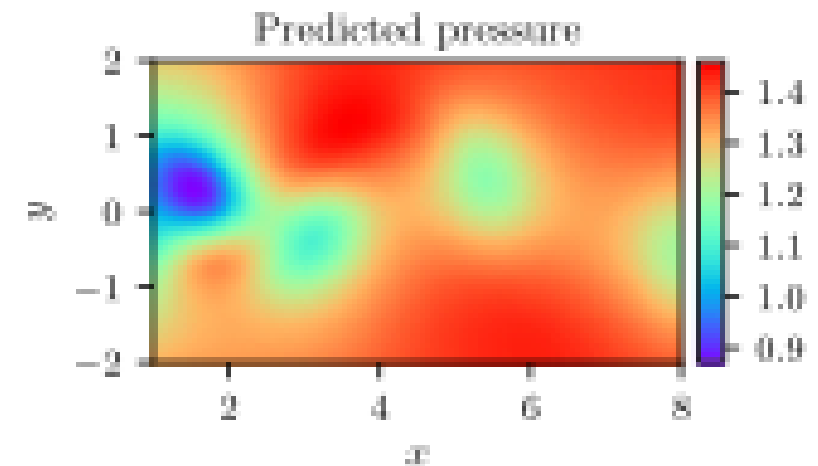
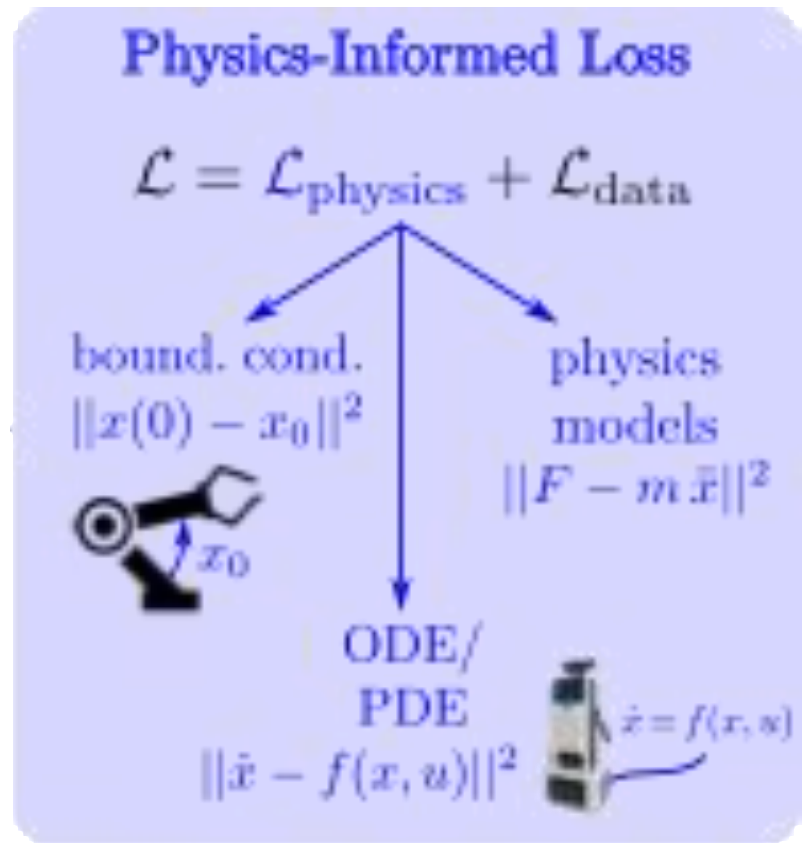


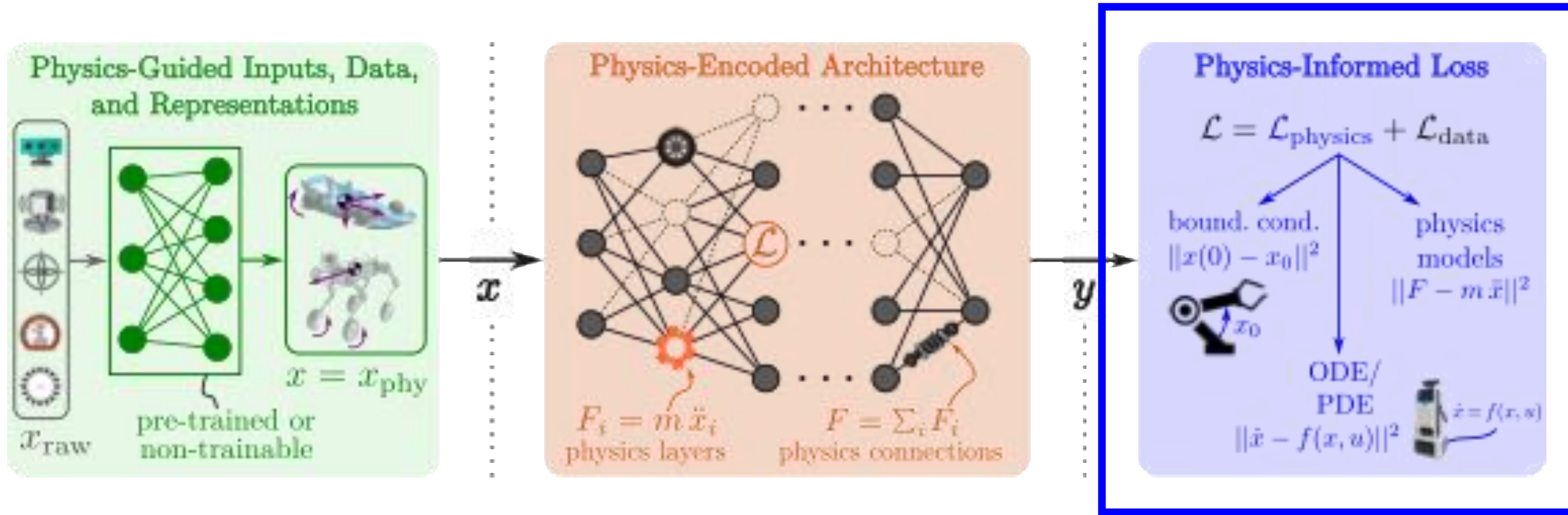
Physics-Informed Loss Functions

Dr. Mattia Piccinini &
Prof. Gastone Pietro Rosati Papini

University of Trento &
Technical University of Munich,
Autonomous Vehicle Systems Lab



Physics-Informed Learning: Positioning



What Is Physics-Informed Learning?

Governing **physical equations** imposed as ***soft constraints*** in the **loss function**

1. Learn the **solutions** of hard-to-solve physical equations
2. Improve accuracy of imperfect physical models through **physics + data loss**

Better generalization in **data-scarce regimes**, compared to purely data-driven NNs



ASME
SETTING THE STANDARD

ASME Journal of Computing and Information Science in Engineering
Online journal at:
<https://asmedigitalcollection.asme.org/computingengineering>

