MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Physics enhanced machine learning for discovery of phase field models

Sandeep Reddy Bukka, Lekshmi Sreekala, Pawan Goyal, Christoph Freysoldt, Jaber Mianroodi

12 April, 2022

BiGmax Workshop 2022, Bochum

Supported by:

BiGmax MAX PLANCK RESEARCH NETWORK on big-data-driven materials science

Partners:

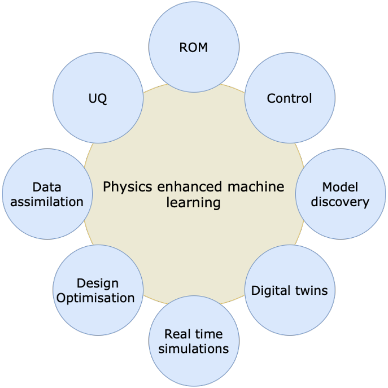
Max-Planck-Institut für Eisenforschung GmbH

Physics enhanced machine learning

Innovative machine learning methods which incorporate fundamental principles and empirical knowledge of the systems.

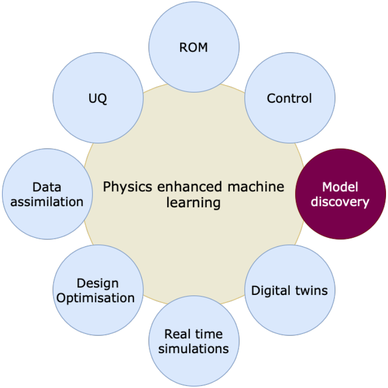
Physics enhanced machine learning

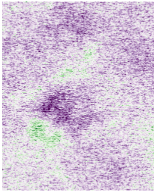
Innovative machine learning methods which incorporate fundamental principles and empirical knowledge of the systems.



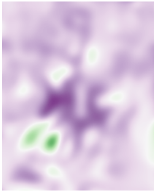
Physics enhanced machine learning

Innovative machine learning methods which incorporate fundamental principles and empirical knowledge of the systems.

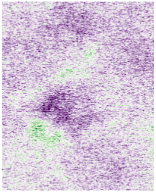




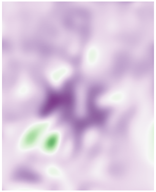
Raw data



Processed data



Raw data



Processed data

Aim: Discovering model

$$\frac{\partial \phi}{\partial t} = \Delta(f(\phi) - \kappa \Delta \phi)$$

Problem statement

1. Model form is "known"
   - Estimate the unknown parameters of the model with physics informed neural nets

Problem statement

1. Model form is “known”
   - Estimate the unknown parameters of the model with physics informed neural nets
2. Model form is “unknown”
   - Create a dictionary of candidate functions
   - Use sparse regression to learn from the dictionary the interpretable model suitable for the experimental data

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

Physics informed neural networks

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

Physics informed neural networks

1. Neural Net (NN) for $\hat{\phi}$

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

Physics informed neural networks

1. Neural Net (NN) for $\hat{\phi}$
2. Estimate Γ, $\hat{\phi}_t$ from the NN

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

Physics informed neural networks

1. Neural Net (NN) for $\hat{\phi}$
2. Estimate Γ, $\hat{\phi}_t$ from the NN
3. Equation loss and data loss

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

Physics informed neural networks

1. Neural Net (NN) for $\hat{\phi}$
2. Estimate Γ, $\hat{\phi}_t$ from the NN
3. Equation loss and data loss
4. Minimize total loss

