École polytechnique fédérale de Lausanne
Computer Vision Lab (CVLab)

# Informing Neural Networks with Simplified Physics for Better Flow Prediction

MSc in Computational Sciences and Engineering
Master’s project

**Student**
Fedor Sergeev
fedor.sergeev@epfl.ch

**Supervisors**
Prof. Pascal Fua
Dr. Jonathan Donier

February 10, 2023

# Contents

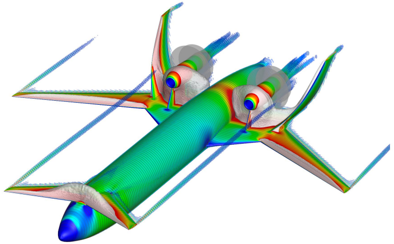
1. Motivation
    - Deep learning in engineering design
    - Physics-informed neural networks
2. Method
    - Combining physics-informed and surrogate models
    - Simplifying Navier-Stokes to potential flow
3. Experiments
    - Solving potential flow with PINNs
    - Predicting viscous flow around an airfoil with PINN+DNN

# Contents

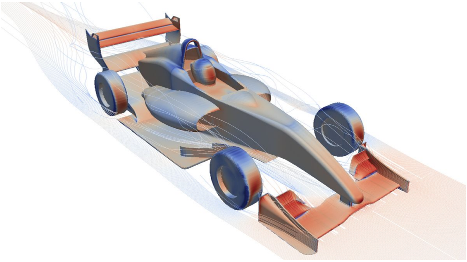

1. Motivation
    - Deep learning in engineering design
    - Physics-informed neural networks
2. Method
    - Combining physics-informed and surrogate models
    - Simplifying Navier-Stokes to potential flow
3. Experiments
    - Solving potential flow with PINNs
    - Predicting viscous flow around an airfoil with PINN+DNN

# CFD applications

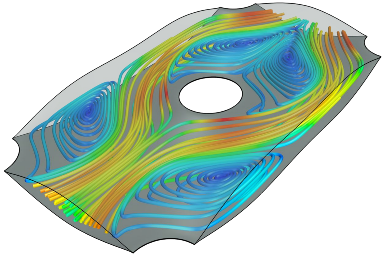
Aerospace



Automotive



Heat exchangers



Energy

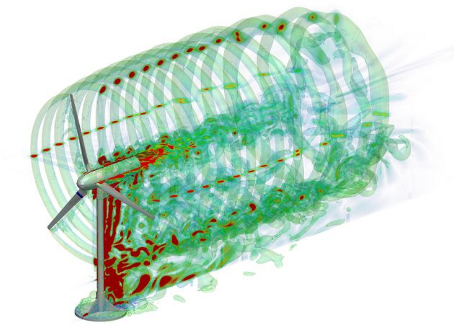
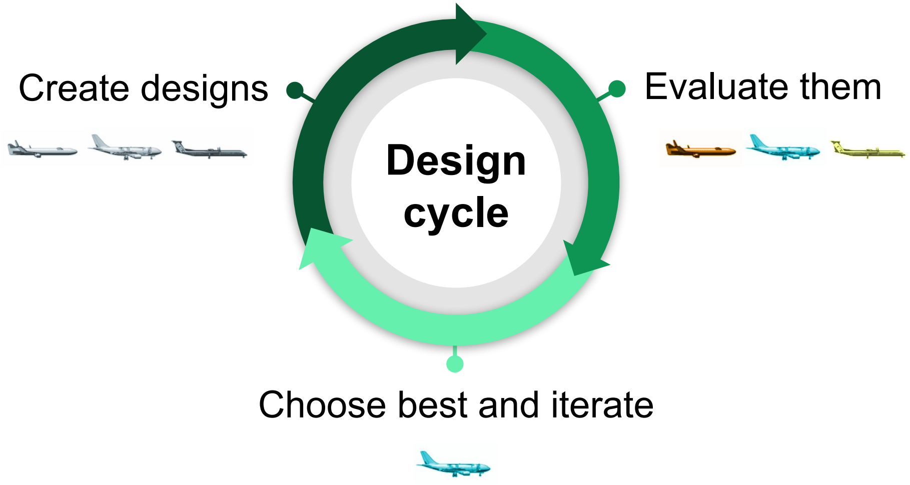


Image sources: [2-4]

# Deep learning in engineering design

- Need: a method to quickly evaluate the designs
- Numerical methods
  - Accurate
  - Physical predictions
  - Can be slow
- Deep learning
  - Train on simulation data
  - Much quicker
  - Lower accuracy
  - Possibly unphysical predictions




⇒ How to improve a DL model when the underlying physics is known?

Image sources: [1]

# Physics-informed neural networks (PINNs)

## How to incorporate physics into a model?

### Incorporating physics into ML

#### Data

**Nondimentionalisation**

Remove physical dimensions from equations by substitution

#### Model architecture

**Hard constraints**

Modify the output to satisfy boundary conditions (BCs)

$$\tilde{f}(x) = f(x) \cdot b(x) + f_D(x)$$

$$b(x) = \begin{cases} 0, & \text{on the boundary} \\ > 0, & \text{otherwise} \end{cases}$$

