UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITÁ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale DII

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Physics-Informed Neural Networks applicate alla meccanica computazionale

Relatore: Mirco Zaccariotto

Laureando: Samuele Schiavon
Matricola: 2056504

Anno Accademico 2023/2024

Indice

Elenco delle figure

Elenco delle tabelle

Abbreviazioni

lr learning rate. 9, 11–13, 62

AdaGrad Adaptive Gradient. 10–12

b.c. Boundary Condition. 22, 49, 55

DCNN Deep Convolutional Neural Network. 21

DNN Deep Neural Network. 61

DSV De Saint Venant. iv, vi, 17, 44–46, 48, 50, 54, 62–66, 68–71, 73–79

FEM Finite Element Method. v, 1, 17, 21, 62, 80, 87, 92, 94–96, 98–101, 103, 108–113

GD Gradient Descent. 8, 9, 11

GPU Graphic Processing Unit. 61

MAE Mean Absolute Error. 6

ML Machine Learning. 1, 2, 13, 14, 19, 60

MSE Mean Squared Error. 6, 62

NAG Nesterov Accelerated Gradient. 10

NN Neural Network. vi, 2, 4, 6, 7, 9, 16, 19, 21, 62

PDE Partial Differential Equation. 14, 17, 19

PINN Physics-Informed Neural Network. iii, 14, 15, 20, 21, 47, 62, 68, 112

PyPI Python Package Index. 60

SciML Scientific Machine Learning. iii, 14, 16, 19, 20, 112

SGD Stochastic Gradient Descent. 9, 10, 12

SSE Sum Squared Error. 6

Capitolo 1

Introduzione

Oggigiorno l'intelligenza artificiale è di fondamentale importanza ed è diffusa per moltissime applicazioni, riguardanti l'ingegneria, la medicina, la biologia e molti altri ambiti scientifici. All'interno dell'Intelligenza Artificiale, comunemente conosciuta come AI, esiste una branca che sta guadagnando successo e importanza: si tratta del Machine Learning (ML). Il ML consiste nel fornire al computer stesso la capacità di comprendere le relazioni nascoste dietro ad un determinato problema, rendendolo quindi in grado di collegare tra loro i dati forniti alla macchina e di migliorarne la conoscenza senza la necessità di una persona fisica che programmi il computer. Il ML è spesso utilizzato nelle auto a guida autonoma, nel riconoscimento di immagini e video, nel riconoscimento e produzione vocale, negli avvisi per auto e traffico tramite navigatore, nella diagnosi medica, nella sicurezza dei dispositivi elettronici, ed è utilizzato anche negli sport con tecnologie come il fuorigioco semiautomatico e la tecnologia della linea di porta utilizzata durante le partite di calcio, ecc.

L'esplosione dell'utilizzo di questo fondamentale strumento è principalmente dovuto a 3 motivi: il primo è che ad oggi possiamo dire di possedere una quantità di dati tale da allenare in maniera opportuna una rete neurale per un determinato scopo; in secondo luogo l'incremento della potenza di calcolo dei microprocessori e l'introduzione di schede grafiche con accelerazioni software dedicate risultano di fondamentale importanza per consentire a questo strumento di performare in tempi modesti; ultimo, ma non per importanza, lo sviluppo da parte di diverse compagnie di librerie e software contenenti framework "maturi" per deep learning e algoritmi ottimizzati per l'allenamento delle reti neurali.

In questo elaborato verranno prese in esame alcune strategie per l'uso del Machine Learning alla meccanica computazionale. L'obiettivo dello studio è quello di creare modelli surrogati tramite l'utilizzo di reti neurali in modo fornire un'alternativa ai software standard utilizzati per le simulazioni FEM. Questo è dovuto ai prolungati tempi di risposta che questi software hanno, a differenza del tempo di risposta di una rete neurale artificiale. Infatti, a training effettuato, il vantaggio consiste nel ridotto tempo di risposta che presenta una rete neurale rispetto ad un'analisi FEM tradizionale.

Capitolo 2

Reti neurali

La Rete Neurale, o Neural Network (NN), è propriamente l'oggetto informatico che viene utilizzato nel Machine Learning (ML). Una NN è sostanzialmente una funzione matematica che presenta una serie di parametri liberi chiamati pesi e bias (Fig. 2.1). Questa funzione, come ogni funzione scalare o vettoriale esistente, riceve una o più variabili in input e fornisce uno o più output, il tutto secondo le funzioni di attivazione presenti al suo interno ed alla struttura dei vari layer che la compongono. Per una migliore comprensione dello strumento in questione è bene definire i concetti che stanno alla base della sua composizione e funzionamento.

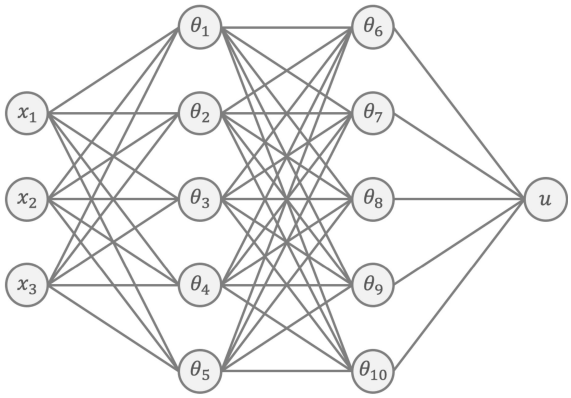


Figura 2.1: Esempio di una rete neurale [34]

2.1 Layer

Per avere bene un quadro generale di come si compone una rete neurale è necessario introdurre la sua struttura. La struttura di una rete neurale è composta di layer, o strati, composti ciascuno da uno o più neuroni. Vi sono 3 tipologie di layer: *input layer*, *hidden layer* ed *output layer* (Fig. 2.2).

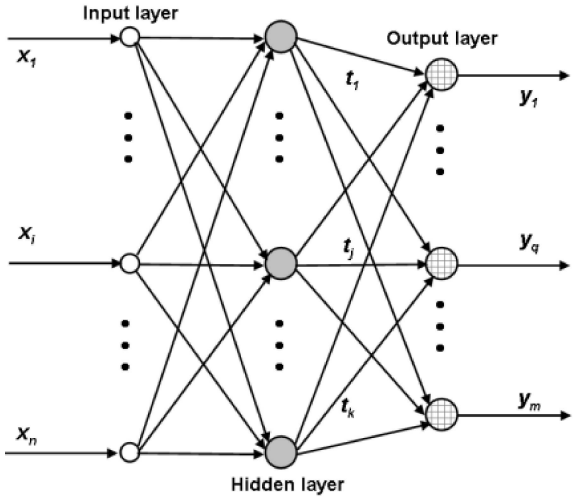


Figura 2.2: Rappresentazione dei differenti tipi di layers [27]

L'*input layer* presenta un numero di neuroni pari alle variabili di input della funzione che la rete neurale deve rappresentare, oppure un numero pari alle caratteristiche di un problema che devono essere legate tra di loro; questo layer è differente dagli altri in quanto i neuroni che vi sono presenti sono sprovvisti di parametri (pesi e bias) e di una funzione di attivazione. Non vi è eseguita nessuna operazione numerica, il che implica che il valore di output è pari al valore di input stesso.

Un *hidden layer* invece presenta un numero di neuroni arbitrario, scelto dal progettista in base alla complessità del problema da risolvere. I neuroni di un singolo *hidden layer* presentano la medesima funzione di attivazione, ma ognuno è caratterizzato dal proprio bias e dal proprio set di pesi che lo collega agli output del layer precedente. L'input di ogni neurone è dato da una combinazione lineare dei valori di output dei neuroni del layer precedente, ove i coefficienti moltiplicativi sono rappresentati dai pesi. A questo input viene sommato un bias, caratteristico del neurone: su questa somma è poi eseguita la funzione di attivazione che ne determina il valore di output. I valori calcolati dai neuroni di un hidden layer sono poi passati ai neuroni del layer successivo e così via. Una rete neurale presenta una quantità di hidden layer determinata dalla complessità del problema e dalle caratteristiche intrinseche che la rete deve riuscire a rappresentare.

Infine, l'*output layer* presenta un numero di neuroni pari al numero di output che la rete neurale deve rappresentare. Il comportamento dei neuroni in questo layer è il medesimo di quelli presenti in un *hidden layer*, anche se solitamente presentano un comportamento lineare e sono quindi sprovvisti di una funzione di attivazione.

2.2 Neurone

Il neurone è l'elemento di base di una rete. Pensato ad imitare il comportamento di un neurone biologico, questo riceve degli input numerici e tramite la propria funzione di attivazione produce un output. Un generico neurone può essere espresso come

$$x_j^{n+1} = \phi\left(b_j^n + \sum_{i=1}^{k_n} w_{ji}^n \cdot x_i^n\right) \tag{2.1}$$

che rappresenta il *j-esimo* neurone al *n-esimo* layer [27]. x_j^{n+1} rappresenta l'uscita del neurone che, a sua volta, è un ingresso dei neuroni presenti al layer $n+1$. Il termine x_i^n rappresenta i valori di ingresso ai neuroni del *n-esimo* layer (ovvero i valori in uscita del $(n-1)$-*esimo* layer; l'*input layer* corrisponde a $n = 0$), w e b sono i pesi e bias della NN, ϕ è la cosiddetta *funzione di attivazione* e k_n è il numero di neuroni presenti al *n-esimo* layer.

L'intero *n-esimo* layer può essere rappresentato da un'equazione matriciale:

$$\mathbf{X}^{n+1} = \phi\left(\mathbf{B}^n + \mathbf{A}^n \cdot \mathbf{X}^n\right) \tag{2.2}$$

dove per $n = 0$ ci si riferisce a $X^0 = X$ il vettore di input della NN, mentre per $n = m$ dove m è il numero dell'ultimo layer ci si riferisce a $X^m = Y$ il vettore di output della rete [27].

I neuroni si differenziano in lineari e non lineari, a seconda della presenza di una funzione di attivazione al loro interno (Fig. 2.3).

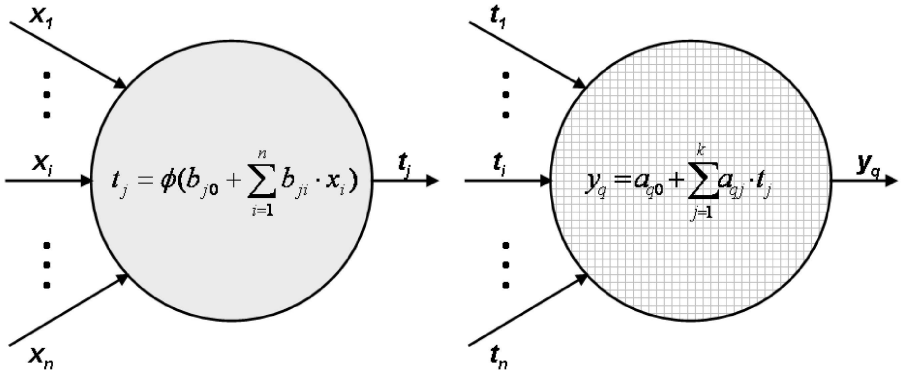


Figura 2.3: Differenze tra un neurone non lineare (sinistra) ed un neurone lineare (destra) [27]

Seguendo le notazioni utilizzate si può notare come solitamente i neuroni non lineari siano utilizzati all'interno della rete, mentre quelli lineari per rappresentare gli output.

2.3 Funzione di attivazione

La funzione di attivazione è una funzione matematica propria del neurone e ne caratterizza il comportamento. Questa funzione determina il valore dell'output e introduce la non linearità nell'intera rete neurale. Vi sono diverse funzioni di attivazione attualmente in uso:

Nome	**Funzione**, g	**Derivata,** g'
Identity	x	1
Rectified Linear Unit (ReLU)	$\begin{cases} 0, \text{ if x} \leq 0 \\ \text{x, if x} > 0 \end{cases}$	$\begin{cases} 0, \text{ if x} \leq 0 \\ 1, \text{ if x} > 0 \end{cases}$
Leaky ReLU	$\begin{cases} 0.01\,x, \text{ if } x < 0 \\ x, \text{ if } x \geq 0 \end{cases}$	$\begin{cases} 0.01, \text{ if } x < 0 \\ 1, \text{ if } x \geq 0 \end{cases}$
Parametric ReLU (PReLU)	$\begin{cases} \alpha\,x, \text{ if } x < 0 \\ x, \text{ if } x \geq 0 \end{cases}$	$\begin{cases} \alpha, \text{ if } x < 0 \\ 1, \text{ if } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)	$\begin{cases} e^x - 1, \text{ if } x < 0 \\ x, \text{ if } x \geq 0 \end{cases}$	$\begin{cases} e^x, \text{ if } x < 0 \\ 1, \text{ if } x \geq 0 \end{cases}$
Scaled ELU (SELU)	$\lambda \begin{cases} \alpha(e^x - 1), \text{ if } x \leq 0 \\ x, \text{ if } x > 0 \end{cases}$	$\lambda \begin{cases} \alpha\,e^x, \text{ if } x \leq 0 \\ 1, \text{ if } x > 0 \end{cases}$
Sigmoid	$\dfrac{1}{1+e^{-x}}$	$\dfrac{e^{-x}}{(1+e^{-x})^2}$
Sigmoid Linear Unit (SiLU)	$\dfrac{x}{1+e^{-x}}$	$\dfrac{1+e^{-x}+xe^{-x}}{(1+e^{-x})^2}$
Hyperbolic tangent (tanh)	$\dfrac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - g^2$
Softplus	$\ln(1+e^x)$	$\dfrac{1}{1+e^{-x}}$

Tabella 2.1: Lista di funzioni di attivazione attualmente utilizzate [40, 35, 17]

2.4 Pesi

I pesi caratterizzano ogni collegamento tra neuroni della rete. Ad ogni neurone è associato un peso per ogni valore di input che possiede. Questi pesi saranno i parametri che verranno aggiornati nella fase di training, determinando così l'importanza di un input rispetto ad un altro [27].

2.5 Bias

Ogni neurone possiede un parametro caratteristico chiamato bias, o valore di soglia, che, in poche parole, determina se e in quale misura il neurone debba attivarsi. Ogni neurone riceve, dal precedente layer, una somma di input 'pesati' ai quali va sommato un bias: su questa somma poi va eseguita la funzione di attivazione del neurone [27].

2.6 Loss function

Una volta calcolati gli output su dei valori di input di test, si calcola il costo o perdita che misura la differenza tra il valore ottenuto e il valore atteso. Queste differenze vengono gestite tramite una *loss function*, o funzione di perdita o di costo, che misura l'errore della rete, ovvero quanto una rete predice accuratamente i risultati. L'obiettivo della fase di training è quello di minimizzare la loss function, così da minimizzare la differenza tra un valore calcolato dalla rete e quello atteso. Esistono diversi tipi di loss function che descrivono l'accuratezza di una rete:

Loss function	**Definizione**
Mean Absolute Error (MAE)	$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$
Mean Squared Error (MSE)	$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
Sum Squared Error (SSE)	$\mathcal{L} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
Huber Loss	$\mathcal{L} = \begin{cases} 0.5\,(y_i - \hat{y}_i)^2, \text{ if } \lvert y_i - \hat{y}_i \rvert \leq \delta \\ \delta\,(\lvert y_i - \hat{y}_i \rvert - 0.5\,\delta), \text{ if } \lvert y_i - \hat{y}_i \rvert > \delta \end{cases}$

Tabella 2.2: Diversi modi con cui misurare il costo, o perdita, di una previsione. Con a disposizione n campioni, y_i è il valore di output dalla NN mentre $\hat{y}_i$ identifica il valore atteso. Il valore δ si riferisce ad un parametro arbitrario per la Huber Loss

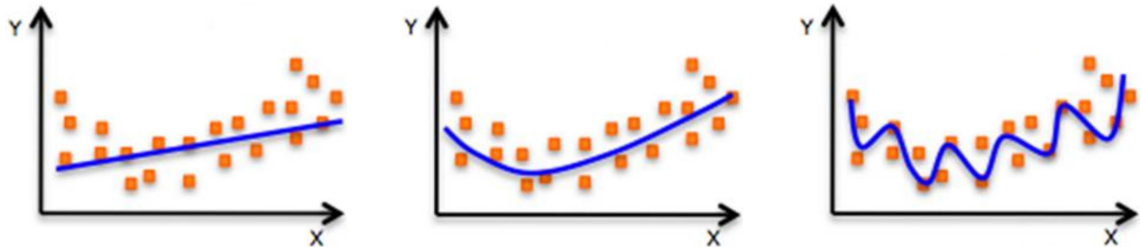
Esistono poi delle loss functions di tipo Cross-Entropy [35] che riguardano però problemi di categorizzazione che non verranno trattati in questo studio. Mentre le definizioni rappresentate in Tab. 2.2 confrontano direttamente i risultati predetti e previsti, nelle funzioni di perdita di tipo Cross-Entropy viene valutata la probabilità di distribuzione del valore atteso e predetto: questo è possibile solo se si conoscono a priori le classi o categorie sulla quale la NN deve allenarsi.

