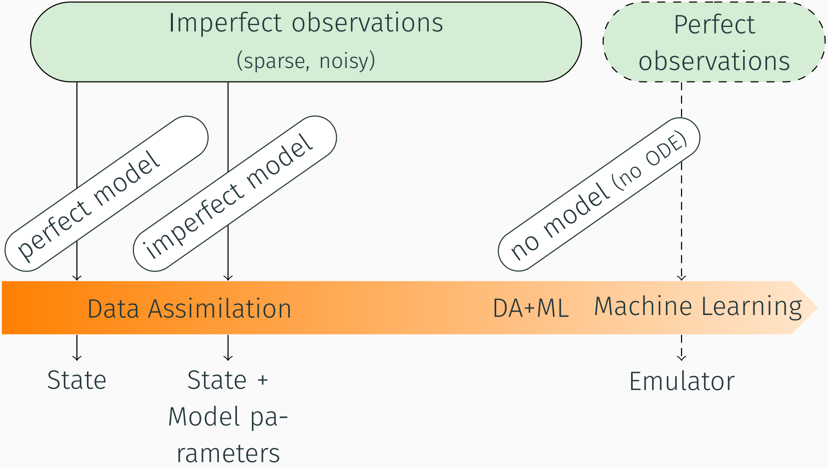
Typology of problems and approaches



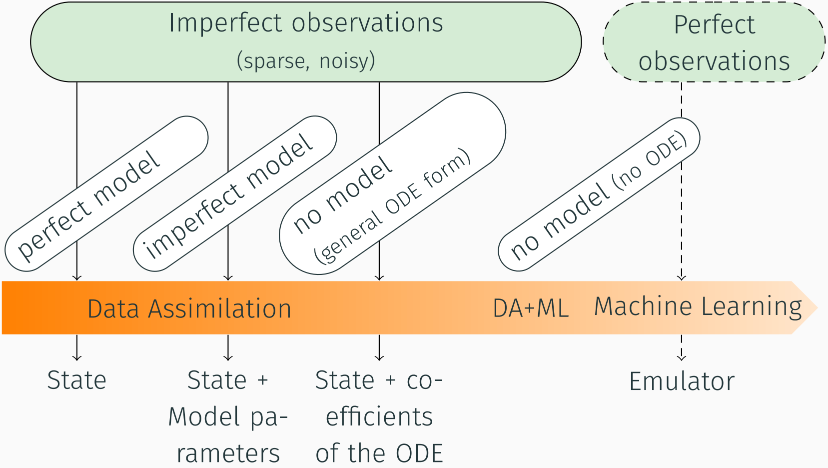
Typology of problems and approaches



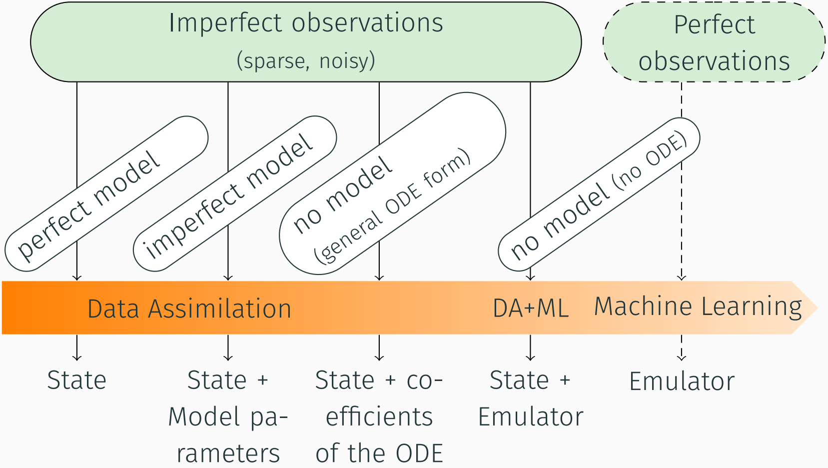
Typology of problems and approaches



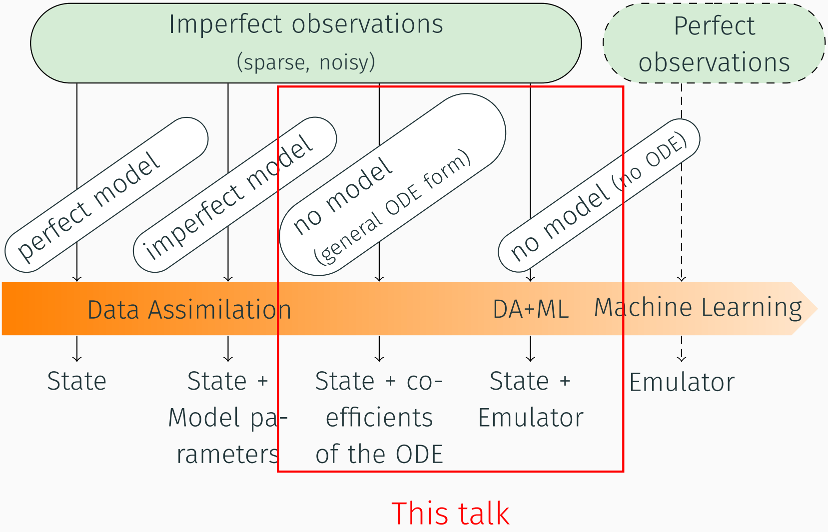
Typology of problems and approaches



Typology of problems and approaches



Typology of problems and approaches



Our Objective:

Producing an accurate and reliable emulator of a numerical model given sparse and noisy observations

Specification of the problem

Data

Multidimensional time series $\mathsf{y}_k^{\mathrm{obs}}$ $(1 \leq k \leq K)$ observed from an underlying dynamical process:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$$

