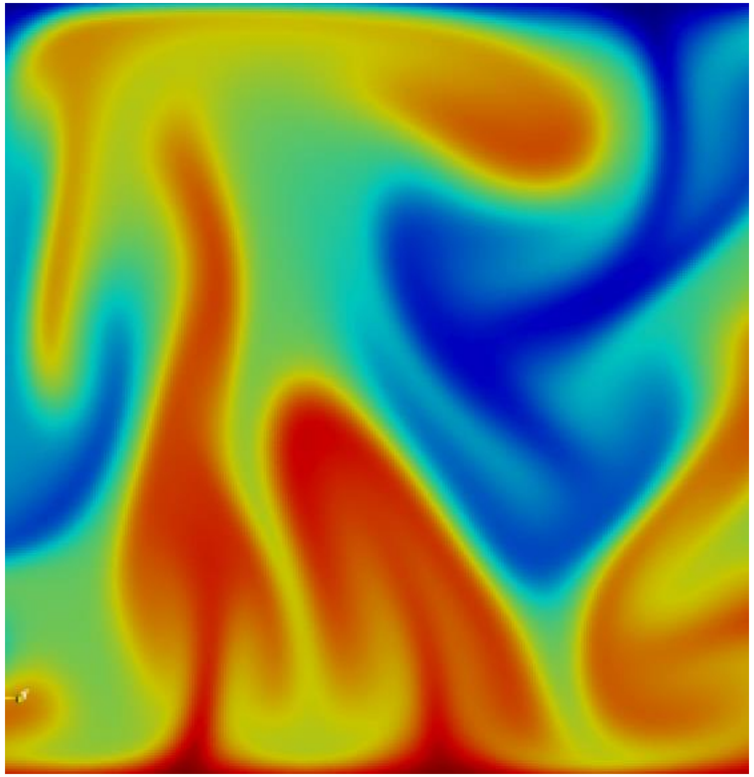
UNIVERSITÄT BIELEFELD
Faculty of Technology

Koopman-based Modeling of Rayleigh-Bénard Convection

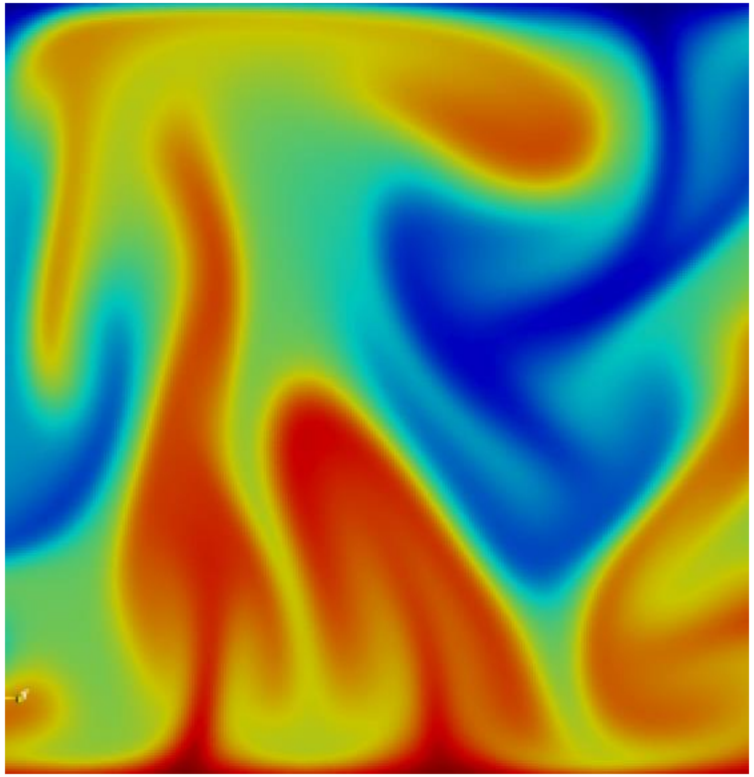


Presenter: Michiel Straat

Slides created by:
Thorben Markmann

Machine Learning Group, Bielefeld University

Table of contents



Dynamical Systems

- Dynamical Systems
  1. state space
  2. time evolution rule (dynamics)
- Fluid Dynamics
  - model the flow of fluids
  - partial differential equations (**PDE**s)
  - non-linear **Navier-Stokes** equations
- wide range of applications
  - active flow control and shape optimization [1]
  - wind energy, combustion, nuclear fusion [2]

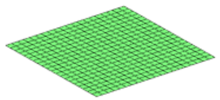


Fig.1a Wave equation

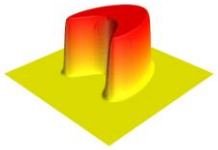


Fig.1b Heat equation

[1] J. Rabault, F. Ren, W. Zhang, H. Tang, and H. Xu, "Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization," J Hydrodyn, vol. 32, no. 2, pp. 234–246, Apr. 2020
[2] S. Werner and S. Peitz, "Learning a model is paramount for sample efficiency in reinforcement learning control of PDEs." arXiv, Mar. 13, 2023.

