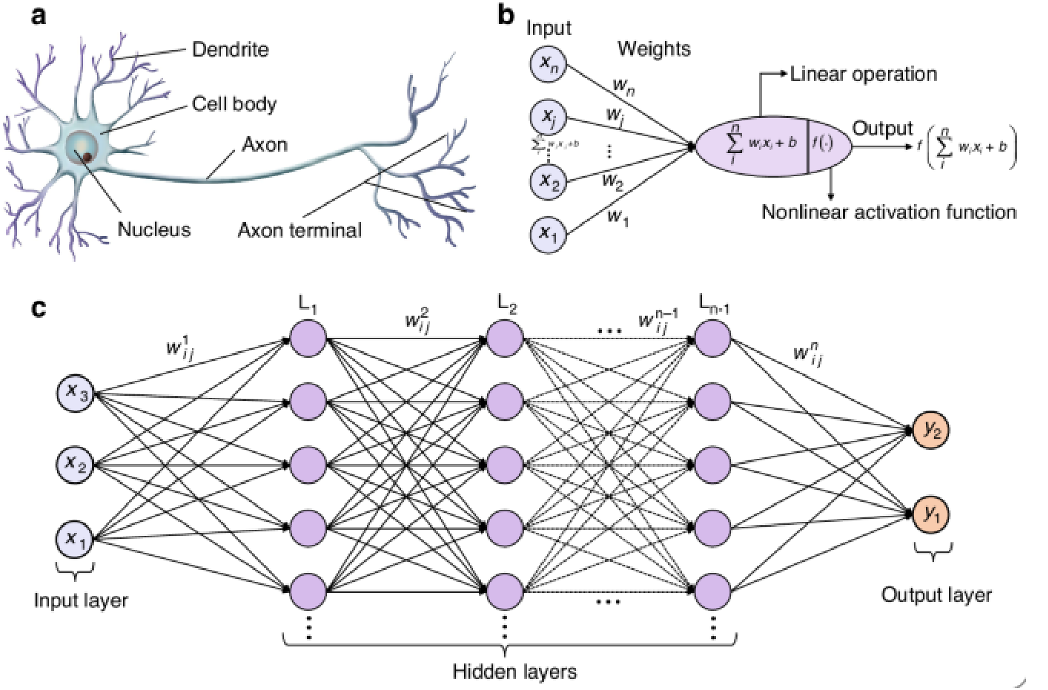
Physics-Informed Neural Networks

A Framework For Solving Two Stream Instability Problem

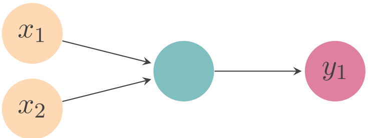
Hosein Shetaie

Aban 1404



a
Dendrite
Cell body
Axon
Nucleus
Axon terminal
b
Input
Weights
x_n
x_j
x_2
x_1
w_n
w_j
w_2
w_1
Linear operation
$\sum_i^n w_i x_i + b$
$f(\cdot)$
Output
$f\left(\sum_i^n w_i x_i + b\right)$
Nonlinear activation function
c
L_1
L_2
L_{n-1}
w_{ij}^1
w_{ij}^2
w_{ij}^{n-1}
w_{ij}^n
x_3
x_2
x_1
y_2
y_1
Input layer
Hidden layers
Output layer

