Tokamak 3D Equilibrium Reconstruction

A Deep Learning approach

Lorenzo Rossi

Università di Roma "Tor Vergata"

July 11, 2022

TOR VERGATA
UNIVERSITY OF ROME
School of Engineering

Contents

Introduzione

- L'obiettivo di questa tesina è quello di fornire uno strumento per la ricostruzione in 3D dell'equilibrio del plasma tramite la risoluzione numerica delle PDE caratterizzanti il plasma;
- La ricostruzione dell'equilibrio del plasma è necessaria al miglioramento dell'efficienza fusionistica e alla protezione delle componenti che costituiscono il Tokamak.
- Le equazioni MHD (*MagnetoHydroDynamics*) descrivono la dinamica del Plasma e il suo equilibrio;

Dinamica del plasma

La dinamica del plasma viene descritta da:

- Continuity equation (1): $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \nu) = 0$;
- Momentum equation (3): $\rho \frac{\partial \nu}{\partial t} + \rho(\nu \cdot \nabla)\nu = J \times B - \nabla p$;
- Ideal Ohm's law (3): $E + \nu \times B = 0$;
- Faraday's law (3): $\frac{\partial B}{\partial t} = -\nabla \times E$;
- "Low frequency" Ampere's law (3): $\mu_0 J = \nabla \times B$;
- Magnetic divergence (1): $\nabla \cdot B = 0$;
- Energy (1): $\frac{d}{dt}(\frac{p}{\rho^\gamma})$;

1

1

- $(\cdot) \triangleq \mathit{numerodiequazioni}$

Assumendo che:

- Il plasma si trovi in regime stazionario: $\frac{\partial}{\partial t} = 0$
- Riferimento in v ($\nu = 0$);

Si ottiene:

- Equazione del momento del plasma in equilibrio $J \times B = \nabla p$;
- Legge di Ampere $\nabla \times B = \mu_0 J$
- Equazione della divergenza $\nabla \cdot B = 0$

[2]

---

[2] Dato $f(x, y, z) = f_1 \overrightarrow{i} + f_2 \overrightarrow{j} + f_3 \overrightarrow{k}$ si definiscono:

- Gradiente $\nabla f = \frac{\partial f}{\partial x} \overrightarrow{i} + \frac{\partial f}{\partial x} \overrightarrow{j} + \frac{\partial f}{\partial x} \overrightarrow{k}$;
- Divergenza $\nabla \cdot f = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} + \frac{\partial f_3}{\partial z}$;
- Rotore $\nabla \times f = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{k} & \overrightarrow{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ f_1 & f_2 & f_3 \end{vmatrix}$

Grad-Shafranov

Per giungere infine all'equazione di Grad-Shafranov occorre supporre simmetria toroidale. In particolare:

$$\frac{\partial}{\partial \phi} = 0$$

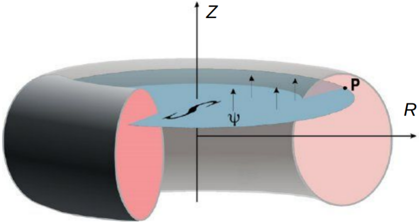
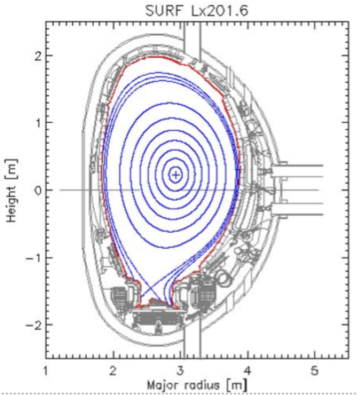


Figure: Simmetria toroidale Tokamak

Equazione di Grad-Shafranov

$$p = f(\psi)$$
$$F = g(\psi)$$
$$\frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r}\frac{\partial \psi}{\partial r} + \frac{\partial^2 \psi}{\partial z^2} = -\mu_0 r^2 \frac{dp}{d\psi} - \frac{1}{2}\frac{dF^2}{d\psi}$$

Limitazioni



Questo metodo consente di ricostruire efficientemente l'equilibrio del plasma. Tuttavia:

- Il plasma non è sempre in regime stazionario;
- La simmetria toroidale non è sempre rispettata.

Una valida alternativa per ottenere più informazioni sul processo in questione viene fornita dal metodo Physics Informed Neural Network basati sul deep learning.

Deep Learning

Deep Learning

Il deep Learning è una branca del Machine Learning che studia l'apprendimento automatico tramite l'utilizzo di architetture stratificate dette **neural network multilayers**.