Rayleigh-Bénard Convection (RBC)

- layer of fluid heated from below
- **turbulent** fluid motion
- models energy/heat transfer through **convection** and **conduction**
- Model **thermal convection** in nature
  - oceans and atmosphere
  - motion of gases in stars
- Key parameters:
  - Rayleigh nr. (Ra)
  - Prandtl nr. (Pr)

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}} \nabla^2 \boldsymbol{u} + T\boldsymbol{i}\,,$$
$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \frac{1}{\sqrt{RaPr}} \nabla^2 T\,,$$

Eq.1. governing PDE[1]

[1] "Demo - Rayleigh Benard — shenfun 4.1.4 documentation." [Online]. Available: https://shenfun.readthedocs.io/en/latest/rayleighbenard.html

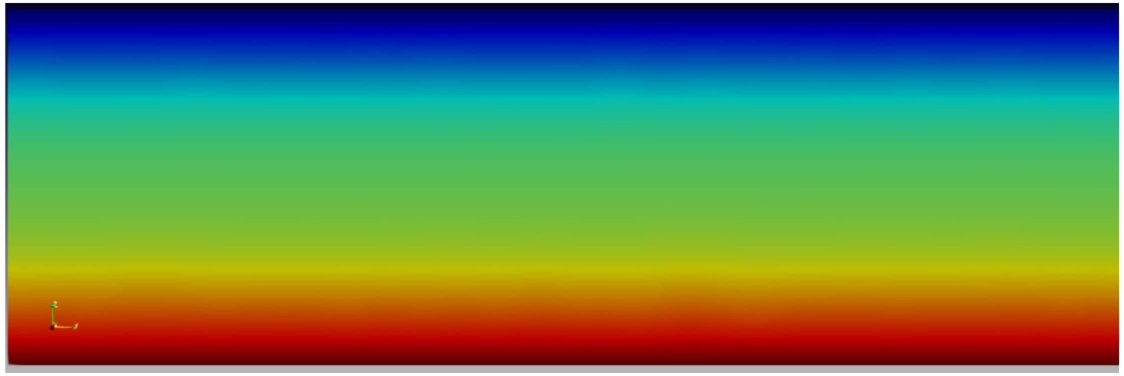


Fig.2 time evolution of temperature field T

Rayleigh-Bénard Convection

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}}\nabla^2 \boldsymbol{u} + T\boldsymbol{i}\,,$$
$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \frac{1}{\sqrt{RaPr}}\nabla^2 T\,,$$

Representations of the system:

1. full system state
   - Temperature Field *T*
   - Velocity Field **u**

2. local convective field
   $$j_{conv} = u_y \cdot (T - T_{avg})$$

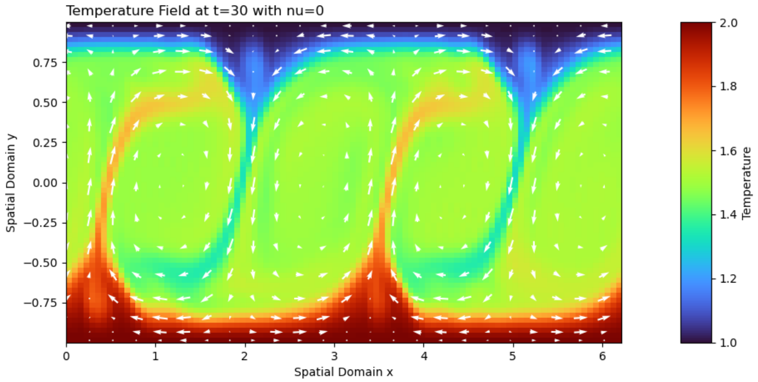


Fig.3 system state

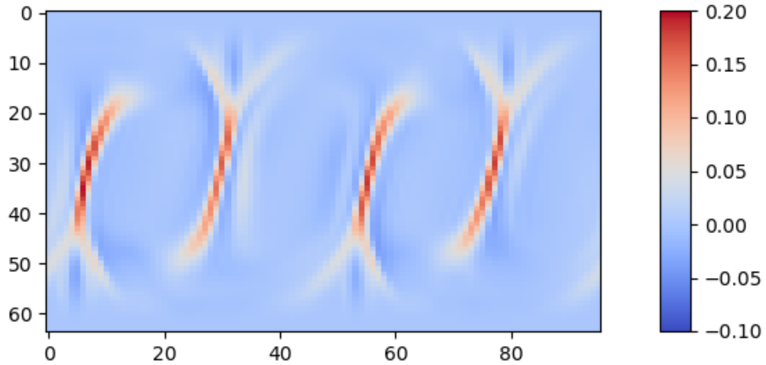


Fig.4 local convective field

Solving Rayleigh-Bénard Convection

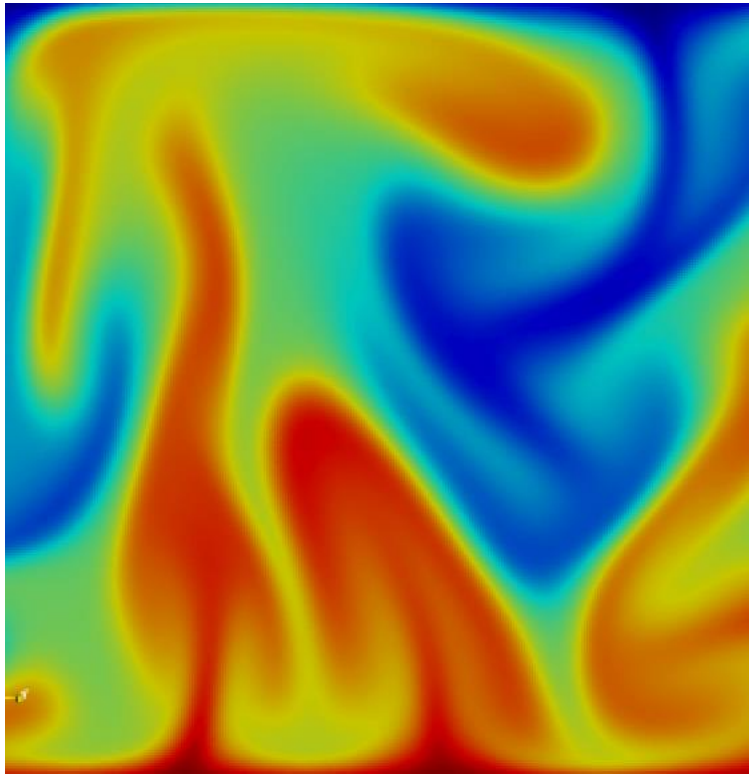
- Direct Numerical Simulation Solver (DNS)
  - solve Navier-Stokes equations **numerically**
  - based on the Shenfun python package[1]

- RBC-Dataset
  - 30 episodes using different initial conditions
  - 500 time steps per episode
  - 15000 system states
  - ~500MB of data

[1] “Demo - Rayleigh Benard — shenfun 4.1.4 documentation.” [Online]. Available: https://shenfun.readthedocs.io/en/latest/rayleighbenard.html

Table of contents

1. Dynamical Systems
   - Rayleigh-Bénard Convection
2. **Control**
   - **Model Predictive Control**
3. Modeling Rayleigh-Bénard
   - Paper WIP
4. Future Work



Controlling Rayleigh-Bénard Convection

- control strategies to **suppress the convective heat exchange**
- control by applying small temperature fluctuations to the lower boundary
- numerous **industrial applications**
  - crystal growth processes in silicon wafer production [2]
  - quality endangered by fluid motion

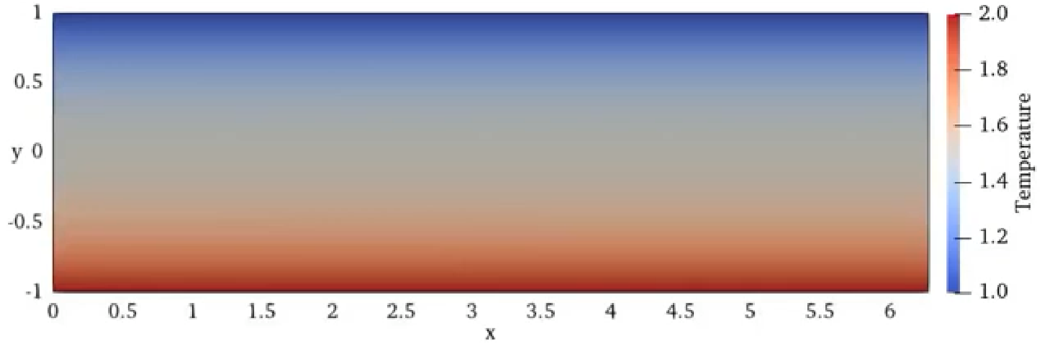


Fig.5 uncontrolled RBC[1]

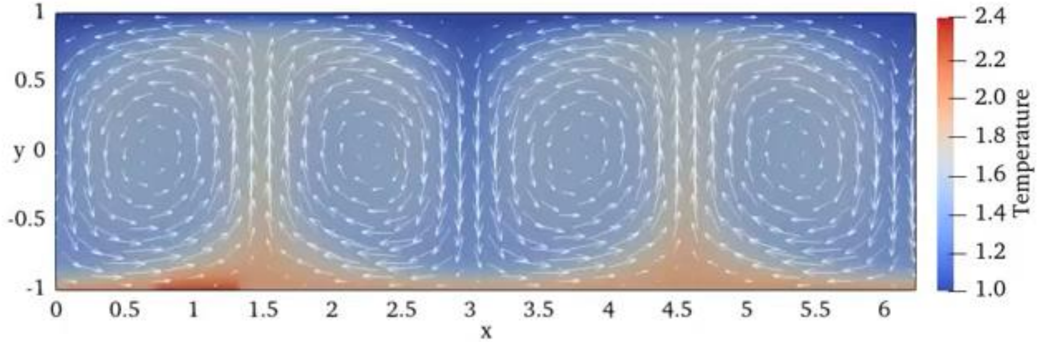


Fig.6 controlled RBC using RL[1]

[1] Vignon, C., Rabault, J., Vasanth, J., Alcántara-Ávila, F., Mortensen, M., & Vinuesa, R. (2023). Effective control of two-dimensional Rayleigh–Bénard convection: Invariant multi-agent reinforcement learning is all you need. *Physics of Fluids*, *35*(6), 065146.
[2] Müller G. Convection and inhomogeneities in crystal growth from the melt. In: Crystal growth from the melt. Springer; 1988. p. 1–136.