- $\mathcal{H}_k$ is the known observation operator: $\mathbb{R}^{N_x} \to \mathbb{R}^p$
- $\epsilon_k^{\mathrm{obs}}$ is a noise

Specification of the problem

Data

Multidimensional time series $\mathsf{y}_k^{\mathrm{obs}}$ $(1 \leq k \leq K)$ observed from an underlying dynamical process:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$$

- $\mathcal{H}_k$ is the known observation operator: $\mathbb{R}^{N_x} \to \mathbb{R}^p$
- $\epsilon_k^{\mathrm{obs}}$ is a noise

Underlying dynamical model:

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi(\mathsf{x})$$

Specification of the problem

Data

Multidimensional time series $\mathsf{y}_k^{\mathrm{obs}}$ $(1 \le k \le K)$ observed from an underlying dynamical process:

$$\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$$

- $\mathcal{H}_k$ is the known observation operator: $\mathbb{R}^{N_x} \to \mathbb{R}^p$
- $\epsilon_k^{\mathrm{obs}}$ is a noise

Underlying dynamical model:

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi(\mathsf{x})$$

Resolvent:

$$\mathsf{x}_{k+1} = \mathsf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathsf{x})\,\mathrm{d}t,$$

Two complementary goals

1. Inferring the ODE using **only** DA algorithm [Bocquet et al., 2019]:

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi_{\mathsf{A}}(\mathsf{x}), \qquad \phi_{\mathsf{A}}(\mathsf{x}) = \mathsf{A}\mathsf{r}(\mathsf{x}),$$

where $\mathsf{r}(\mathsf{x}) \in \mathbb{R}^{N_p}$ is specified and $\mathsf{A} \in \mathbb{R}^{N_x \times N_p}$ is to be determined.

Two complementary goals

1. Inferring the ODE using **only** DA algorithm [Bocquet et al., 2019]:

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi_{\mathsf{A}}(\mathsf{x}), \qquad \phi_{\mathsf{A}}(\mathsf{x}) = \mathsf{A}\mathsf{r}(\mathsf{x}),$$

   where $\mathsf{r}(\mathsf{x}) \in \mathbb{R}^{N_p}$ is specified and $\mathsf{A} \in \mathbb{R}^{N_x \times N_p}$ is to be determined.

2. Emulation of the resolvent **combining** DA and ML [Brajard et al., 2019]:

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

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

First goal: Inferring the ODE using DA

First goal: ODE representation for the surrogate model

Ordinary differential equations (ODEs) representation of the surrogate dynamics

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi_{\mathsf{A}}(\mathsf{x}), \qquad \phi_{\mathsf{A}}(\mathsf{x}) = \mathsf{A}\mathsf{r}(\mathsf{x}),$$

where

- $\mathsf{A} \in \mathbb{R}^{N_x \times N_p}$ is a matrix of coefficients to be determined.
- $\mathsf{r}(\mathsf{x})$ is a vector of nonlinear regressors of size N_p. For instance, for one-dimensional spatial systems and up to bilinear order:

$$\mathsf{r}(\mathsf{x}) = \left[1, \{x_n\}_{0 \le n < N_x}, \{x_n x_m\}_{0 \le n \le m < N_x}\right].$$

First goal: ODE representation for the surrogate model

Ordinary differential equations (ODEs) representation of the surrogate dynamics

$$\frac{\mathrm{d}\mathsf{x}}{\mathrm{d}t} = \phi_{\mathsf{A}}(\mathsf{x}), \qquad \phi_{\mathsf{A}}(\mathsf{x}) = \mathsf{A}\mathsf{r}(\mathsf{x}),$$

where

- $\mathsf{A} \in \mathbb{R}^{N_x \times N_p}$ is a matrix of coefficients to be determined.
- $\mathsf{r}(\mathsf{x})$ is a vector of nonlinear regressors of size N_p. For instance, for one-dimensional spatial systems and up to bilinear order:

$$\mathsf{r}(\mathsf{x}) = \left[1, \{x_n\}_{0 \leq n < N_x}, \{x_n x_m\}_{0 \leq n \leq m < N_x}\right].$$

$N_p = \binom{N_x+1}{2} = \frac{1}{2}(N_x + 1)(N_x + 2).$

$\longrightarrow$ Intractable in high-dimension! Typically, $N_x = \mathcal{O}(10^{6-9})$.

Reducing the number of regressors

Locality

Physical locality of the physics: all multivariate monomials in the ODEs have variables x_n that belong to a stencil, i.e. a local arrangement of grid points around a given node.

In 1D and with a stencil of size $2L+1$, the size of the dense $\mathbf{A}$ is

$$N_x \times N_\mathrm{a} \quad \text{where} \quad N_\mathrm{a} = \sum_{l=L+1}^{2L+2} l = \frac{3}{2}(L+1)(L+2).$$

Reducing the number of regressors

Locality

Physical locality of the physics: all multivariate monomials in the ODEs have variables x_n that belong to a stencil, i.e. a local arrangement of grid points around a given node.
In 1D and with a stencil of size $2L+1$, the size of the dense $\mathbf{A}$ is

$$N_x \times N_{\mathrm{a}} \quad \text{where} \quad N_{\mathrm{a}} = \sum_{l=L+1}^{2L+2} l = \frac{3}{2}(L+1)(L+2).$$

Homogeneity

Moreover, we can additionally assume translational invariance. In that case $\mathbf{A}$ becomes a vector of size N_{a}.