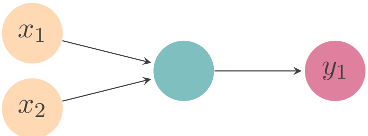
Networks for AND logic



$$y_1 = f(w_1 x_1 + w_2 x_2 + b)$$

x_1	x_2	$y_1 = x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

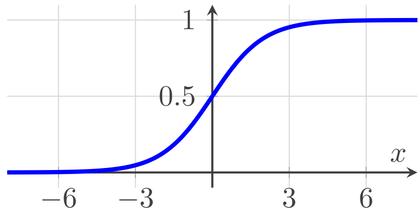
Networks for AND logic



$$y_1 = f(w_1x_1 + w_2x_2 + b)$$

x_1	x_2	$y_1 = x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

$$f(x) = \frac{1}{1 + e^{-x}}$$

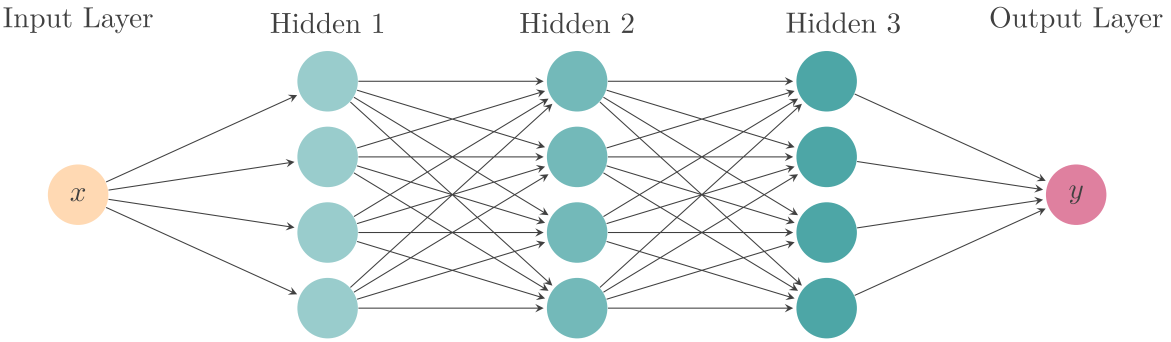


Networks for AND logic

$$y_1 = f(5.5x_1 + 5.5x_2 - 8)$$

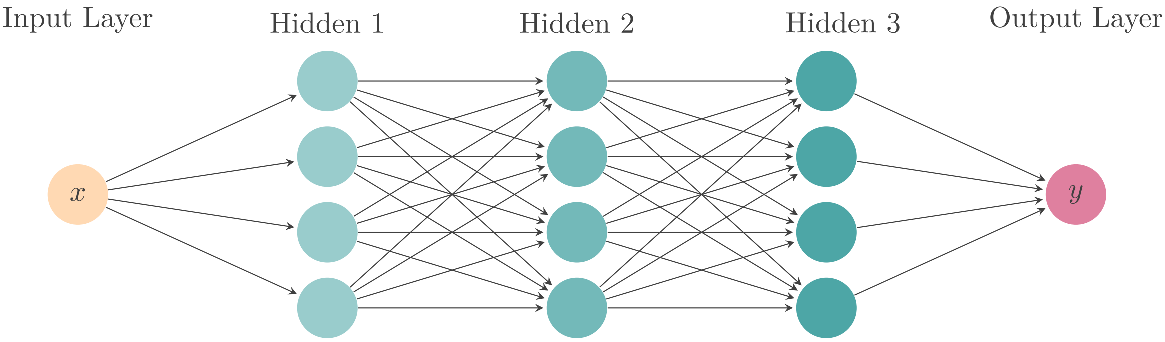
x_1	x_2	Weighted Sum	f(sum)	Output
0	0	-8.0	0.0003	0
0	1	-2.5	0.075	0
1	0	-2.5	0.075	0
1	1	3.0	0.952	1

Table: Neural network output for AND operation.



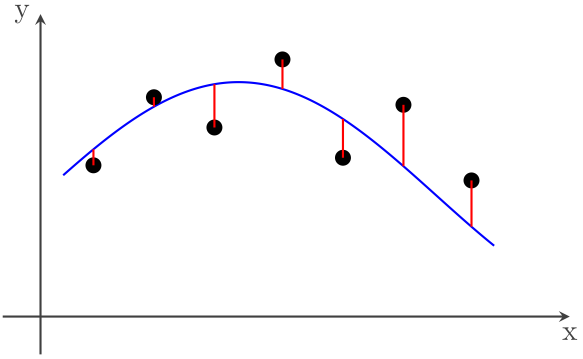
Input Layer
Hidden 1
Hidden 2
Hidden 3
Output Layer
x
y

Training a Neural Networks

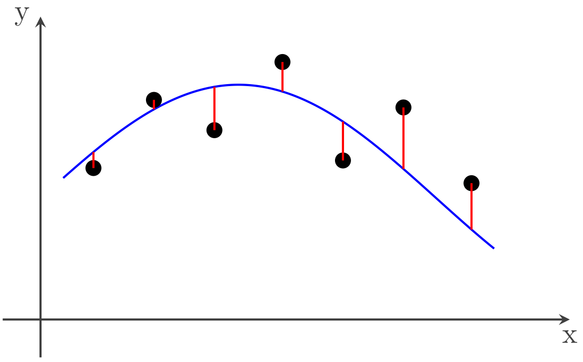


$$\text{Loss} = \frac{1}{N} \sum_{i}^{N} \left(y_{\text{nn}}(x_i) - y_{\text{true}}(x_i)\right)^2$$

$$\text{Cost} = \frac{1}{N}\sum_{i}^{N}\left(y_{\text{nn}}(x_i) - y_{\text{true}}(x_i)\right)^2$$



$$\text{Cost} = \frac{1}{N}\sum_{i}^{N}\left(y_{\text{nn}}(x_i) - y_{\text{true}}(x_i)\right)^2$$



