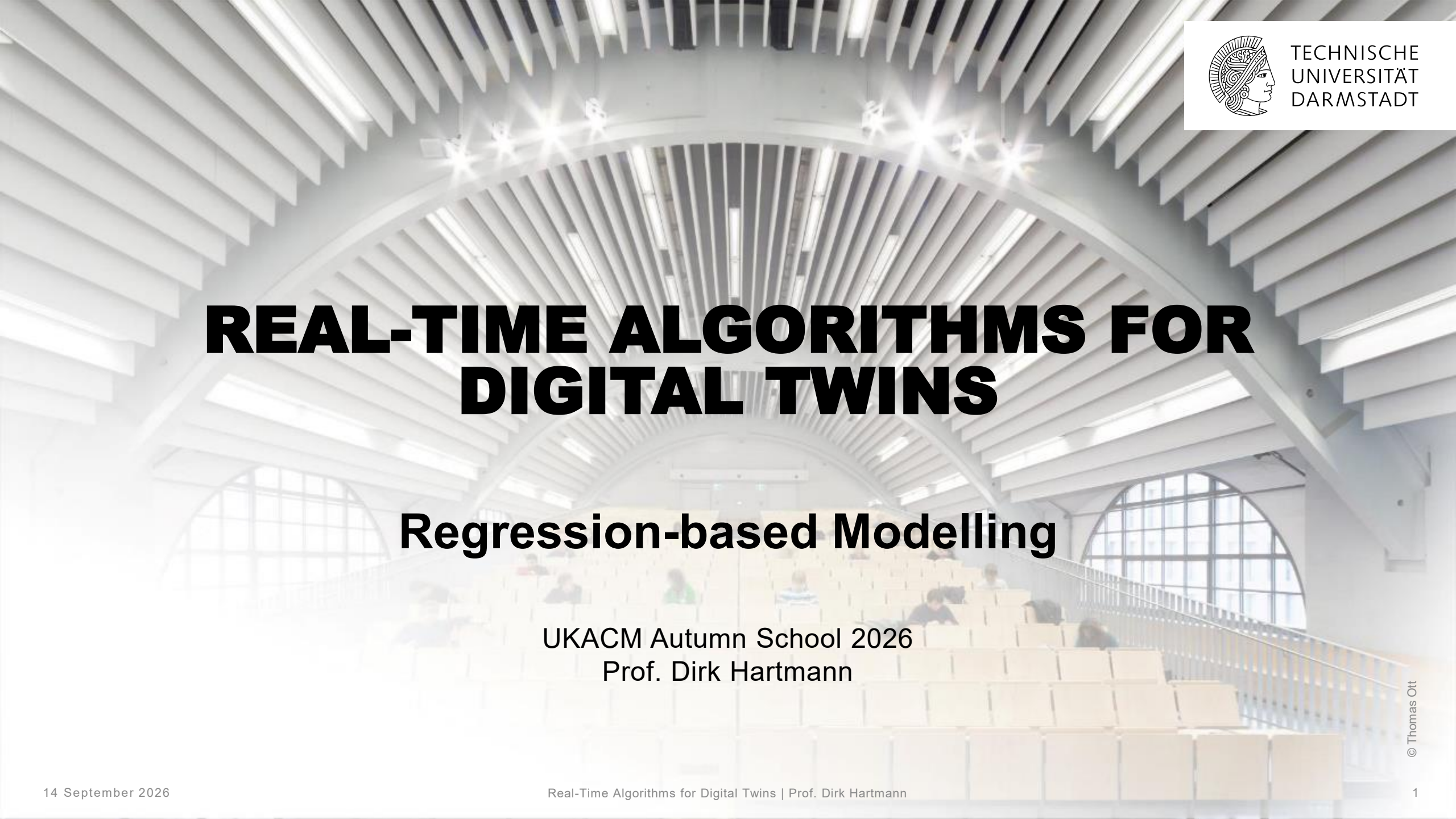


TECHNISCHE UNIVERSITÄT DARMSTADT

REAL-TIME ALGORITHMS FOR DIGITAL TWINS

Regression-based Modelling

UKACM Autumn School 2026
Prof. Dirk Hartmann

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Regression-based Modelling – Introduction

2 Singular Value Decomposition & Proper Orthogonal Decomposition

3 Operator Inference

4 Neural Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

Task [Regression]:
Regression aims to identify the relationship between independent variables $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ and dependent variables $\boldsymbol{Y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_n)$ and some unknown parameters $\boldsymbol{\beta}$, i.e.,

$$\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\beta}),$$

where the regression function $\boldsymbol{f}(\cdot)$ is typically prescribed and the parameters $\boldsymbol{\beta}$ are found by optimizing the *goodness-of-fit* of this function to the data.

► **Goodness-of-fit:**

Maximum Error l_∞ : $E_\infty(\boldsymbol{f}) = \max_{1<k<n} |\boldsymbol{f}(\boldsymbol{x_k}) - \boldsymbol{y_k}|$

Mean Absolute Error l_1 : $E_1(\boldsymbol{f}) = \frac{1}{n}\sum_{k=1}^{n} |\boldsymbol{f}(\boldsymbol{x_k}) - \boldsymbol{y_k}|$

Least Squares Error l_2 : $E_2(\boldsymbol{f}) = \left(\frac{1}{n}\sum_{k=1}^{n} |\boldsymbol{f}(\boldsymbol{x_k}) - \boldsymbol{y_k}|^2\right)^{1/2}$

TECHNISCHE UNIVERSITÄT DARMSTADT

- Curve fitting is the most basic regression technique usually resulting in solutions that com from solving a linear system

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

Example [Least Squares Polynomial Fitting]
Given data (x_k, y_k) choose the parameter β to fit the curve $f(x) = \beta_0 + \beta_1 x + \cdots + \beta_m x^m$ such that $E_2(f) = \sum_{k=1}^{n} |f(x_k) - y_k|^2$ is minimal.

- In the minimum it holds $\partial \beta_i E_2(f) = 0$ for $0 \leq i \leq m$, which leads to a linear system $\boldsymbol{A} \cdot \boldsymbol{\beta} = \boldsymbol{b}$, which can be solved effectively, i.e., do not need any optimization algorithm.

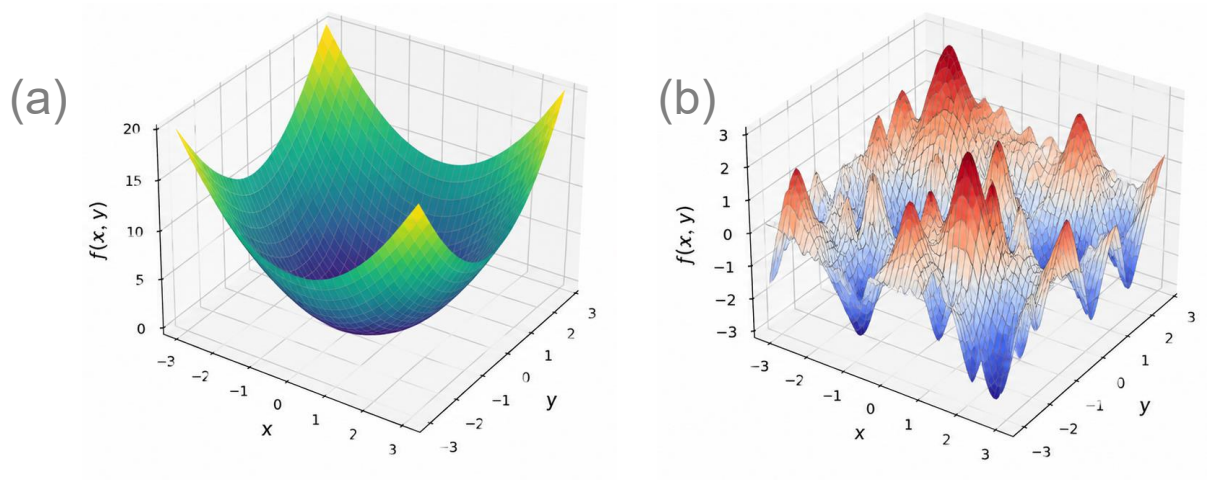
- Least-squares fitting has the critical advantage, that the optimization is often inexpensive.

