Learning Physical Laws with AI

Energy Days 2025
Prague - 7 Nov 2025
Florian Sobieczky – SCCH
(Software Competence Center Hagenberg)

Overview

1. Different Qualities of of Physics Informedness
2. Zoo of Methods
3. Proposition of a Method for Learning Physical Laws
4. Experiment

Physics and Predictive Modeling

In the **Supervised Learning** Universe, Predictive Models are parametrized functions of type

$$\mathrm{u}\colon \mathrm{X} \times \Theta \rightarrow \mathrm{Y}$$

which, by a training process on a given data-set $\mathrm{D} = \{(\mathrm{x(j)}, \mathrm{y(j)})\}_{\mathrm{j<N}}$, allow estimation of the loss-minimizing parameter $\hat{\theta} \in \Theta$ and associated trained model ('approximator')

$$\hat{\mathrm{u}}\colon X \rightarrow Y, \hat{\mathrm{u}}(x) := \mathrm{u}(x, \hat{\theta}).$$

If this model describes a physical process fulfilling a differential equation (ODE, PDE)

$$\mathrm{L[u]} = \mathrm{f},$$

then it is desirable (Property D) that approaching f by an approximator $\hat{\mathrm{f}}$ implies

$$\mathrm{L}\widehat{[\mathrm{u}]} \rightarrow \mathrm{f}.$$

Physics and Predictive Modeling

In the **Supervised Learning** Universe, Predictive Models are parametrized functions of type

$$\mathrm{u}\colon \mathrm{X} \times \Theta \rightarrow \mathrm{Y}$$

which, by a training process on a given data-set $\mathrm{D} = \{(\mathrm{x(i)}, \mathrm{y(i)})\}_{\mathrm{i<N}}$, allow estimation of the loss-minimizing parameter $\hat{\theta} \in \Theta$ and associated trained model (‘approximator’)

$$\hat{\mathrm{u}}\colon X \rightarrow Y, \hat{u}(x) := \mathrm{u}(x, \hat{\theta}).$$

If this model describes a physical process fulfilling a differential equation (ODE, PDE)

$$\mathrm{L[u]} = \mathrm{f},$$

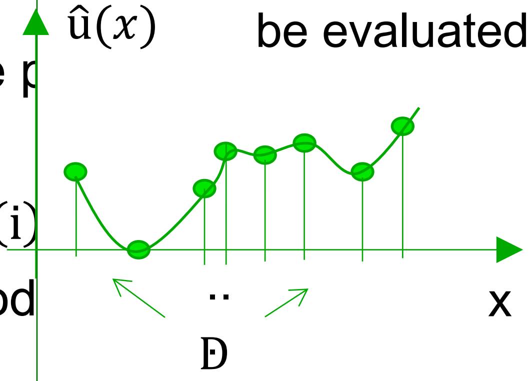
then it is desirable (Property D) that approaching f by an approximator $\hat{\mathrm{f}}$ implies

$$\mathrm{L}[\widehat{\mathrm{u}}] \rightarrow \mathrm{f}.$$

Question: *How do we learn about $L[u] = f$ from a well generalizing model $\hat{u}$ trained on a sufficiently large given data set D?*

Physics and Predictive Modeling

$\hat{\mathrm{u}}$ not only approximates u but also interpolates (can be evaluated) outside of D!



In the **Supervised Learning** Universe, Predictive Models are p

$$\mathrm{u}\colon \mathrm{X} \times \Theta \rightarrow \mathrm{Y}$$

which, by a training process on a given data-set $\mathrm{D} = \{(\mathrm{x(i)}, \mathrm{y(i)}$

loss-minimizing parameter $\hat{\theta} \in \Theta$ and associated trained mod

$$\hat{\mathrm{u}}\colon X \rightarrow Y, \hat{u}(x) := u(x, \hat{\theta}).$$

If this model describes a physical process fulfilling a differential equation (ODE, PDE)

$$\mathrm{L[u]} = \mathrm{f},$$

then it is desirable (Property D) that approaching f by an approximator $\hat{\mathrm{f}}$ implies

$$\mathrm{L}[\widehat{\mathrm{u}}] \rightarrow \mathrm{f}.$$

Question: *How do we learn about $L[u] = f$ from a well generalizing model $\hat{u}$ trained on a sufficiently large given data set D?*

Different Approaches

PINN PIRL

SINDy KAN LNN

HNN DeepXDE ResNET NeuralDE ...

Three different Qualities:

(A) Physical Knowledge is put into predictive model,

(B) Physical Knowledge is derived from predictive model,

(C) Physical Knowledge is explained using an underlying base model.

PINN (Physics Informed Neural Networks)

Idea: Use **Physical Loss** l_{ph} in addition to **Prediction Loss** i l_{pr} in total loss function $(1-\lambda)l_{pr} + \lambda\, l_{ph}$ during training of neural network to condition on ‘physical solutions’.

λ: Regularization Parameter

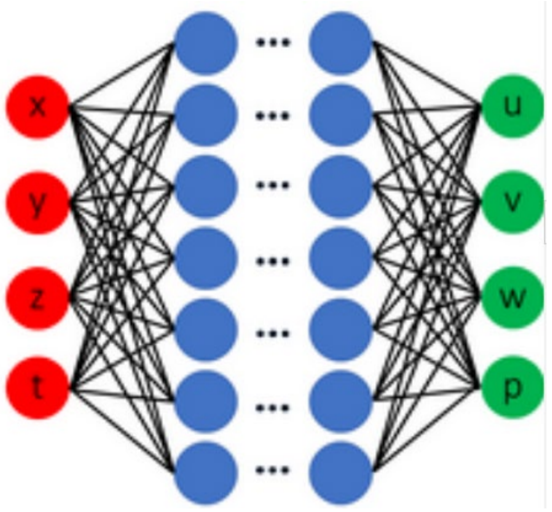
l_{pr}: Evaluated on D: $||\, u - u_{obs}||$

l_{ph}: Evaluated on functions on X

using automatic differentiation

E.G. $l_{ph}(u) \ = ||\, \kappa \cdot u'' - \dot{u}||$

κ is trated as optimization parameter like neural weights.



Benefits:

- Faster Training (A) due to stronger ‘guiding’ constraints
- Makes NNet applicable to small Datasets D as evaluation of l_{ph} possible on all of X.
- During training (=optimization) also infer physical constants (B).