#### Training procedure

**PDE-residuals as loss terms**

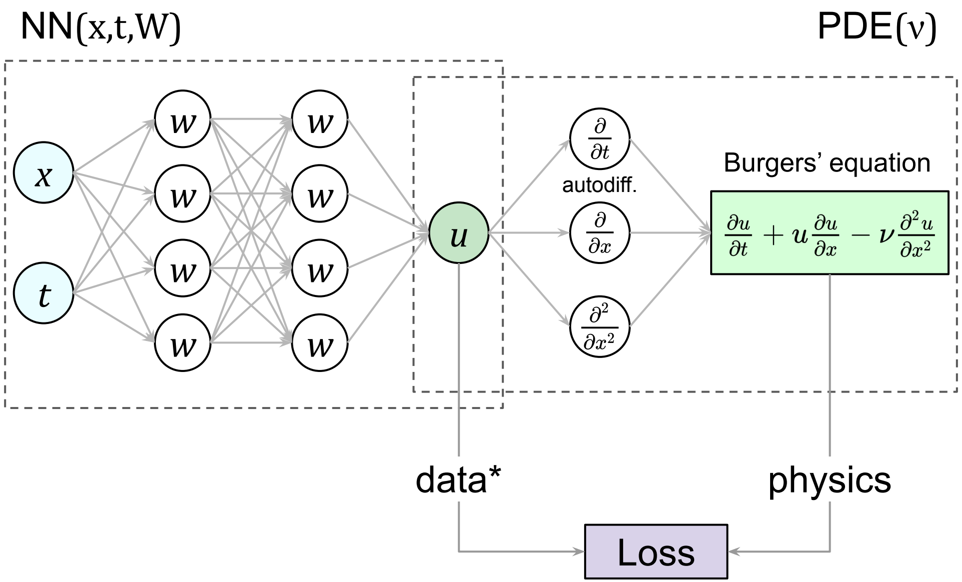
Train a (un-)supervised model to satisfy PDEs

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}}$$

# Physics-informed neural networks (PINNs)

## Using PDE-residuals as losses

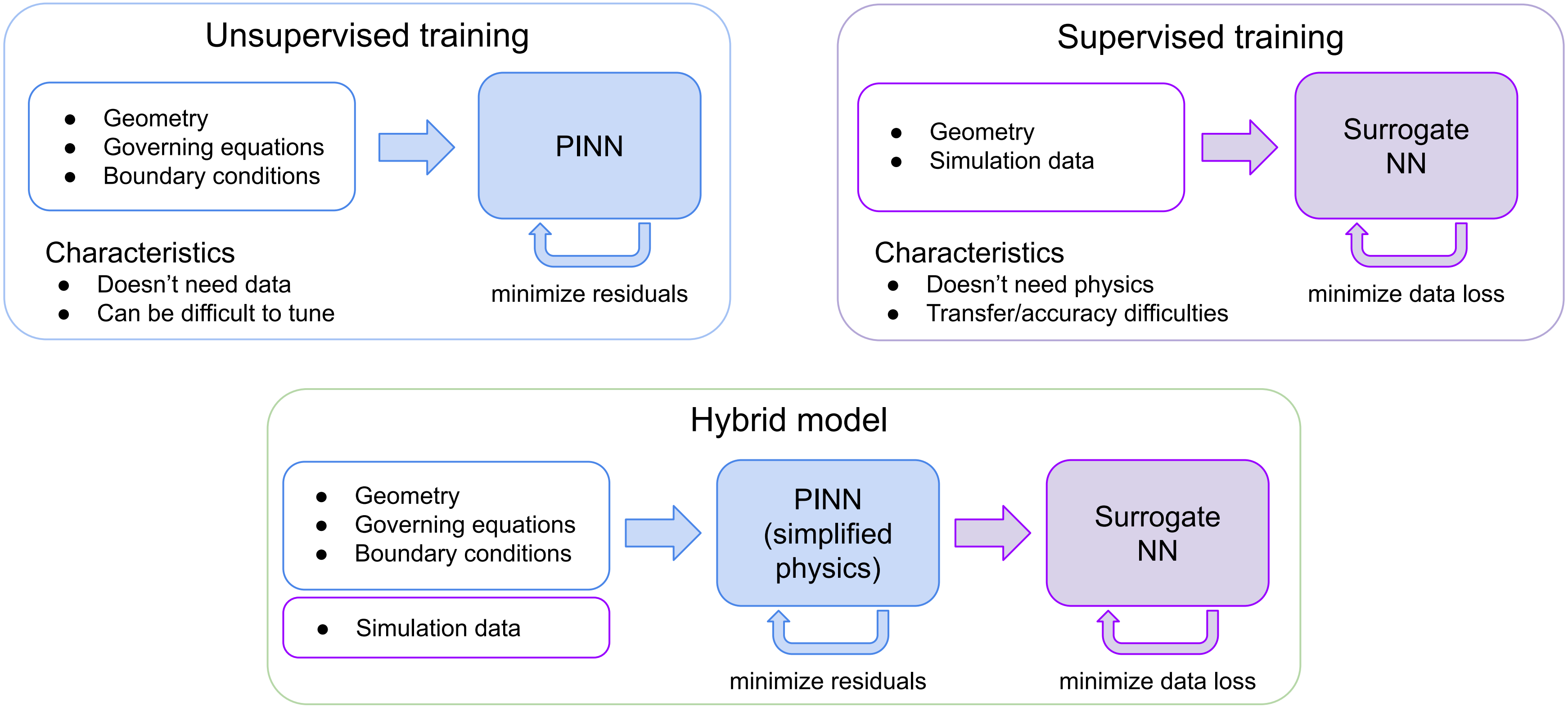
- Explicitly incorporate PDEs into the model
- Challenges
  - Choosing loss weights
  - Ensuring convergence stability and speed



# Contents


1. Motivation
    - Deep learning in engineering design
    - Physics-informed neural networks
2. Method
    - Combining physics-informed and surrogate models
    - Simplifying Navier-Stokes to potential flow
3. Experiments
    - Solving potential flow with PINNs
    - Predicting viscous flow around an airfoil with PINN+DNN

# Combining physics-informed and surrogate models



# Navier-Stokes equations

- Variables
  - Coordinate $\boldsymbol{x}$
  - Time $t$
  - Velocity $\boldsymbol{u}$
  - Pressure $p$
  - Kinematic viscosity $\nu$
  - Dynamic viscosity $\mu$
- Equations
  - Conservation of linear momentum
  - Conservation of mass (= continuity equation)
  - Boundary conditions
    - Initial
    - Dirichlet
    - Neumann

Assumptions
- ➔ continuous 1-phase fluid
- ➔ incompressible
- ➔ steady flow
- ➔ no external forces/sources
- ➔ constant temperature

$$
\begin{cases}
\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \nabla^2 \boldsymbol{u} = -\frac{1}{\rho}\nabla p \\
\nabla \cdot \boldsymbol{u} = 0 \\
\boldsymbol{u}|_{t=0} = \boldsymbol{u}_0 \\
\boldsymbol{u}|_{\Gamma_D} = f_D(\boldsymbol{x}, t) \\
-p\boldsymbol{n} + \mu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} |_{\Gamma_N} = \boldsymbol{g}_n(x, t)
\end{cases}
$$

