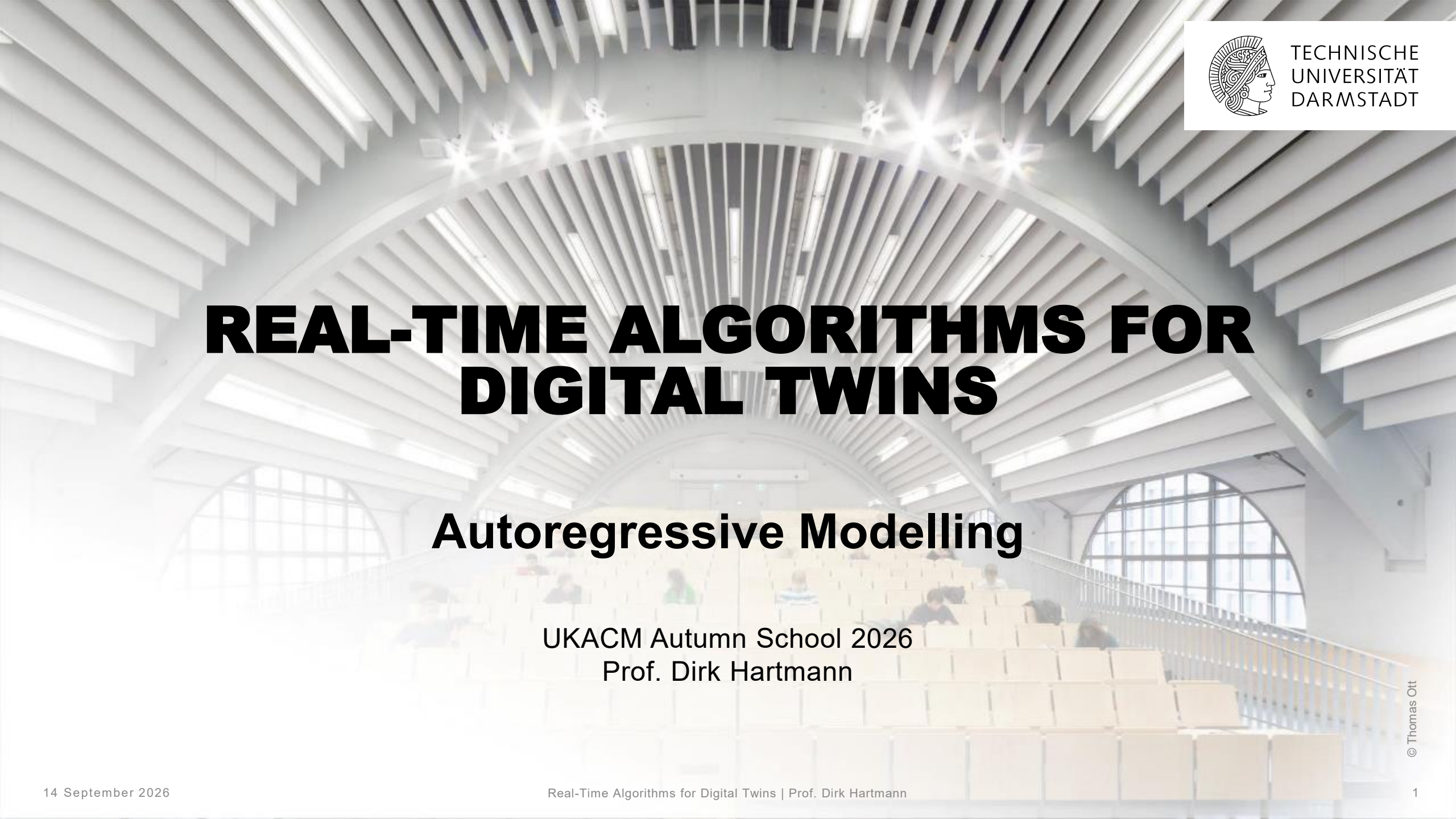
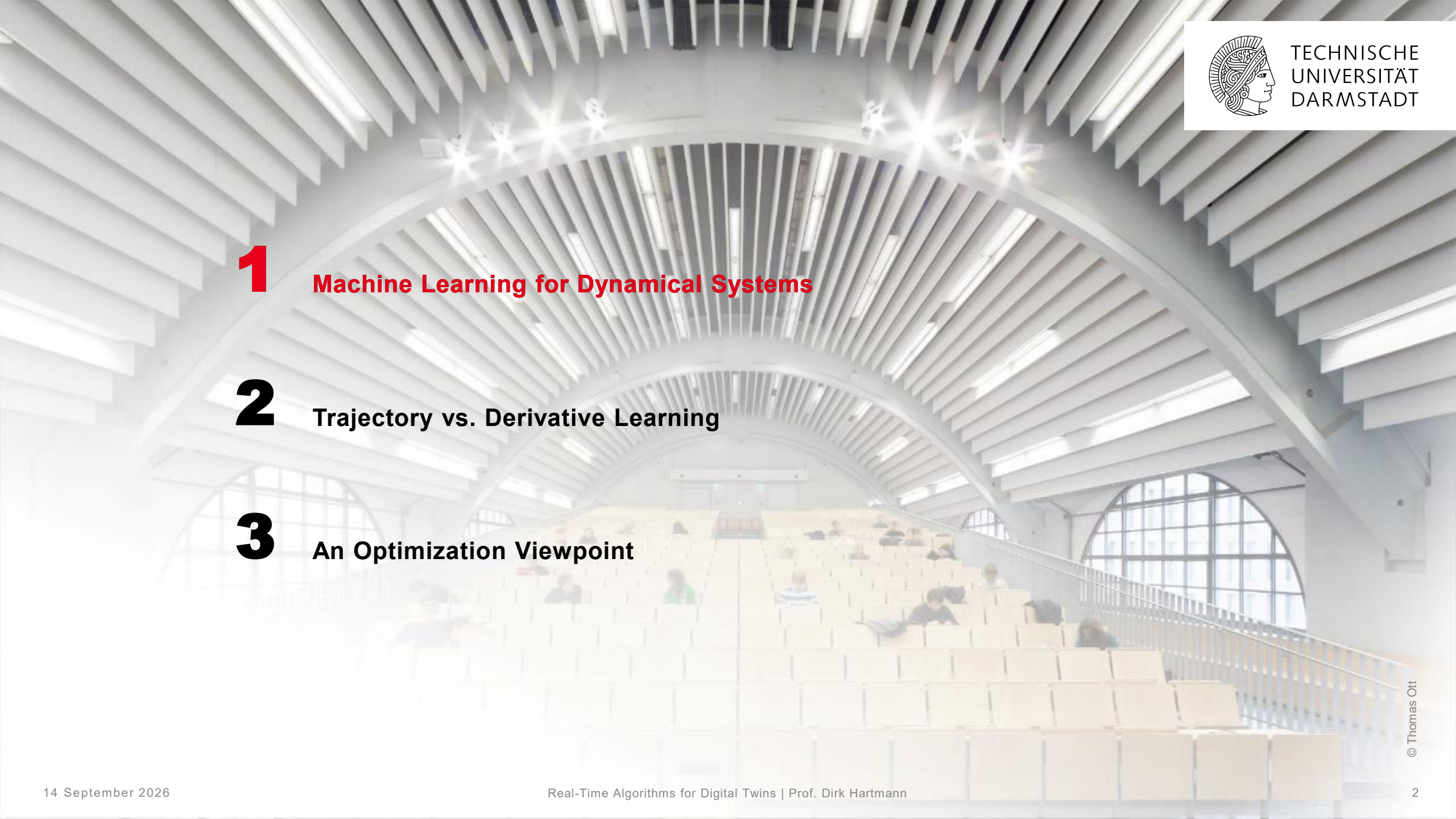


