Physics-informed Neural Networks in RL context

Imola Fodor

AI Engineer, Global Tech, Electrolux

Physics-informed Neural Networks (PINNs)

PINNs incorporate physical laws into NNs by including terms that correspond to a system's governing differential equation into the loss function.

Governing laws (PDEs) of nature

- PDEs encode conservation laws the solution has to satisfy:
  - Conservation of mass
  - Conservation of momentum (linear, angular) – PDE Navier-Stokes Eq.
  - Conservation of energy – Heat equation

Objective: Predicting field variables having spatial information and time

Using PINNs in RL context?

Complex realistic systems of interest	Governing equation known
Neuroscience	NO
Epidemiology	NO
Ecology	NO
Turbulence, fluid flow (aircraft design)	YES
Protein folding	YES
Combustion	YES
Quantum systems	YES
Electro-magnetic systems	YES

Training

- Inputs :
  - ***Measurement (md) / IC, BC data***
    - x_1 I/O at $t = 0$ initial conditions (IC)
    - x_2 I/O at boundary conditions (BC)
  - ***Virtual data*** from high fidelity simulation
    - x_3 I/O solution at collocation points
- Original Loss function
  - MSE_p - Compute partial derivatives of the output variables in terms of the input variables (spatio/temporal usually)
  - From pivotal paper (Raissi,Perdikaris, Karniadakis, 2019)

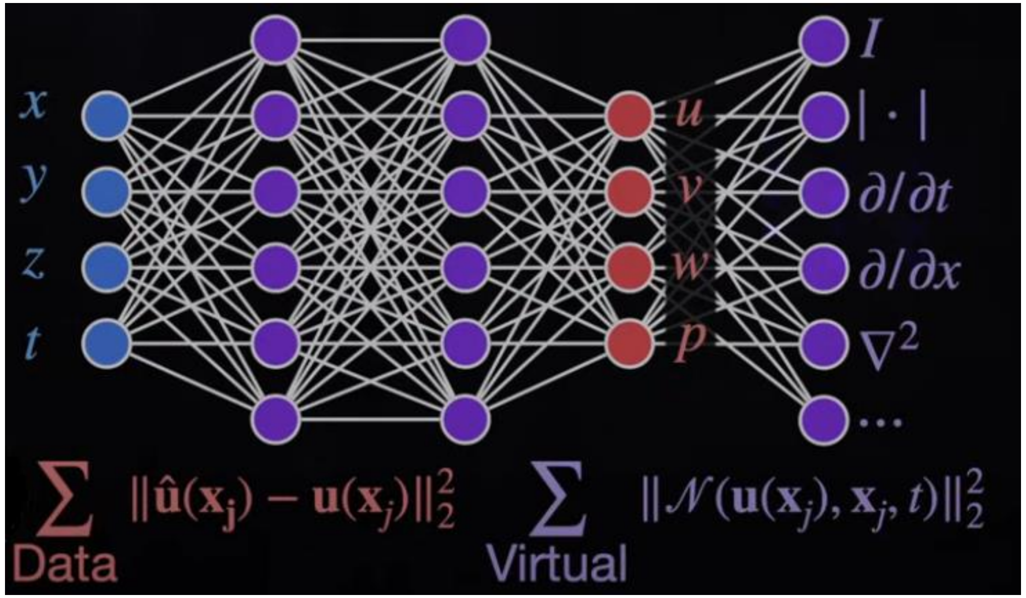
$$MSE = MSE_{IC} + MSE_{BC} + MSE_p$$

- Newer interpretation to possible loss functions:

$$L_1 = L_{IC\&BC} + L_p$$ *minimum requirement*

$$L_2 = L_{md} + L_p$$

$$L_3 = L_{IC\&BC} + L_{md} + L_p$$ *complete loss*



S. Brunton on PINNs YT link

In-depth: Input Data

- L_d loss:
- L_p loss:

Example Damped Harmonic Oscillator (1D problem)

