Connecting Data Assimilation and Neural ODEs

Arata Amemiya[1], Noboru Isobe[2], Takemasa Miyoshi[1]

1. RIKEN Center for Computational Science
2. The University of Tokyo

Overview

- The basics of Neural ODEs and similarity with data assimilation methods
- ML-based approach for model bias correction
- Discussion : possible use of Neural ODEs

Neural ODE

An approximate map $\boldsymbol{x} \to \boldsymbol{y}$ with **continuous dynamics** of hidden units (Chen et al., 2018)

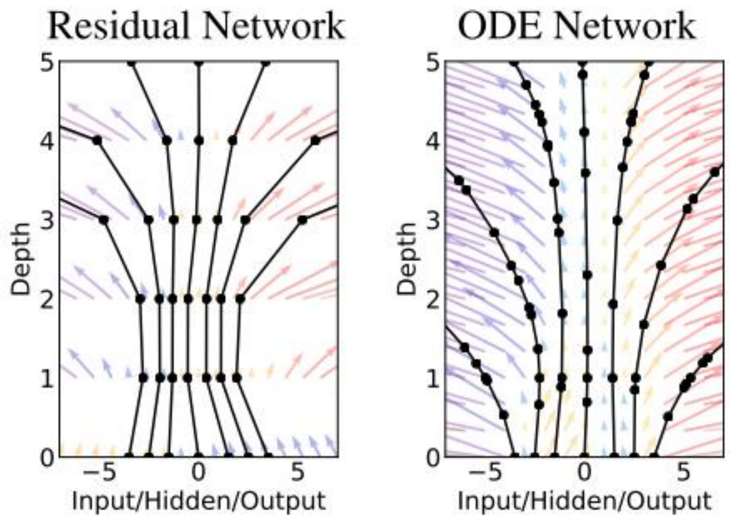
Maps $\ell_\theta^1, \ell_\theta^2$ and a function f_θ :
specified by neural networks

$$z_0 = \ell_\theta^2(x)$$
$$z_t = z_0 + \int_0^t f_\theta(z_s)ds$$
$$y = \ell_\theta^1(z_T)$$

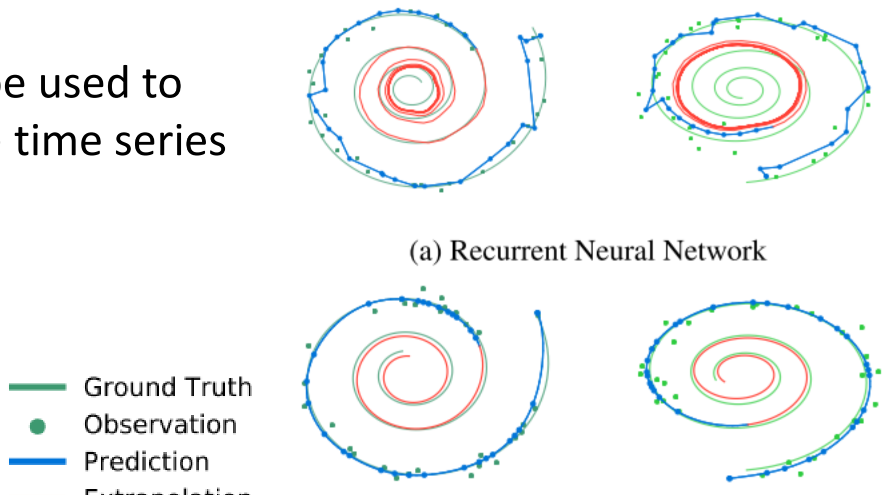
$\longleftrightarrow$

Residual Neural Networks :
Discrete dynamics

$$z_{t+1} = z_t + f(z_t, \theta_t)$$



Neural ODEs can be used to
fit and extrapolate time series



(a) Recurrent Neural Network

(b) Latent Neural Ordinary Differential Equation

Training in Neural ODE

Loss function depends on all z over the integration period

$$\boxed{L(z(t_1)) = L\left(z_0 + \int_{t_0}^{t_1} f(z(t), t, \theta)dt\right)}$$

