Physics Informed Neural Networks (PINNs) for MHD Modelling

Hubert Baty, ObAS, Université de Strasbourg, France

&

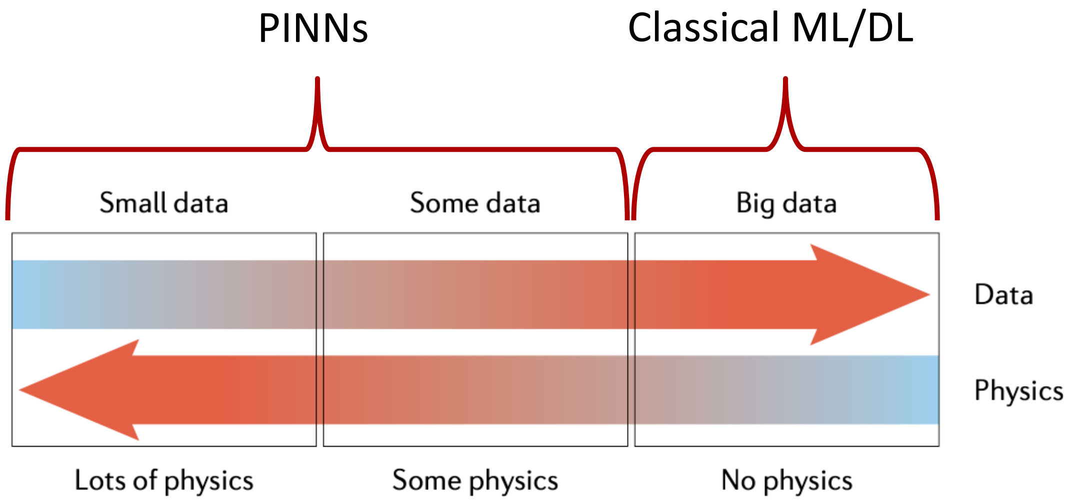
Vincent Vigon, IRMA (INRIA & Tonus team), Université de Strasbourg, France

Thanks to Emmanuel Franck & Victor-Michel Dansac (IRMA)

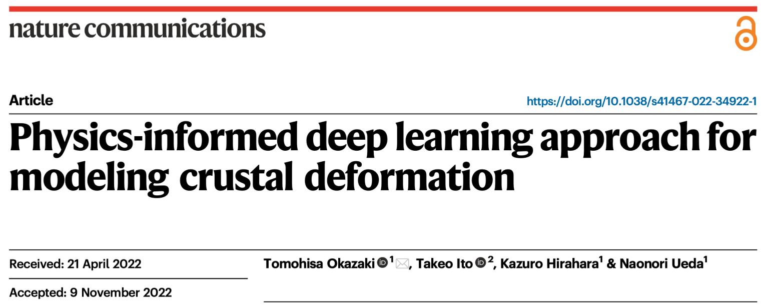
What is it about ?

- **PINNs: optimizations deep learning (DL) based methods for academic & industrial research**
  -> recent strong surge of interest in many fields !

- **PINNs seamlessly incorporate data and physical laws (ODEs or PDEs) in a unified way**
  -> application to many different problems



See review by Karniadakis et al., Nature reviews 2021


nature communications

Article

https://doi.org/10.1038/s41467-022-34922-1

Physics-informed deep learning approach for modeling crustal deformation

Received: 21 April 2022

Accepted: 9 November 2022

Tomohisa Okazaki[1], Takeo Ito[2], Kazuro Hirahara[1] & Naonori Ueda[1]

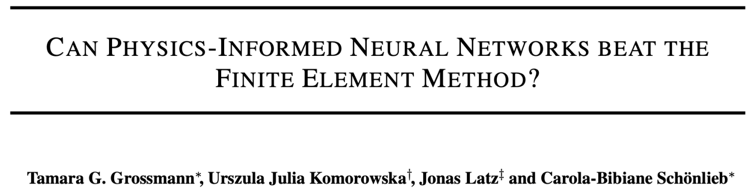

frontiers in Big Data

ORIGINAL RESEARCH
published: 19 November 2021
doi: 10.3389/fdata.2021.669097

The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?

Stefano Markidis *

KTH Royal Institute of Technology, Stockholm, Sweden


CAN PHYSICS-INFORMED NEURAL NETWORKS BEAT THE FINITE ELEMENT METHOD?

Tamara G. Grossmann*, Urszula Julia Komorowska†, Jonas Latz‡ and Carola-Bibiane Schönlieb*

Current concern

Presentation plan

- **Basics of PINNs**
- **Application to MHD equilibria**
- **Application to MHD reconnection**
- **Conclusions and prospectives**

Potentiality of PINNs:
aim to test advantages/drawbacks
vs traditional solvers

Basics of PINNs

- **Differential equation in a bounded domain:**

- PDE in residual form: $\mathcal{F}\left[u(\boldsymbol{x}), \boldsymbol{x}\right] = 0$

Differential operator

- **Define a data set of N_c collocation points:**

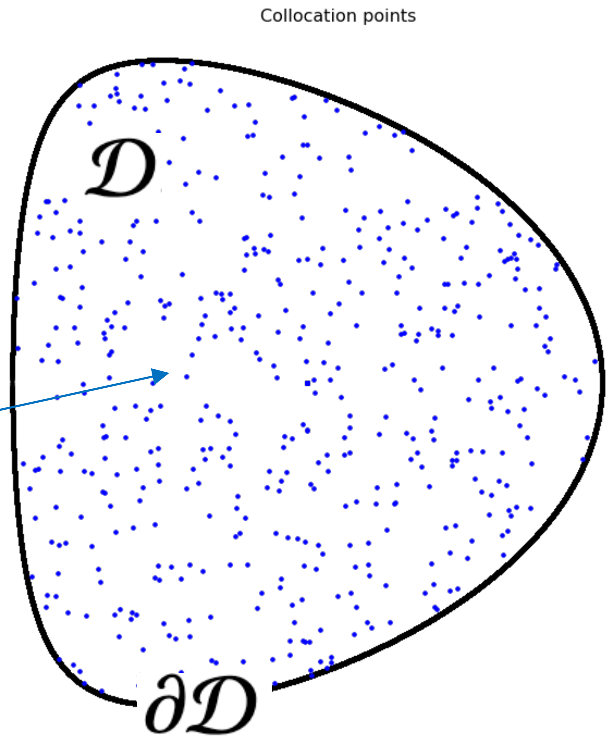
=> physics-based loss function:

$$\mathcal{L}_{\mathcal{F}}(\theta) = \frac{1}{N_c} \sum_{j=1}^{N_c} \left| \mathcal{F}[u_\theta(\boldsymbol{x}_j), \boldsymbol{x}_j] \right|^2$$