Controlling Rayleigh-Bénard Convection

- control strategies to **suppress the convective heat exchange**
- The Nusselt number measures the convective heat exchange
  - ratio of convective to conductive heat flux
- mean of the local convective field is proportional to the Nusselt Number:
  - $Nu = 1 + \sqrt{Ra * Pr}\langle j_{conv} \rangle_{A,t}$

$$Nu_{\text{inst}} = \frac{\left\langle \partial \mathbf{u}/\partial y T \right\rangle_{x,y} - \kappa \left\langle \partial \langle T \rangle_x / \partial y \right\rangle_y}{\kappa \left(T_H - T_C \right) / H}$$

Fig.7 Nusselt Number

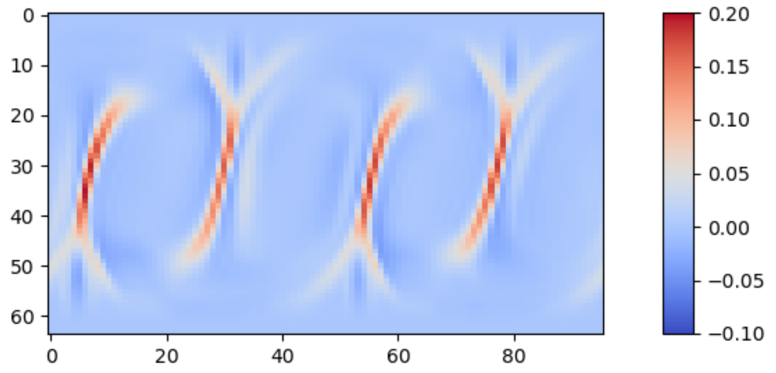


Fig.8 local convective field

Control Methods for Turbulent Fluids

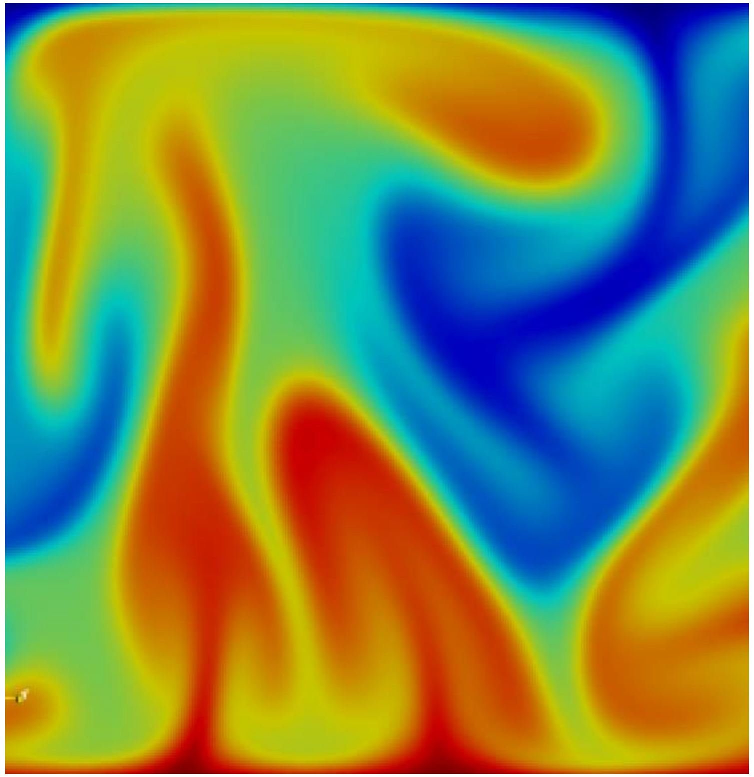
- Control of turbulent fluids is **not trivial**
  - **conventional control methods fail** for highly turbulent Rayleigh-Bénard Convection
- **Model-Free** Reinforcement Learning
  - almost all literature on RBC
- Problems
  - **massive amount of data** from numerical simulations
  - e.g. large 3D-RBC simulations take multiple gigabytes of raw data for each time step
  - numerical simulations are **slow**

➢ need for **sample efficiency** and new control methods

Model Predictive Control

- **Model-Based** Reinforcement Learning
- model system dynamics, e.g. using **Machine Learning** methods, without solving the underlying nonlinear equations
- online learning using Dyna
- **data-driven Reduced-order Models** to approximate the dynamics