- L'unità fondamentale di una neural network multilayers è il **neurone** che esegue una singola operazione non lineare.
- Ogni neurone viene connesso con uno o più neuroni a seconda dell'organizzazione della rete;
- L'architettura più utilizzata è di una rete stratificata in feedforward
  - La connessione tra i neuroni avviene solo tra livelli adiacenti
  - Ad ogni connessione si associa un peso
- Utilizzo della **backpropagation**: i pesi associati ad ogni neurone vengono aggiornati calcolando l'uscita della rete e propagando l'errore all'indietro verso i livell più alti

Physics Informed Neural Network

Physics Informed Neural Network

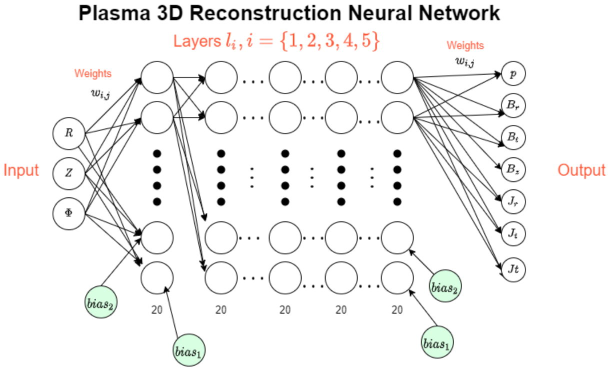
Il Physics Informed Neural Network è un metodo di deep learning basato su reti neurali per risolvere le PDE (*Partial Differential Equation*).

- Permettono di risolvere numericamente equazioni differenziali molto complesse;
- Le soluzioni delle PDE minimizzano una funzione di costo dipendente dalle equazioni fisiche;
- La funzione di costo deve essere ben modellata per aderire al modello preso in considerazione;
- Si necessitano di condizioni al contorno ben strutturate;
- Processo di training elevato;

PDE Neural Network - Struttura

L'architettura della rete neurale utilizzata per risolvere analiticamente le PDE del plasma è la seguente:

- Rete neurale **feedforward fullyconnect**: ogni neurone di ogni livello è collegato con i neuroni del livello successivo;
- Ogni neurone applica all'input la tangentoide $o_j = \frac{e^{net_j} - e^{-net_j}}{e^{net_j} + e^{-net_j}}$;
- Backpropagation tramite di errore;



PDE Neural Network

Input: $R \in [1,61,4.31], Z \in [-2.0250, 2.0250], \phi \in \{0 : 0.5 : 2\pi\}$
Output:$p, B_r, B_z, B_\phi, I_r, I_z, I_\phi$
Weight Factor = $\alpha = \begin{bmatrix} 1 & 0.001 & 0.01 & 1 & 1 & 1 \end{bmatrix}$

$\nabla \cdot B = 0$

$\downarrow$

Loss1=$\frac{1}{r}\frac{\partial r B_r}{\partial r} + \frac{1}{r}\frac{\partial B_\phi}{\partial \phi} + \frac{\partial B_z}{\partial z}$

$\nabla \times B = \mu_0 J$

$\downarrow$

Loss2=$(\frac{1}{r}\frac{\partial B_z}{\partial \phi} - \frac{\partial B_\phi}{\partial z})\mathbf{r} + (\frac{\partial B_r}{\partial z} - \frac{\partial B_z}{\partial r})\phi - \frac{1}{r}(\frac{\partial (r B_\phi)}{\partial r} - \frac{\partial B_r}{\partial \phi})\mathbf{z} - \mu_0 \mathbf{J}$

$J \times B = \nabla p$

$\downarrow$

Loss3=$J_\phi B_z - J_z B_\phi - \frac{\partial p}{\partial R} + J_z B_r - J_r B_z - \frac{\partial p}{\partial \phi} + J_r B_\phi - J_\phi B_r - \frac{\partial p}{\partial Z}$

Vincoli al bordo $p_0, B_{r0}, B_{t0}, B_{Z0}$ noti

$\downarrow$

Loss4=$\frac{mean(p-p_0)^2}{mean(p_0)^2} + \frac{mean(B_z-B_{z0})^2}{mean(B_{z0})^2} + \frac{mean(B_r-B_{r0})^2}{mean(B_{r0})^2} + \frac{mean(B_\phi-B_{\phi0})^2}{mean(B_{\phi0})^2} + \frac{mean(J_z-J_{z0})^2}{mean(J_{z0})^2} + \frac{mean(J_r-J_{r0})^2}{mean(J_{r0})^2} + \frac{mean(J_\phi-J_{\phi0})^2}{mean(J_{\phi0})^2}$