mean squared error

θ**: parametrization**

a differentiation tool is needed



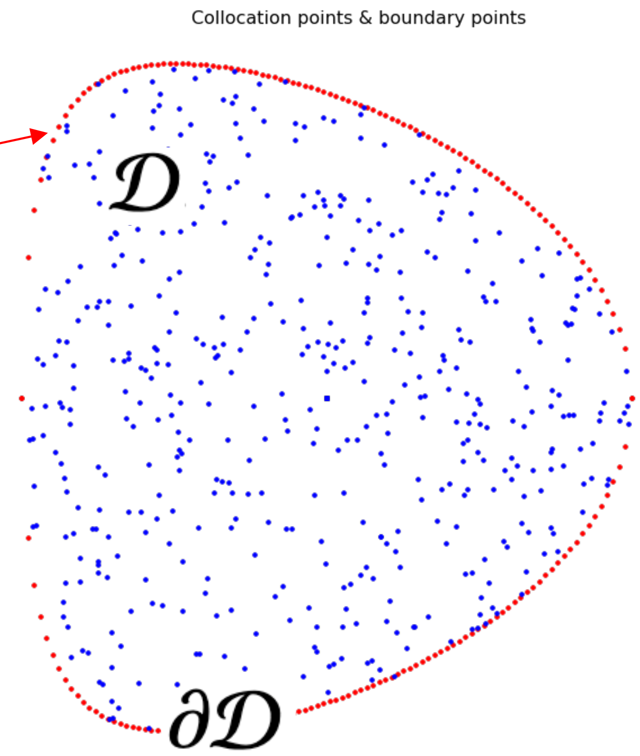
- **Minimization method to find the optimal solution =>** $u_\theta(x)$

Basics of PINNs

- **Differential equation in a bounded domain:**

  - Dirichlet boundary conditions (BCs)

  (Neumann/Robin conditions are also possible)



- **Define a data set of N_{data} boundary points:**

  => Training data loss function:

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left| u_\theta(\boldsymbol{x}_i) - u_i^{data} \right|^2$$

mean squared error

θ **: parametrization**

-> N_{data} can also include the data knowledge of some interior points …

- **Minimization method using a weighted total loss to find the optimal solution => $u_\theta(x)$**

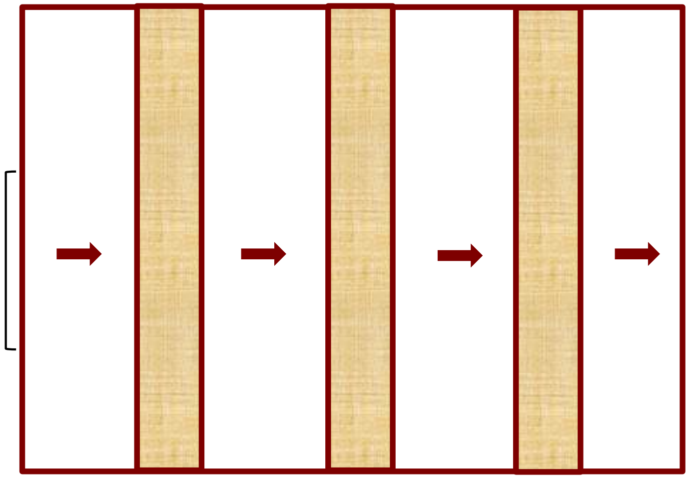
$$\mathcal{L}(\theta) = \omega_{data}\mathcal{L}_{data}(\theta) + \omega_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\theta)$$

weights

Basics of PINNs

- **Minimization using a <u>feed-forward neural network</u> -> universal non-linear approximator**

$\boldsymbol{x} = (x, y)$

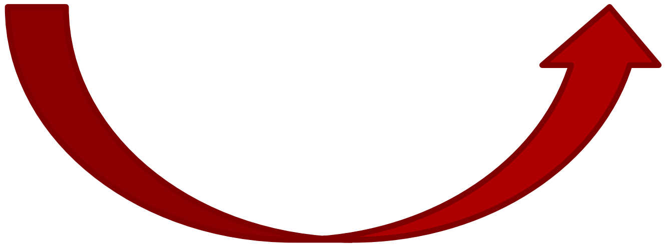


-> $u_\theta(\boldsymbol{x})$

θ **: parametrization**

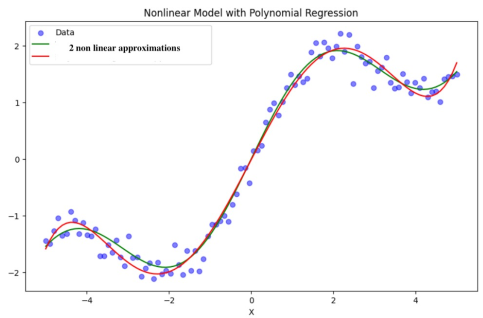
knowledge of u_i^{data} at $\boldsymbol{x_i}$

$$u_\theta(\boldsymbol{x}) = (\mathcal{N}^{(L)} \circ \mathcal{N}^{(L-1)} \dots \mathcal{N}^{(0)})(\boldsymbol{x})$$



- Finding map between inputs and output

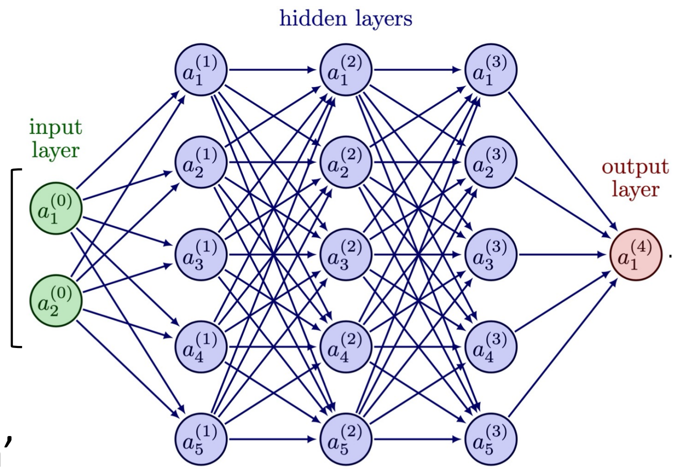
- Recursive way -> sequence of non linear functions



Basics of PINNs

- **Minimization using a feed-forward neural network -> universal non-linear approximator**

$\boldsymbol{x} = (x, y)$



-> $u_\theta(\boldsymbol{x})$

σ : Activation function -> ‘tanh’

$$\mathcal{N}^{(l)}(\boldsymbol{x}) = \sigma(\boldsymbol{W}^{(l)}\mathcal{N}^{(l-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(l)})$$

$$u_\theta(\boldsymbol{x}) = (\mathcal{N}^{(L)} \circ \mathcal{N}^{(L-1)} \dots \mathcal{N}^{(0)})(\boldsymbol{x})$$

weight matrices and biases: $\theta = \{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}_{l=1,L}$