Bayesian analysis of the problem

Bayesian view on state and model estimation:

$$p(\mathsf{A}, \mathbf{x}_{0:K} | \mathbf{y}_{0:K}) = \frac{p(\mathbf{y}_{0:K} | \mathbf{x}_{0:K}, \mathsf{A}) p(\mathbf{x}_{0:K} | \mathsf{A}) p(\mathsf{A})}{p(\mathbf{y}_{0:K})}.$$

Bayesian analysis of the problem

Bayesian view on state and model estimation:

$$p(\mathbf{A}, \mathbf{x}_{0:K} | \mathbf{y}_{0:K}) = \frac{p(\mathbf{y}_{0:K} | \mathbf{x}_{0:K}, \mathbf{A}) p(\mathbf{x}_{0:K} | \mathbf{A}) p(\mathbf{A})}{p(\mathbf{y}_{0:K})}.$$

Data assimilation cost function assuming Gaussian error statistics and Markovian dynamics:

$$\mathcal{J}(\mathbf{A}, \mathbf{x}_{0:K}) = \frac{1}{2} \sum_{k=0}^{K} \|\mathbf{y}_k - \mathbf{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} + \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{x}_k - \mathbf{F}_{\mathbf{A}}(\mathbf{x}_{k-1})\|^2_{\mathbf{Q}_k^{-1}} - \ln p(\mathbf{x}_0, \mathbf{A}),$$

where $\mathbf{F}_{\mathbf{A}}$ is the resolvent of the model between t_k and $t_k + \Delta_t$.

$\longrightarrow$ Allows to handle partial and noisy observations.

Bayesian analysis of the problem

Bayesian view on state and model estimation:

$$p(\mathbf{A}, \mathbf{x}_{0:K}|\mathbf{y}_{0:K}) = \frac{p(\mathbf{y}_{0:K}|\mathbf{x}_{0:K}, \mathbf{A})p(\mathbf{x}_{0:K}|\mathbf{A})p(\mathbf{A})}{p(\mathbf{y}_{0:K})}.$$

Data assimilation cost function assuming Gaussian error statistics and Markovian dynamics:

$$\mathcal{J}(\mathbf{A}, \mathbf{x}_{0:K}) = \frac{1}{2}\sum_{k=0}^{K} \|\mathbf{y}_k - \mathbf{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} + \frac{1}{2}\sum_{k=1}^{K} \|\mathbf{x}_k - \mathbf{F}_{\mathbf{A}}(\mathbf{x}_{k-1})\|^2_{\mathbf{Q}_k^{-1}} - \ln p(\mathbf{x}_0, \mathbf{A}),$$

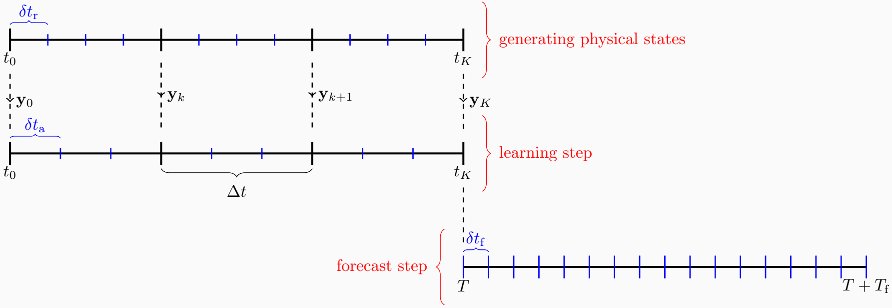
where $\mathbf{F}_{\mathbf{A}}$ is the resolvent of the model between t_k and $t_k + \Delta_t$.

$\longrightarrow$ Allows to handle partial and noisy observations.

Typical machine learning cost function with $\mathbf{H}_k = \mathbf{I}_k$ in the limit $\mathbf{R}_k \longrightarrow \mathbf{0}$:

$$\mathcal{J}(\mathbf{A}) \approx \frac{1}{2}\sum_{k=1}^{K} \|\mathbf{y}_k - \mathbf{F}_{\mathbf{A}}(\mathbf{y}_{k-1})\|^2_{\mathbf{Q}_k^{-1}} - \ln p(\mathbf{y}_0, \mathbf{A}).$$

Experiment setup



Experiment setup

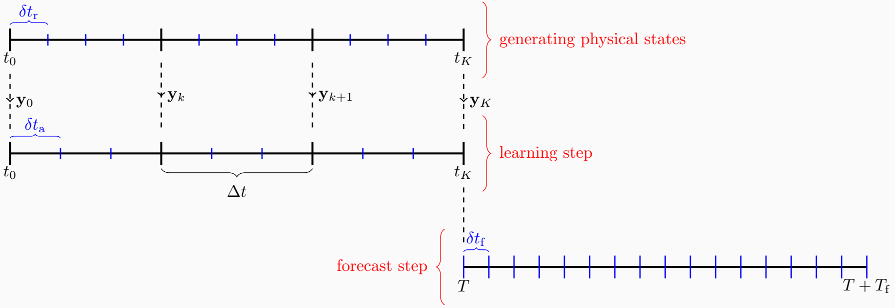


Illustration using a Lorenz 96 model:

- Size of the state $N_x = 40$
- Integration scheme: 4th order RK (RK4)
- Integration time step: $\delta t_r = \Delta t = 0.05$
- integration length: $K = 50$

Case studies

Model	scheme	time step	Observation noise
Identifiable	RK4	$\delta t_a = \Delta t = 0.05$	0
Non identifiable	RK2	$\delta t_a = 0.05/N_c$	0
Identifiable	RK4	$\delta t_a = \Delta t = 0.05$	$\sigma_y > 0$

Identifiable model:

- The true model $\phi(\mathsf{x})$ is included in the candidates $\phi_\mathsf{A}(\mathsf{x})$,
- The integration scheme and the step time used for generating the observations is the same as the one used for the surrogate model.