Loss=[Loss1 Loss2 Loss3 Loss4 Loss5 Loss6]*α^T

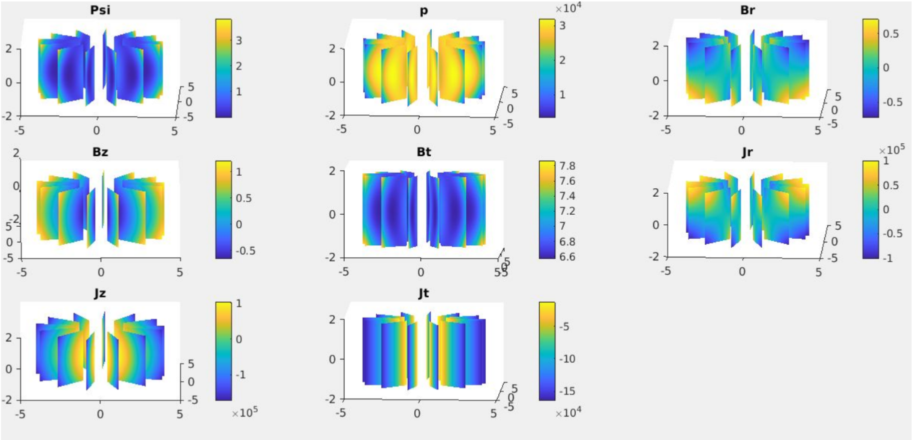
Condizioni al contorno

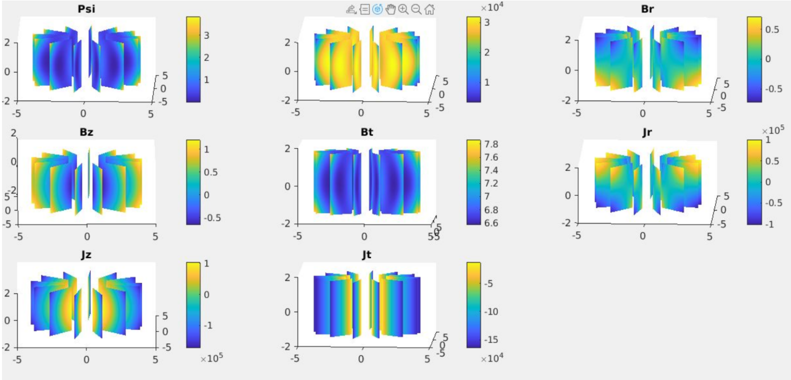
La soluzione delle PDE in $\phi = \{0, 2\pi\}$ potrebbe portare a soluzione diverse quando queste, per periodicità, devono essere identiche. Per evitare questo comportamento, occorre aggiungere una quinda funzione di costo:

$$
\begin{aligned}
Loss5 = {} & \frac{mean(p - p_f)^2}{mean(p_f)^2} + \frac{mean(B_r - B_{r,f})^2}{mean(B_{r,f})^2} + \frac{mean(B_z - B_{z,f})^2}{mean(B_{z,f})^2} + \frac{mean(B_t - B_{t,f})^2}{mean(B_{t,f})^2} + \\
& + \frac{mean(J_z - J_{z,f})^2}{mean(J_{z,f})^2} + \frac{mean(J_r - J_{r,f})^2}{mean(J_{r,f})^2} + \frac{mean(J_\phi - J_{\phi,f})^2}{mean(J_{\phi,f})^2}
\end{aligned}
$$

$$
\begin{aligned}
Loss6 = {} & \frac{mean(p - p_i)^2}{mean(p_i)^2} + \frac{mean(B_r - B_{r,i})^2}{mean(B_{r,i})^2} + \frac{mean(B_z - B_{z,i})^2}{mean(B_{z,i})^2} + \frac{mean(B_t - B_{t,i})^2}{mean(B_{t,i})^2} + \\
& + \frac{mean(J_z - J_{z,i})^2}{mean(J_{z,i})^2} + \frac{mean(J_r - J_{r,i})^2}{mean(J_{r,i})^2} + \frac{mean(J_\phi - J_{\phi,i})^2}{mean(J_{\phi,i})^2}
\end{aligned}
$$

Risultato atteso:





3

Figure: Risultato finale: 10000 epoche (16h di training)

[3] Learning eseguito su una macchina Dell XPS 8940:

- CPU Intel i7-11700 8c/16t @4.8Ghz;
- RAM:32GB DDR4;
- GPU:Nvidia RTX 3070;
- OS:Debian GNU Linux 10 ×86_64;

Considerazioni

- Il risultato ottenuto è congruente alla configurazione reale;
- L'addestramento tramite GPU può abbattere notevolmente il tempo di addestramento;
- Le PDE utilizzate assumono che $\frac{\partial}{\partial \phi} \neq 0$ e quindi la rete neurale potrebbe essere utilizzata anche per configurazioni asimmetriche;
- Una maggiore risoluzione di ϕ può aumentare notevolmente la spesa computazionale.