[1] Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021)

PINN (Physics Informed Neural Networks)

Or by a more general approach:

$$u_t = \sum_k \lambda_k \Phi_k(u, u_x, u_{xx}, \ldots)$$

and find λ_k by Symbolic Regression.

Idea: Use **Physical Loss** l_{ph} in addition to **Prediction Loss** i l_{pr} in total loss function $(1-\lambda)l_{pr} + \lambda\, l_{ph}$ during training of neural network to condition on 'physical solutions'.

λ: Regularization Parameter

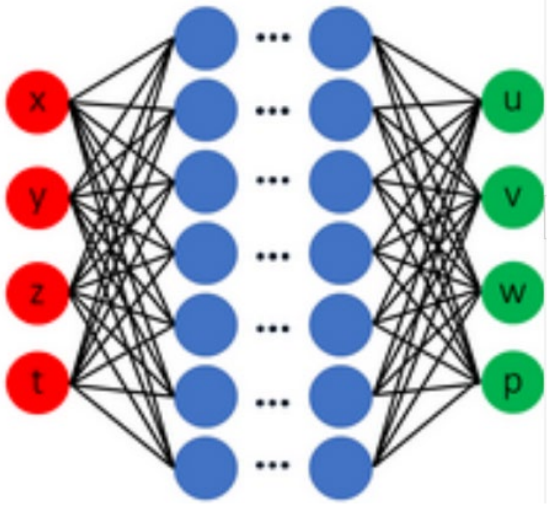
l_{pr}: Evaluated on D: $|| u - u_{obs} ||$

l_{ph}: Evaluated on functions on X

using automatic differentiation

E.G. $l_{ph}(u) = || \kappa \cdot u'' - \dot{u} ||$

κ is trated as optimization parameter like neural weights.



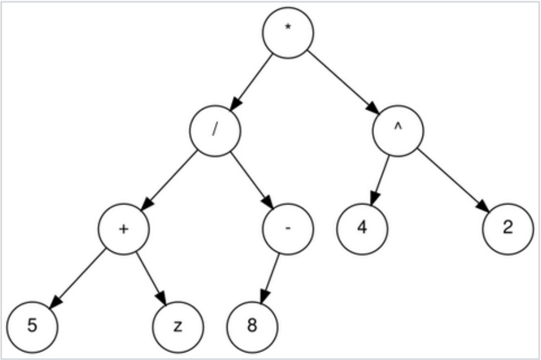
- Faster Training (A) due to stronger 'guiding' constraints
- Makes NNet applicable to small Datasets D as evaluation of l_{ph} possible on all of X.
- During training (=optimization) also infer physical constants (B).

[2] Heteroscedastic uncertainty quantification in Physics-Informed Neural Networks.
Claessen, O., Shapovalova, Y. & Heskes, T. (2024). In *ICLR 2024 Workshop on AI4DifferentialEquations In Science, Vienna, Austria, May 11, 2024* (pp. 1-6). S.l.: OpenReview

Sindy (Sparse Identification of Non-Linear Dynamics)

$$\mathbf{X} = \begin{bmatrix} \mathbf{x^T(t_1)} \\ \mathbf{x^T(t_2)} \\ \vdots \\ \mathbf{x^T(t_m)} \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix}$$

$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi}$$



Binary algebraic expression tree equivalent to ((5 + z) / -8) * (4 ^ 2)

$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & \\ 1 & \mathbf{X} & \mathbf{X^2} & \mathbf{X^3} & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix}$$

$$\xi_{\mathbf{k}} = \arg\min_{\xi'_{\mathbf{k}}} \left|\left|\dot{\mathbf{X}}_{\mathbf{k}} - \mathbf{\Theta}(\mathbf{X})\xi'_{\mathbf{k}}\right|\right|_{\mathbf{2}} + \lambda\left|\left|\xi'_{\mathbf{k}}\right|\right|_{\mathbf{1}}$$

Dynamical System: $\mathbf{x}'(\mathrm{t}) = \mathbf{f}(\mathbf{x}(\mathrm{t}))$

$$\dot{x}_k = \mathbf{\Theta}(\mathbf{x})\xi_{\mathbf{k}}$$

Linear Regression on data points

Potential candidates for solutions to differential equation: ‘Library’, Combinations via binary expressions

L1-Regression: Sparseness

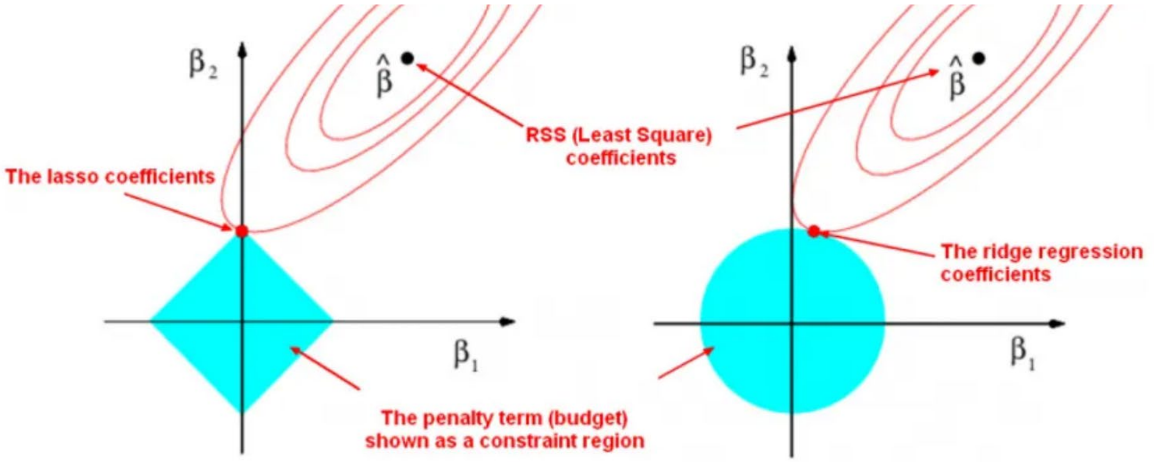
"Discovering governing equations from data by sparse identification of nonlinear dynamical systems"arXiv1509.03580
Brunton, Steven L.; Proctor, Joshua L.; Kutz, J. Nathan (2016-04-12). . Proceedings of the National Academy of Sciences. ***113*** *(15): 3932–3937. :*

Sindy (Sparse Identification of Non-Linear Dynamics)