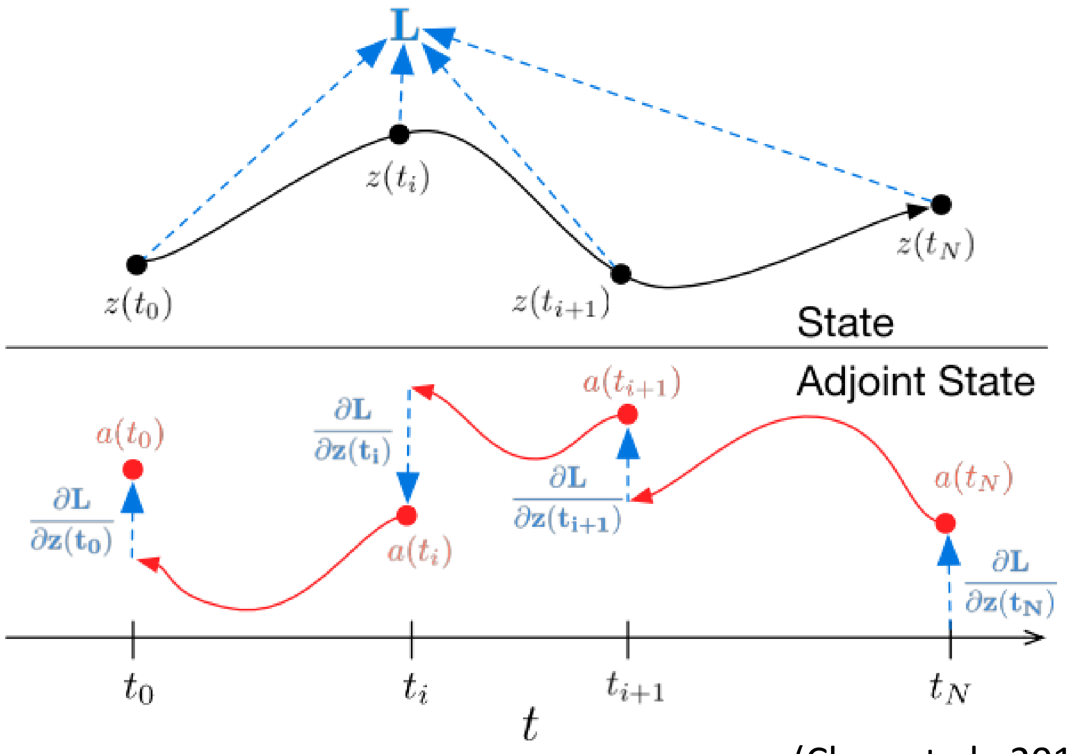
To obtain $\partial L/\partial z_0$ and $\partial L/\partial \theta$,
"**adjoint**" $\mathbf{a}(t) \equiv \partial L/\partial \mathbf{z}(t)$ is used

$\mathbf{a}(t)$ is obtained by backward integration of the ODE starting from $\mathbf{a}(t_1) = \partial L/\partial \mathbf{z}(t_1)$

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

$\partial L/\partial \theta$ can be obtained by another integration

$$\frac{d}{dt}\frac{\partial L}{\partial \theta} = -\mathbf{a}(t)^T \frac{\partial f(z(t), t, \theta)}{\partial \theta}$$



(Chen et al., 2018)

* Backward integration is separated into periods between each pair of observation time

* The implementation is straightforward with standard ODE solvers

Similarity with adjoint method in data assimilation

Loss function in 4-dimensional variational method (4D-Var)

$$\boxed{L = \frac{1}{2}\left[\left(\boldsymbol{x}_0 - \boldsymbol{x}^{\boldsymbol{b}}\right)^T \boldsymbol{B}^{-1}\left(\boldsymbol{x}_0 - \boldsymbol{x}^b\right) + \left(H(\boldsymbol{x}_n) - \boldsymbol{y}_n\right)^T \boldsymbol{R}^{-1}\left(H(\boldsymbol{x}_n) - \boldsymbol{y}_n\right)\right]}$$

x: model state

y: observation

$\boldsymbol{B}$: background error covariance

$\boldsymbol{R}$: observation error covariance

H: observation operator

$\boldsymbol{H}$: tangent linear operator of H

Model equation

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x})$$

Forward and adjoint operator

$$\boldsymbol{x}_{k+1} = \boldsymbol{\mathcal{M}}(\boldsymbol{x}_k) = \boldsymbol{x}_k + \int_{t_k}^{t_{k+1}} f\left(x(t)\right)dt$$

$$\boldsymbol{M}_k^{\boldsymbol{T}} = \left(\frac{\partial \boldsymbol{\mathcal{M}}}{\partial \boldsymbol{x}}\right)^T$$

$$\frac{\partial L}{\partial \boldsymbol{x}_k} = \left(\frac{\partial \boldsymbol{x}_{k+1}}{\partial \boldsymbol{x}_k}\right)^T \frac{\partial L}{\partial \boldsymbol{x}_{k+1}} = \boldsymbol{M}_k^{\boldsymbol{T}} \frac{\partial L}{\partial \boldsymbol{x}_{k+1}} \longleftrightarrow$$

