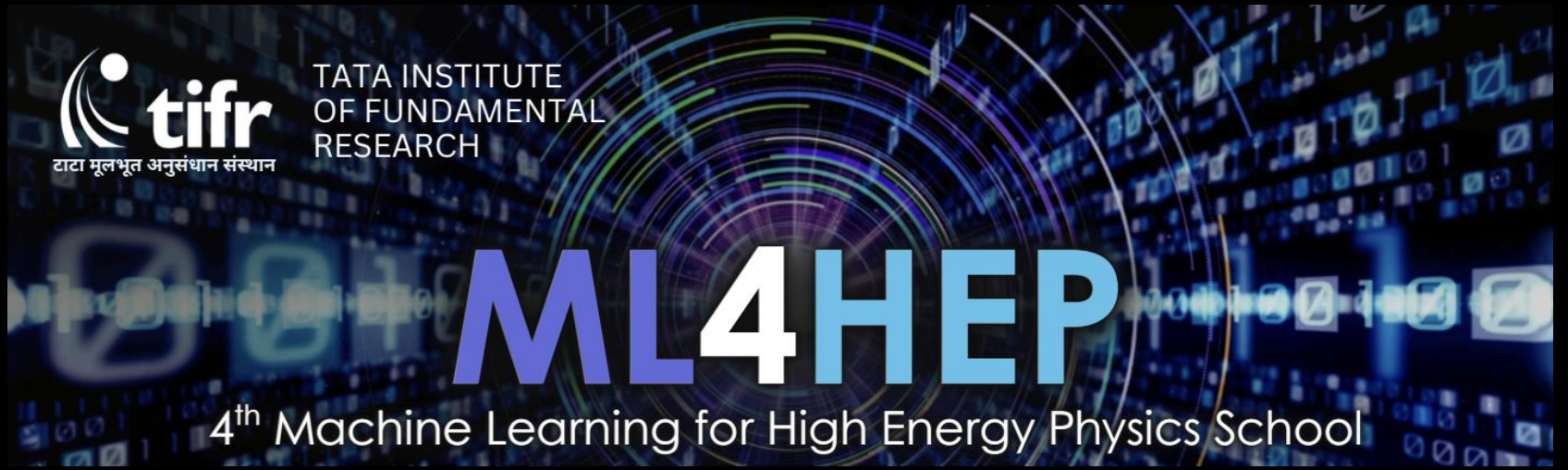


Testing GlueNN

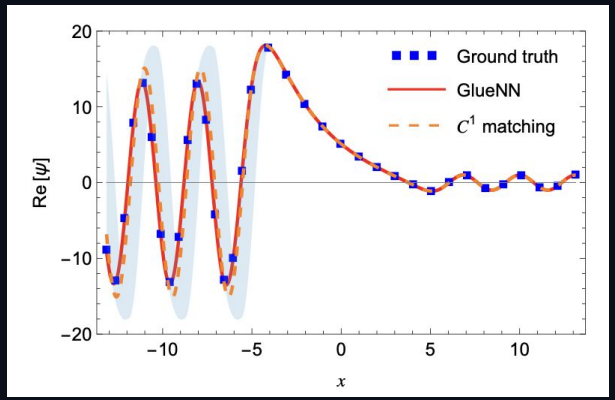
Partha Konar

Physical Research Laboratory, Ahmedabad

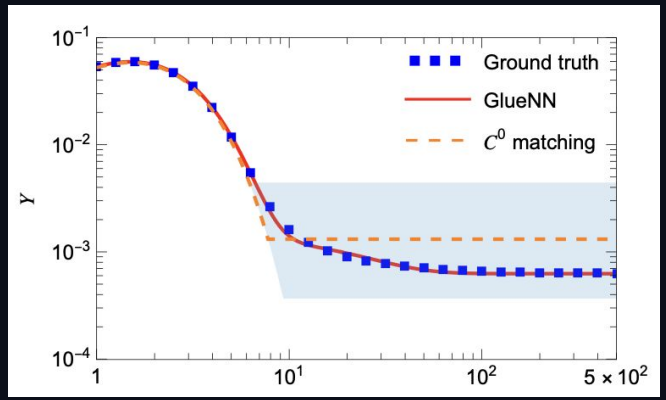
Case Study: The Interpretability Frontier - on - Physics Informed Neural Network

GlueNN: Gluing Patchwise Analytic Solutions

Bridging global neural approximations with local analytical physics to create "Glass Box" architectures



1-D quantum tunneling



Chemical reaction in hot thermal bath

Motivation

A case in which the equation admits two characteristic asymptotic forms defined in separate patches, $\{p_1 : x \ll a\}$ and $\{p_2 : x \gg a\}$.
In these regions, the dynamics is effectively described by: $D_1[f_1(x,c_1^{(i)})] = 0$ and $D_2[f_2(x,c_2^{(i)})] = 0$,
where $f_1(x,c_1^{(i)})$ and $f_2(x,c_2^{(i)})]$ represent the general analytic solutions in their respective patches,
and $c_1^{(i)}$ and $c_2^{(i)}$ denote the sets of undetermined coefficients to be fixed by boundary or matching conditions.