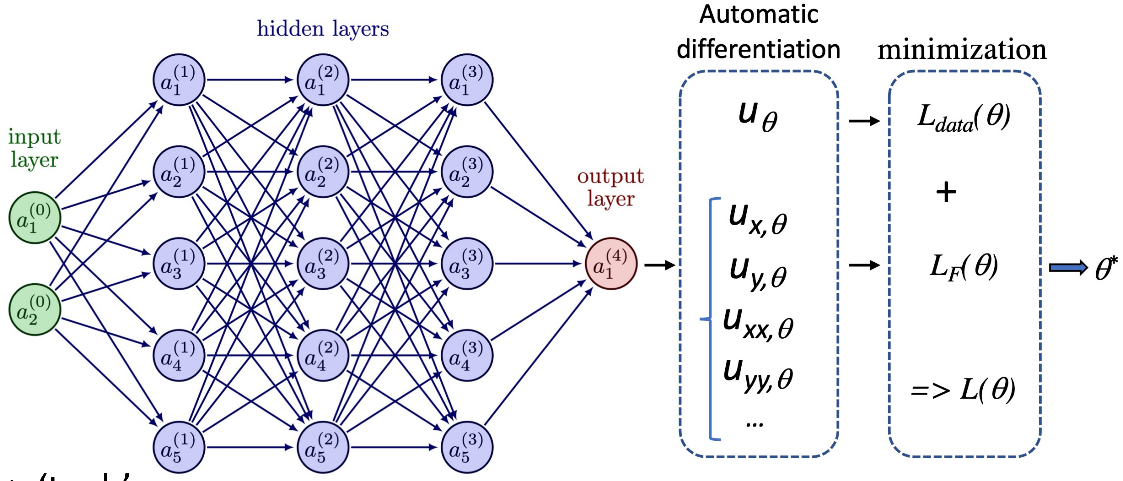
-> trainable parameters

Hidden layers -> affine maps & nonlinear activation function

Units: **artificial neurons** -> brain-inspired

Basics of PINNs

- **Minimization using a feed-forward neural network for PINNs**



σ : Activation function -> 'tanh'

$$\mathcal{N}^{(l)}(\boldsymbol{x}) = \sigma(\boldsymbol{W}^{(l)}\mathcal{N}^{(l-1)}(x) + \boldsymbol{b}^{(l)})$$

$$u_\theta(\boldsymbol{x}) = (\mathcal{N}^{(L)} \circ \mathcal{N}^{(L-1)} \ldots \mathcal{N}^{(0)})(\boldsymbol{x})$$

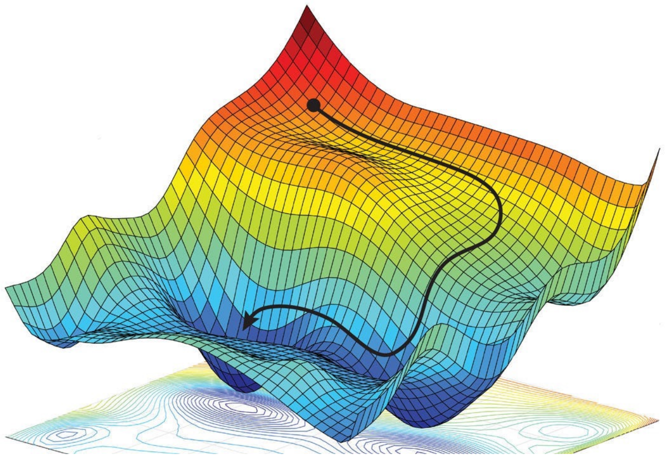
weight matrices and biases: $\theta = \{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}_{l=1,L}$

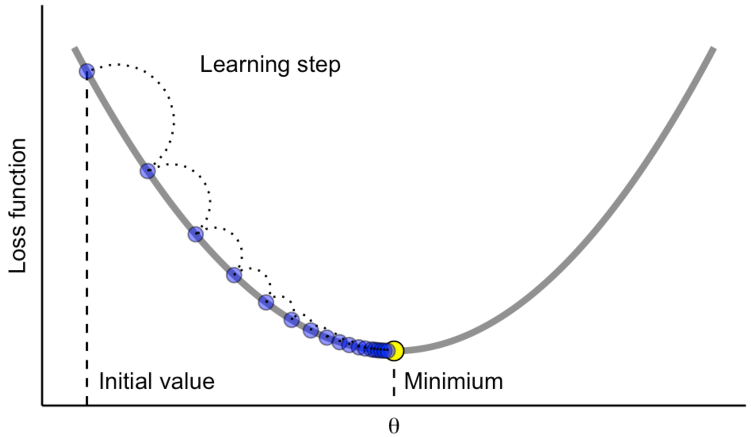
- **A gradient descent algorithm: l_r is the learning rate**

$$\theta_{i+1} = \theta_i - l_r \nabla_\theta \mathcal{L}(\theta_i) \qquad \theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta)$$

Parameters are iteratively calibrated during the **training process**

Basics of PINNs





- **Many pitfalls: regions with plateau (zero gradient), multiple local minima, too high or too low learning rate, …**

  => Efficient optimizers are needed (a stochastic one is used) !

- **A gradient descent algorithm: l_r is the learning rate**

$$\theta_{i+1} = \theta_i - l_r \nabla_\theta \mathcal{L}(\theta_i) \qquad \theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta)$$

↑ **a differentiation tool is needed**

A complete (iteration) pass across the network is called **epoch** in ML/DL

Basics of PINNs

- **Python libraries for deep learning are very efficient and optimized**

  - Pytorch and Tensorflow (used in this work)

  - Different optimizers for gradient descent (Adam is used)

  - Automatic differentiation is used for gradient descent (w.r.t. θ) and for differential operator (w.r.t. inputs) => contrary to traditional methods the derivatives are computed exactly !

