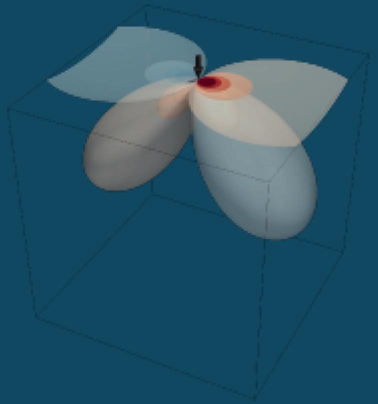
Meeting July 29, 2024
Forward and inverse modeling of wave propagation combining classical and machine learning approaches

Student: Oscar Andrés Rincón Cardeño
Advisors: Nicolas Guarín Zapata and Silvana Montoya



Applied Mechanics research group
Universidad EAFIT

Content

- Literature review
- Reproduction of the results presented in Raissi et al. (2019)
- Problem statement

Literature review

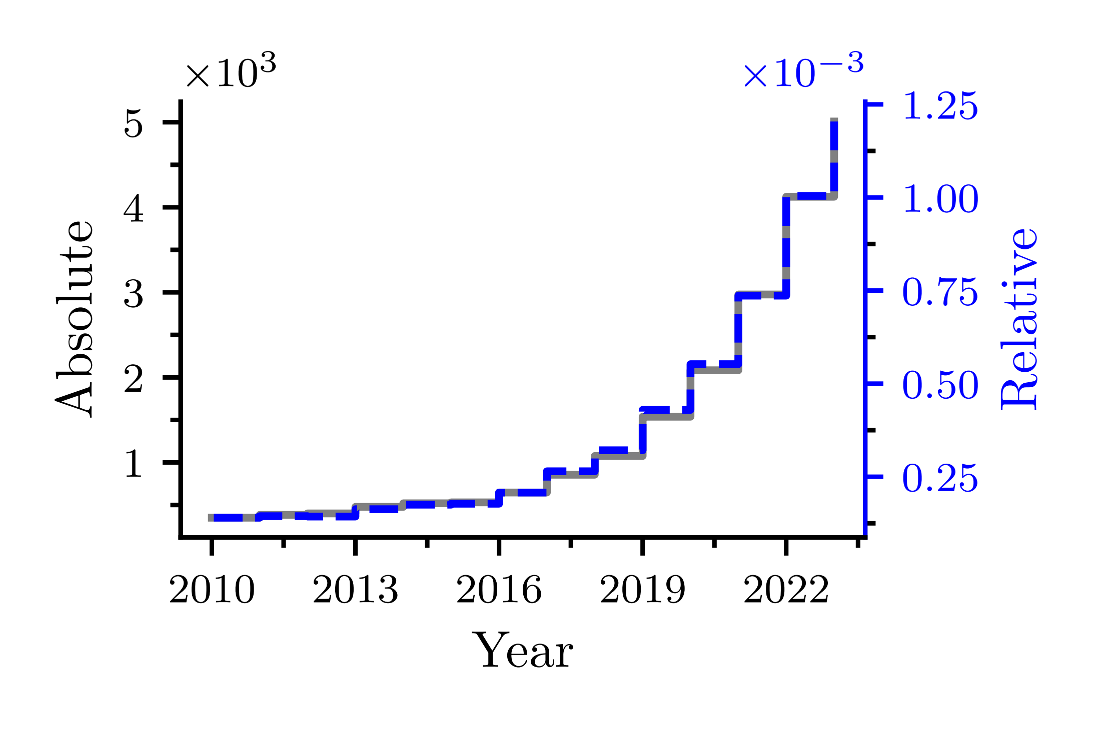
1. Modeling of Mechanical Wave Propagation - Narrative
2. Standard Numerical Methods to Model Wave Equation - Narrative
3. Machine learning Methods to Model Mechanical Wave Propagation - Narrative
4. Applications- Systematic

https://github.com/oscar-rincon/review-elastic-waves/tree/main/refs

Introduction

Queries:

- `"machine learning" OR "deep learning" OR "neural networks" AND "wave propagation" OR "wave equation" AND (modeling OR modelling OR model OR simulation)`
- `PUBYEAR > 2009 AND PUBYEAR < 2024`