arXiv: 2601.05889 [cs.LG]

Case Study: The Interpretability Frontier - on - Physics Informed Neural Network

GlueNN: Gluing Patchwise Analytic Solutions

Bridging global neural approximations with local analytical physics to create "Glass Box" architectures

How GlueNN Works

1. Local Deconstruction: Leverages known local analytic solutions or asymptotic behaviors at specific physical limits.

2. Promoted Coefficients: Instead of curve-fitting the solution directly, it promotes integration constants to scale-dependent, learnable coefficient functions managed by the NN.

3. The Gluing Mechanism: Constrained by the governing differential equation to smoothly interpolate across different physical regimes.

The "Glass Box" Solution

Natively Physics-Aware: The neural network only handles transition regions where analytical approximations shift, rather than treating the entire domain as a black box.

Radical Interpretability: Because the model outputs coefficient functions of analytic equations, physicists can directly extract meaningful parameters that are completely hidden in standard numerical solvers.

Attribute	Traditional PINNs	GlueNN Framework
Target of Learning	Global solution field directly: u(x)	**Scale-dependent coefficients of local analytic expansions**
Interpretability	Low (Black-box numerical field output)	**High (Readable analytic components)**
Boundary Matching	Requires ad hoc, manually tuned boundary losses	**Naturally glues regimes via differential constraints**
Parameter Extraction	Difficult; requires post-processing of numerical data	**Direct; physical constants embedded in coefficients**

Physics-Informed Neural Networks (PINNs)

Bridging Deep Learning and Physical Laws via Differentiable Optimization

1. BASIC CONCEPT

- Rather than learning solely from pure data, the model is regularized to respect the underlying governing laws of the system.
- Neural networks that incorporate physical laws described by differential equations into their loss functions to guide the learning process toward solutions that are more consistent with the underlying physics

2. WHY & WHERE IT IS USEFUL

- Use prior physics knowledge.
- Make more accurate predictions outside of the training data set.
- Are more effective with limited or noisy training data.

3. CORE MATHEMATICAL PRINCIPLE

PINNs minimize a multi-term objective loss function using Automatic Differentiation (AD), penalizing deviations from physical laws:

$$Loss = L_{data} + \lambda \cdot L_{physics}$$

The physical loss component $\mathbf{L_{physics}}$ penalizes deviations from the governing PDE/ODE residue, ensuring strict physical compliance across all domain collocation points.

THE CORE MATHEMATICAL PRINCIPLE

Physics-Informed Neural Networks (PINNs)

Regularizing neural networks using automatic differentiation to respect governing physical laws

1. Governing Principle

Instead of relying purely on observational data, PINNs embed physical laws directly into the loss function during training. This forces the model to respect scientific principles even in regions devoid of empirical measurements.

- **Collocation Points:** Evaluated across a grid of domain coordinates.
- **Inductive Bias:** Restricts search space to physically plausible solutions.

2. Concrete Example: Damped Harmonic Oscillator

Displacement **x(t)** of a damped Harmonic Oscillator system over time.

- **Black Box Model:** Might predict physically impossible amplification or sustained perpetual motion.
- **PINN Approach:** Minimizes the oscillator equation residual, ensuring displacement decays over time as energy dissipates.

Mathematical Formulation

PINNs optimize parameters via a multi-objective loss function utilizing Automatic Differentiation (AD) to evaluate exact gradients without numerical approximation:

$$Loss = L_{data} + \lambda \cdot L_{physics}$$

Where the components represent:

L_{data} : Standard Mean Squared Error matching known boundary conditions and sparse experimental measurements.

$L_{physics}$: The mean squared residual of the differential equation **$f(u, \partial u/\partial x, \ldots) = 0$** evaluated over arbitrary collocation points.

λ : A regularization hyperparameter balancing optimization priority between empirical data and physical laws.

IMPLEMENTATION STUDY

Solving the Damped Harmonic Oscillator with PINNs

PyTorch workflow mapping physical constraints directly onto network gradients via automatic differentiation

```python
# 1. Define Model and Physics Loss
class PINN(nn.Module):
  def __init__(self):
    super().__init__()
    self.net = nn.Sequential(nn.Linear(1,32), nn.Tanh(),
                             nn.Linear(32,32), nn.Tanh(),
                             nn.Linear(32,1))
  def forward(self, t): return self.net(t)


def physics_loss(model, t, mu=0.5, k=4.0):
  t.requires_grad_(True)
  x = model(t)
  dx = grad(x, t, grad_outputs=ones_like(x), create_graph=True)[0]
  d2x = grad(dx, t, grad_outputs=ones_like(dx), create_graph=True)[0]
  return mean((d2x + mu*dx + k*x)**2)


# 2. Optimization Loop
opt = Adam(model.parameters(), lr=1e-3)
for epoch in range(2001):
  loss = data_loss(model) + 1e-4 * physics_loss(model, t_colloc)
  opt.zero_grad(); loss.backward(); opt.step()
```