- **Many PINNs-variants**

  - Method above -> 'vanilla-PINNs', popularized after Raissi et al. (2019)

  - BC's can be imposed with 'hard constraints' by specific trial functions for the solution -> see Lagaris (1998), but difficult to use for non cartesian geometry and/or non homogeneous conditions

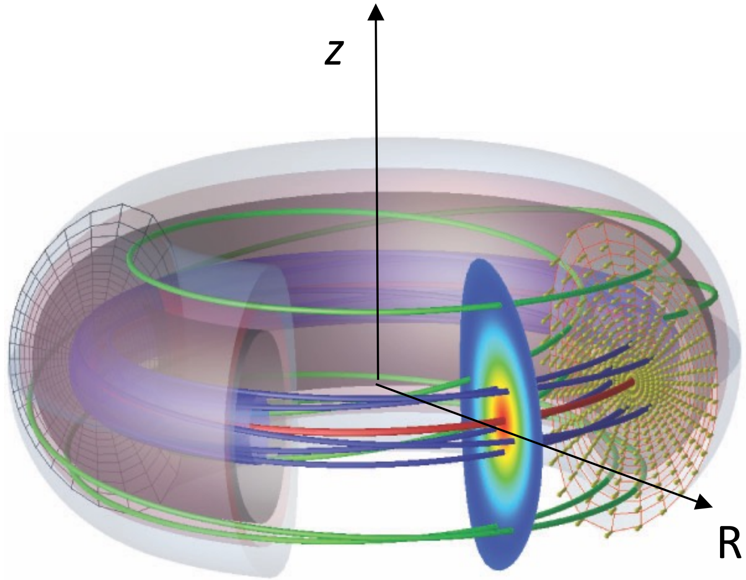
Application to MHD equilibria

- **Axisymmetric ideal MHD (tokamak, …) equilibria**

$$R\frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial\psi}{\partial R}\right)+\frac{\partial^2\psi}{\partial z^2}=-\mu_0 R^2\frac{\partial P}{\partial\psi}-F\frac{\partial F}{\partial\psi}$$

-> Grad-Shafranov (GS) equation

ψ **is the poloidal flux**, $F(\psi)$ is the net poloidal current, and $P(\psi)$ is the thermal pressure



ITER-like equilibria
http://homepage.tudelft.nl/20x40/MHDeq.html

- **PINNs solver (for fixed-boundary problem)**

Similar solvers under development: Jang et al. Maryland university 2023, Kaltsas & Throumoulopoulos 2022 (also include toroidal flow effect)

Our equation residual is:

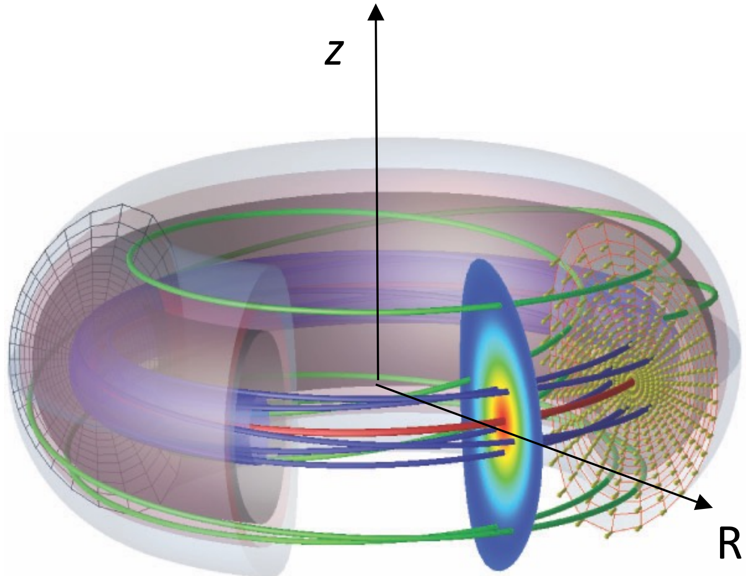
$$\left[R\frac{\partial^2\psi}{\partial R^2}+R\frac{\partial^2\psi}{\partial z^2}-\frac{\partial\psi}{\partial R}\right]+RH(R,z,\psi)=0$$

$$H(R,z,\psi)=\mu_0 R^2\frac{\partial P}{\partial\psi}+F\frac{\partial F}{\partial\psi}$$

Application to MHD equilibria

- **Axisymmetric ideal MHD (tokamak, …) equilibria**

$$R\frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial\psi}{\partial R}\right)+\frac{\partial^2\psi}{\partial z^2}=-\mu_0 R^2\frac{\partial P}{\partial\psi}-F\frac{\partial F}{\partial\psi}$$



- **Solov'ev equilibrium (1) for GS equation**

$$H = f_0(R^2 + R_0^2)$$

Exact analytical solution -> for error and for BCs !
see Deriaz et al. 2011

$$\psi = \frac{f_0 R_0^2}{2}\left[a^2 - z^2 - \frac{(R^2-R_0^2)^2}{4R_0^2}\right]$$

$$\partial\mathcal{D} = \left[R = R_0\sqrt{1+\frac{2a\cos\alpha}{R_0}}, z = aR_0\sin\alpha, \alpha = [0:2\pi]\right]$$

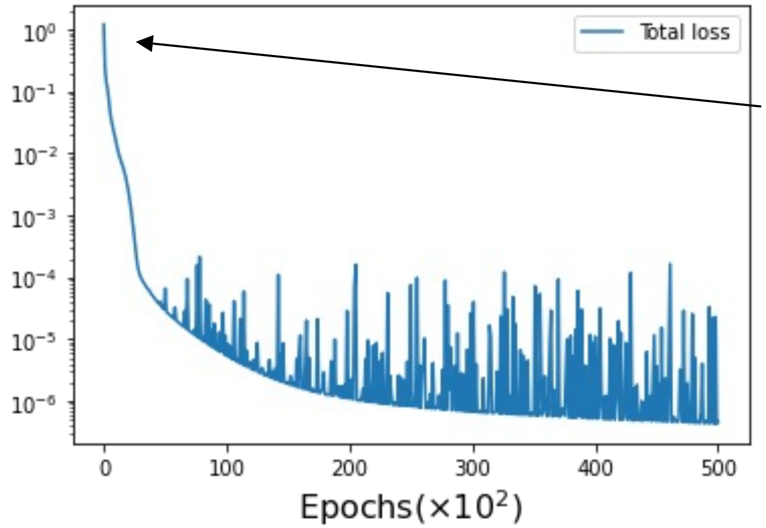
R_0 , a : major, minor radii

f_0 : arbitrary factor

- Application using: f_0 = 1, R_0 = 1 , a = 0.5 (μ_0 = 1)