Case 1: Identifiable model and perfect observations

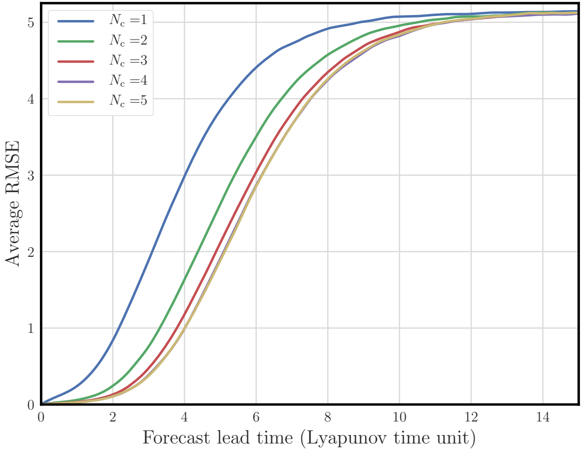
Comparison of the ODE coefficients

$\|\mathsf{A}_{\mathrm{a}} - \mathsf{A}_{\mathrm{r}}\|_{\infty} \sim 10^{-13}$,
where A_{r} are the coefficients of the reference equation (truth) and A_{a} are the coefficients of the surrogate ODE.

Almost perfect reconstruction to the machine precision.

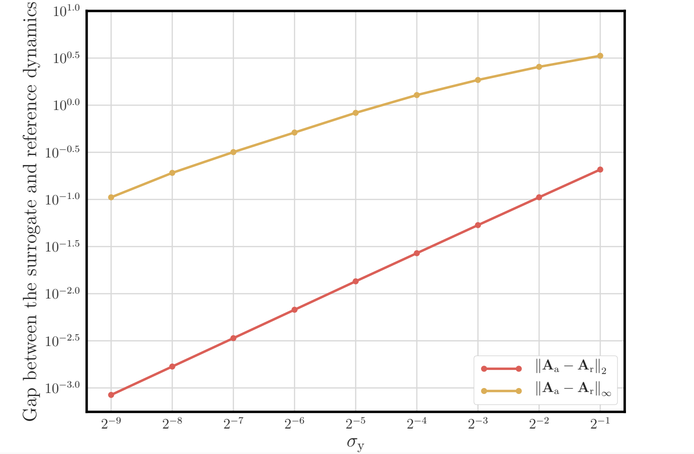
Case 2: Non-identifiable model and perfect observations

Surrogate model based on an RK2 scheme, $\delta t_a = \Delta t / N_c$.
Analysis of the modelling depth as a function of N_c.



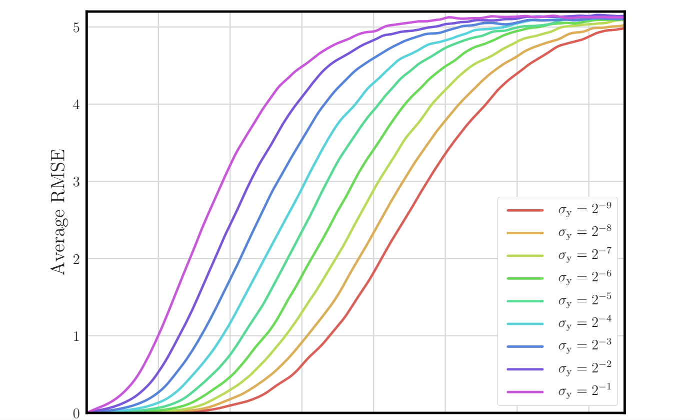
Case 3: Identifiable model and imperfect observations

ODE coefficients



Case 3: Identifiable model and imperfect observations

Forecast skill



Remarks: connections between Data assimilation and machine learning

Data Assimilation	Machine Learning

Remarks: connections between Data assimilation and machine learning

Data Assimilation	Machine Learning
Dynamical system	Residual deep neural network

Remarks: connections between Data assimilation and machine learning

Data Assimilation	Machine Learning
Dynamical system	Residual deep neural network
Parametrized forecasting model	Layer of a neural network

Remarks: connections between Data assimilation and machine learning

Data Assimilation	Machine Learning
Dynamical system	Residual deep neural network
Parametrized forecasting model	Layer of a neural network
Optimization	Training

Remarks: connections between Data assimilation and machine learning

Data Assimilation	Machine Learning
Dynamical system	Residual deep neural network
Parametrized forecasting model	Layer of a neural network
Optimization	Training
Adjoint modelling	Backpropagation

Remarks: connections between Data assimilation and machine learning

Data Assimilation	Machine Learning
Dynamical system	Residual deep neural network
Parametrized forecasting model	Layer of a neural network
Optimization	Training
Adjoint modelling	Backpropagation
Locality assumption	Convolutional layers

Second goal: Emulating a model by combining DA and ML

General remarks

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.

General remarks

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.

General remarks

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.

Idea

Combining both approaches to develop accurate emulator of numerical models.

Proposed algorithm

- Observations: $\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$
- The neural net: $\mathsf{x}_{k+1} = \mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) + \epsilon_k^{\mathrm{m}} = \mathsf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathsf{x})\,\mathrm{d}t$

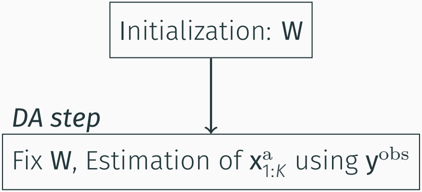
Proposed algorithm

- Observations: $\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$
- The neural net: $\mathsf{x}_{k+1} = \mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) + \epsilon_k^{\mathrm{m}} = \mathsf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathsf{x})\,\mathrm{d}t$