From: L. Ibadullayeva. Mastering Overfitting: A Deep Dive into Lasso and Ridge Regularization with Python and R, https://medium.com/@lala.ibadullayeva/mastering-overfitting-a-deep-dive-into-lasso-and-ridge-regularization-with-python-and-r-p-d0bd2a7a9328

$$\mathbf{X} = \begin{bmatrix} \mathbf{x^T(t_1)} \\ \mathbf{x^T(t_2)} \\ \vdots \\ \mathbf{x^T(t_m)} \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix}$$

$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & \\ 1 & \mathbf{X} & \mathbf{X}^2 & \mathbf{X}^3 & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix}$$



$$\dot{x}_k = \mathbf{\Theta}(\mathbf{x})\xi_\mathbf{k}$$

Dynamical System: $\mathbf{x}'(\mathrm{t}) = \mathbf{f}(\mathbf{x}(\mathrm{t}))$

Linear Regression on data points

Potential candidates for solutions to differential equation: ‘Library’, Combinations via binary expressions

L1-Regression: Sparseness

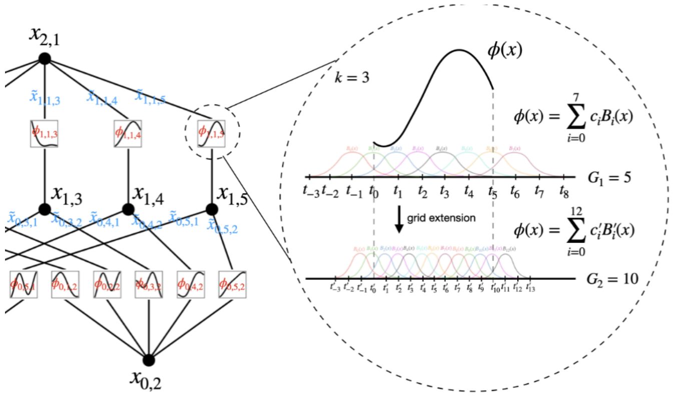
"Discovering governing equations from data by sparse identification of nonlinear dynamical systems"arXiv1509.03580
Brunton, Steven L.; Proctor, Joshua L.; Kutz, J. Nathan (2016-04-12). . Proceedings of the National Academy of Sciences. ***113*** *(15): 3932–3937. :*

KAN (Kolmogorov Arnold Networks)

$$f(\mathbf{x}) = f(x_1, \cdots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

V. Arnold , A. Kolmogorov (1956)

Idea: Neural Network acts on input layer also by summing univariate functions -> they need not be relu's.



Every continuous function of n variables on bounded domain is composition of continuous *univariate* functions by addition.

Composition of n functions

- Activation function *not* relu(x), but arbitary.
- Symbolic Regression done by KAM where nodes represent sums of univariate (activation) functions of former layer.

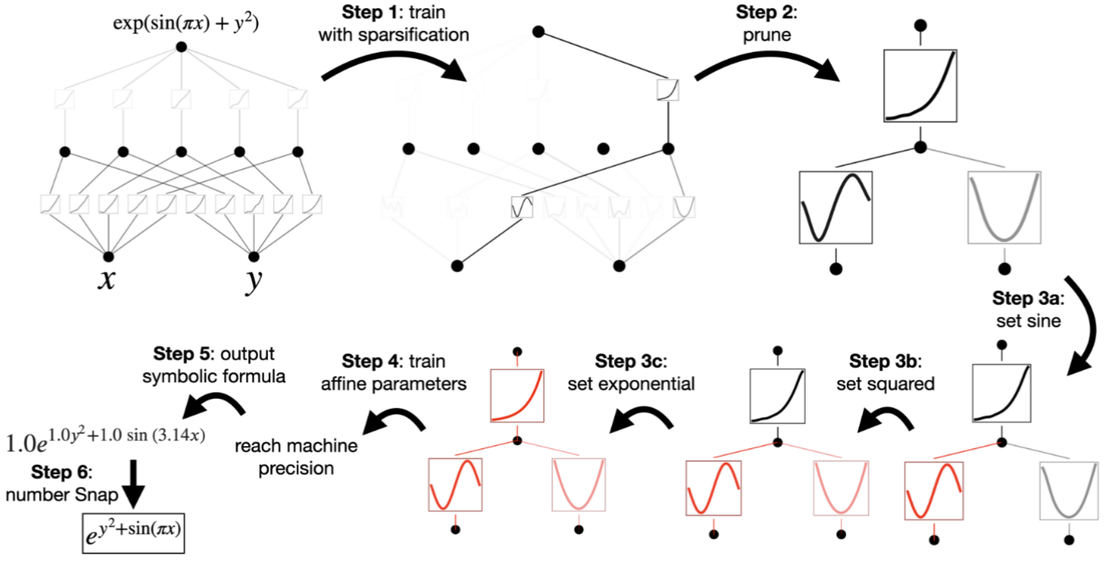
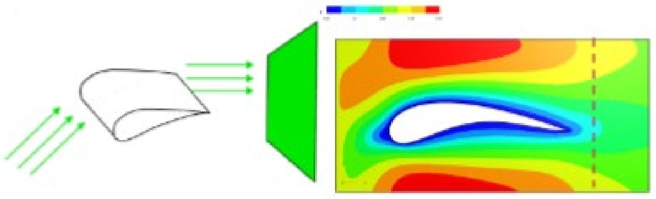


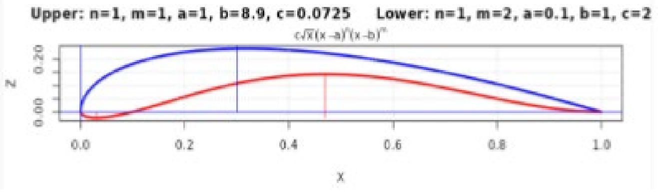
Figure 2.4: An example of how to do symbolic regression with KAN.