Application to MHD equilibria

- **Results for Solov'ev equilibrium (1)**

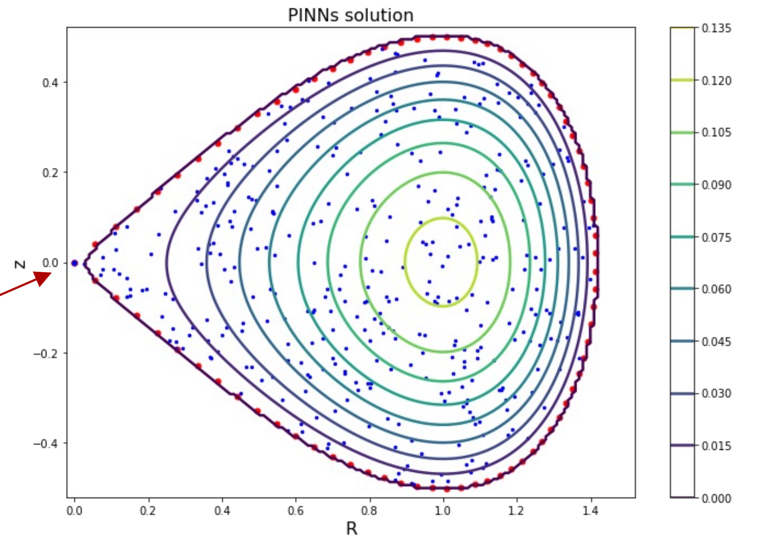


θ parameters
Randomly initialized

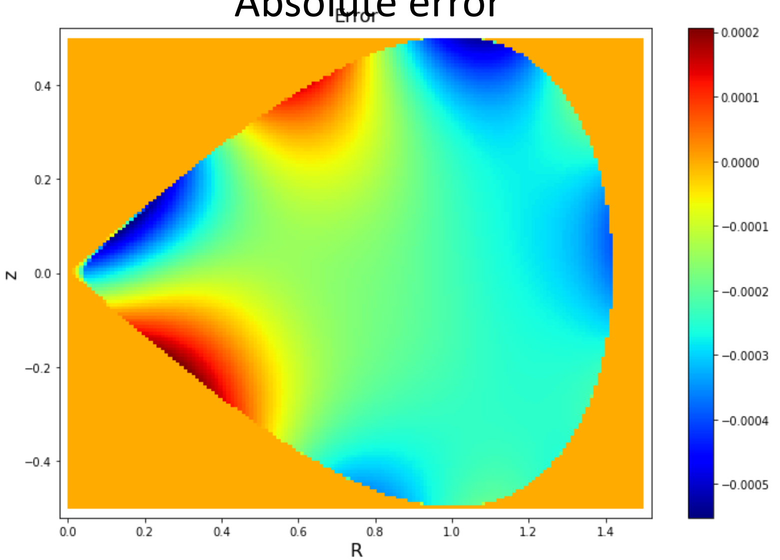
X-point

a third set of points is used to test ->

ψ –isocontours



Absolute error



- Parameters used:

$l_r = 2.\ 10^{-4}$, $\omega_{data} = \omega_F = 1$, $N_c = 800$, $N_{data} = 80$

7 hidden layers with 20 neurons/layer -> **2601 parameters**

Adam optimizer (stochastic gradient descent)

Training stopped after 50 000 epochs

- a few minutes on a single (8 cores) CPU

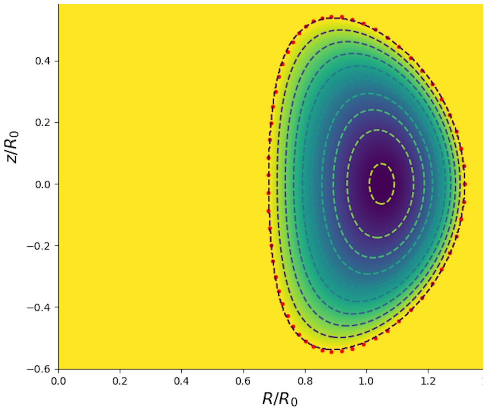
θ

Application to MHD equilibria

- **Results for D-shaped ITER-like Solov'ev (2) and non-linear equilibria**

$$H(R, z, \psi) = (1 - A)R^2 + A$$

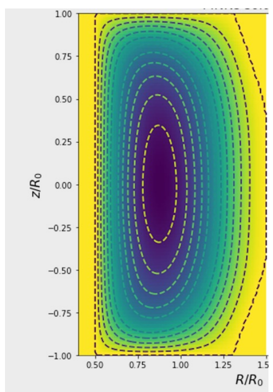
See Cerfon & Freidberg 2010 (A = - 0.155)



$$H(R, z, \psi) = (AR^2 + B)(1 - \psi)^{0.6}$$

procedure is the same without extra effort !

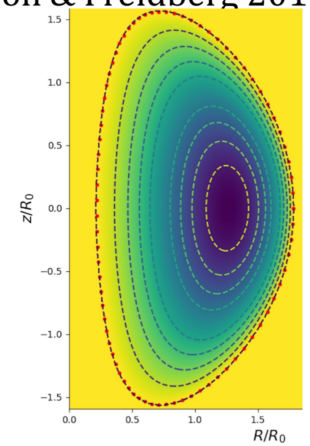
See Itakagi et al. 2004



- Lagaris BC ($\psi = 0$) rectangular domain

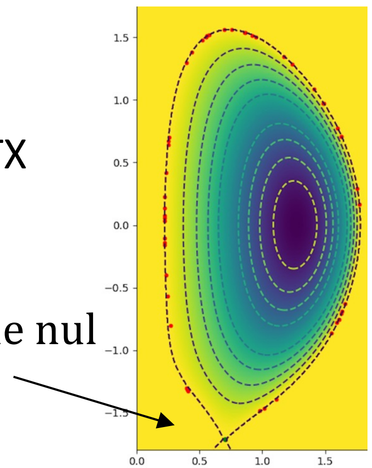
- **Results for other configurations: spherical tokamak (NSTX-like), spheromak, FRC**

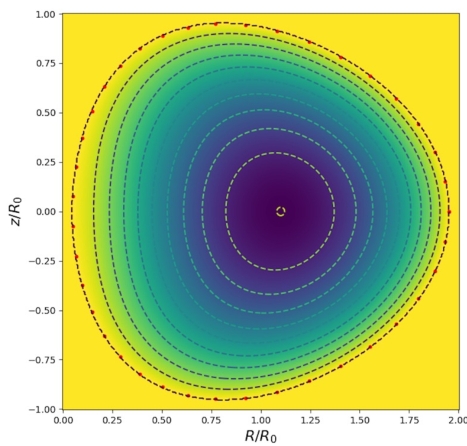
(Cerfon & Freidberg 2010)

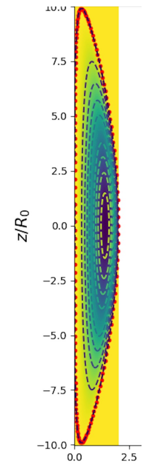


NSTX

Single nul







Application to MHD equilibria

- **PINNs: interesting alternatives to classical methods (finite element FE …)**

-> Easy to handle, meshless methods (collocation & training data sets can be very small)

-> Once trained, the solution (and derivatives) instantaneously obtained

-> Could be used in many different ways: adding data knowledge for learning unknown physical terms (inverse problem for profile reconstruction)

- not done here

-> The precision is only good/average (but can be ameliorated -> conclusions)