Model form is known

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown

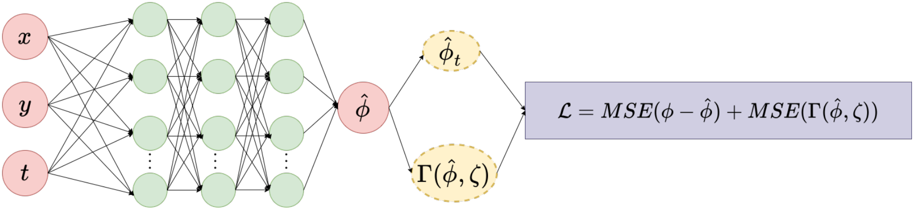
Physics informed neural networks

1. Neural Net (NN) for $\hat{\phi}$
2. Estimate Γ, $\hat{\phi}_t$ from the NN
3. Equation loss and data loss
4. Minimize total loss
5. Weights and ζ

Model form is known

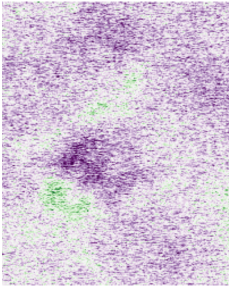
$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

model form of Γ is known, ζ is unknown



[Raissi et al. '19]

Application



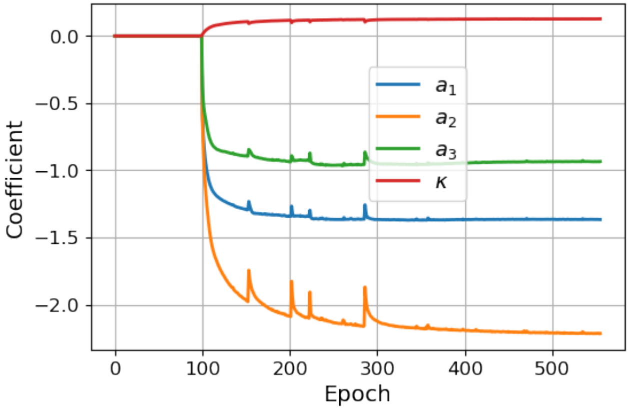
Scanning Transmission Electron Microscopy (STEM) data

Cahn-Hilliard equation

$$\frac{\partial \phi}{\partial t} = \Delta \left(\frac{\partial F(\phi)}{\partial \phi} \right)$$

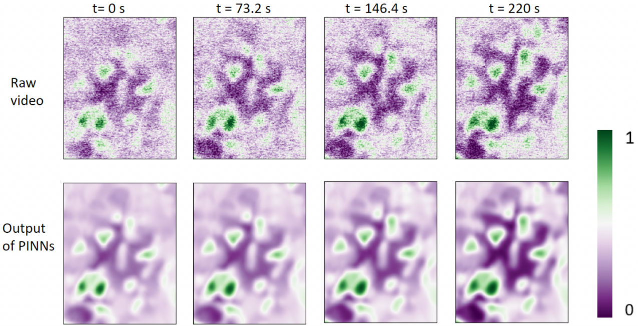
$$\frac{\partial F(\phi)}{\partial \phi} = a_1 \phi(1-\phi) + a_2 \phi(1-\phi)(2\phi - 1) + a_3 \phi(1-\phi)(2\phi-1)^2 + \kappa \Delta \phi$$

Results



0.0
−0.5
−1.0
−1.5
−2.0
Coefficient
0
100
200
300
400
500
Epoch
a_1
a_2
a_3
κ

Results



Model form is unknown

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

Γ is unknown, ζ is unknown

Model form is unknown

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

Γ is unknown, ζ is unknown

Important components

1. Sparse regression

Model form is unknown

$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

Γ is unknown, ζ is unknown

Important components

1. Sparse regression
2. Dictionary

Model form is unknown

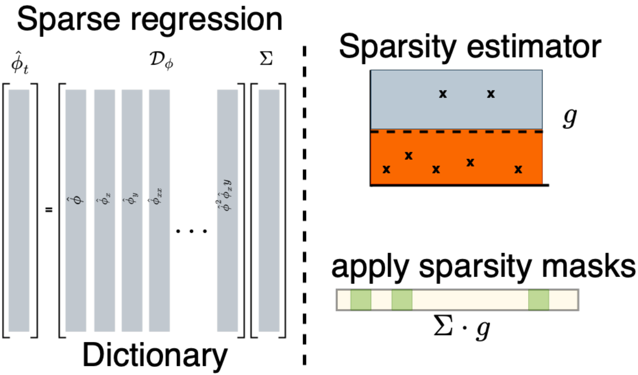
$$\boxed{\hat{\phi}_t = \Gamma(\hat{\phi}, \zeta)}$$