Fig 3: a 1D damped harmonic oscillator

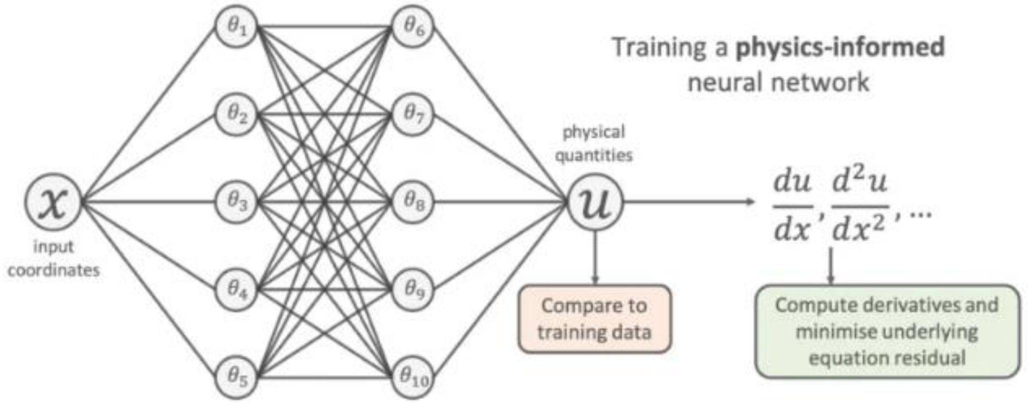


Fig 4: schematic of physics-informed neural network

$$\min \frac{1}{N}\sum_{i}^{N}(u_{\mathrm{NN}}(x_i;\theta) - u_{\mathrm{true}}(x_i))^2$$
$$+\frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dx^2} + \mu\frac{d}{dx} + k\right]u_{\mathrm{NN}}(x_j;\theta)\right)^2$$

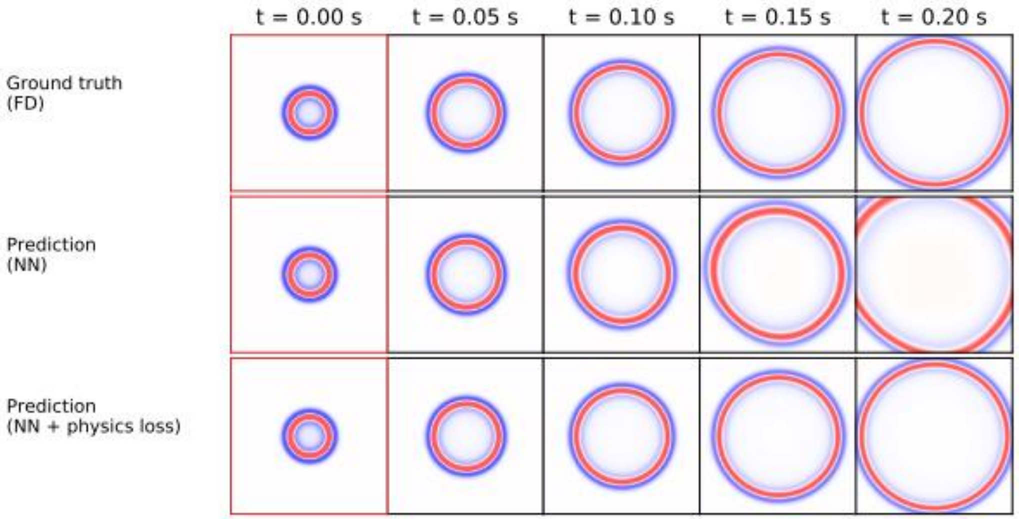
$$m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0$$

Underlying physics equation (2° order DE)

Loss function with physics embedded

Source Ben Moeseley Blog https://benmoseley.blog/blog/, [YT link]

Example Simulate waves



Results

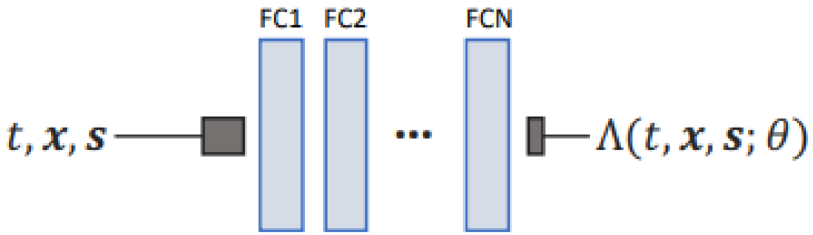


FIG. 1. Physics-informed neural network used to solve the wave equation. The input to the network is a single point in time and space (t, x) and its output is an approximation of the wavefield solution at this location. We extend the original PINN approach proposed by Raissi et al. (2019) by further conditioning the network on the initial source location s. We use a fully connected network architecture with 10 layers, 1024 hidden channels, softplus activation functions before all hidden layers and a linear activation function for the final layer.