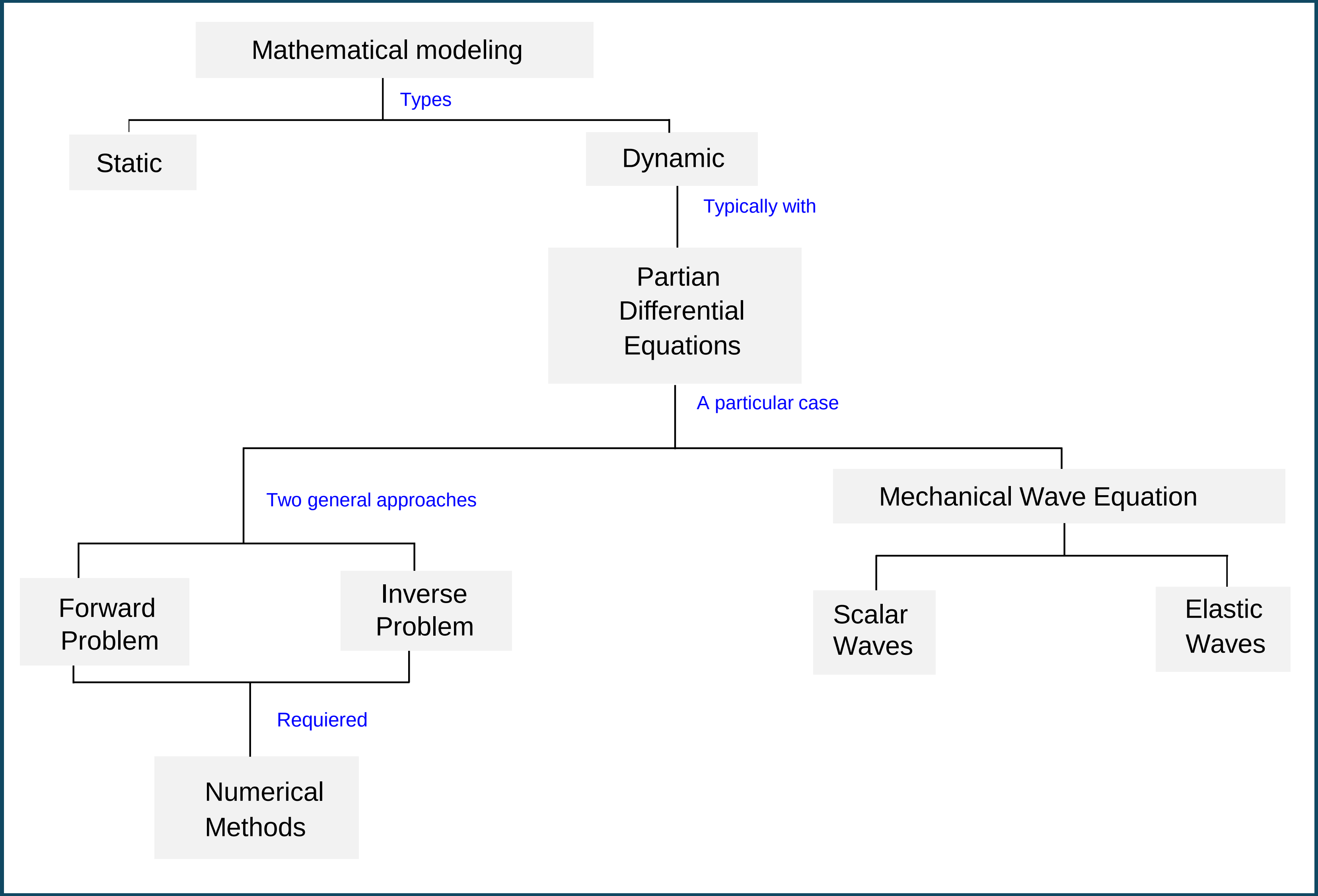


Possible causes

- ❑ Hardware
  - GPU
  - Storage
- ❑ Available data
- ❑ Open-source packages
  - Tensorflow
  - PyTorch
  - JAX

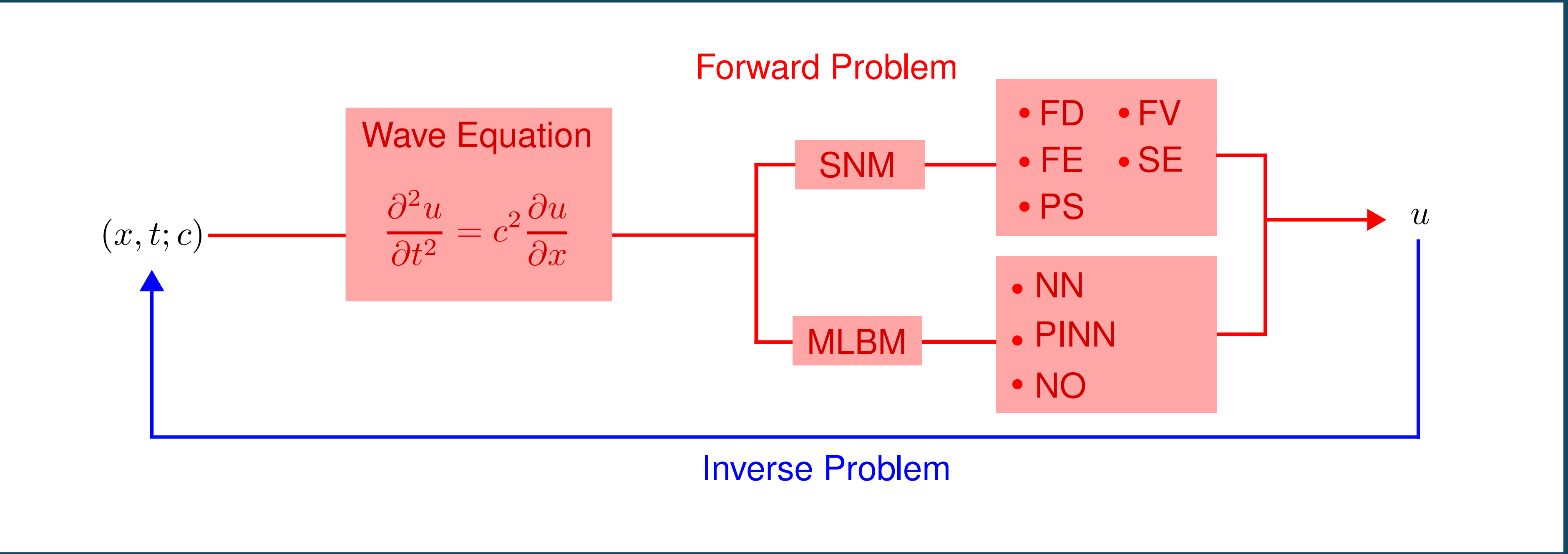
https://github.com/oscar-rincon/review-elastic-waves/tree/main/main/publications_number_year