[2] Zhang, Z., Wang, Q., Zhang, Y. *et al.* Physics-informed neural networks with hybrid Kolmogorov-Arnold network and augmented Lagrangian function for solving partial differential equations. *Sci Rep* **15**, 10523 (2025). https://doi.org/10.1038/s41598-025-92900-1

PIRL
Physics Informed Reinforcement Learning

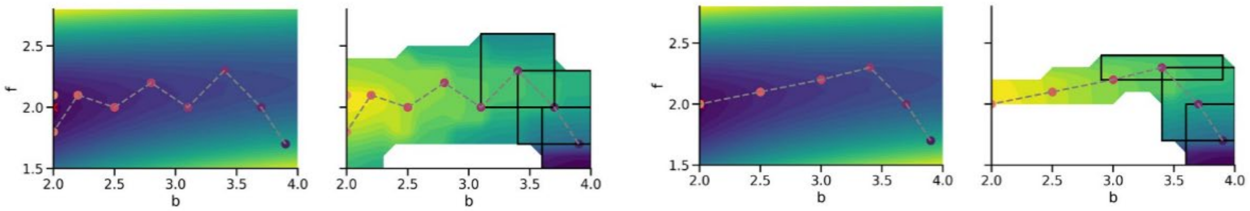
MDP with (X, A, P, R)

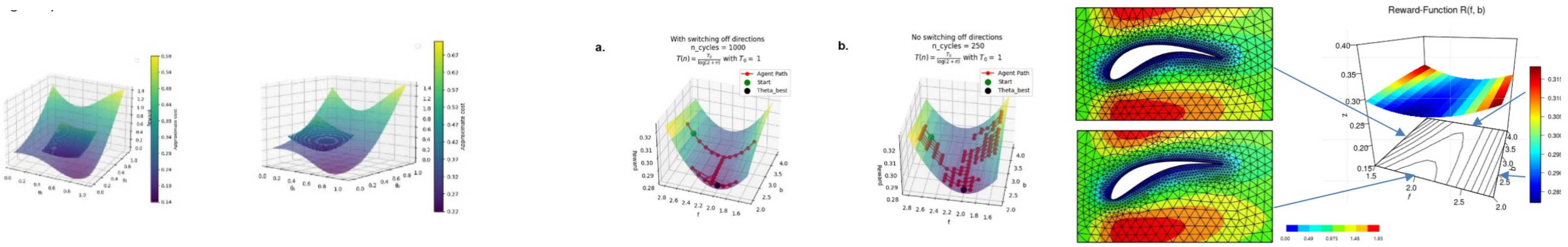




Add Physics Information in

- the reward Function R,
- the transition measure P,
- the value function $\mathrm{V}(\mathrm{x}) = \mathbf{E}[\mathrm{R}(\mathrm{x}(\mathrm{j}))]$.





[3] Florian Sobieczky, Alfredo Lopez, Erika Dudkin, Christopher Lackner, Matthias Hochsteger, Bernhard Scheichl, Helmut Sobieczky, **Reinforcement Learning for Accelerated Aerodynamic Shape Optimisation,**

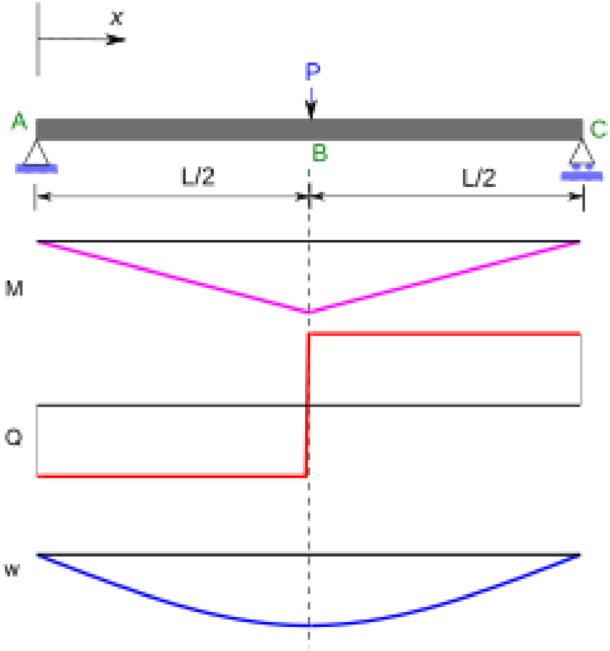
Motivating Example: Beam Bending

$$M = -EI\frac{d^2w}{dx^2}$$

$$(EI(x)\,w''(x))'' = q(x)$$

E: Young'modulus
I: Area moment of inertia
EI: 'Stiffness'

$$Q = -\frac{d}{dx}\left(EI\frac{d^2w}{dx^2}\right)$$



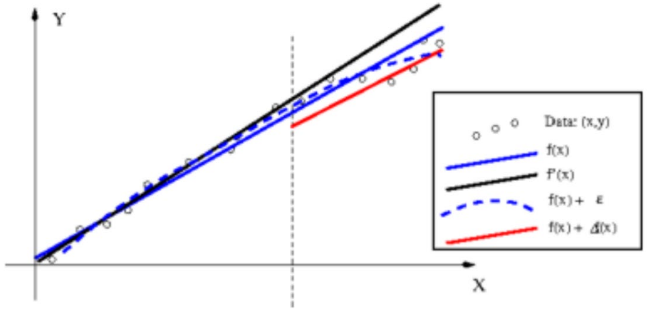
- Limiting Case: Bernoulli-Assumptions (No shearing/torsion):
- Reality slightly different: Moments of Deviation not zero
- Define surrogate (base-) model w_0 & consider residual error to w_0 -> Train model $\hat{\epsilon}$ predicting these errors
- Define hybrid model $\hat{w}(x,i) = w_0(x) + \hat{\epsilon}^{(i)}(x)$

Deflection curve

('Biegelinie')

$$w(x) = -\frac{\int_0^x \left(\int_0^x M_y(\xi)\,\mathrm{d}\xi + C_1\right)\mathrm{d}\xi + C_2}{EI_y}$$

X-ODE: Calculate Source Term of Linear (inhomogeneous) ODEs



- BAPC: **B**efore and **A**fter prediction **P**arameter **C**omparison [1] - **If**

i.) $\mathrm{Y_i} = f_\theta(X_i) + \epsilon_i$, ii.) $\epsilon_\mathrm{i} = \hat{\epsilon}(X_i) + \Delta\epsilon_i$, iii.) $\mathrm{Y_i} - \hat{\epsilon}(X_i) = f_{\theta'}(X_i) + \epsilon_i{}'$

(f_θ is the 'base model', $\hat{\epsilon}$ models anomaly, $Y_i - \hat{\epsilon}(X_i)$ is data-correction),

then $f_\theta(x_i) - f_{\theta'}(x_i)$ is an 'explanation' (interpretation/sketch) of $\hat{\epsilon}$ at $\mathrm{x_i}$.