$$\rho \nabla \cdot \left(\frac{1}{\rho} \nabla u \right) - \frac{1}{v^2} \frac{\partial^2 u}{\partial t^2} = -\rho \frac{\partial^2 f}{\partial t^2} , \qquad (3)$$

2D acoustic wave equation

Solving the wave equation with physics-informed deep learning, B. Moeseley et al, 2020
https://arxiv.org/pdf/2006.11894

Physics-informed MBRL

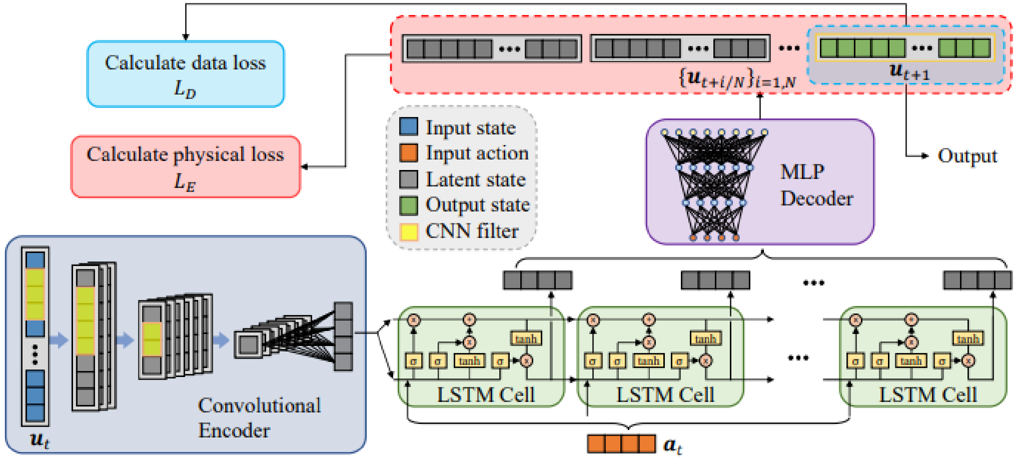


Figure 3: Schematics of LSTM-based neural network architecture for transition model

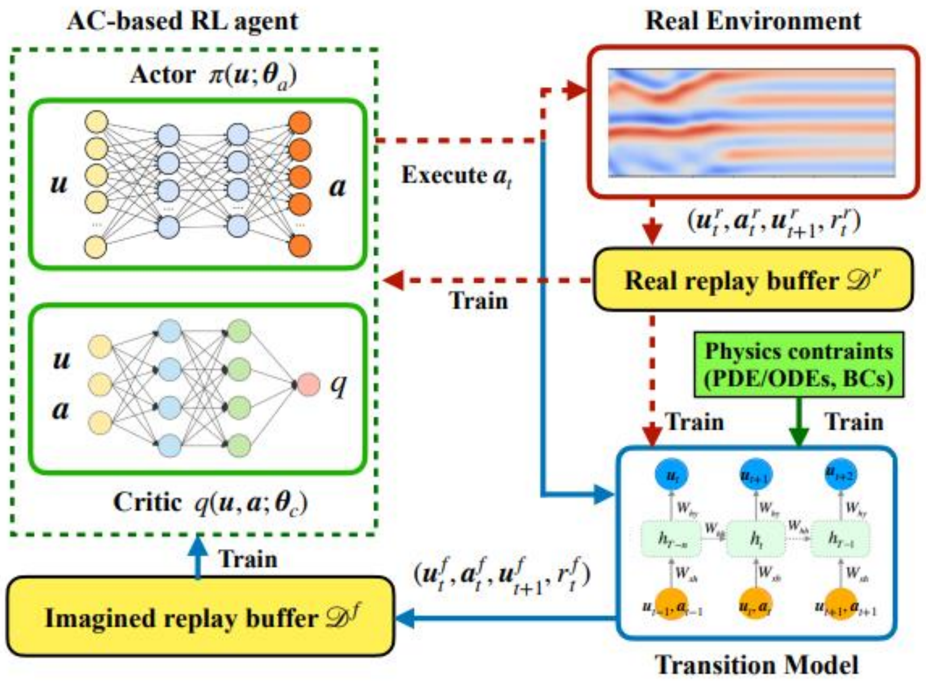


Figure 2: Schematics of Dyna-style physics-informed model-based actor-critic algorithm.