# Navier-Stokes equations → potential flow

Navier-Stokes

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \nabla^2 \boldsymbol{u} = -\frac{1}{\rho}\nabla p$$

$$\nabla \cdot \boldsymbol{u} = 0$$

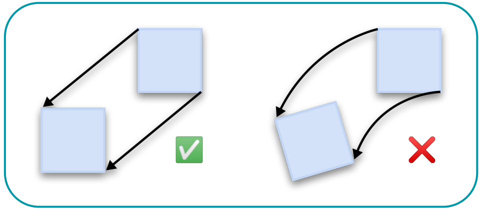
+ irrotationality

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{u} = 0$$

Potential flow

$$\boldsymbol{u} = \nabla F$$

$$\Delta F = 0$$



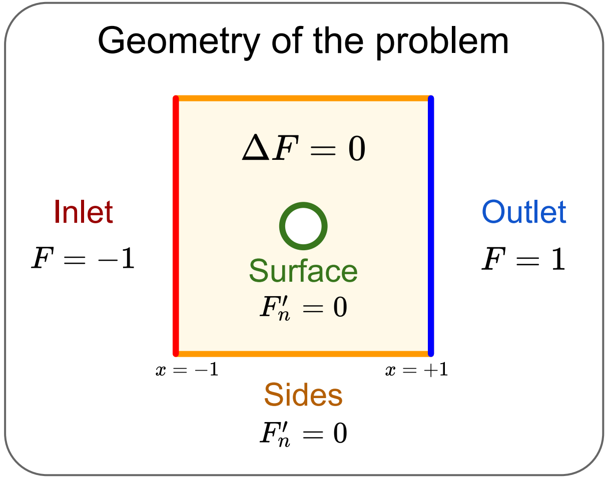
# Contents


1. Motivation
   - Deep learning in engineering design
   - Physics-informed neural networks
2. Method
   - Combining physics-informed and surrogate models
   - Simplifying Navier-Stokes to potential flow
3. Experiments
   - Solving potential flow with PINNs
   - Predicting viscous flow around an airfoil with PINN+DNN

# PINNs for potential flow

## Soft constraints

- Laplace problem with mixed BCs
- Analytical solution available for a circular cylinder
- Model
  - Fully-connected NN [1, 100x5, 1]
  - MSE for all terms, using Adam with LR decay
- Execution
  - Sample points in the domain and on the boundaries
  - Predict solution
  - Evaluate PDE and BC residuals
  - Compute individual losses and the weighted sum
  - Calculate grads and do the optimization step




$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{inlet}} + \mathcal{L}_{\text{outlet}} + \mathcal{L}_{\text{sides}} + 20 \cdot \mathcal{L}_{\text{surface}}$$

$$\mathcal{L}_{\text{PDE}} = \text{MSE}(\Delta \hat{F}(x), 0)$$

$$\mathcal{L}_{\text{inlet}} = \text{MSE}(F(x_{\text{inlet}}), -1) \qquad \mathcal{L}_{\text{sides}} = \text{MSE}(\hat{F}'_n(x_{\text{sides}}), 0)$$

$$\mathcal{L}_{\text{outlet}} = \text{MSE}(F(x_{\text{outlet}}), +1) \qquad \mathcal{L}_{\text{surface}} = \text{MSE}(\hat{F}'_n(x_{\text{surface}}), 0)$$

# PINNs for potential flow

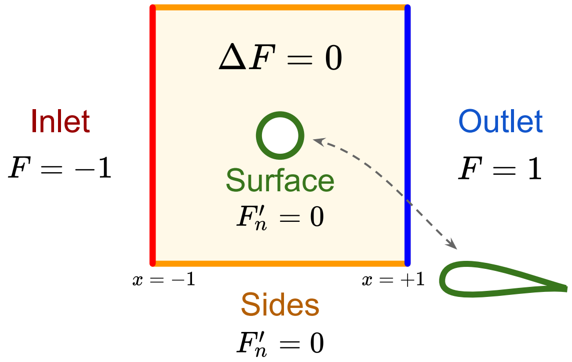
## Hard constraints

- Laplace problem with mixed BCs
- Analytical solution available for a circular cylinder
- Model
  - Fully-connected NN [1, 100x5, 1]
  - MSE for all terms, using Adam with LR decay
- “Hard constraints”
  - Reduces the number of loss terms
  - Transform model output as

$$\widetilde{NN}(x, y) = (1 - x)(1 + x) \cdot NN(x, y) + x$$

$$\Longrightarrow$$

Geometry of the problem




$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \cancel{\mathcal{L}_{\text{inlet}} + \mathcal{L}_{\text{outlet}}} + \mathcal{L}_{\text{sides}} + 20 \cdot \mathcal{L}_{\text{surface}}$$