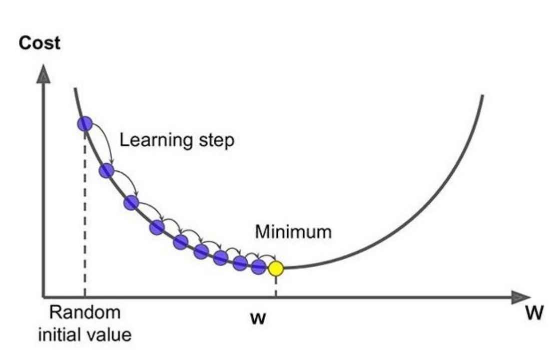


Table of Contents

From Data-Driven to Physics-Driven Learning

The Limitations of Purely Data-Driven Models

- Require large, high-quality datasets
- Perform poorly when data are scarce or noisy
- May violate physical laws (e.g., conservation of mass, energy, momentum)

Motivation for Physics-Guided Learning

- In scientific problems, data are limited but laws are known
- Use domain knowledge (PDEs, constraints) to guide learning
- Integrate physics directly into the loss function
- Learn from both data + physics instead of data alone

Journal of Computational Physics
Volume 378, 1 February 2019, Pages 686-707

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b], G.E. Karniadakis [a]

Show more

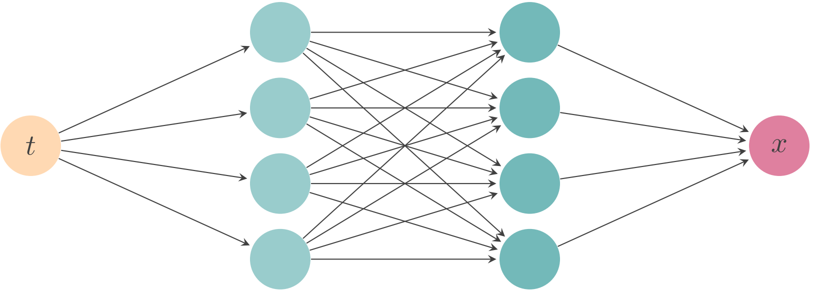
Add to Mendeley Share Cite

https://doi.org/10.1016/j.jcp.2018.10.045

Get rights and content

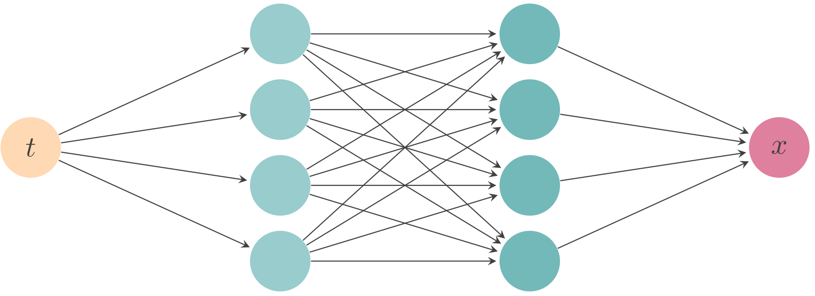
Highlights

- We put forth a deep learning framework that enables the synergistic combination of mathematical models and data.
- We introduce an effective mechanism for regularizing the training of deep neural networks in small data regimes.

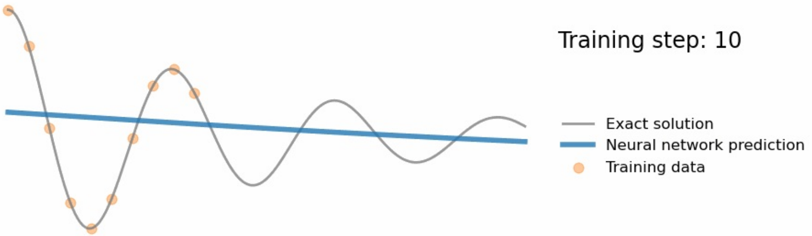


$$\text{Loss} = \frac{1}{N}\sum_{i}^{N}\left(x_{\text{nn}}(t_i) - x_{\text{true}}(t_i)\right)^2$$