Figure 2, 3, 4 from Physics-informed Dyna-Style Model-Based Deep Reinforcement Learning for Dynamic Control, Xin-Yang Liu and Jian-Xun Wang, 2021

Example CartPole-v0 in MBRL

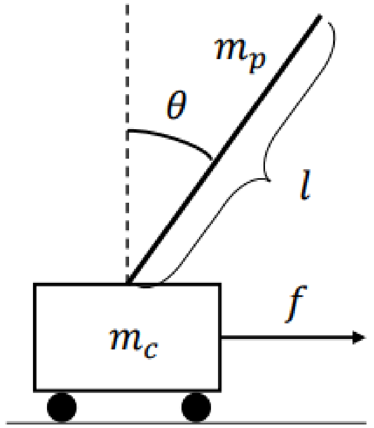
$(\tilde{F}: u_t, a_t \rightarrow u_t + 1)$

$$L_D = \frac{1}{n_b} \sum_{j=1}^{n_b} \left\| \tilde{\boldsymbol{u}}_{t+1}^{(j)} - \boldsymbol{u}_{t+1}^{(j)} \right\|_{L_2}, \quad (2.10)$$

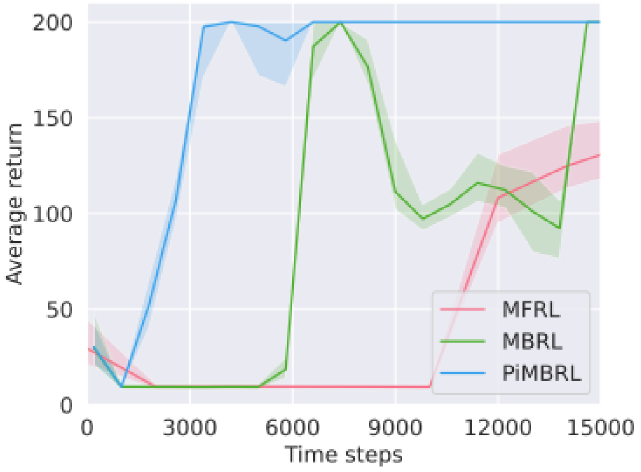
$$L_E = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n_b} \left\| \frac{d\tilde{\boldsymbol{u}}_{t+i/N}^{(j)}}{dt} - \mathcal{F}(\tilde{\boldsymbol{u}}_{t+i/N}^{(j)}, \boldsymbol{a}_{t+i/N})^{n_b} \right\|_{L_2}, \quad (2.11)$$

- ***Measurement / IC, BC data***
  - n_b number of batches from real environment $\mathcal{D}^r$
- ***Virtual data*** from high fidelity simulation
  - N intermediate steps from fictious environment $\mathcal{D}^f$
- Loss function

L_D L_E are not summed, but calculated under different conditions



(a) Cart-Pole



(b) RL Performance

Figure 4: (a) Schematic diagram of the Cart-Pole environment. (b) Performance curve of PiMBRL versus standard model-free TD3 (MFRL) and dyna-like model-based TD3 (MBRL) in Cart-Pole environment. The solid lines indicate averaged returns of 100 randomly selected test episodes, while the shaded area represents the return distribution of all test samples.

Advantages/Drawbacks of PINNs (overall)

Pros	Cons
Sample efficient	Hard optimization problem
Checks if NN is «physical»	The governing physics (PDE) should be known
Can act as Reduced order model (ROM)	... ?
Good predictive accuracy outside the data distribution	
... ?	
Research on learning the PDE	

Example Inverted Pendulum

- Github code: https://github.com/MathPhysSim/PINNs_LNNs_HNNs
- Initial conditions for $t = 0$:

$\varphi = -\frac{\pi}{2}$ rad, $\dot{\varphi} = 0$ rad/s

- Measurement I/O data $\mathcal{S} = \{(t_i, \varphi_i), t_i \in R^1, i = 1, \ldots, N_d\}$.
- Virtual data, only time steps t at input, 150 time steps

- NN topology borrowed from Ben Moesley.