Γ is unknown, ζ is unknown

Important components

1. Sparse regression
2. Dictionary
3. Estimator

Model form is unknown



Model form is unknown

Challenges!!

Model form is unknown

Challenges!!

1. Noisy data

Model form is unknown

Challenges!!

1. Noisy data
2. Features are unknown

Model form is unknown

Challenges!!

1. Noisy data
2. Features are unknown
3. Computation of derivatives

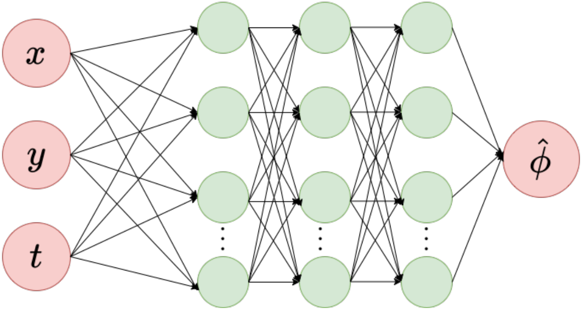
Model form is unknown

Challenges!!

1. Noisy data
2. Features are unknown
3. Computation of derivatives
4. Numerical approximations are poor

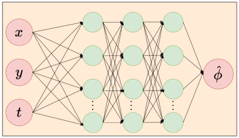
Model form is unknown

Function approximator



Model form is unknown

Function approximator



Training loop

- Compute $\hat{\phi}$

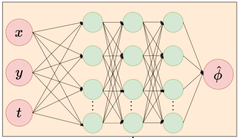
Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]

Model form is unknown

Function approximator



$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\phi_i - \hat{\phi}_i)^2$$

Data loss

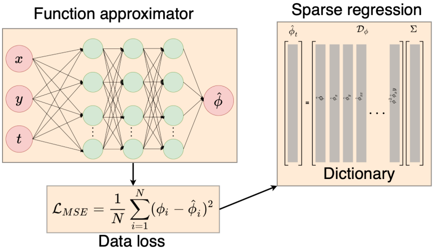
Training loop

- Compute $\hat{\phi}$
- Compute MSE loss

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]



Training loop

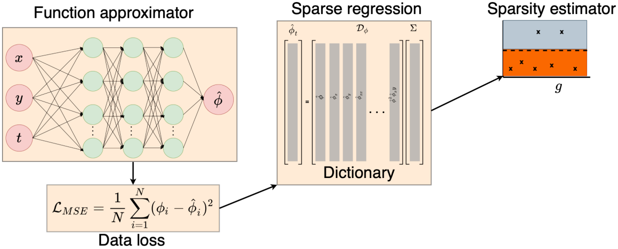
- Compute $\hat{\phi}$
- Compute MSE loss
- Do sparse regression

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]

Model form is unknown



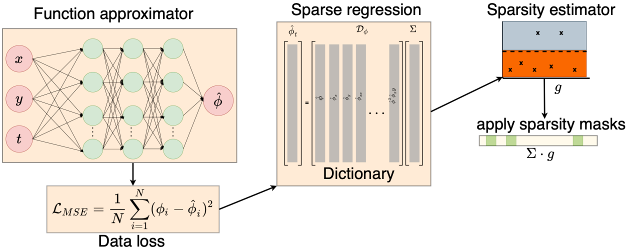
Training loop

- Compute $\hat{\phi}$
- Compute MSE loss
- Do sparse regression
- Apply estimator to compute masks