Table of contents



Modeling Rayleigh-Bénard

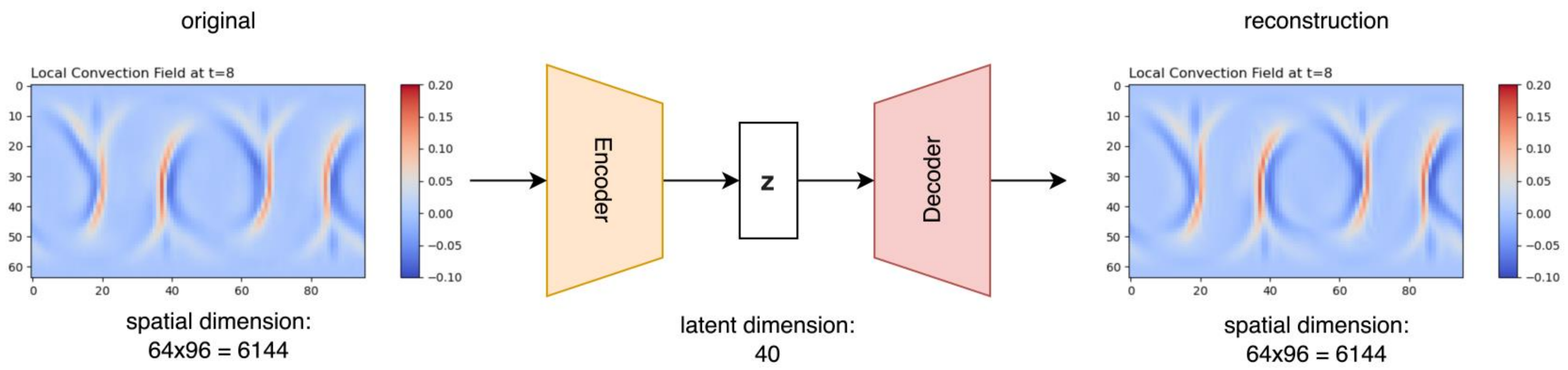
- **Goal**: find a good architecture to model RBC in Model-Predictive Contol
- Paper (WIP):
  - offline learning of the system dynamics
  - no control applied yet
  - model requirements:
    - sample efficiency
    - precise short-term prediction
  - models in literature aim for long-term predictions

Modeling Rayleigh-Bénard

- **Goal**: find a good architecture to model RBC in Model-Predictive Contol
- Paper (WIP):
  - proposed architecture combines:
    - Dimension Reduction: **Autoencoder**
    - Time stepping: **Koopman Operator**

Dimension Reduction - Autoencoder

- spatial dimension of the system may be high
- actual dynamics can live on a **low-dimensional manifold**
- Autoencoder trained on simulation data:
  - convolutional autoencoder
  - simple encoder/decoder with 4 convolutional layers
  - evaluated latent dimensions in the range [10, 200]