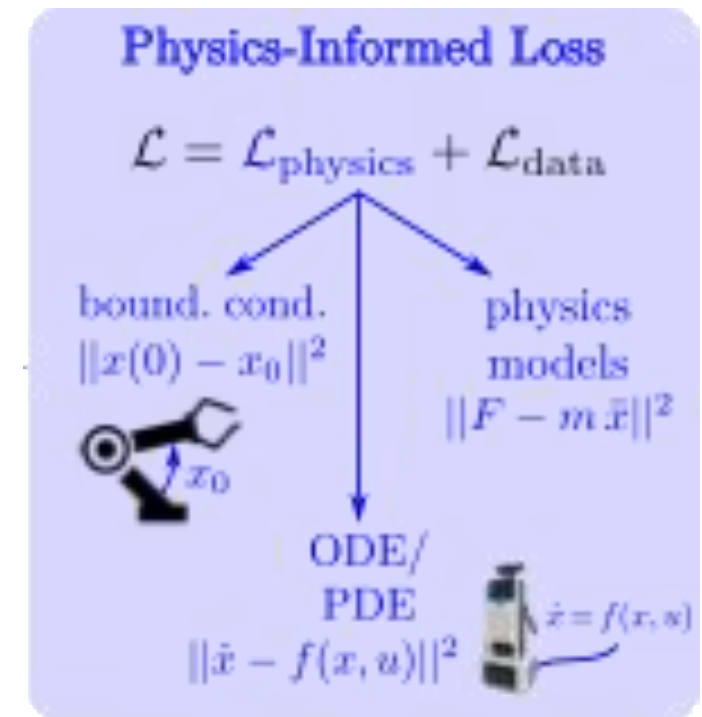


Salah A. Faroughi¹
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666
e-mail: salah.faroughi@txstate.edu

Nikhil M. Pawar
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666

Célio Fernandes
Geo-Intelligence Laboratory,

**Physics-Guided, Physics-Informed and Physics-Encoded
Neural Networks and Operators in
Scientific Computing: Fluid and
Solid Mechanics**



Physics-Informed Learning: Seminal Paper

Neural Networks to *solve* or *discover* **Partial Differential Equations** (PDEs)

Nonlinear **PDE**: $u_t + \mathcal{N}[u; \lambda] = 0$

$u(t, x)$ = unknown solution, $u_t = \frac{\partial u(t, x)}{\partial t}$, λ = known or unknown parameters,
 $\mathcal{N}[u; \lambda]$ = nonlinear operator

Includes physical PDEs for diffusion processes, fluid dynamics, ... and ODEs!

Question 1: given λ , can a NN learn the solution $u(t, x)$?

Question 2: what is the best λ to fit observed data? → „discovery“ of PDEs

Neural Networks to Solve Partial Differential Equations

$$\boxed{u_t + \mathcal{N}[u; \lambda] = 0} \quad \longrightarrow \quad f := u_t + \mathcal{N}[u]$$

Learn the unknown $u(t, x)$ with a **Neural Network (NN)**

Loss function for NN training: $\boxed{MSE = MSE_u + MSE_f}$ (MSE = Mean Square Error)

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \quad \longrightarrow \quad \text{Initial and **boundary conditions** (known)}$$

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \quad \longrightarrow \quad \text{Penalize violations of the governing **PDE**}$$

Commonly known as **Physics-Informed Neural Networks (PINNs)**

Neural Networks to Solve Partial Differential Equations

$$u_t + \mathcal{N}[u; \lambda] = 0$$

Remarks:

- $u(t, x)$ is a **general-purpose NN** (no physics in the architecture!)
- **No theoretical guarantee** to find the exact solution of the PDE
- **In practise**: If the PDE is well-posed and its solution is unique, a sufficiently deep NN will approximate the solution with good accuracy
- Once trained, PINN evaluation is **way faster** than PDE numerical solvers

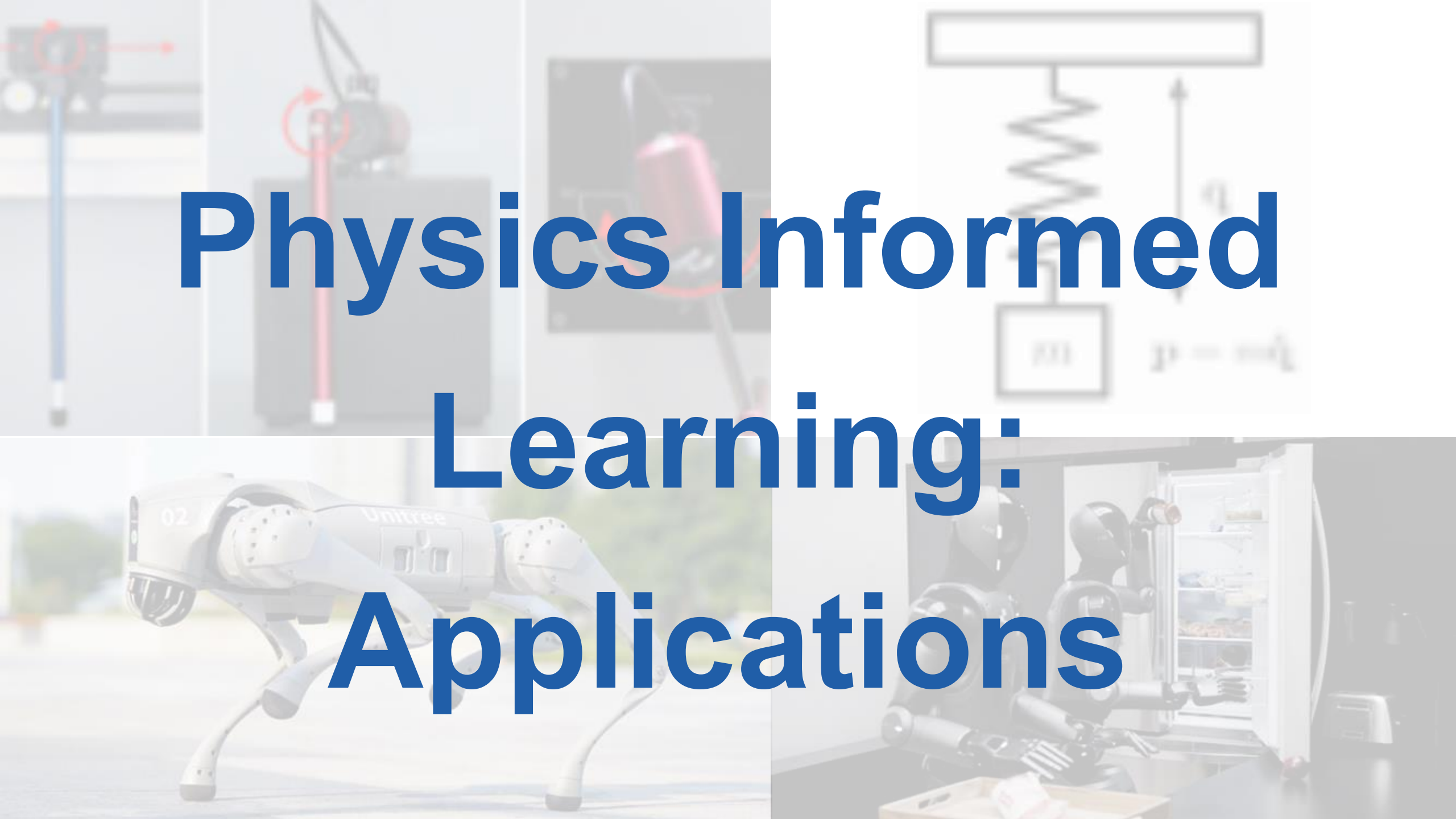
Extended approach: combine **data + physics *regularization* loss** to improve generalization of NNs in data-scarce operation

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{physics}$$

Questions & Discussion



Physics Informed Learning: Applications



Physics-Informed Neural Networks (PINNs) - Examples

Learn solutions of the **Schrödinger equation** (wave propagation)

$$ih_t + 0.5h_{xx} + |h|^2h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2 \operatorname{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$

➡ $f := ih_t + 0.5h_{xx} + |h|^2h$

- **Nonlinear PDE** with **periodic** boundary conditions
- $h(t, x) = [u(t, x) \ v(t, x)]$ is a **complex-valued** solution
- $h(t, x)$ is learned with a 5-layer **NN**