TECHNISCHE UNIVERSITÄT DARMSTADT

- Choosing a non-linear function , e.g., $f(x) = \beta_1 \cos(\beta_2 x + \beta_3) + \beta_4$, the system to be solved becomes nonlinear and appropriate optimization algorithms are required.

- Nonlinear systems can have no solutions, several solutions, as well as an infinite number of solutions. Contrary to linear systems, the existence and potential uniqueness is more difficult to check.

- Convex functions are a class of functions where corresponding statements can be made. They also guarantee convergence, while non-convex functions can have many pitfalls



Two objective function landscapes representing (a) a convex function and (b) a non-convex function.

Source: SL Brunton, JN Kutz (2019): Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press

TECHNISCHE UNIVERSITÄT DARMSTADT

- In classical regression, we often work with underdetermined or overdetermined (linear) systems leading to the following optimization problems

$$\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmin}}(\|\boldsymbol{A} \cdot \boldsymbol{x} - \boldsymbol{b}\|_2 + \lambda g(\boldsymbol{x})) \qquad \text{[Overdetermined System]}$$

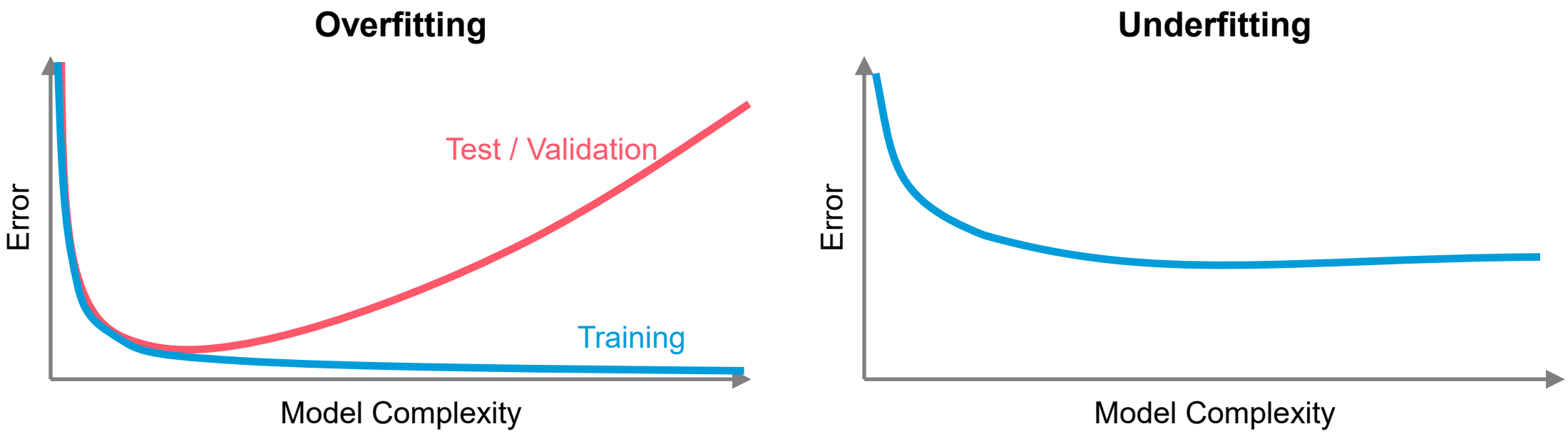
$$\text{or} \quad \boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, g(x) \quad \text{subject to} \quad \|\boldsymbol{A} \cdot \boldsymbol{x} - \boldsymbol{b}\|_2 \leq \epsilon, \qquad \text{[Underdetermined System]}$$

  where $g(x)$ is a given penalization – also known as regularization – with penalty parameter λ for over-determined systems, e.g., $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$

- Oten the exact family of curve is not known, e.g., the detailed polynomial degree, and optimization methods are used to choose the best model.

- A concern thus is to not overfit or underfit the data.

Model Selection
Overfitting and Underfitting



Prototypical behavior of over- and underfitting data. ***Overfitting:*** *Increasing the number of training iterations leads to an improved reduction on the training data while increasing the error on the validation / test data.* ***Underfitting:*** *the error performance is limited due to the restrictions on model complexity,*

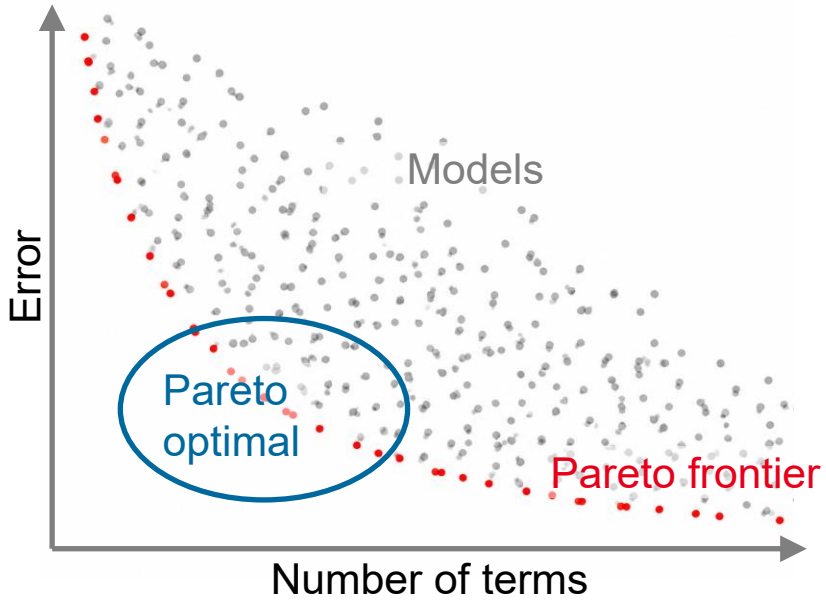
Source: SL Brunton, JN Kutz (2019): Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press

TECHNISCHE UNIVERSITÄT DARMSTADT

- **How to choose the right model?**
- **Occam's razor / Lex Parsimoniae:** Among competing hypotheses, the one with the fewest assumptions should be selected, or when you have two competing theories that make exactly the same predictions, the simpler one is the more likely.
- In regression, we optimize

$$\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmin}}(\|\boldsymbol{A} \cdot \boldsymbol{x} - \boldsymbol{b}\|_2),$$

  which does not explicitly enforce any constraints on $\boldsymbol{x}$.



- To have some control for avoiding under or over fitting one typically controls the "complexity" of $\boldsymbol{x}$ by appropriate regularization, i.e.,

$$\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A} \cdot \boldsymbol{x} - \boldsymbol{b}\|_2 + \lambda_1 \|\boldsymbol{x}\|_1 + \lambda_2 \|\boldsymbol{x}\|_2$$

  where the parameters λ_1 and λ_2 control the penalization of norms.

