Physics informed Neural Networks for solving PDEs in Plasma Physics

Nick McGreivy (Princeton/PPPL)
Phil Travis (UCLA)

How does it work?

- **Represent the solution to your differential equation with a neural network. For example, if the solution is f(x,v,t) then your neural network's inputs are x, v, and t, and it's output is f.**
- **Write a loss function which is zero if the initial conditions, the boundary conditions, and the PDE are all satisfied at a selection of chosen points in the domain and boundary. Then minimize that loss function with optimization methods. This is a simple and elegant way of solving PDEs.**

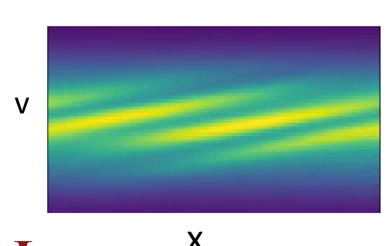
Successes:

A physics informed neural network (PINN) can be successful at solving:

- **Non-coupled PDEs**
- **Parameter estimation in PDEs**
- **High-dimensional systems**
- **Arbitrary domains (no mesh!)**

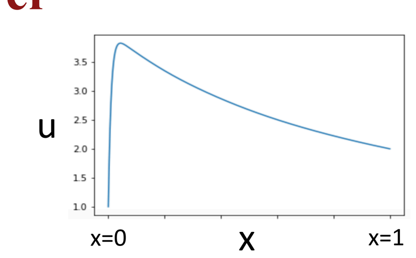
Example: Vlasov Free Streaming

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



Example: Boundary Layer

$$\epsilon\frac{\partial^2 u}{\partial x^2} + (1+x)\frac{\partial u}{\partial x} + u = 0$$



Failures: Coupled PDEs

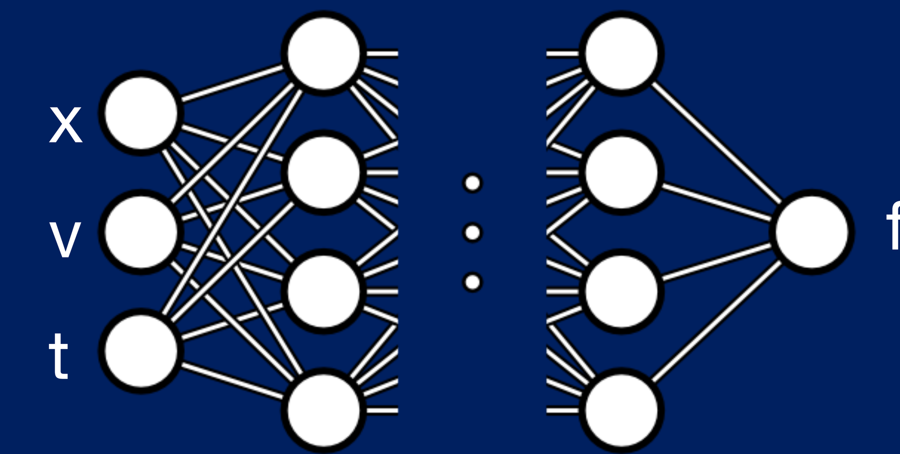
To represent the solution to a set of coupled PDEs, we need either multiple outputs or multiple neural networks. The additional equations regularize the solution, making optimization difficult.