Loss function: $MSE = MSE_0 + MSE_b + MSE_f$

Initial condition: $MSE_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2$

Schrödinger eq.: $MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$

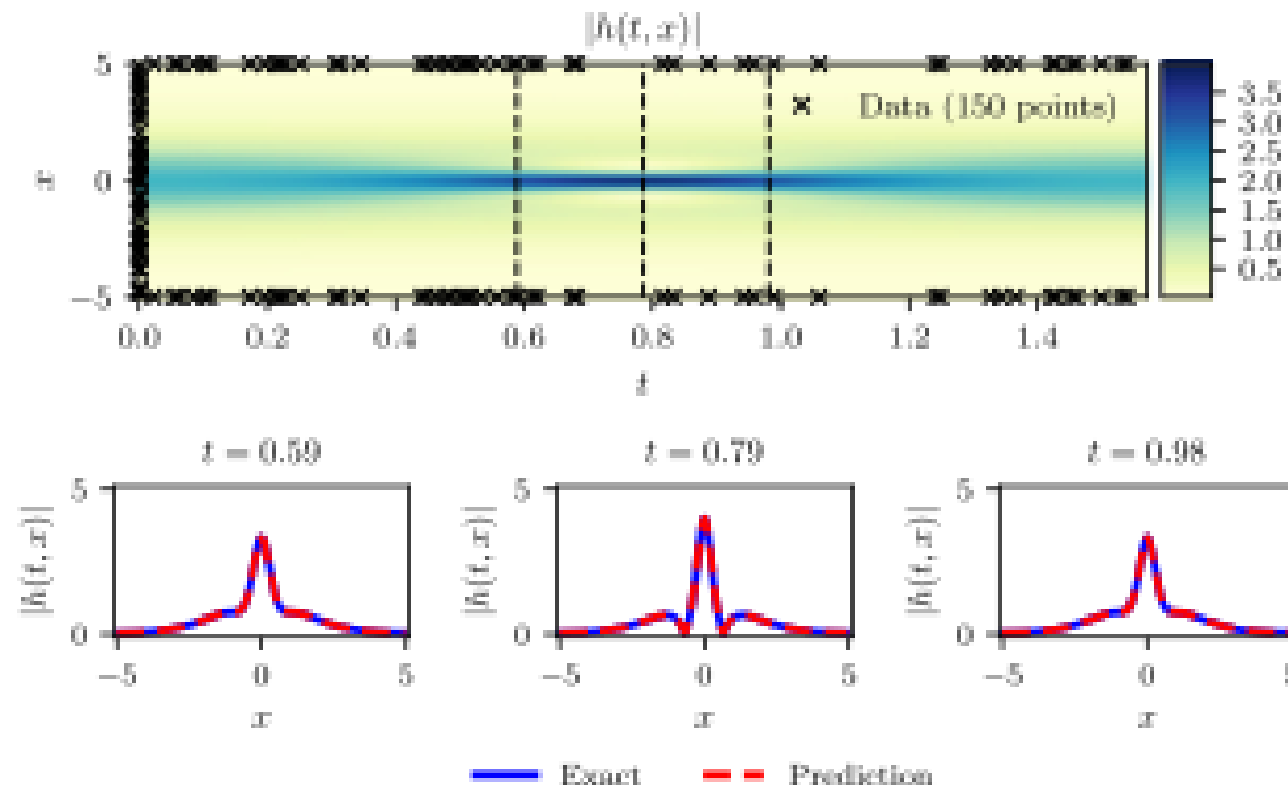
Periodic B.C.: $MSE_b = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2 \right)$

Physics-Informed Neural Networks (PINNs) - Examples

Learn solutions of the **Schrödinger equation** (wave propagation)

Note: **no ground-truth values** of $h(t, x)$ are required to train the NN!

- Only $h(0, x)$ (initial conditions) + periodic boundaries



Physics-Informed Neural Networks (PINNs) - Examples

Predict the **deformation** of a pneumatic **soft robot finger**

Continuum mechanics PDE (Navier-Cauchy model):

$$f := (\lambda + \mu) \frac{\partial}{\partial x} (u_x + u_z) + \mu \left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_z}{\partial z^2} \right) + f_x$$

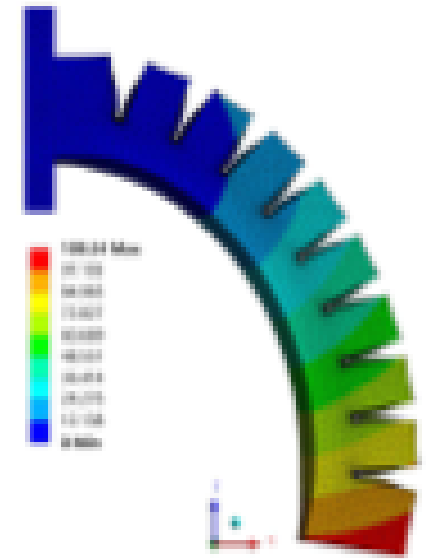
$$g := (\lambda + \mu) \frac{\partial}{\partial z} (u_x + u_z) + \mu \left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_z}{\partial z^2} \right) + f_z$$

u_x, u_z = deformations along x and z, f_x, f_z = body forces due to gravity

Loss function: $\mathcal{L} = \mathcal{L}_{u_x, u_z} + \mathcal{L}_{f, g}$

Initial condition: $\mathcal{L}_{u_x, u_z} = \frac{1}{N} \sum_{i=1}^N \left(|\hat{u}_x^i - u_x^i|^2 + |\hat{u}_z^i - u_z^i|^2 \right) |_{p^i}$

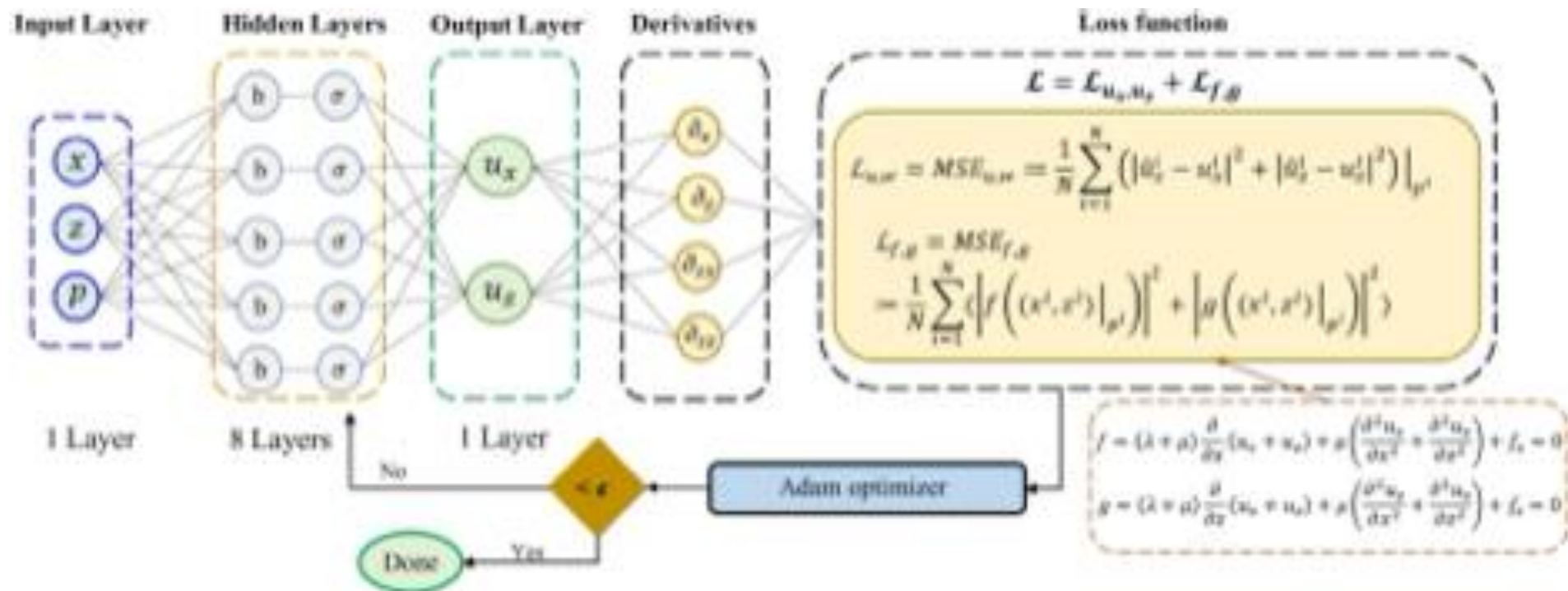
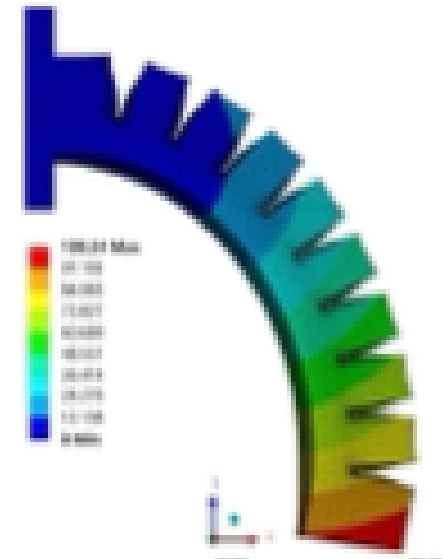
Navier-Cauchy eq.: $\mathcal{L}_{f, g} = \frac{1}{N} \sum_{i=1}^N \left(|f(x^i, z^i)|_{p^i}|^2 + |g(x^i, z^i)|_{p^i}|^2 \right)$