- Maximum relative error is of order 10^{-4} versus 10^{-5} - 10^{-10} for finite-element codes

see Lee & Cerfon 2015, and Lutjens et al. 1996 (CHEASE code)

- No scaling laws of the error with the hyperparameters:

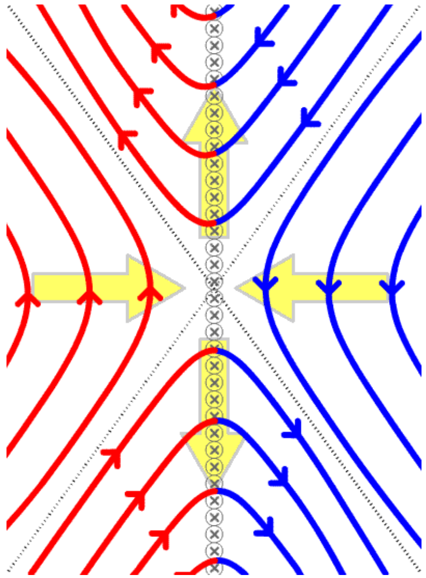
l_r, number of layers/neurons, N_{data}, N_c, weights

Application to MHD reconnection

- **2D steady-state reconnection**

- **Craig-Henton exact analytical solutions for incompressible inviscid plasmas in 2D cartesian coordinates**

Craig & Henton ApJ 1995, see also Baty & Nishikawa MNRAS 2016



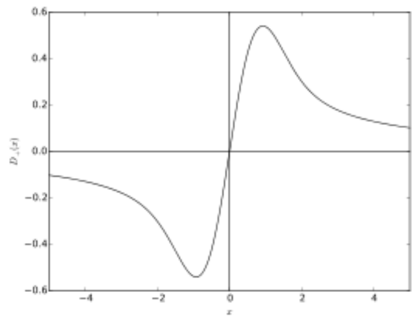
Schematic view (Wikipédia)

Square spatial domain $[-1, 1]^2$

$$\boldsymbol{B} = \left(\beta x, -\beta y + \frac{E_d}{\eta\mu} Daw(\mu x)\right) \qquad \boldsymbol{V} = \left(-\alpha x, \alpha y - \frac{\beta}{\alpha}\frac{E_d}{\eta\mu} Daw(\mu x)\right)$$

for $\beta = 0$ => pure annihilation with a stagnation point flow

$$\mu^2 = \frac{\alpha^2 - \beta^2}{2\eta\alpha}$$ Dawson function -> $$Daw(x) = \int_0^x \exp(t^2 - x^2)dt$$



$0 < \beta < 1$

η : resistivity, E_d : reconnection rate

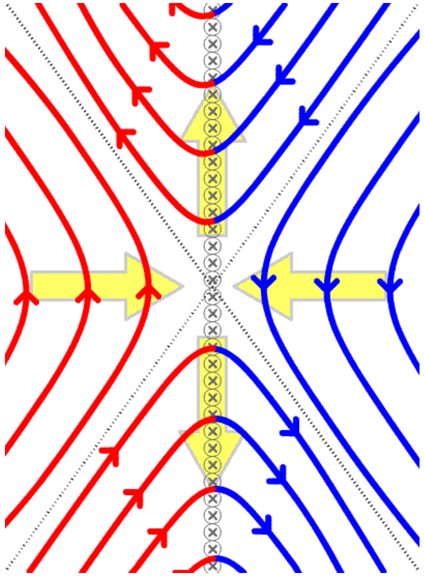
Application to MHD reconnection

- **PINNs code for 2D steady-state reconnection**

$$\boldsymbol{V}\cdot\nabla\boldsymbol{V}-(\nabla\times\boldsymbol{B})\times\boldsymbol{B}+\nabla P=0 \qquad \nabla\cdot\boldsymbol{V}=0$$

$$\nabla\times(\boldsymbol{V}\times\boldsymbol{B})+\eta\nabla^2\boldsymbol{B}=0 \qquad \nabla\cdot\boldsymbol{B}=0$$

-> dimensionless MHD equations



Schematic view (Wikipédia)

- First ever PINNs solver for dynamical MHD ?

- 6 scalar PDEs => 6 physics-based partial loss functions
- 5 scalar variables => 5 output neurons
- Dirichlet BCs for ***V*** and ***B*** imposed at boundaries using exact solution

- Parameters used:

$l_r = 2.\,10^{-4}$, $\omega_{data} = \omega_F = 1$, $N_c = 700$, $N_{data} = 120$ (30 per boundary)

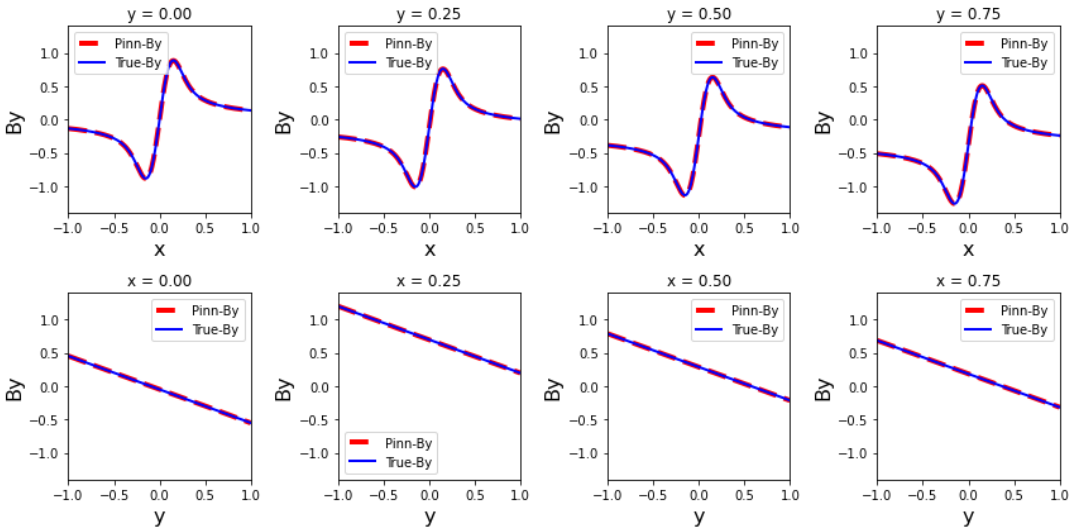
9 hidden layers with 30 neurons/layer -> **7716 parameters** **<-** θ