2.7 Training

Il training, o apprendimento, di una rete neurale consiste nell'aggiornare i pesi e i bias secondo un *optimizer*, o algoritmo di apprendimento, che opera sulla funzione di perdita cercandovi il valore minimo, consentendo quindi alla NN di comprendere il problema che stiamo trattando. Il processo con il quale vengono aggiornati i pesi e bias secondo un algoritmo di apprendimento si chiama *backpropagation*, poichè è un processo che parte dai risultati e ripercorre "all'indietro" tutta la rete aggiornando i parametri presenti.

In questa fase viene utilizzato un *training dataset*, o set di dati, con il quale la rete potrà allenarsi, provvisto di input che vengono forniti alla rete neurale e di output attesi da quest'ultima. Il training è un processo iterativo e si compone di un numero di arbitrario di cicli ripetuti: una rete neurale non può essere modellata dal training dataset in un unico aggiornamento dei dati. Per comprendere meglio la fase di training è bene introdurre due iperparametri fondamentali che la caratterizzano: il *batch size* e il numero di *epochs*. Un training dataset può essere suddiviso in più *batch*, o gruppi di campioni di dati, la cui dimensione è definita dal batch size. La rete viene quindi eseguita su un singolo batch, accumulando così l'errore di più campioni di dati; sull'errore accumulato viene poi eseguito l'optimizer e vengono aggiornati i parametri della rete. Successivamente si prosegue al batch successivo. Il numero di *epochs* definisce il numero di volte in cui l'algoritmo di apprendimento lavorerà attraverso l'intero set di dati di addestramento, ovvero avrà terminato i batch su cui eseguire la NN. Questo implica che il termine di una *epoch* definisce che ogni campione presente nel set di dati ha avuto la possibilità contribuire all'aggiornamento dei parametri del modello. Il processo di training continua poi per un numero indefinito di epochs e si interrompe imponendo una condizione sull'errore oppure limitando il numero massimo di epochs eseguibili. La differenza tra i due iperparametri è che il batch size corrisponde al numero di campioni del dataset su cui viene eseguita la rete neurale prima dell'aggiornamento dei parametri del modello, mentre il numero di *epochs* è il numero di volte in cui la rete neurale è stata eseguita sull'intero dataset. Considerando m campioni nel training dataset, il batch size (b_s) deve essere $1 \leq b_s \leq m$, mentre il numero di epochs (t) può essere $1 \leq t < \infty$.

Nonostante un numero molto elevato di epochs possa portare la funzione di perdita a tendere allo zero, un problema che si può verificare nella fase di training è l'overfitting (Fig. 2.4).



(a) (b) (c)

Figura 2.4: Differenza di un caso di underfitting (a), right fitting (b) e overfitting (c) [27]

Questo problema consiste nell'eseguire l'addestramento per un numero troppo elevato di epochs, particolarizzando la rete rispetto al training dataset piuttosto che sulle caratteristiche generali che questo rappresenta. Durante questa fase è bene quindi studiare l'andamento della funzione di perdita anche rispetto ad un testing dataset, per costruzione identico ad un training dataset ma con valori diversi. Il suo scopo è quello di valutare le performance della rete senza che i risultati calcolati vengano utilizzati per allenarla. Solitamente il training e il testing dataset vengono estratti da un insieme più grande di campioni disponibili per l'addestramento di una rete. Una volta che, durante l'addestramento, l'errore rispetto al testing dataset inizia a mantenersi costante o addirittura a crescere rispetto all'errore del training dataset, ha inizio il fenomeno di overfitting.

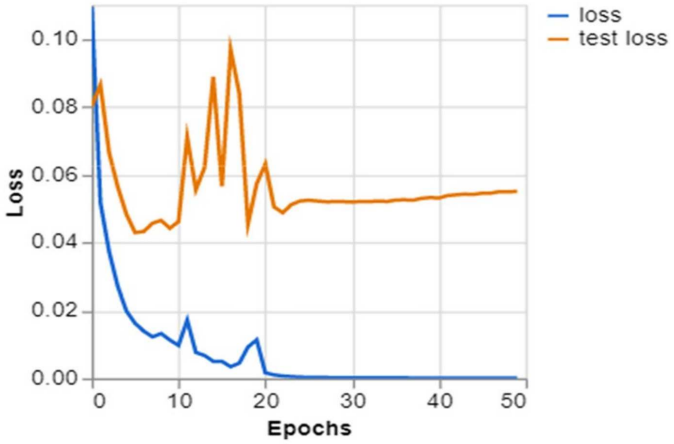


Figura 2.5: Confronto tra il costo del training con quello di un testing dataset [1]

2.7.1 Gradient Descent

Il Gradient Descent (GD) è un algoritmo iterativo del primo ordine pensato per trovare il minimo di una funzione multidimensionale differenziabile. L'idea è quella di fare ripetuti step nella direzione opposta al gradiente della loss function calcolata nel punto corrente, poichè questa è la direzione della discesa più ripida. Il gradiente della funzione di perdita viene rappresentato come

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \mathbf{x}, \mathbf{y}) \tag{2.3}$$

ove il vettore θ_t rappresenta i parametri della rete all'iterazione t, mentre i vettori $\mathbf{x}$ e $\mathbf{y}$, presenti nell'Eq. 2.3, rappresentano gli input e output forniti dal dataset. Il pedice t è il contatore delle iterazioni durante il training, ovvero le epoch, il che significa che per $t = 0$ ci riferiamo alle condizioni di partenza (solitamente random). Il Gradient Descent è l'algoritmo di ottimizzazione più semplice ma altresì più utilizzato, tanto che il suo funzionamento sta alla base di ogni optimizer esistente. L'algoritmo nella sua semplicità risulta

$$\begin{aligned} \mathbf{g}_t &= \nabla \mathcal{L}(\boldsymbol{\theta}_t) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta\, \mathbf{g}_t \end{aligned} \tag{2.4}$$

dove il parametro η rappresenta il learning rate (lr), o tasso di apprendimento, oppure step size, ed è arbitrario. Questo parametro definisce la velocità con la quale si vuole far allenare la NN, con il rischio che, se troppo elevato, può portare a non convergere verso un minimo. I vettori $\mathbf{x}$ e $\mathbf{y}$ rappresentano il batch di dati estratto dal training dataset sul quale viene eseguita la rete neurale. Considerando un dataset composto di m campioni, in base al batch size (b_s) scelto l'algoritmo viene denominato differentemente: quando $b_s = 1$ si parla di Stochastic Gradient Descent (SGD); quando $b_s = m$ di parla semplicemente di Gradient Descent (GD), o di Batch Gradient Descent; quando $1 < b_s < m$ si parla di Mini-Batch Gradient Descent.

Nonostante la sua semplicità, il Gradient Descent presenta alcune sfide da affrontare:

- É opportuno scegliere un valore ottimale del lr. Se questo è troppo piccolo, il training può richiedere molto tempo per convergere; al contrario se troppo elevato l'algoritmo può entrare in stallo e non convergere ad un minimo;
- É opportuno scegliere un batch size adeguato. Se questo è troppo elevato, e il training dataset presenta un gran numero di campioni, risulta necessaria una grande memoria per il calcolo; al contrario se troppo piccolo vi è un'elevata varianza nei parametri del modello (problema del SGD);
- La presenza di un tasso di apprendimento costante per tutti i parametri è un fattore limitante. Vi possono essere parametri che richiedono diversi learning rate, che possono altresì variare durante la fase di apprendimento;
- La semplicità di questi algoritmi può far rimanere "intrappolato" il training nei minimi locali.

2.7.2 Momentum

Una semplice euristica, chiamato metodo dello slancio, o meglio conosciuto come *momentum method*, consiste nel muoversi più velocemente lungo direzioni del gradiente precedentemente ritenute valide e nel rallentare lungo le direzioni in cui la pendenza ha subito un cambio improvviso. Questo metodo è stato pensato come implementazione dell'algoritmo SGD, con l'obiettivo di ridurne l'elevata varianza che presenta. L'algoritmo di SGD con il metodo del *momentum* (dall'inglese, "Quantità di moto") permette di tenere in memoria l'aggiornamento dei parametri della rete $\Delta\theta_t = \theta_t - \theta_{t-1}$ ad ogni iterazione e di determinarne il nuovo valore come una combinazione lineare dell'attuale gradiente e il precedente aggiornamento. In questo metodo viene utilizzato un ulteriore iperparametro α: è un fattore di decadimento esponenziale compreso tra 0 e 1 che determina il contributo relativo del gradiente corrente e dei gradienti precedenti alla variazione dei parametri [42]. Il valore $\alpha = 0.9$ è ampiamente utilizzato [36], perciò ci si riferirà principalmente a questo valore se non diversamente specificato. L'algoritmo si presenta così:

$$\begin{aligned} \Delta\boldsymbol{\theta}_t &= \alpha\,\Delta\boldsymbol{\theta}_{t-1} - \eta\,\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_t), \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}_t \end{aligned} \tag{2.5}$$

Questo accelera la convergenza verso la direzione rilevante e riduce la fluttuazione verso la direzione irrilevante, trascurando anche il rumore presente nel dataset (composto per esempio di misurazioni). Il nome momentum deriva da un'analogia con la quantità di moto in fisica: il vettore dei parametri θ, pensato come una particella che viaggia attraverso lo spazio dei parametri, subisce un'accelerazione dal gradiente della perdita ("forza") [42]. A differenza del classico SGD, questa "particella" tende a continuare a viaggiare nella stessa direzione, evitando oscillazioni.

Pro:

- Riduce le oscillazioni e l'elevata varianza dei parametri
- Converge più velocemente del SGD

Contro:

- Viene aggiunto un ulteriore iperparametro che deve essere selezionato manualmente e con precisione

2.7.3 Nesterov Accelerated Gradient

Nel caso in cui il contributo del momentum fosse troppo elevato, l'algoritmo potrebbe non raggiungere i minimi locali e continuare ad oscillarvi intorno. Quindi, per risolvere questo problema, è stato sviluppato l'algoritmo Nesterov Accelerated Gradient (NAG), definito come metodo di visione anticipata. Dalla definizione dell'algoritmo basato sul momentum si sa che verrà utilizzato il termine $\alpha\,\Delta\boldsymbol{\theta}_{t-1}$ per modificare i parametri, quindi la quantità $\boldsymbol{\theta}_t + \alpha\,\Delta\boldsymbol{\theta}_{t-1}$ dice in maniera approssimativa i valori futuri. Così viene calcolata la loss function in base ai parametri futuri anziché a quelli attuali [8]:

$$\begin{aligned} \Delta\boldsymbol{\theta}_t &= \alpha\,\Delta\boldsymbol{\theta}_{t-1} - \eta\,\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_t + \alpha\,\Delta\boldsymbol{\theta}_{t-1}), \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}_t \end{aligned} \tag{2.6}$$

2.7.4 AdaGrad method

Adaptive Gradient (AdaGrad) è una famiglia di algoritmi che si basa sul SDG [16]. Questo metodo adatta il tasso di apprendimento tramite il momentum ad ogni iterazione t, come per il NAG, ma lo fa per ciascun parametro individualmente, utilizzando l'intera sequenza delle stime del gradiente calcolate fino quell'iterazione. Questo avviene tramite la sommatoria dei prodotti di Hadamard [3], ovvero della moltiplicazione elemento per elemento, del vettore gradiente $\mathbf{g}$ con se stesso, vale a dire tramite il vettore $\mathbf{v}_t$ definito come

$$\mathbf{v}_t = \sum_{\tau=1}^{t} \mathbf{g}_\tau \odot \mathbf{g}_\tau \tag{2.7}$$

dove $\mathbf{g}_\tau = \boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_\tau)$ è il gradiente della funzione di perdita all'iterazione τ. La moltiplicazione di Hadamard di un vettore con se stesso viene anche rappresentata ponendo il vettore al quadrato, quindi in questo caso con $\mathbf{g}_t^2$

Di conseguenza il vettore $\mathbf{v}_t$ è aggiornato ad ogni iterazione e l'espressione di aggiornamento dei parametri assume la forma

$$\begin{aligned} \mathbf{v}_t &= \mathbf{v}_{t-1} + \mathbf{g}_t^2 \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta \, \frac{\mathbf{g}_t}{\sqrt{\mathbf{v}_t} + \varepsilon} \end{aligned} \tag{2.8}$$

dove ε è un termine di smoothing, solitamente con un valore compreso tra 10^{-8} e 10^{-10}, aggiunto per migliorare la stabilità numerica e per evitare problemi di singolarità.

Dall'Eq. 2.8 è evidente che la regola di aggiornamento per AdaGrad adatta il learning rate per ogni j-esimo parametro $\eta \, (\sqrt{v_{t,j}} + \varepsilon)^{-1}$, mentre i metodi standard del tipo GD hanno un *lr* fisso pari a η.

2.7.5 AdaDelta method

Mentre tutti gli algoritmi presentati fin'ora usano unicamente metodi del primo ordine, come il gradiente, per minimizzare la funzione di perdita, i metodi del secondo ordine, come ad esempio il metodo di Newton, fanno uso della matrice Hessiana o di approssimazioni ad essa [43]. Sebbene ciò fornisca ulteriori informazioni sulla curvatura della funzione di perdita utili per l'ottimizzazione, il calcolo di informazioni di secondo ordine è spesso oneroso dal punto di vista computazionale. Dato questo inconveniente, per modelli di grandi dimensioni è stata proposta un'approssimazione diagonale della matrice Hessiana [5]. Una volta calcolata la diagonale dell'Hessiana, diag($\mathbf{H}$), l'aggiornamento dei parametri assume la forma

$$x_{t+1} = x_t - \frac{g_t}{|\text{diag}(\mathbf{H}_t)| + \varepsilon} \tag{2.9}$$

dove x sono i parametri di una generica funzione f e $g_t = \nabla f$. L'algoritmo di AdaDelta è basato sul funzionamento di AdaGrad [47], creato per migliorarne i due principali inconvenienti: il continuo decadimento dei learning rate durante il training e la necessità di un *lr* globale selezionato manualmente. Nell'algoritmo AdaGrad, poiché ogni termine del vettore $\mathbf{v}_t$ è positivo (Eq. 2.7), questa somma accumulata continua ad aumentare iterazione dopo iterazione, riducendo di fatto il *lr* per ciascun parametro. Dopo molte iterazioni, il learning rate può risultare infinitesimamente piccolo: sotto questo punto di vista si differenzia AdaDelta. Al posto di accumulare la somma dei quadrati dei gradienti nel corso di tutte le iterazioni, considera una media ponderata esponenziale della seguente forma

$$\mathbf{v}_t = \rho \, \mathbf{v}_{t-1} + (1 - \rho) \, \mathbf{g}_t^2 \tag{2.10}$$

dove ρ è un iperparametro noto come tasso di decadimento, simile ad α utilizzato nel metodo del momentum, che controlla l'importanza delle osservazioni passate del gradiente rispetto a quelle attuali. Le scelte tipiche per il tasso di decadimento ρ sono 0.95 o 0.90, che sono le scelte predefinite per l'ottimizzatore AdaDelta in TensorFlow [44] e PyTorch [14] rispettivamente. Vi è poi un'ulteriore differenza: nello sviluppo

dell'algoritmo si è riflettuto sugli aggiornamenti, Δx, che questo porta ai parametri, x, ed al confronto tra le unità di misura dei due. Per intendersi, le unità di misura (*udm*) degli aggiornamenti portati da algoritmi come SGD, momentum e AdaGrad si relazionano al gradiente anziché ai parametri stessi

$$udm\ \Delta x \propto udm\ g \propto \frac{\partial f}{\partial x} \propto \frac{1}{udm\ x} \tag{2.11}$$

assumendo che la funzione f sia priva di unità di misura. D'altra parte, con metodi del secondo ordine, come il metodo di Newton che utilizza le informazioni della matrice Hessiana, si hanno le unità di misura corrette per gli aggiornamenti dei parametri. Riferendosi all'Eq. 2.9,

$$\Delta x \propto H^{-1} g \propto \frac{\frac{\partial f}{\partial x}}{\frac{\partial^2 f}{\partial x^2}} \propto udm\ x, \tag{2.12}$$

si ottiene una corrispondenza [47] ed è proprio da questa intuizione che AdaDelta assume connotati di un algoritmo del secondo ordine. Questo viene fatto stimando i valori di una matrice Hessiana tramite l'Eq. 2.9, ottenendo

$$\Delta x = \frac{\frac{\partial f}{\partial x}}{\frac{\partial^2 f}{\partial x^2}} \quad \Longrightarrow \quad \frac{1}{\frac{\partial^2 f}{\partial x^2}} = \frac{\Delta x}{\frac{\partial f}{\partial x}}. \tag{2.13}$$