Dimension Reduction - Autoencoder

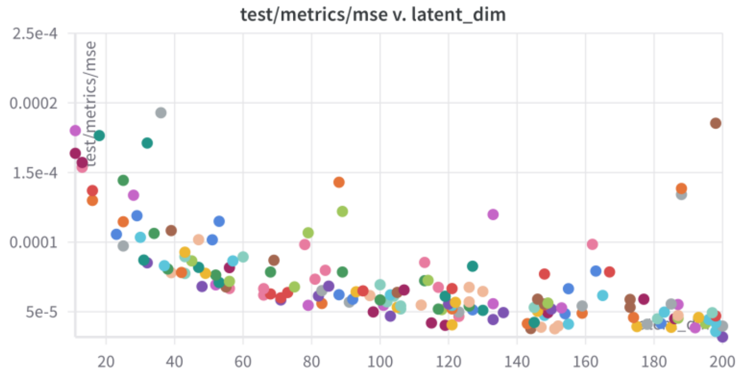


Fig.10 MSE vs latent dimension

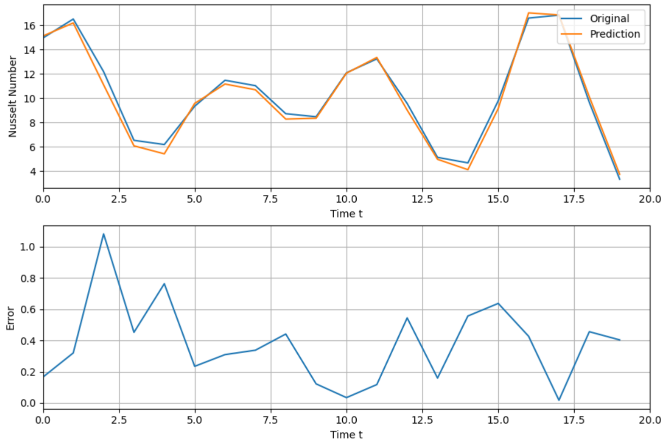


Fig.11 Nusselt Number in original and reconstructed sequence

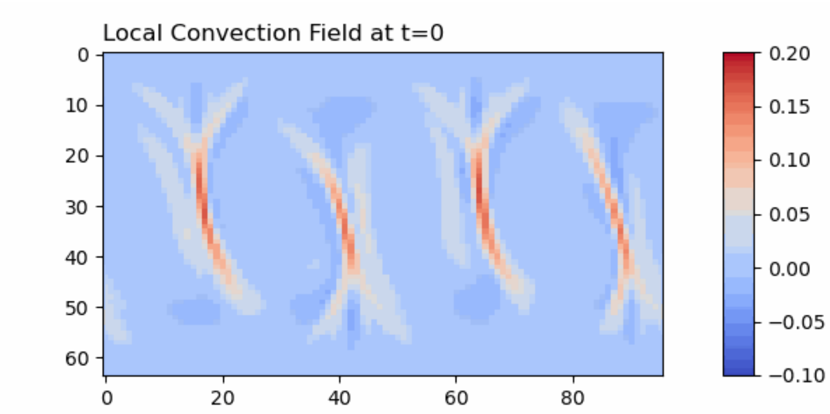


Fig.12a original sequence

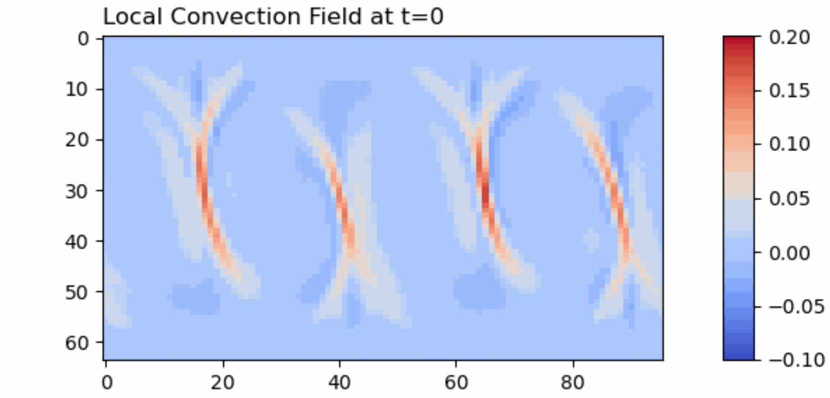


Fig.12b reconstructed sequence

Time Stepping

- Time-Stepping in latent space of AE
- given an input x_t predict next states of the system $(x_{t+0}, x_{t+1}, \dots, x_{t+T})$
- Methods from Machine Learning and Fluid Dynamics
  - Neural Networks: LSTM, GRU, …
  - Gaussian Processes
  - Koopman Operator
- How to include control?

Koopman Theory

- Linearization of the dynamics $\mathrm{F_t}$
- Define **observable functions** on system state
  - $y_t = g(x_t)$
  - e.g. $\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}$
- **Koopman operator K** performs time stepping in observable function space

Dynamics:

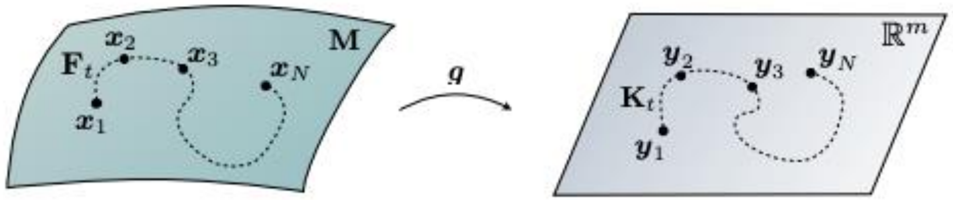
$$\frac{d}{dt}x = f(x) \implies x_{k+1} = \mathrm{F}_t(x_k)$$

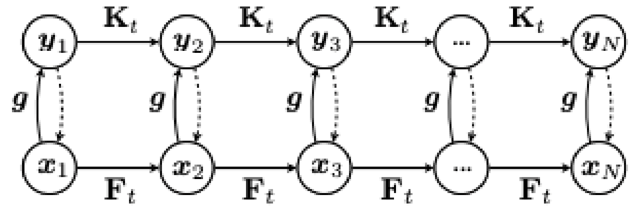
(Discrete-time Update)

Koopman operator:

$$K_t g = g \circ \mathrm{F}_t \implies g(x_{k+1}) = K_t g(x_k)$$

(Discrete-time Update)





Koopman Theory - Example

- Nonlinear dynamics:

$$\dot{x}_1 = \mu x_1$$
$$\dot{x}_2 = \lambda(x_2 - x_1^2)$$

- Koopman linear system:

$$\frac{d}{d_t}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \text{ for } \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}$$

Koopman Operator - Pros and Cons

- Cons:
  - How to choose observable functions?
  - Trade-off between dimensionality and accuracy
  - Prediction is probably worse compared to nonlinear models (neural networks, LSTM, …)

- Pros:
  - Linear Model
  - Adding control to the system is easy
  - Explainability of the model
  - We don’t need a perfect model for RL -> even accurate predictions of “only” 15 time steps can give a huge advantage in sample efficiency

Linear Recurrent Autoencoder Network

- Dimensionality Reduction via Autoencoder
- Timestepping via Koopman Operator
- Combination is also called Linear Recurrent Autoencoder Network (LRAN)
- Learn both together
  - Autoencoder learns the observable functions (!)

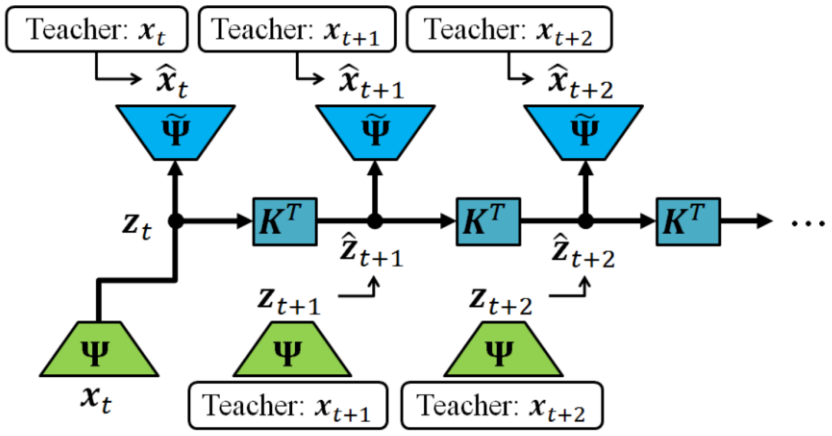


Fig 13. Linear Recurrent Autoencoder Network (LRAN) architecture[1]

[1] S. E. Otto and C. W. Rowley, “Linearly Recurrent Autoencoder Networks for Learning Dynamics,” SIAM J. Appl. Dyn. Syst., vol. 18, no. 1, pp. 558–593, Jan. 2019, doi: 10.1137/18M1177846.