Continuous adjoint equation

$$\boxed{\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f(\boldsymbol{z}(t), t, \theta)}{\partial \boldsymbol{z}}}$$

Optimization of the initial state x_0 given the observation y_n uses $\partial L/\partial x_0$

$$\frac{\partial L}{\partial \boldsymbol{x}_0} = \boldsymbol{B}^{-1}\left(\boldsymbol{x}_0 - \boldsymbol{x}^b\right) + \boldsymbol{M}_0^{\boldsymbol{T}}\, \boldsymbol{M}_1^{\boldsymbol{T}}\, \boldsymbol{M}_2^{\boldsymbol{T}} \dots\, \boldsymbol{M}_{n-1}^{\boldsymbol{T}} \boldsymbol{H}_n^{\boldsymbol{T}} \boldsymbol{R}^{-1}\left(H(\boldsymbol{x}_n) - \boldsymbol{y}_n\right)$$

Extension of Neural ODE : learning from time series

Neural ODE : Map from $\boldsymbol{x} \to \boldsymbol{y}$ $(\boldsymbol{z}(t_0) \to \boldsymbol{z}(t_1)$ in hidden state)

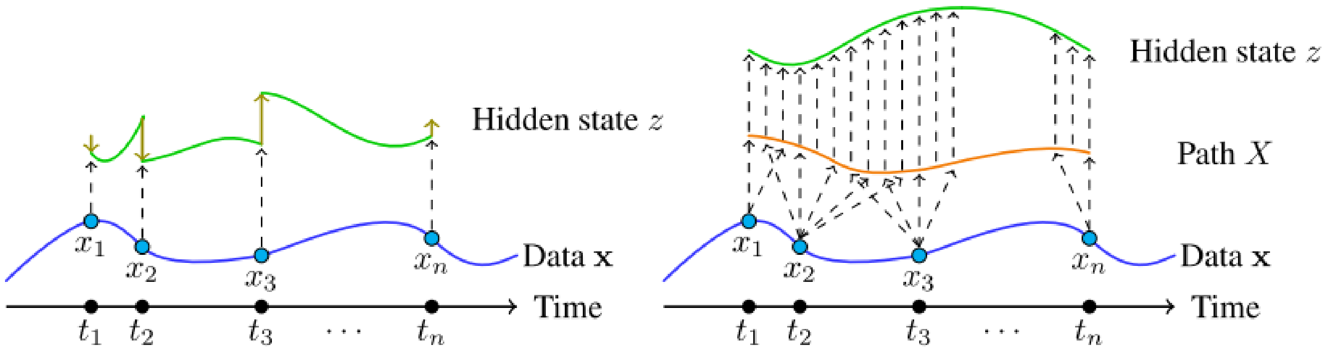
- Equivalent to neural networks

Neural controlled differential equation (Neural CDE) : Map from $\boldsymbol{z}(t_0, t_1, \dots t_{N-1}) \to \boldsymbol{z}(t_N)$

- Equivalent to recurrent neural networks

$$z_t = z_0 + \int_0^t f_\theta(z_s) dX_s$$
$$= \int_0^t f_\theta(z_s) \frac{dX(s)}{ds} ds$$

(“path” X is the interpolation of the data series $x_0, x_1, \dots$)



(Kidger et al., 2020)

The further extension of Neural CDE using rough-path theory :
Neural Rough Differential Equations (Neural RDE) (Morrill et al., 2021)

Model bias correction problem

"Model" in atmosphere/ocean study = (mostly) Knowledge-based model

Components :

- Dynamics
- Moist convection
- Small-scale topography
- Turbulence
- Cloud microphysics
- etc.

