Reduced order modeling of dynamical systems using ANNs for water circulation

CFDML 2020

Alberto C. Nogueira Jr., João L. S. Almeida, Guillaume Auger, and Campbell D. Watson

June 19, 2020

Outline

1. Motivation
2. Goal and problem description
3. Approach
4. Reduced order modeling
5. Validation example: Lorenz discrete system
6. Hydrodynamics example: FCNN ROM and LSTM ROM
7. Conclusions

Motivation

- General **circulation models** are essential tools in **weather forecasting** and **environmental sciences**.
- **Oil spill** monitoring, **algae bloom** prediction and detection of **freshwater contamination** by **salt** are just a few examples of harmful events that can be addressed by circulation models.
- These models solve **discretized complex physical equations**, in order to compute the **evolution of circulation states** over time.
- High resolution **numerical solutions** of such models are extremely **computational** and **time consuming**.
- **ML** based **low-dimensional surrogate models** are a promising alternative to **speed up** circulation forecasts **without undermining the quality** of predictions.

Goal and problem description

- **Goal:** Develop **fast** and **accurate low-dimensional surrogate** models to produce **36 hours** depth-averaged hydrodynamics **forecasts** of a freshwater lake.
- **Application:** Prevent a freshwater basin contamination by salt and harmful biological agents using water circulation prediction.
- **Scenario:** A complex dynamical system consisting of a realistic geometric representation of Lake George (LG), in NY, together with challenging atmospheric forcing terms.

Approach

ANN-based reduced order modeling

- Reduce the dimensionality of the available circulation data sets generated by the numerical simulator SUNTANS using the *Proper Orthogonal Decomposition* (POD) technique.
- Perform the separation of spatial and temporal variables by projecting the input data onto the reduced spatial basis.
- Apply *Artificial Neural Networks* (ANN) with supervised learning to predict the evolution of the resulting time series associated to the POD temporal coefficients.

Snapshot data source

- After spatial discretization, the numerical circulation solver provides the following set of time dependent ODEs

$$\frac{dQ_h(t)}{dt} + \mathcal{N}_h(Q_h(t)) = S_h(t), \quad t \in \mathcal{T}$$

where $Q_h : \mathcal{T} \to \mathbb{R}^M$ is the discrete solution, M is the number of DOFs, $\mathcal{N}_h$ and $S_h : \mathcal{T} \to \mathbb{R}^M$ are the discrete nonlinear operator and source term S, respectively, and $\mathcal{T} \subset [0, T]$.

Assumptions

- We assume that reduced basis (RB) functions do exist and define the L-dimensional space

$$\mathbb{V}_{rb} = span\{\psi_1, \ldots, \psi_L\} \subset \mathbb{V}_h$$

where $\mathbb{V}_h$ is a subspace of a Hilbert space $\mathbb{V}$ defined over the original domain $\Omega \subset \mathbb{R}^d (d = 1, 2, 3)$ of the problem.

- An approximation of the full-order system can be expressed by the ansatz

$$Q_{rb}(\mathbf{x}, t) = \overline{q}(\mathbf{x}) + q'(\mathbf{x}, t) = \overline{q}(\mathbf{x}) + \sum_{i=1}^{L} a_i(t)\psi_i(\mathbf{x})$$

where $\mathbf{a}(t) = [a_1(t), \ldots, a_L(t)]^T \in \mathbb{R}^L$; $\overline{q}$ and q' are mean and fluctuating quantities.

Snapshot POD

- The snapshot POD method produces a set of L orthonormal basis functions defined over M degrees of freedom[1]

$$\Phi = [\Psi_1, \ldots, \Psi_L] \in \mathbb{R}^{M \times L} \text{ with } \Psi_i = \psi_i(x_j).$$

- Q_{rb} can be written as the product of a column-wise matrix of spatial functions Ψ_i and a row-wise matrix of temporal coefficients $\mathbf{a}(t)$ defined over N instants as

$$Q_{rb} = \overline{q}(\mathbf{x}) + \Phi\mathcal{A} \text{ with } \Phi \in \mathbb{R}^{M \times L}, \ \mathcal{A} \in \mathbb{R}^{L \times N},$$

$$\mathcal{A} = [\mathbf{a}(t_1), \ldots, \mathbf{a}(t_N)].$$

[1] M = number of discrete nodes x_j of a given mesh multiplied by the number of field variables.

Validation example

Lorenz discrete system

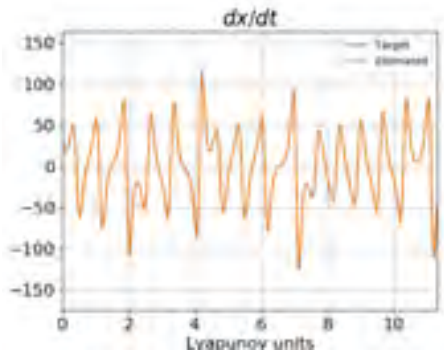
- The Lorenz system consists of the following set of ODEs:

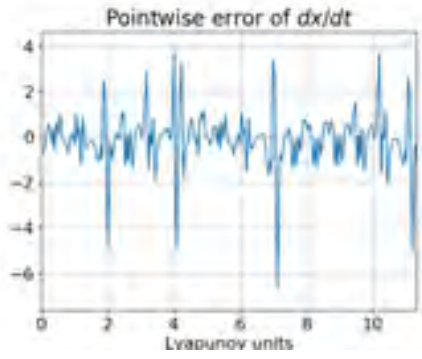
$$\dot{x} = \sigma(y - x),$$
$$\dot{y} = x(\rho - z) - y,$$
$$\dot{z} = xy - \beta z.$$

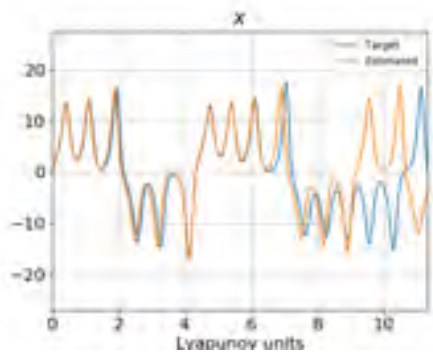
- Time derivatives of the coordinates x, y and z can be interpreted as **three separate time series**.
- We trained a FCNN to **predict** the **deterministic and chaotic** regimes of the Lorenz system.
- We performed a **hyperparameter optimization** to find the **best** neural network **architecture** for predicting the **chaotic case**.
- Then, we ran the **same FCNN** to predict **both system regimes**.