Quindi, basandosi sulle Eq.2.12 e 2.13, la componente del secondo ordine viene costruita utilizzando un vettore $\mathbf{u}_t$ che media gli aggiornamenti dei parametri in maniera analoga al vettore $\mathbf{v}_t$. Ad ogni step, la forma che assume l'algoritmo è la seguente

$$\begin{aligned} \mathbf{v}_t &= \rho\, \mathbf{v}_{t-1} + (1-\rho)\, \mathbf{g}_t^2 \\ \Delta \boldsymbol{\theta}_t &= -\frac{\sqrt{\mathbf{u}_{t-1} + \varepsilon}}{\sqrt{\mathbf{v}_t + \varepsilon}}\, \mathbf{g}_t \\ \mathbf{u}_t &= \rho\, \mathbf{u}_{t-1} + (1-\rho)\, \Delta \boldsymbol{\theta}_t^2 \end{aligned} \tag{2.14}$$

dove poi

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\Delta \theta}_t \tag{2.15}$$

2.7.6 Adam method

L'algoritmo Adam deriva da *adaptive moment estimation* ed è stato pensato per combinare i vantaggi di AdaGrad e di RMSprop, ovvero un altro algoritmo molto diffuso ma mai ufficialmente pubblicato [15]. La nota comune risiede nell'utilizzo del quadrato del gradiente, usufruendo di parametri che scalano il learning rate per ogni parametro ad ogni iterazione [26]

$$\begin{aligned}
\mathbf{m}_t &= \beta_1\, \mathbf{m}_{t-1} + (1-\beta_1)\, \mathbf{g}_t, \\
\mathbf{v}_t &= \beta_2\, \mathbf{v}_{t-1} + (1-\beta_2)\, \mathbf{g}_t^2, \\
\hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1-\beta_1^t}, \\
\hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1-\beta_2^t}.
\end{aligned} \tag{2.16}$$

Le medie mobili pesate del gradiente, $\mathbf{m}_t$, e del suo quadrato, $\mathbf{v}_t$, controllano i tassi di decadimento delle medie stesse tramite l'utilizzo di iperparametri $\beta_1,\ \beta_2 \in [0, 1)$. Tramite i vettori appena presentati l'aggiornamento dei parametri dipende da un learning rate globale

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\, \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon} \tag{2.17}$$

Il metodo è semplice da implementare e richiede poca memoria. Nel complesso, si è riscontrato che Adam è robusto e adatto a un'ampia gamma di problemi di ottimizzazione nel campo del ML [26].

Capitolo 3

Scientific Machine Learning

Il campo del ML scientifico, o Scientific Machine Learning (SciML) (Fig. 3.1), è cresciuto rapidamente negli ultimi anni grazie alla ricerca che viene condotta in una vasta gamma di ambiti scientifici, come per esempio nella scienza del clima [41], nella meccanica dei fluidi [23] e nella biologia [24]. Data la novità dello strumento in studio, sono sempre più elevate le aspettative che lo SciML acceleri la scoperta scientifica e affronti le più grandi sfide dell'umanità [30, 4].

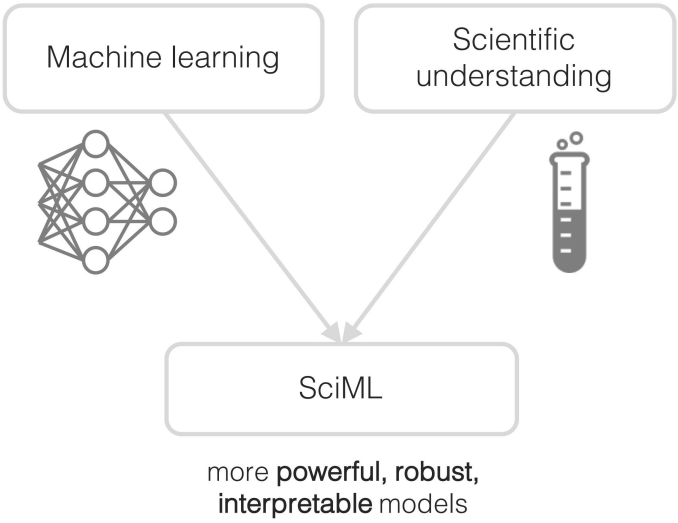


Figura 3.1: SciML overview [34]

Nel mondo della scienza computazionale, al giorno d'oggi, ci si ritrova ad affrontare problemi complessi che non consentono di determinare soluzioni analitiche a meno di semplificazioni, che siano assunzioni restrittive o discretizzazioni di spazio e/o tempo, per semplificare i problemi da risolvere.

Vengono quindi introdotte le reti neurali informate sulla fisica, altresì chiamate Physics-Informed Neural Network (PINN) [38]: si tratta di reti neurali allenate a svolgere compiti rispettando le leggi della fisica descritte da equazioni alle derivate parziali generalmente non lineari (Partial Differential Equation (PDE)). Tali reti neurali sono vincolate a rispettare qualsiasi simmetria, invarianza o principio di conservazione derivante dalle leggi fisiche che governano i dati osservati. Questa costru-

zione semplice ma potente consente di affrontare un'ampia gamma di problemi nella scienza computazionale e introduce una tecnologia potenzialmente trasformativa che porta allo sviluppo di nuove strategie per l'apprendimento sui dati, nuove classi di risolutori numerici per equazioni alle derivate parziali, nonché nuovi approcci basati sui dati per l'inversione e l'identificazione di modelli fisici.

La classificazione di questi nuovi strumenti può avvenire da diversi punti di vista. Uno di questi è la classificazione dal lato *costruttivo*, ovvero categorizzazione della rete a seconda della modalità con cui vengono incorporati i principi fisici (e.g. tramite architettura della rete, funzione di perdita o approcci ibridi). Un altro modo, invece, è quello di concentrarsi sulla tipologia di problemi da affrontare (e.g. simulazione, problema inverso e determinazione di modelli fisici), ovvero l'obiettivo per cui la rete è stata creata [34].

Qui di seguito verranno introdotte e brevemente analizzate le principali categorie di classificazione appena presentate.

3.1 Problemi scientifici

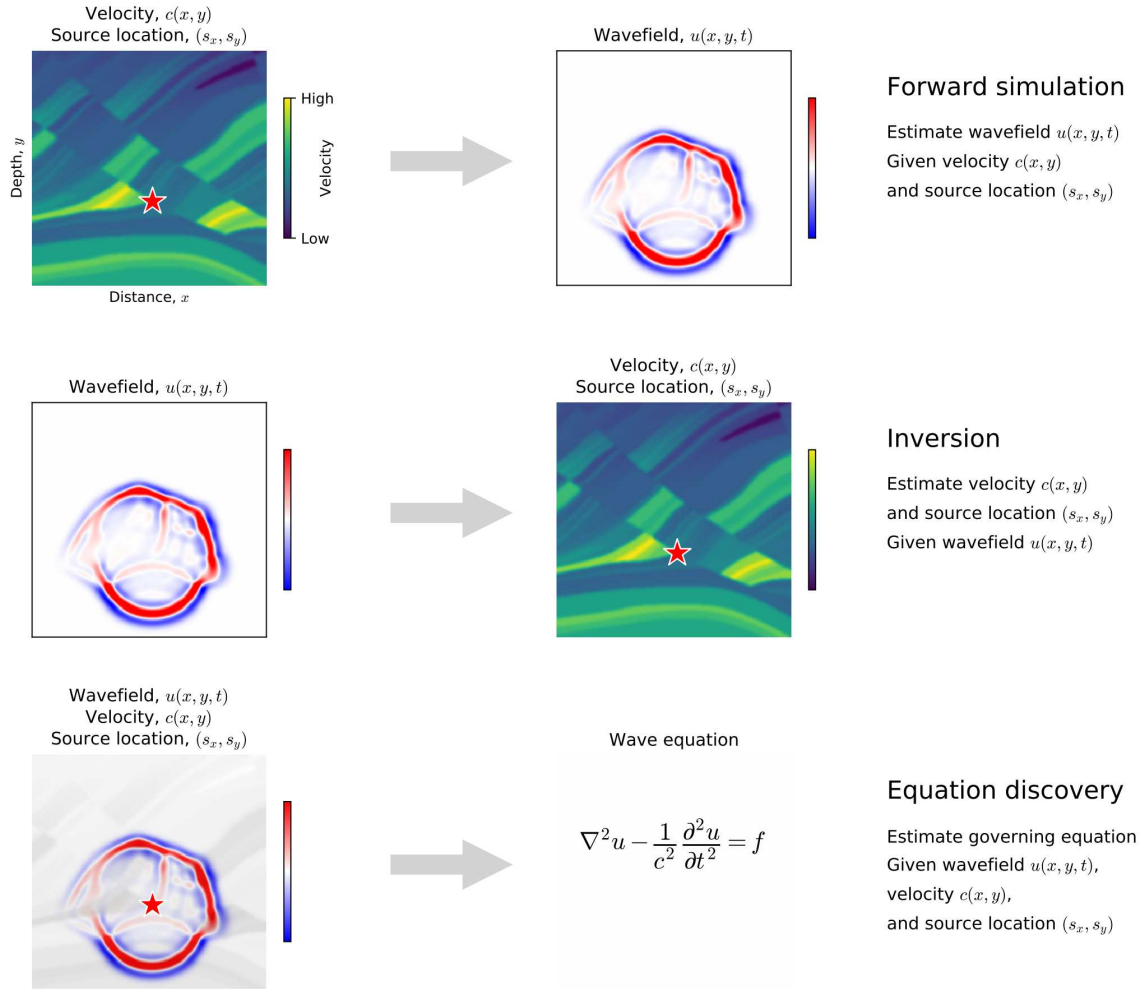


Figura 3.2: Diversi tipi di problemi che una PINN è chiamata a risolvere. In questo grafico vengono visualizzati ciascuno di questi problemi utilizzando l'esempio delle onde sismiche che si propagano attraverso la Terra a seguito di un terremoto [34]

Nella forma più generale, il SciML è chiamato a risolvere 3 principali problemi: simulazione, problema inverso e determinazione di modelli fisici (Fig. 3.2). Si consideri un generale problema fisico descritto da una relazione del tipo:

$$b = F(a) \tag{3.1}$$

ove a si riferisce ai parametri di input del problema, F descrive il modello fisico del sistema in analisi e b rappresenta le grandezze risultanti forniti F ed a

3.1.1 Simulazione

La simulazione è il problema che punta ad ottenere b una volta forniti F ed a. Data la generalità della descrizione del problema fisico, a e b possono contenere valori scalari, vettori, tensori, funzioni o altri elementi descrittivi del sistema. Per molti casi, il problema della simulazione punta a risolvere un set di equazioni differenziali scritte nella forma

$$\begin{aligned} \mathcal{D}[u(x);\, \lambda] &= f(x), \quad x \in \Omega \\ \mathcal{B}_k[u(x)] &= g_k(x), \quad x \in \Gamma_k \subset \partial\Omega \end{aligned} \tag{3.2}$$

per $k = 1, 2, \ldots, n_b$ dove $\mathcal{D}$ rappresenta un operatore differenziale, $\mathcal{B}_k$ un insieme di operatori al contorno, $u \in \mathbb{R}^{d_u}$ è la soluzione all'equazione differenziale, $f(x)$ è una forzante, $g(x)$ è un insieme di funzioni al contorno, x è un vettore di input all'interno del dominio $\Omega \subset \mathbb{R}^d$ (ovvero x è un vettore d-dimensionale), $\partial\Omega$ rappresenta il contorno di Ω e λ è un set aggiuntivo di parametri dell'operatore differenziale.

Un esempio di simulazione [39] è rappresentato in Fig. 3.3, dove una NN avente due ingressi (x e t) ed un'uscita (u) viene allenata seguendo l'equazione di Burger in un dominio 1D con le condizioni al contorno di Dirichlet, ovvero secondo il modello

$$\begin{aligned} u_t + uu_x - \frac{0.01}{\pi} u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ u(0, x) = -\sin(\pi x), \\ u(t, -1) = u(t, 1) = 0. \end{aligned} \tag{3.3}$$

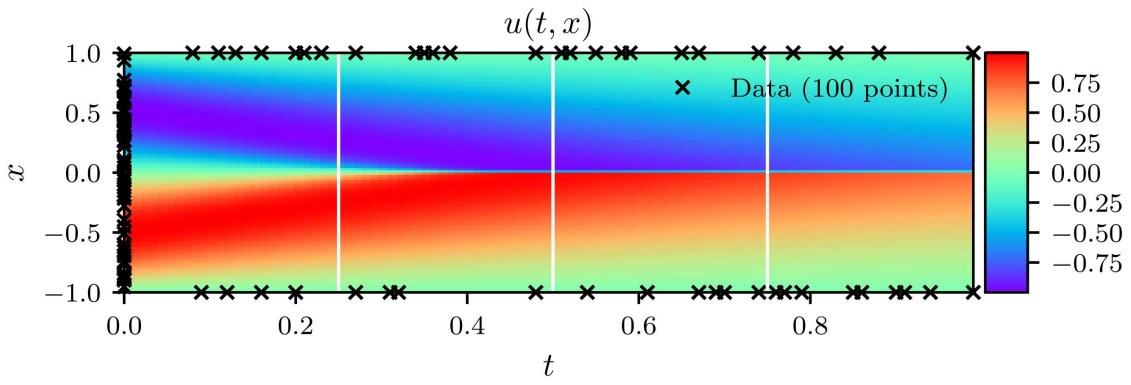


Figura 3.3: Soluzione all'Equazione di Burger prevista $u(t,x)$ insieme ai dati di training iniziali e di contorno. In aggiunta sono stati utilizzati 10.000 collocation points generati utilizzando il campionamento *Latin Hypercube Sampling* (LHS) [39]

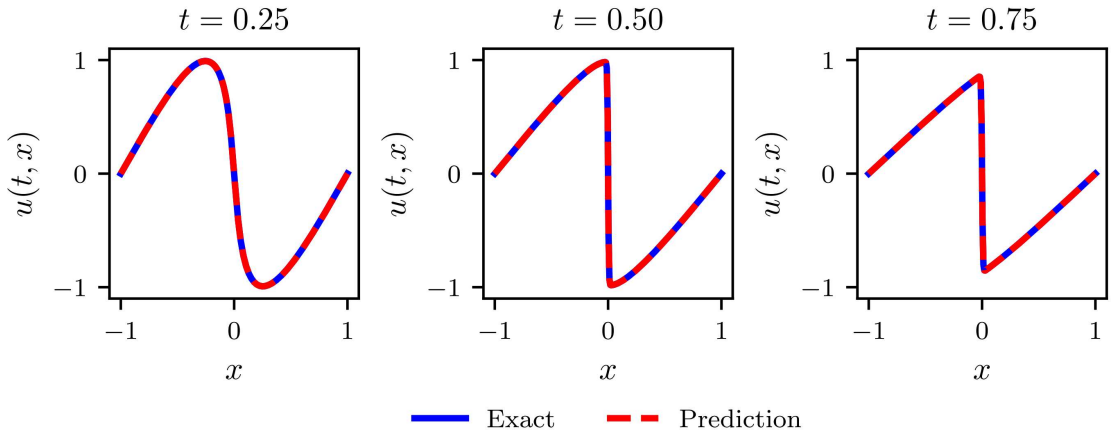


Figura 3.4: Confronto tra le soluzioni previste ed esatte corrispondenti ai tre istanti temporali rappresentate dalle linee verticali bianche rappresentate in Fig. 3.3 [39]

Esistono diversi problemi scientifici di cui conosciamo soluzioni analitiche, come ad esempio i sistemi massa-molla-smorzatore o per i problemi di De Saint Venant: dal sistema di equazioni differenziali, definite le condizioni al contorno, si riesce a ricavare una funzione esatta tramite la quale è possibile a descrivere l'andamento della soluzione. Però, in quasi tutti i casi, la risoluzione di questa tipologia di problema richiede un approccio numerico. Esistono diversi metodi utilizzabili e il più appropriato dipende dal tipo problema e dal dominio scientifico del caso. Nel caso della meccanica strutturale, un metodo popolare per sistemi modellati da PDE è il metodo degli elementi finiti, o Finite Element Method (FEM). In questa strategia di analisi la principale sfida da affrontare riguarda la generazione di una mesh che sia appropriata al problema che si sta cercando di risolvere. Unito al notevole sforzo umano nella definizione di mesh di alta qualità, questa soluzione presenta anche un elevato costo computazionale.