Naive NN



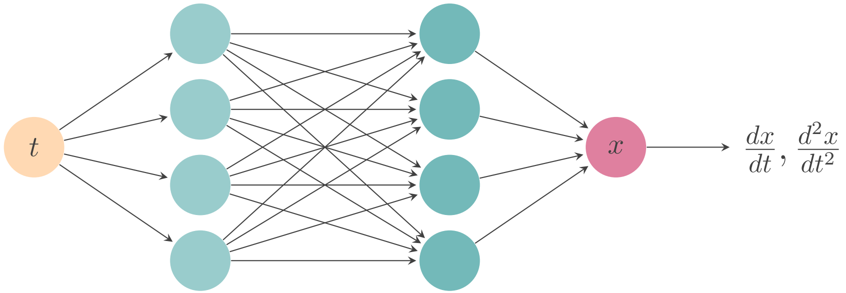
$$\text{Loss} = \frac{1}{N}\sum_{i}^{N}\left(x_{\text{nn}}(t_i) - x_{\text{true}}(t_i)\right)^2$$

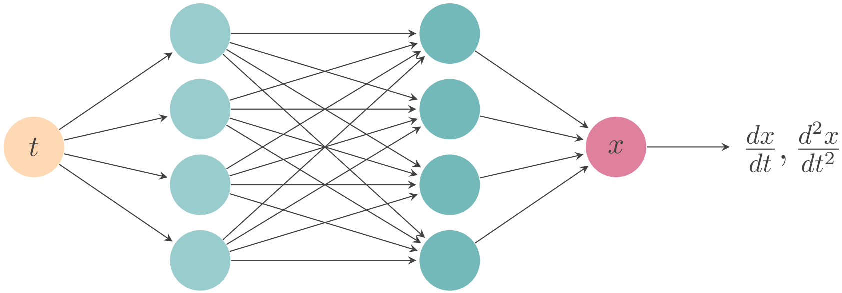


Oscillator

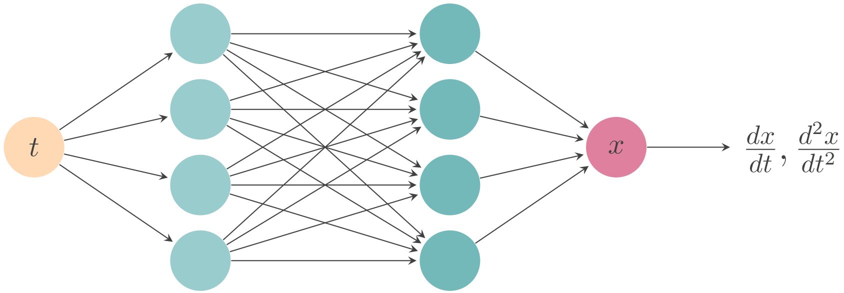
$$m\frac{d^2x}{dt^2} + \mu\frac{dx}{dt} + kx = 0$$

PINN

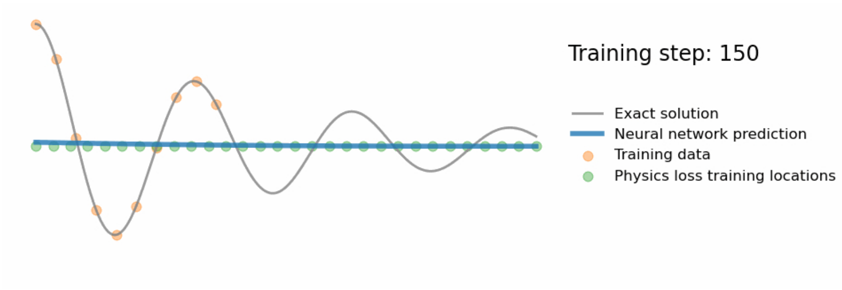




$$\text{Loss} = \frac{1}{N}\sum_{i}^{N}\left(x_{\text{nn}}(t_i) - x_{\text{true}}(t_i)\right)^2 + \frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]x_{\text{nn}}(t_j)\right)^2$$



$$\text{Loss} = \frac{1}{N}\sum_{i}^{N}(x_{\text{nn}}(t_i) - x_{\text{true}}(t_i))^2 + \frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]x_{\text{nn}}(t_j)\right)^2$$



Forward problem

Solving a differential equation by knowing only the initial and or boundary conditions and the goal is to infer them from partial or indirect data.

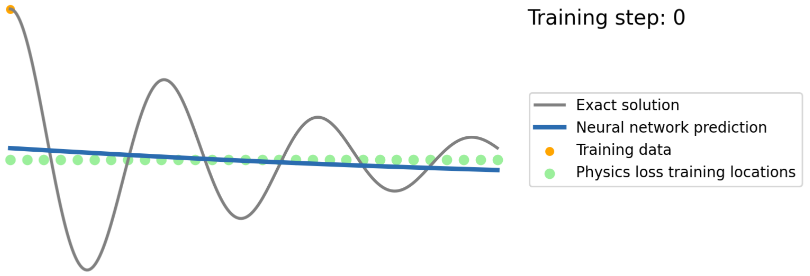
Solving a differential equation by knowing only the initial and or boundary conditions and the goal is to infer them from partial or indirect data.