Modeling of Mechanical Wave Propagation



https://github.com/oscar-rincon/review-elastic-waves/tree/main/figs

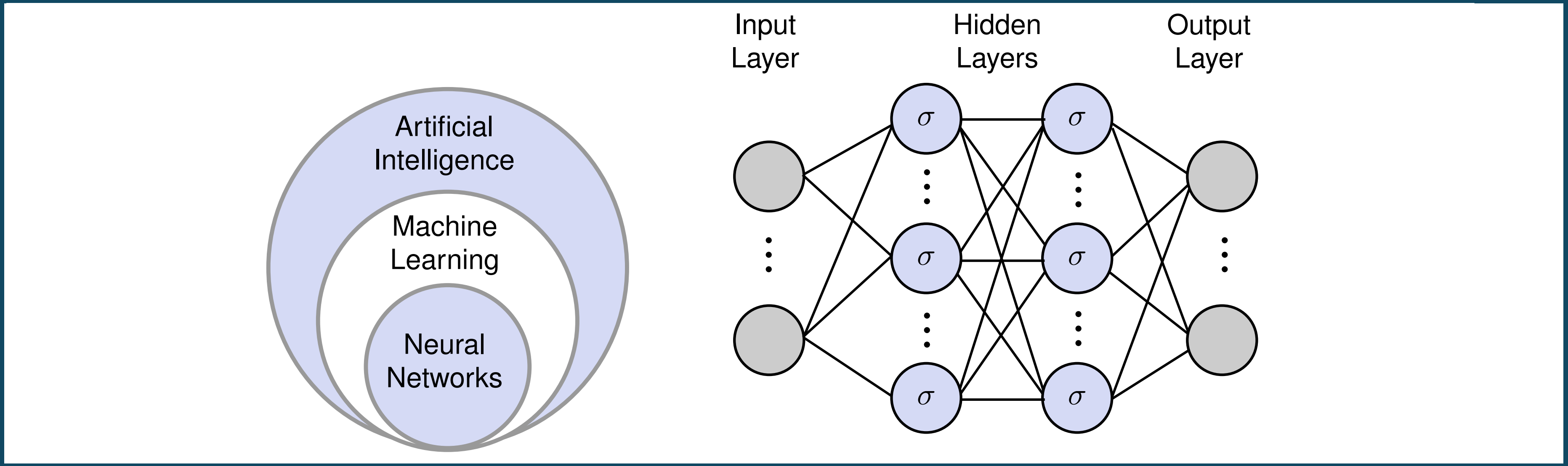
Modeling of Mechanical Wave Propagation



https://github.com/oscar-rincon/review-elastic-waves/tree/main/figs

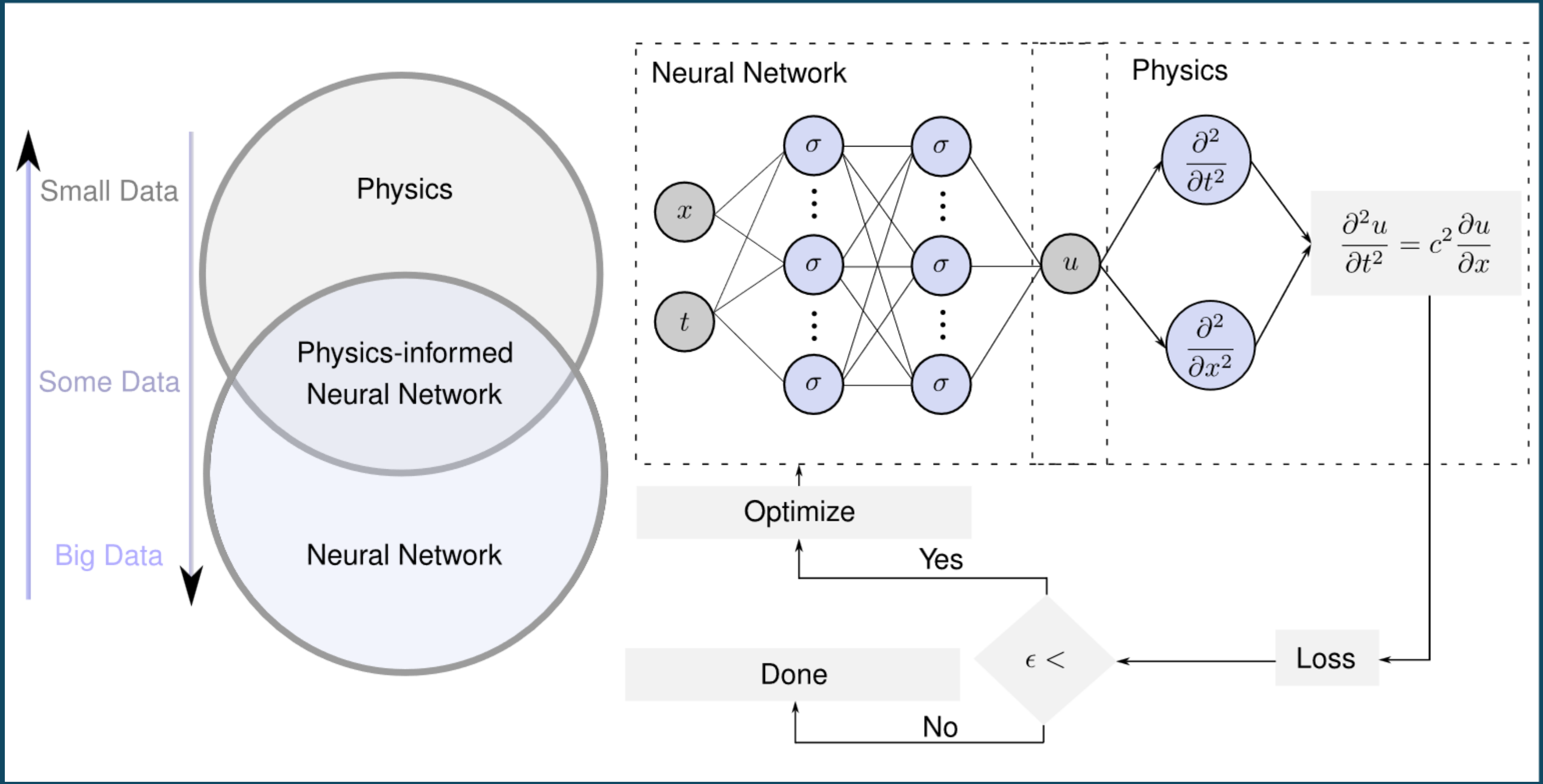
Machine learning Methods to solve Diferential Equations