TECHNISCHE UNIVERSITÄT DARMSTADT

REAL-TIME ALGORITHMS FOR DIGITAL TWINS

Autoregressive Modelling

UKACM Autumn School 2026
Prof. Dirk Hartmann



TECHNISCHE UNIVERSITÄT DARMSTADT

1 Machine Learning for Dynamical Systems

2 Trajectory vs. Derivative Learning

3 An Optimization Viewpoint

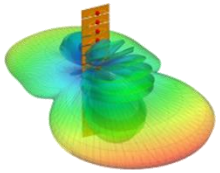
Machine Learning for Dynamical Systems

Dynamical System Identification (“Learning Dynamics from Observations”)

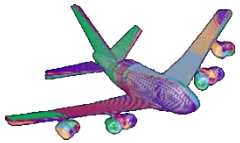
$$\frac{d}{dt}\boldsymbol{x} = f\,(\boldsymbol{x}, t, \boldsymbol{u})$$

- $\boldsymbol{x}$: **State**
- $\boldsymbol{u}$: (external, control) **Actuation**
- f: **Dynamics** — System of Differential Equations
- t: **Time**

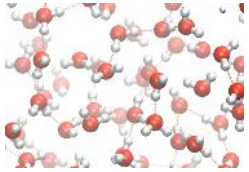
Motion



Electro-dynamics



Mechanics



Chemistry

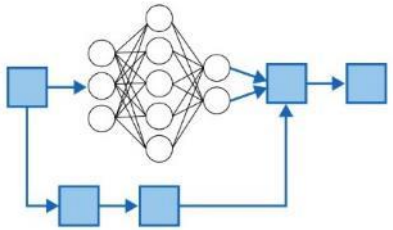
Fluid Dynamics

...

Task [Dynamical Systems Identification]:

Given a set of N_S trajectories $\left\{\left(\hat{x}_i^j, \hat{u}_i^j\right)\right\}_{i=0,\dots,N_T}^{j=0,\dots,N_S}$ learn a (part of a) model,

e.g., a Neural Network $\boldsymbol{NN}(x(t;\theta), t, u)$ such that it reproduces the given data.



Machine Learning for Dynamical Systems

Dynamical System Identification (“Learning Dynamics from Observations”)

TECHNISCHE UNIVERSITÄT DARMSTADT

- This is a typical natural sciences task: We observe what we are interested in and build systems that can recreate the observation. The latter we typically call reality, even though we can never obtain the true underlying state of the system.

- This is a typical engineering task: We look for proxy models of the true system so that tasks such as control and forecasting can be accomplished

- Many variables can be observed, e.g., in terms of sensor data, such that often massive data is available. Since Machines are usually better than humans when it comes to evaluate massive amounts of (heterogeneous) data, we can use machines to „learn“ the dynamical system.

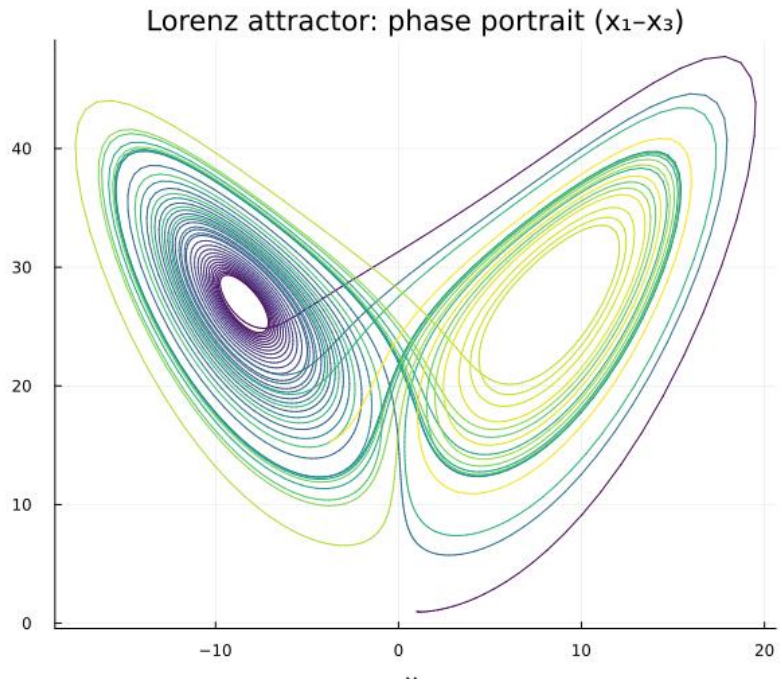
- However, in general inferring $f(x)$ is an ill-posed problem, thus this is typically a challenging endeavor.