Model equations are always imperfect
→ hybrid modeling approach may improve the accuracy

$$\frac{dX}{dt} = f(X) + g(X) + \varepsilon$$

f : Knowledge-based model

g : Data-driven model

Hybrid modeling of spatio-temporal chaos

Pathak et al. (2018)

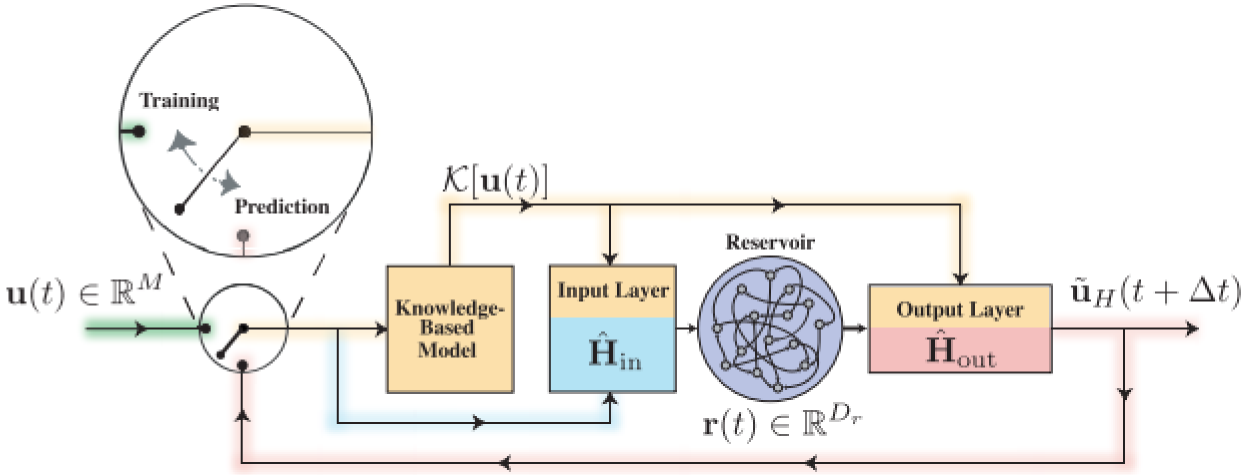
Hybrid modeling by combining
an imperfect knowledge-based model and a ML-based (Reservoir computing) model

Pure ML-based model :

$$X_{t+1} = \mathcal{G}(X_t, \ldots)$$

Hybrid model :

$$X_{t+1} = \mathcal{G}(X_t, \mathcal{F}(X_t), \ldots)$$



Hybrid modeling was demonstrated
on Lorenz63 and Kuramoto-Sivashinsky model experiment

Training data : **True time series** generated by $X_{t+1} = \mathcal{F}_{\text{true}}(X_t)$

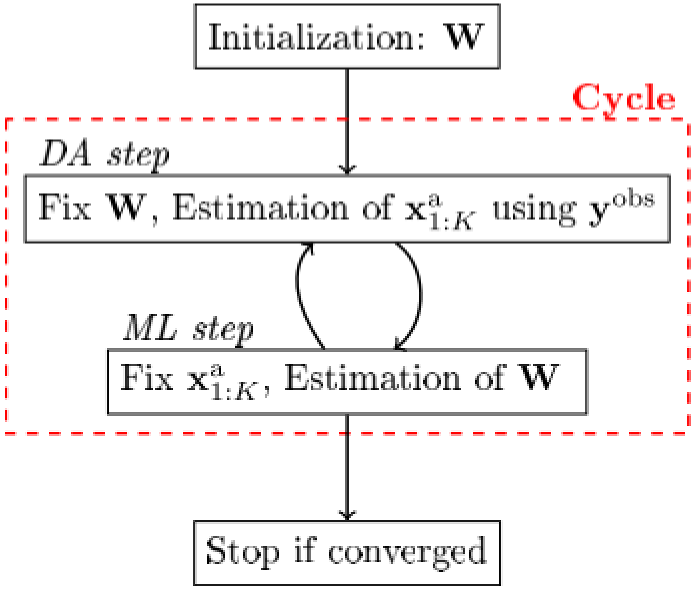
⇔ partial, noisy and temporally sparse observation in real case

ML-based prediction using data assimilation

ML-based prediction based on **noisy and irregularly sampled** y^{obs} by the combination with DA

given $\mathcal{G}(x), y^{\mathrm{obs}}(0 \dots T)$
estimate $x^{\mathrm{a}}(0 \dots T)$

given $x^{\mathrm{a}}(0 \dots T)$
estimate $\mathcal{G}(x)$



(Brajard et al. 2019)

DA step

$$x^f_{k+1} = \mathcal{G}(x^a_k) \qquad \xrightarrow{\text{DA}} \qquad x^a_{k+1}$$
$$y^{\mathrm{obs}}$$

ML step

$$\mathbf{x}_{k+1} = \underbrace{\mathcal{G}(\mathbf{x}_k)}_{\text{ML model}} + \epsilon^{\mathrm{m}}_k = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathcal{M}(\mathbf{x})\,\mathrm{d}t,$$