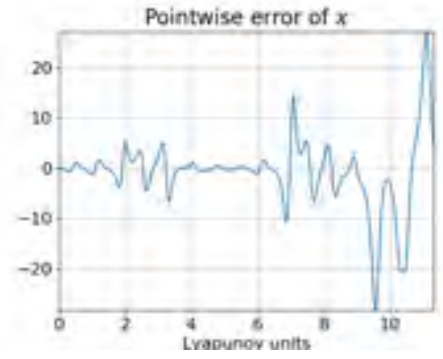
Lorenz discrete system

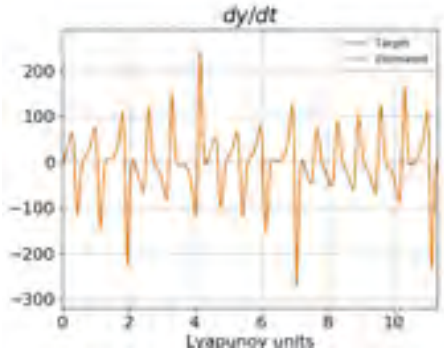
Chaotic setup

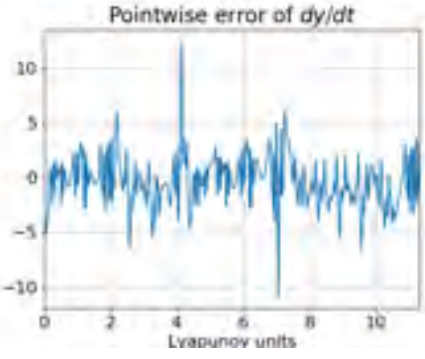


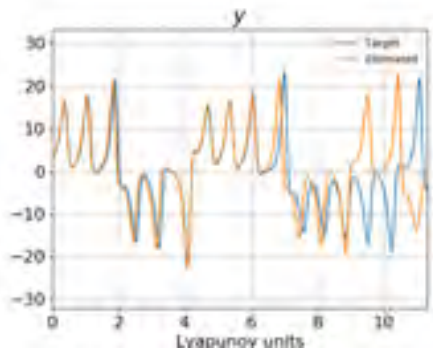


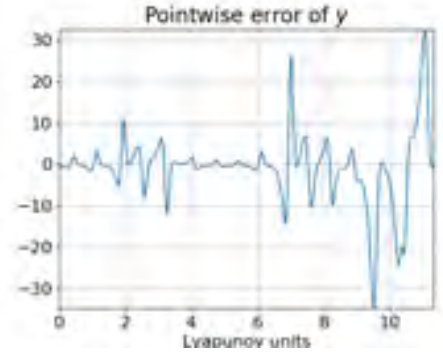


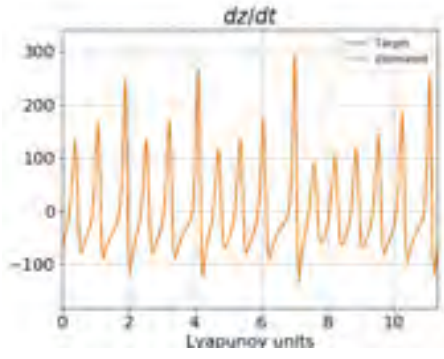


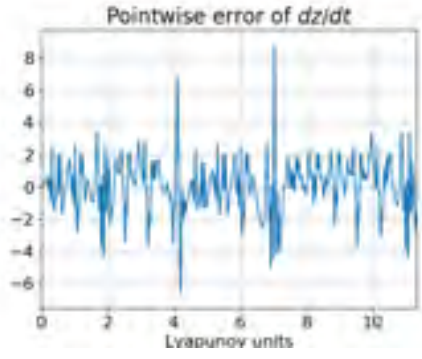


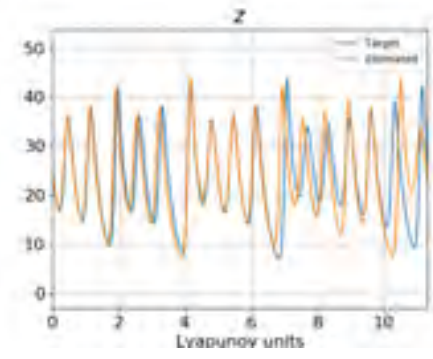


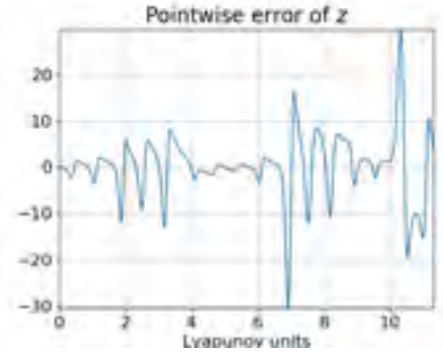








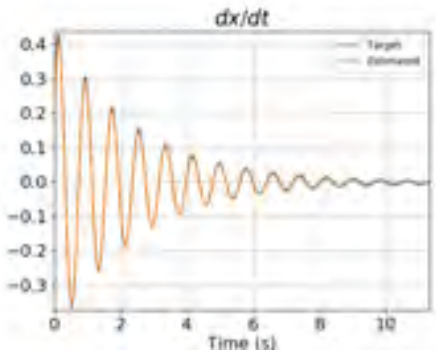


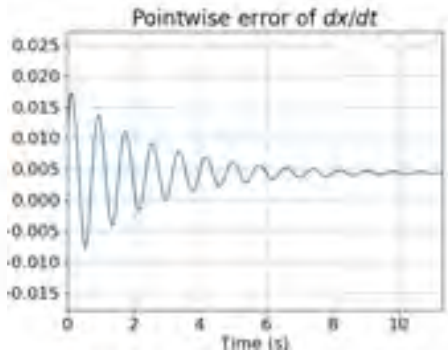


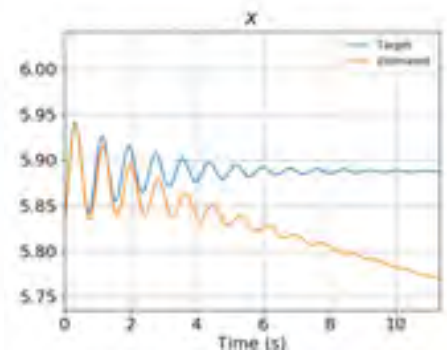
$\dot{x}, \dot{y}, \dot{z}$ $\quad e_{\dot{x}} = \dot{x}_{ref} - \dot{x}$ $\quad x, y, z$ $\quad e_x = x_{ref} - x$