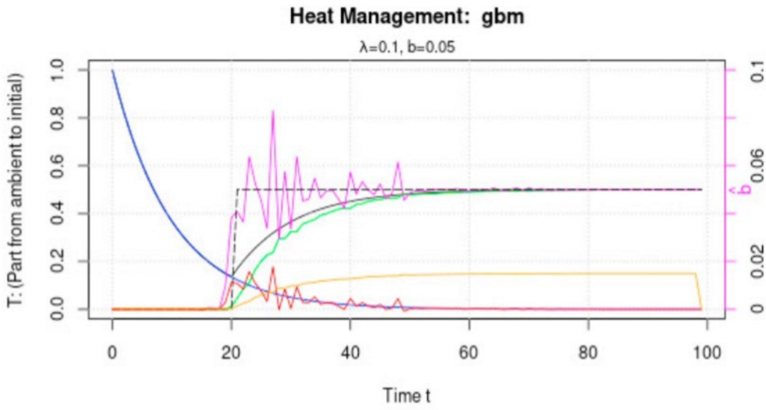
Discovering effect of air-friction:
Acceleration is not a constant!

https://doi.org/10.48550/arXiv.2103.07155

Now, consider the linear ODE $u' = a(x)u(x) + b(x)$.

- Considering the solution to the **homogeneous** ODE the base-model,

from training the residual error there is a formular [2] for the source term, given the predictive model ($\hat{\epsilon}$): $\hat{\mathrm{b}}(x) = \hat{\epsilon}'(x) - a(x)\hat{\epsilon}(x)$.



[1] A. Lopez, F. Sobieczky, Surrogate Modeling for Explainable Predictive Time Series Corrections. https://doi.org/10.48550/arXiv.2412.1989.
[2] Florian Sobieczky, Erika Dudkin, Jan Zenisek, Learning the inhomogenous term of a linear ODE, Procedia Computer Science, Volume 232, 2024, Pages 1548-1553, SSN 1877-0509, https://doi.org/10.1016/j.procs.2024.01.152.

Given the trained corrector $\hat{\epsilon}$, what is the estimator of the source term?

$$y_t' = a_t \cdot y_t + b_t$$

$$y_t = e^{A_t}\left(Y_0 + \int_0^t e^{-A_s} b_s ds\right)$$

$$A_t = \int_0^t a_s ds$$

$$y_t^{(h)} = Y_0\, e_t^A.$$

$$\widehat{\varepsilon}_t = e^{A_t} \int_0^t e^{-A_s} \widehat{b}_s ds \quad => \quad \widehat{b}_t = \widehat{\varepsilon}_t' - a_t \cdot \widehat{\varepsilon}_t$$

ELSEVIER

CrossMark

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 232 (2024) 1548–1553

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Industry 4.0 and Smart Manufacturing

Learning the inhomogenous term of a linear ODE

Florian Sobieczky[a,*], Erika Dudkin[a], Jan Zesinek[b]

[a] Software Competence Center Hagenberg (SCCH), Softwarepark 32a, 4232 Hagenberg i. Mkr., Austria

[b] University of Applied Sciences Upper Austria

Learning physically explainable corrections

1. Observe a Phenomenon, explain it with a linear DE:
   - $\mathrm{L}[\mathrm{u}_0] = 0$ (u_0 is the ‘base’-model, or surrogat model)
2. Reality is different
   - In truth, $\mathrm{L}[\mathrm{u}] = \mathrm{f}$ holds, where $\mathrm{u} = \mathrm{u}_0 + \epsilon$.
3. Train a model $\hat{\epsilon}$ for the residual error $\epsilon_i(x)$ under conditions i

$$\epsilon_\mathrm{i}(x) = \hat{\epsilon}(x, i) + \Delta\epsilon_i(x)$$

   so that you can approximate the real solution u by $\hat{\mathrm{u}}(x, i) = u_0(x) + \hat{\epsilon}(x, i)$ with an error of $\Delta\epsilon_\mathrm{i}(x)$.

4. Define: $\hat{\mathrm{f}} \coloneqq L[\hat{u}]$. Then it follows from the linearity of $\mathrm{L}[\cdot]$

$$L[\hat{\epsilon}] = L[u - u_0] = \hat{\mathrm{f}} - L[u_0],$$ and so: $\hat{\mathrm{f}} = L[\hat{\epsilon}]$.

Learning physically explainable corrections

1. Observe a Phenomenon, explain it with a linear DE:
   - $\mathrm{L}[\mathrm{u}_0] = 0$ (u_0 is the ‘base’-model, or surrogat model)
2. Reality is different
   - In truth, $\mathrm{L}[\mathrm{u}] = \mathrm{f}$ holds, where $\mathrm{u} = \mathrm{u}_0 + \epsilon$.
3. Train a model $\hat{\epsilon}$ for the residual error $\epsilon_i(x)$ under conditions i

$$\epsilon_{\mathrm{i}}(x) = \hat{\epsilon}(x, i) + \Delta\epsilon_i(x)$$

   so that you can approximate the real solution u by $\hat{\mathrm{u}}(x,i) = u_0(x) + \hat{\epsilon}(x,i)$ with an error of $\Delta\epsilon_{\mathrm{i}}(x)$.

4. Define: $\hat{\mathrm{f}} \coloneqq L[\hat{u}]$. Then it follows from the linearity of $\mathrm{L}[\cdot]$ (using just ‘x‘ instead of ‘(x, i)‘ ...)

$$L[\hat{\epsilon}] = L[u - u_0] = \hat{\mathrm{f}} - L[u_0], \quad \text{and so:} \quad \hat{\mathrm{f}} = L[\hat{\epsilon}].$$

5. Does it also fulfill Property (D)?

Learning physically explainable corrections

1. Observe a Phenomenon, explain it with a linear DE:
   - $\mathrm{L}[\mathrm{u}_0] = 0$ (u_0 is the ‘base’-model, or surrogat model)
2. Reality is different
   - In truth, $\mathrm{L}[\mathrm{u}] = \mathrm{f}$ holds, where $\mathrm{u} = \mathrm{u}_0 + \epsilon$.
3. Train a model $\hat{\epsilon}$ for the residual error $\epsilon_i(x)$ under conditions i

$$\epsilon_{\mathrm{i}}(x) = \hat{\epsilon}(x, i) + \Delta\epsilon_i(x)$$

   so that you can approximate the real solution u by $\hat{\mathrm{u}}(x, i) = u_0(x) + \hat{\epsilon}(x, i)$ with an error of $\Delta\epsilon_{\mathrm{i}}(x)$

4. Define: $\hat{\mathrm{f}} \coloneqq L[\hat{u}]$. Then it follows from the linearity of $\mathrm{L}[\cdot]$ (using just ‘x‘ instead of ‘(x, i)‘ ...)