Linear Recurrent Autoencoder Network

Loss function

$$J(\boldsymbol{\theta}_{enc}, \boldsymbol{\theta}_{dec}, \boldsymbol{\theta}_{\mathbf{K}}) = \underset{\mathbf{x}_t,\ldots,\mathbf{x}_{t+\mathcal{T}-1}\sim P_{data}}{\mathbb{E}} \frac{1}{1+\beta}\left[\sum_{\tau=0}^{\mathcal{T}-1} \frac{\delta^t}{N_1(\delta)} \frac{\left\|\hat{\mathbf{x}}_{t+\tau} - \mathbf{x}_{t+\tau}\right\|^2}{\left\|\mathbf{x}_{t+\tau}\right\|^2 + \epsilon_1} + \beta \sum_{\tau=1}^{\mathcal{T}-1} \frac{\delta^{t-1}}{N_2(\delta)} \frac{\left\|\hat{\mathbf{z}}_{t+\tau} - \mathbf{z}_{t+\tau}\right\|^2}{\left\|\mathbf{z}_{t+\tau}\right\|^2 + \epsilon_2}\right] + \Omega(\boldsymbol{\theta}_{enc}, \boldsymbol{\theta}_{dec}, \boldsymbol{\theta}_{\mathbf{K}}),$$

with $$N_1(\delta) = \sum_{\tau=0}^{\mathcal{T}-1} \delta^\tau, \qquad N_2(\delta) = \sum_{\tau=1}^{\mathcal{T}-1} \delta^{\tau-1}$$

- mean squared prediction error
  - normalized MSE
- error for reconstruction and hidden var
- δ decaying weight
- β weight of reconstruction and hidden error

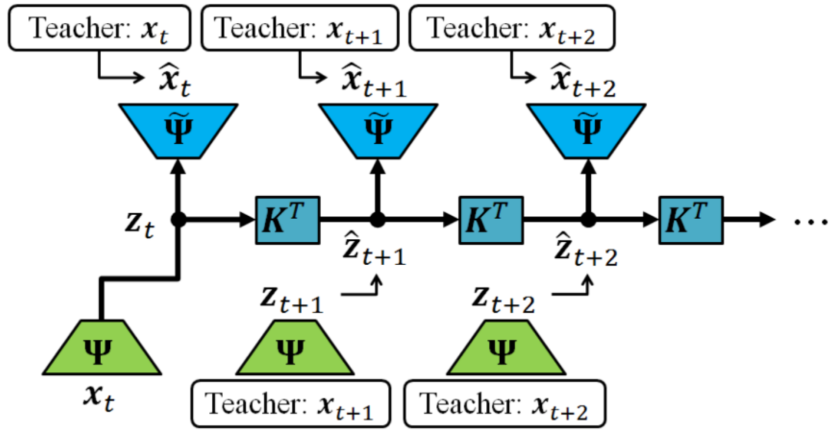


Fig 13. Linear Recurrent Autoencoder Network (LRAN) architecture[1]

[1] S. E. Otto and C. W. Rowley, “Linearly Recurrent Autoencoder Networks for Learning Dynamics,” SIAM J. Appl. Dyn. Syst., vol. 18, no. 1, pp. 558–593, Jan. 2019, doi: 10.1137/18M1177846.

Linear Recurrent Autoencoder Network

- Some first results:
  - LRAN trained on sequences of length 10
  - evaluated on sequences of 50 time steps