$$\text{Loss} = (x_{\text{nn}}(t=0)-1)^2 + \lambda_1 \left(\frac{dx_{\text{nn}}}{dt}(t=0) \right)^2 + \frac{\lambda_2}{N} \sum_{j}^{N} \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] x_{\text{nn}}(t_j) \right)^2$$

Forward problem

Solving a differential equation by knowing only the initial and or boundary conditions and the goal is to infer them from partial or indirect data.

$$\text{Loss} = (x_{\text{nn}}(t=0)-1)^2 + \lambda_1 \left(\frac{dx_{\text{nn}}}{dt}(t=0)\right)^2 + \frac{\lambda_2}{N}\sum_j^N \left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right] x_{\text{nn}}(t_j)\right)^2$$



Inverse problem

In an inverse problem, the equation form is known, but some parts of it are unknown and the goal is to infer them from partial or indirect data.

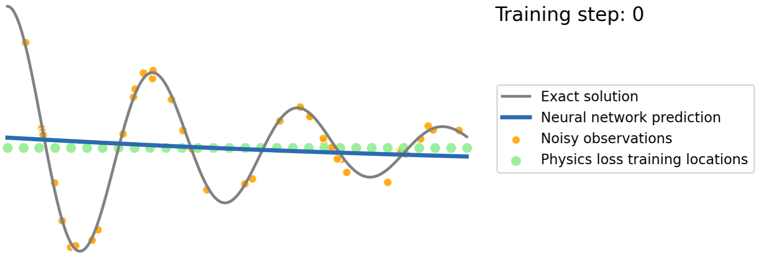
Inverse problem

In an inverse problem, the equation form is known, but some parts of it are unknown and the goal is to infer them from partial or indirect data.

$$\text{Loss} = \frac{\lambda}{N}\sum_{i}^{N}\left(x_{\text{nn}}(t_i) - x_{\text{true}}(t_i)\right)^2 + \frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]x_{\text{nn}}(t_j)\right)^2$$

In an inverse problem, the equation form is known, but some parts of it are unknown and the goal is to infer them from partial or indirect data.

$$\text{Loss} = \frac{\lambda}{N}\sum_{i}^{N}(x_{\text{nn}}(t_i) - x_{\text{true}}(t_i))^2 + \frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]x_{\text{nn}}(t_j)\right)^2$$



PINN for solving Vlasov-Poisson equation

IOPscience Journals Books Publishing Support Login

MACHINE LEARNING
Science and Technology

PAPER • OPEN ACCESS

Physics-informed neural networks for solving forward and inverse Vlasov–Poisson equation via fully kinetic simulation

Baiyi Zhang, Guobiao Cai, Huiyan Weng, Weizong Wang, Lihui Liu and Bijiao He

Published 26 October 2023 • © 2023 The Author(s). Published by IOP Publishing Ltd

Machine Learning: Science and Technology, Volume 4, Number 4

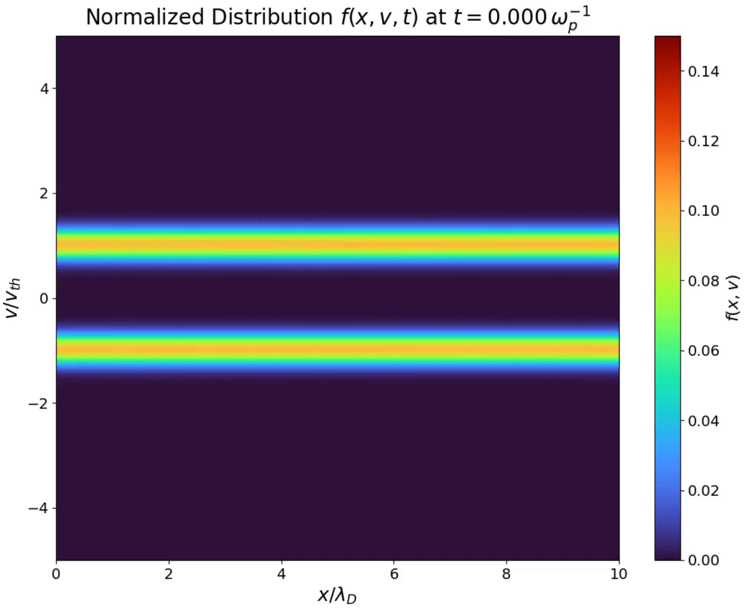
Citation Baiyi Zhang *et al* 2023 *Mach. Learn.: Sci. Technol.* **4** 045015