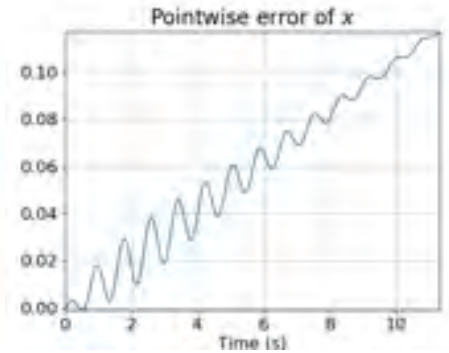
Lorenz discrete system

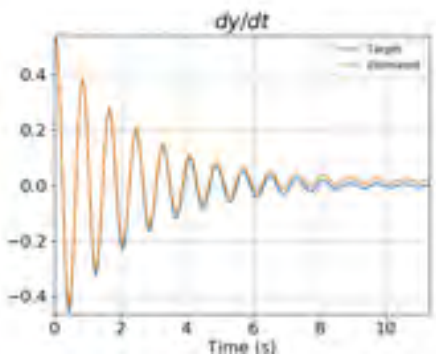
Deterministic setup

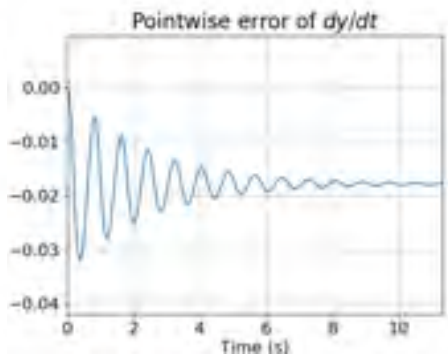


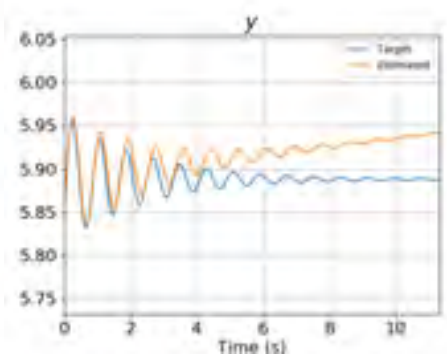


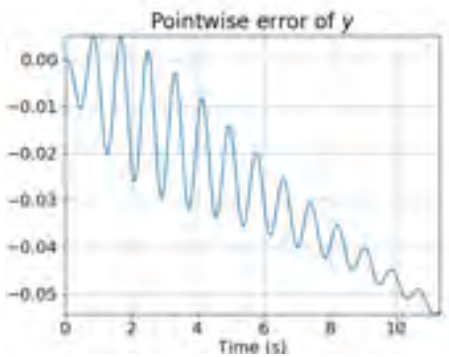


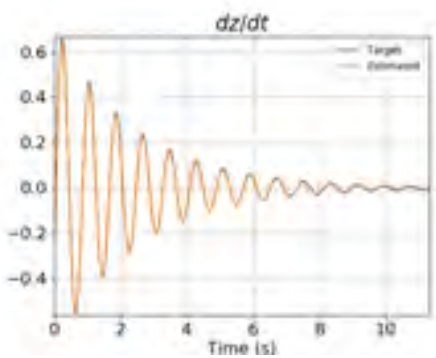


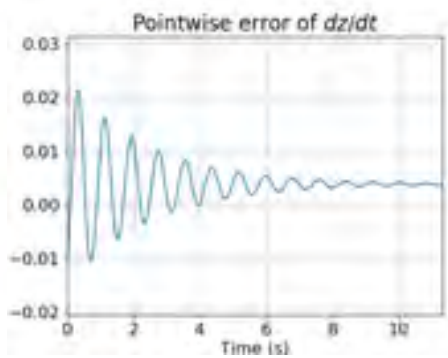


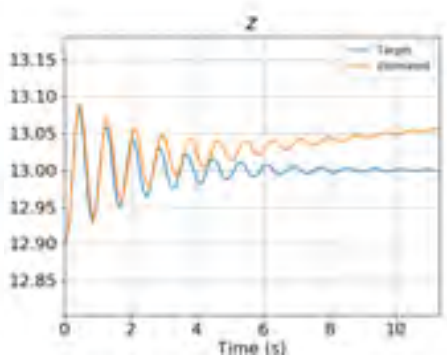


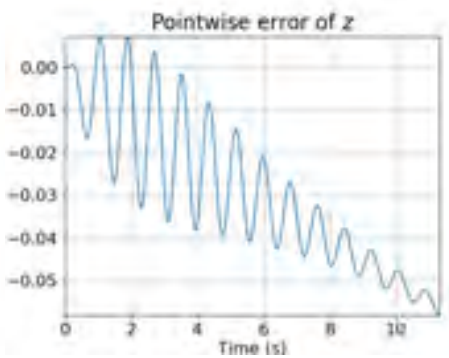












$\dot{x}, \dot{y}, \dot{z}$ $\quad e_{\dot{x}} = \dot{x}_{ref} - \dot{x}$ $\quad x, y, z$ $\quad e_x = x_{ref} - x$

Mimicking the discrete system

- **FCNN-ROM** mimics the evolution of an **ODE system** that represents the **dynamics** of the **original discretized system**

$$\Phi \dot{\mathbf{a}}(t) = \frac{dQ_{rb}(t)}{dt} \approx \frac{dQ_h(t)}{dt} = S_h(t) - \mathcal{N}_h(Q_h(t)).$$

- Thus

$$\frac{d\mathbf{a}(t)}{dt} \approx \Phi^T \left(S_h(t) - \mathcal{N}_h(Q_h(t)) \right).$$

Setting up FCNN to learn time derivatives

- FCNN is designed to receive the **temporal coefficients** $\mathbf{a}(t)$ of the POD expansion as **input** and to return the **time derivative** of these coefficients as **output**

$$\frac{d\mathbf{a}(t)}{dt} = \mathbf{W}^{[l]}\mathbf{G}^{[l-1]} + \mathbf{b}^{[l]},$$

where $\mathbf{G}^{[l-1]}$ is the activation matrix of the penultimate layer.