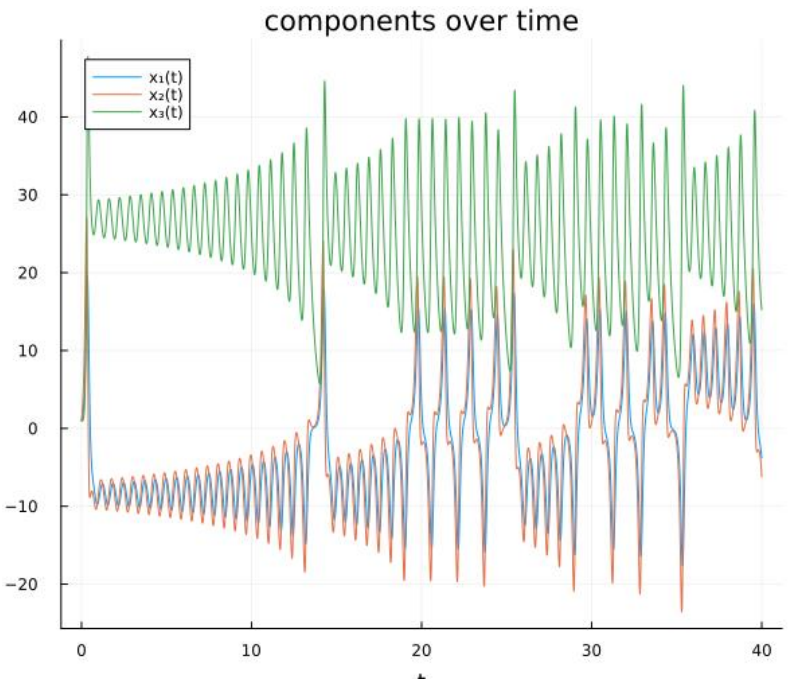
TECHNISCHE UNIVERSITÄT DARMSTADT

- To demonstrate the usefulness of Neural Networks in the context of application in dynamical systems, we will consider the Lorenz system

$$\partial_t x = \sigma(y - x)$$
$$\partial_t y = x(\rho - z) - y$$
$$\partial_t z = xy - \beta z$$

with parameters
$\sigma = 10$, $\rho = 28$, and $\beta = 8/3$.





Machine Learning for Dynamical Systems
Task

TECHNISCHE UNIVERSITÄT DARMSTADT

Task [Learning a Dynamical System]:

Given a set of k trajectories $\{x_0, x_1, \dots x_n\}^{\{i\}}$, $1 \leq i \leq k$, with $x_j^{\{i\}} = x^{\{i\}}(j\delta t)$, we would like to train a neural network such that

$$C = \sum_{i=1}^{k} \sum_{j=0}^{n-1} \frac{1}{2} \left\| x_{j+1}^{\{i\}} - F(x_j^{\{i\}}, p) \right\|_2^2$$

- Alternative formulations and architectures are possible as well, e.g.,

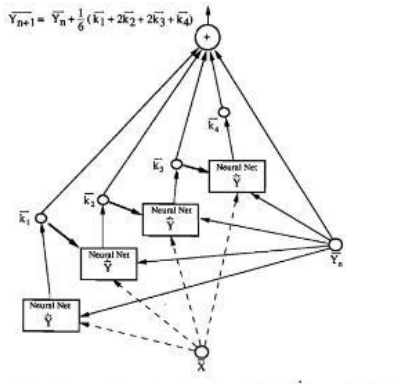
$$\mathrm{C} = \sum_{i=1}^{k} \sum_{j=0}^{n-1} \frac{1}{2} \left\| x_{j+1} - (x_j + \delta_t F(x_j, p)) \right\|_2^2,$$

  or even more complicated ones. Many, corresponding formulations have been inspired by successful numerical algorithms.

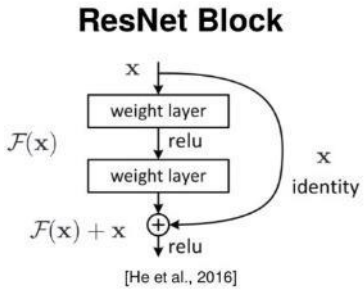
Machine Learning for Dynamical Systems
A zoo of approaches and architectures

