

Tokamak 3D Equilibrium Reconstruction

A Deep Learning approach

Lorenzo Rossi

Università di Rome "Tor Vergata"

July 10, 2022



Contents

- 1 Introduzione
- 2 Equilibrio del plasma
- 3 Grad-Shafranov
- 4 Deep Learning
- 5 Physics Informed Neural Network
- 6 PDE Neural Network
 - Struttura
 - Errore
- 7 Condizioni al contorno
- 8 Risultati
- 9 Considerazioni

L'obiettivo di questa tesina è quello di fornire una ricostruzione 3D del plasma in forma analitica. Tramite le equazioni MHD (*MagnetoHydroDynamics*) è possibile considerare il plasma come un fluido conduttore soggetto all'azione di un campo magnetico.

Equilibrio

L'equilibrio è quella situazione in cui tutte le forze agenti su di essa hanno risultante nulla.

La ricostruzione dell'equilibrio del plasma è necessaria al miglioramento dell'efficienza fusionistica e alla protezione delle componenti che costituiscono il Tokamak.

Assumendo che:

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

Si ottiene:

- $J \times B = \nabla p$
- $\mu_0 J = \nabla \times B$
- $\nabla \cdot B = 0$

1

¹Dato $f(x, y, z) = f_1 \vec{i} + f_2 \vec{j} + f_3 \vec{k}$ si definiscono:

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

Grad-Shafranov

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

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

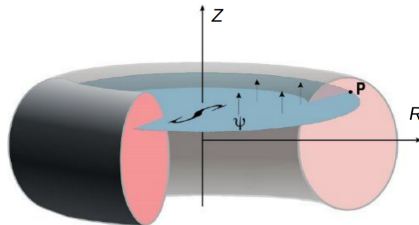


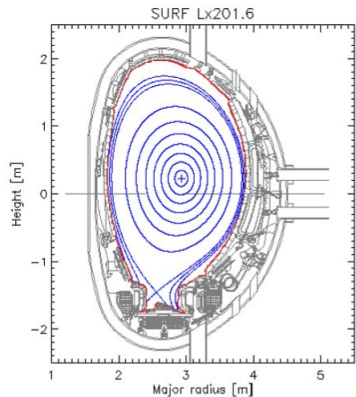
Figure: Simmetria toroidale Tokamak

Equazione di Grad-Shafranov

$$p = f(\psi)$$

$$F = g(\psi)$$

$$\frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r} \frac{\partial \psi}{\partial r} + \frac{\partial^2 \psi}{\partial z^2} = -\mu_0 r^2 \frac{dp}{d\psi} - \frac{1}{2} \frac{dF^2}{d\psi}$$



Sebbene questo metodo consenta 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

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 livelli più alti

Physics Informed Neural Network

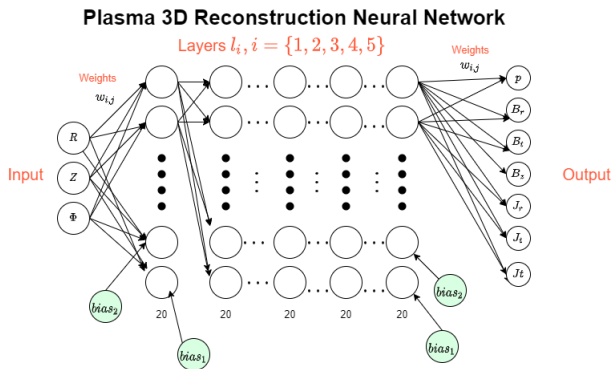
Il Physics Informed Neural Network è un metodo di deep learning basato su reti neurali per risolvere le PDE (*Partial Differential Equation*).