$$\mathcal{L}_{\text{PDE}} = \text{MSE}(\Delta \hat{F}(x), 0)$$

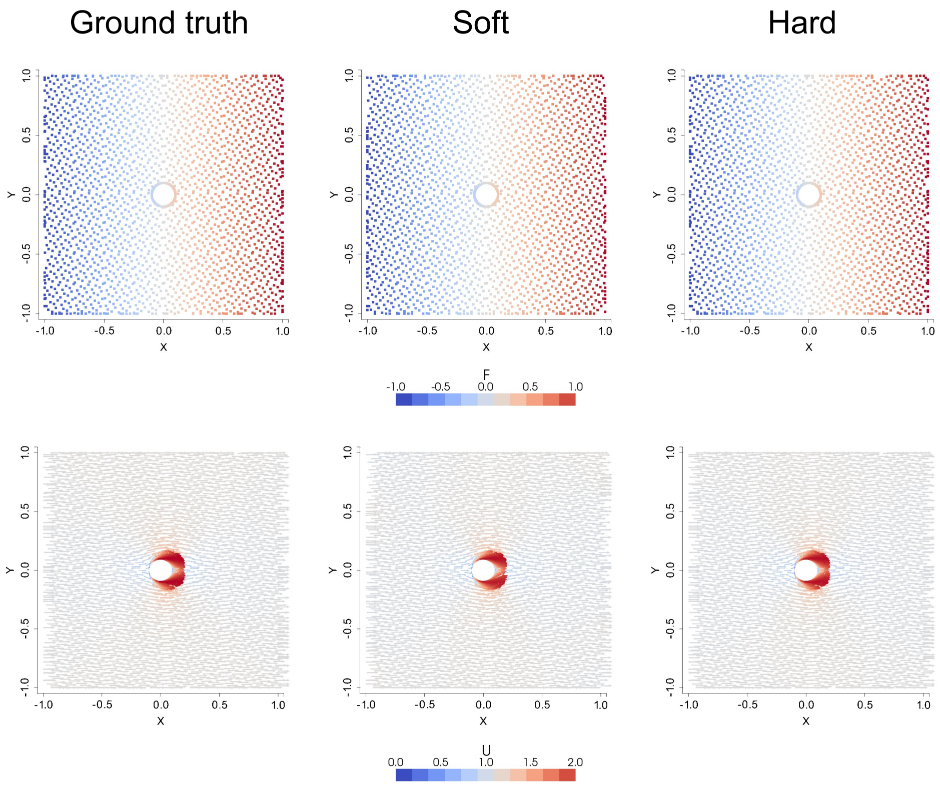
$$\cancel{\mathcal{L}_{\text{inlet}} = \text{MSE}(F(x_{\text{inlet}}), -1)} \qquad \mathcal{L}_{\text{sides}} = \text{MSE}(\hat{F}'_n(x_{\text{sides}}), 0)$$

$$\cancel{\mathcal{L}_{\text{outlet}} = \text{MSE}(F(x_{\text{outlet}}), +1)} \qquad \mathcal{L}_{\text{surface}} = \text{MSE}(\hat{F}'_n(x_{\text{surface}}), 0)$$

# PINNs for potential flow

## Results: cylinder

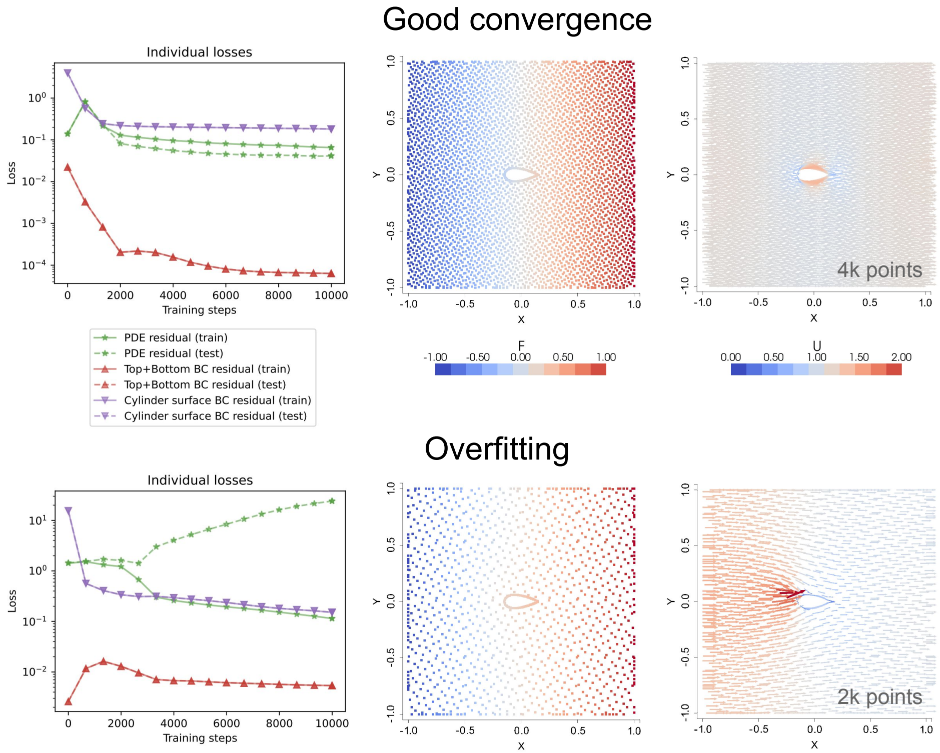
- Solved pot. flow
  - Cylinder: soft and hard
  - Airfoil: hard constraints
- Convergence and training speed heavily depends on the number of sampled points
- Overfitting may result in non-physical predictions



# PINNs for potential flow

## Results: airfoil

- Solved pot. flow
  - Cylinder: soft and hard
  - Airfoil: hard constraints
- Convergence and training speed heavily depends on the number of sampled points
- Overfitting may result in non-physical predictions



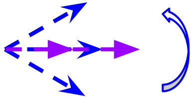
# Viscous flow with PINN+DNN

## Data

- Flow around NACA airfoils
  - Incompressible fluid at $T$=*const*
  - Simulated with RANS
  - 1096 samples (90% train)
  - Varying geometry
  - Varying freestream velocity $\boldsymbol{V}$
- Task
  - Given the geometry and $\boldsymbol{V}$
  - Predict velocity, pressure and turbulent viscosity

Freestream velocity

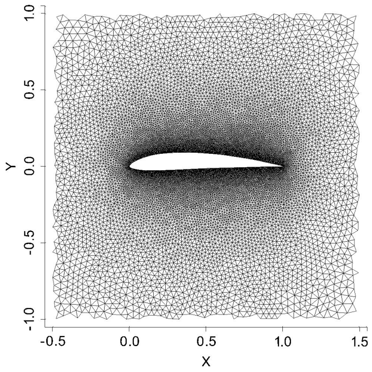
$$\boldsymbol{V} = \begin{pmatrix} V\cos\varphi \\ V\sin\varphi \end{pmatrix}$$