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

※Loss function is weighted by error covariance matrix P^a_k obtained by DA step

Use of ML for model bias correction

Brajard et al. (2019) : ML model for the whole forecast function

$$x_{k+1} = \mathcal{G}(x_k)$$

ML model for correcting the knowledge-based model $f(x)$

$$x_{k+1} = x_k + \int_k^{k+1} f(x(t))dt + \mathcal{G}(x_k) \qquad \text{or} \qquad x_{k+1} = \mathcal{G}\left(x_k + \int_k^{k+1} f\big(x(t)\big)dt \;, x_k\right)$$

Estimation of bias correction term $\mathcal{G}(x_k)$ from forecasts x_k^f and analyses x_k^a has been demonstrated with various modeling methods

- Danforth and Kalnay (2008) : Reduced-order linear function using Singular Value Decomposition
- Bonavita and Laloyaux (2020) : Neural Network using analyses obtained by 4D-Var

Previous experiment

Nature run : Lorenz96 + additional term

$$\frac{d}{dt}x_k = x_{k-1}(x_{k+1} - x_{k-2}) - x_k + F + \underline{f_{\mathrm{add}}(\boldsymbol{x})}$$

additional term (= negative model bias)

$$f_{\mathrm{add}}(\boldsymbol{x}) = 0.2x_{k-1}(x_{k+1} - x_{k-2})$$

Observation : $\Delta t = 0.05$

(Amemiya et al., *in prep.*)

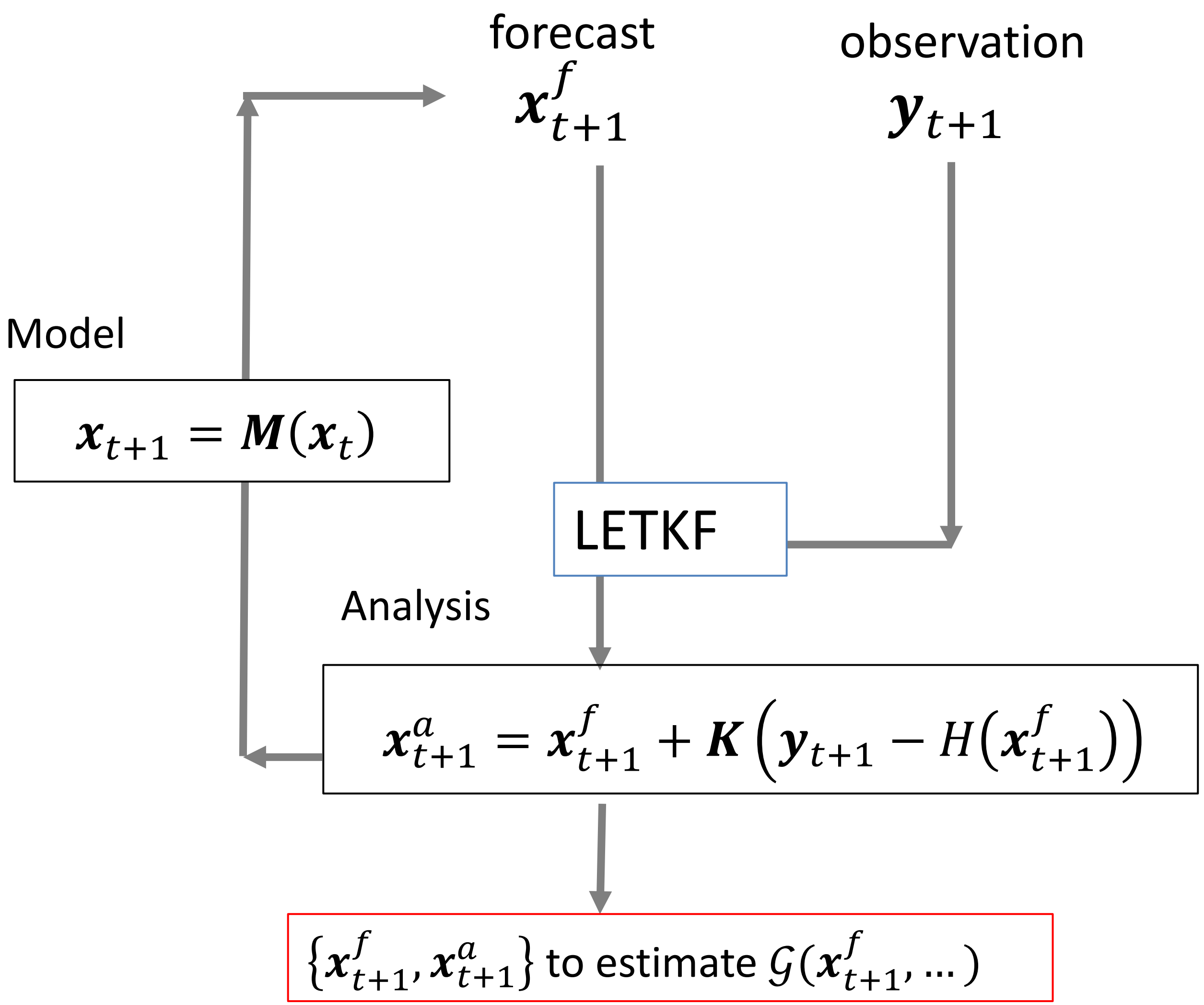
$$y_k = x_k + \epsilon \qquad \epsilon \sim N\,(0, R)$$