TECHNISCHE UNIVERSITÄT DARMSTADT

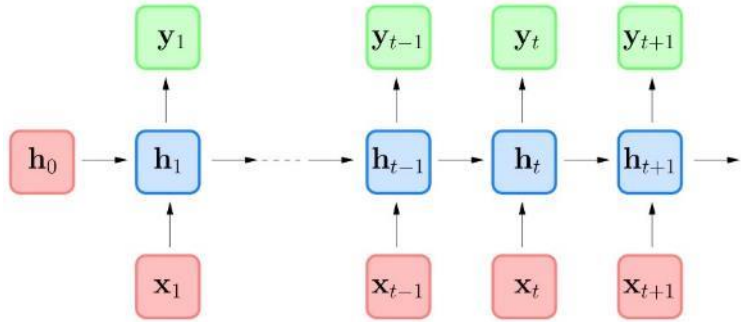
Runge-Kutte Neural Networks



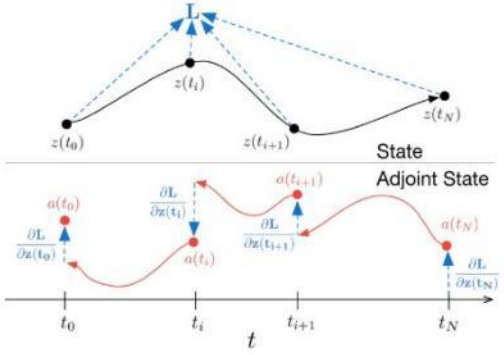
Residual Neural Networks

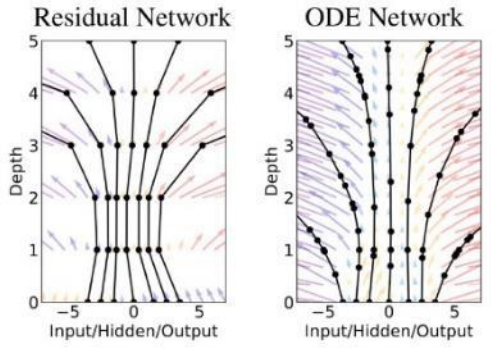


Recurrent Neural Networks



Neuro Ordinary Differential Equations

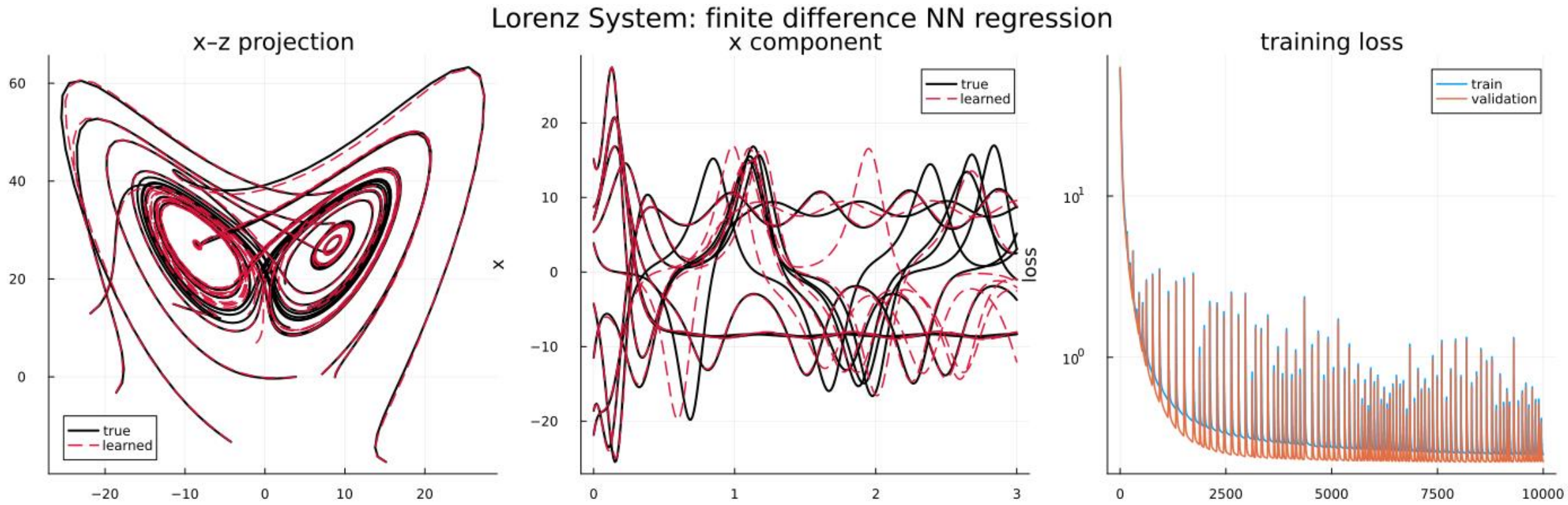




Source: R Rico-Martinez, KK rischer, I Kevrekidis, MC Kube, JL Hudson (1992). Discrete-vs. continuous-time nonlinear signal processing of Cu electrodissolution data. Chemical Engineering Communications, 118(1), 25-48.
RT Chen, Y Rubanova, J Bettencourt, DK Duvenaud (2018): Neural ordinary differential equations. *Advances in neural information processing systems*, *31*.
K He, X Zhang, S Ren, J Sun (2016). Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition*.
A Tealab (2018-12-01). "Time series forecasting using artificial neural networks methodologies: A systematic review. Future Computing and Informatics Journal. 3 (2): 334–340.

Machine Learning for Dynamical Systems

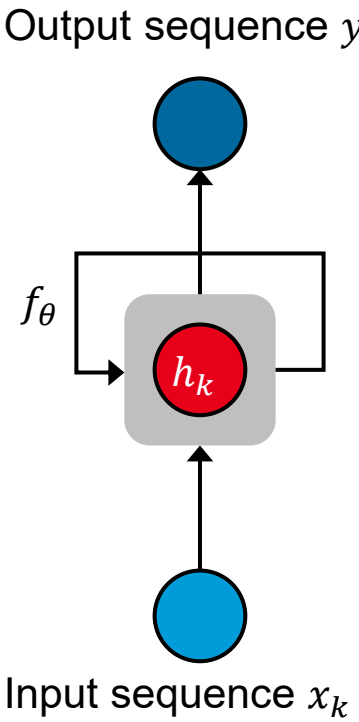
Example: Finite difference time step approximation for the Lorenz system



https://github.com/Dirk-Hartmann/real-time-digital-twins/blob/main/scripts/ml/Lorenz_NN_finitediff.jl

TECHNISCHE UNIVERSITÄT DARMSTADT

- Recurrent Neural Networks (RNN) emerged from the field of natural language processing, speech recognition, well before today's transformer architectures.
- Unlike Feedforward Neural Networks a the time history of the sequence or memory is used to train the neural network.
- The history of RNNs begins in the 1980s but only found a breakthrough with the Long Short-Term Memory (LSTM) architecture through its filtering architecture avoiding the vanishing gradient problem.
- Other prominent architectures are the Gated Recurrent Unit (GRU) or Echo State Networks (ESN).



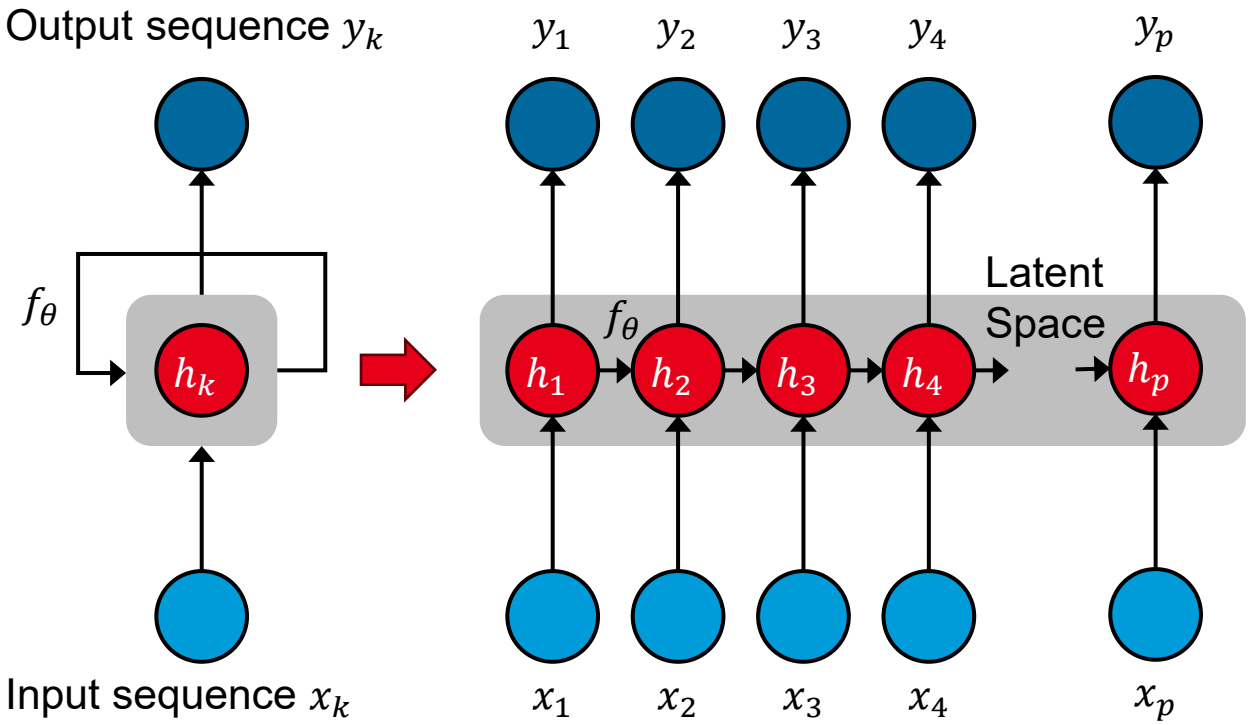
TECHNISCHE UNIVERSITÄT DARMSTADT

- RNNs are inspired from dynamical systems, i.e., they model flow maps
$$x_{k+1} = f(x_k, \theta) = f_\theta(x_k)$$
advancing a solution forward in time with $x_k = x(t_k)$ and $x_{k+1} = x(t_{k+1})$

- Different to the approach above, RNNs trains over a sequence of temporal snapshots, to allow for the history (memory) of the solution to shape the neural network model.