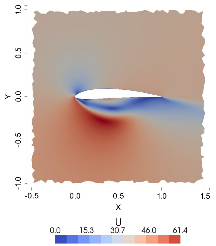
$$V \in [10, 100]$$

$$\varphi \in \left[-\frac{\pi}{8}, +\frac{\pi}{8}\right]$$

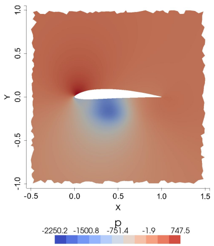
Mesh



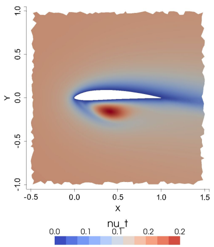

Velocity




Pressure



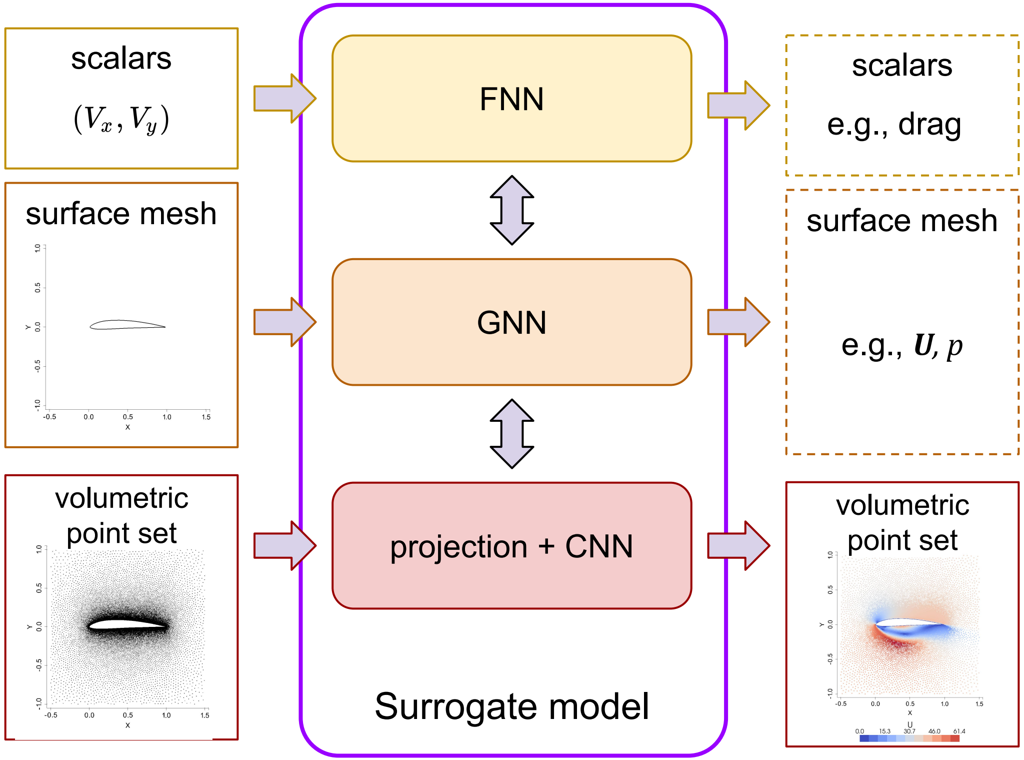

Turb. visc.



# Viscous flow with PINN+DNN

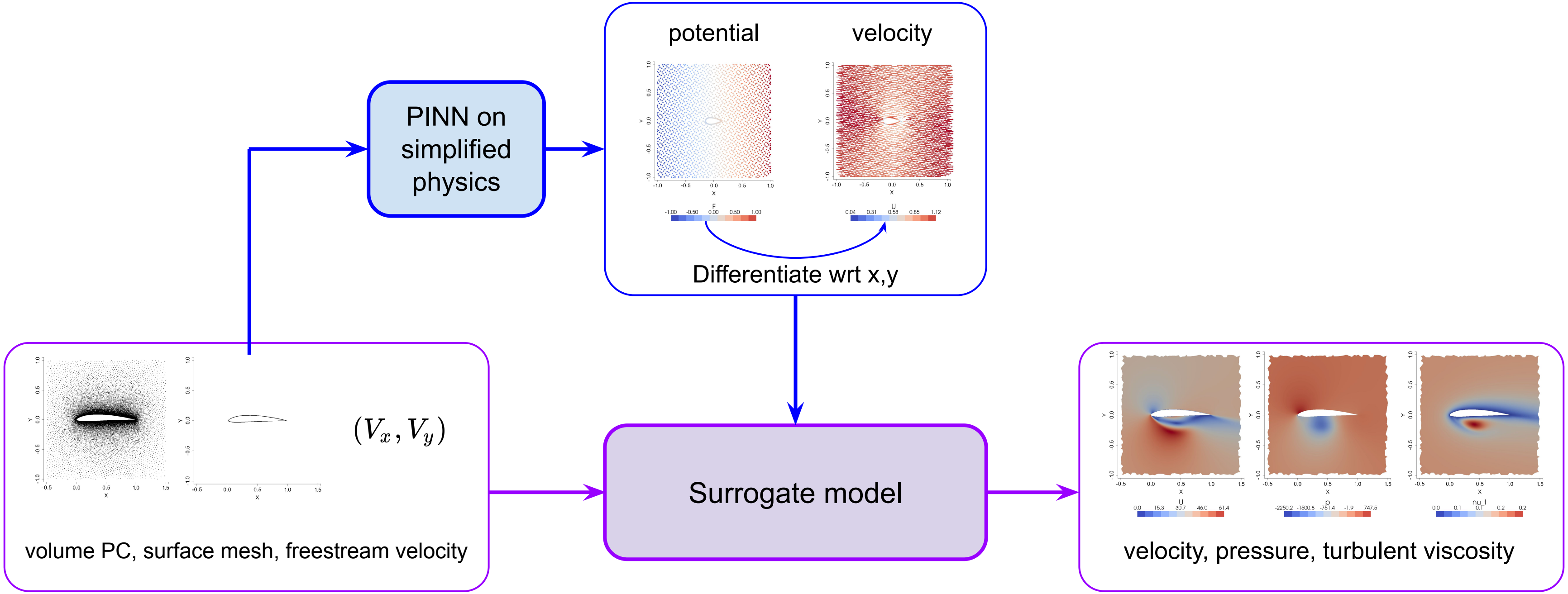
## Surrogate DNN architecture

- Zampieri et al.
- Geometric DNN for predicting scalars, values on meshes and point sets
- We use it only to predict on point sets