3.1.2 Problema inverso

Il problema inverso è strettamente legato al problema di simulazione e la sua risoluzione è di fondamentale importanza per molte applicazioni nel mondo reale. Questo problema prevede di trovare i valori che descrivono i parametri di input del problema fornito un insieme cospicuo di misurazioni del sistema. Riferendoci al sistema generale descritto dall'Eq. 3.1, conoscendo F, l'obiettivo è di trovare a dato b.

I problemi inversi sono diffusi in tutti i campi della scienza e risolverli è essenziale per molteplici utilizzi del mondo reale, come la diagnostica attraverso immagini sismiche, diagnostica attraverso immagini di risonanze magnetiche, riduzione del rumore di immagini, problemi di design, ecc. Fondamentalmente si tratta di problemi di ricerca, simile al processo di training di una rete: per tentativi vengono provati diversi valori di input che vengono valutati con una funzione di costo rispetto al risultato fornito b. Se F è differenziabile, si procede quindi alla ricerca del valore minimo della funzione di costo, tramite algoritmi basati sul gradiente, che determina il valore ottimale di a. Questo tipo di problemi sono molto complessi da risolvere poiché sono tipicamente mal posti e i valori osservati nel mondo reale sono solitamente affetti da rumore e limitati. Proprio per questo non vi sono sufficienti informazioni per una soluzione univoca; inoltre, essendo una soluzione per tentativi, risulta un processo oneroso in termini computazionali.

Un semplice problema inverso è, per esempio, la determinazione delle condizioni al contorno di un pendolo inverso incernierato ad un carrello, il quale scorre su un binario orizzontale (Fig. 3.5). Di questo problema si conosce b (Eq. 3.1, determinato da misurazioni (con rumore) di $x(t_i)$ e $\phi(t_i)$, e la fisica che lo governa, tramite le equazioni che legano tra loro i vari parametri del pendolo come angoli, masse, forze, etc.

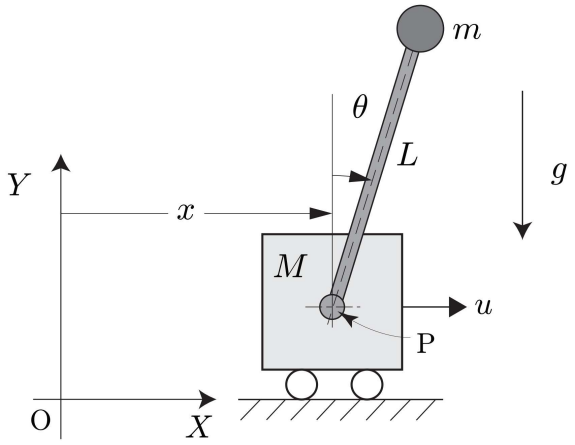


Figura 3.5: Rappresentazione di un pendolo inverso [19]

3.1.3 Determinazione di modelli fisici

Come ultimo problema si affrontano i casi di cui non si comprende a pieno il sistema, ovvero non si è sicuri di come definire il modello fisico F nell'Eq. 3.1. Si potrebbe pensare a questo come a un diverso tipo di problema di inversione in cui l'intero modello fisico, o almeno le sue parti sconosciute, vengono invertite sulla base di alcune

osservazioni del sistema. La determinazione di modelli fisici riguarda in generale la stima di parametri che legano le variabili di un problema. In ogni sistema di PDEs vi sono una o più funzioni incognite legate da derivate parziali delle stesse; queste correlazioni sono accompagnate da fattori moltiplicativi che ne determinano la rilevanza all'interno del modello fisico stesso. Tradizionalmente, la scoperta di nuove teorie per la descrizione della realtà osservabile si è basata sulla straordinaria intuizione umana. Lo SciML sta aiutando questa scoperta permettendo di automatizzare il processo e/o apprendere processi complessi difficili da intuire [10, 32, 46, 9].

3.2 Implementazione dei principi fisici

In questa sezione viene effettuata un'overview degli attuali approcci del SciML. Ci si concentra principalmente sul modo in cui la parte scientifica viene incorporata nell'algoritmo di ML e di seguito verranno presentate diverse strategie di implementazione, ovvero attraverso l'architettura della NN, la funzione di perdita e l'uso di approcci ibridi.

3.2.1 Architettura

Il primo insieme di approcci SciML considerati sono quelli che modificano l'architettura dell'algoritmo di ML in modo che incorpori *vincoli scientifici*, ovvero equazioni e/o condizioni derivati da principi scientifici attinenti al problema che si intende studiare. Invece di trattare un algoritmo di ML come una "scatola nera", ne viene modificata la struttura in modo tale che obbedisca a queste restrizioni imposte. Incorporare principi scientifici in questo modo può limitare la gamma di modelli che l'algoritmo può apprendere. Un modo per modificare la struttura consiste nel codificare le simmetrie nel modello della rete neurale. Infatti, simmetrie come l'invarianza traslazionale e rotazionale svolgono un ruolo fondamentale in fisica e dettano la forma delle leggi della natura. Così come l'incomprimibilità di un materiale, l'irrotazionalità di un fluido o la conservazione di energia di un sistema sono i fondamenti su cui si fonda la fisica della natura, implementarli dal punto di vista costruttivo del modello di ML rende la rete più generalizzabile e consistente a livello fisico.

3.2.2 Funzione di costo

La successiva classe di approcci che viene considerata è quella in cui la funzione di costo viene utilizzata per addestrare l'algoritmo di ML, modificandone il contenuto in modo da includere vincoli scientifici. Questo è solitamente considerato un modo "soft" di imporre vincoli, poiché il modello punta unicamente a minimizzare la funzione di perdita e può quindi ancora disobbedire ai suoi termini vincolanti. Da un punto di vista di apprendimento automatico, i termini presenti nella funzione di perdita possono essere visti come regolatori che restringono lo spazio possibile delle soluzioni che la rete può ritenere valide. Includere un significato scientifico in questo modo può migliorare considerevolmente la convergenza dei modelli, può consentire loro di generalizzare meglio i problemi e permette di ridurre la quantità di dati di addestramento richiesti. Una possibilità è quella di spingere il modello ad obbedire a determinate leggi di conservazione attraverso la sua funzione di perdita, oppure un

vincolo più forte potrebbe essere quello di includere la conoscenza delle equazioni che governano il sistema stesso. Questa idea ha portato a molteplici approcci di SciML, ognuno dei quali si sta rapidamente trasformando in un proprio sottocampo [34].

3.3 Physics-Informed Neural Network

Uno modo recente e popolare utilizzato per affrontare le sfide descritte in precedenza è tramite l'utilizzo di Physics-Informed Neural Network (PINN), progettate per risolvere problemi descritti tramite equazioni differenziali. Le idee iniziali alla base delle Physics-Informed Neural Networks sono state sviluppate negli anni '90 da Lagaris et al. [29] e sono state recentemente reintrodotte ed estese da Raissi et al. [38] utilizzando moderne tecniche di deep learning. Uno schema generico è mostrato in Fig. 3.6: ad alto livello, una PINN è progettata per modellare la soluzione, $u(x)$, di un'equazione differenziale utilizzando una rete neurale, $NN(x;\, \theta)$, per approssimare direttamente la soluzione stessa, ovvero

$$NN(x,\, \theta) \approx u(x) \tag{3.4}$$

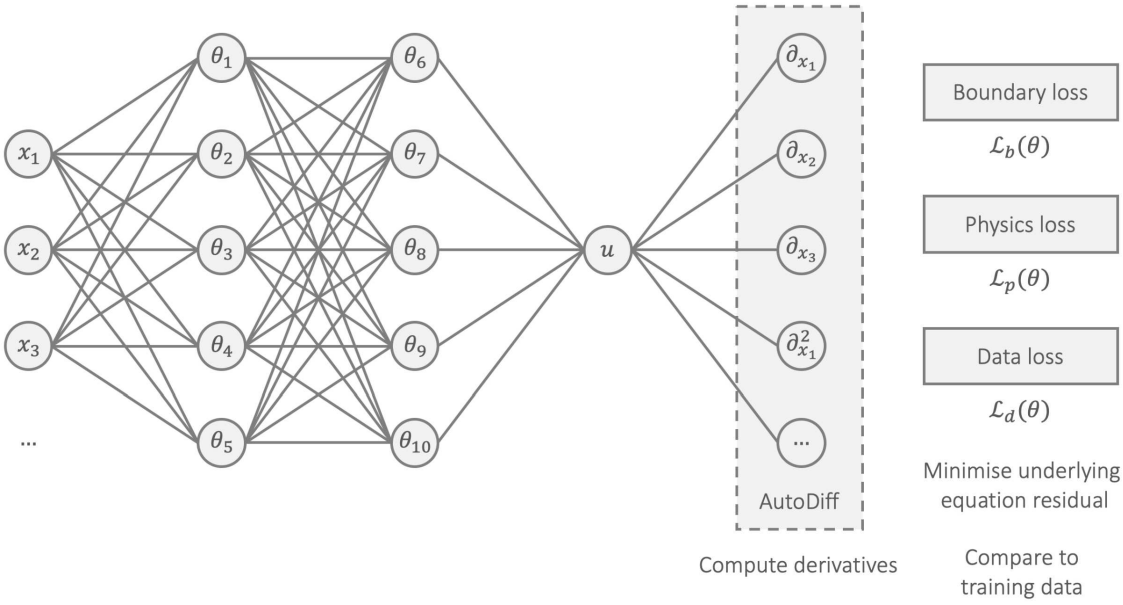


Figura 3.6: Un esempio di PINN [34]

É importante sottolineare che le PINNs si allenano utilizzando una funzione di costo che penalizza il residuo del set di equazioni differenziali che descrivono il problema. Più precisamente, la funzione di costo assume la forma

$$\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + \mathcal{L}_p(\theta) \tag{3.5}$$

dove

$$\mathcal{L}_b(\theta) = \sum_k \frac{1}{N_{bk}} \sum_j^{N_{bk}} \left\| \mathcal{B}_k[NN(x_{kj};\, \theta)] - g_k(x_{kj}) \right\|^2, \tag{3.6}$$

$$\mathcal{L}_p(\theta) = \frac{1}{N_p} \sum_i^{N_p} \left\| \mathcal{D}_k[NN(x_i,\, \theta); \lambda] - f(x_i) \right\|^2 \tag{3.7}$$

che riprende le definizioni e la nomenclatura dell'Eq. 3.2. Il primo termine, indicato come *boundary loss*, $\mathcal{L}_b(\theta)$, fornisce alla funzione di costo il contributo delle k condizioni al contorno del problema. Il secondo termine, indicato come *physics loss*, $\mathcal{L}_p(\theta)$, cerca di garantire che la soluzione obbedisca al sistema di equazioni differenziali del sistema, minimizzandone il residuo in un insieme di punti campionati su tutto il dominio, $\{x_i\}$, chiamati *collocation points*. Intuitivamente, la *physics loss* spinge la rete neurale ad apprendere una soluzione che sia coerente con l'equazione differenziale, mentre il *boundary loss* tenta di garantire che la soluzione sia unica, abbinando la soluzione alle condizioni al contorno note.

Pertanto, allenare una Physics-Informed Neural Network equivale a risolvere un'equazione differenziale. A differenza di una rete neurale standard, che viene addestrata con un dataset formato da un cospicuo numero di campioni, chiamati *data points*, una PINN richiede un dataset per il training di gran lunga più ridotto, riuscendovi comunque ad estrapolare la soluzione. Inoltre, queste offrono potenziali vantaggi rispetto ai metodi tradizionali quando vengono utilizzati per risolvere equazioni differenziali. In primo luogo, forniscono una soluzione continua e meshless, quindi si tratta di soluzioni esenti da discretizzazioni utilizzate negli approcci tradizionali, come le analisi FEM. In secondo luogo, forniscono gradienti analitici e trattabili rispetto ai loro input, che sono preziosi per compiti come l'inversione e l'analisi della sensibilità.

3.3.1 Dataset

Come accennato in precedenza, il dataset utilizzato per allenare una PINN svolge la medesima funzione di un dataset "standard", ma vi differisce dal lato costruttivo. Un dataset "standard" prevede di fornire alla rete una serie di input ed il rispettivo valore di output atteso, chiamato anche *true value*. Ad esempio, è stato effettuato uno studio medico con l'obbiettivo di creare un modello capace individuare specifiche malattie polmonari tramite l'utilizzo di reti neurali [1] ed è stata utilizzata una Deep Convolutional Neural Network (DCNN), ovvero una tipologia di rete molto diffusa per il training di reti tramite l'utilizzo di immagini. In questo caso il dataset è composto da 16 435 immagini ottenute da scansioni ai raggi-x e TAC, suddiviso in 4 classi diverse (Sano (10.325), COVID-19 (3.749), Polmonite batterica (883) e Polmonite virale (1.478)). Quindi, all'interno del dataset ogni input (immagine) è correlato al proprio output (stato di salute) e vi è creata l'associazione tra i due. Di conseguenza, nella creazione del dataset il comportamento che la NN è chiamata ad apprendere deve essere prima eseguito manualmente tramite l'ausilio dell'essere umano. Questo tipo di training dataset, infatti, è utilizzato in quello che viene chiamato *supervised machine learning*, ovvero appendimento automatico supervisionato, dove il dataset costruito per il training deve essere esatto.

Nel caso, invece, di *unsupervised machine learning*, ovvero di apprendimento automatico non supervisionato, il dataset utilizzato non presenta valori di output attesi, componendosi quindi unicamente di una lista di input. In tal modo non è necessario associare gli input ai true value, evitando il compito oneroso da parte dell'essere umano. In questi casi il dataset è composto da punti appartenenti ad un dominio matematico, come possono essere coordinate spaziali cartesiane 2D (x, y) o 3D (x, y, z), oppure coordinate spaziali polari 3D $(\rho, \varphi, \vartheta)$, o anche coordinate spazio-temporali (x, t). Questi collocation points campionati all'interno del dominio del problema in esame rappresentano il contenuto del dataset utilizzato per determinare la physics loss, $\mathcal{L}_p(\vartheta)$.

Per i collocation points definiti al contorno, che quindi definiscono le condizioni al contorno, o Boundary Condition (b.c.), del problema, si conoscono i risultati della funzione incognita, e/o delle derivate parziali. Parlando in senso pratico, questi punti sono composti, e operano, alla stessa maniera dei data points presenti nei dataset utilizzati nel supervised learning. D'altra parte, concettualmente parlando, vi possono essere più tipologie di data points nell'apprendimento non supervisionato: come accennato in precedenza, i *boundary points*, ovvero punti al contorno, gli *initial points*, ovvero punti che definiscono le condizioni iniziali di un problema temporale e, per casi pratici, le misurazioni, ovvero punti di cui si conosce una o più grandezze grazie a sensori e/o trasduttori posti su di un componente testato o dall'estrapolazione di dati ottenuti da simulazioni.