DOI 10.1088/2632-2153/ad03d5

Article PDF

Authors Figures Tables References

Table of Contents



Normalized Distribution $f(x, v, t)$ at $t = 0.000\,\omega_p^{-1}$
v/v_{th}
x/λ_D
$f(x, v)$
4
2
0
−2
−4
0
2
4
6
8
10
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

Vlasov-Poisson equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon}$$

Vlasov-Poisson equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon}$$

Density:

$$\rho = e(n_e - Z_i n_i)$$

$$n_{\mathrm{e}} = \int_{-\infty}^{+\infty} f\,(t, x, v)\,\mathrm{d}v$$

Vlasov-Poisson equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon}$$

Density:

$$\rho = e(n_e - Z_i n_i)$$
$$n_\mathrm{e} = \int_{-\infty}^{+\infty} f(t, x, v)\,\mathrm{d}v$$

1D Vlasov-Poisson equation:

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + \frac{eE}{m_\mathrm{e}}\frac{\partial f}{\partial v} = 0$$
$$\frac{\partial E}{\partial x} = \frac{e\left(n_\mathrm{e} - n_i\right)}{\varepsilon}$$

Normalization:

$$x = \frac{x^*}{\lambda_{\mathrm{D}}},\ v = \frac{v^*}{v_{\mathrm{th}}},\ t = \frac{t^*}{\omega_{\mathrm{p}}^{-1}},\ m = \frac{m^*}{m_{\mathrm{e}}},\ q = \frac{q^*}{e},\ E = E^* \frac{e\lambda_{\mathrm{D}}}{kT_{\mathrm{e}}}$$

Vlasov-Poisson equation

Normalization:

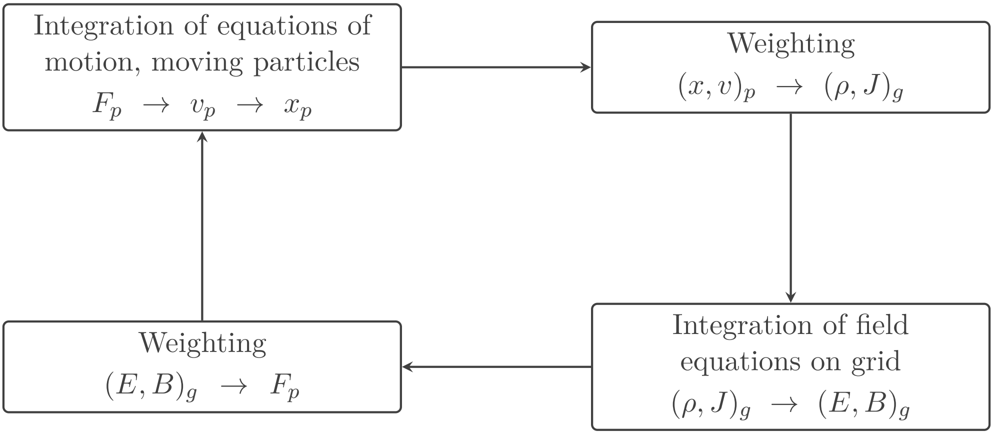
$$x = \frac{x^*}{\lambda_{\mathrm{D}}},\ v = \frac{v^*}{v_{\mathrm{th}}},\ t = \frac{t^*}{\omega_{\mathrm{p}}^{-1}},\ m = \frac{m^*}{m_{\mathrm{e}}},\ q = \frac{q^*}{e},\ E = E^* \frac{e\lambda_{\mathrm{D}}}{kT_{\mathrm{e}}}$$

Normalized 1D Vlasov-Poisson equation:

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \mathrm{E}\frac{\partial f}{\partial v} = 0$$

$$\frac{\partial \mathrm{E}}{\partial x} = \int_{-\infty}^{+\infty} f\left(t, x, v\right) \mathrm{d}v - 1$$