Physics-Informed Neural Networks (PINNs) - Examples

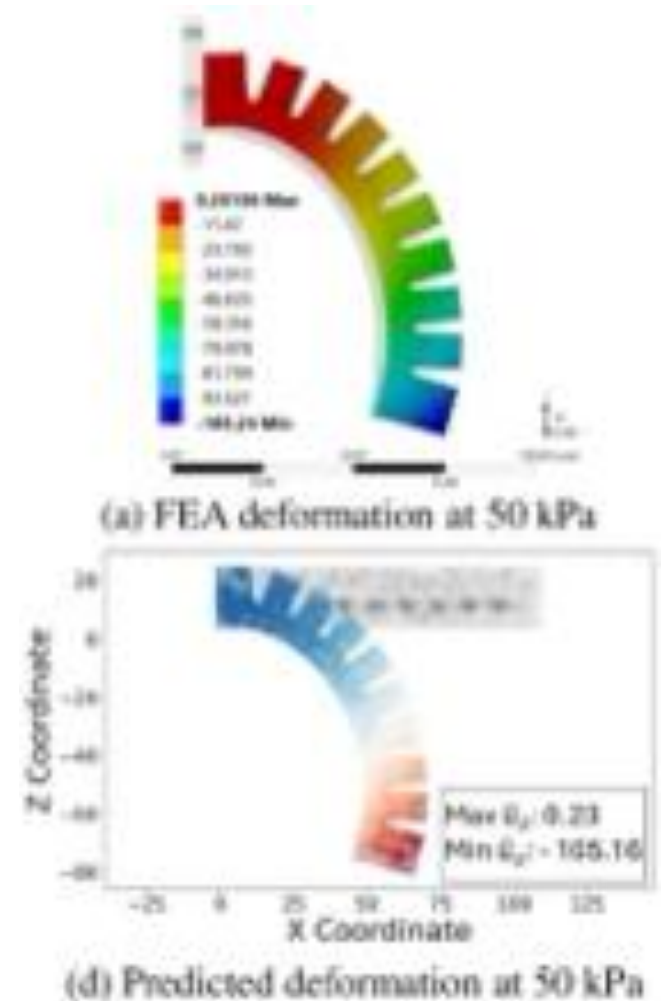
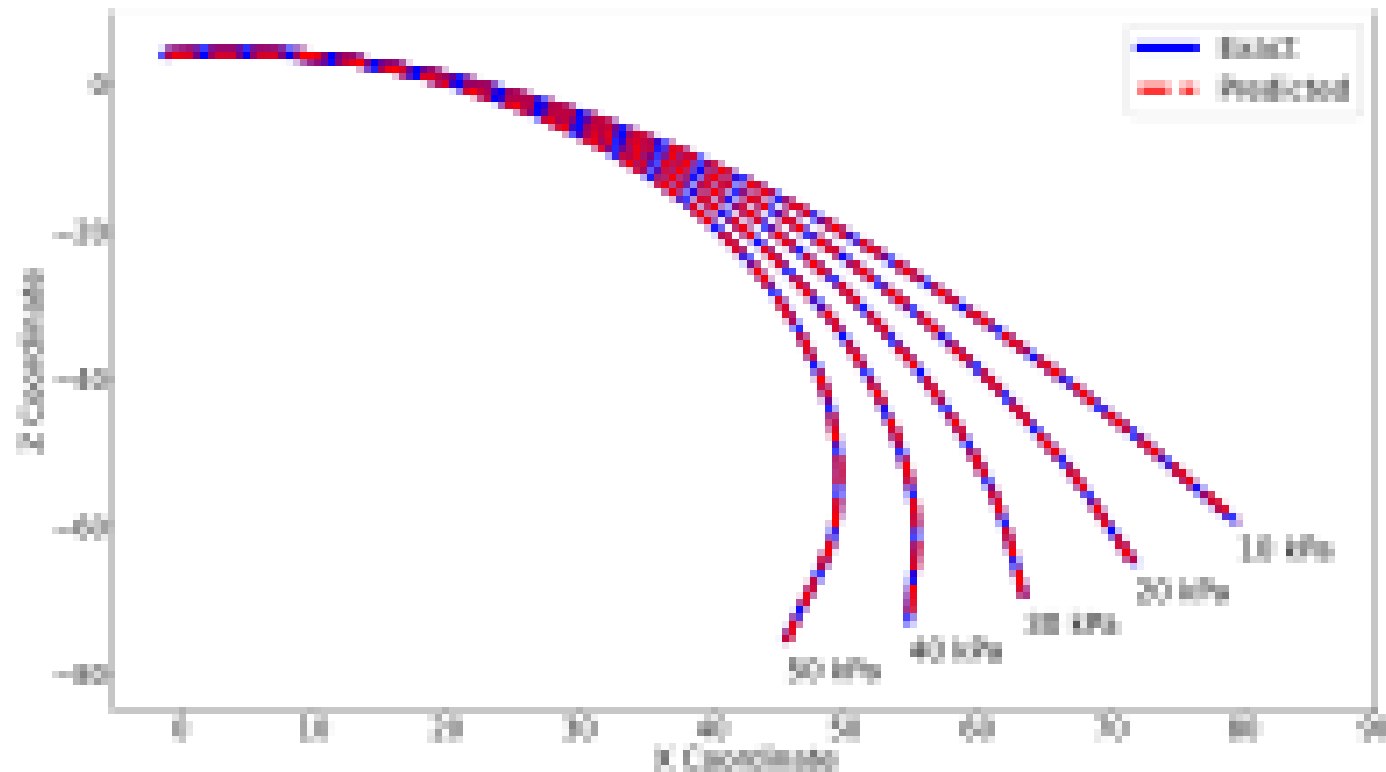
Predict the **deformation** of a pneumatic **soft robot finger**