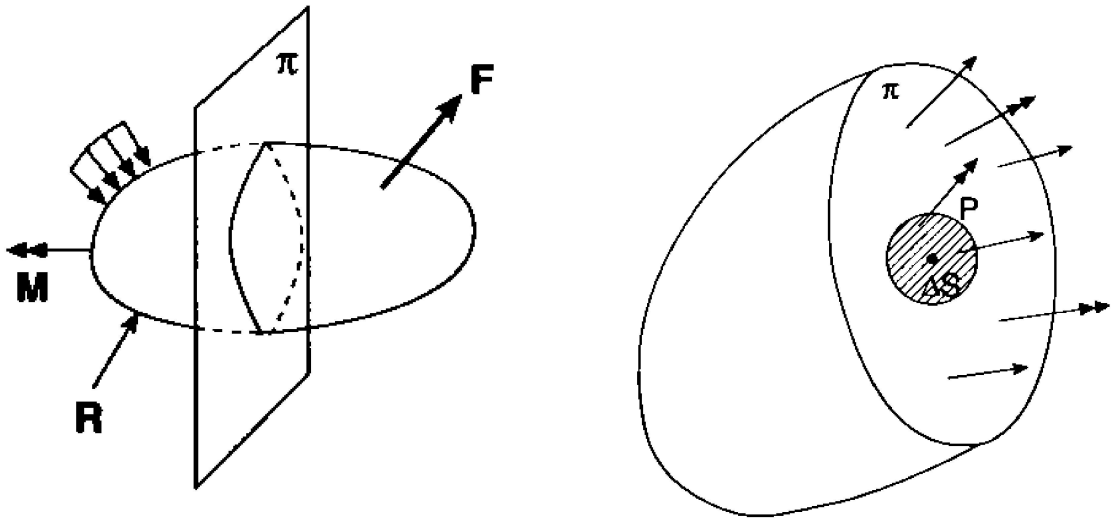
Capitolo 4

Introduzione alla meccanica strutturale

In questo capitolo vengono introdotti diversi stumenti atti allo studio di sforzi e deformazioni a sui sono soggetti corpi che rispondono a carichi e vincoli: viene introdotto il tensore della tensione di Cauchy, per lo studio dell'equilibrio dei corpi, il tensore della deformazione infinitesima, che descrive le deformazioni in campo lineare, e il tensore della deformazione di Green. Si introducono i materiali lineari, che rispondono a leggi costitutive lineari, dove deformazioni e stress sono correlati da coefficienti costanti. Vi è una distinzione tra le diverse tipologie di non linearità di un problema: la prima riguarda la non linearità del materiale, ovvero quando si trattano materiali che non rispondono alla legge di Hooke [31] come possono essere materiali iperelastici o compositi; la seconda riguarda la non linearità geometrica, che tratta problemi in cui non si può applicare l'ipotesi di piccoli spostamenti. Questa è la chiave che differenzia le analisi lineari e non lineari che verranno effettuate in questo studio. In ogni caso, per tutti i problemi trattati, si rimane nell'ipotesi di piccole deformazioni.

4.1 Tensore della tensione

Nella meccanica strutturale l'analisi di componenti e strutture sottoposti a carichi e condizioni di vincolo permette di determinare lo stato tensionale, ovvero come il componente risponde alle sollecitazioni applicate. Nel momento in cui un corpo caricato si trova in quiete, si dice che esso è in equilibrio: questo implica che la somma degli sforzi esterni applicati è pari a zero. Si immagini ora di avere un corpo in equlibrio sottoposto a carichi esterni (Fig. 4.1a): le forze applicate si compongono in una risultante nulla e il corpo in questione è in quiete.



(a) (b)

Figura 4.1: Generico corpo soggetto a carichi e vincoli (a). Corpo sezionato da un piano π [31]

Si pensi ora di tagliare a metà l'oggetto, come rappresentato in Fig. 4.1b: ciascuna delle due parti prese singolarmente non starebbe in equilibrio. Questo ci porta a pensare che all'interno si vengono a generare forze e momenti uguali e contrari che controbilanciano i carichi esterni e garantiscano l'equilibrio di ciascuna parte. In questo frangente, la teoria di Cauchy definisce che sulla superficie del taglio, in un intorno ΔS di P (Fig. 4.1b), siano validi i seguenti limiti:

$$\lim_{\Delta S \to 0} \frac{\mathbf{F}(\Delta S)}{\Delta S} = \mathbf{t}_n(P), \tag{4.1}$$

$$\lim_{\Delta S \to 0} \frac{\mathbf{M}(\Delta S)}{\Delta S} = 0. \tag{4.2}$$

I termini $\mathbf{F}$ e $\mathbf{M}$ sono la forza e il momento agenti nell'intorno di P, mentre $\mathbf{t}_n(P)$ è detto vettore tensione e rappresenta una forza superficiale. Il simbolo utilizzato per descrivere il vettore evidenzia due fatti importanti: la tensione dipende dal punto di applicazione, che implica una variazione di intensità e/o direzione da punto a punto presenti nel taglio; inoltre questo vettore dipende dal piano di taglio effettuato, poiché è evidenziata la dipendenza rispetto alla normale di quest'ultimo.

4.1.1 Il tetraedro di Cauchy

Tramite l'ausilio del teorema del tetraedro di Cauchy nel caso piano, osserviamo in che modo il vettore tensione cambia al variare del taglio effettuato. Si pensi di ritagliare un triangolo all'interno di un corpo in equilibrio che, essendo una parte di questo, dovrà essere anch'esso in equilibrio. In questo triangolo, rappresentato in Fig. 4.2 agiscono lungo i bordi tutte le forze di superficie descritte finora e all'interno vi agiscono le forze di volume. Definendo le equazioni di equilibrio, otteniamo

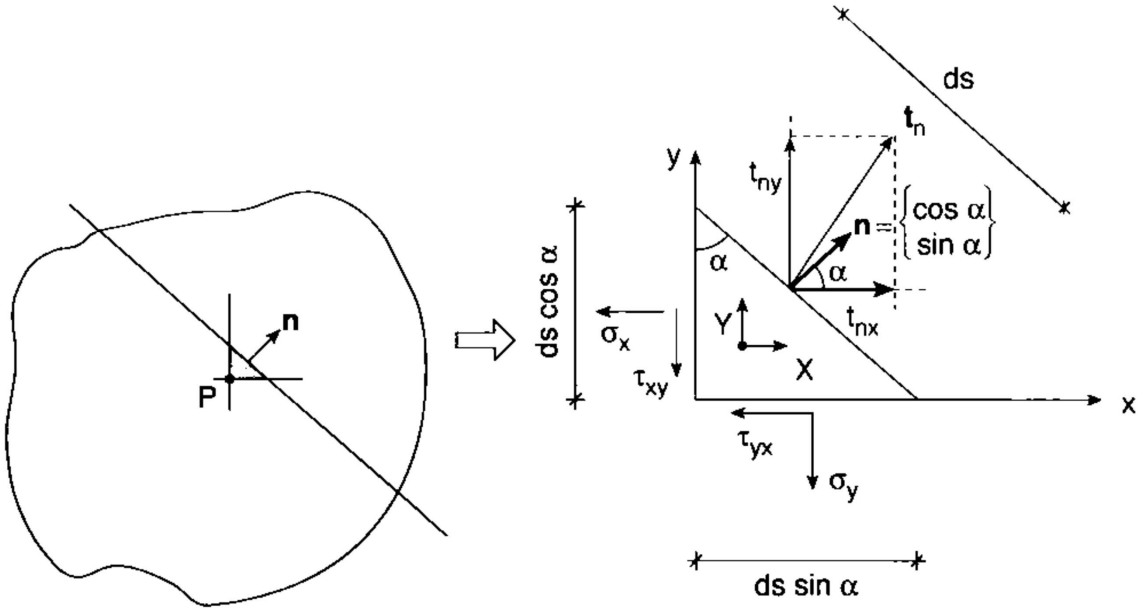


Figura 4.2: Tetraedro di Cauchy in 2D [31]

$$-\sigma_x\, ds \cos\alpha - \tau_{yx}\, ds \sin\alpha + t_{nx}\, ds + X\, ds\, \cos\alpha\, ds\, \sin\alpha/2 = 0 \tag{4.3}$$

$$-\tau_{xy}\, ds\, \cos\alpha - \sigma_y\, ds\, \sin\alpha + t_{ny}\, ds + Y\, ds\, \cos\alpha\, ds\, \sin\alpha/2 = 0. \tag{4.4}$$

Trascurando le forze di volume che sono moltiplicate per un ordine superiore di ds, e rappresentando i risultati in forma matriciale, risulta

$$\left\{ \begin{array}{c} t_{nx} \\ t_{ny} \end{array} \right\} = \left[\begin{array}{cc} \sigma_x & \tau_{yx} \\ \tau_{xy} & \sigma_y \end{array} \right] \left\{ \begin{array}{c} \cos\alpha \\ \sin\alpha \end{array} \right\} \quad \Rightarrow \quad \mathbf{t}_n = \mathbf{T}\,\mathbf{n} \tag{4.5}$$

Queste equazioni si possono generalizzare al caso tridimensionale, dove un versore è della forma $\mathbf{n} = \{\alpha_x, \alpha_y, \alpha_z\}^T$, tramite la relazione

$$\mathbf{t}_n = \mathbf{T}\,\mathbf{n} \quad \Rightarrow \quad \left\{ \begin{array}{c} t_{nx} \\ t_{ny} \\ t_{nz} \end{array} \right\} = \left[\begin{array}{ccc} \sigma_x & \tau_{yx} & \tau_{zx} \\ \tau_{xy} & \sigma_y & \tau_{zy} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{array} \right] \left\{ \begin{array}{c} \alpha_x \\ \alpha_y \\ \alpha_z \end{array} \right\}. \tag{4.6}$$

dove la matrice $\mathbf{T}$ rappresenta il tensore delle tensioni. Le tensioni agenti su di un volumetto cubico infinitesimo in un dominio 3D sono quindi rappresentate in Fig. 4.3.

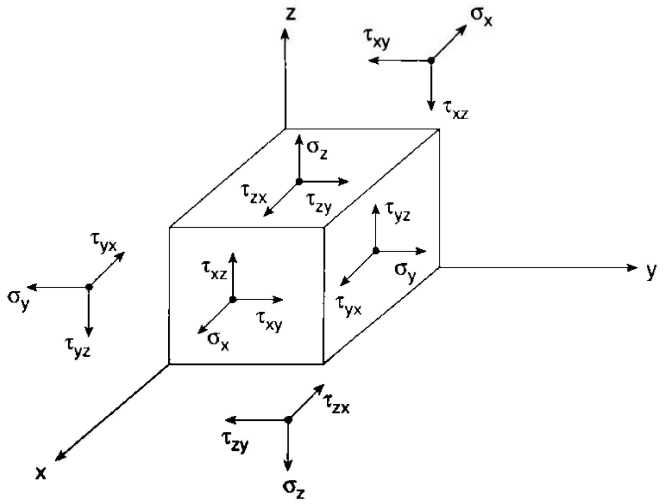


Figura 4.3: Volume infinitesimo [31]

4.1.2 Equazioni indefinite di equilibrio

Il prossimo passo è quello di comprendere come il vettore $\mathbf{t}_n$ dipenda dal punto di applicazione P. Data la relazione espressa nell'Eq. 4.6 si è già definita la dipendenza del vettore tensione con il piano di taglio tramite la normale $\mathbf{n}$: questo ci porta a dedurre che la dipendenza del punto di applicazione risieda all'interno del tensore delle tensioni.

Viene analizzato ora l'equilibrio lungo x del volumetto infinitesimo (Fig. 4.4): utilizzando l'espansione di Taylor e tenendo conto solo dei termini del primo ordine otteniamo

$$-\sigma_x\,dy\,dz + \left(\sigma_x + \frac{\partial\sigma_x}{\partial x}dx + \dots\right)dy\,dz - \tau_{yx}\,dx\,dz + \left(\tau_{yx} + \frac{\partial\tau_{yx}}{\partial y}dy + \dots\right)dx\,dz$$

$$-\tau_{zx}\,dx\,dy + \left(\tau_{zx} + \frac{\partial\tau_{zx}}{\partial z}dz + \dots\right)dx\,dy + X\,dx\,dy\,dz = 0 \tag{4.7}$$

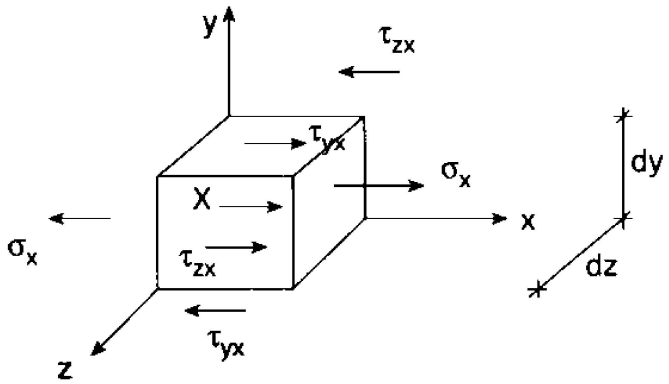


Figura 4.4: Componenti delle tensioni lungo x [31]

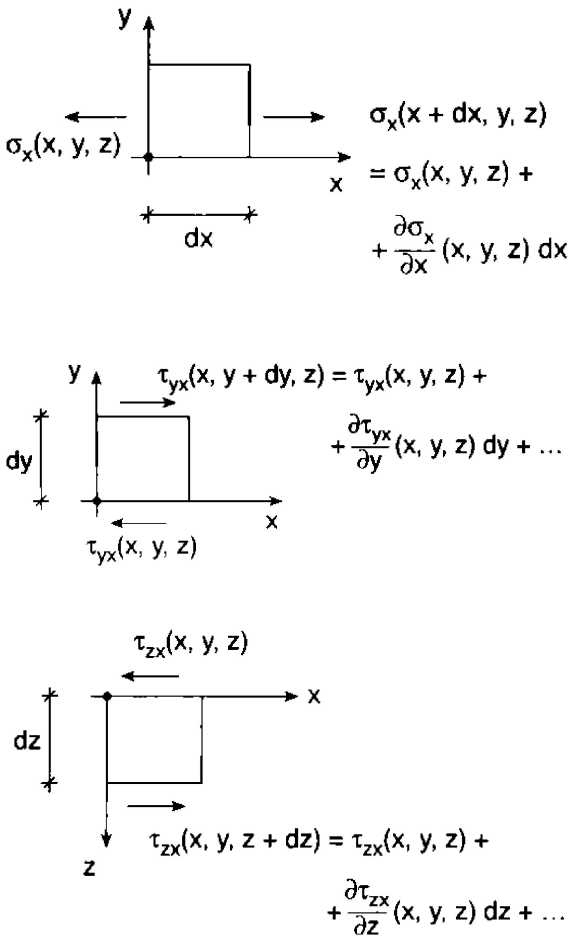


Figura 4.5: Andamento delle componenti delle tensioni lungo x [31]

da cui, semplificando,

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + X = 0 \tag{4.8}$$

Ragionando con lo stesso metodo rispetto anche alle direzioni y e z risulta

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + Y = 0 \tag{4.9}$$

$$\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + Z = 0 \tag{4.10}$$

In forma matriciale le espressioni precedenti possono essere riassunte come

$$\text{div } \mathbf{T} + \mathbf{X} = 0 \tag{4.11}$$

A seguito della determinazione dell'equilibrio rispetto alle traslazioni, imponiamo anche l'equilibrio rispetto alle rotazioni. Si parte analizzando la rotazione attorno all'asse z (Fig. 4.6), ottenendo

$$-\big(\tau_{yx}\,dx\,dz\big)\frac{dy}{2} - \left[\left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y}dy + \dots\right) dx\,dz\right]\frac{dy}{2} + \big(\tau_{xy}\,dy\,dz\big)\frac{dx}{2} + \\ + \left[\left(\tau_{xy} + \frac{\partial \tau_{xy}}{\partial x}dx + \dots\right) dy\,dz\right]\frac{dx}{2} = 0 \tag{4.12}$$

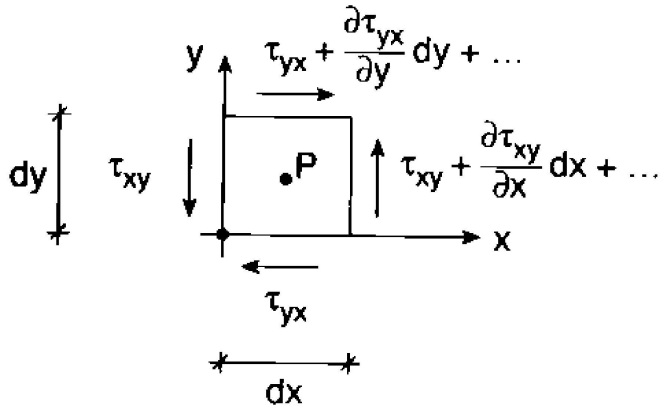


Figura 4.6: Componenti che danno contributo alla rotazione attorno a z

Semplificando e trascurando infinitesimi di ordine superiore otteniamo

$$\tau_{xy} = \tau_{yx}. \tag{4.13}$$

Ripetendo il medesimo procedimendo per gli altri assi otteniamo

$$\begin{aligned} \tau_{yz} &= \tau_{zy} \\ \tau_{zx} &= \tau_{xz} \end{aligned} \tag{4.14}$$

Questo determina che il tensore delle tensioni è simmetrico, ovvero che le componenti incognite sono solo 6 $(\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{xz}, \tau_{yz})$

$$\mathbf{T} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} \tag{4.15}$$

4.2 Tensore delle deformazioni

A seguito delle relazioni appena trovate che determinano lo stato tensionale del corpo in esame, è necessario studiare la configurazione che questo assume a seguito di spostamenti e deformazioni. É bene comprendere come gli spostamenti del corpo

contribuiscano alla deformazione dello stesso e separarli da spostamenti relativi unicamente a moti rigidi. Un diverso approccio utilizzato per ricavare i tensori delle deformazioni di nostro interesse si può trovare in Appendice A, che parte da un concetto più generale di moti dei punti materiali fino a semplificarsi al caso di piccoli spostamenti.