- Training over a sequence of snapshots the cost function becomes

$$C = \sum_{i=1}^{k} \sum_{j=0}^{n-p} \sum_{q=1}^{p} \frac{1}{2} \left\| y_{j+q}^{\{i\}} - F_\theta\left(F_\theta(\dots F_\theta(x_j)\dots)\right) \right\|_2^2$$



TECHNISCHE UNIVERSITÄT DARMSTADT

1 Machine Learning for Dynamical Systems

2 Trajectory vs. Derivative Learning

3 An Optimization Viewpoint

Trajectory vs Derivative Learning
Neural Ordinary Differential Equations (i)

TECHNISCHE UNIVERSITÄT DARMSTADT

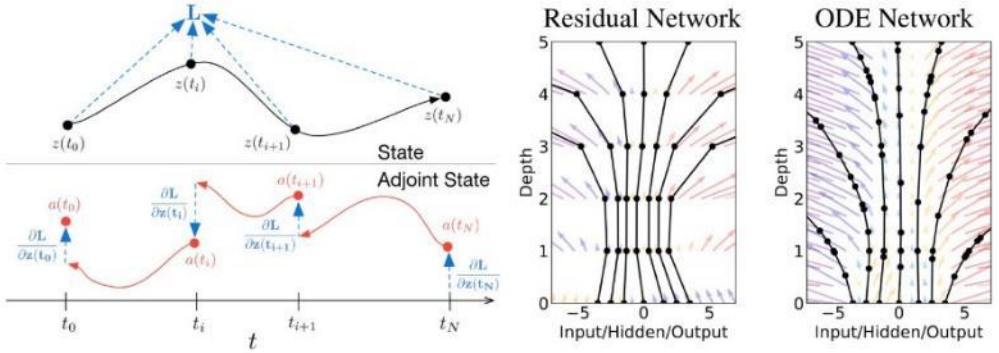
Task:

$$\frac{d}{dt}y(t) = \mathcal{NN}_{dyn}\big(y(t), u(t); w_{dyn}\big) t \in (0, T)$$
$$y(0) = y_0$$

where $\mathcal{N}_{dyn}$ is a (feed forward) neural network

Idea [Neuro Ordinary Differential Equations]:

a) Model continuous time series by learning a vector field
b) Memory-efficient training through adjoint method / trajectory learning



- "Bullshit that I and others have said about Neural ODEs", D. Duvenaud Reflecting on Neural ODEs | NeurIPS 2019 (youtu.be/YZ-_E7A3V2w)

Source: RT Chen, Y Rubanova, J Bettencourt, DK Duvenaud (2018): Neural ordinary differential equations. *Advances in neural information processing systems*, *31*.

Trajectory vs Derivative Learning
Neural Ordinary Differential Equations (ii)

TECHNISCHE UNIVERSITÄT DARMSTADT

- The core idea of the Neural ODE paper is effectively to promote trajectory based instead of derivative training.
- For a set of trajectories $\left\{\left(\hat{y}_i^j, \hat{u}_i^j\right)\right\}_{i=0,\dots,N_T}^{j=0,\dots,N_S}$:

Derivative Learning (Forward-Euler based)

$$w_{dyn}^* = \underset{w_{dyn}}{\operatorname{argmin}} \sum_{j=1}^{N\_s} \sum_{i=0}^{N_T-1} \left\| \frac{\hat{y}_{i+1}^j - \hat{y}_i^j}{t_{i+1} - t_i} - \mathcal{NN}_{dyn}(\hat{y}_i^j, \hat{u}_i^j; w_{dyn}) \right\|^2$$

Trajectory Learning

$$w_{dyn}^* = \underset{w_{dyn}}{\operatorname{argmin}} \sum_{j=1}^{N\_s} \sum_{i=0}^{N_T-1} \left\| \hat{y}_i^j - y^j(t_i) \right\|^2$$

$$\text{s.t. } y^j(t_i) = y^j(t_{i-1}) + (t_i - t_{t-1}) \cdot \mathcal{NN}_{dyn}\left(y^j(t_{i-1}), \hat{u}_{i-1}^j; w_{dyn}\right)$$

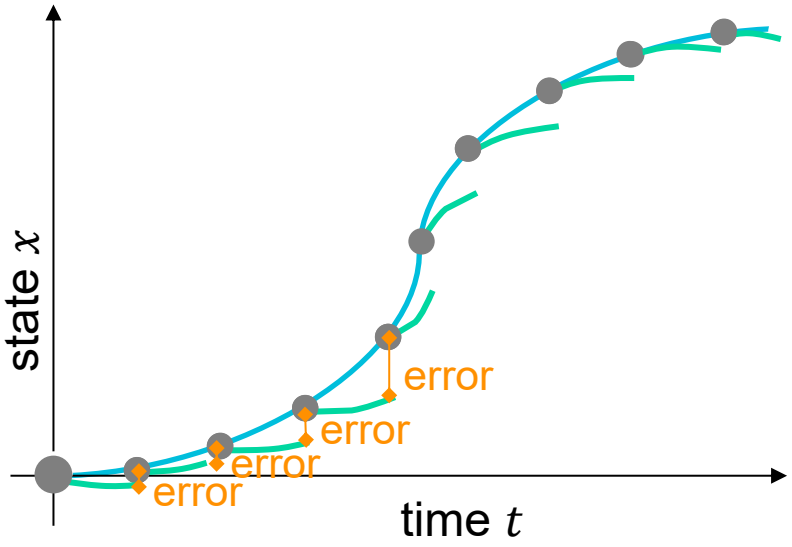
$$\text{and } y^j(0) = \hat{y}_0^j$$

Source: RT Chen, Y Rubanova, J Bettencourt, DK Duvenaud (2018): Neural ordinary differential equations. *Advances in neural information processing systems*, *31*.