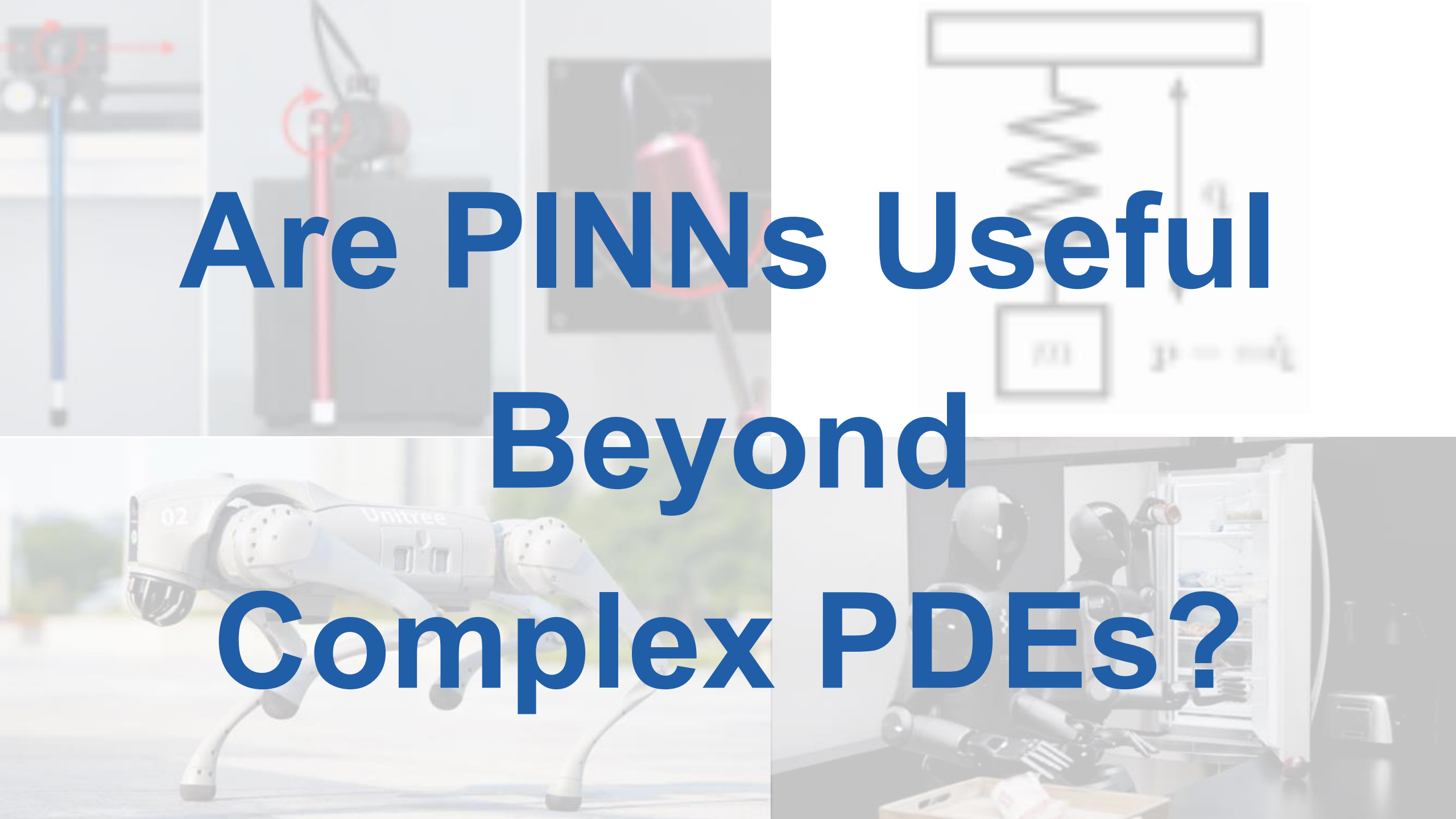
- **PINN** to learn the deformations u_x , u_z from x , z , applied air pressure p
- Benchmarking against **numerical PDE solution** (FEA with ANSYS)



Physics-Informed Neural Networks (PINNs) - Examples

The trained PINN enables **real-time** deformation prediction (way faster than FEA solver)



The background is a collage of four images. Top-left: A robotic arm with a red vertical shaft and a red circular arrow indicating rotation. Top-right: A physics diagram of a mass-spring system with a mass 'm' and displacement 'x', with the equation $x = x_0 \cos(\omega t)$ shown. Bottom-left: A white quadruped robot, labeled 'Unitree' and '02', in a running pose. Bottom-right: A robotic arm with a gripper reaching into an open refrigerator.

Are PINNs Useful Beyond Complex PDEs?

Physics-Informed Neural Networks (PINNs) - Examples



Learn the **dynamics** of a quadrotor

Quadrotor **dynamics**: Ordinary Differential Equation (**ODE**)

$$\begin{aligned}\dot{x}(t) &= v_x(t), \quad \dot{y}(t) = v_y(t), \quad \dot{z}(t) = v_z(t) \\ \dot{v}_x(t) &= \left(c_{\phi(t)} s_{\theta(t)} c_{\psi(t)} + s_{\phi(t)} s_{\psi(t)} \right) \frac{u_p(t)}{m} \\ \dot{v}_y(t) &= \left(c_{\phi(t)} s_{\theta(t)} s_{\psi(t)} - s_{\phi(t)} c_{\psi(t)} \right) \frac{u_p(t)}{m} \\ \dot{v}_z(t) &= -g + c_{\phi(t)} c_{\theta(t)} \frac{u_p(t)}{m} \\ \dot{\phi}(t) &= \omega_1(t), \quad \dot{\theta}(t) = \omega_2(t), \quad \dot{\psi}(t) = \omega_3(t) \\ \dot{\omega}_1(t) &= \frac{J_2 - J_3}{J_1} \omega_2(t) \omega_3(t) + \frac{J_p}{J_1} \omega_p(t) \omega_2(t) + \frac{1}{J_1} \tau_1(t) \\ \dot{\omega}_2(t) &= \frac{J_3 - J_1}{J_2} \omega_1(t) \omega_3(t) - \frac{J_p}{J_2} \omega_p(t) \omega_1(t) + \frac{1}{J_2} \tau_2(t) \\ \dot{\omega}_3(t) &= \frac{J_1 - J_2}{J_3} \omega_1(t) \omega_2(t) + \frac{1}{J_3} \tau_3(t).\end{aligned}$$

$$\partial_t \hat{y} + N[\hat{y}; \lambda] = 0$$



$$l := \partial \hat{y} + N[\hat{y}]$$



Learn $\hat{y}(t)$ with a NN

Physics-Informed Neural Networks (PINNs) - Examples



Learn the **dynamics** of a quadrotor

$$\partial_t \hat{y} + N[\hat{y}; \lambda] = 0 \quad \longrightarrow \quad l := \partial \hat{y} + N[\hat{y}]$$

Loss function: $MSE = \boxed{MSE_{\hat{y}}} + \gamma MSE_l$

Data loss:

$$\boxed{MSE_{\hat{y}} = \frac{1}{N_{\hat{y}}} \sum_{i=1}^{N_{\hat{y}}} \frac{1}{N_t} \sum_{j=1}^{N_t} |\hat{y}_i(t^j) - \hat{y}_{i,ref}^j|^2}$$

N_t = n. data samples

$N_{\hat{y}}$ = n. outputs

Supervised learning with collected training data $\hat{y}_{ref}$

Physics-Informed Neural Networks (PINNs) - Examples



Learn the **dynamics** of a quadrotor

$$\partial_t \hat{y} + N[\hat{y}; \lambda] = 0 \quad \longrightarrow \quad l := \partial \hat{y} + N[\hat{y}]$$