# Viscous flow with PINN+DNN

## Hybrid PINN+DNN architecture

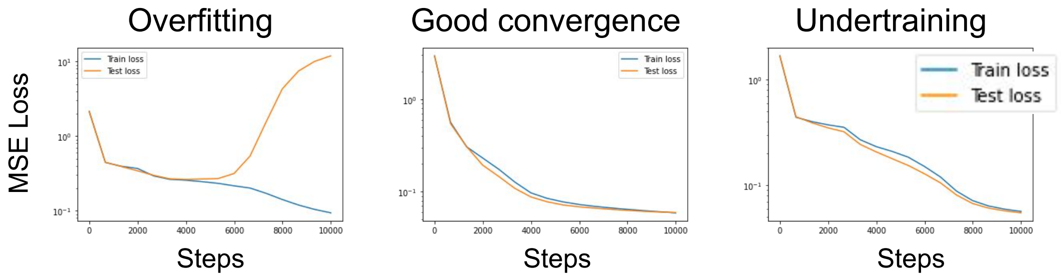


# Viscous flow with PINN+DNN

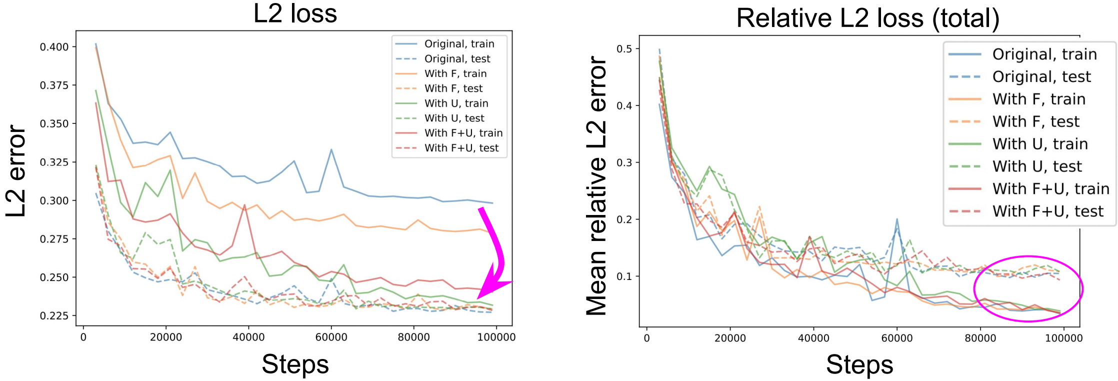
## Results

- Pot. flow training takes 30h (±2 min/sample, 1k samples)
- Distance between train and test significantly reduced
- Accuracy is the same (marginal improvement)

PINN convergence examples




Surrogate DNN convergence

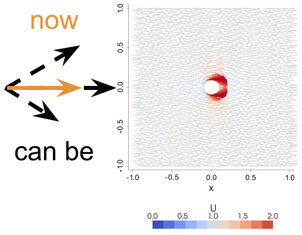


# Future work

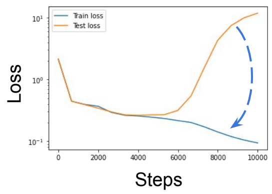
## Addressing limitations, transfer learning

- Addressing limitations
  - Directional potential flow
  - PINN training stability
  - Uncertainty estimation
- Transfer learning
  - Unsupervised PINNs work on out-of-distribution samples