Example: Vlasov-Maxwell

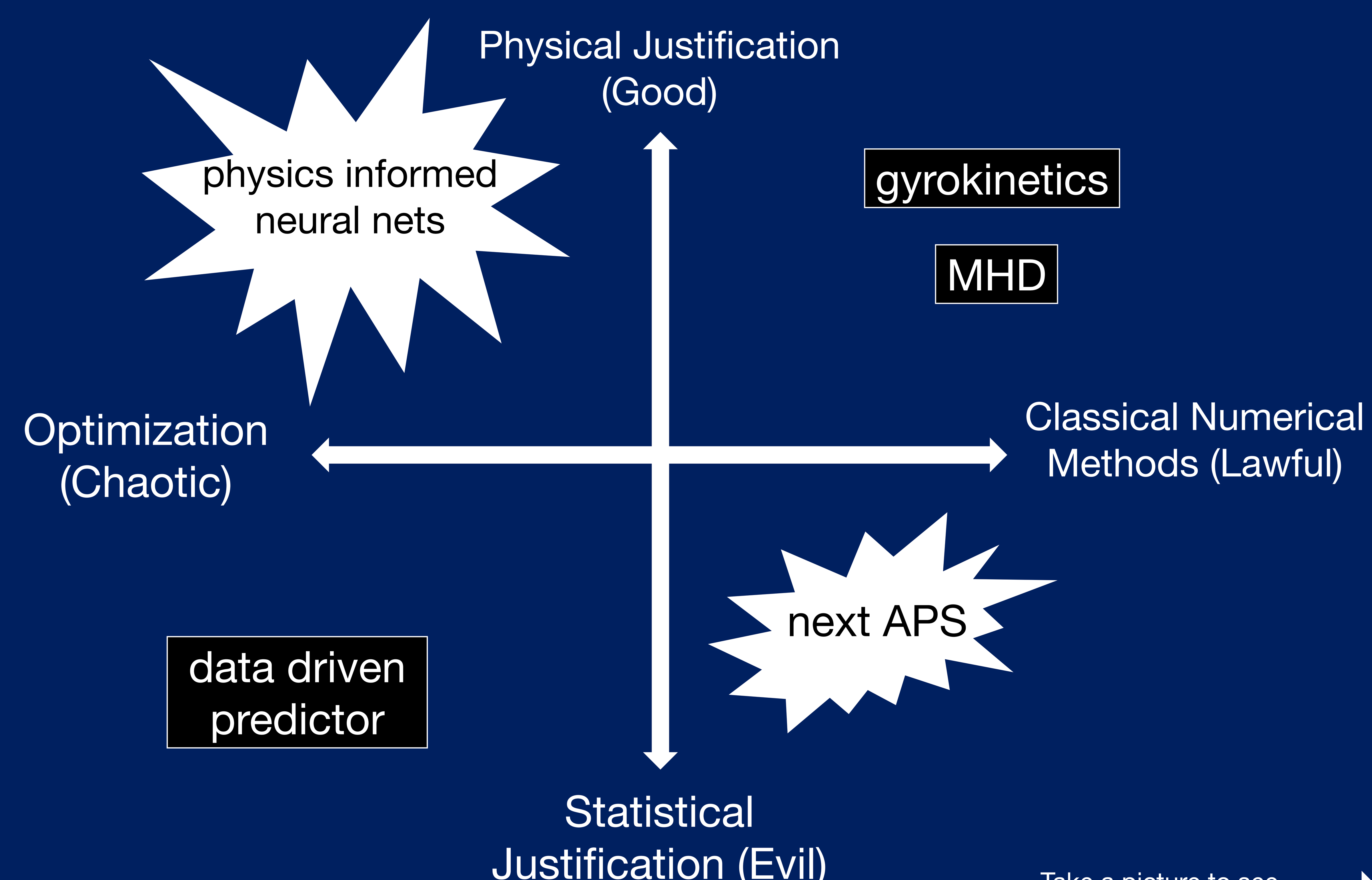
$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + \frac{e}{m}\frac{\partial \phi}{\partial x}\frac{\partial f}{\partial v} = 0 \qquad \frac{\partial^2 \phi}{\partial x^2} - \frac{e}{\epsilon_0}\int f(t,x,v)dv + \frac{en_i}{\epsilon_0} = 0$$

Example: Cold Fluid Equations

$$mn\frac{du}{dt} = -enE \qquad \epsilon_0\frac{\partial E}{\partial x} = -en + en_i \qquad \frac{\partial n}{\partial t} + \frac{\partial}{\partial x}(nu) = 0$$



Neural networks can solve challenging PDEs, but have struggled to solve the ones we care about.



Take a picture to see our github repo!