Initialization: W

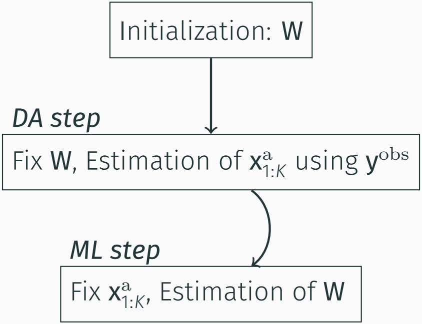
Proposed algorithm

- Observations: $\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$
- The neural net: $\mathsf{x}_{k+1} = \mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) + \epsilon_k^{\mathrm{m}} = \mathsf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathsf{x}) \, \mathrm{d}t$



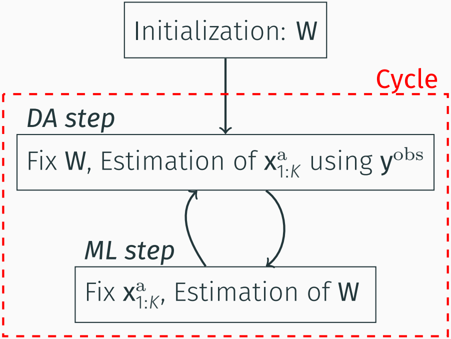
Proposed algorithm

- Observations: $\mathsf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathsf{x}_k) + \epsilon_k^{\mathrm{obs}}$
- The neural net: $\mathsf{x}_{k+1} = \mathcal{G}_{\mathsf{W}}(\mathsf{x}_k) + \epsilon_k^{\mathrm{m}} = \mathsf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathsf{x})\,\mathrm{d}t$



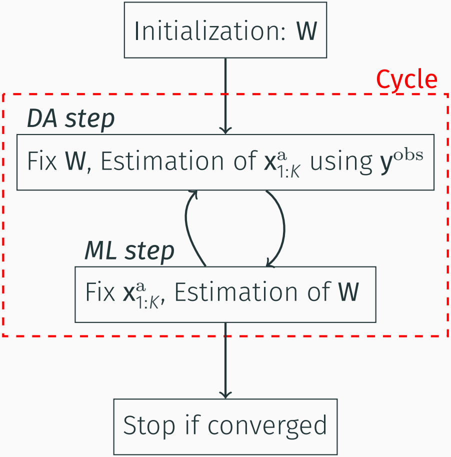
Proposed algorithm

- Observations: $\mathbf{y}_k^{\text{obs}} = \mathcal{H}_k(\mathbf{x}_k) + \epsilon_k^{\text{obs}}$
- The neural net: $\mathbf{x}_{k+1} = \mathcal{G}_{\mathbf{W}}(\mathbf{x}_k) + \epsilon_k^{\text{m}} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathbf{x})\, \mathrm{d}t$



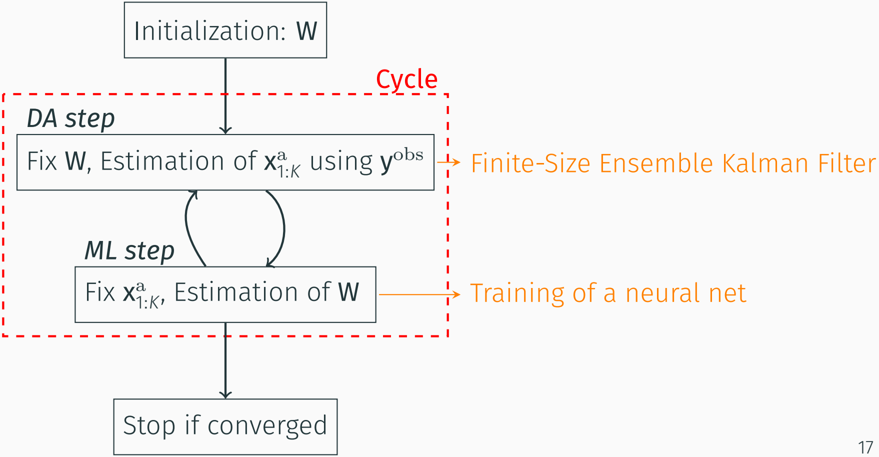
Proposed algorithm

- Observations: $\mathbf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathbf{x}_k) + \epsilon_k^{\mathrm{obs}}$
- The neural net: $\mathbf{x}_{k+1} = \mathcal{G}_{\mathbf{W}}(\mathbf{x}_k) + \epsilon_k^{\mathrm{m}} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathbf{x})\,\mathrm{d}t$



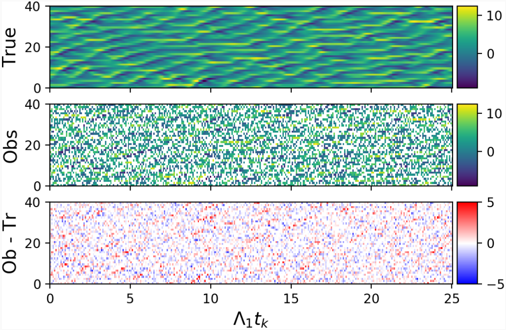
Proposed algorithm

- Observations: $\mathbf{y}_k^{\mathrm{obs}} = \mathcal{H}_k(\mathbf{x}_k) + \epsilon_k^{\mathrm{obs}}$
- The neural net: $\mathbf{x}_{k+1} = \mathcal{G}_{\mathbf{W}}(\mathbf{x}_k) + \epsilon_k^{\mathrm{m}} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \phi(\mathbf{x})\,\mathrm{d}t$



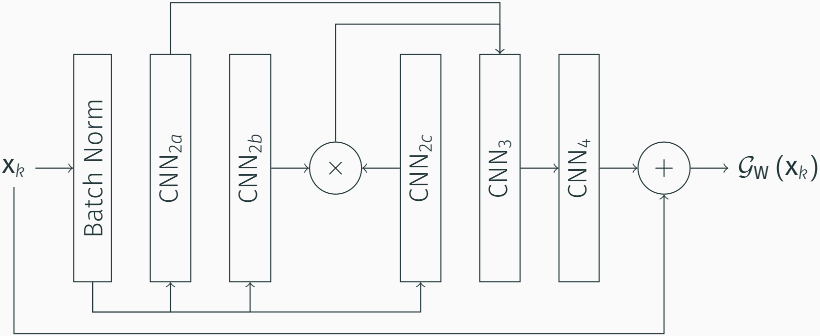
Numerical experiment: Lorenz 96 model

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



- $\mathcal{H}_k$ 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.