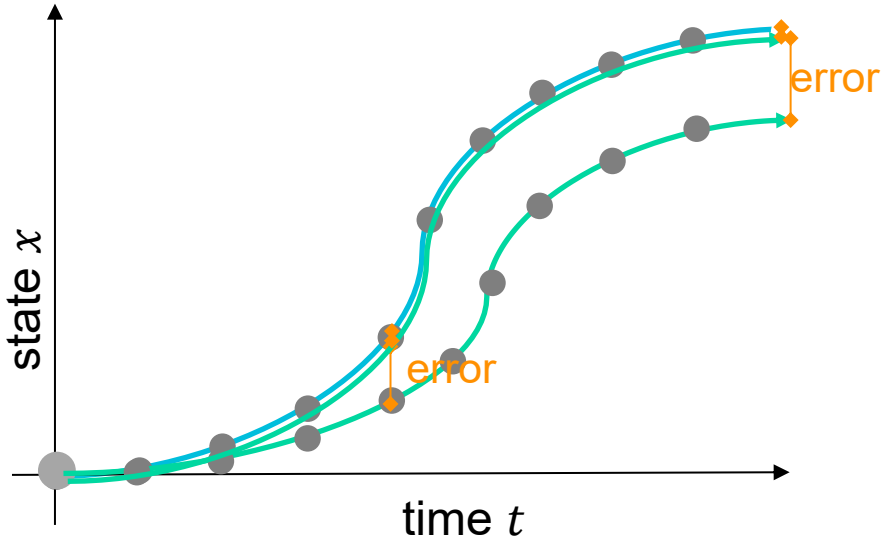
Trajectory vs Derivative Learning

- The core idea of the Neural ODE paper is effectively to promote trajectory based instead of derivative training.
- For a set of trajectories $\left\{\left(\hat{y}_i^j, \hat{u}_i^j\right)\right\}_{i=0,\ldots,N_T}^{j=0,\ldots,N_S}$:

Derivative Learning (Forward-Euler based)



Trajectory Learning



Source: RT Chen, Y Rubanova, J Bettencourt, DK Duvenaud (2018): Neural ordinary differential equations. *Advances in neural information processing systems*, *31*.

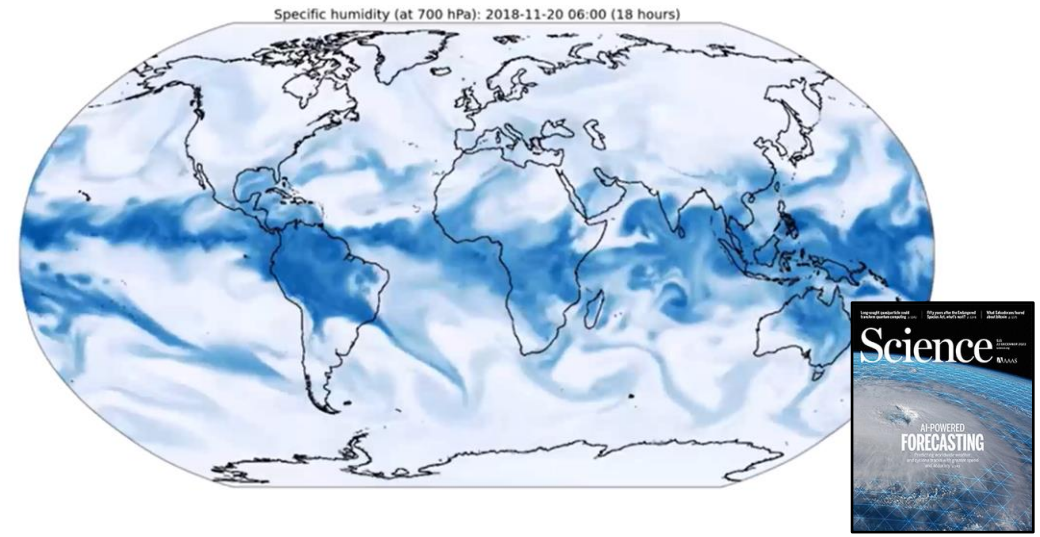
Trajectory vs Derivative Learning

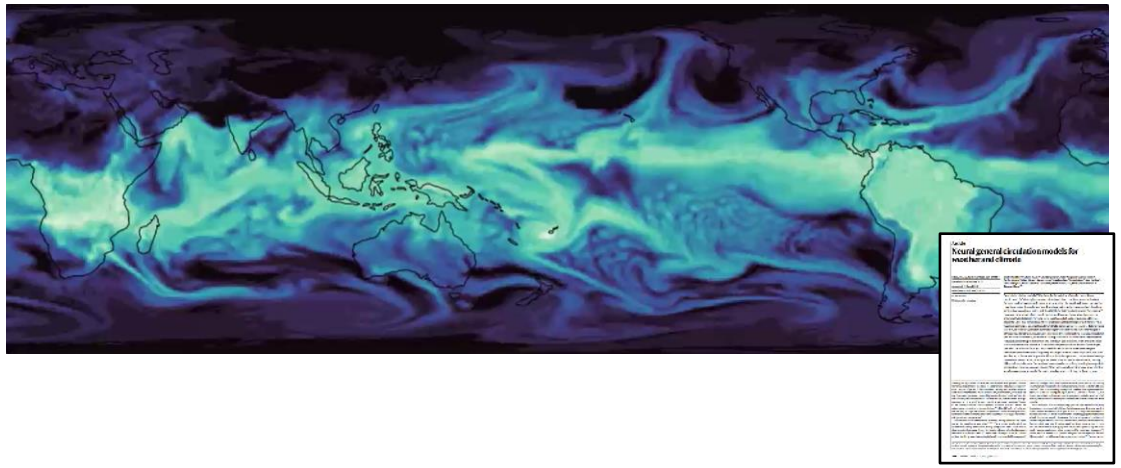
TECHNISCHE UNIVERSITÄT DARMSTADT

- Trajectory-based training is computationally more challenging but significantly reduces the effect of noise. Typically, less data is needed.
- Higher order time derivative estimation, advanced regularization approaches (e.g., total variational regularization[1]) or imposing additional structure (e.g., symmetries[2]) can typically improve derivative-based training.
- Trajectory-based training can be achieved via roll-outs and leveraging automatic differentiation of modern machine learning frameworks.
- Similar arguments of course hold for Recurrent Neural Networks, but Neural ODE provide more flexibility when it comes to variable time steps, integration in hybrid architectures, …
- Contrary to PINNS and other approaches, these approaches guarantee for a sequential nature of time (i.e., consistency with the arrow of time)

Source: 1) R Chartrand (2011): Numerical differentiation of noisy, non-smooth data. International Scholarly Research Notices, 2011(1), 164564.
2) L Gkimisis, IP Duff, P Goyal, P Benner (2025): On the representation of energy-preserving quadratic operators with application to Operator Inference. arXiv preprint arXiv:2503.10824.