4.2.1 Formulazione lineare delle deformazioni

Si consideri un corpo deformabile arbitrario in una configurazione iniziale (*c.i.*) che subisce uno spostamento generico verso una configurazione deformata (*c.d.*). Le equazioni che legano le posizioni del corpo in *c.i.* e in *c.d.* sono definite come

$$\begin{aligned} x'(x,y,z) &= x + u(x,y,z) \\ y'(x,y,z) &= y + v(x,y,z) \\ z'(x,y,z) &= z + w(x,y,z) \end{aligned} \tag{4.16}$$

dove (x, y, z) rappresenta la *c.i.*, (x', y', z') rappresenta la *c.d.*, (u, v, w) rappresenta la funzione spostamento $\mathbf{s}$ [31].

In generale, la funzione spostamento di un generico punto P appartenente al corpo, $\mathbf{s}(P)$, è sottoposta a limitazioni fisico-meccaniche e matematiche che delimitano il dominio nel quale la teoria è applicabile:

1. Un punto P non può andare a finire in due punti distinti: questo comporterebbe creazione di materia. Si definisce, quindi, $\mathbf{s}(P)$ come una funzione nel senso matematico, che associa ad un elemento del dominio un unico elemento del codominio.

2. Due punti P e Q non possono congiungersi in un unico punto: questo comporterebbe distruzione o compenetrazione di materia. Queste prime limitazioni rendono quindi $\mathbf{s}(P)$ una funzione biunivoca: le Eq. 4.16 possono essere invertite ottenendo $x(x', y', z')$, $y(x', y', z')$, $z(x', y', z')$.

3. Due punti P e Q vicini in *c.i.* devono essere vicini anche in *c.d.*: in caso contrario si formerebbero delle cricche/fessure. Questo determina che $\mathbf{s}(P)$ deve essere continua.

4. L'immagine di un segmento interno al corpo deve essere una curva senza spigoli: in caso contrario si avrebbero rotture a livello microscopico. Ne segue che $\mathbf{s}(P)$ deve essere derivabile.

5. La funzione $\mathbf{s}(P)$ deve rispettare i vincoli interni ed esterni, come ad esempio $\mathbf{s}(P) = 0$ in presenza di un incastro nel punto P.

Ipotesi di piccoli spostamenti

L'ipotesi di piccoli spostamenti suggerisce di studiare ciò che avviene nell'intorno di un generico punto O che consideriamo l'origine del sistema di riferimento (Fig. 4.7).

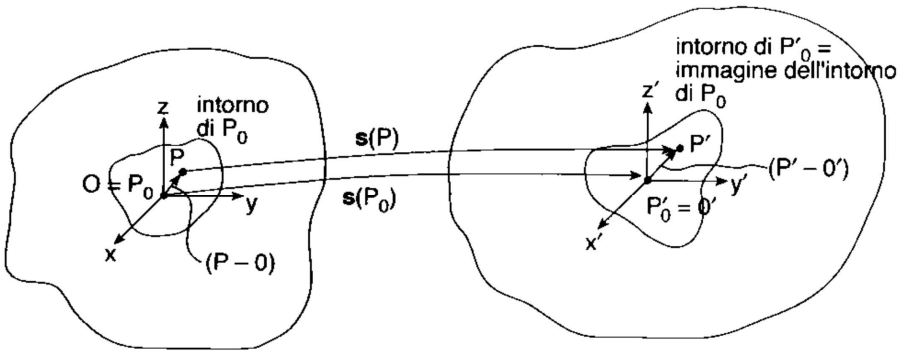


Figura 4.7: Spostamento dell'intorno di un generico punto O [31]

Lo spostamento di un punto P nell'intorno di O, ovvero origine del sistema di riferimento con coordinate pari a $(0,0,0)$, può essere calcolato a partire da quello di quest'ultimo mediante lo sviluppo in serie di Taylor:

$$\begin{aligned} u(P) = u(x,y,z) = u(0,0,0) + \frac{\partial u}{\partial x}(0,0,0)\,x + \frac{\partial u}{\partial y}(0,0,0)\,y + \frac{\partial u}{\partial z}(0,0,0)\,z + \\ + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(0,0,0)\,x^2 + \frac{1}{2}\frac{\partial^2 u}{\partial y^2}(0,0,0)\,y^2 + \frac{1}{2}\frac{\partial^2 u}{\partial z^2}(0,0,0)\,z^2 + \\ + \frac{\partial^2 u}{\partial x \partial y}(0,0,0)\,xy + \frac{\partial^2 u}{\partial x \partial z}(0,0,0)\,xz + \frac{\partial^2 u}{\partial y \partial z}(0,0,0)\,yz + \dots \end{aligned} \tag{4.17}$$

Trascurando i termini di ordine superiore al primo ed eseguendo la medesima operazione anche per le altre direzioni, si può scrivere in forma matriciale

$$\begin{Bmatrix} u_P \\ v_P \\ w_P \end{Bmatrix} = \begin{Bmatrix} u_O \\ v_O \\ w_O \end{Bmatrix} + \begin{bmatrix} \dfrac{\partial u}{\partial x} & \dfrac{\partial u}{\partial y} & \dfrac{\partial u}{\partial z} \\ \dfrac{\partial v}{\partial x} & \dfrac{\partial v}{\partial y} & \dfrac{\partial v}{\partial z} \\ \dfrac{\partial w}{\partial x} & \dfrac{\partial w}{\partial y} & \dfrac{\partial w}{\partial z} \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \quad \Longrightarrow \quad \mathbf{s}_P = \mathbf{s}_O + \nabla \mathbf{s} \cdot \overline{OP} \tag{4.18}$$

dove $\nabla \mathbf{s}$ rappresenta il gradiente di $\mathbf{s}_P$ calcolato in O. Questa può essere rappresentata come la somma di una parte simmetrica e una antisimmetrica, definite rispettivamente:

$$\mathbf{E} = \frac{\nabla \mathbf{s} + \nabla \mathbf{s}^T}{2} = \begin{bmatrix} \dfrac{\partial u}{\partial x} & \dfrac{1}{2}\left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right) & \dfrac{1}{2}\left(\dfrac{\partial u}{\partial z} + \dfrac{\partial w}{\partial x}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial v}{\partial x} + \dfrac{\partial u}{\partial y}\right) & \dfrac{\partial v}{\partial y} & \dfrac{1}{2}\left(\dfrac{\partial v}{\partial z} + \dfrac{\partial w}{\partial y}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial w}{\partial x} + \dfrac{\partial u}{\partial z}\right) & \dfrac{1}{2}\left(\dfrac{\partial w}{\partial y} + \dfrac{\partial v}{\partial z}\right) & \dfrac{\partial w}{\partial z} \end{bmatrix} \tag{4.19}$$

$$\mathbf{W} = \frac{\nabla \mathbf{s} - \nabla \mathbf{s}^T}{2} = \begin{bmatrix} 0 & \frac{1}{2}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) & 0 & \frac{1}{2}\left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right) & 0 \end{bmatrix}, \quad (4.20)$$

che trasforma l'Eq. 4.18 in

$$\mathbf{s}_P = \mathbf{s}_O + \mathbf{W} \cdot \overline{OP} + \mathbf{E} \cdot \overline{OP} \quad (4.21)$$

Qui si nota come $\mathbf{s}_O$ e $\mathbf{W}$ descrivono rispettivamente la traslazione e la rotazione rigida dell'intorno di O. Ne segue che la deformazione è descritta dal termine $\mathbf{E}$: questo viene chiamato tensore della deformazione infinitesima [31].

Dilatazione lineare

Dal momento che è stato definito il tensore delle deformazioni infinitesime, si ricerca il significato di ciascuna delle sue componenti. Viene introdotta la deformazione lineare ε_n, altresì chiamata deformazione ingegneristica ε_E, definita come

$$\varepsilon_n = \frac{l_f - l_i}{l_i} \quad (4.22)$$

che, data l'ipotesi di piccoli spostamenti ($l_i \approx l_f$), può anche essere scritta come $\varepsilon_n = (l_f - l_i)/l_f$ [31]. Questa è positiva per allungamenti e negativa per accorciamenti, adimensionale e dell'ordine di $\varepsilon_n \approx \pm 10^{-4}$. Inoltre dipende sia dal punto in cui si calcola che dalla direzione di interesse n.

Si consideri ora un segmento, generato da un generico punto P ed un punto Q appartenente al suo intorno, di lunghezza $l_i = dx$ allineato lungo l'asse x. A seguito della deformazione il segmento si troverà nella configurazione descritta dai punti P′ e Q′ (Fig. 4.8).

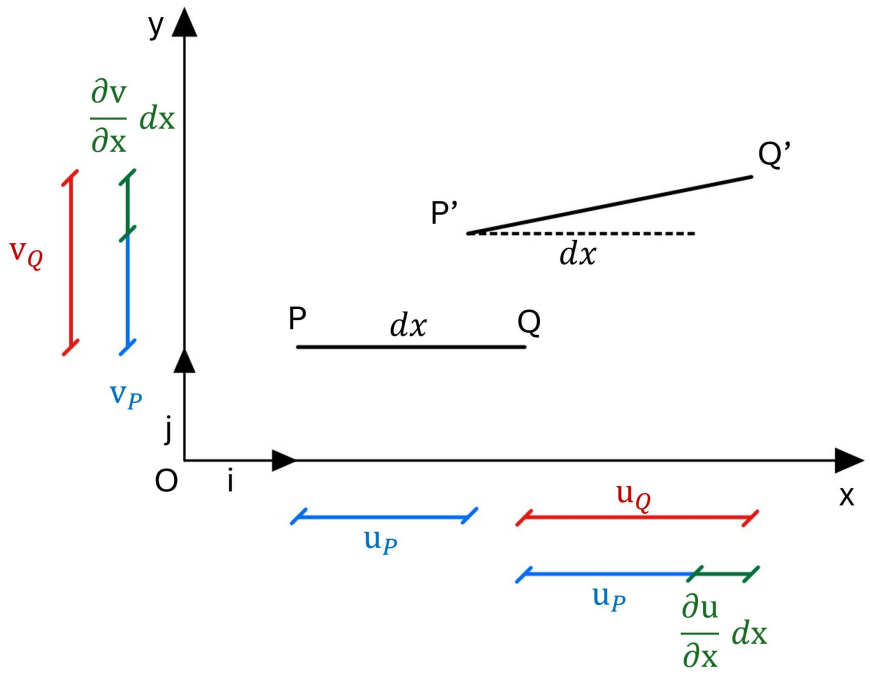


Figura 4.8: Rappresentazione del coefficiente di dilatazione lineare

Utilizzando l'Eq. 4.18 otteniamo

$$\begin{aligned} u_Q &= u_P + \left(\frac{\partial u}{\partial x}\right)_P dx + \dots \\ v_Q &= v_P + \left(\frac{\partial v}{\partial x}\right)_P dx + \dots \end{aligned} \tag{4.23}$$

dove consideriamo solamente le derivate lungo x in quanto siamo interessati alla dilatazione lineare ε_x. Calcoliamo la lunghezza finale come

$$l_f = \sqrt{\left(dx + \frac{\partial u}{\partial x}dx\right)^2 + \left(\frac{\partial v}{\partial x}dx\right)^2} \approx \left|dx + \frac{\partial u}{\partial x}dx\right| \tag{4.24}$$

dove trascuriamo $\dfrac{\partial v}{\partial x}dx$ perché è un infinitesimo di ordine superiore. Utilizzando l'Eq. 4.22, otteniamo

$$\varepsilon_x = \frac{l_f - l_i}{l_f} = \frac{dx + \dfrac{\partial u}{\partial x}dx - dx}{dx + \dfrac{\partial u}{\partial x}dx} = \frac{\partial u}{\partial x}. \tag{4.25}$$

Ripetendo il medesimo ragionamento per le direzioni y e z risulta

$$\varepsilon_y = \frac{\partial v}{\partial y} \quad \text{e} \quad \varepsilon_y = \frac{\partial w}{\partial z}. \tag{4.26}$$

Si può quindi affermare che i termini sulla diagonale del tensore **E** misurano le variazioni di lunghezza lungo gli assi del sistema di riferimento [31].

Scorrimento angolare

Compresa la natura degli elementi diagonali della matrice **E**, si è interessati a definire il ruolo degli elementi extra-diagonali della stessa. Viene introdotto il coefficiente di scorrimento angolare relativo alle generiche direzioni n e m, definito come

$$\gamma_{nm} = \alpha_i - \alpha_f \tag{4.27}$$

che misura la differenza tra l'angolo iniziale e finale a seguito della deformazione [31]. Questo è un valore positivo per angoli che si riducono e negativo per angoli che aumentano, adimensionale e dell'ordine di $\gamma_{nm} \approx \pm 10^{-4}$. Inoltre dipende sia dal punto in cui si calcola che dalle direzioni di interesse n e m.

Si considerino ora due segmenti, determinati dai punti P, Q e R, di lunghezza $\overline{PQ} = dx$ e $\overline{PR} = dy$, posti ortogonalmente l'uno rispetto all'altro e allineati rispettivamente lungo x e lungo y. A deformazione avvenuta i due segmenti si presentano nella deformazione deformata descritta dai punti P′, Q′ e R′ (Fig. 4.9).