- Machine Learning
  - Reinforced Suport Vector Machine
  - Neural Networks
- Neural Networks
  - Data Driven
  - Physics Based
- Shallow Networks
  - Extreme Learning Machine – Pseudo-inverse
- Deep Learning
  - Backpropagation



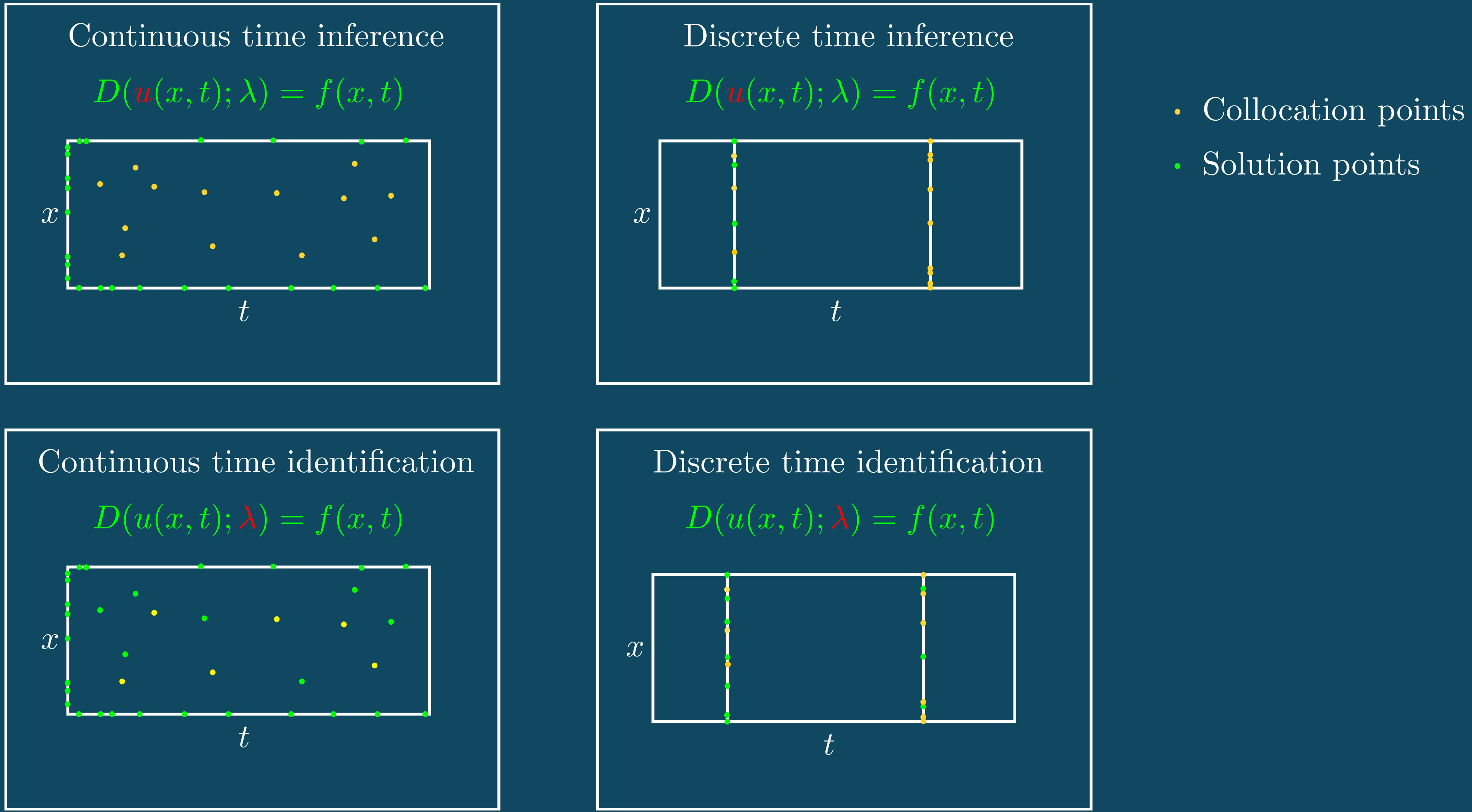
https://github.com/oscar-rincon/review-elastic-waves/tree/main/figs