- 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;
- Processo di training elevato;
- Si necessitano di condizioni al contorno;

PDE Neural Network - Struttura

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

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



PDE Neural Network

Input: $R \in [1, 61, 4.31]$, $Z \in [-2.0250, 2.0250]$, $\phi \in \{0 : 0.5 : 2\pi\}$

Output: $p, B_r, B_z, B_\phi, \dot{I}_r, I_z, \dot{I}_\phi$

$$\text{Weight Factor} = \alpha = [1 \quad 0.001 \quad 0.01 \quad 1 \quad 1 \quad 1]$$

$$\nabla \cdot B = 0$$



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

$$\nabla \times B = \mu_0 J$$



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

$$J \times B = \nabla p$$



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

Vincoli al bordo $p_0, B_{r0}, B_{t0}, B_{z0}$ noti



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

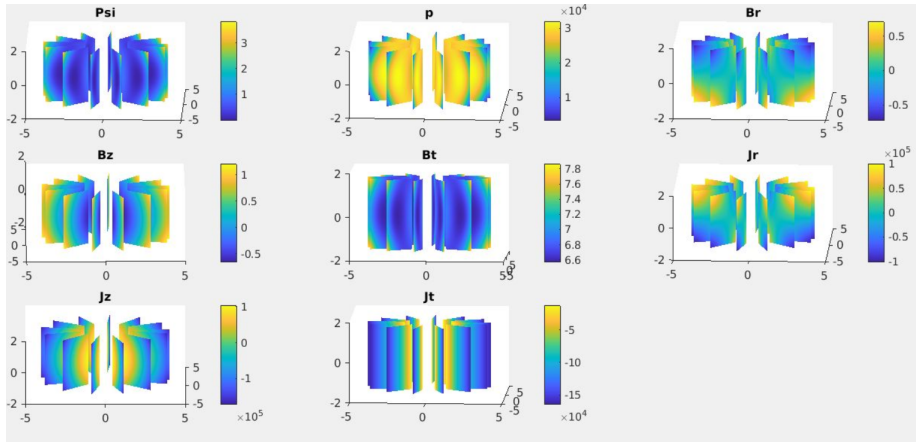
$$\text{Loss} = [\text{Loss1} \text{ Loss2} \text{ Loss3} \text{ Loss4} \text{ Loss5} \text{ Loss6}] * \alpha^T$$

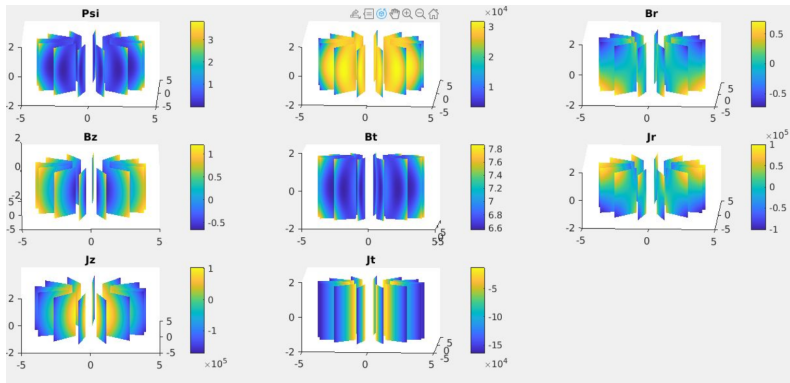
La soluzione delle PDE in $\phi = \{0, 2\pi\}$ potrebbe portare a soluzioni diverse quando queste, per periodicità, devono essere identiche. Per evitare questo comportamento, occorre aggiungere una quinta funzione di costo:

$$Loss5 = \frac{\text{mean}(p - p_f)^2}{\text{mean}(p_f)^2} + \frac{\text{mean}(B_r - B_{r,f})^2}{\text{mean}(B_{r,f})^2} + \frac{\text{mean}(B_z - B_{z,f})^2}{\text{mean}(B_{z,f})^2} + \frac{\text{mean}(B_t - B_{t,f})^2}{\text{mean}(B_{t,f})^2}$$

$$Loss6 = \frac{\text{mean}(p - p_i)^2}{\text{mean}(p_i)^2} + \frac{\text{mean}(B_r - B_{r,i})^2}{\text{mean}(B_{r,i})^2} + \frac{\text{mean}(B_z - B_{z,i})^2}{\text{mean}(B_{z,i})^2} + \frac{\text{mean}(B_t - B_{t,i})^2}{\text{mean}(B_{t,i})^2}$$

Risultato atteso:





2

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

²Learning 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;

- 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.