PIC simulation



two stream instability

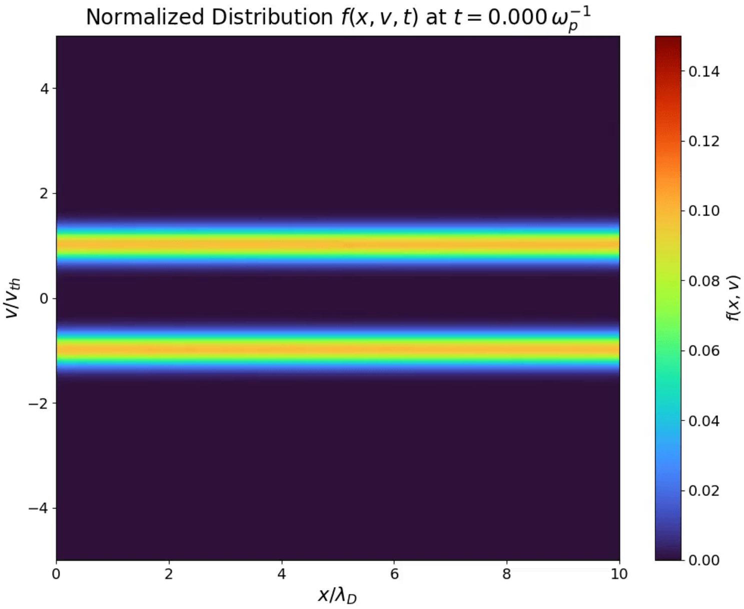
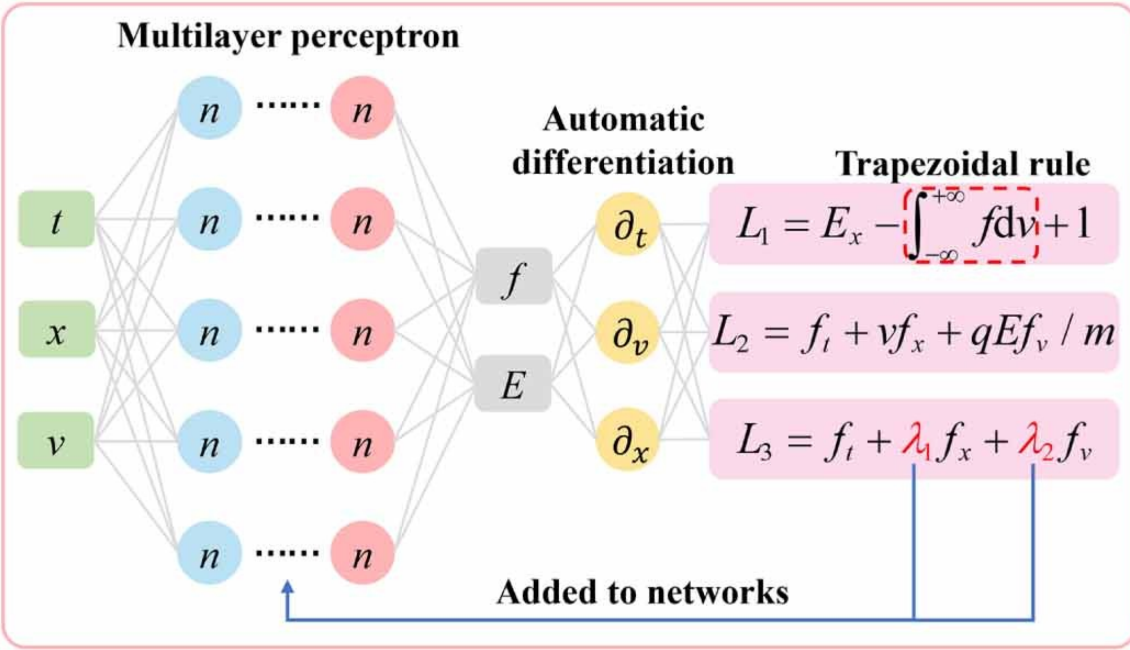


Table of Contents

4 PINN-Vlasov



Multilayer perceptron
t
x
v
n
n
f
E
Automatic differentiation
∂_t
∂_v
∂_x
Trapezoidal rule
$L_1 = E_x - \int_{-\infty}^{+\infty} f \mathrm{d}v + 1$
$L_2 = f_t + v f_x + qEf_v / m$
$L_3 = f_t + \lambda_1 f_x + \lambda_2 f_v$
Added to networks

$$L = L_{\text{eq}} + L_{\text{IC}} + L_{\text{BC}}$$

$$L = L_{\text{eq}} + L_{\text{IC}} + L_{\text{BC}}$$

$$L_{\text{eq}} = \frac{1}{N_{\text{eq}}} \sum_{n=1}^{N_{\text{eq}}} \left[\left(\frac{\partial f}{\partial t} + v \frac{\partial f}{\partial x} + \frac{qE}{m} \frac{\partial f}{\partial v} \right)^2 + \left(\frac{\partial E}{\partial x} - \int_{-\infty}^{+\infty} f \,\mathrm{d}v + 1 \right)^2 \right]_n$$

$$L_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{n=1}^{N_{\text{IC}}} \left[\left(f_{\text{IC}}^{\text{r}} - f_{\text{IC}}^{\text{t}} \right)^2 \right]_n$$

$$L_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{n=1}^{N_{\text{BC}}} \left[\left(f_{\text{BC}}^{\text{r}} - f_{\text{BC}}^{\text{t}} \right)^2 \right]_n$$