Trajectory vs. Derivative Learning
Google GraphCast and Neural GCM

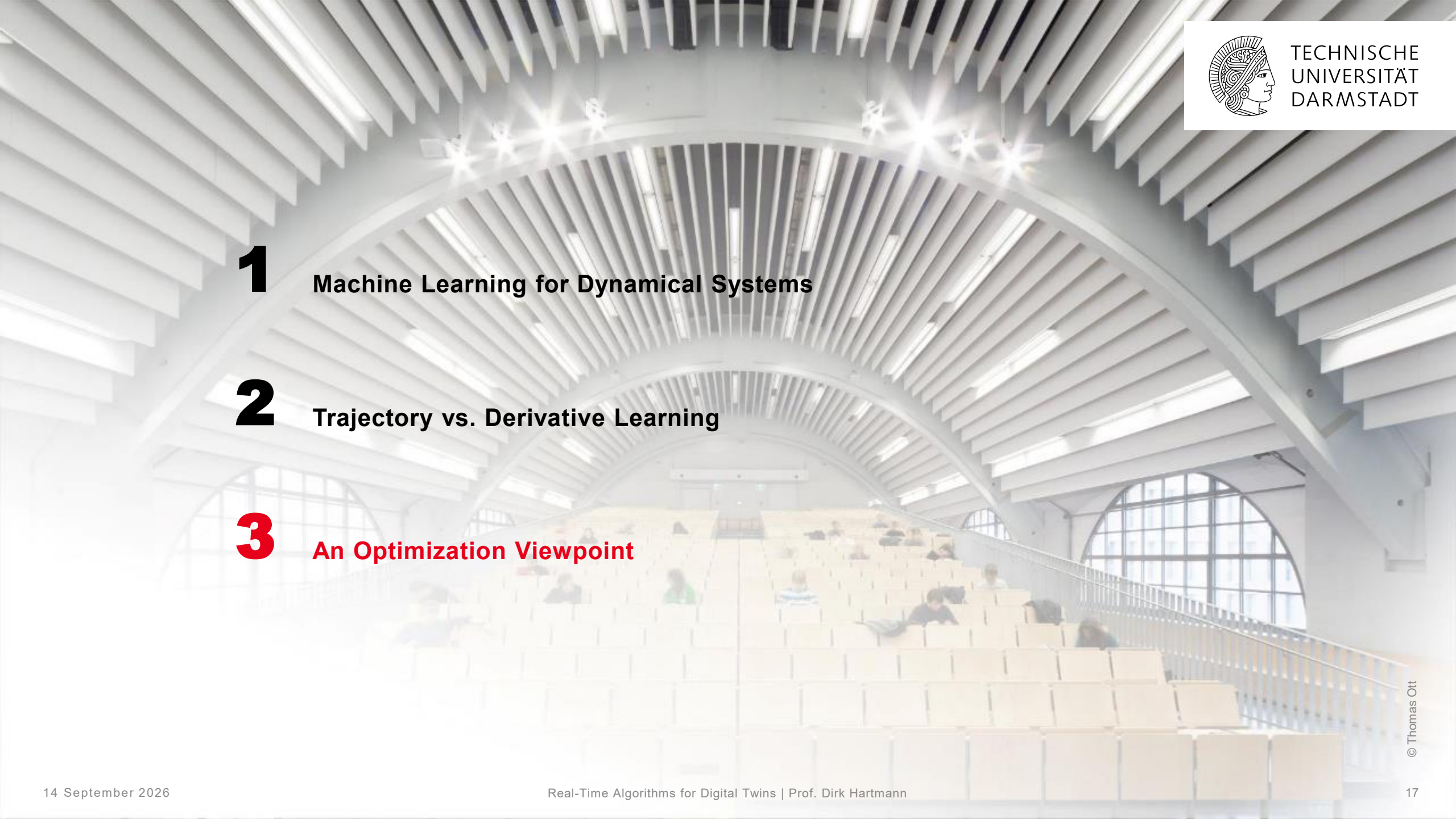




GraphCast (Dec 2022, DeepMind)
- Graph-based Surrogate Model
- Autoregressive Training
- Accurate 10-day Forecast

NeuralGCM (Nov 2023, Google Research)
- Hybrid Model Architecture
- Autoregressive Training
- Outperforms state-of-the-art forecast accuracy between 5 and 15 days

Source: R Lam, A Sanchez-Gonzalez, M Willson, P Wirnsberger, M Fortunato, F Alet, ... , P Battaglia (2023):.Learning skillful medium-range global weather forecasting. Science, 382(6677), 1416-1421.
D Kochkov, J Yuval, I Langmore, P Norgaard, J Smith, G Mooers ... & S Hoyer (2024): Neural general circulation models for weather and climate. *Nature*, *632*(8027), 1060-1066.



TECHNISCHE UNIVERSITÄT DARMSTADT

1 Machine Learning for Dynamical Systems

2 Trajectory vs. Derivative Learning

3 An Optimization Viewpoint

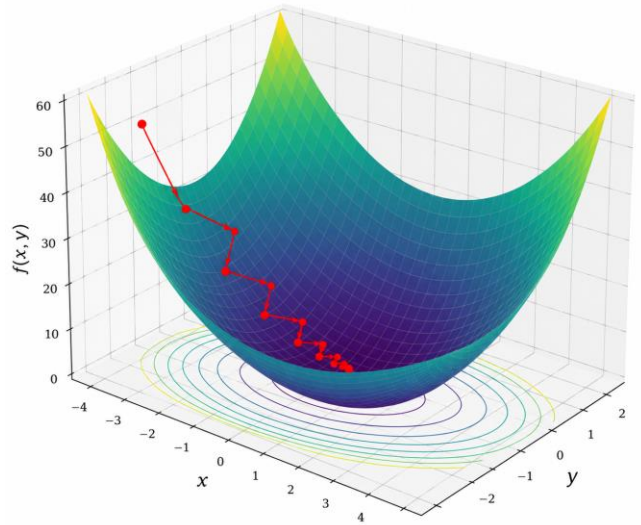
TECHNISCHE UNIVERSITÄT DARMSTADT

- Optimization algorithms are at the core of many engineering and machine learning principles.

Task [Unconstrained Optimization]:
Let $f: \mathbb{R}^n \to \mathbb{R}$ a continuous function. Find $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ such that

$$\tilde{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathbb{R}^n}{\operatorname{argmin}} f(\boldsymbol{x})$$

Respectively, $f(\tilde{\boldsymbol{x}}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$



Gradient descent algorithm applied to the function $f(x, y) = x^2 + 3y^2$.