Loss Function

$$\text{Loss} = L_{equation} + L_{init} + L_{boundary}$$

$$L_{equation} = \sum_{i=1}^{N}\left(\left[\frac{\partial}{\partial t} + \mathcal{N}_x\right]\hat{f}(t_i, x_i)\right)^2$$

$$L_{init} = \sum_{i=1}^{N_{init}}\left(\hat{f}(t=0, x_i) - f_0(x_i)\right)^2$$

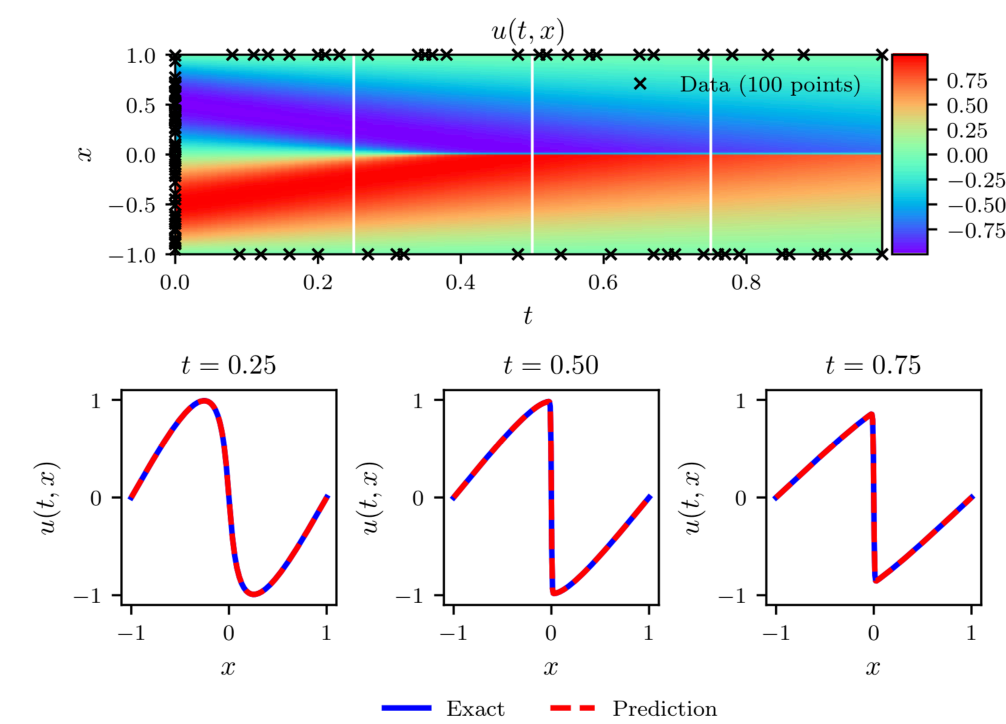
$$L_{boundary} = \sum_{i=1}^{N_b}\left(\hat{f}(t_i, x_{max}) - \hat{f}(t_i, x_{min})\right)^2$$

N: Number of training points in the domain
N_{init}: Number of training points in the initial conditions
N_b: Number of training points on the boundary
$\hat{f}$: Prediction of Network

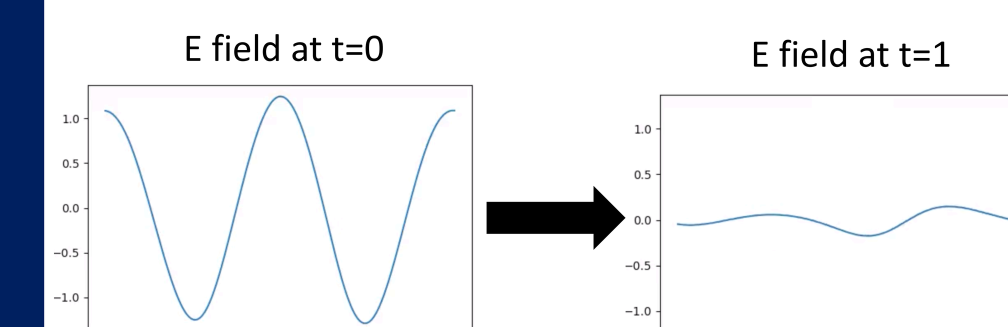
Who reintroduced this idea?

Mazair Raissi (Brown)

Take a picture to see his 2017 paper!



What does failure look like?



For the Vlasov-Maxwell system, the solver tries to minimize the loss function but does so in a way which breaks conservation laws while minimizing loss function.

Different idea: Hamiltonian Neural Networks

Neural networks can learn conservation properties if given the right inductive bias.