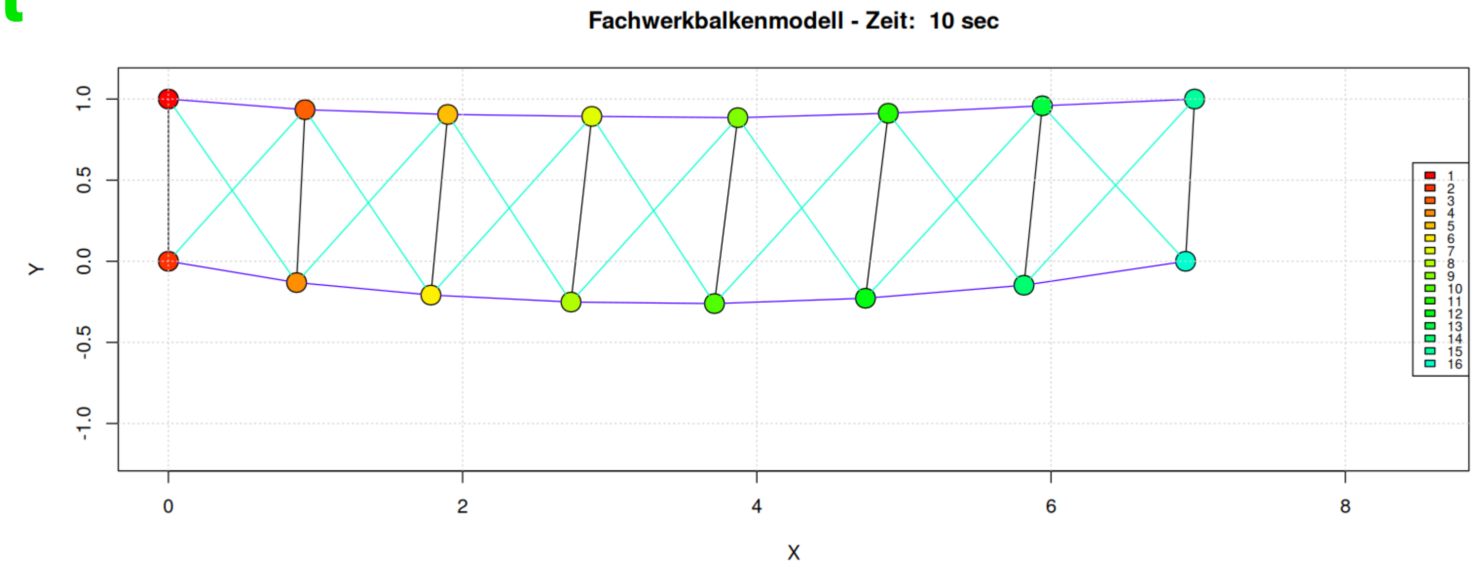
$$L[\hat{\epsilon}] = L[u - u_0] = \hat{\mathrm{f}} - L[u_0], \text{ and so: } \hat{\mathrm{f}} = L[\hat{\epsilon}].$$

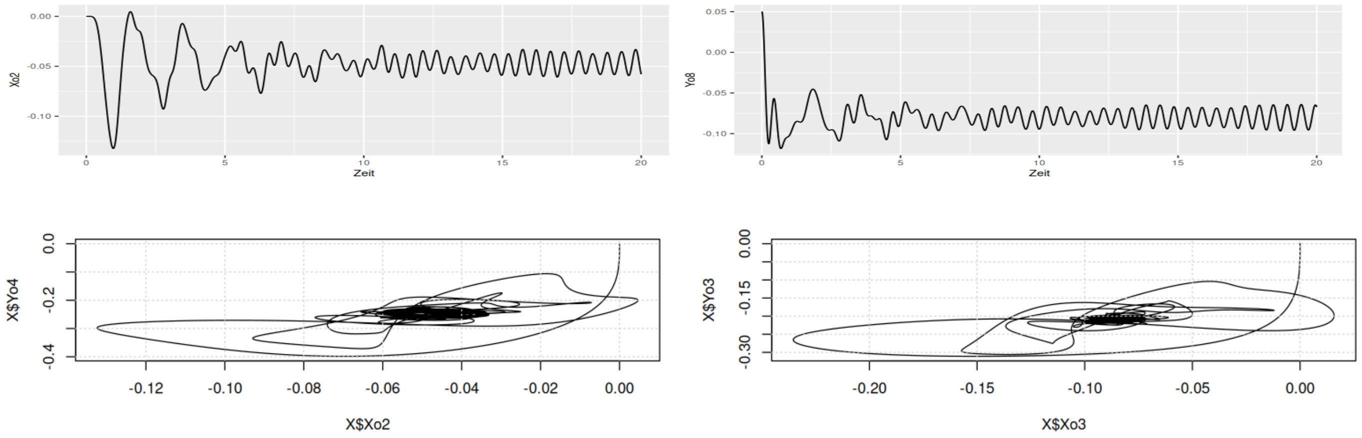
5. Does it also fulfill Property (D)?

   Answer: Yes, since $\mathrm{L}[\cdot]$ is *linear*, $\Delta\epsilon \rightarrow 0 \Rightarrow \hat{\epsilon} \rightarrow \epsilon \Rightarrow$

$$\hat{\mathrm{u}} \rightarrow \mathrm{u} \Rightarrow \hat{\mathrm{f}} = L[\hat{\mathrm{u}}] \rightarrow L[u] = \mathrm{f}$$

Numerical Experiment





- 16 Points of Unit Mass (‘Fachwerk’)
- Linear Forces, springs’ constants
- Initial arrangement: Lattice
- ‘Static determ., ‘Auflager’ $Y_{u8} = \text{const.}$
- Influence of the Bending Moment
- Asymptotic Form = Deflection-line $\mathrm{u}(\mathrm{x})$
- Compare with $\mathrm{w}_0(\mathrm{x})$ (base-model)
- $\mathrm{w}_0(x) = \frac{\mathrm{q}}{24\mathrm{EI}}(x^4 + C_3x^3 + C_2x^2 + C_1x)$