Source: SL Brunton, JN Kutz (2019): Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Regression-based Modelling – Introduction

2 Singular Value Decomposition & Proper Orthogonal Decomposition

3 Operator Inference

4 Neural Networks

Singular Value Decomposition
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Singular Value Decomposition is among the most important matrix factorizations in the computational era. It is a cornerstone of science and engineering.[1] It is the basis for many techniques in dimension reduction.

- In the context of Data-Driven methods we are interested in analyzing large data sets $X \in \mathbb{R}^{n \times m}$:

$$\boldsymbol{X} = \begin{bmatrix} | & & | \\ \vec{x}_1 & \dots & \vec{x}_m \\ | & & | \end{bmatrix}$$

  where each data point $1 \leq i \leq m$ is an n-dimensional vector, i.e., $\vec{x}_i \in \mathbb{R}^n$. If not given naturally in this form, many data can be transferred to this.

- Often the individual data points will represent simulation snapshots – a time series of simulation data. That is, $\vec{x}_k = \vec{x}(k\Delta t)$ and $n \gg m$.

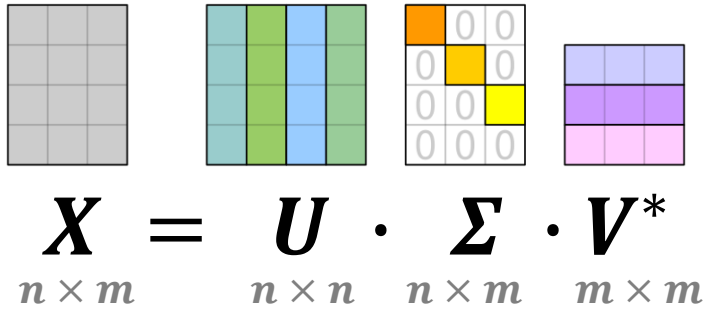
1) GW Stewart (1993): On the Early History of the Singular Value Decomposition. SIAM Review

Singular Value Decomposition
Definition

Singular Value Decomposition (SVD): The SVD is a unique matrix decomposition that exists for every complex-valued matrix $\boldsymbol{X} \in \mathbb{C}^{n \times m}$ such that

$$\boldsymbol{X} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^*.$$

Here, $\boldsymbol{U} \in \mathbb{C}^{n \times n}$ and $\boldsymbol{V} \in \mathbb{C}^{m \times m}$ are unitary matrices with orthonormal columns / rows and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$ is a real non-negative diagonal matrix.



- The columns of $\boldsymbol{U}$ are called the left singular vectors of $\boldsymbol{X}$, the columns of $\boldsymbol{V}$ singular vectors of $\boldsymbol{X}$, and entries in $\boldsymbol{\Sigma}$ singular values (usually arranged in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_m \geq 0$)

$$\boldsymbol{X} = \sum_{k=1}^{m} \sigma_k \, \vec{u}_k \vec{v}_k.$$

- The SVD is essentially a separation of variables between the columns and rows of a data matrix, where *U encodes spatial patterns while V encodes temporal patterns.*

Singular Value Decomposition
Computation

TECHNISCHE UNIVERSITÄT DARMSTADT

- The SVD is numerically stable and albeit similarities to eigen decomposition is guaranteed to exist for any matrix.

- Efficient implementations for SVD are available in nearly all modern computer languages, thus we will take this for granted. (Albeit the specific implementations are important and also enlightening).

```
43
44  #Assemble the data matrix
45  φ, φ_t, φ_v = assemble_data_matrices(sol);
46
47  #Perform an SVD
48  U,Σ,V = svd(φ_t);
49
50  #Analyze the SVD / Plot SVD information
51  plot_svd_spectrum("Fig\\svd_spectrum.png", Σ);
52  plot_pod_error("Fig\\pod_error.png", U, φ_t, φ_v)
53
```

Singular Value Decomposition
Optimal low-rank Matrix Approximations (i)

TECHNISCHE UNIVERSITÄT DARMSTADT

- SVD provides a hierarchy of **optimal low-rank approximations** $\widetilde{\boldsymbol{X}}$ to the matrix $\boldsymbol{X}$, keeping the leading r values and vectors and disregarding the rest (often σ_k decrease rapidly):

$$\widetilde{\boldsymbol{X}} = \sum_{k=1}^{r} \sigma_k \, \vec{u}_k \vec{v}_k .$$

Theorem [Eckart-Young]: The optimal rank-r approximation to X is given by the rank-r SVD truncation $\widetilde{\boldsymbol{X}}$:

$$\underset{\widetilde{\boldsymbol{X}},\, \text{s.t. } \operatorname{rank}(\widetilde{\boldsymbol{X}})=r}{\operatorname{argmin}} \left\| \boldsymbol{X} - \widetilde{\boldsymbol{X}} \right\|_F = \widetilde{U} \cdot \tilde{\Sigma} \cdot \tilde{V}^* ,$$

with the Frobenius norm $\|\boldsymbol{X}\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \left| x_{ij} \right|^2 \right)^{1/2}$.

- It is possible to exactly quantify the error, i.e., $\left\| \boldsymbol{X} - \widetilde{\boldsymbol{X}} \right\|_{\boldsymbol{F}}^{\boldsymbol{2}} = \sum_{k=r+1}^{m} \sigma_k^2$.

- The corresponding relative error, can be interpreted as the fraction of kinetic energy missing in the approximation $\widetilde{\boldsymbol{X}}$.

Singular Value Decomposition
Optimal low-rank Matrix Approximations (ii)

TECHNISCHE UNIVERSITÄT DARMSTADT

- The SVD also provides an optimal rank-r approximation in the matrix 2 norm / Spectral norm, i.e.,

$$\underset{\tilde{X},\ \text{s.t.}\ \operatorname{rank}(\tilde{X})=r}{\operatorname{argmin}} \left\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\right\|_2 = \widetilde{\boldsymbol{U}}\cdot\widetilde{\boldsymbol{\Sigma}}\cdot\widetilde{\boldsymbol{V}}^{\star}$$

  with $\|\boldsymbol{X}\|_2 = \max_{\vec{v}\neq 0} \frac{\|\boldsymbol{X}\cdot\vec{v}\|_2}{\|\vec{v}\|_2}$

- The error expression is rather simple

$$\left\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\right\|_2 = \sigma_{r+1}$$

which follows from the expansion of the SVD and orthogonality of vectors $\vec{v}_k$.

- There are many applications of SVD, e.g.,
  - Linear Algebra: Improve the condition number of matrices (effectively cutting off the small singular values)
  - Data Analysis: Data transformation to more interpretable representations (Principal Component Analysis)
  - Reduced Order Modelling: Identification of dominant dynamical modes in which the system can be represented
  - …

…

Proper Orthogonal Decomposition (POD)

TECHNISCHE UNIVERSITÄT DARMSTADT

Proper Orthogonal Decomposition:

Given a dynamical system

$$\partial_t \vec{x}(t) = \boldsymbol{A} \cdot \vec{x}(t) + \vec{F}\big(\vec{x}(t)\big) + \boldsymbol{B}c(t)$$

and trajectory data $\boldsymbol{X} = [\vec{x}(t_0), \dots, \vec{x}(t_m)]$, we use SVD $\boldsymbol{X} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^T$ to obtain the dominant orthogonal components $\widetilde{\boldsymbol{U}}$. We then model the system only in these dominant modes, i.e., $\vec{x}(t) = \sum_{k=1}^{r} a_k(t)\vec{u}_k$.