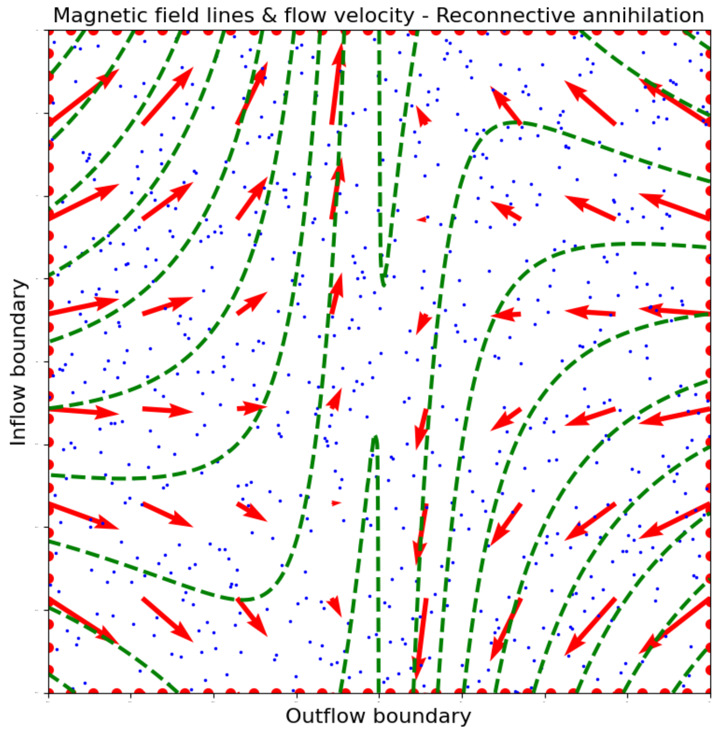
Adam optimizer

Training stopped after 25 000 epochs (40 minutes on a single 8 core CPU)

Application to MHD reconnection

- **Results for 2D steady-state reconnection solution**

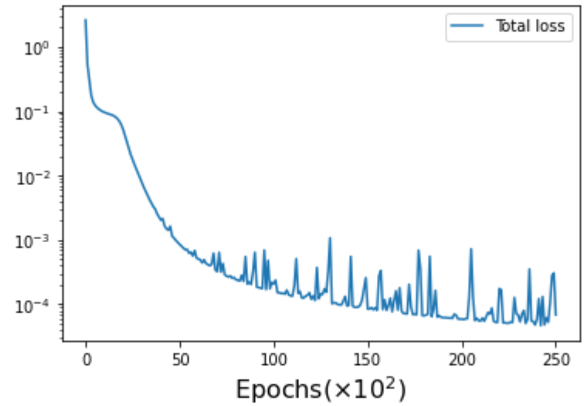




Maximum absolute/relative error is of order 10^{-3}

- Parameters used:

E_d **= 0.1 ,** β **= 0.5 and** α **= 1 ,** η **= 10^{-2}**



- **It works with a reasonable CPU time (less than 1 h)**

the precision: relative maximum error of order 10^{-3}

Conclusions and prospectives

- **PINNs offers a complementary approach & perhaps alternative**

- Drawbacks: -> possible improvements

1. Training can be long/difficult and CPU time consuming = > possible improvements

- GPU acceleration

- Adaptive variants (loss functions with adaptive sampling, optimizers, …)

2. The precision is good/average (not enough for some applications ?) -> 2nd order optim.

under development

- Advantages:

1. Easy to handle and mesh-free

2. Once trained, solutions/derivatives are instantaneously obtained

3. Can be used in different ways: promising complementary approach !

-> Finding unknown physics (sources terms for equilibria) -> inverse problems
in combination with more data

-> Solving multiple solutions (equilibrium, and for reconnection)

under development **see Baty (2023) for ODE's**

Conclusions and prospectives

- **Prospectives**

- Exploit reconnection solver -> reconsider other fast reconnection solutions (see Priest & Forbes book 2000)

- Extend to three dimensional MHD equlibria and dynamics

- Extend to time-dependent dynamics (use of data from traditional solvers ?)

Thank you for your attention

Bibliography

- **PINNs technique to solve PDEs and ODEs**
- Raissi et al. 2019, Journal of Computational Physics, 378, 686
- Lagaris et al. 1998, IEEE transactions on neural networks, 9(5), 987
- Karniadakis et al., Nature reviews, 422, 440
- Baty 2023, Astronomy and Computing 44, 100734
- Baty & Baty 2023 (Solving differential equations using physics informed deep learning: a hand-on tutorial with benchmark tests) 2023arXiv230212260B