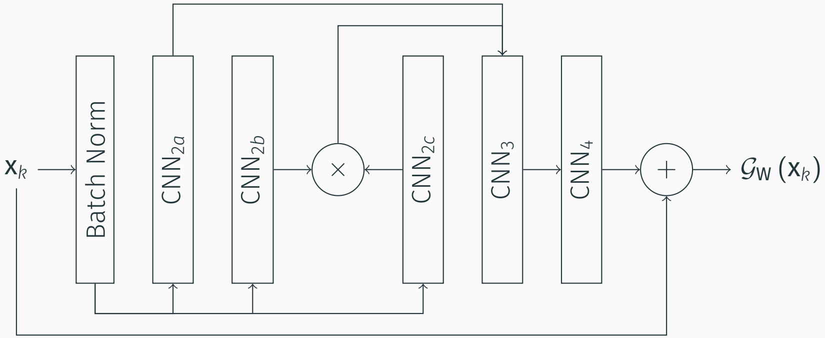
Neural Network setup



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),

Neural Network setup

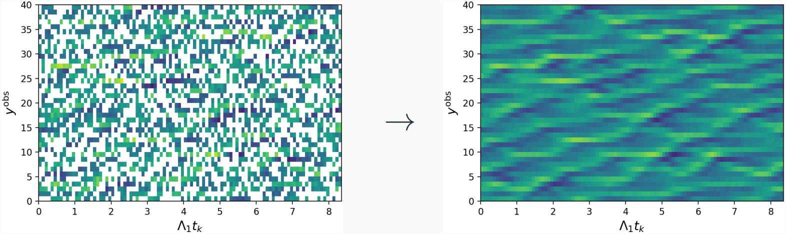


Layer	number of unit	filter size	number of weights
1 (batchnorm)			2
2 (bilinear)	24×3	5	144×3
3 (convolutive)	37	5	8917
4 (linear)	1	1	38

Residual bi-linear convolutive neural network (9391 weights), compared with $N_\mathrm{a} = 18$ in case of ODE parametrization.

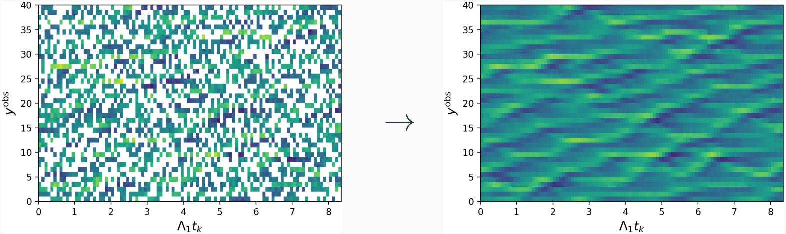
Evaluation

- Interpolating the observations:



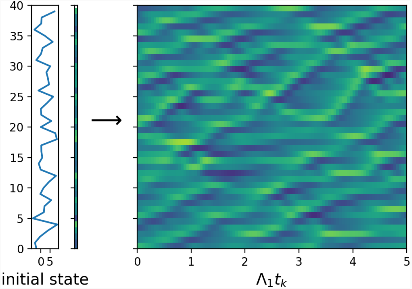
Evaluation

- Interpolating the observations:
  Score: RMSE-a (Root-mean square error of the analysis)



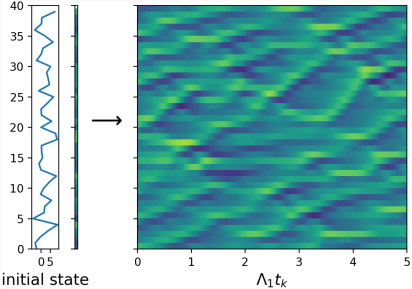
Evaluation

- Interpolating the observations:
  Score: RMSE-a (Root-mean square error of the analysis)
- Forecasting skill



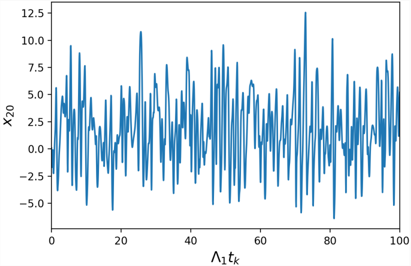
Evaluation

- Interpolating the observations:
  Score: RMSE-a (Root-mean square error of the analysis)
- Forecasting skill
  Score: RMSE-f (Root-mean square error of the forecast as a function of leading time)



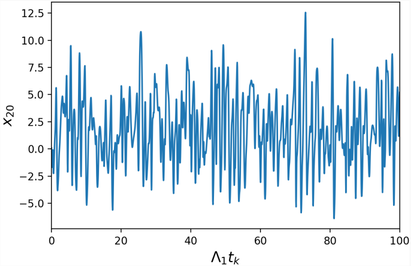
Evaluation

- Interpolating the observations:
  Score: RMSE-a (Root-mean square error of the analysis)
- Forecasting skill
  Score: RMSE-f (Root-mean square error of the forecast as a function of leading time)
- Reproducing the long-term dynamics