- Typically $\vec{x} \in \mathbb{R}^n$ with $n \gg 1$ obtained from the discretization of a PDE, i.e., the individual entries represent nodal valued.

- Once the modal basis is decided the governing equations for the $a_k(t)$ can be determined by projection of the full system. The corresponding Reduced Order Model is then

$$\partial_t \vec{\tilde{x}}(t) = \widetilde{\boldsymbol{A}} \cdot \vec{x}(t) + \vec{\tilde{F}}\left(\vec{\tilde{x}}(t)\right) + \widetilde{\boldsymbol{B}}u(t)$$

with $\vec{\tilde{x}}(t) = \boldsymbol{U}^\star \cdot \vec{x}(t)$, $\widetilde{\boldsymbol{A}} = \boldsymbol{U}^\star \cdot \boldsymbol{A} \cdot \boldsymbol{U}$, $\widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{U}}^\star$, and $\vec{\tilde{F}}\left(\vec{\tilde{x}}(t)\right) = \boldsymbol{U}^\star \cdot \vec{F}(\widetilde{\boldsymbol{U}} \cdot \vec{\tilde{x}}(t))$.

Proper Orthogonal Decomposition (POD)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reduced Order Model:

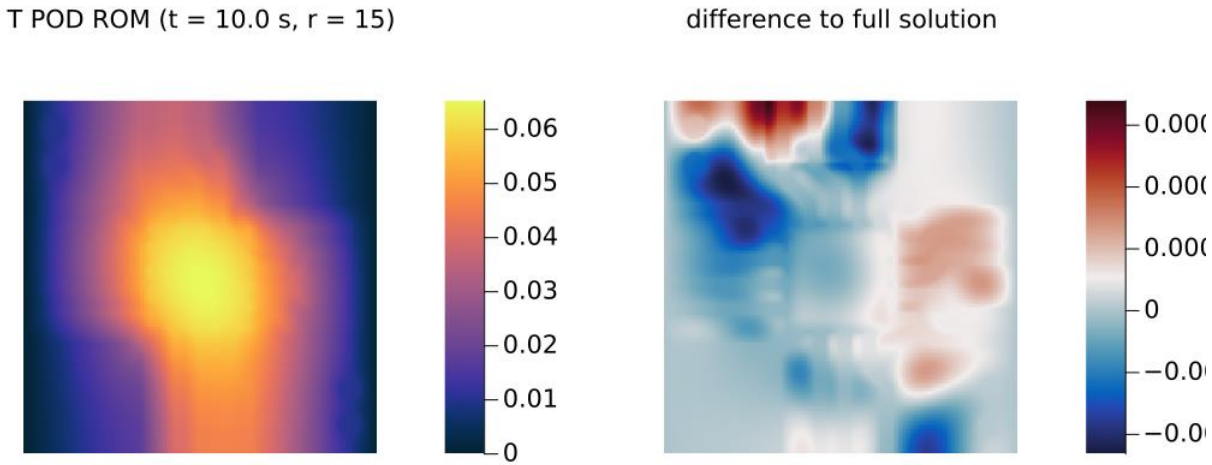
$$\partial_t \vec{\tilde{x}}(t) = \widetilde{\boldsymbol{A}} \cdot \vec{x}(t) + \vec{\tilde{F}}\left(\vec{\tilde{x}}(t)\right) + \widetilde{\boldsymbol{B}} u(t)$$

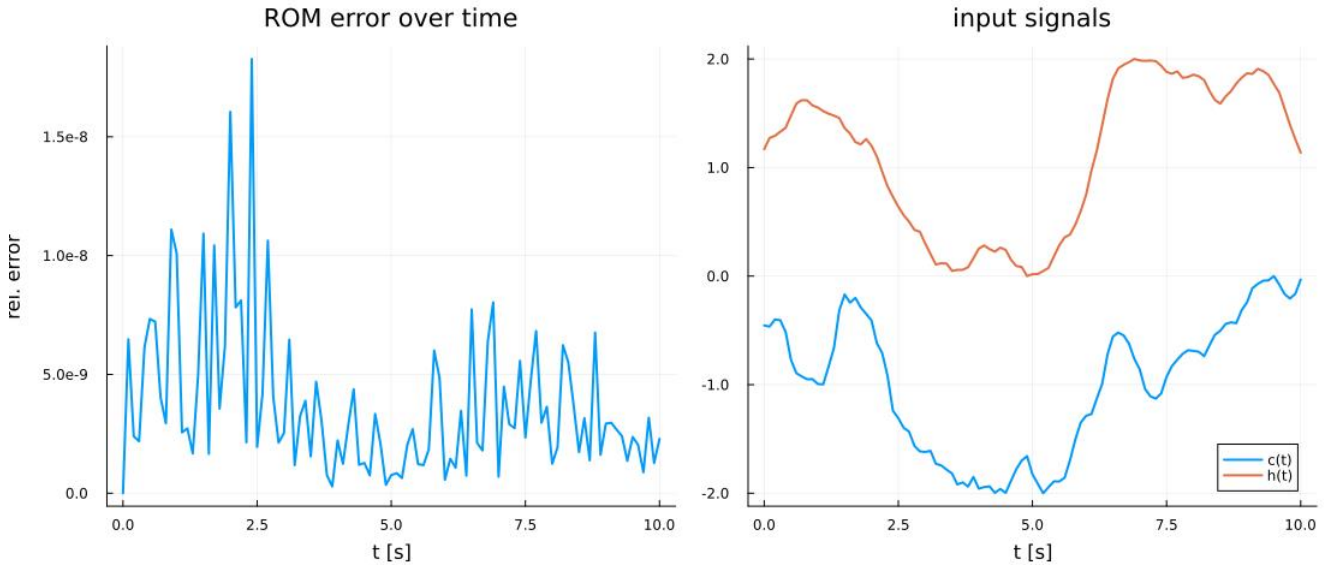
with $\vec{\tilde{x}}(t) = \boldsymbol{U}^\star \cdot \vec{x}(t)$, $\widetilde{\boldsymbol{A}} = \boldsymbol{U}^\star \cdot \boldsymbol{A} \cdot \boldsymbol{U}$, $\widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{U}}^\star$, and $\vec{\tilde{F}}\left(\vec{\tilde{x}}(t)\right) = \boldsymbol{U}^\star \cdot \vec{F}(\widetilde{\boldsymbol{U}} \cdot \vec{\tilde{x}}(t))$.

- If we have access to the full system / simulation code $\widetilde{\boldsymbol{A}} = \boldsymbol{U}^\star \cdot \boldsymbol{A} \cdot \boldsymbol{U}$ can be determined by simple matrix multiplication. However, the reduction of $\vec{\tilde{F}}\left(\vec{\tilde{x}}(t)\right) = \boldsymbol{U}^\star \cdot \vec{F}\left(\widetilde{\boldsymbol{U}} \cdot \vec{\tilde{x}}(t)\right)$ might be very costly.

- Under certain circumstances efficient methods for reduction / approximation of $\vec{\tilde{F}}\left(\vec{\tilde{x}}(t)\right) = \boldsymbol{U}^\star \cdot \vec{F}\left(\widetilde{\boldsymbol{U}} \cdot \vec{\tilde{x}}(t)\right)$ are available, these are referred to hyper reduction methods, one of these being *Discrete Empirical Interpolation (DEIM)* leveraging concepts from sparse sensing.