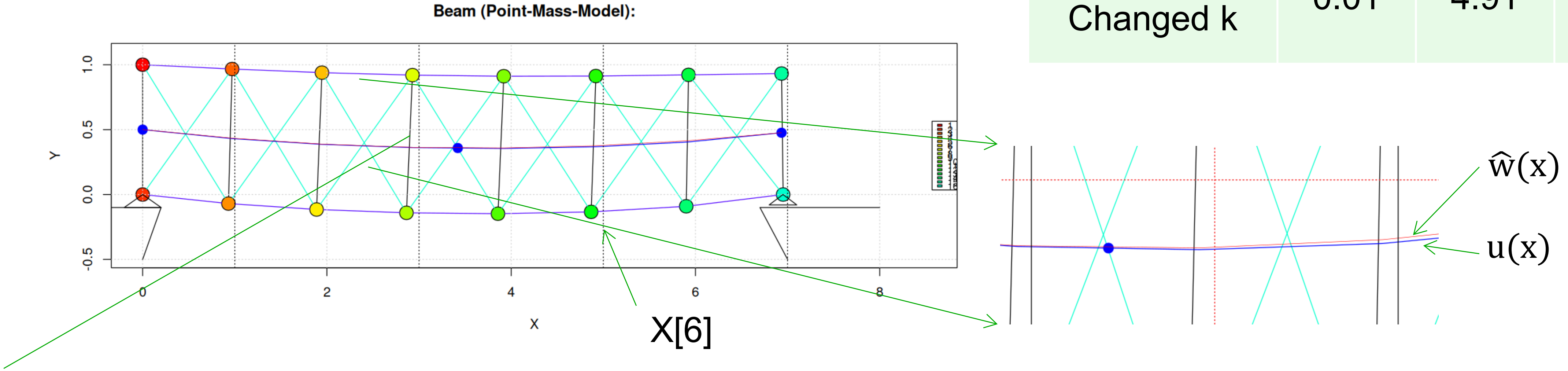
Evaluation of Experiment (Slide 7)

- Training a predictive model $\hat{\epsilon}$ as the corrector for the Bernoulli base-model given by the homogeneous solution of

$$\mathrm{EI}w''(x) = M(x)$$

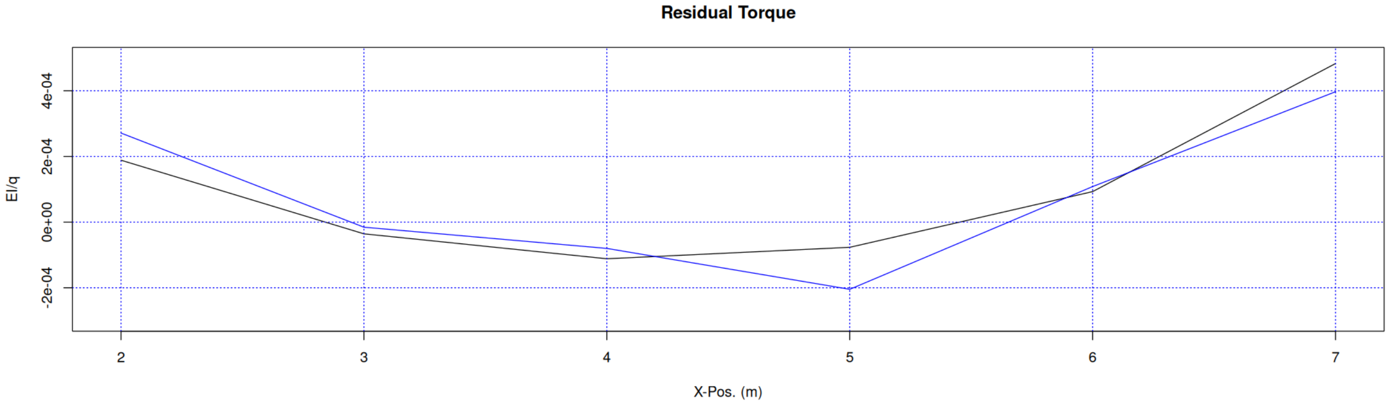
if $M(\mathrm{x}) = \mathrm{mgx} + \Delta M(\mathrm{x})$, where mgx is the bending moment of gravity acting on each mass-point, equally, what is the correcting torque $\Delta \mathrm{M}(\mathrm{x})$ in the Bernoulli-picture to produce the observed correction $\hat{\epsilon}(x)$ in $(x) = \mathrm{w}_0(x) + \hat{\epsilon}(x)\hat{\mathrm{w}}$.

Mode	$\hat{\epsilon}(x_5)$ mm	$\hat{\epsilon}(x_6)$ mm	$\epsilon(\mathbf{x}_7)$ mm
A: Unchanged k	3.78	7.96	9.72
B: Changed k	0.01	4.91	7.22



This Spring-Constant has been changed in Mode B to a value only one tenth of the original (Mode A) value (equal to all other spring constants).