Physics-informed neural networks - PINN



https://github.com/oscar-rincon/review-elastic-waves/tree/main/figs

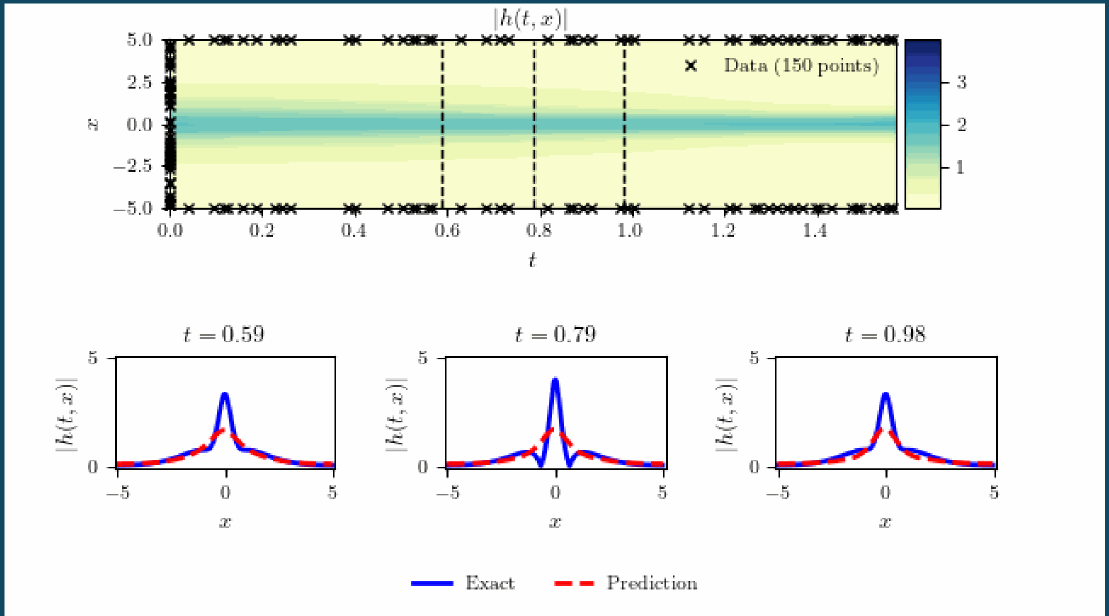
Raissi et al. (2019)



https://github.com/oscar-rincon/ReScience-PINNs

Continuous time inference

Continuous Forward Schrodinger Equation	
PDE equations	$f_u = u_t + 0.5v_{xx} + v(u^2+v^2), f_v = v_t + 0.5u_{xx} + u(u^2+v^2)$
Initial conditions	$u(0,x) = 2\text{sech}(x), v(0,x) = 0$
Periodic boundary conditions	$u(t,-5) = u(t,5), v(t,-5) = v(t,5), u_x(t,-5) = u_x(t,5), v_x(t,-5) = v_x(t,5)$
The output of net	$[u(t,x), v(t,x)]$
Layers of net	$[2] + 4 \times [100] + [2]$
Sample count from collection points	20000
Sample count from the initial condition	50
Sample count from boundary conditions	50
Loss function	$\text{MSE}_0 + \text{MSE}_b + \text{MSE}_c$