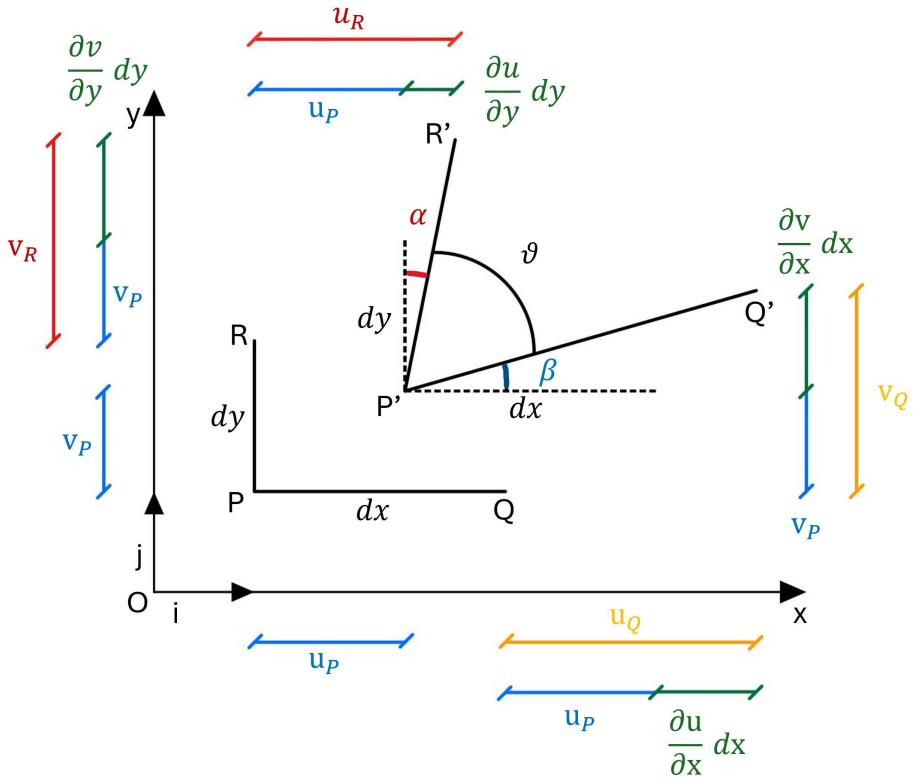


Figura 4.9: Rappresentazione del coefficiente di scorrimento angolare

Utilizzando l'Eq. 4.18 otteniamo

$$
\begin{aligned}
u_Q &= u_P + \left(\frac{\partial u}{\partial x}\right)_P dx + \dots \\
v_Q &= v_P + \left(\frac{\partial v}{\partial x}\right)_P dx + \dots \\
u_R &= u_P + \left(\frac{\partial u}{\partial y}\right)_P dy + \dots \\
v_R &= v_P + \left(\frac{\partial v}{\partial y}\right)_P dy + \dots
\end{aligned}
\tag{4.28}
$$

dove per ogni singolo segmento consideriamo solo le derivate lungo la rispettiva direzione principale. Gli angoli α e β vengono quindi definiti come

$$
\alpha \approx \tan\alpha = \frac{\dfrac{\partial u}{\partial y}dy}{\dfrac{\partial v}{\partial y}dy + dy} = \frac{\dfrac{\partial u}{\partial y}dy}{\left(1 + \dfrac{\partial v}{\partial y}\right)dy} \approx \frac{\partial u}{\partial y} \tag{4.29}
$$

$$
\beta \approx \tan\alpha = \frac{\dfrac{\partial v}{\partial x}dx}{\dfrac{\partial u}{\partial x}dx + dx} = \frac{\dfrac{\partial v}{\partial x}dx}{\left(1 + \dfrac{\partial u}{\partial x}\right)dx} \approx \frac{\partial v}{\partial x} \tag{4.30}
$$

ove viene applicata l'ipotesi di piccoli spostamenti che consente di porre $\alpha \approx \tan\alpha$ e $\beta \approx \tan\beta$. Inoltre vengono trascurate le quantità $\dfrac{\partial v}{\partial y}$ e $\dfrac{\partial u}{\partial x}$ poiché sono $\ll 1$. Il coefficiente di scorrimento angolare misura quindi

$$
\gamma_{xy} = \frac{\pi}{2} - \vartheta = \alpha + \beta = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{4.31}
$$

Ripetendo la medesima operazione per i piani xz e yz risulta

$$
\gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}, \quad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \tag{4.32}
$$

Si può quindi affermare che i termini non diagonali del tensore **E** misurano le variazioni angolari tra assi del sistema di riferimento [31].

Tensore delle deformazioni infinitesime

Dalle componenti definite nei precedenti paragrafi possiamo quindi determinare le equazioni di congruenza che legano gli spostamenti alle deformazioni

$$\varepsilon_{xx} = \frac{\partial u}{\partial x}, \quad \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$$
$$\varepsilon_{yy} = \frac{\partial v}{\partial y}, \quad \varepsilon_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{4.33}$$
$$\varepsilon_{zz} = \frac{\partial w}{\partial z}, \quad \varepsilon_{zx} = \frac{1}{2}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)$$

Di conseguenza è possibile costruire il tensore della deformazione infinitesima esplicitando ognuna delle 6 componenti indipendenti

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{zz} \end{bmatrix} \tag{4.34}$$

4.2.2 Formulazione non-lineare di Green

Nelle analisi lineari, nello studio tra le relazioni tensioni-deformazioni, la formulazione delle componenti del tensore delle deformazioni **E** viene svolta assumendo l'ipotesi di piccoli spostamenti, ovvero si suppone che le posizioni iniziali e finali siano praticamente coincidenti.

Questo implica che nello sviluppo in serie di Taylor, per considerare lo spostamento di un punto a partire da uno adiacente, vengono considerati solo termini del primo ordine, trascurando i termini di ordine superiore. Quando, però, ci troviamo di fronte ad un problema in presenza di grandi spostamenti questa ipotesi non è più valida e vi è necessaria una distinzione tra le coordinate iniziali e finali dei punti durante le deformazioni.

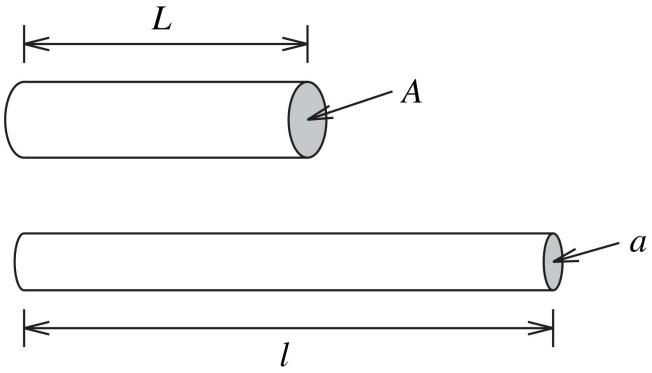


Figura 4.10: Deformazione monodimensionale di un'asta [7]

Si parte prendendo in considerazione un caso monodimensionale di un'asta soggetta a trazione, che si allunga da una lunghezza iniziale L ad una finale l (Fig. 4.10). Il

modo più semplice per misurare la deformazione dell'asta è tramite la deformazione ingegneristica

$$\varepsilon_E = \frac{l - L}{L}. \tag{4.35}$$

Una diversa definizione della deformazione molto più facilmente generalizzabile nei casi del continuum è la deformazione di Green, definita come

$$\varepsilon_G = \frac{l^2 - L^2}{2L^2}. \tag{4.36}$$

Indipendentemente dalla tipologia di definizione di deformazione utilizzata, tramite una semplice analisi dell'espansione di Taylor, possiamo notare che per piccoli spostamenti ($l \approx L$) la deformazione di Green coincide con la deformazione ingegneristica

$$\begin{aligned} \varepsilon_G(l \approx L) &\approx \frac{(l + \Delta l)^2 - l^2}{2l^2} \\ &= \frac{1}{2}\frac{l^2 + \Delta l^2 + 2l\Delta l - l^2}{l^2} \\ &\approx \frac{\Delta l}{l}. \end{aligned} \tag{4.37}$$

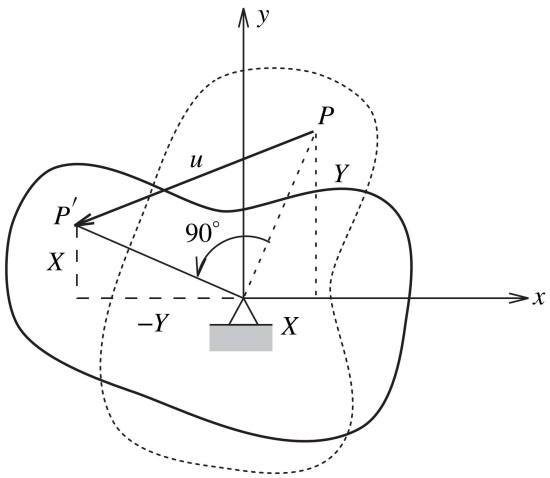


Figura 4.11: Rotazione nel piano di 90° di un corpo 2D [7]

Si consideri ad esempio un generico corpo soggetto ad una rotazione nel piano di 90° rispetto all'origine degli assi (Fig. 4.11). Gli spostamenti u e v di un punto P appartenente al corpo corrispondono a

$$u = -Y - X \quad \text{e} \quad v = X - Y \tag{4.38}$$

dove X e Y sono le coordinate iniziali del punto P. Di conseguenza, applicando le Eq. 4.33, risulta

$$\varepsilon_{xx} = \varepsilon_{yy} = -1 \quad \text{e} \quad \varepsilon_{xy} = 0 \tag{4.39}$$

Considerando che il corpo in questione è soggetto ad una rotazione rigida è chiaro che le formulazioni proposte sono incorrette, in quanto determinano una deformazione che non avviene. Perciò è d'obbligo ridefinire matematicamente la deformazione per corpi soggetti a grandi spostamenti.

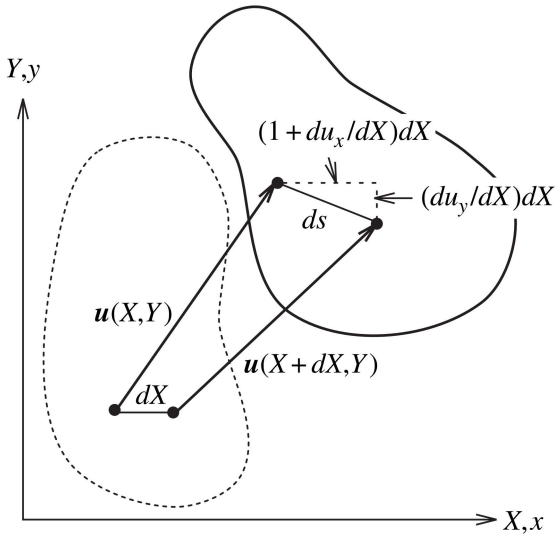


Figura 4.12: Generica deformazione di un corpo bidimensionale [7]

Quindi, consideriamo un elemento infinitesimo di lunghezza dX inizialmente parallelo all'asse x che, appartente ad un corpo soggetto a carichi esterni, si deforma fino a raggiungere la lunghezza ds. Tramite gli spostamenti rappresentati in Fig. 4.12 valutiamo la lunghezza ds come

$$ds^2 = \left(dX + \frac{\partial u_x}{\partial X} dX\right)^2 + \left(\frac{\partial u_y}{\partial X} dX\right)^2 \tag{4.40}$$

dove $u_x = u$ è lo spostamento lungo x mentre $u_y = v$ è lo spostamento lungo y. Basandoci sulla definizione di deformazione di Green per il caso monodimensionale (Eq. 4.36), possiamo ricavarci la componente x del caso di esempio preesntato in Fig. 4.12, ovvero

$$\begin{aligned} E_{xx} =& \frac{ds^2 - dX^2}{2dX^2} \\ =& \frac{\partial u}{\partial X} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial X}\right)^2 + \left(\frac{\partial v}{\partial X}\right)^2\right]. \end{aligned} \tag{4.41}$$

Per piccoli spostamenti, possiamo vedere come trascurando i termini quadratici otteniamo che $E_{xx} = \varepsilon_{xx}$. Possiamo verificare, inoltre, che $E_{xx} = 0$ per il caso rappresentato in Fig. 4.11, il che coincide a ciò che viene intuitivo pensare.

Tensore delle deformazioni di Green

In un'ottica più generale, possiamo definire l'intero tensore delle deformazioni nel dominio 3D basandoci sulla formulazione di Green, definendo ciascuna componente

$$
\begin{aligned}
E_{xx} &= \frac{\partial u}{\partial X} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial X}\right)^2 + \left(\frac{\partial v}{\partial X}\right)^2 + \left(\frac{\partial w}{\partial X}\right)^2\right] \\
E_{yy} &= \frac{\partial v}{\partial Y} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial Y}\right)^2 + \left(\frac{\partial v}{\partial Y}\right)^2 + \left(\frac{\partial w}{\partial Y}\right)^2\right] \\
E_{zz} &= \frac{\partial w}{\partial Z} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial Z}\right)^2 + \left(\frac{\partial v}{\partial Z}\right)^2 + \left(\frac{\partial w}{\partial Z}\right)^2\right] \\
E_{xy} &= \frac{1}{2}\left(\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial X}\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X}\frac{\partial v}{\partial Y} + \frac{\partial w}{\partial X}\frac{\partial w}{\partial Y}\right] \\
E_{xz} &= \frac{1}{2}\left(\frac{\partial u}{\partial Z} + \frac{\partial w}{\partial X}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial X}\frac{\partial u}{\partial Z} + \frac{\partial v}{\partial X}\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial X}\frac{\partial w}{\partial Z}\right] \\
E_{yz} &= \frac{1}{2}\left(\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial Y}\frac{\partial u}{\partial Z} + \frac{\partial v}{\partial Y}\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y}\frac{\partial w}{\partial Z}\right]
\end{aligned}
\tag{4.42}
$$

Considerando che il tensore delle deformazioni è simmetrico, possiamo quindi comporlo come segue

$$
\mathbf{E} = \begin{bmatrix} E_{xx} & E_{xy} & E_{xz} \\ E_{xy} & E_{yy} & E_{yz} \\ E_{xz} & E_{yz} & E_{zz} \end{bmatrix} \tag{4.43}
$$

4.3 Legame costitutivo

Da un punto di vista matematico, per risolvere il problema del corpo deformabile in maniera univoca, è necessario che il numero di equazioni e di funzioni incognite sia il medesimo.

Lo studio dello stato tensionale del corpo ha introdotto 6 funzioni incognite nel problema $(\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{xz}, \tau_{yz})$, mentre lo studio del comportamento deformativo del corpo ha introdotto altre 9 funzioni incognite $(u, v, w, \varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{xy}, \varepsilon_{xz}, \varepsilon_{yz})$. Le equazioni che legano queste funzioni sono le 6 equazioni differenziali di congruenza

(Eq. 4.33 o Eq. 4.42) e le 3 equazioni differenziali di equilibrio (Eq. 4.11) oltre alle condizioni al contorno.

Con un totale di 15 funzioni incognite e solo 9 equazioni, è necessario definire altre 6 equazioni che chiudano il problema. Difatti, è intuitivo pensare che al problema manchi l'informazione sul materiale utilizzato, poiché la natura del materiale influenza la soluzione del problema stesso. Di conseguenza le 6 equazioni mancanti devono tenere conto del comportamento del materiale e prendono il nome di *equazioni di legame costitutivo*.

4.3.1 Materiali isotropi

Nella descrizione del comportamento di un materiale è di fondamentale importanza la prova monoassiale, che misura la relazione tra la deformazione subita dal materiale e la tensione che si genera all'interno di esso. In Fig. 4.13 è rappresentato lo schema del caso più complesso possibile che descrive le principali caratteristiche che si verificano durante la prova sperimentale. Nella sua semplicità l'esperimento consiste nella trazione di un provino di un dato materiale, con sezione e lunghezza iniziale A_o e L_o. Misurando la forza di trazione e l'allungamento del provino che ne consegue, e dividendo le quantità misurate per le condizioni iniziali A_o e L_o, è possibile seguire l'andamento della la tensione nominale media rispetto al coefficiente di dilatazione lineare medio.