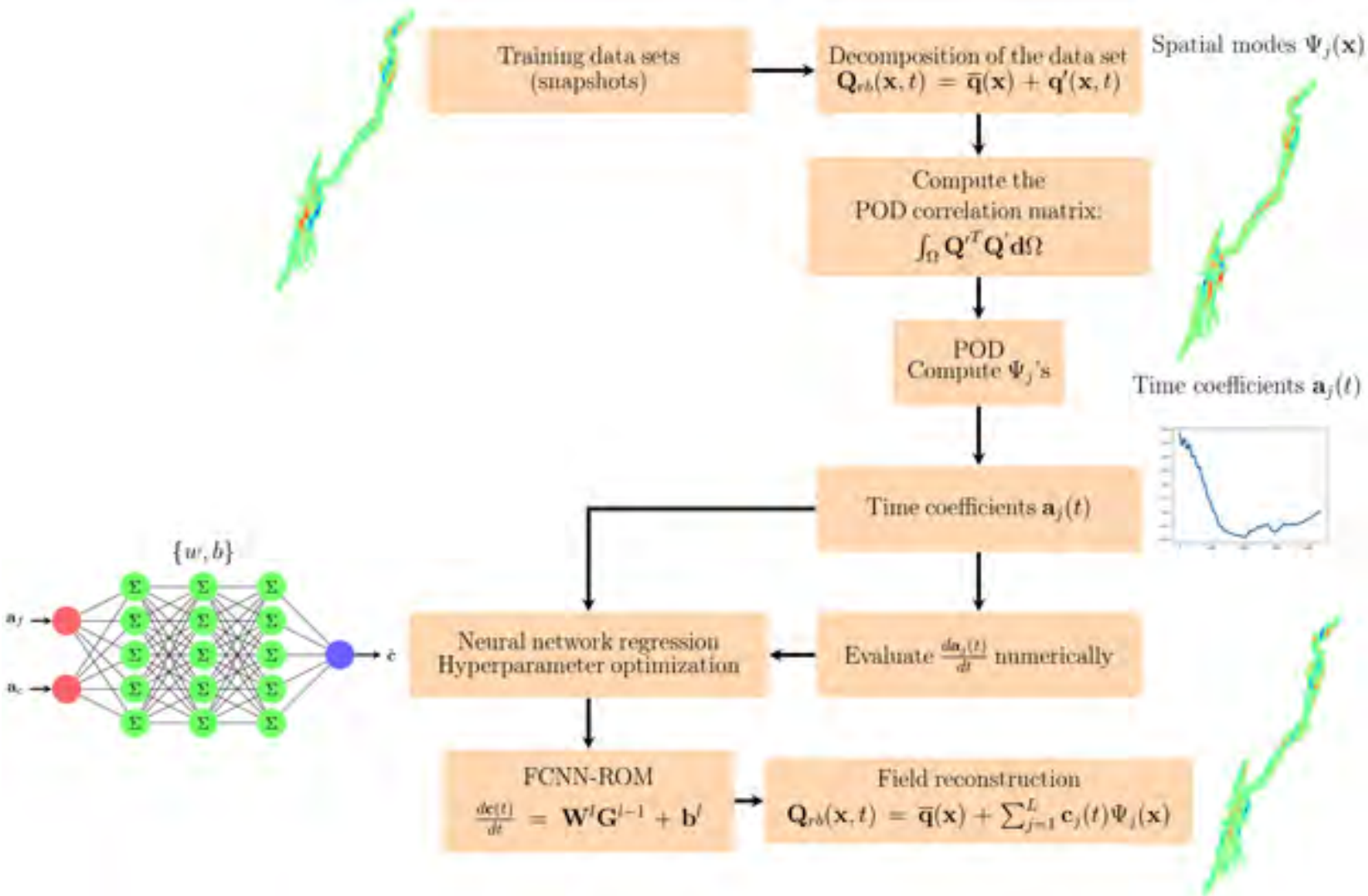
Making predictions

- FCNN's **loss function** is **minimized** with respect to the **time derivative** of the **temporal modes** computed with a **high resolution Finite Difference** scheme.
- FCNN estimates a **set of coupled ODEs** which is then **integrated** using a multi-stage **Runge-Kutta scheme**.
- Such approach allows to **predict the circulation field variables beyond the training window** because $\overline{q}(\mathbf{x})$ and $\Psi_i(\mathbf{x})$ depend only on the spatial coordinates $\mathbf{x}$.[2]

[2] RK time-steps used to reconstruct flow field variables don't even need to be the same size as those used for training.

FCNN-ROM

Algorithm workflow



Hydrodynamics example - FCNN-ROM

Experiment setup

- **Datasets**: Outputs of circulation and atmospheric fields simulated by SUNTANS and WRF from April 1^{st} to 20^{th}, 2019.
- **FCNN architecture**: Hyperparameter optimized.
- **Optimization algorithm**: 5040 iterations of Adam.
- **Training & Testing**: 19 batches of 36h for training and 1 batch of 36h for testing.
- **Accuracy target**: **5%** error in L2-norm with respect to the full order model computed with SUNTANS.
- **Dimensionality reduction**: 5 modes[3].

[3] based on a given energy content preserved

Hydrodynamics example - FCNN-ROM

Energy preserved by the ROM

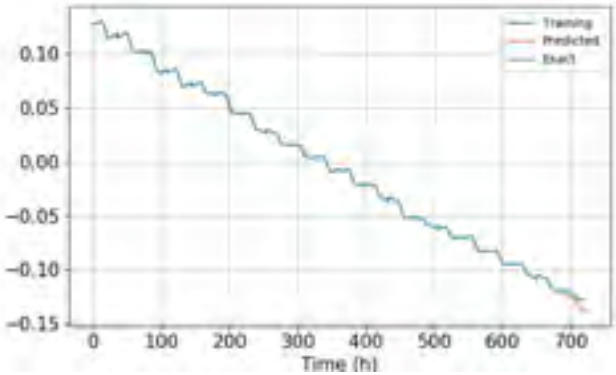
Table: Percentage of preserved energy in each spatial mode.