### Directional pot. flow

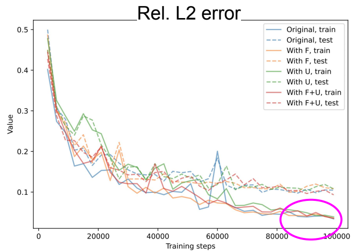



### PINN training stability



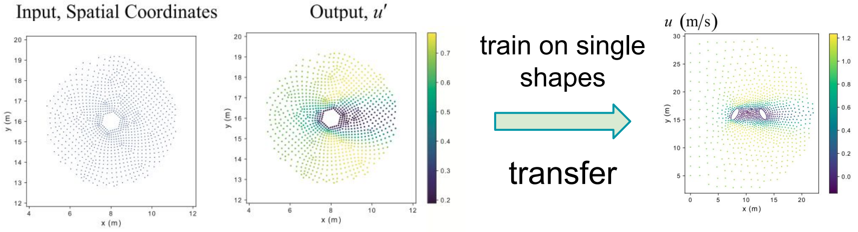

Adjust training parameters for better convergence

### Uncertainty estimation



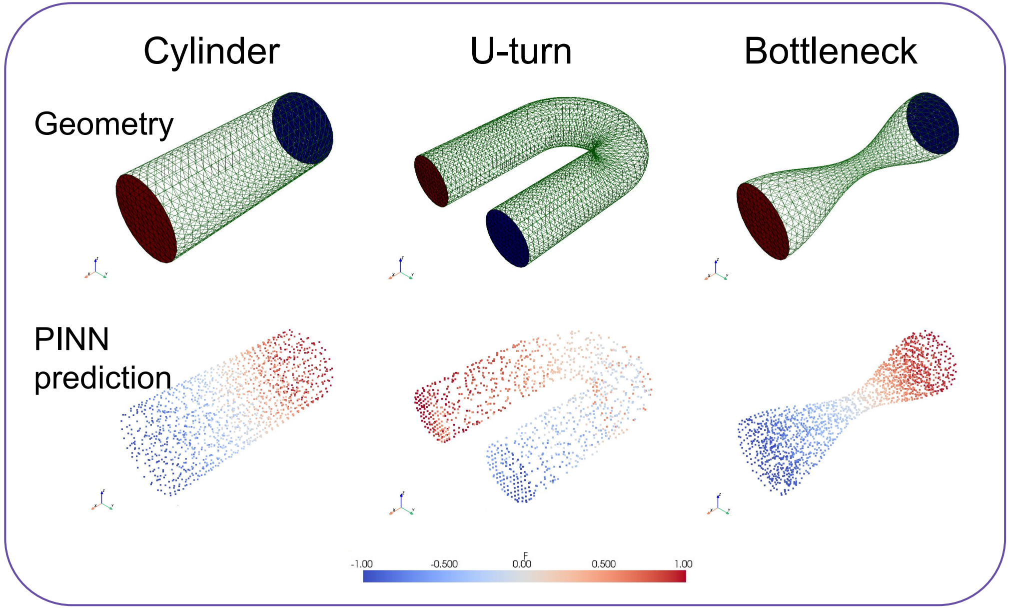

Very close ⇒ we need confidence intervals

### Transfer learning



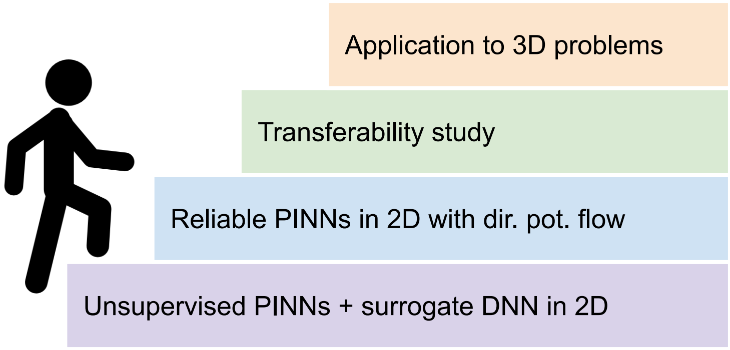
# Future work
## Application to 3D problems

- Applications to 3D problems
  - External flow
  - Automotive, aerospace, etc.
  - Internal flow
  - Heat exchangers, energy, etc.
- Potential flow in circular tubes
  - Inside $\Delta F = 0$
  - Inlet $F = -1$
  - Outlet $F = 1$
  - Surface $F'_n = 0$



# Conclusion

- DNNs are becoming increasingly important for engineering
- Hybrid PINN+DNN approaches can lead to improved
  - Accuracy
  - Reliability
  - Transferability
- Simplified physics allows for easier training of PINNs
- Experiments for external flow around an airfoil show potential



# Photographic credits

[1] Chang et al. in ShapeNet: An Information-Rich 3D Model Repository
[2] AlbertsFlyStudio, CC BY 4.0 https://creativecommons.org/licenses/by/4.0, via Wikimedia Commons
[3] By Flocess - Own work, CC BY-SA 4.0, https://en.wikipedia.org/w/index.php?curid=61866681
[4] By Fraunhofer-Gesellschaft, https://www.windenergie-cfd.de/en/aerodynamics-for-wind-turbines/Meshing-and-CFD-Simulations-of-Wind-Turbines.html
[5] Kashefi, Mukerjib in Physics-informed PointNet
[6] Modified from Mattheakis et al. Physical Symmetries Embedded in Neural Networks
[7] Redrawn from Karniadakis et al., Physics-informed machine learning
[8] (modified) By FSund - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53827516
[9] Man walking up the stairs icon: https://i.pinimg.com/originals/d1/89/c5/d189c5d9b4d9491a2bc30826976a062a.png
[10] Eivazi et al., Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations

# Additional material

# Physics-informed neural networks (PINNs)

## How to incorporate physics into a model?

### Incorporating physics into ML

#### Data

**Augmentation**

Augment data to capture observational biases

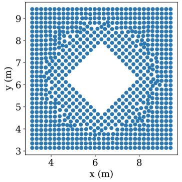
$$f(x) = f(-x)$$

$$X = \{x_i,\ f(x_i)\}$$

$$\Rightarrow\ \tilde{X} = X \cup \{-x_i, f(x_i)\}$$

**Sampling**

Sample around the shape to encode it

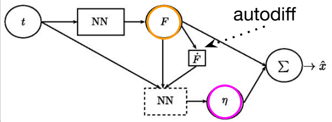



#### Model architecture

**Enforcing symmetry**

**Hard constraints**

Modify the function to be even/odd, satisfy conservation laws or boundary conditions




$$\tilde{f}(x) = f(x) \cdot b(x) + f_D(x)$$

$$b(x) = \begin{cases} 0, & \text{on the boundary} \\ > 0, & \text{otherwise} \end{cases}$$

#### Training procedure

**PDE-residuals as loss terms**

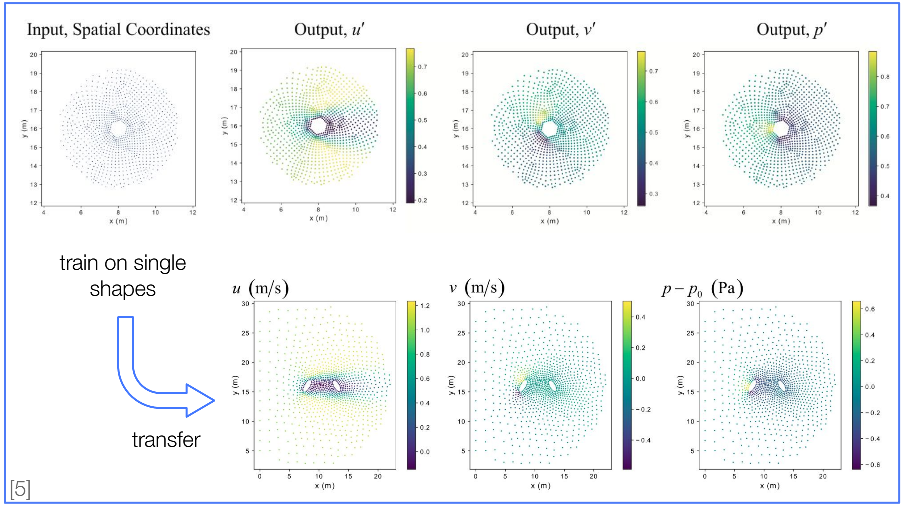
Train a (un-)supervised model to satisfy PDEs

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}}$$

Image sources: [5-4]

# Changing data and loss to include physics

- Task
  - predict steady fluid flow: p,v
  - over various shapes in 2D
  - (geometric shapes and airfoils)
- Method
  - pointcloud encodes the shape geometry
  - residuals of governing PDEs used as
    a. convergence metric
    b. loss term