Reduced Order Modelling
Proper Orthogonal Decomposition + Galerkin Projection





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

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Regression-based Modelling – Introduction

2 Singular Value Decomposition & Proper Orthogonal Decomposition

3 Operator Inference

4 Neural Networks

Operator Inference
Idea

TECHNISCHE UNIVERSITÄT DARMSTADT

Proper Orthogonal Decomposition:

Given a dynamical system $\partial_t \vec{x}(t) = \boldsymbol{A} \cdot \vec{x}(t) + \vec{F}\left(\vec{x}(t)\right) + \boldsymbol{B}c(t)$ and trajectory data $\boldsymbol{X} = [\vec{x}(t_0), \dots, \vec{x}(t_m)]$, we use SVD $\boldsymbol{X} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^T$ to obtain the dominant orthogonal components $\widetilde{\boldsymbol{U}}$. We then model the system only in these dominant modes, i.e., $\vec{x}(t) = \sum_{k=1}^{r} a_k(t)\vec{u}_k$.

Operator Inference (OI):

Learn a quadratic model, i.e.,

$$\partial_t \vec{\tilde{x}}(t) = \tilde{A} \cdot \vec{x}(t) + \tilde{H} \cdot \left(\vec{\tilde{x}}(t) \otimes \vec{\tilde{x}}(t)\right) + \tilde{B}u(t)$$

with $\vec{\tilde{x}}(t) = U^\star \cdot \vec{x}(t)$, etc as usual and $P \otimes Q \equiv \left[p_{ij}Q\right]_{ij}$, via a regression problem

$$(A, H, B) = \underset{A,H,B}{\operatorname{argmin}} \left\| \partial_t X - [A\ H\ B] \cdot \begin{bmatrix} X \\ X * X \\ U \end{bmatrix} \right\|_F^2 + \mathcal{R}(A, H, B)$$

with the Khatri-Rao product $X * X \coloneqq [x_0 \otimes x_0, \dots, x_K \otimes x_k]$ and an appropriate regularization $\mathcal{R}(A, H, B)$.

Operator Inference
Idea

TECHNISCHE UNIVERSITÄT DARMSTADT

- OI yields an optimal approximation in the L_2 obtainable from the available data

Theorem [Willcox and Peherstorfer 2016]
In the limit of an infinite amount of stat, the operator inference model of order r recovers the POD model of order r in the case of a quadratic model.

- OI can be combined with hyper-reduction, specifically with Discrete Empirical Interpolation Method (DEIM)[1].
- Using the concept of lifting, one can cover any polynomial model, not only quadratic ones[2].
- Specific constraints on the matrices can be imposed to enforce stability[2,3], e.g., ensuring that eigenvalues of A are nonnegative.

Sources: 1) P Benner, P Goyal, B Kramer, B Peherstorfer, K Willcox (2020): Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. CMAME
2) E Qian, B Kramer, B Peherstorfer, K Willcox (2020): Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. Physica D: Nonlinear Phenomena
3) P Goyal, I Pontes Duff, P Benner (2023): Inference of Continuous Linear Systems from Data with Guaranteed Stability. arXiv
4) P Goyal, I Pontes Duff, P Benner (2023): Guaranteed Stable Quadratic Models and their applications in SINDy and Operator Inference. arXiv

Operator Inference
Challenges and Software

TECHNISCHE UNIVERSITÄT DARMSTADT

Challenges:

in addition to the challenges associated with the SVD

- Good estimates of time derivatives $\partial_t X$ in the data are crucial for success, since $[\mathrm{A}, \mathrm{H}, \mathrm{B}] = \underset{A,H,B}{\operatorname{argmin}} \|\partial_t X - \cdots\|_F^2$. In some circumstances very high approximation orders might be needed.

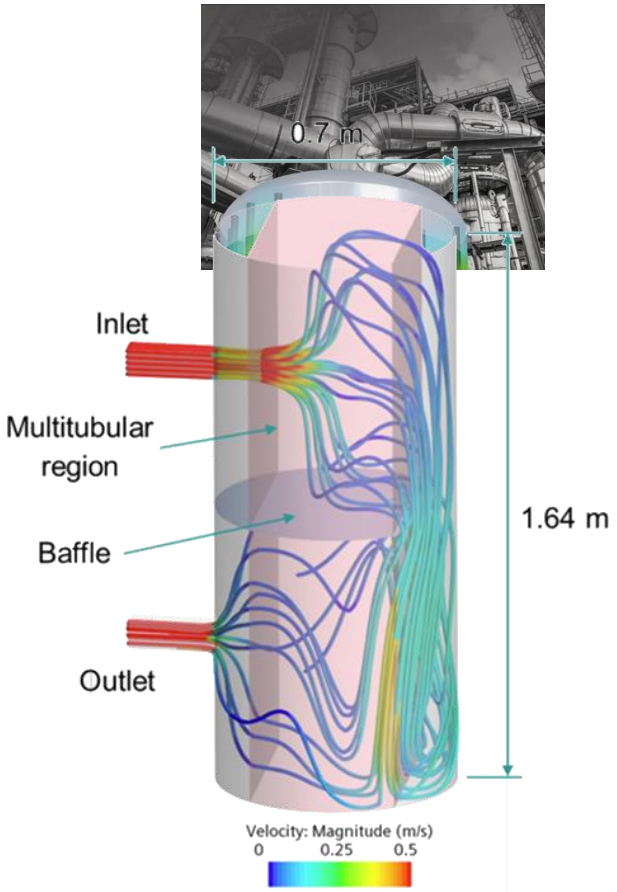
- The choice of the right regularization $\mathcal{R}(A, H, B)$ is challenging. Typically, this is done by automated hyper-parameter studies.

- Albeit the identification of the right operator structure should be straight forward, practice has shown that this is often a source of error due to limited knowledge of the underlying physics / systems.

Software:

- Availability of several open-source packages, e.g., the Operator Inference Libraries (for MATLAB and Python) are provided by Karen Willcox Lab, c.f., https://github.com/Willcox-Research-Group.

Operator Inference
Example (i)



Cooling flow velocities and pressure are modelled by Navier Stokes plus Brinkman law for the porous part

$$\rho_c \left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} \right) = -\nabla p + \mu \Delta \boldsymbol{v} + \boldsymbol{\alpha}(\boldsymbol{x})\boldsymbol{v} \qquad \text{in } \Omega$$

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega$$

Cooling fluid temperature and solid temperature are modelled as isotropic mediums

$$\rho_c c_{p,c} \left(\frac{\partial T_c}{\partial t} + (\boldsymbol{v} \cdot \nabla) T_c \right) = \nabla \cdot (\boldsymbol{\mathcal{K}}_c \nabla T_c) + \chi_{\Omega_s} q(T_s - T_c) \qquad \text{in } \Omega$$

$$\rho_s c_{p,s} \frac{\partial T_s}{\partial t} = \nabla \cdot (\boldsymbol{\mathcal{K}}_s \nabla T_s) - q(T_s - T_c) + \mathcal{P}(t, T_s) \qquad \text{in } \Omega_s$$

Source: D. Hartmann, L. Failer (2021): A Differentiable Solver Approach to Operator Inference. arXiv