$\Psi_1(\mathbf{x})$	$\Psi_2(\mathbf{x})$	$\Psi_3(\mathbf{x})$	$\Psi_4(\mathbf{x})$	$\Psi_5(\mathbf{x})$	Total preserved energy
91.4%	2.4%	1.1%	0.6%	0.5%	96.0%

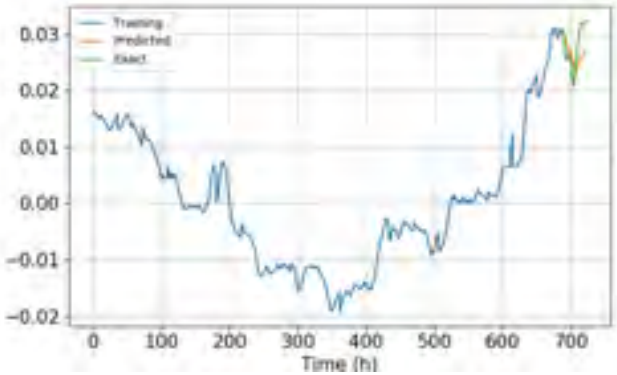
- It is worth noting that the preserved energy of each spatial mode is the same regardless of the circulation variable since the eigenvalues of the correlation matrix are the same for all circulation variable.

Hydrodynamics example - FCNN-ROM

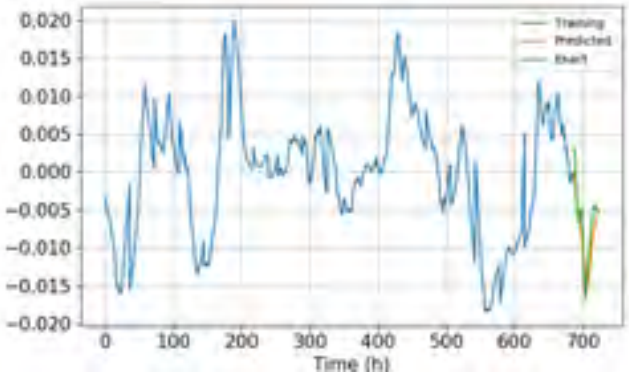
Time coefficients corresponding to the 5 spatial modes[4]