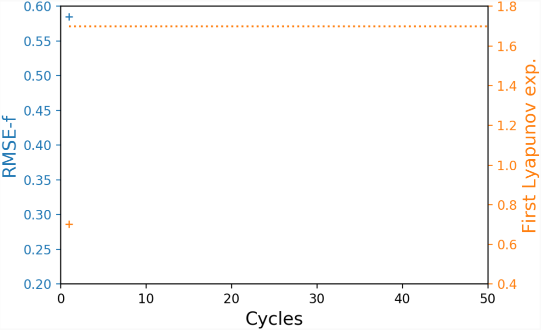


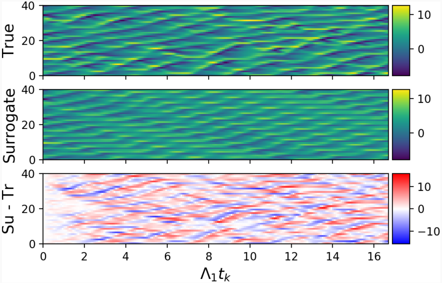
Evaluation

- Interpolating the observations:
  Score: RMSE-a (Root-mean square error of the analysis)
- Forecasting skill
  Score: RMSE-f (Root-mean square error of the forecast as a function of leading time)
- Reproducing the long-term dynamics
  Score: Lyapunov exponents and PSD (Power Spectral Density) compared with the true model.

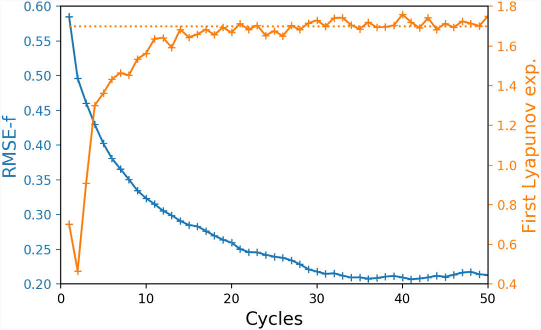


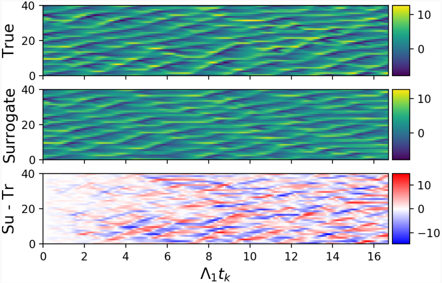
Convergence of the algorithm



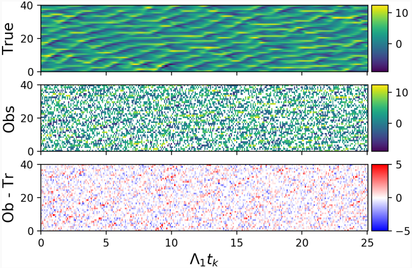


Convergence of the algorithm



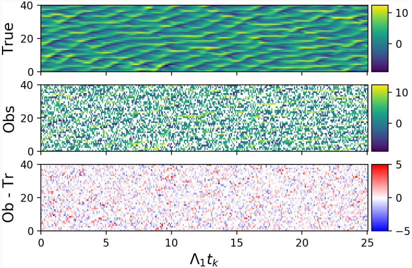


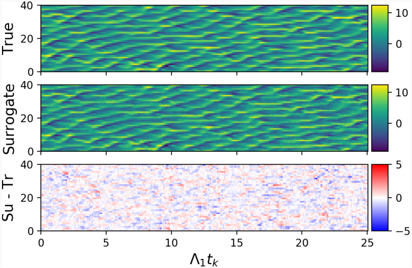
Interpolation



RMSE (obs)= 1

Interpolation

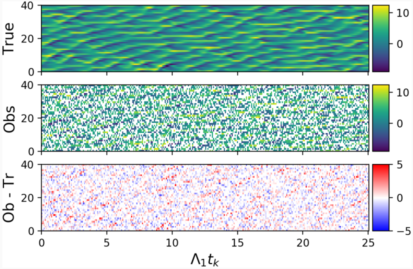




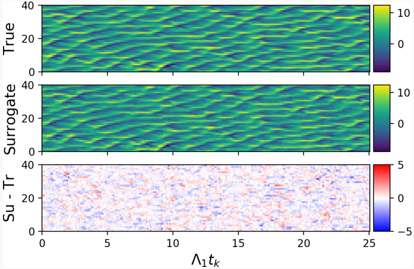
RMSE (obs)= 1

RMSE-a= 0.8

Method	RMSE-a
DA with surrogate model	0.80



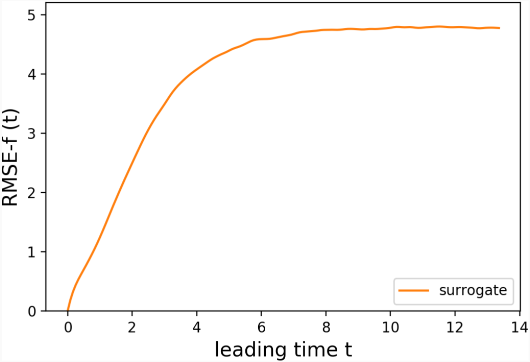
RMSE (obs)= 1



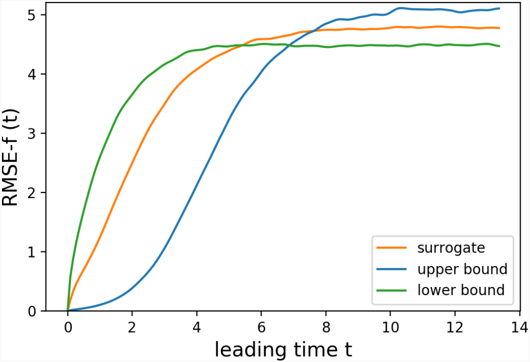
RMSE-a= 0.8

	Method	RMSE-a
Lower bound	Quadratic interpolation	2.32
	DA with surrogate model	0.80
Upper bound	DA with true model	0.34

Forecast skill



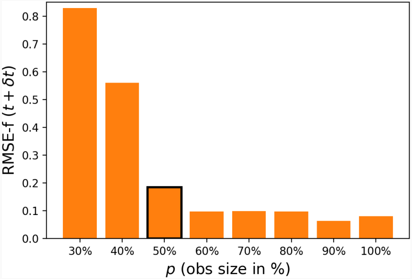
Forecast skill



- **Lower bound**: Neural Net trained with observation interpolated using quadratic interpolation (no data assimilation).
- **Upper bound**: Neural Net trained with "perfect" observations (complete, no noise).