Loss function: $MSE = \boxed{MSE_{\hat{y}}} + \boxed{\gamma MSE_l}$

Data loss:

$$MSE_{\hat{y}} = \frac{1}{N_{\hat{y}}} \sum_{i=1}^{N_{\hat{y}}} \frac{1}{N_t} \sum_{j=1}^{N_t} |\hat{y}_i(t^j) - \hat{y}_{i,ref}^j|^2$$

N_t = n. data samples

$N_{\hat{y}}$ = n. outputs

Physics loss:

$$MSE_l = \frac{1}{N_{\hat{y}}} \sum_{i=1}^{N_{\hat{y}}} \frac{1}{N_l} \sum_{j=1}^{N_l} |l(\hat{y}_i(t^k))|^2$$

Penalize violations of physics equations

Physics-Informed Neural Networks (PINNs) - Examples



Learn the **dynamics** of a quadrotor

$$\partial_t \hat{y} + N[\hat{y}; \lambda] = 0 \quad \longrightarrow \quad l := \partial \hat{y} + N[\hat{y}]$$

Loss function: $MSE = MSE_{\hat{y}} + \gamma MSE_l$ γ weighs our **physics confidence**

Data loss:

$$MSE_{\hat{y}} = \frac{1}{N_{\hat{y}}} \sum_{i=1}^{N_{\hat{y}}} \frac{1}{N_t} \sum_{j=1}^{N_t} |\hat{y}_i(t^j) - \hat{y}_{i,ref}^j|^2$$

N_t = n. data samples

$N_{\hat{y}}$ = n. outputs

Physics loss:

$$MSE_l = \frac{1}{N_{\hat{y}}} \sum_{i=1}^{N_{\hat{y}}} \frac{1}{N_l} \sum_{j=1}^{N_l} |l(\hat{y}_i(t^k))|^2$$

N_l = n. collocation points t_k

Physics-Informed Neural Networks (PINNs) - Examples

Learn the **dynamics** of a quadrotor

Results for parameter estimation:

- **Outperforming** an **Extended Kalman Filter** (EKF)



Parameter	Value	Units	Known for EKF	Known for PINNs
m	0.65	kg	YES	NO
d	0.165	m	YES	NO
J_x	0.03	$\text{kg} \cdot \text{m}^2$	NO	NO
J_y	0.025	$\text{kg} \cdot \text{m}^2$	NO	NO
J_z	0.045	$\text{kg} \cdot \text{m}^2$	NO	NO
b	3.50	N/rad/s	NO	NO
k	0.06	$\text{N} \cdot \text{m/rad/s}$	NO	NO

Table 3. Performance indices for the x model errors expressed in meters.

	MAE	MSE	ISE	IAE	ITAE
EKF	0.0514	0.0039	5.8711	77.17	599.94
PINNs	0.0367	0.0022	3.3427	55.07	432.52

Physics-Informed Neural Networks (PINNs) - Examples

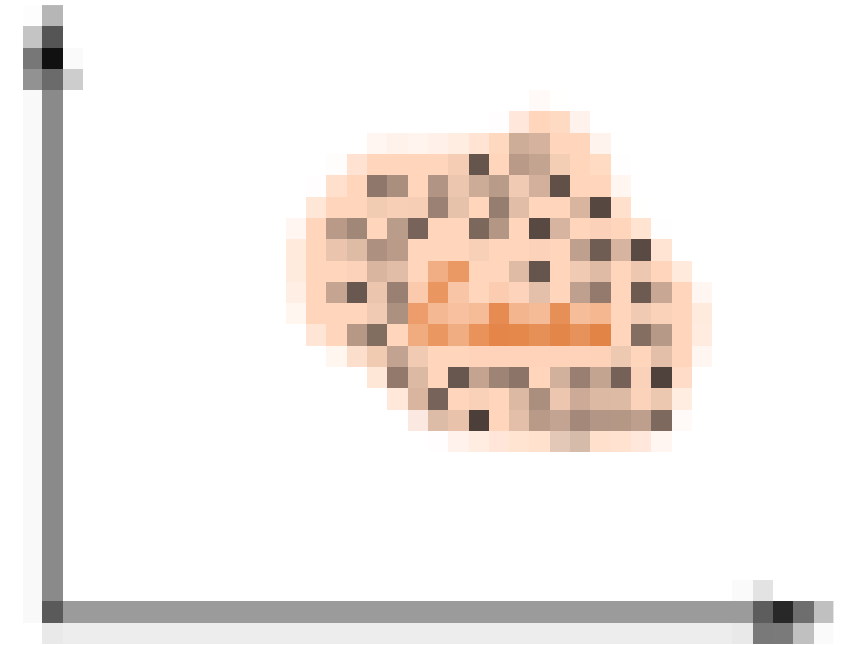
Common approach in **real-world physical systems & robotics**

Combine **data** + physics *regularization* loss

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{physics}$$

Highly effective when:

- Real **data** is concentrated in **small domain** regions → use $\mathcal{L}_{data}$ there
- We have real **data** measurements + an (imperfect) **physics model**



Physics-Informed Neural Networks (PINNs) - Examples

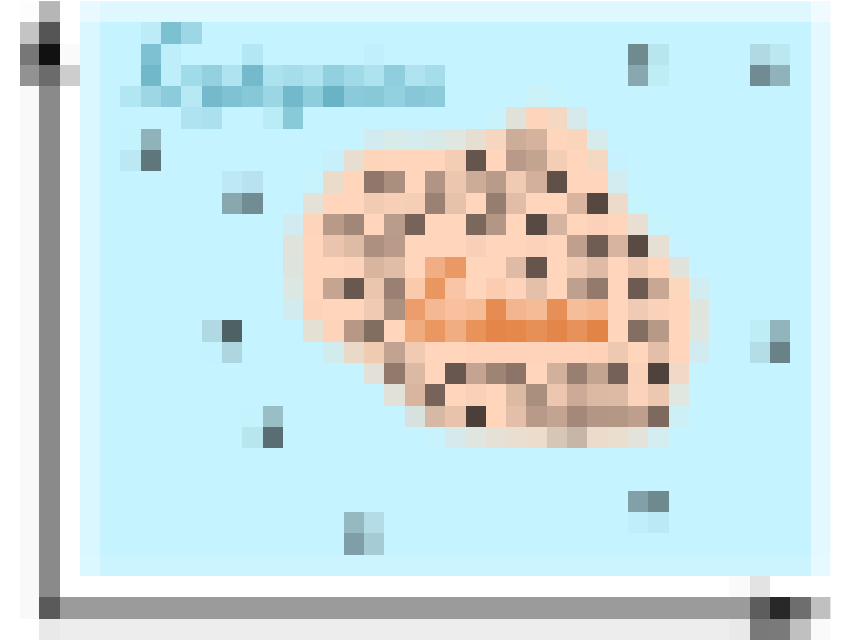
Common approach in **real-world physical systems & robotics**

Combine **data** + physics *regularization* loss

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{physics}$$

Highly effective when:

- Real **data** is concentrated in **small domain** regions → use $\mathcal{L}_{data}$ there
- We have real **data** measurements + an (imperfect) **physics model**
- Use $\mathcal{L}_{physics}$ to **generalize** well in data-scarce regions
- Note: $\mathcal{L}_{physics}$ does **not** require ground truth data



Can PINN Loss Function Contain Other Terms?