(Imperfect) Forecast model

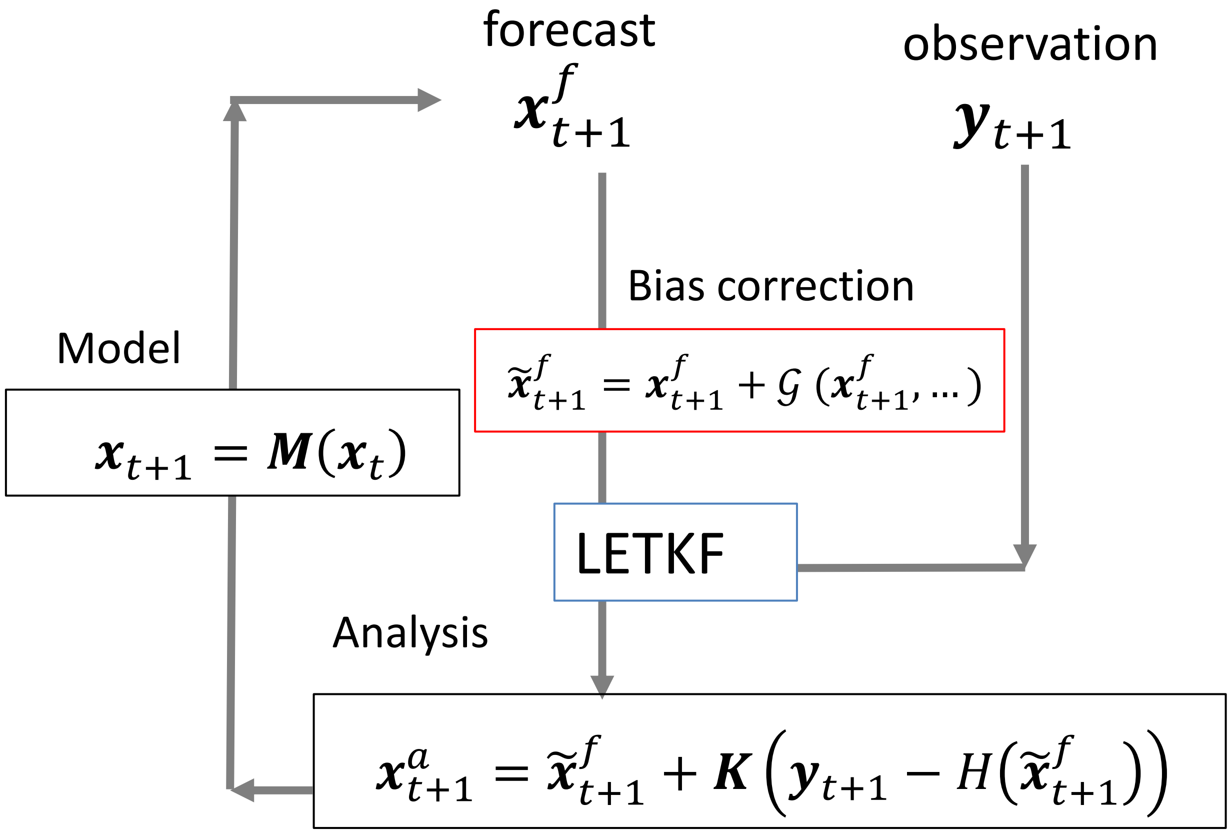
$$\frac{d}{dt}x_k = x_{k-1}(x_{k+1} - x_{k-2}) - x_k + F$$