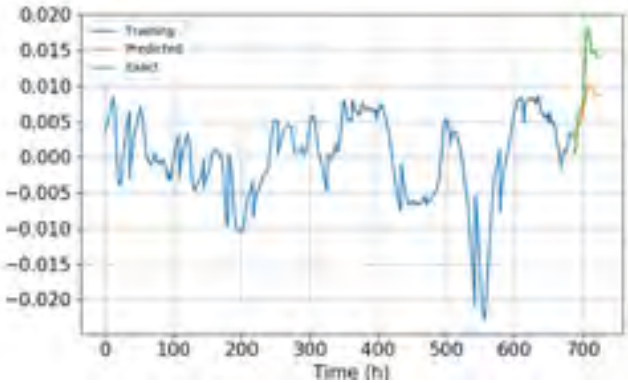
a) $\mathbf{a}_1(t)$



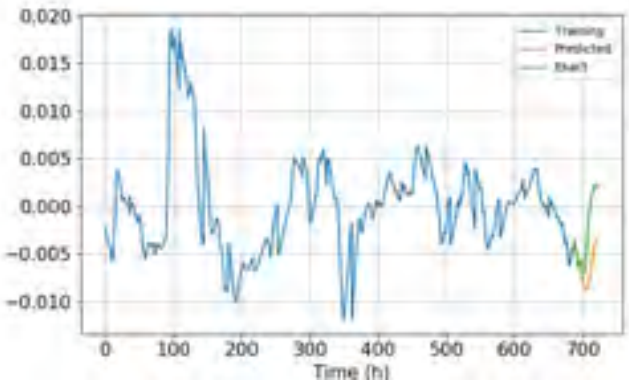
b) $\mathbf{a}_2(t)$



c) $\mathbf{a}_3(t)$



d) $\mathbf{a}_4(t)$



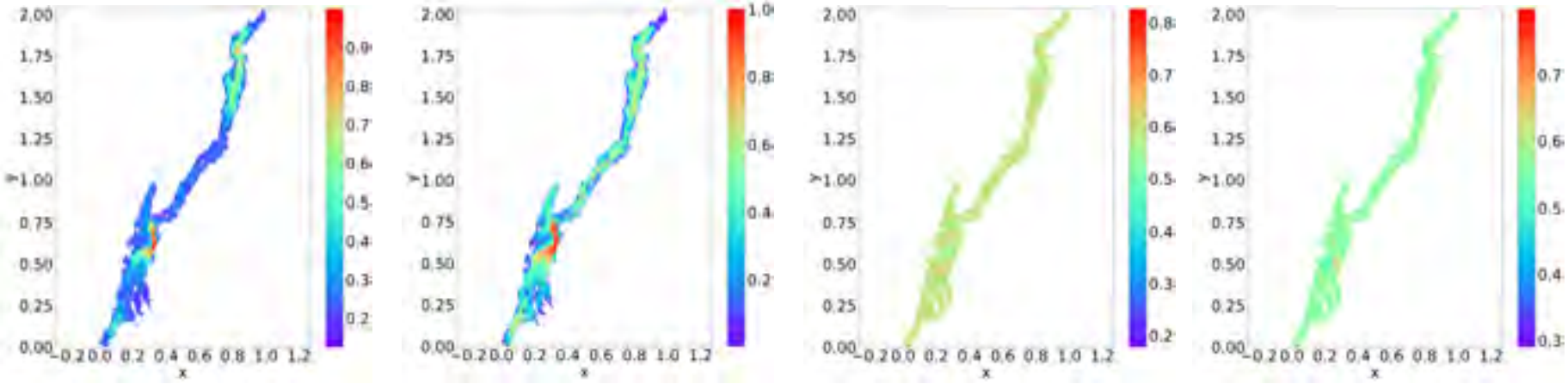
e) $\mathbf{a}_5(t)$

[4] Training: blue line; True values: green line; Predicted values: orange line

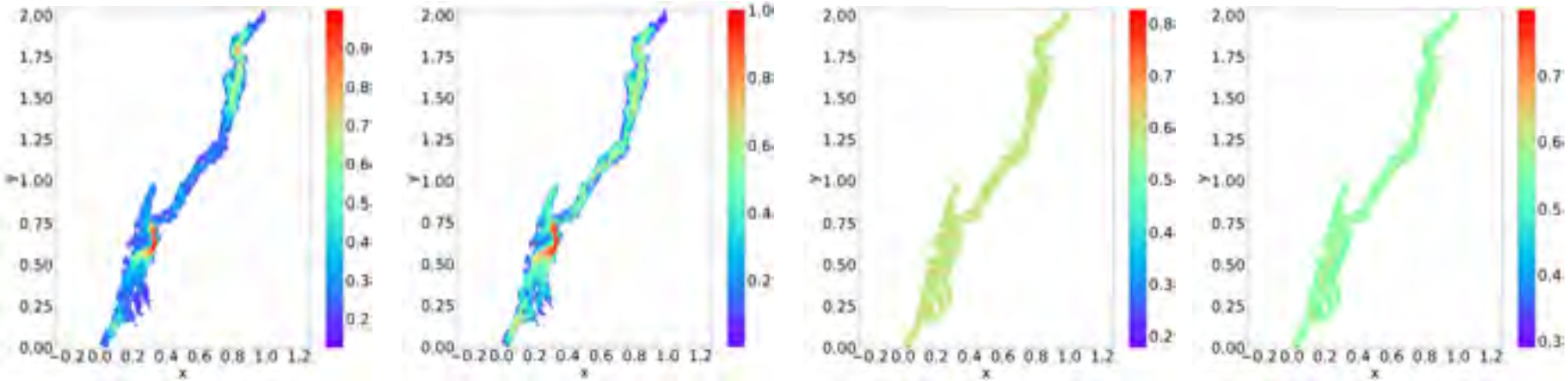
Hydrodynamics example - FCNN-ROM

Results: 36h Forecast - Reference Solution vs. Approximation

Density ρ **Temperature T** **East velocity u** **North velocity v**



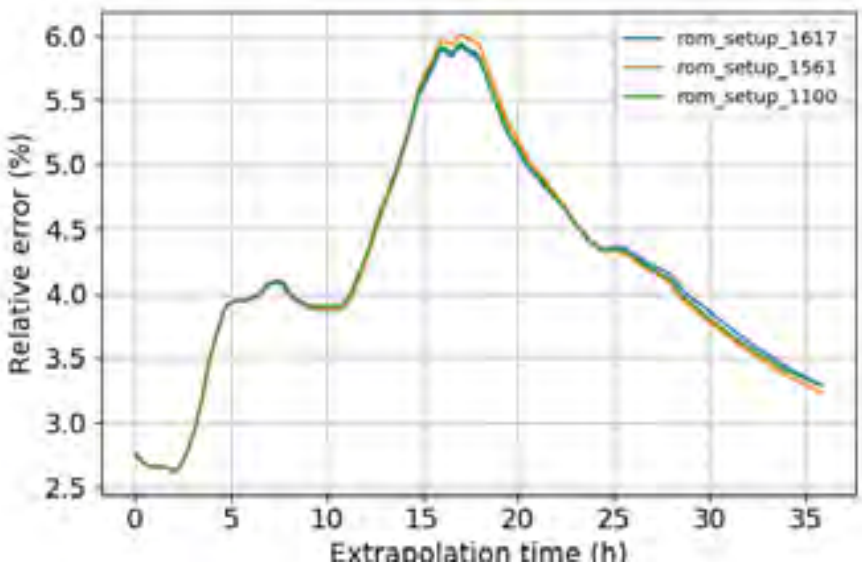
True



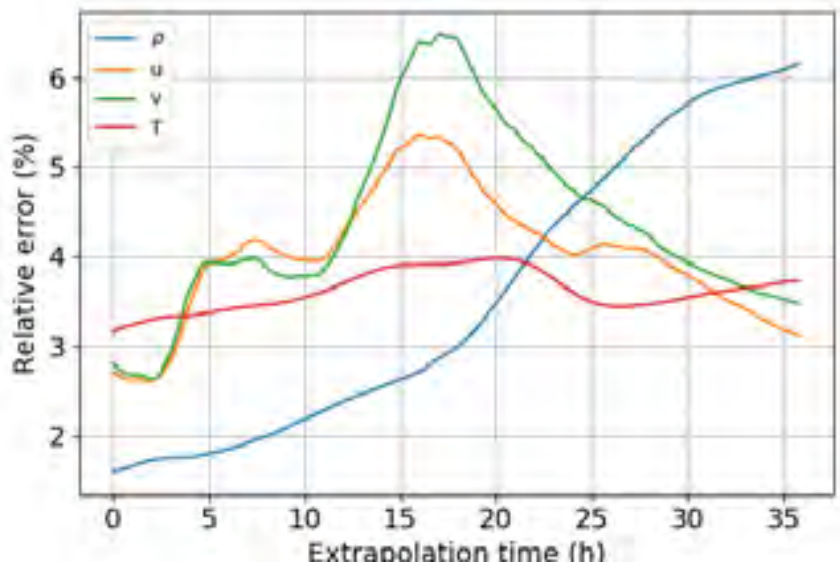
Predicted

Hydrodynamics example - FCNN-ROM

Results: Relative error in L^2-norm vs. prediction time[5]



a) Relative error for the 3 best hyperparameter settings



b) Relative error for each variable: ρ, u, v, T

Figure: a) Best performances based on the average $\frac{u+v}{2}$ error; b) Best configuration: model 1617.

[5] Model 1617 - hidden layers: 8; neurons per layer: 83, 65, 67, 145, 59, 103, 81, 129; dropout: 0.3 in each layer; learning rate: 1e-05; Adam: 5040 iter.

Hydrodynamics example - LSTM-ROM

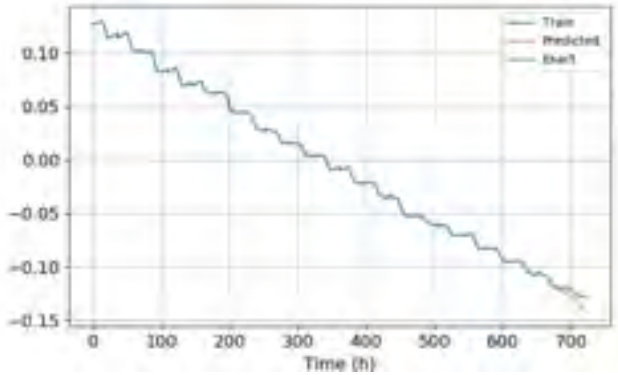
Experiment setup

- **Datasets**: Outputs of circulation and atmospheric fields simulated by SUNTANS and WRF from April 1^{st} to 20^{th}, 2019.
- **LSTM architecture**: Hyperparameter optimized.
- **Optimization algorithm**: 1000 iterations of Adam.
- **Moving window**: Asymmetrical window with 200 min input and 10 min output[6].
- **Training & Testing**: 19 batches of 36h for training and 1 batch of 36h for testing.
- **Accuracy target**: **5%** error in L2-norm with respect to the full order model computed with SUNTANS.
- **Dimensionality reduction**: 5 modes