Physics-Informed Neural Networks (PINNs) - Examples

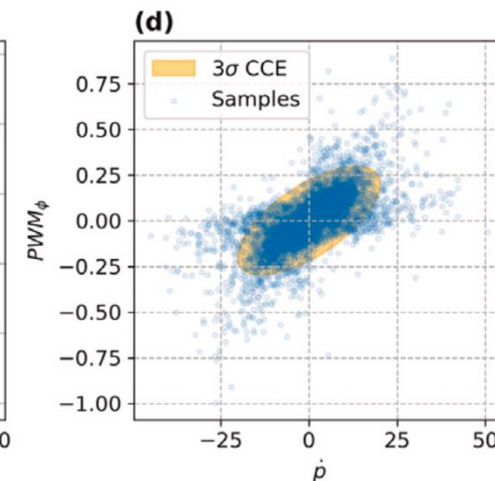
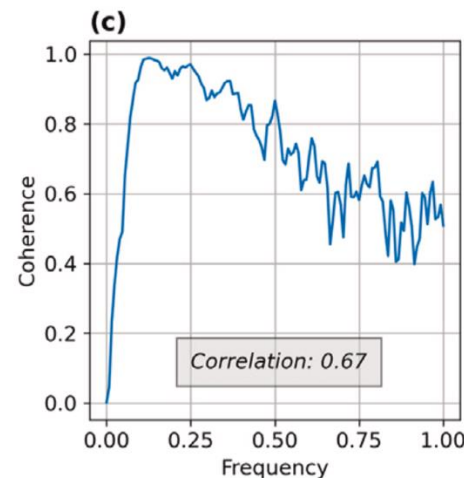
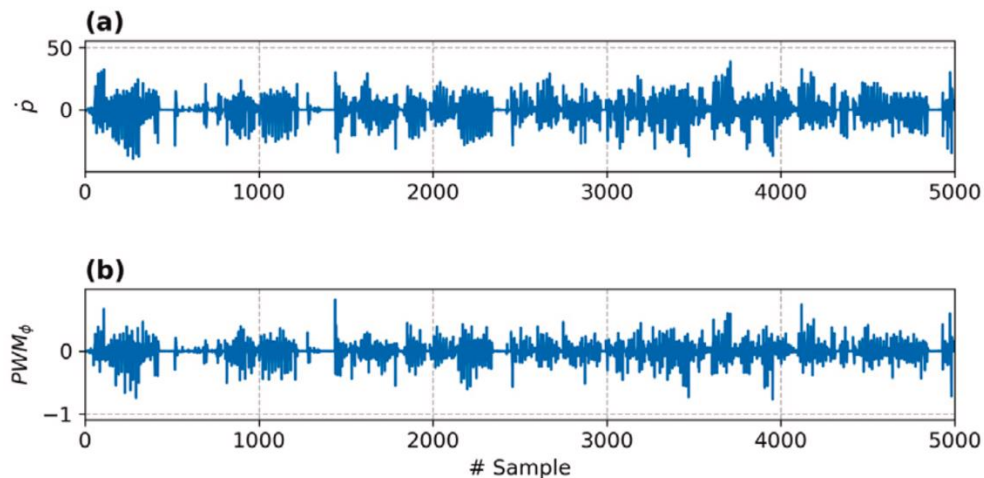
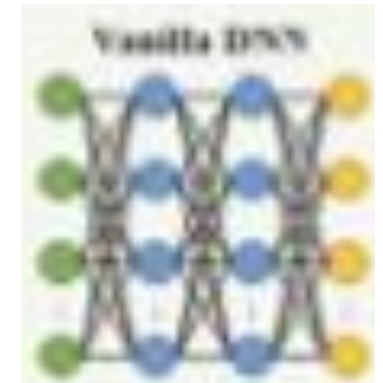
Learn the **inverse** dynamics of a quadrotor

$\hat{Y} = \text{PINN}(X, \theta)$, X = quadrotor states, $\hat{Y}$ = control inputs (PWM motor signals)

Loss function idea: Embed **conservation of momentum**

→ Exploit **correlation** of inputs and accelerations:

- **Roll** acceleration $\dot{p}$ highly correlated to PWM motor signal to generate it



Physics-Informed Neural Networks (PINNs) - Examples

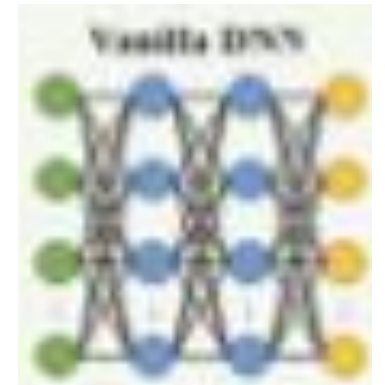
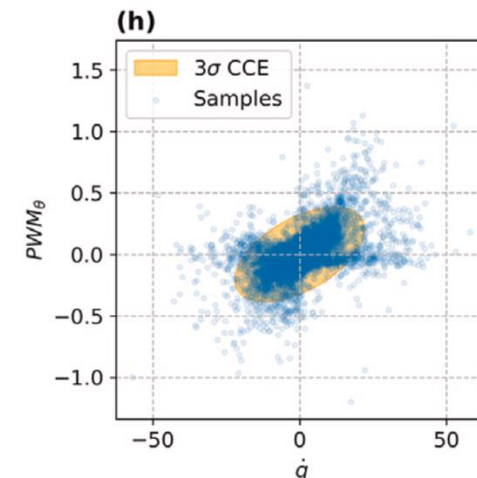
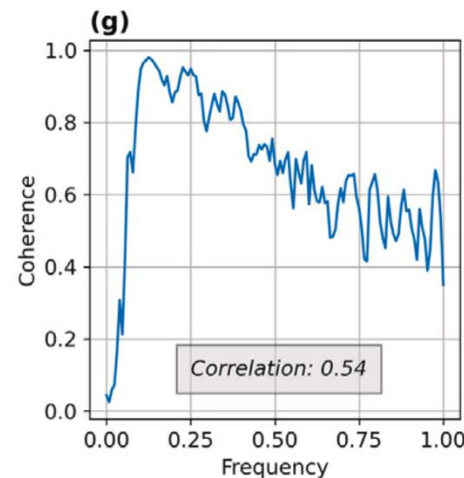
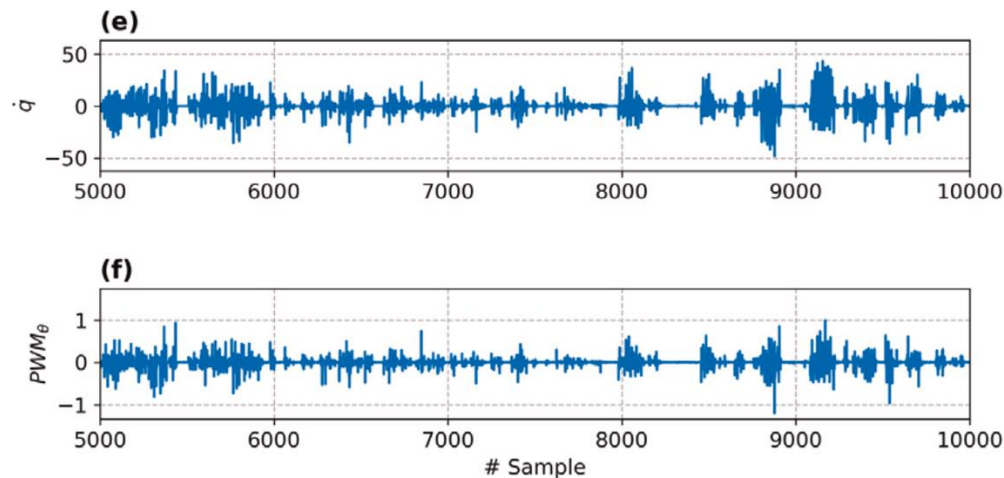
Learn the **inverse dynamics** of a quadrotor