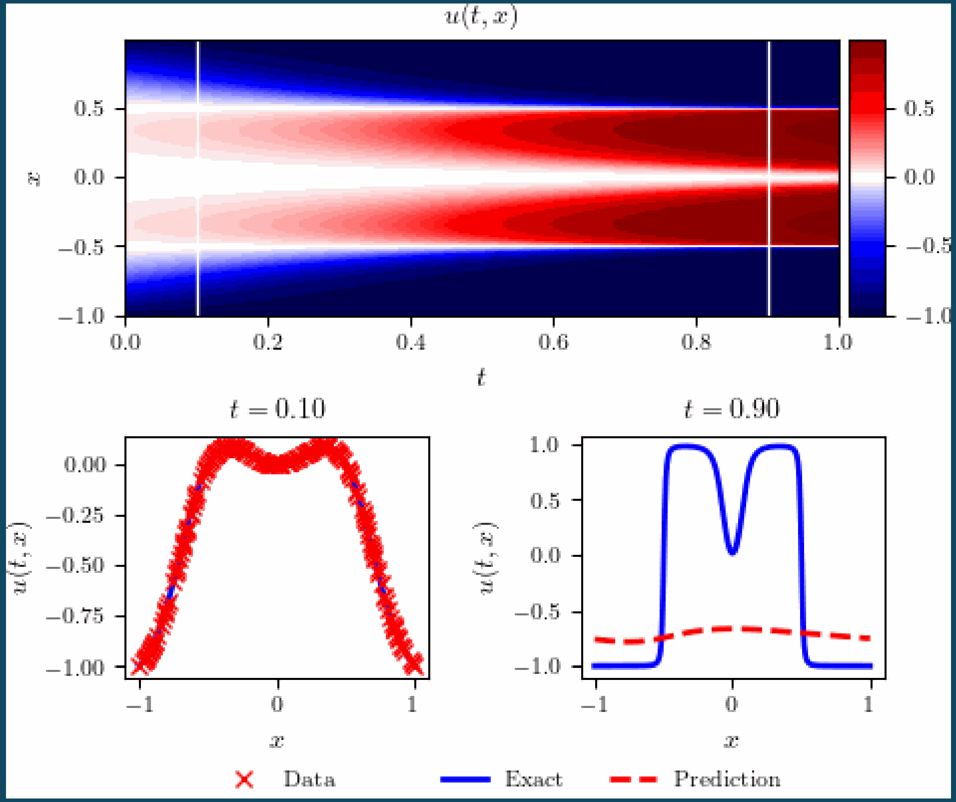


- Total training time: 5.652×10^2 seconds
- Total number of iterations: $20,968$
- L_2: 1.307×10^{-3}

https://github.com/oscar-rincon/ReScience-PINNs/tree/main/main/continuous_time_inference%20(Schrodinger)

Discrete time inference

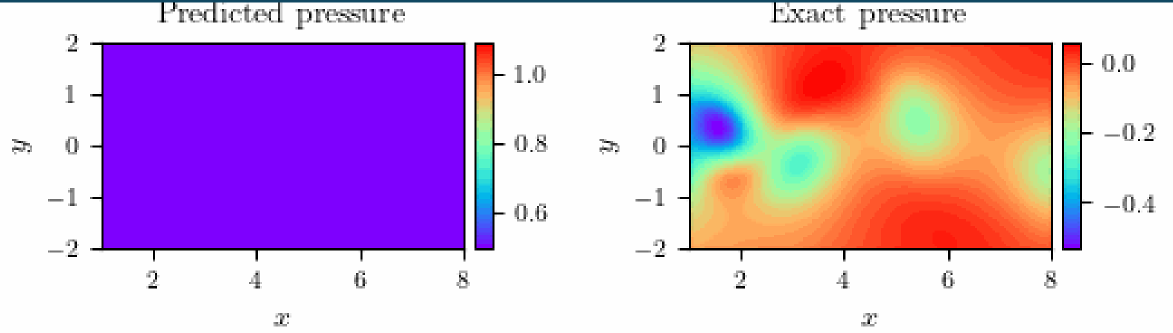
Discrete Forward AC Equation	
PDE equations	$f^{n+c_j} = 5.0u^{n+c_j} - 5.0(u^{n+c_j})^3 + 0.0001u_{xx}^{n+c_j}$
Periodic boundary conditions	$u(t,-1) = u(t,1), u_x(t,-1) = u_x(t,1)$
The output of net	$[u_1^n(x), \ldots, u_q^n(x), u_{q+1}^n(x)]$
Layers of net	$[1] + 4 * [200] + [101]$
The number of stages (q)	100
Sample count from collection points at t_0	200*
Sample count from solutions at t_0	200*
$t_0 \to t_1$	$0.1 \to 0.9$
Loss function	$\mathrm{SSE}_s^0 + \mathrm{SSE}_c^0 + \mathrm{SSE}_b^1$
* Same points used for collocation and solutions.	



- Total training time: 5.652×10^2 seconds
- Total number of iterations: $20,968$
- $\mathbb{L}_2$: 1.307×10^{-3}

https://github.com/oscar-rincon/ReScience-PINNs/tree/main/main/discrete_time_inference%20(AC)

Discrete time identification



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.827(uu_x + vu_y) = -p_x + 0.01791(u_{xx} + u_{yy})$ $v_t + 0.827(uv_x + vv_y) = -p_y + 0.01791(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.829(uu_x + vu_y) = -p_x + 0.01848(u_{xx} + u_{yy})$ $v_t + 0.829(uv_x + vv_y) = -p_y + 0.01848(v_{xx} + v_{yy})$

Continuous Inverse Navier-Stokes Equation	
PDE equations	$f = u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy}), g = v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy})$
Assumptions	$u = \psi_y, v = -\psi_x$
The output of net	$[\psi(t,x,y), p(t,x,y)]$
Layers of net	$[3] + 8 \times [20] + [2]$
Sample count from collection points	5000*
Sample count from solution	5000*
Loss function	$\mathrm{SSE}_s + \mathrm{SSE}_c$
* Same points used for collocation and solutions.	