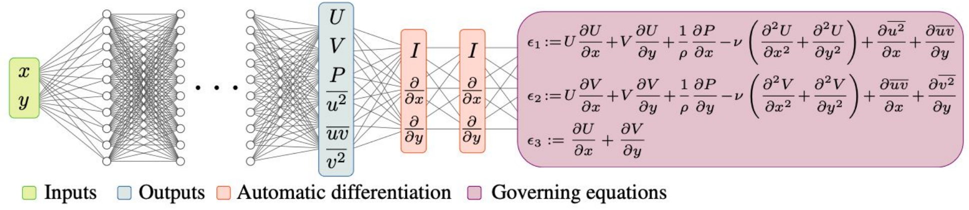



Figure 1: A schematic of PINNs for solving the RANS equations for a general two-dimensional set-up. The left part of the model is a FNN, and the right part is the formulation of the RANS equations using AD. [10]

# Changing model architecture to satisfy function symmetry

- Task
  - solve noisy regression for
    - even/odd functions
    - equations of motion (Hamiltonian mechanics)
  - satisfy symmetries exactly
  - (conservation laws ↔ symmetry)
- Method
  - Introduce hub neurons to satisfy constraints exactly

**Fig. 1.** The new NN architecture with an even hub layer to guarantee the underlying even symmetry.

Last layer output $\hat{x}(t) = \sum_{i=1}^{N} w_i \, h_i(t) + b$

Even/odd requirement (⇒ constraint on the bias term) $\hat{x}(t) = s\hat{x}(-t)$, $s = \pm 1$

⇒ Even/odd decomposition $H_i^{\pm} = h_i(t) \pm h_i(-t)$

$$\hat{x}(t) = \frac{1}{2}\left(\sum_{i=1}^{N} w_i H_i^{+} + 2b\right) + \frac{1}{2}\left(\sum_{i=1}^{N} w_i H_i^{-}\right)$$

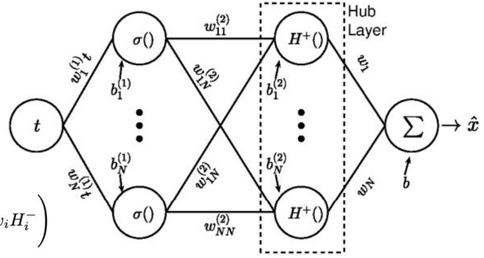
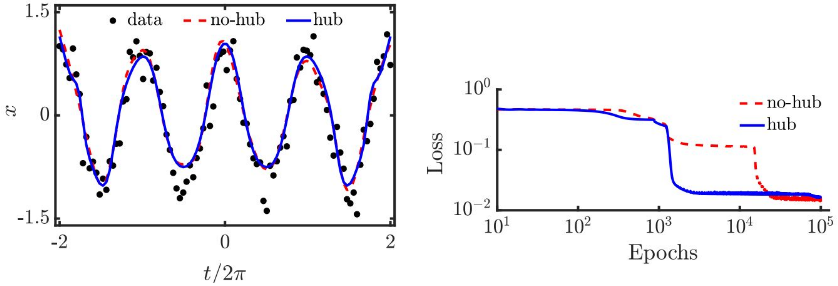



**Fig. 2.** Left: Regression on noisy data from an even function. Right: MSE in training.

$$x(t) = \cos(t) + \epsilon, \quad t \in [-2\pi,\ 2\pi]\,. \quad \epsilon \sim \mathcal{N}(0, \sigma)$$



# Changing model architecture to satisfy energy conservation

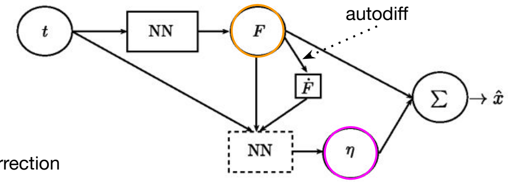
- Task
  - solve noisy regression for
    - even/odd functions
    - equations of motion (Hamiltonian mechanics)
  - satisfy symmetries exactly
  - (conservation laws ↔ symmetry)
- Method
  - Introduce hub neurons to satisfy constraints exactly

**Fig. 4.** Schematic of a NN architecture that guarantees a regression that fits data generated from an energy-conserving process.

Eq. of motion (1d, suppose E=const) $\ddot{x} + \frac{\partial V(x)}{\partial x} = 0$

Prediction $\hat{x}(t) = F(t) + \eta(t)$

regression prediction ($F(t)$), conservation correction ($\eta(t)$)




Total energy $E = \frac{1}{2}\dot{\hat{x}}^2 + V(\hat{x})$ ⇒ second NN solves ODE $\dot{\eta} = -\dot{F} \pm \sqrt{2(E - V(F+\eta))}$

**Fig. 5.** Left: Regression lines and noisy data. Lower: Total energy in time. Right: Phase-space trajectories. The hub network is able to correct the regression to conserve the total energy and to predict closed trajectories in phase-space.

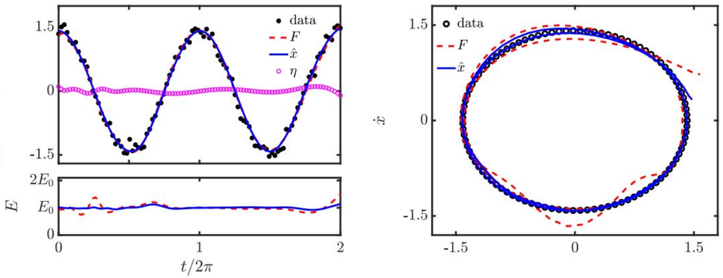
Harmonic oscillator

Potential energy $V(x) = \frac{1}{2}\omega^2 x^2$

Trajectory $x(t) = \sqrt{2}\cos(t)$

Total energy $E_0 = 1$

$(w = 1)$




Source: [6]

# Potential flow with PINNs

## Results: cylinder

- Solved pot. flow
  - Cylinder: soft and hard
  - Airfoil: hard constraints
- Convergence and training speed heavily depends on the number of sampled points
- Overfitting may result in non-physical predictions