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]



Training loop

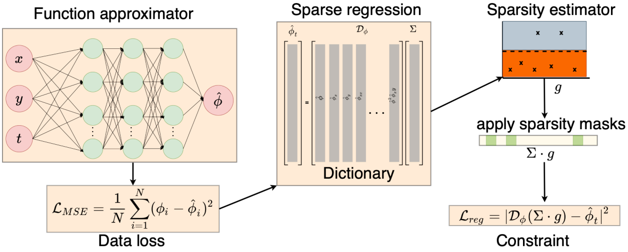
- Compute $\hat{\phi}$
- Compute MSE loss
- Do sparse regression
- Apply estimator to compute masks
- Apply masks

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]

Model form is unknown



Training loop

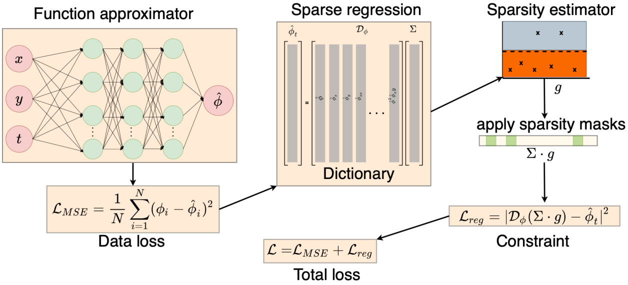
- Compute $\hat{\phi}$
- Compute MSE loss
- Do sparse regression
- Apply estimator to compute masks
- Apply masks
- Compute the constraint via regression

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]

Model form is unknown



Training loop

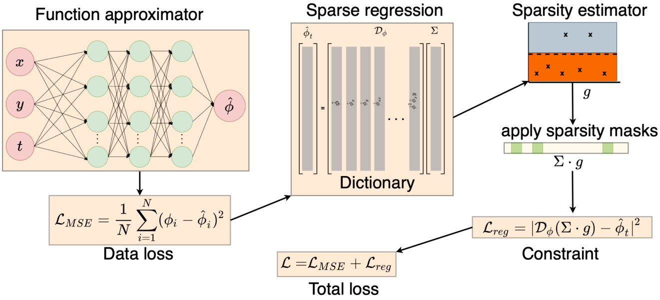
- Compute $\hat{\phi}$
- Compute MSE loss
- Do sparse regression
- Apply estimator to compute masks
- Apply masks
- Compute the constraint via regression
- Calculate total loss

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

[Both et al. '21]

Model form is unknown



Training loop

- Compute $\hat{\phi}$
- Compute MSE loss
- Do sparse regression
- Apply estimator to compute masks
- Apply masks
- Compute the constraint via regression
- Calculate total loss
- Gradient descent
- Update weights
- Iterate until convergence
- Output the learned coeff's

Initialisation

- Function approximator: Feed Forward NN
- Dictionary: polynomial order 3, derivative order 2
- Sparse regression: Lasso
- Sparsity estimator: Threshold
- Constraint: Least squares

Examples

1. One dimensional Allen-Cahn Equation
2. One dimensional Cahn-Hilliard Equation

Results

1D Allen-Cahn equation

$$\phi_t \quad = \quad 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$

1D Allen-Cahn equation

$$\phi_t = 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$
- Function approximator : Tanh, 4 hidden layers, 50 neurons

Results

1D Allen-Cahn equation

$$\phi_t = 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$
- Function approximator : Tanh, 4 hidden layers, 50 neurons
- Sparse regressor: Lasso

Results

1D Allen-Cahn equation

$$\phi_t = 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$
- Function approximator : Tanh, 4 hidden layers, 50 neurons
- Sparse regressor: Lasso
- Estimator: Threshold with 0.1 tolerance