- Optimization is a well-established field and specifically with the recent training algorithms for neural networks the field has seen some advanced gradient descent innovations.
- If we do not impose any additional structure (e.g., convexity) on f apart from differentiability, then we can only make statements about local minimizers.
- While efficient optimization routines are usually available as black boxes, it is often important to understand their weaknesses and strength to choose an optimal methods. Knowing the inner machinery is crucial to achieve this.

TECHNISCHE UNIVERSITÄT DARMSTADT

- So far, we have focused on **unconstrained optimization** problems,

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$$

  i.e., the "search space" has been the complete $\mathbb{R}^n$.

- However, often we would like to solve a **constrained optimization problem**. That is,

$$\min_{\boldsymbol{x}\in X} f(\boldsymbol{x})$$

  with $X = \{\boldsymbol{x} \in \mathbb{R}^n : g(\boldsymbol{x}) \leq 0, h(\boldsymbol{x}) = 0\}$, i.e., defined by equality and inequality constraints via continuous functions $g\colon \mathbb{R}^n \to \mathbb{R}^m$ and $h\colon \mathbb{R}^n \to \mathbb{R}^p$.

Idea [Reparameterization]: One option would be to reparametrize the set defined by $h(\boldsymbol{x}) = 0$ and optimize over the reparametrized set.

For example, Let $X = \{\boldsymbol{x} \in \mathbb{R}^2 : h(\boldsymbol{x}) = \boldsymbol{x}^2 - 1 = 0\}$. Choosing $\boldsymbol{x} = (\cos\theta\,, \sin\theta)$ allows to reformulate the constrained optimization problem $\min_{\boldsymbol{x}\in X} f(\boldsymbol{x})$ in a constrained optimization problem $\min_{\theta\in\mathbb{R}} f(\cos\theta\,, \sin\theta)$

TECHNISCHE UNIVERSITÄT DARMSTADT

Task [Trajectory Learning]:
We are given a model $\partial_t \boldsymbol{u}(t;\theta) = \boldsymbol{f}(\boldsymbol{u}(t;\theta),\theta)$ where we would like to fit the parameters θ such that the model solution $\boldsymbol{u}(t;\theta)$[1] matches a given trajectory $\boldsymbol{u}(t)$.

That is, we would like to solve the following optimization problem

$$\min_{\theta} \int_{\tau=0}^{T} (\boldsymbol{u}(t;\theta) - \boldsymbol{u})^2 d\tau \quad \text{with} \quad \partial_t \boldsymbol{u}(t;\theta) = \boldsymbol{f}(\boldsymbol{u}(t;\theta),\theta).$$

- The formulation can be generalized to more complex optimization formulations, e.g., $\min_{\theta} J(\boldsymbol{u}(t;\theta),\theta)$ with $\partial_t \boldsymbol{u}(t;\theta) = \boldsymbol{f}(\boldsymbol{u}(t;\theta),\theta)$.
- Usually, machine learning approaches heavily exploit automatic differentiability provided by modern frameworks.
- This is actually an optimal control problem. A well-established mathematical framework is available. Using the KKT conditions (control community) or adjoint formulations (Shape and topology optimization community), the problem can be solved without any automatic differentiation.

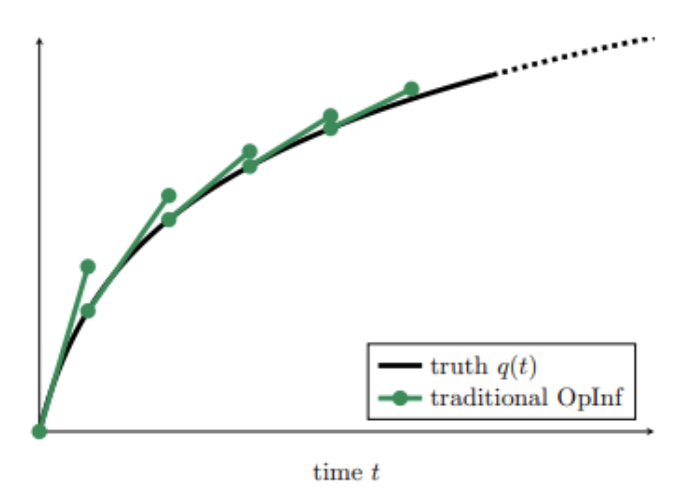
1) We explicitly highlight the dependence of $\boldsymbol{u}(t;\theta)$ on θ – if θ is changed, $\boldsymbol{u}(t;\theta)$likely also changes.

Optimization
Trajectory Fitting / Learning in the Context of Operator Inference

TECHNISCHE UNIVERSITÄT DARMSTADT

- For simplicity, let us consider for the moment a linear problem, i.e., $H \equiv 0$ and $B \equiv 0$

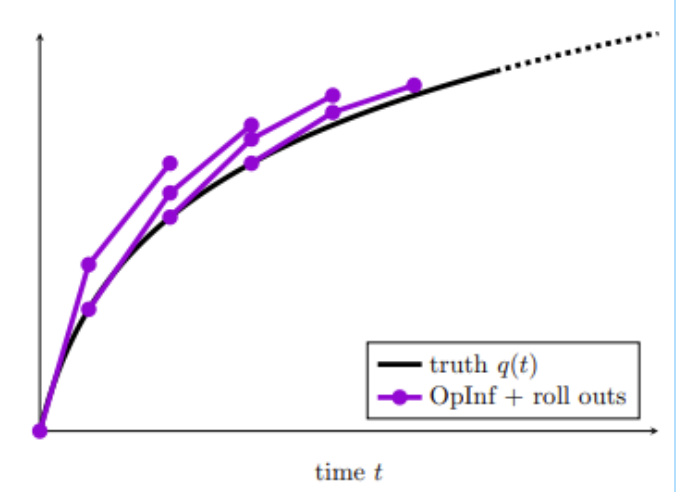
Derivative Fitting – Operator Inference



Least-Square Optimization Problem:

$$\arg\min_A \sum_i (\partial_t x_i - A x_i)^2$$

Trajectory Fitting – Operator Inference



Constraint Optimization Problem:

$$\arg\min_A \sum_i (x_i - \tilde{x}_i)^2$$

such that $\partial_t \tilde{x} = A\tilde{x}$

Source: W Uy, D Hartmann, B Peherstorfer(2023): Operator inference with roll outs for learning reduced models from scarce and low-quality data; Comput. Math. Appl..

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr. Dirk Hartmann

S2|17
Schlossgartenstraße 8
64289 Darmstadt

E-mail dirk.hartmann@tu-darmstadt.de

Homepage: https://dirk-hartmann.github.io
TU Darmstadt: https://www.temf.tu-darmstadt.de/digitaltwin
LinkedIn: https://www.linkedin.com/in/dirkhartmann

The Art of the Possible – Blog: https://blogs.sw.siemens.com/art-of-the-possible