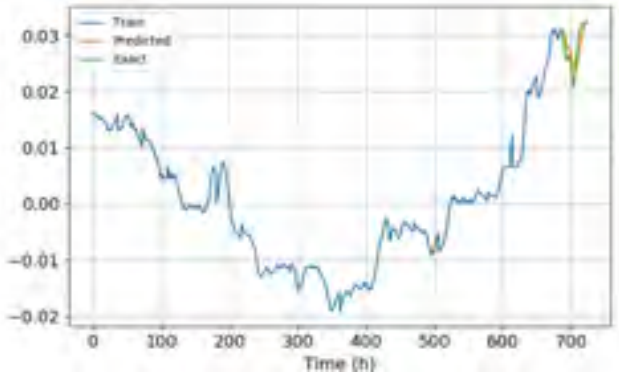
[6] Data input has 10 min resolution.

Hydrodynamics example - LSTM-ROM

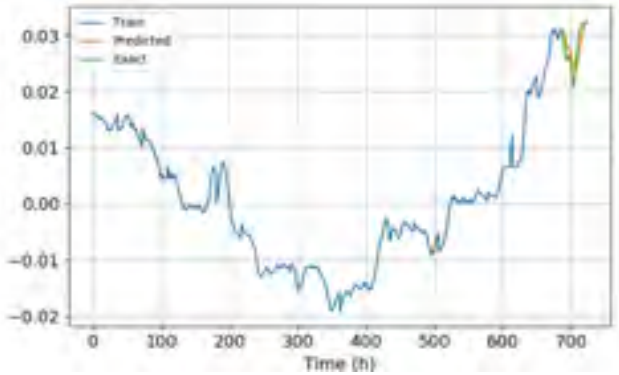
Time coefficients corresponding to the 5 spatial modes[7]



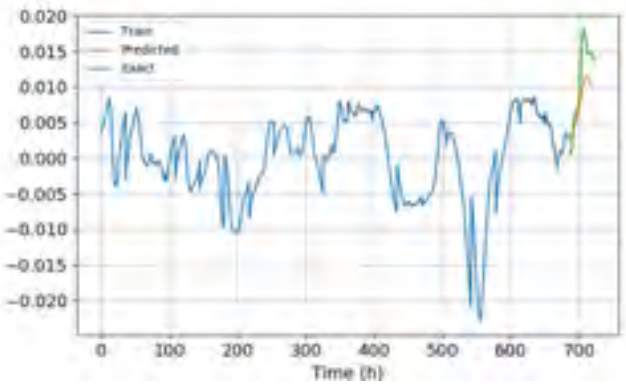
a) $\mathbf{a}_1(t)$



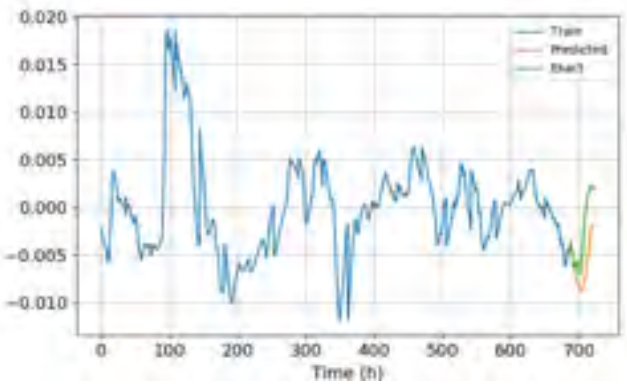
b) $\mathbf{a}_2(t)$



c) $\mathbf{a}_3(t)$



d) $\mathbf{a}_4(t)$



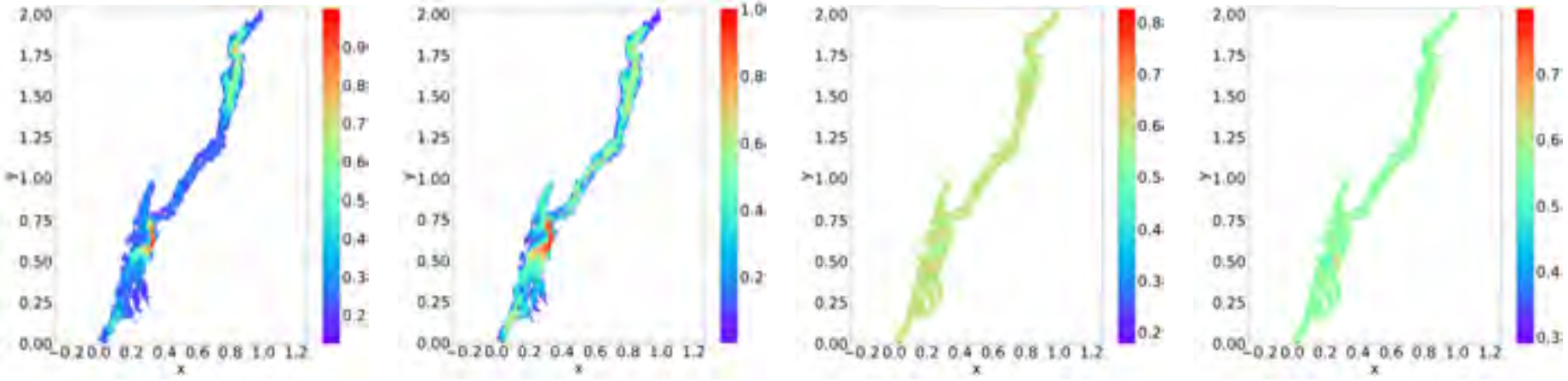
e) $\mathbf{a}_5(t)$

[7] Training: blue line; True values: green line; Predicted values: orange line; Obs.: LSTM-ROM and FCNN-ROM share all spatial modes