Implementation

Step 1: Estimation of bias correction function



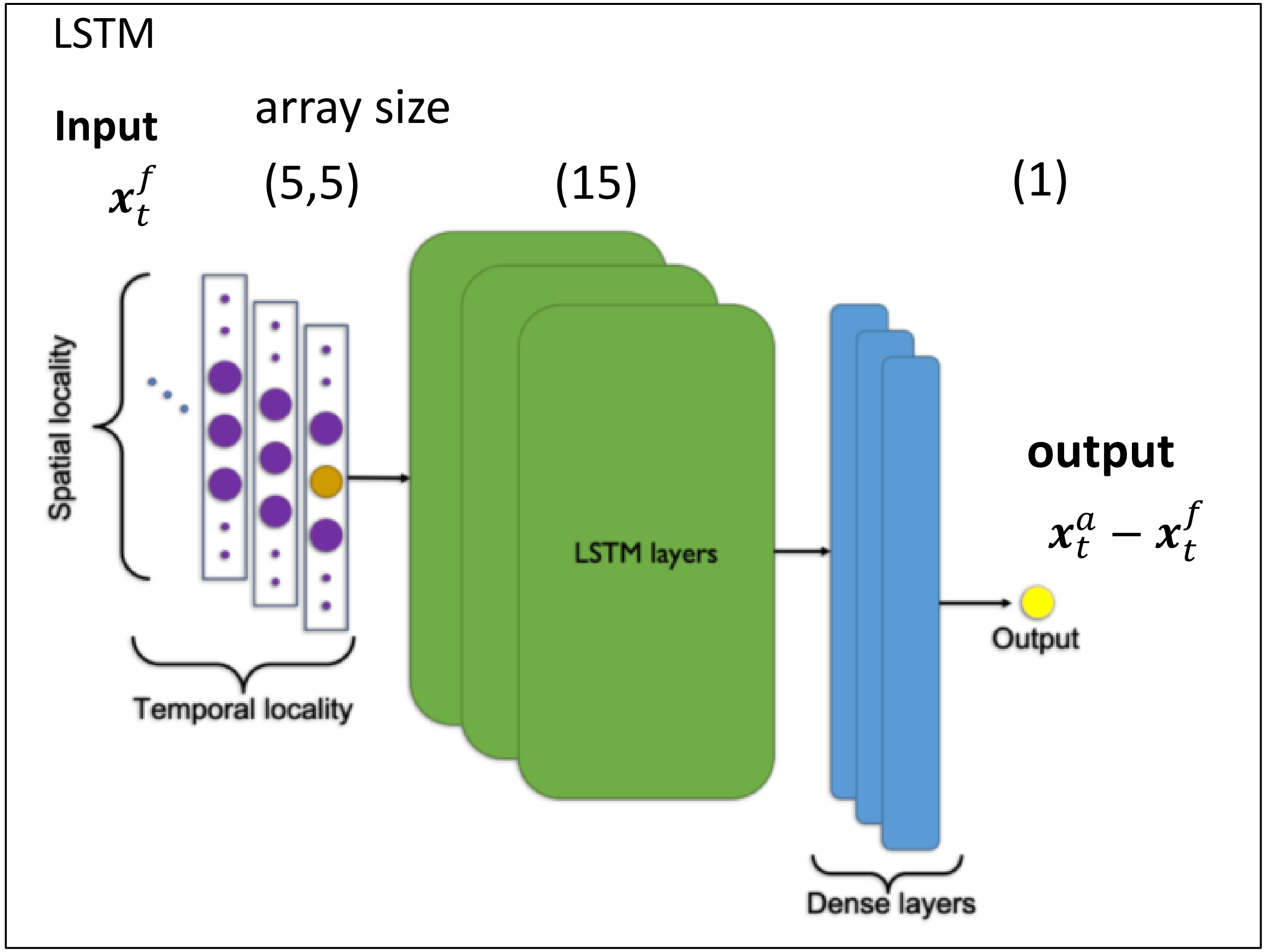
Step 2: Data assimilation with corrected forecasts