Clean data

- Total training time: 26.440×10^3 seconds
- Total number of iterations: $231,424$
- Error in estimating λ_1: 0.007 %
- Error in estimating λ_2: 1.864 %

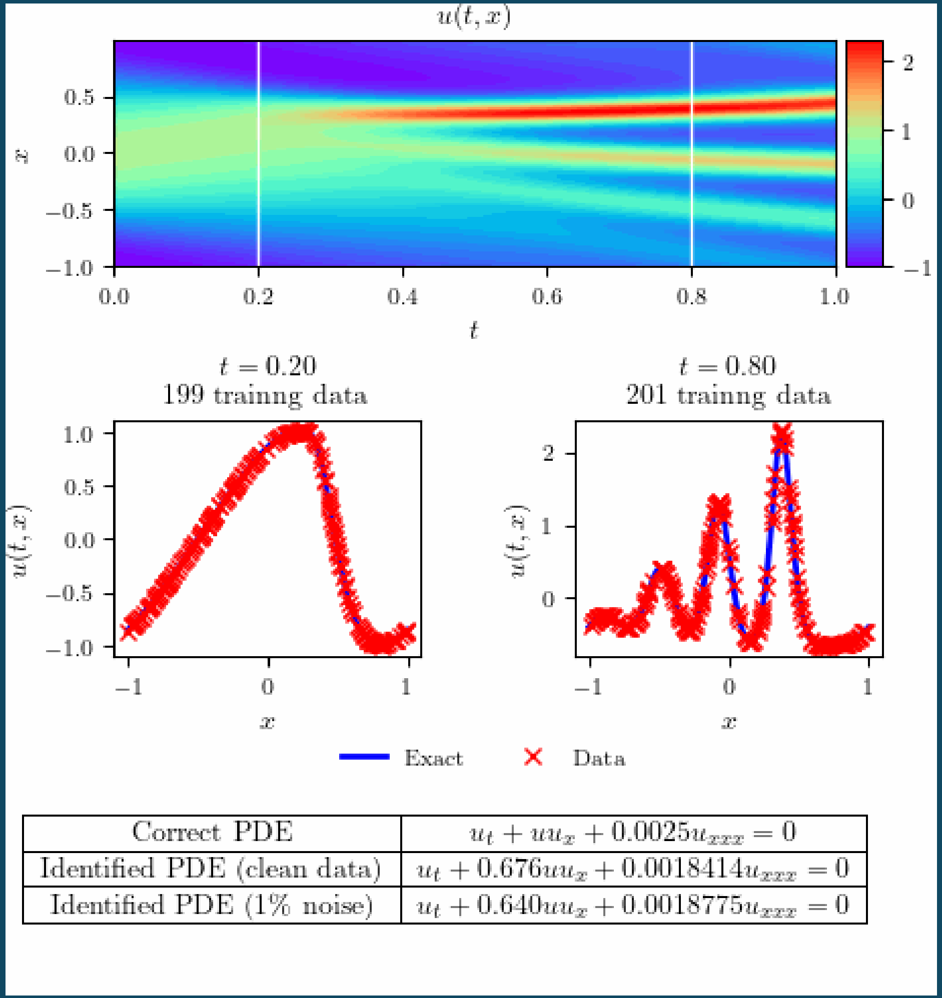
Noisy data

- Total training time: 26.236×10^3 seconds
- Total number of iterations: $228,766$
- Error in estimating λ_1: 0.029 %
- Error in estimating λ_2: 3.290 %

https://github.com/oscar-rincon/ReScience-PINNs/tree/main/main/continuous_time_identification%20(Navier-Stokes)

Continuous time identification

Discrete Inverse KdV Equation	
PDE equations	$f^{n+c_j} = -\lambda_1 u^{n+c_j} u_x^{n+c_j} - \lambda_2 u_{xxx}^{n+c_j}$
Dirichlet boundary conditions	$u(t,-1) = u(t,1) = 0$
The output of net	$[u_1^n(x), \dots, u_q^n(x), u_{q+1}^n(x)]$
Layers of net	$[1] + 3 \times [50] + [50]$
The number of stages (q)	50
Sample count from collection points at t_0	250^*
Sample count from solutions at t_0	250^*
$t_0 \to t_1$	$0.1 \to 0.9$
Loss function	$\mathrm{SSE}_s^0 + \mathrm{SSE}_c^0 + \mathrm{SSE}_b^1$
* Same points used for collocation and solutions.	