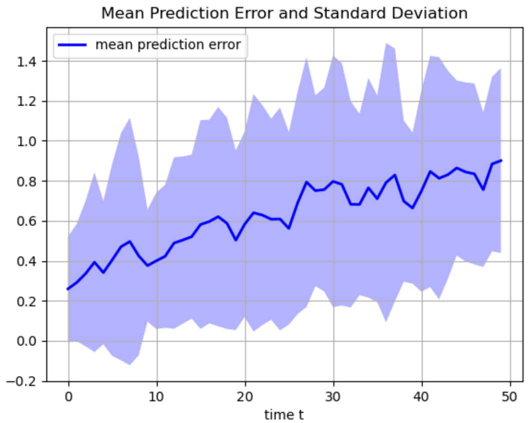


Fig 14. mean prediction error

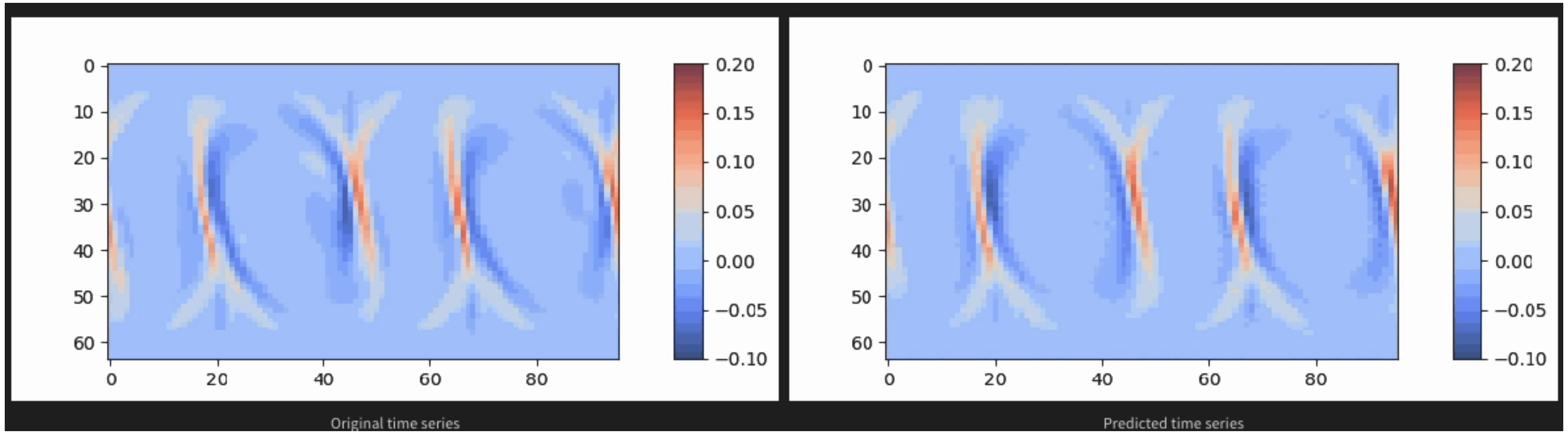


Fig 13. original and predicted sequence

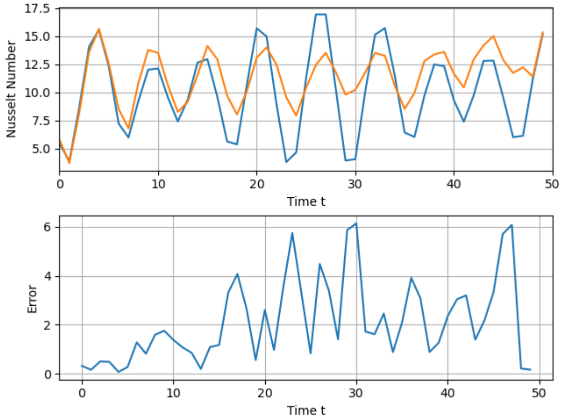


Fig 15. Nusselt number error

Experiment 1: Hyperparameters

- Hyperparameter Sweep
  - Weights β and δ
  - model complexity (latent dimension and layer number)
  - training sequence length

Experiment 2: Compare to other methods

- Combine different dimension reduction and timestepping techniques
  - dimension reduction: POD/PCA (linear), Autoencoder (nonlinear)
  - timestepping: Koopman (linear), DMD (linear), GRU (nonlinear)

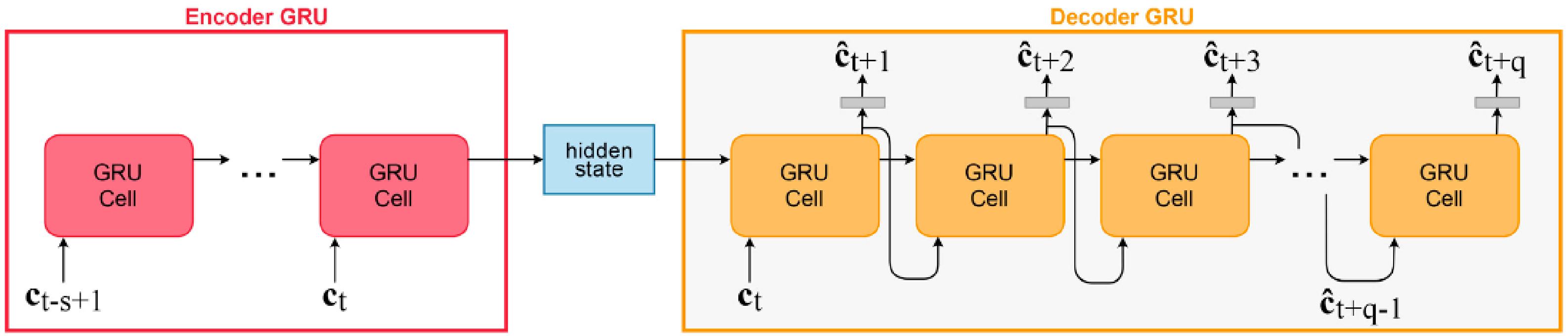


Fig 16. GRU Autoencoder for seq2seq[1]

[1] Pandey et al., "Direct Data-Driven Forecast of Local Turbulent Heat Flux in Rayleigh–Bénard Convection."

Experiments - TODO

- Experiment 3: Data Efficiency
  - omit x% from the dataset
- Experiment 4: Robustness to System Parameters
  - test model for increasing turbulent behavior (↑Rayleigh Number)
- Experiment 5: Improving Koopman Operator
  - sparse eigenvalues -> decrease observable dimensions
- Experiment 6: Explore Latent/Observable Space
  - visualize timestepping

Future Work

- control
  - Model-predictive Control
  - Reinforcement Learning
- going gaussian
  - variational autoencoders seemed to work better for dimension reduction
  - how to perform timestepping?
    - ELBO requires mean and variance

    → timestepping on mean and variance instead of z?
- working on full state instead of local convective field

Thank you for listening

Temperature field of numerical simulation of highly turbulent Rayleigh-Bénard convection (Rayleigh number: 10^{13})