$$L = L'_{\mathrm{eq}} + L_{\mathrm{f}}$$

$$L = L^{'}_{\mathrm{eq}} + L_{\mathrm{f}}$$

$$L^{'}_{\mathrm{eq}} = \frac{1}{N^{'}_{\mathrm{eq}}} \sum_{n=1}^{N^{'}_{\mathrm{eq}}} \left[\left(\frac{\partial f}{\partial t} + \lambda_1 v \frac{\partial f}{\partial x} + \lambda_2 E \frac{\partial f}{\partial v} \right)^2 + \left(\frac{\partial E}{\partial x} - \int_{-\infty}^{+\infty} f \, \mathrm{d}v + 1 \right)^2 \right]_n$$

$$L_{\mathrm{f}} = \frac{1}{N_{\mathrm{f}}} \sum_{n=1}^{N_{\mathrm{f}}} \left[\left(f^{\mathrm{\ r}} - f^{\mathrm{\ t}} \right)^2 \right]_n$$

$$L = L_{\text{eq}}^{'} + L_{\text{f}}$$

$$L_{\text{eq}}^{'} = \frac{1}{N_{\text{eq}}^{'}} \sum_{n=1}^{N_{\text{eq}}^{'}} \left[\left(\frac{\partial f}{\partial t} + \lambda_1 v \frac{\partial f}{\partial x} + \lambda_2 E \frac{\partial f}{\partial v} \right)^2 + \left(\frac{\partial E}{\partial x} - \int_{-\infty}^{+\infty} f \, \mathrm{d}v + 1 \right)^2 \right]_n$$

$$L_{\text{f}} = \frac{1}{N_{\text{f}}} \sum_{n=1}^{N_{\text{f}}} \left[\left(f^{\text{ r}} - f^{\text{ t}} \right)^2 \right]_n$$

$$\lambda_1 = 1.0$$

$$\lambda_2 = \frac{q}{m} = \frac{-1.0}{1.0} = -1.0$$

Table of Contents

Challenges

- **Electric field provided to the model in forward problem** In the paper, the electric field $E(x,t)$ was directly supplied to the PINN during forward training. this may limit the model's ability to infer E self-consistently from the physics.
- **Integration in the Poisson equation** The Poisson term $\dfrac{\partial E}{\partial x} = \int f \, dv - 1$ requires numerical integration. The choice of integration method (e.g., trapezoidal rule) affects stability and accuracy.
- **Unknown weighting of loss components** The relative importance of each loss term $(L_{\text{eq}}, L_{\text{IC}}, L_{\text{BC}})$ is not clearly defined-improper weighting can hinder convergence or bias the network.
- **Sampling and resampling strategy** The performance strongly depends on how training points are chosen in (t, x, v). Optimal sampling frequency and resampling intervals remain open research questions.

Challenges

- **Electric field provided to the model in forward problem** In the paper, the electric field $E(x,t)$ was directly supplied to the PINN during forward training. this may limit the model's ability to infer E self-consistently from the physics.
- **Integration in the Poisson equation** The Poisson term $\dfrac{\partial E}{\partial x} = \int f\, dv - 1$ requires numerical integration. The choice of integration method (e.g., trapezoidal rule) affects stability and accuracy.
- **Unknown weighting of loss components** The relative importance of each loss term $(L_{\text{eq}}, L_{\text{IC}}, L_{\text{BC}})$ is not clearly defined-improper weighting can hinder convergence or bias the network.
- **Sampling and resampling strategy** The performance strongly depends on how training points are chosen in (t, x, v). Optimal sampling frequency and resampling intervals remain open research questions.

Observation

Balancing physical accuracy, numerical stability, and training efficiency is still the main challenge for PINN-Vlasov frameworks.

- **Physics-Informed Neural Networks (PINNs)** bridge the gap between deep learning and physical modeling.
- By embedding **governing PDEs** into the loss function, they ensure physical consistency.
- Demonstrated capability for solving both **forward and inverse problems** with high interpretability.

Future Work

- **Improving computational efficiency** — faster training for complex PDEs.
- **Adaptive sampling** — dynamic selection of collocation points.
- **Multi-physics extension** — include magnetic fields, collisions, higher dimensions (2D/3D).

Future Work

- **Improving computational efficiency** — faster training for complex PDEs.
- **Adaptive sampling** — dynamic selection of collocation points.
- **Multi-physics extension** — include magnetic fields, collisions, higher dimensions (2D/3D).

Future PINNs = Physics + Data + Efficiency → Scientific AI.

Q&A

Thank you for listening!