Data-based Hyper-Reduction
Alternative Methods

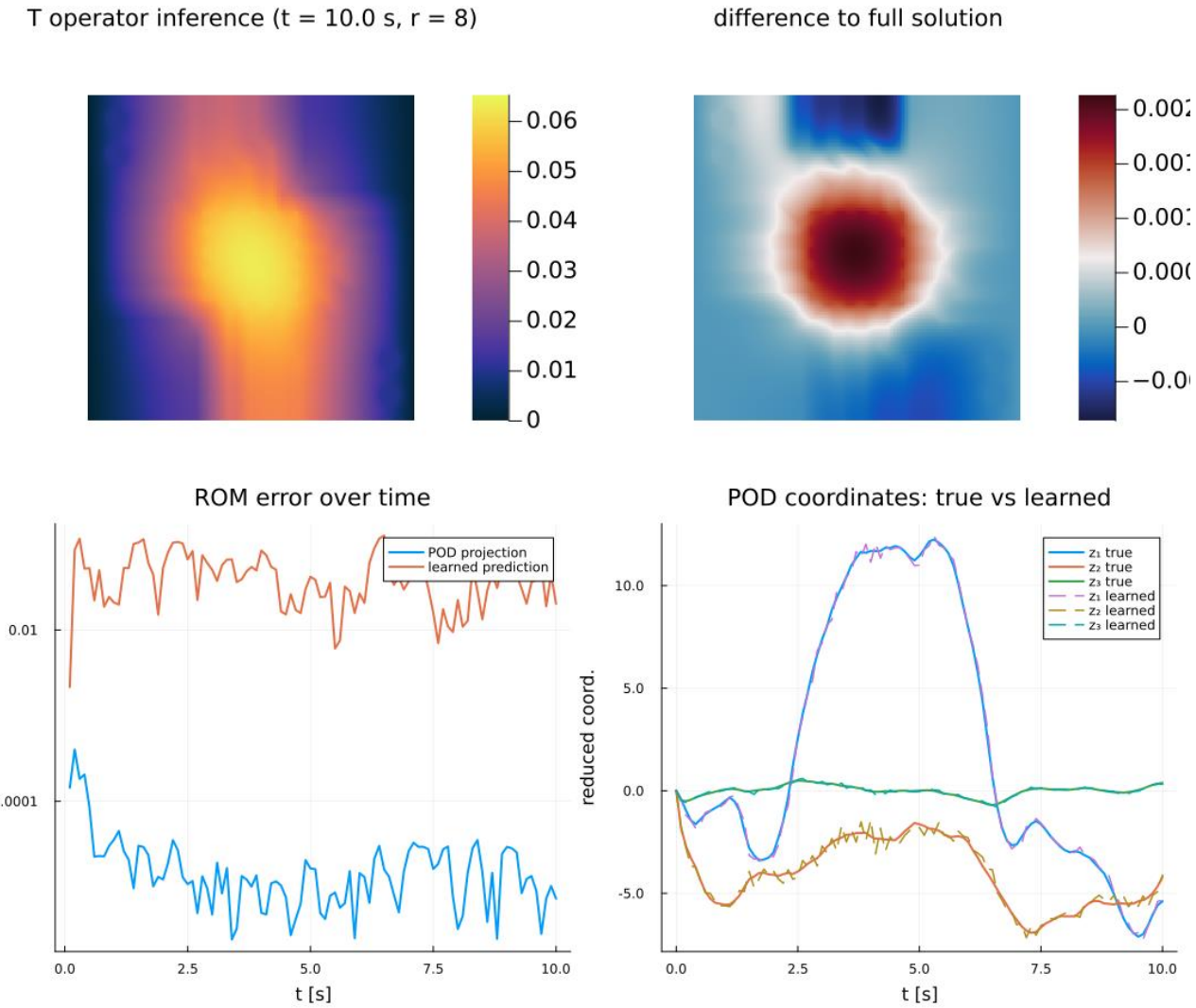
TECHNISCHE UNIVERSITÄT DARMSTADT

- The field of Operator-Inference is a very active field of research with many more new developments, e.g., non-linear POD extensions, Exact Operator Inference.

- Operator-Inference is just one of many other options such as Dynamic Mode Decompostion or **Sparse** Identification of Nonlinear Dynamics (SINDy).

Source: SL Brunton, JN Kutz (2019): Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press

Reduced Order Modelling
Proper Orthogonal Decomposition + Operator Inference



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

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Regression-based Modelling – Introduction

2 Singular Value Decomposition & Proper Orthogonal Decomposition

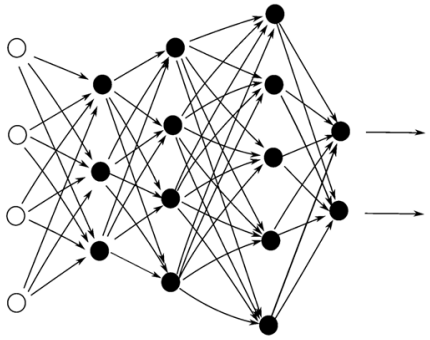
3 Operator Inference

4 Neural Networks

Neural Networks
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- The theoretical inception of Neural Networks has a more than 4 decades of history. However, only the emergence of effectively programable massively parallel compute units, i.e., CUDA in 2004, and the availability of abundant data, e.g., Image Net in 2009, made Neural Networks popular / scalable.
- Neural Networks were not even listed as one of the top 10 algorithms in data mining in 2008[1].
- Deep Learning is a loosely defined term and refers to Neural Networks with “many” layers.
- A Neural Network is a specific parameterized function $F(x, p)$, were $p \in \mathbb{R}$ indicates the parameters.
- The parameters p are then chosen such that the function regresses or classifies data optimally

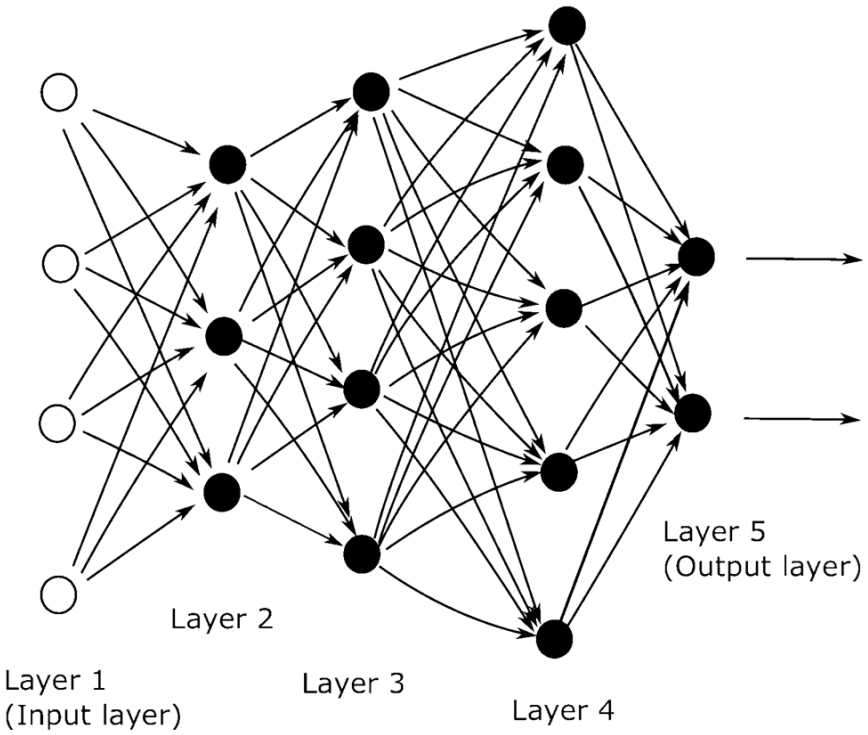


Sources: 1) X Wu, V Kumar, JR Quinlan, et al (2008): Top 10 algorithms in data mining. Knowledge and Information Systems
2) J Deng, W Dong, R Socher, LJ Li, K Li, L Fei-Fei (2009): ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR

Neural Networks
A simple example (i)

TECHNISCHE UNIVERSITÄT DARMSTADT

- A Neural Network has at least two but usually multiple layers L. Each layer l, for $l = 1,2, \dots, L$, contains n_l neurons.
- The first layer $l = 1$ is called Input Layer and n_1 is the dimension of the input data. The layer $l = L$ is called the Output Layer and n_L is the dimension of the output data. All other layers are referred to as Hidden Layers. Overall, the network maps from $\mathbb{R}^{n_1}$ to $\mathbb{R}^{n_L}$.
- In each layer l, every neuron j outputs a single real number, which in a densely connected Neural Network is passed to every neuron in the next layer.