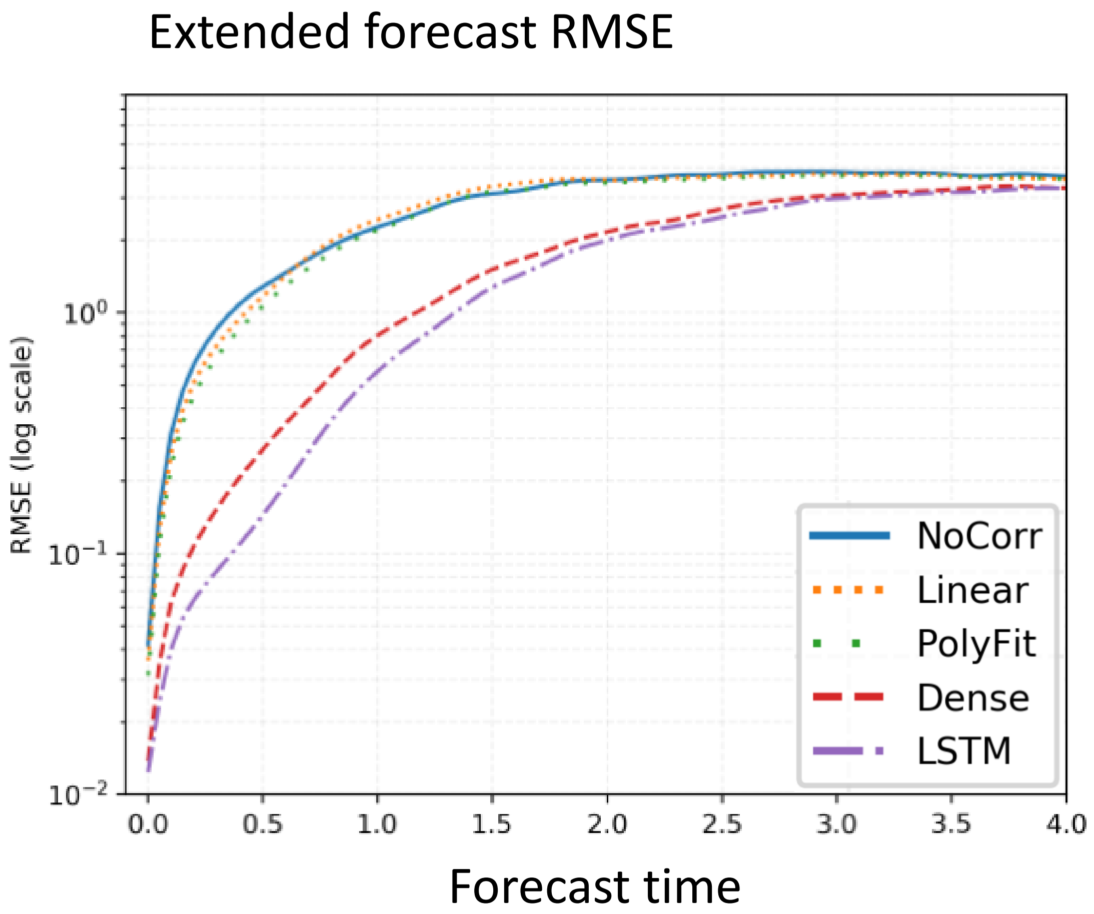


Bias correction methods

Comparison of different models for $\mathcal{G}(\boldsymbol{x}^f_{t+1}, \dots)$

- Polynomial regression
- Neural Network
- LSTM





The limitation of Neural Network model

Observation and analysis time interval $\Delta t \gg$ Numerical integration dt

↓

Bias correction term may not simply correspond to $f_{\mathrm{add}}(\boldsymbol{x})$

1. Direct addition

$$x_{k+1} = \int_k^{k+1} f(x(t))dt + \mathcal{G}(x_k)$$

- forecast time series have jumps every Δt

2. Linear approximation

$$x_{k+1} = \int_k^{k+1} \left[f(x(t)) + \frac{1}{\Delta t} \mathcal{G}(x_k) \right] dt$$

- large error when model state rapidly evolves during Δt (e.g. weather radar)

→ what if we can directly model the tendency term?

Discussion: the use of Neural ODEs

Model bias correction with data assimilation :
training with the analysis and forecast $\{\boldsymbol{x}^f, \boldsymbol{x}^a\}$

- Linear regression and neural networks
  → estimate the correction term every Δt

- Neural ODE
  → estimate the possible missing term directly

$$x_{k+1} = \int_k^{k+1} \left(\underbrace{f\big(x(t)\big)}_{\text{known model equation}} + \underbrace{g\big(x(t)\big)}_{\text{unknown missing term}} \right) dt$$

known model equation

unknown missing term
= Neural ODES trained with $\{\boldsymbol{x}^f, \boldsymbol{x}^a\}$