Step-by-Step Mechanism

1. Exact Gradient Computation

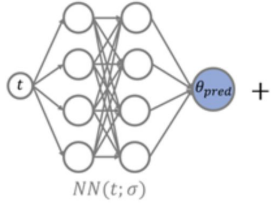
Instead of numerical differences, automatic differentiation computes exact continuous derivatives *dx/dt* and d^2x/dt^2 directly at collocation points.

2. Embedding the Physical Residual

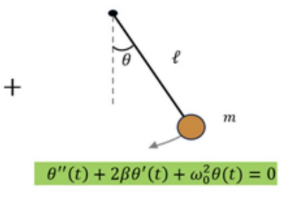
The ODE term (d2x + mu*dx + k*x) is penalized. The network parameters adjust until the output respects the dampening laws of motion.

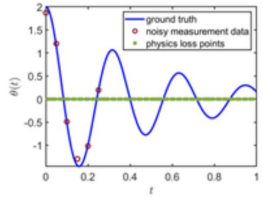
3. Regularized Multi-Task Optimization

Loss combines boundary conditions (forces *x(0)=1, dx/dt(0)=0*) and the continuous physical regularizer *L_physics*.



+





$$\min_{\sigma} \frac{1}{N}\sum_{i=1}^{N}\left|\theta_{pred}(t_i;\sigma) - \theta_{meas}(t_i)\right|^2$$

$$+\frac{1}{M}\sum_{j=1}^{M}\left|\frac{\partial^2}{\partial t^2}\theta_{pred}(t_j;\sigma) + 2\beta\frac{\partial}{\partial t}\theta_{pred}(t_j;\sigma) + \omega_0^2\theta_{pred}(t_j;\sigma)\right|^2$$

IMPLEMENTATION STUDY

Solving the Damped Harmonic Oscillator with PINNs

PyTorch workflow mapping physical constraints directly onto network gradients via automatic differentiation

```python
# 1. Define Model and Physics Loss
class PINN(nn.Module):
  def __init__(self):
    super().__init__()
    self.net = nn.Sequential(nn.Linear(1,32), nn.Tanh(),
                             nn.Linear(32,32), nn.Tanh(),
                             nn.Linear(32,1))
  def forward(self, t): return self.net(t)


def physics_loss(model, t, mu=0.5, k=4.0):
  t.requires_grad_(True)
  x = model(t)
  dx = grad(x, t, grad_outputs=ones_like(x), create_graph=True)[0]
  d2x = grad(dx, t, grad_outputs=ones_like(dx), create_graph=True)[0]
  return mean((d2x + mu*dx + k*x)**2)


# 2. Optimization Loop
opt = Adam(model.parameters(), lr=1e-3)
for epoch in range(2001):
  loss = data_loss(model) + 1e-4 * physics_loss(model, t_colloc)
  opt.zero_grad(); loss.backward(); opt.step()
```

Step-by-Step Mechanism

1. Exact Gradient Computation

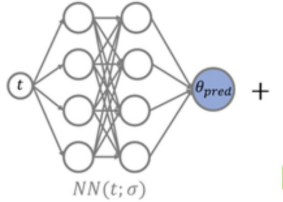
Instead of numerical differences, automatic differentiation computes exact continuous derivatives *dx/dt* and d^2x/dt^2 directly at collocation points.

2. Embedding the Physical Residual

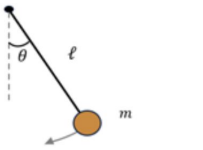
The ODE term (d2x + mu*dx + k*x) is penalized. The network parameters adjust until the output respects the dampening laws of motion.

3. Regularized Multi-Task Optimization

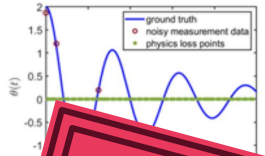
Loss combines boundary conditions (forces *x(0)=1, dx/dt(0)=0*) and the continuous physical regularizer *L_physics*.



+



$$\theta''(t) + 2\beta\theta'(t) + \omega_0^2\theta(t) = 0$$



$$\min_{\sigma} \frac{1}{N}\sum_{i=1}^{N}\left|\theta_{pred}(t_i;\sigma) - \theta_{meas}(t_i)\right|^2$$

$$+\frac{1}{M}\sum_{j=1}^{M}\left|\frac{\partial^2}{\partial t^2}\theta_{pred}(t_j;\sigma) + 2\beta\frac{\partial}{\partial t}\theta_{pred}(t_j;\sigma) +\right.$$