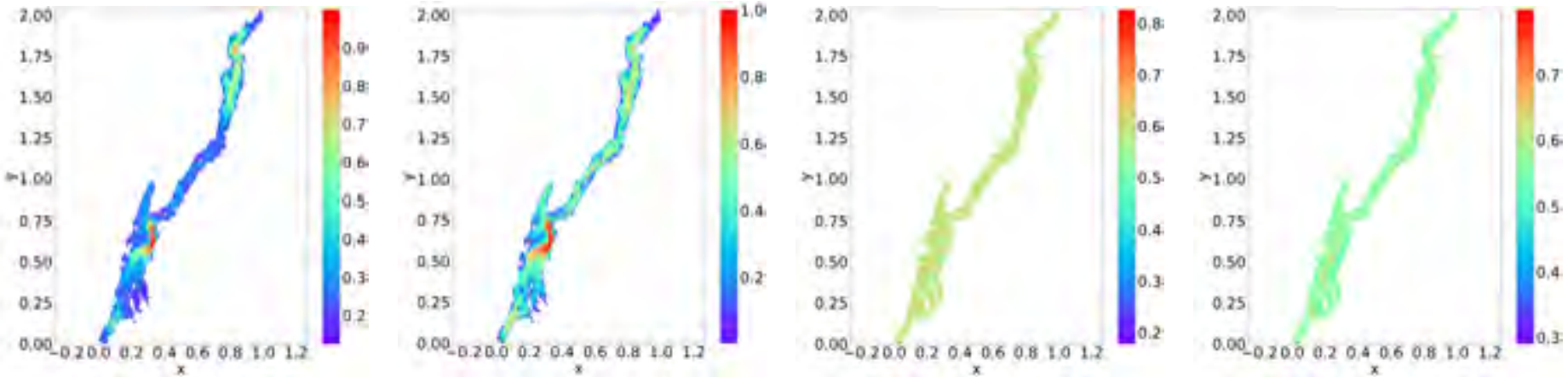
Hydrodynamics example - LSTM-ROM

Results: 36h Forecast - Reference Solution vs. Approximation

Density ρ | **Temperature T** | **East velocity u** | **North velocity v**



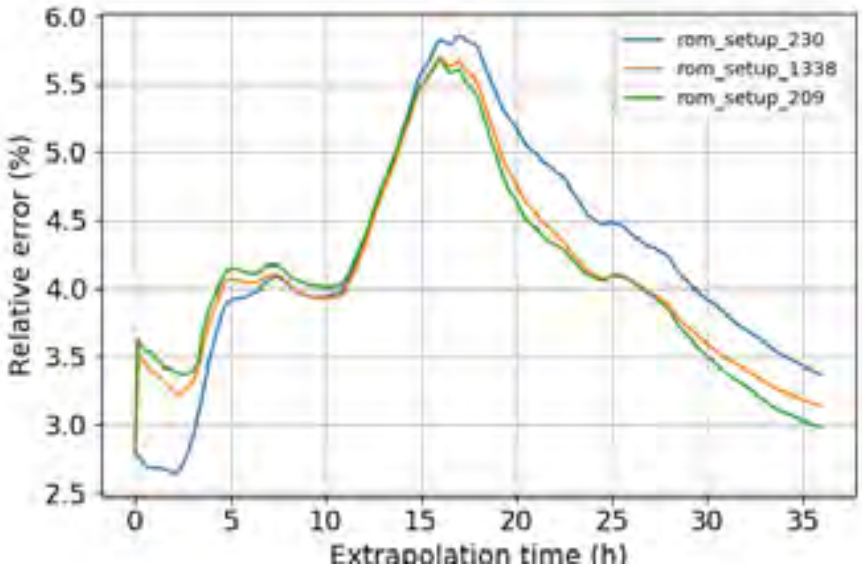
True



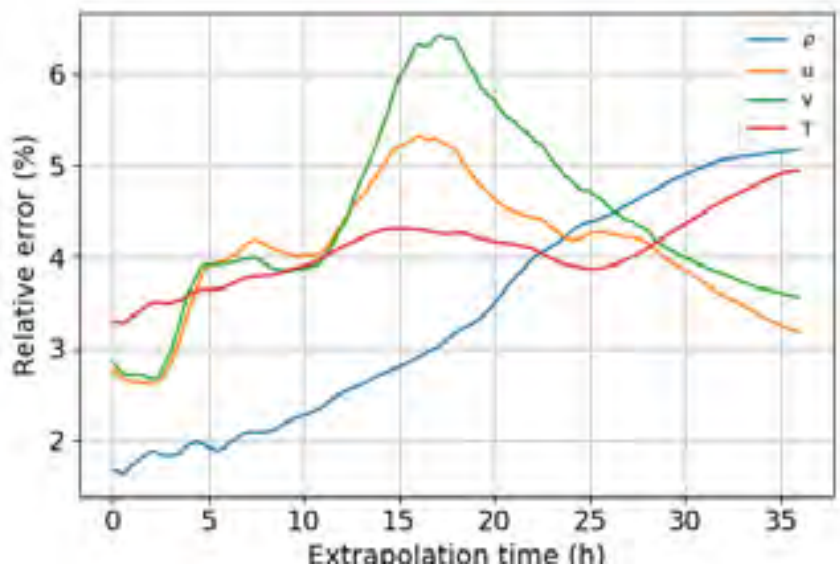
Predicted

Hydrodynamics example - LSTM-ROM

Results: Relative error in L^2-norm vs. prediction time[8]



a) Relative error for the 3 best hyperparameter settings



b) Relative error for each variable: ρ, u, v, T

Figure: a) Best performances based on the average $\frac{u+v}{2}$ error; b) Best configuration: model 230.

[8] Model 230 - hidden layers: 3; LSTM cells per layer: 133, 92, 129; Dropout: 0.2, 0.3, 0.3; Learning rate: 0.01; Adam: 1000 iter; Window input: 200 min; Window output:10 min.

Conclusions

Hydrodynamics ROMs

- **POD** technique showed great promise for hydrodynamics application since it **drastically reduces the computational cost of simulations**.
- **Hyperparameter optimation** was **crucial** to **generate accurate models**.
- The **target error** was **achieved** by the best hyper-parameter setup for **all physical variables except density** in both cases (FCNN and LSTM-ROM).
- **FCNN-ROM** used approximately **20% of the learning parameters** used by **LSTM-ROM** to perform equivalently[9].

[9] FCNN-ROM: 55.101 wheights and biases; LSTM-ROM: 264.690.

Thank you!

albercn@br.ibm.com
cwatson@us.ibm.com