Results

1D Allen-Cahn equation

$$\phi_t = 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$
- Function approximator : Tanh, 4 hidden layers, 50 neurons
- Sparse regressor: Lasso
- Estimator: Threshold with 0.1 tolerance
- Constraint: Least squares regression

Results

1D Allen-Cahn equation

$$\phi_t = 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

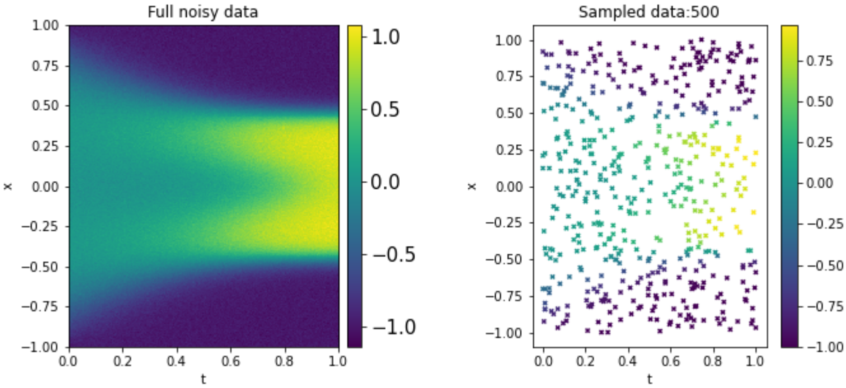
- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$
- Function approximator : Tanh, 4 hidden layers, 50 neurons
- Sparse regressor: Lasso
- Estimator: Threshold with 0.1 tolerance
- Constraint: Least squares regression
- Noise: 5%

1D Allen-Cahn equation

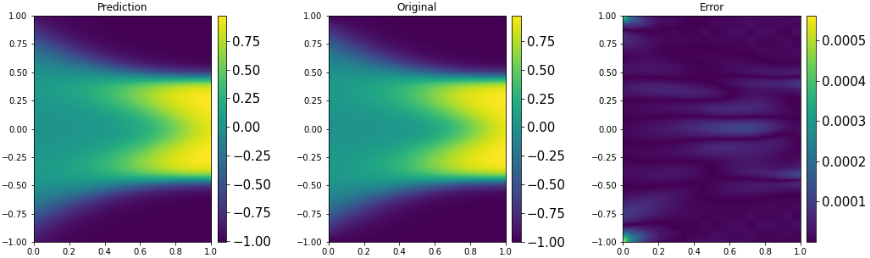
$$\phi_t = 0.01\phi_{xx} - 5(\phi^3 - \phi)$$

Parameters

- Library :
  $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$
- Function approximator : Tanh, 4 hidden layers, 50 neurons
- Sparse regressor: Lasso
- Estimator: Threshold with 0.1 tolerance
- Constraint: Least squares regression
- Noise: 5%
- Samples: 500



Full noisy data
x
t
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
−0.5
−1.0
Sampled data:500
x
t
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



Library : $[1,\ \phi_x,\ \phi_{xx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^3,\ \phi^3\phi_x,\ \phi^3\phi_{xx}]$

Coeff's : $[0, 0, 0.0103, 4.898, 0, 0, 0, 0, 0, -4.867, 0, 0]$

Results

original equation

$$\phi_t = 0.01\phi_{xx} - 5\phi^3 - 5\phi$$

Results

original equation

$$\phi_t = 0.01\phi_{xx} - 5\phi^3 - 5\phi$$

learned equation

$$\phi_t = 0.0103\phi_{xx} - 4.867\phi^3 - 4.898\phi$$

Results

1D Cahn-Hilliard equation

$$\begin{aligned}\phi_t &= (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx} \\ \phi_t &= 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}\end{aligned}$$

Parameters

- Library:

$[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$

Results

1D Cahn-Hilliard equation

$$\phi_t = (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx}$$
$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Parameters

- Library:

  $[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$
- Function approximator: Sine, 4 hidden layers, 64 neurons