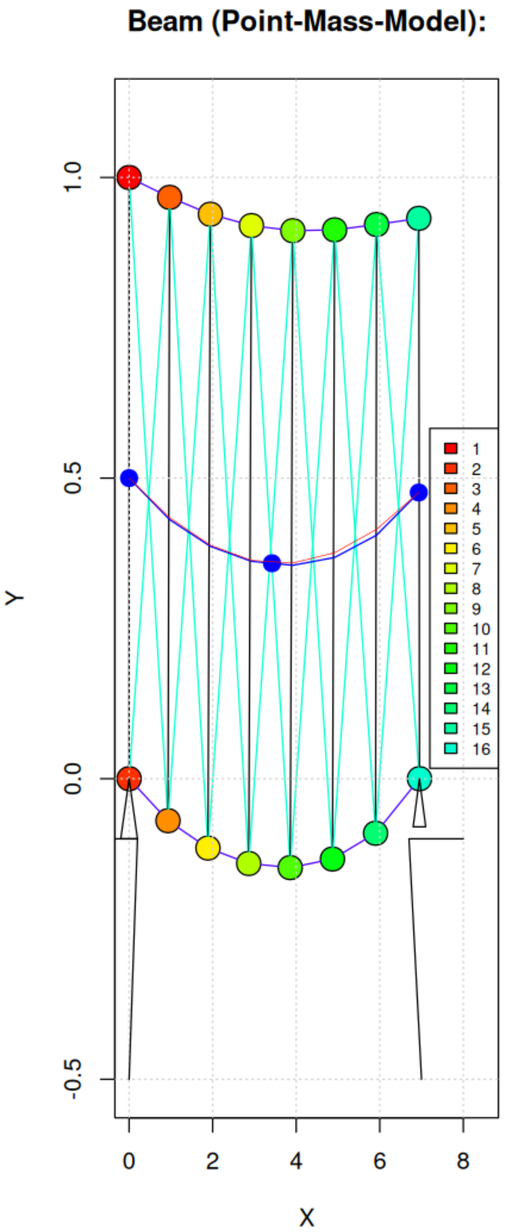
Perturbation of a Spring Constant



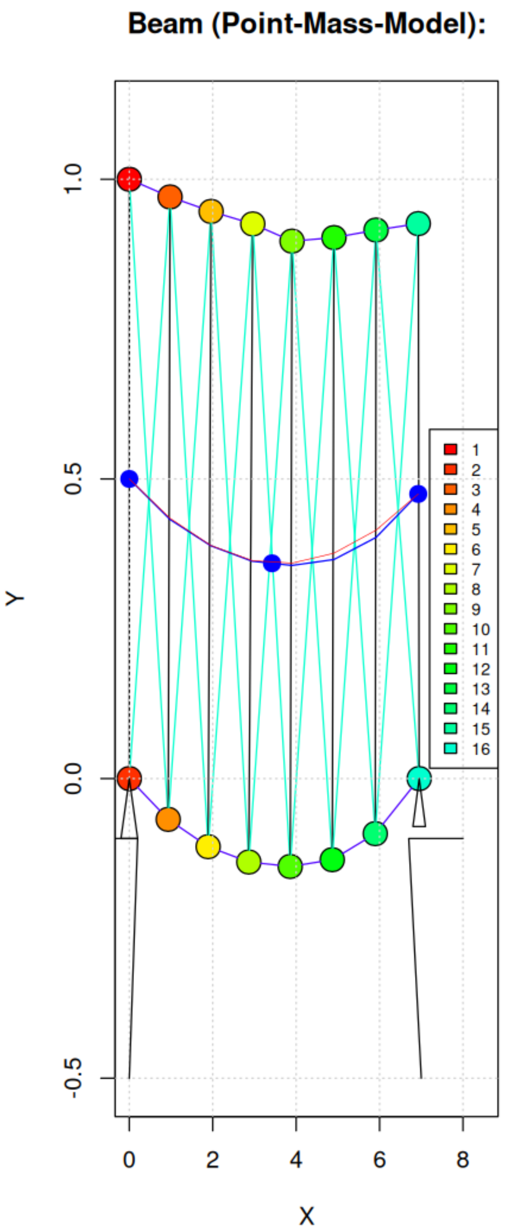
$$\widehat{w}(x) = w_0(x) + \hat{\epsilon}(x) \qquad \widehat{M} = M_0(x) + \widehat{\Delta M}(x)$$

- Right hand side mounting ‘moves’, left hand side mounting fixed.
- Mode 1: Same Spring Constant for all Springs.
- Mode 2: 4th vertical Spring Constant reduced (factor 0.1).
- “Bend” in upper line of masses => Larger correction $\Delta M(x)$

$$\widehat{M}(x) = M_0(x) + \widehat{\Delta M}(x) \text{ for } x \sim 5 \text{ and } 7.$$



Mode 1



Mode 2

Conclusion

Physics Informed Machine Learning is related to (physical) domain knowledge (conveyed in the form of differential equations, parameters or boundary conditions).

- (A) In the forward problem, can we obtain solutions to differential equations faster and more accurately?
- (B) In the inverse problem, can we determine parameters, and additional terms and boundary in the training process of the model?
- If using a hybrid (base-model + AI) approach, can we discover terms of the D.E. which were not previously suggested in a library of candidates?

Conclusion

Physics Informed Machine Learning is related to (physical) domain knowledge (conveyed in the form of differential equations, parameters or boundary conditions).

- (A) In the forward problem, can we obtain solutions to differential equations faster and more accurately? ✓

- (B) In the inverse problem, can we determine parameters, and additional terms and boundary in the training process of the model? ✓

- If using a hybrid (base-model + AI) approach, can we discover terms of the D.E. which were not previously suggested in a library of candidates? ✓

... if the D.E. is linear and the term is the source term (R.H.S.).

Thank you for your attention!

PINN:

Karniadakis, G. E. et al. Physics-informed machine learning. Nat. Rev. Phys. 3, 422–440 (2021)

SINDY:

Brunton, Steven L.; Proctor, Joshua L.; Kutz, J. Nathan (2016-04-12). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences. 113 (15): 3932–393

KAN:

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, Max Tegmark KAN: Kolmogorov-Arnold Networks https://arxiv.org/abs/2404.19756

BAPC:

E. Dudkin, F. Sobieczky. Local Surrogate Modeling: BAPC by Error Emplification. https://link.springer.com/article/10.1007/s00521-025-11535-5

A. Lopez, F. Sobieczky, Surrogate Modeling for Explainable Predictive Time Series Corrections. https://doi.org/10.48550/arXiv.2412.1989,

F. Sobieczky, M. Geiß. Explainable AI by BAPC,Before and After correction Parameter Comparison, https://doi.org/10.48550/arXiv.2103.07155

X-ODE:

F. Sobieczky, E. Dudkin, J. Zenisek, Learning the inhomogenous term of a linear ODE, Procedia Computer Science, Volume 232, 2024, Pages 1548-1553, SSN 1877-0509, https://doi.org/10.1016/j.procs.2024.01.152 ,

Supplement:
What if corrections of other than source term are observed?

- $\mathrm{L}_0[u_0] = 0$ <=> $\sum a_i {u_0}^{(i)} = 0$
- $\mathrm{L[u]} = 0$ <=> $(\mathrm{a}_0 + \mathrm{a})_1 \mathrm{u}^1 + \ldots \left(+(\mathrm{a})_j + \Delta a_j\right) \mathrm{u}^j + \cdots \mathrm{a_n u}^{(n)}$
- Then correction is of form $\mathrm{L}_0[u] = -\Delta \mathrm{a_j u}^{(j)}$ ... Coupling between solution and system.
- Cannot simply define $\hat{\mathrm{f}} \coloneqq \mathrm{L}[\hat{\epsilon}]$ because there will be coupling between $\hat{\epsilon}$ and $\Delta \mathrm{a_j}$.
- However: Approximation possible – Take $\hat{\mathrm{f}} \coloneqq \mathrm{L}_0\ [\hat{\epsilon}\}$. Then: .... $\Delta \mathrm{a_j} = \frac{\mathrm{L}_0[\hat{\epsilon}]}{{u_0}^{(j)}}$. —-> Outlook: New Paper!