Clean data

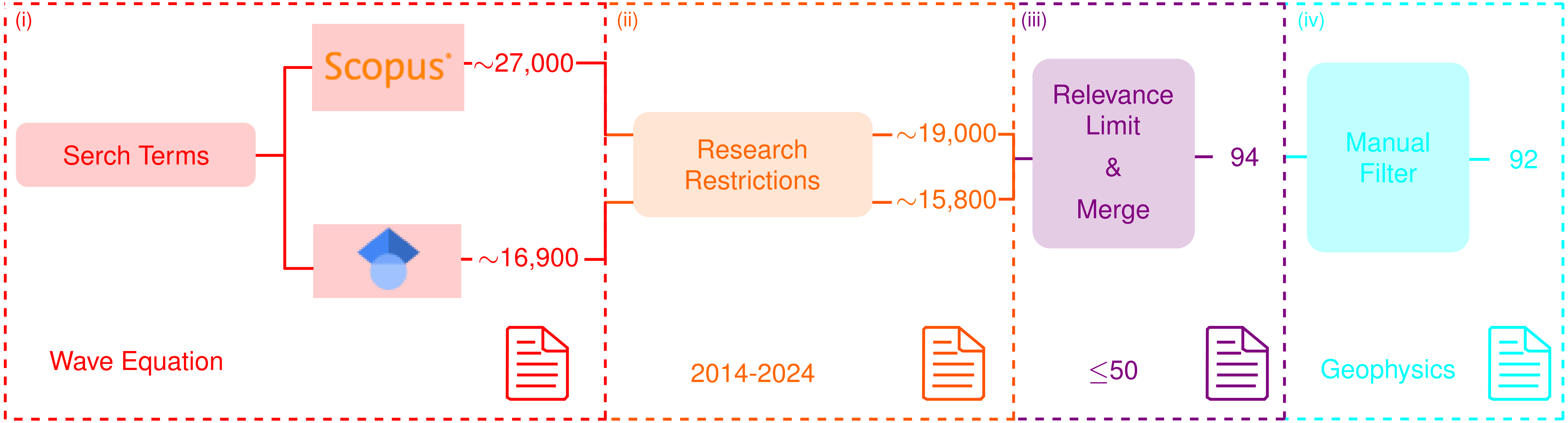
- Total training time: 1.53754×10^3 seconds
- Total number of iterations: $61,348$
- Error in estimating λ_1: 0.004 %
- Error in estimating λ_2: 0.005 %

Noisy data

- Total training time: 1.7998×10^3 seconds
- Total number of iterations: $53,235$
- Error in estimating λ_1: 0.119 %
- Error in estimating λ_2: 0.048 %

https://github.com/oscar-rincon/ReScience-PINNs/tree/main/main/discrete_time_identification%20(KdV)

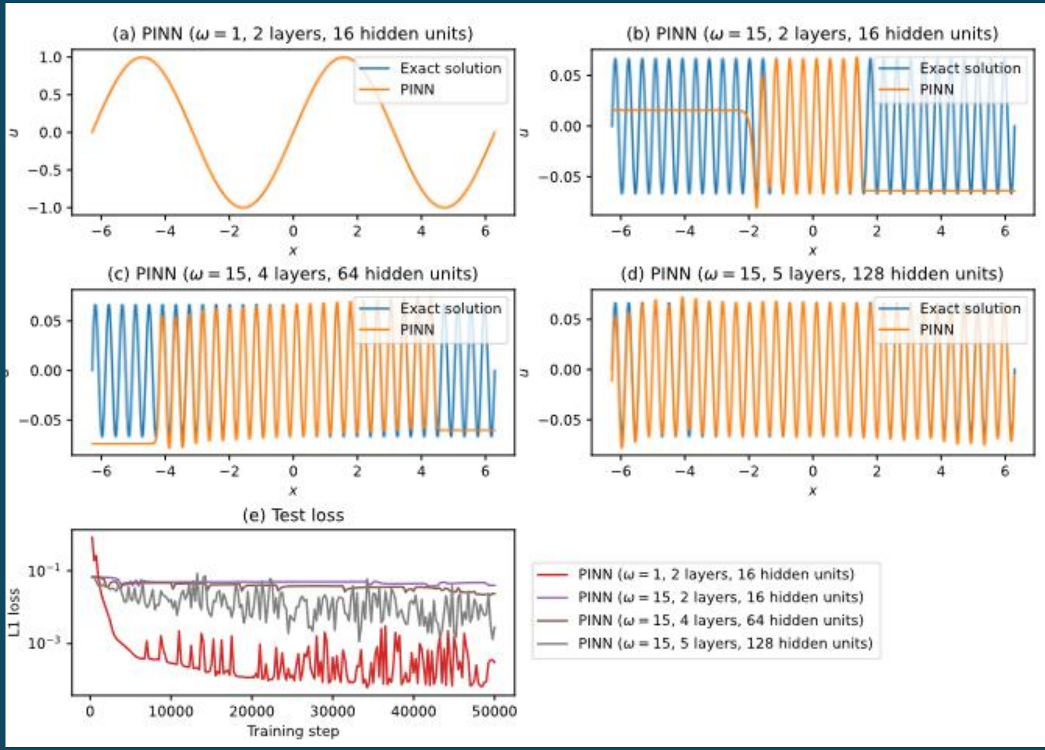
Systematic Review of Applications



Advantages / Disadvantages of PINN vs Standard numerical methods

- ❑ Simple implementation -> Interdisciplinary
- ❑ Easily adaptable to both problems with (inverse) or without data (forward)
- ❑ Fast model evaluation – (seconds)
- ❑ Possibility of transfer learning

- ❑ Uncertainty on required architecture
- ❑ Unacurate
- ❑ Slow training times - (hours vs minutes)
- ❑ Possible local minimals
- ❑ Function approximation of Large or complex domains



Extreme Learning Machine - ELM

- It runs quickly – pseudoinverse
- Comparable accuracies to standard methods
- Not local minimals
- Can be used with orthogonal neural networks
- Still not applied to elastic wave equations