Results

1D Cahn-Hilliard equation

$$\phi_t = (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx}$$
$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Parameters

- Library:

  $[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$
- Function approximator: Sine, 4 hidden layers, 64 neurons
- Sparse regression: Lasso

Results

1D Cahn-Hilliard equation

$$\phi_t = (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx}$$
$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Parameters

- Library:

  $[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$
- Function approximator: Sine, 4 hidden layers, 64 neurons
- Sparse regression: Lasso
- Estimator: Threshold with 0.5 tolerance

Results

1D Cahn-Hilliard equation

$$\phi_t = (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx}$$
$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Parameters

- Library:

  $[1,\ \phi_x,\ \boldsymbol{\phi_{xx}},\ \phi_{xxx},\ \boldsymbol{\phi_{xxxx}},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \boldsymbol{\phi^2\phi_{xx}},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$
- Function approximator: Sine, 4 hidden layers, 64 neurons
- Sparse regression: Lasso
- Estimator: Threshold with 0.5 tolerance
- Constraint: Least squares regression

Results

1D Cahn-Hilliard equation

$$\phi_t = (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx}$$
$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Parameters

- Library:

  $[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$
- Function approximator: Sine, 4 hidden layers, 64 neurons
- Sparse regression: Lasso
- Estimator: Threshold with 0.5 tolerance
- Constraint: Least squares regression
- Noise: 5%

Results

1D Cahn-Hilliard equation

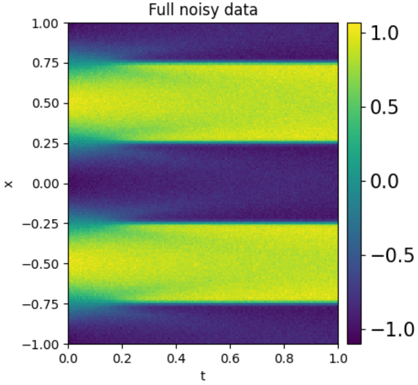
$$\phi_t = (10^{-2}(\phi^3 - \phi) - 10^{-6}\phi_{xx})_{xx}$$
$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

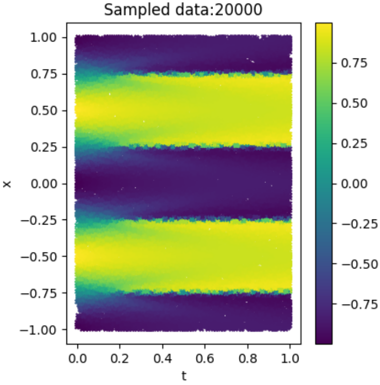
Parameters

- Library:

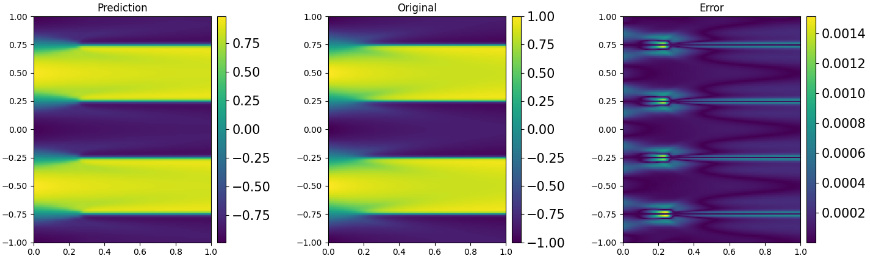
  $[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$
- Function approximator: Sine, 4 hidden layers, 64 neurons
- Sparse regression: Lasso
- Estimator: Threshold with 0.5 tolerance
- Constraint: Least squares regression
- Noise: 5%
- Samples: 20000

Results


Full noisy data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
x
0.0
0.2
0.4
0.6
0.8
1.0
t
1.0
0.5
0.0
−0.5
−1.0