- This is a very small Neural Network, e.g., the Alex Net architecture which achieved groundbreaking image classification results in 2021 used 650,000 neurons and today transformer networks have billions of neurons.

Source: CF Higham, DJ Higham (2019): Deep Learning: An Introduction for Applied Mathematicians, SIAM Review

Neural Networks
A simple example (ii)

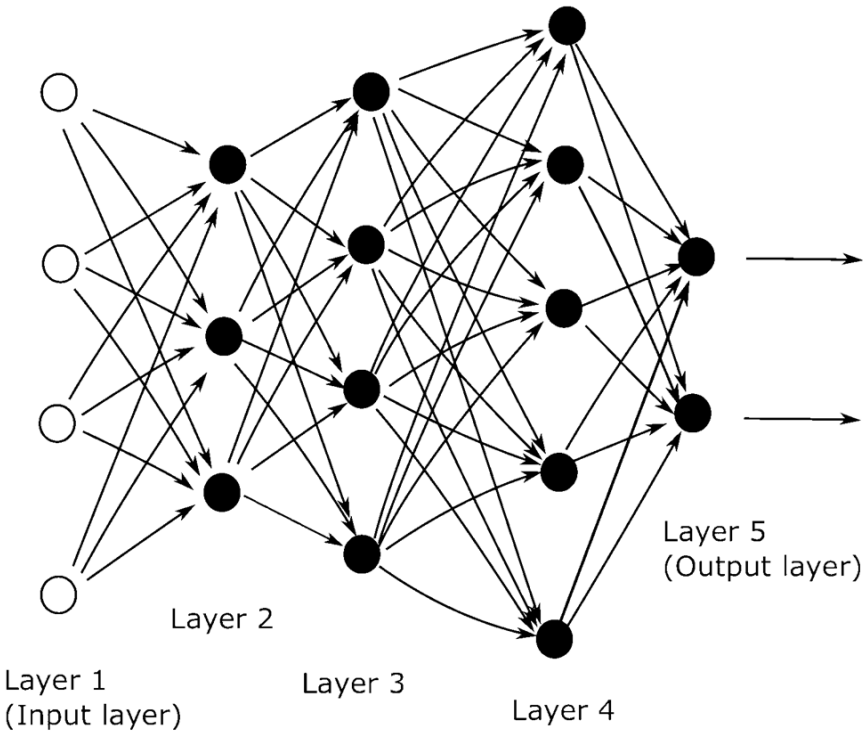
- Each neuron k in each layer l forms it‘s own weighted linear combination of the inputs (Weighted Input) coming from the previous layer $l-1$, i.e.,

$$z_j^{[l]} = \sum_k w_{jk} a_k + b_j$$

where $w_{jk}^{[l]}$ is the weight that neuron j at layer l applied to the output $a_k^{[l-1]}$ from neuron k at layer $l-1$ and $b_j^{[l]}$ is an added bias.

- $z_j^{[l]}$ is then passed through the Activation Function σ providing for each neuron j of the layer l the corresponding output or activation

$$a_j^{[l]} = \sigma(z_j^{[l]})$$



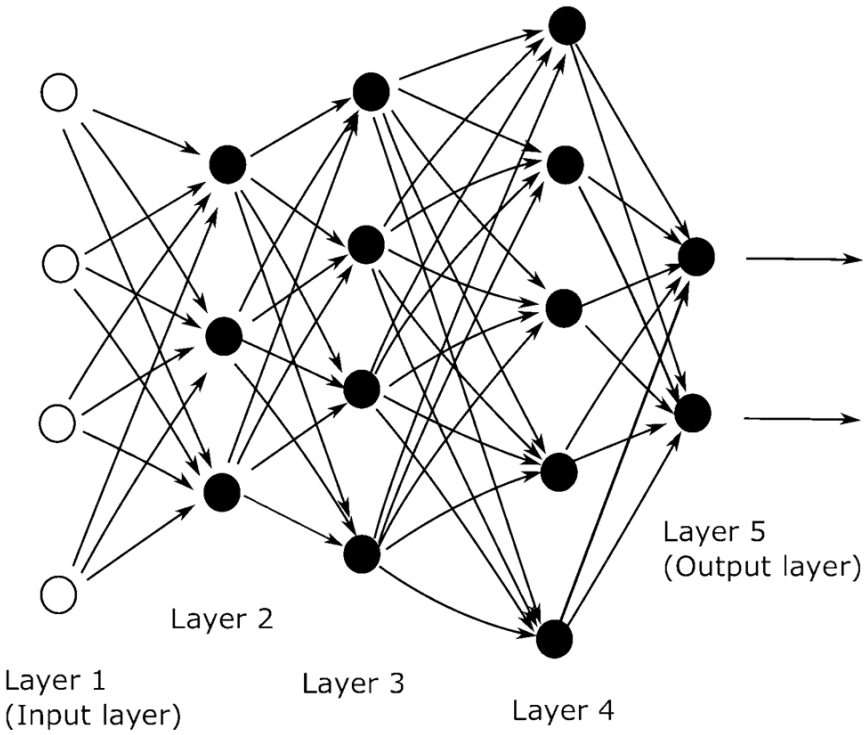
Source: CF Higham, DJ Higham (2019): Deep Learning: An Introduction for Applied Mathematicians, SIAM Review

Neural Networks
A simple example (iv)

TECHNISCHE UNIVERSITÄT DARMSTADT

Example:
Neural Network with with 5 layers:
$n_1 = 4$, $n_2 = 3$, $n_3 = 4$, $n_4 = 5$, and $n_5 = 2$

Thus,
- $W^{[2]} \in \mathbb{R}^{3\times4}$, $W^{[3]} \in \mathbb{R}^{4\times3}$, $W^{[4]} \in \mathbb{R}^{5\times4}$, $W^{[5]} \in \mathbb{R}^{2\times5}$
- $b^{[2]} \in \mathbb{R}^3$, $b^{[3]} \in \mathbb{R}^4$, $b^{[4]} \in \mathbb{R}^5$, $b^{[5]} \in \mathbb{R}^2$



For $x \in \mathbb{R}^4$

$$F(x) = \sigma\left(W^{[5]}\sigma\left(W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}x + b^{[2]}\right) + b^{[3]}\right) + b^{[4]}\right) + b^{[5]}\right) \in \mathbb{R}^2$$

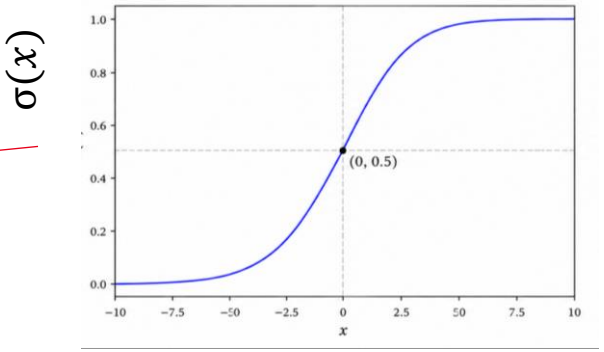
is a function of $3 \cdot (4+1) + 4 \cdot (3+1) + 5 \cdot (4+1) + 2 \cdot (5+1) = 74$ parameters