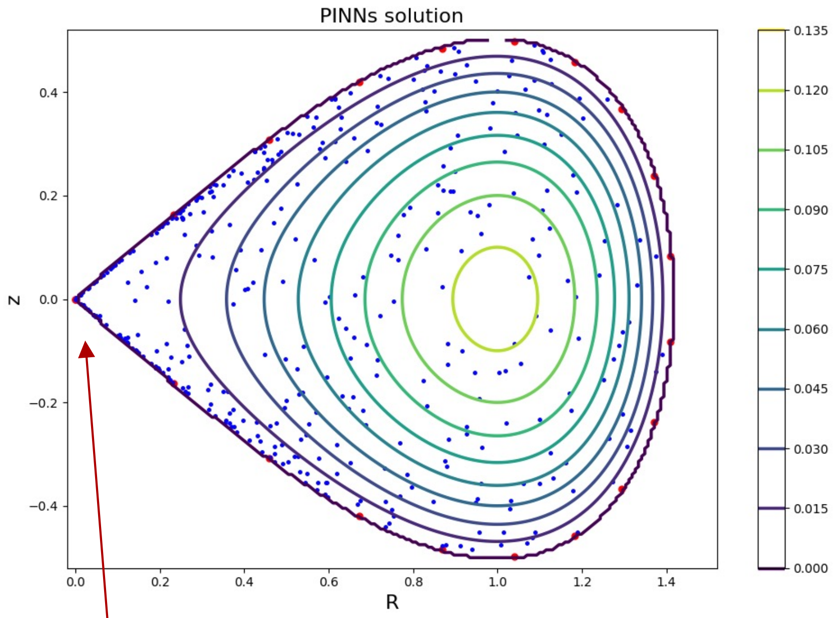
ODEs

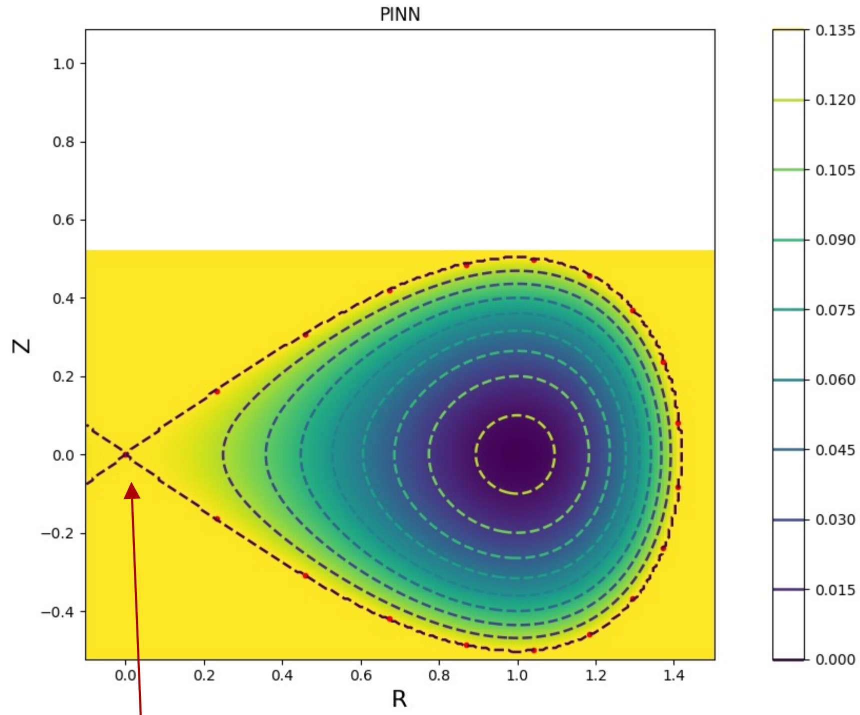
- **Grad-Shafranov equation**
- Deriaz et al. 2011, ESAIM proceedings 32, 76
- Itagaki et al. 2004, Nuclear Fusion 44, 427
- Cerfon and Freidberg 2010, PoP 17, 032502
- Kaltsas and Throumoulopoulos 2022, PoP 29, 022506

- **2D Magnetic reconnection**
- Priest and Forbes 2000, Magnetic Reconnection, book Cambridge University Press
- Craig and Henton 1995, ApJ 450, 280
- Baty and Nishikawa 2016, MNRAS 459, 624

Material presented here is submitted to MNRAS journal (Baty & Vigon 2023)

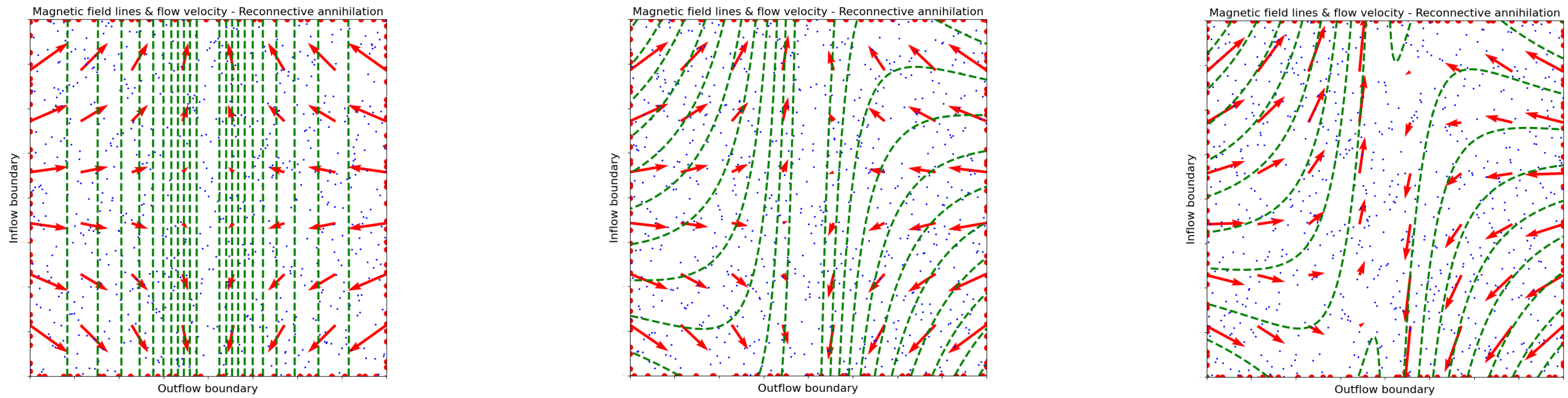
Backup slides (1)



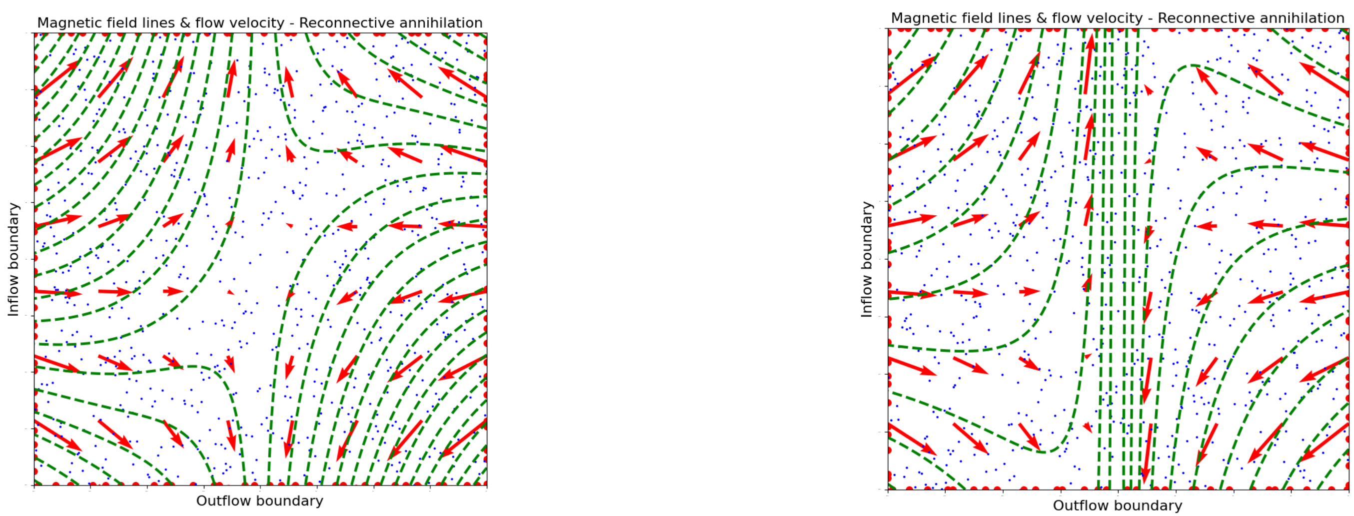


A constraint on zero *x* and *y* first derivative of ψ is added at *X*-point and only 20 Dirichlet training data points are used at boundary

Backup slides (2)



Magnetic reconnection for different β values (0, 0.25, and 0.75) for $\eta = 10^{-2}$



Magnetic reconnection for different resistivity η values (10^{-1} and 10^{-3}) for $\beta = 0.5$