Sampled data:20000
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
x
0.0
0.2
0.4
0.6
0.8
1.0
t
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Results



Library: $[1,\ \phi_x,\ \phi_{xx},\ \phi_{xxx},\ \phi_{xxxx},\ \phi,\ \phi\phi_x,\ \phi\phi_{xx},\ \phi\phi_{xxx},\ \phi\phi_{xxxx},\ \phi^2,\ \phi^2\phi_x,\ \phi^2\phi_{xx},\ \phi^2\phi_{xxx},\ \phi^2\phi_{xxxx}]$

Coeff's: $[0, 0, -2.91 \times 10^{-2}, 0, -1.17 \times 10^{-6}, 0, 0, 0, 0, 0, 0, 0, 4.89 \times 10^{-2}, 0, 0]$

Results

Original equation

$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Results

Original equation

$$\phi_t = 0.03\phi^2\phi_{xx} - 0.01\phi_{xx} - 10^{-6}\phi_{xxxx}$$

Learned equation

$$\phi_t = 4.89 \times 10^{-2}\phi^2\phi_{xx} - 2.91 \times 10^{-2}\phi_{xx} - 1.17 \times 10^{-6}\phi_{xxxx}$$

Outlook

Conclusions

- Physics enhanced machine learning framework
- Modular approach with option to integrate advanced methods
- Results are promising

Outlook

Conclusions

- Physics enhanced machine learning framework
- Modular approach with option to integrate advanced methods
- Results are promising

Ongoing work

- 2D Allen-Cahn /Cahn-Hilliard equation
- Experimental data
- Neural network architectures
- Training strategies

Outlook

Conclusions

- Physics enhanced machine learning framework
- Modular approach with option to integrate advanced methods
- Results are promising

Thank you for your attention!!

Ongoing work

- 2D Allen-Cahn /Cahn-Hilliard equation
- Experimental data
- Neural network architectures
- Training strategies

Selected References (Alphabetical)

Both, G.-J., Choudhury, S., Sens, P., and Kusters, R. (2021).
Deepmod: Deep learning for model discovery in noisy data.
Journal of Computational Physics, 428:109985.

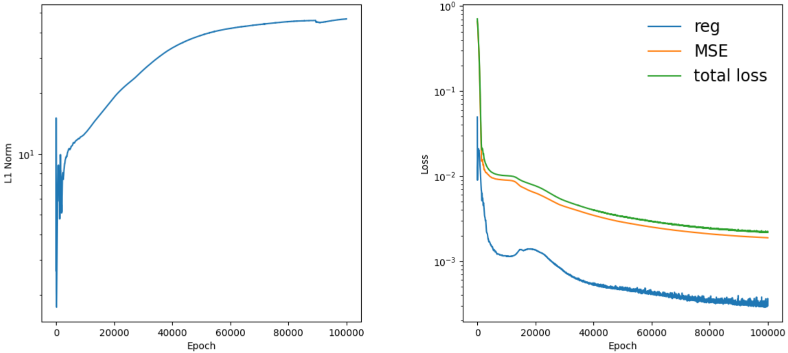
Both, G.-J., Vermarien, G., and Kusters, R. (2020).
Sparsely constrained neural networks for model discovery of pdes.
arXiv preprint arXiv:2011.04336.

Chen, Z., Liu, Y., and Sun, H. (2021).
Physics-informed learning of governing equations from scarce data.
Nature communications, 12(1):1–13.

Kaptanoglu, A. A., de Silva, B. M., Fasel, U., Kaheman, K., Goldschmidt, A. J., Callaham, J., Delahunt, C. B., Nicolaou, Z. G., Champion, K., Loiseau, J.-C., Kutz, J. N., and Brunton, S. L. (2022).
Pysindy: A comprehensive python package for robust sparse system identification.
Journal of Open Source Software, 7(69):3994.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019).
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
Journal of Computational physics, 378:686–707.

1D CH equation



1D AC equation