$\hat{Y} = \text{PINN}(X, \theta)$, X = quadrotor states, $\hat{Y}$ = control inputs (PWM motor signals)

Loss function idea: Embed **conservation of momentum**

→ Exploit **correlation** of inputs and accelerations:

- **Pitch** acceleration $\dot{q}$ highly correlated to PWM motor signal to generate it



Physics-Informed Neural Networks (PINNs) - Examples

Learn the **inverse dynamics** of a quadrotor

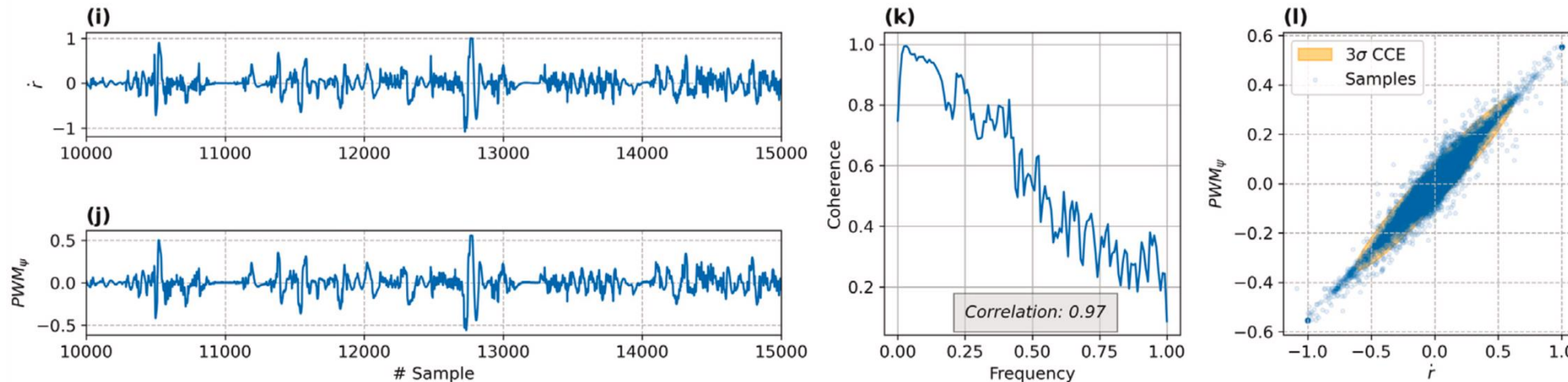
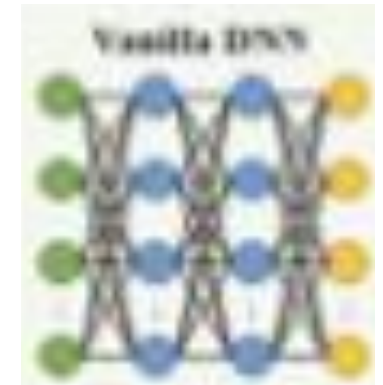


$\hat{Y} = \text{PINN}(X, \theta)$, X = quadrotor states, $\hat{Y}$ = control inputs (PWM motor signals)

Loss function idea: Embed **conservation of momentum**

→ Exploit **correlation** of inputs and accelerations:

- **Yaw** acceleration $\dot{r}$ highly correlated to PWM motor signal to generate it



Physics-Informed Neural Networks (PINNs) - Examples



Learn the **inverse dynamics** of a quadrotor

Loss function idea: Embed **conservation of momentum**

→ **Local Monotonicity (LM)** loss terms:

- Accelerations and PWM signals should *exhibit the same pattern*

$$\mathcal{L} = \boxed{\mathcal{L}_{\text{MSE}}} + \boxed{\sum_i \lambda_{\text{LM}}^i \mathcal{L}_{\text{LM}}^i}, \quad i = \{\phi, \theta, \psi\}$$

$\boxed{\sum_i \lambda_{\text{LM}}^i \mathcal{L}_{\text{LM}}^i}$ = local monotonicity loss for the roll, pitch, yaw rotations (**soft constraints**)

$\boxed{\mathcal{L}_{\text{MSE}}}$ = standard data-driven loss → minimize deviations from ground-truth observations

Physics-Informed Neural Networks (PINNs) - Examples



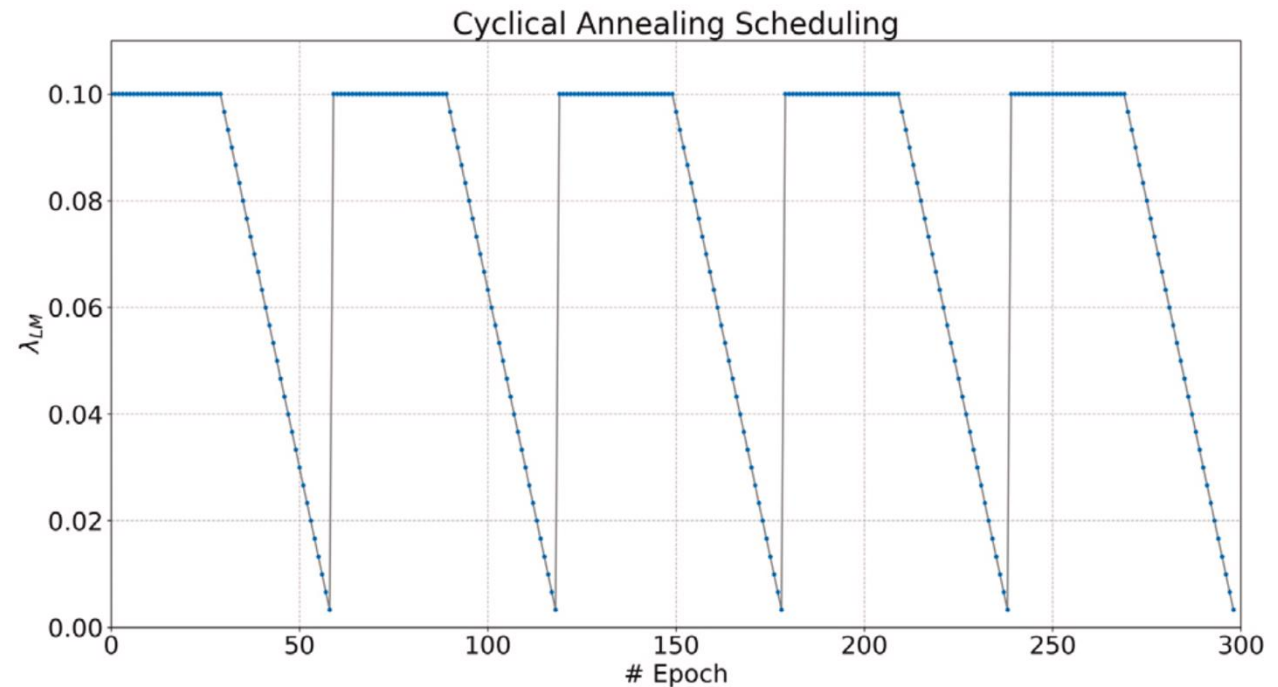
Learn the **inverse dynamics** of a quadrotor

Issue: strong **aerodynamic effects** can **contradict** the local **monotonicity loss**

→ Accelerations and motor commands may no longer be highly correlated!

→ **Cyclical annealing scheduler** for the LM terms to let the NN **learn real aerodyn.** effects

$$\mathcal{L} = \boxed{\mathcal{L}_{\text{MSE}}} + \boxed{\sum_i \lambda_{\text{LM}}^i \mathcal{L}_{\text{LM}}^i},$$



Questions & Discussion