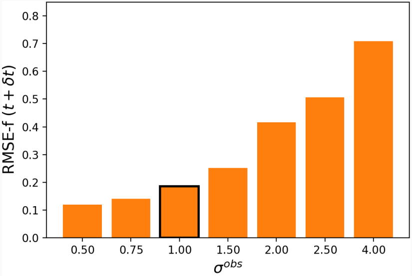
Sensitivity to noise and density of observations

Sensititvity to the density of observations



RMSE-f$(t_0 + \delta t)$
observation noise: $\sigma^{obs} = 1$

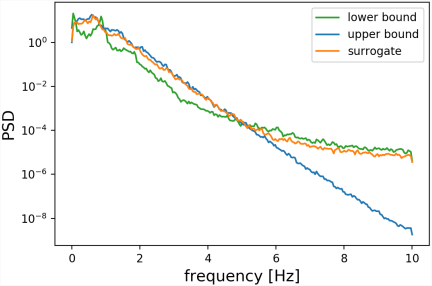
Sensititvity to the noise of observations



RMSE-f$(t_0 + \delta t)$
density of observations: 50%

Reconstruction of the long-term dynamics

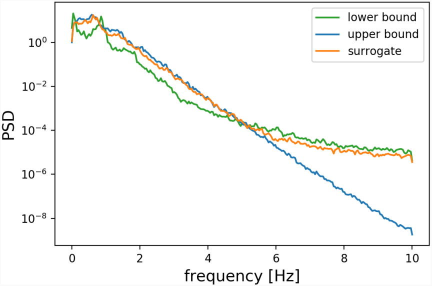
Power spectral density



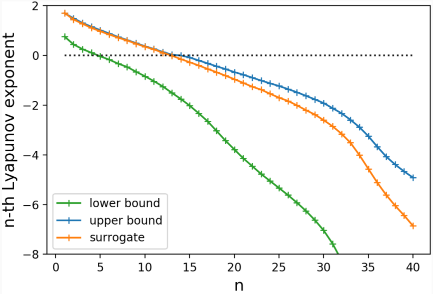
- Lower bound: Neural Net trained with observation interpolated using quadratic interpolation (no data assimilation).
- Upper bound: True model

Reconstruction of the long-term dynamics

Power spectral density



Lyapunov exponents



- Lower bound: Neural Net trained with observation interpolated using quadratic interpolation (no data assimilation).
- Upper bound: True model

Conclusion

Emulate an numerical model given sparse and noisy observations

- Bayesian data assimilation for state and model estimation:
  - equivalent to a machine learning approach,
  - makes use of locality and homogeneity to reduce the dimension of the model parameters.

Conclusion

Emulate an numerical model given sparse and noisy observations

- Bayesian data assimilation for state and model estimation:
  - equivalent to a machine learning approach,
  - makes use of locality and homogeneity to reduce the dimension of the model parameters.
- Combined data assimilation / machine learning:
  - emulate the resolvent of the model,
  - training of the neural nets are performed on state estimated from data assimilation.

Conclusion

Emulate an numerical model given sparse and noisy observations

- Bayesian data assimilation for state and model estimation:
  - equivalent to a machine learning approach,
  - makes use of locality and homogeneity to reduce the dimension of the model parameters.
- Combined data assimilation / machine learning:
  - emulate the resolvent of the model,
  - training of the neural nets are performed on state estimated from data assimilation.

Properties of the neural net surrogate model

- **Interpolation of the observations**: denoising of the observations and interpolation
- **Predictability skills**: sensitive to model noise, and to observation density below 50%
- **Replication of the long-term dynamics properties**

Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino.
Data assimilation as a deep learning tool to infer ODE representations of dynamical models.
Nonlinear Processes in Geophysics Discussions, pages 1–29, 2019.
URL: `https://doi.org/10.5194/npg-2019-7`, `doi:10.5194/npg-2019-7`.

J. Brajard, A. Carrassi, M. Bocquet, and 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.
Geoscientific Model Development Discussions, 2019:1–21, 2019.
URL: `https://www.geosci-model-dev-discuss.net/gmd-2019-136/`, `doi:10.5194/gmd-2019-136`.

julien.brajard@sorbonne-universite.fr, julien.brajard@nersc.no

Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino.
Data assimilation as a deep learning tool to infer ODE representations of dynamical models.
Nonlinear Processes in Geophysics Discussions, pages 1–29, 2019.
URL: `https://doi.org/10.5194/npg-2019-7`, `doi:10.5194/npg-2019-7`.

J. Brajard, A. Carrassi, M. Bocquet, and 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.
Geoscientific Model Development Discussions, 2019:1–21, 2019.
URL: `https://www.geosci-model-dev-discuss.net/gmd-2019-136/`, `doi:10.5194/gmd-2019-136`.

julien.brajard@sorbonne-universite.fr, julien.brajard@nersc.no

---

Call for papers

2-4 October 2019

`https://sites.google.com/view/climateinformatics2019`