Source: CF Higham, DJ Higham (2019): Deep Learning: An Introduction for Applied Mathematicians, SIAM Review

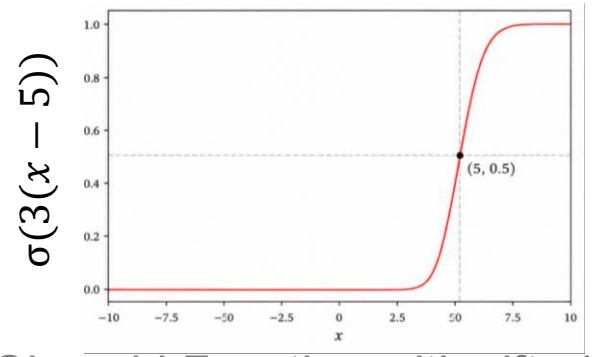
TECHNISCHE UNIVERSITÄT DARMSTADT

Typical activation functions $\sigma(x)$ used in Neural Networks

- **Linear:** $\sigma(x) = x$
- **Sigmoid Function:** $\sigma(x) = \frac{1}{1+\exp -x}$
- **Tanh:** $\sigma(x) = \tanh(x)$
- **Rectified Linear Unit (ReLU):** $\sigma(x) = \begin{cases} 0 \text{ for } x \leq 0 \\ x \text{ for } x > 0 \end{cases}$
- **Leaky ReLU:** $\sigma(x) = \begin{cases} 0.001x \text{ for } x \leq 0 \\ x \text{ for } x > 0 \end{cases}$



Sigmoid Function



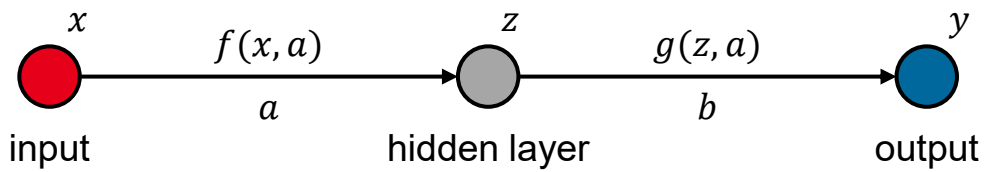
Sigmoid Function with sifted and scaled input

TECHNISCHE UNIVERSITÄT DARMSTADT

Task [Backpropagation]:

Calculate the partial derivatives of the cost function with respect to each $w_{jk}^{[l]}$ and $b_j^{[l]}$. For this derivation, it is sufficient to consider only one training point, i.e., we consider the cost function $C = \frac{1}{2}\left\|y - a^{[L]}\right\|_2^{.2}$

Example [Most Simple Neural Network]:



$$y = g(z, b) = g(f(x, a), b)$$

With the cost function $C = \frac{1}{2}(y_0 - y)^2$. It thus holds

$$\frac{\partial C}{\partial b} = -(y_0 - y)\frac{\partial y}{\partial b}$$

$$\frac{\partial C}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial a}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

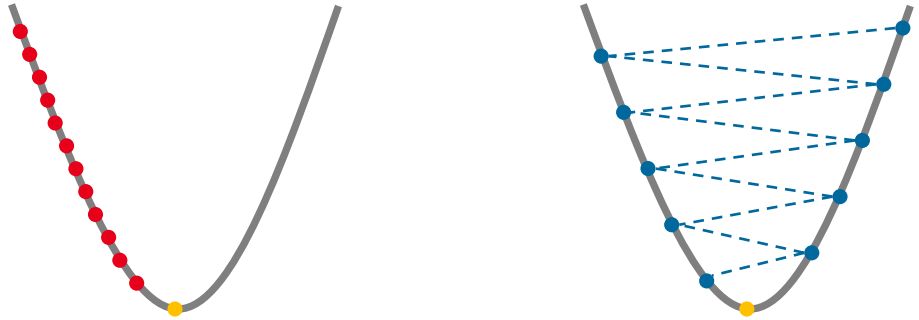
Algorithm [Stochastic Gradient "Descent"]:

1. For counter=1 up to Niter
2. Choose an integer k uniformly at random from $\{1,2,3,\dots,N\}$
3. $a^{[1]} = x^{\{k\}}$ with current training point $x^{\{k\}}$
4. for $l = 2$ up to L
5. $z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$
6. $a^{[l]} = \sigma(z^{[l]})$
7. $\Sigma^{[l]} = \mathrm{diag}(\sigma'(z^{[l]}))$
8. end
9. $\delta^{[L]} = \Sigma^{[L]}(a^{[L]} - y(x^{\{k\}}))$
10. for $l = L - 1$ downto 2
11. $\delta^{[l]} = \Sigma^{[l]}\left(W^{[l+1]}\right)^T \delta^{[l+1]}$
12. end
13. for $l = L - 1$ downto 2
14. $W^{[l]} \to W^{[l]} - \eta\delta^{[l]}a^{[l-1]^T}$
15. $\delta^{[l]} \to \delta^{[l]} - \eta\delta^{[l]}$
16. end
17. end

- For simplicity we consider the basic version where single samples are chosen with replacement

TECHNISCHE UNIVERSITÄT DARMSTADT

- Choosing an appropriate learning rate is crucial for an effective optimization / machine learning model training.
- The convergence rate of decent methods and consequently the model's performance are strongly sensitive to the learning rate.



- While classical optimization methods typically adapt learning rates by means of line-search methods to achieve optimal convergence, the very large optimization problems involved in training neural networks are prohibitively large for corresponding approaches.
- The most popular and simple setting for the learning rate in SGD consists of taking constant values, i.e., $\eta_k = \eta \; \forall k$.

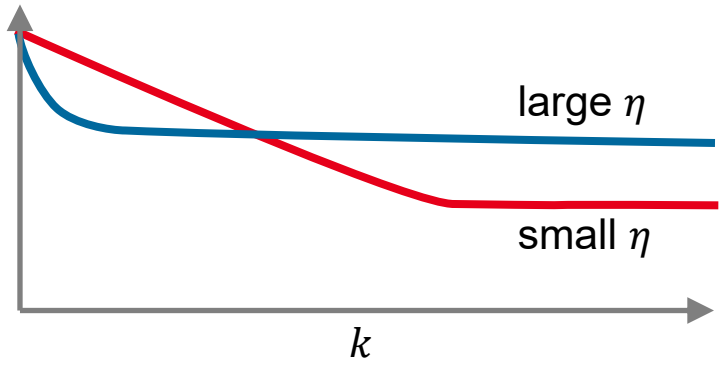
TECHNISCHE UNIVERSITÄT DARMSTADT

- To avoid slow convergence, another strategy is to start with a very large η_0 and reduce it progressively when the loss function features a plateau or reducing it with an inverse decay or an exponential rule

  $\eta_k = \frac{\beta}{\gamma + k}$ (inverse decay)

  $\eta_k = \eta_o \exp(-\gamma k)$ (exponential decay),

  where γ is a positive hyper parameter.



- Instead of choosing the same learning rate for the complete update vector, another alternative are component wise step directions. I.e., take small learning rates for components with high gradients and vice versa – though at the price of introducing additional hyper parameters.

  Examples are the Root Mean Square Propagation method or the Adams method

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