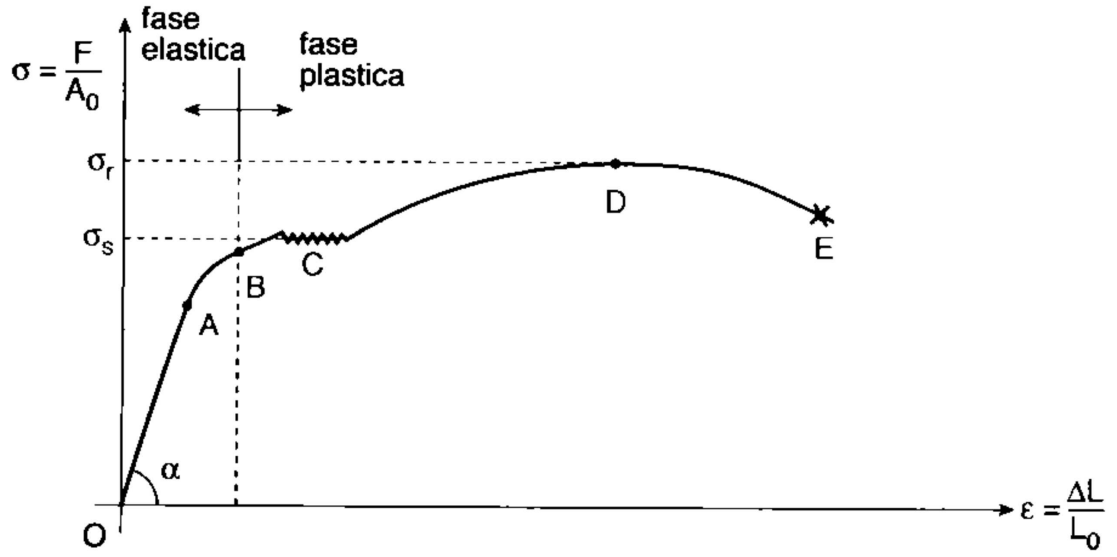


Figura 4.13: Andamento tensione-deformazione [31]

Si identificano 2 fasi principali nella prova monoassiale. La prima è la fase elastica dove la deformazione del corpo dovuta carichi esterni è presente fintanto che questi sono applicati; nel momento in cui questi vengono rimossi il materiale ritorna alla sua configurazione iniziale. La seconda è la fase plastica dove non può più essere recuperata la configurazione iniziale, modificando in maniera permanente il materiale. La zona che verrà considerata in questo studio è quella rappresentata dalla fase elastica, più precisamente dal tratto lineare OA. In questo tratto la tensione e la deformazione sono legate dalla relazione lineare

$$\sigma = E\,\varepsilon + \sigma_o = E\,(\varepsilon - \varepsilon_o) \tag{4.44}$$

chiamata *legge di Hooke*. I materiali che verranno considerati in questo studio sono quelli che vanno sotto il nome di *materiali nello stato naturale*, per i quali $\sigma_o = 0$ e $\varepsilon_o = 0$. Nell'Eq. 4.44 la costante di proporzionalità E è chiamata *modulo di Young* ed è una proprietà intrinseca del materiale: geometricamente è la tangente del tratto lineare OA in Fig. 4.13 e ha le dimensioni di una tensione N/m^2 (molto più comunemente viene usato un suo multiplo, ovvero il $MPa = N/mm^2 = 10^6 Pa$). In Tab. 4.1 sono riportati alcuni materiali e le relative proprietà

Materiale	**E** (**MPa**)	$\boldsymbol{\nu}$ **(Poisson)**	$\boldsymbol{\sigma_s}$ **snerv.** (**MPa**)	$\boldsymbol{\sigma_r}$ **rottura** (**MPa**)	**Densità** (**kg/m^3**)
Acciaio	208000	0.31	400	500	7850
Alluminio	70000	0.35	40	100	2750
Diamante	1200000	0.20		(traz.) 3500 (comp.) 96500	3500
Ghisa	110000	0.22	200	280	7200
Granito	60000	0.15		(traz.) 6 (comp.) 110	2800
Vetro	68000	0.22		50	2200

Tabella 4.1: Valori medi indicativi di alcuni materiali presi in esempio [31]

Nella forma generalizzata al caso tridimensionale la legge di Hooke prende in considerazione tutte le componenti del tensore delle tensioni e del tensore delle deformazioni, legandoli tra loro tramite la relazione lineare

$$\mathbf{T} = \mathbb{C}[\mathbf{E}] + \mathbf{T}_o = \mathbb{C}[\mathbf{E} - \mathbf{E}_o] \tag{4.45}$$

o in componenti, e priva di tensioni e deformazioni residue,

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{Bmatrix} = \begin{bmatrix} C_{xxxx} & C_{xxyy} & C_{xxzz} & C_{xxxy} & C_{xxxz} & C_{xxyz} \\ C_{yyxx} & C_{yyyy} & C_{yyzz} & C_{yyxy} & C_{yyxz} & C_{yyyz} \\ C_{zzxx} & C_{zzyy} & C_{zzzz} & C_{zzxy} & C_{zzxz} & C_{zzyz} \\ C_{xyxx} & C_{xyyy} & C_{xyzz} & C_{xyxy} & C_{xyxz} & C_{xyyz} \\ C_{xzxx} & C_{xzyy} & C_{xzzz} & C_{xzxy} & C_{xzxz} & C_{xzyz} \\ C_{yzxx} & C_{yzyy} & C_{yzzz} & C_{yzxy} & C_{yzxz} & C_{yzyz} \end{bmatrix} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{Bmatrix} \tag{4.46}$$

La classificazione mediante 36 parametri di un materiale è una via poco pratica e risulta molto complicata da affrontare. Esistono tuttavia molte classi di materiali che vengono descritte da un numero inferiore di parametri: per questo studio vengono considerati i materiali isotropi, che necessitano solo di due parametri. Vengono introdotti il *coefficiente di Poisson*, ν, ovvero un numero adimensionale che misura

quanto una tensione ha effetti sulle deformazioni trasversali rispetto a quelle longitudinali sulle quali è applicata, e il *modulo di elasticità tangenziale*, G, che lega le tensioni e deformazioni extra-diagonali (Eq. 4.15, 4.34, 4.43). Un materiale isotropo è caratterizzato dall'uniformità del suo comportamento in tutte le direzioni e presenta quindi un unico modulo di Young, un unico coefficiente di Poisson ed un unico modulo di elasticità tangenziale. Il legame costitutivo può essere scritto nella forma

$$\begin{aligned}
\varepsilon_{xx} &= \frac{1}{E}[\sigma_x - \nu(\sigma_y + \sigma_z)] = \frac{1}{E}[(1+\nu)\,\sigma_x - \nu\,\mathbf{I}_1(\mathbf{T})], \quad \varepsilon_{xy} = \frac{\tau_{xy}}{2G}, \\
\varepsilon_{yy} &= \frac{1}{E}[\sigma_y - \nu(\sigma_x + \sigma_z)] = \frac{1}{E}[(1+\nu)\,\sigma_y - \nu\,\mathbf{I}_1(\mathbf{T})], \quad \varepsilon_{xz} = \frac{\tau_{xz}}{2G}, \\
\varepsilon_{zz} &= \frac{1}{E}[\sigma_z - \nu(\sigma_x + \sigma_y)] = \frac{1}{E}[(1+\nu)\,\sigma_z - \nu\,\mathbf{I}_1(\mathbf{T})], \quad \varepsilon_{yz} = \frac{\tau_{yz}}{2G}
\end{aligned} \tag{4.47}$$

ovvero, invertendo,

$$\begin{aligned}
\sigma_x &= (2\mu+\lambda)\varepsilon_{xx} + \lambda(\varepsilon_{yy} + \varepsilon_{zz}) = 2\mu\,\varepsilon_{xx} + \lambda\,\mathbf{I}_1(\mathbf{E}), \quad \tau_{xy} = 2\mu\,\varepsilon_{xy}, \\
\sigma_y &= (2\mu+\lambda)\varepsilon_{yy} + \lambda(\varepsilon_{xx} + \varepsilon_{zz}) = 2\mu\,\varepsilon_{yy} + \lambda\,\mathbf{I}_1(\mathbf{E}), \quad \tau_{xz} = 2\mu\,\varepsilon_{xz}, \\
\sigma_z &= (2\mu+\lambda)\varepsilon_{zz} + \lambda(\varepsilon_{xx} + \varepsilon_{yy}) = 2\mu\,\varepsilon_{zz} + \lambda\,\mathbf{I}_1(\mathbf{E}), \quad \tau_{yz} = 2\mu\,\varepsilon_{yz},
\end{aligned} \tag{4.48}$$

dove

$$\mu = G, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} = \frac{2G\nu}{1-2\nu} \tag{4.49}$$

sono le costanti di Lamè [31] e nelle quali si è utilizzata la relazione

$$G = \frac{E}{2(1+\nu)}. \tag{4.50}$$

Il termine $\mathbf{I}_1$ è l'invariante principale e rappresenta la traccia, ovvero la somma delle componenti diagonali della matrice presente nell'argomento ($\mathbf{I}_1(\mathbf{E}) = \mathrm{tr}(\mathbf{E}) = E_{11} + E_{22} + E_{33}$). In una scrittura più sintetica le equazioni appena presentate si possono scrivere in forma matriciale, utilizzando il tensore delle tensioni e il tensore delle deformazioni, ottenendo

$$\mathbf{T} = 2\mu\,\mathbf{E} + \lambda\,\mathbf{I}_1(\mathbf{E}) \tag{4.51}$$

Capitolo 5

Teoria della trave e il problema di De Saint Venant

In questo capitolo viene trattata la teoria della trave, necessaria per lo studio delle sollecitazioni e degli spostamenti utilizzando la rete neurale che verrà successivamente implementata. In particolare, l'attenzione principale è rivolta ai carichi di forza normale, alla flessione semplice della trave e all'ipotesi semi-inversa.

5.1 La trave nello spazio

Si può assumere una trave come un solido generato da una figura piana S, chiamata sezione, che si muove nello spazio rimanendo ortogonale alla propria traiettoria s descritta dal proprio baricentro G. Quindi, la curva $G(s)$ è detta asse della trave ed in generale non è rettilinea. Viene definita, quindi, una trave descritta dal proprio asse come un corpo monodimensionale, in casi in cui la lunghezza di questa sia almeno un ordine di grandezza più grande rispetto alla massima dimensione caratteristica della sezione.

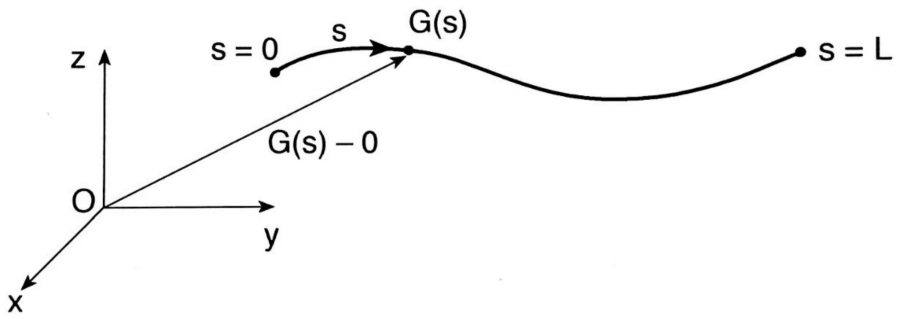


Figura 5.1: Trave descritta tramite il proprio asse $G(s)$ [31]

Poiché la trave è per definizione un oggetto monodimensionale, è utile descriverne le caratteristiche fondamentali facendo riferimento solo al suo asse, pensandolo come una curva regolare nello spazio $s \rightarrow [G(s) - O]$ (Fig. 5.1). La parametrizzazione utilizzata misura quindi la lunghezza della curva da un punto di riferimento. Cal-

colando la derivata del vettore posizione $[G(s) - O]$ rispetto alla traiettoria s, si ottiene il versore tangente $\mathbf{t}$

$$\mathbf{t}(s) = \frac{d[G(s) - O]}{ds} = [G(s) - O]' \tag{5.1}$$

Essendo s l'ascissa che misura la lunghezza della curva, si ha che $\mathbf{t}$ è un versore $|\mathbf{t}(s)| = 1$, ovvero $\mathbf{t}(s) \cdot \mathbf{t}(s) = 1$, dove $\cdot$ indica il prodotto scalare. Derivando questa espressione si ottiene

$$\mathbf{t}'(s) \cdot \mathbf{t}(s) \; + \; \mathbf{t}(s) \cdot \mathbf{t}'(s) \; = \; 2\mathbf{t}'(s) \cdot \mathbf{t}(s) \; = \; 0 \tag{5.2}$$

che dimostra l'ortogonalità tra $\mathbf{t}'$ e $\mathbf{t}$. Tramite

$$\mathbf{t}'(s) = \kappa(s)\, \mathbf{n}(s) = \frac{1}{\rho(s)}\, \mathbf{n}(s) \tag{5.3}$$

si definisce quindi la curvatura della trave $\kappa(s)$ come il modulo di $\mathbf{t}'$, il raggio di curvatura $\rho(s) = 1/\kappa(s)$ come l'inverso della curvatura e la normale principale $\mathbf{n}$ come il versore di $\mathbf{t}'$. Quindi $\mathbf{n}$ è perpendicolare a $\mathbf{t}$. Il raggio di curvatura ρ è il raggio della circonferenza che meglio approssima localmente l'asse della trave; la curvatura è l'indice che misura quanto il comportamento si discosta dalla rettilineità. Per travi rettilinee $\mathbf{t}(s) = cost$ e $\kappa = 0$: per semplicità in questi casi si sceglie l'asse z coincidente con l'asse della trave, per cui $s = z$.

5.1.1 La trave nel piano

Una trave nel piano può essere rappresentata dall'equazione $y = y(x)$ (Fig. 5.2)

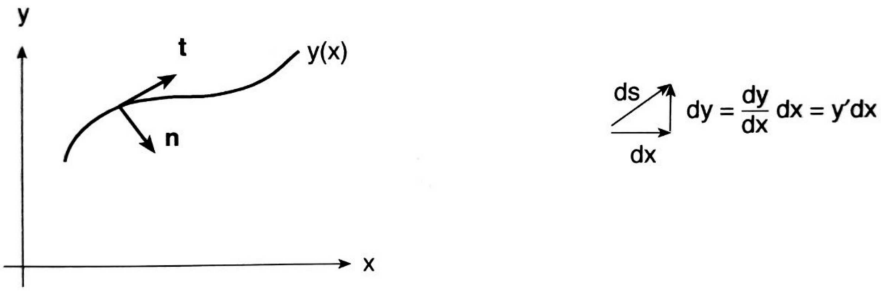


Figura 5.2: Rappresentazione di una trave monodimensionale nel piano xy [31]

Da questa definizione è possibile calcolare la lunghezza dell'elemento infinitesimo della curva ds come:

$$ds = \sqrt{(dx)^2 + (dy)^2} \quad \Longrightarrow \quad \frac{ds}{dx} = \sqrt{1 + y'(x)^2}. \tag{5.4}$$

Quindi in questo caso

$$\mathbf{t} = \frac{d}{ds}\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \cdots = \frac{1}{\sqrt{1+y^{'}(x)^2}}\begin{bmatrix} 1 \\ y^{'}(x) \end{bmatrix} \tag{5.5}$$

$$\mathbf{t}^{'} = \frac{d\mathbf{t}}{dx}\frac{dx}{ds} = \cdots = \frac{-y^{''}(x)}{[1+y^{'}(x)^2]^{3/2}}\frac{1}{\sqrt{1+y^{'}(x)^2}}\begin{bmatrix} y^{'}(x) \\ -1 \end{bmatrix} \tag{5.6}$$

dove

$$\kappa = \frac{-y^{''}(x)}{[1+y^{'}(x)^2]^{3/2}}, \quad \mathbf{n} = \frac{1}{\sqrt{1+y^{'}(x)^2}}\begin{bmatrix} y^{'}(x) \\ -1 \end{bmatrix} \tag{5.7}$$

5.2 Il problema di De Saint Venant

Considerando una trave come un oggetto deformabile è possibile risolvere il problema elastico in maniera completa ed esatta, ricavandovi delle soluzioni analitiche. Questo problema è chiamato problema di De Saint Venant (DSV) e con esso viene considerata la deformabilità della trave e il suo volume, permettendo così di eliminare la semplificazione per cui la trave veniva considerata solo facendo riferimento al proprio asse. La forma della trave è un cilindro rettilineo con base arbitraria con sezione trasversale costante A, detto anche solido del DSV. Il sistema di riferimento del corpo è scelto in modo tale che l'asse z sia allineato all'asse della trave, mentre gli assi x e y siano principali d'inerzia e baricentrici. Effettuando un taglio perpendicolare all'asse della trave (Fig. 5.3) si può individuare il vettore tensione $\mathbf{t}_z = \{\tau_{zx}, \tau_{zy}, \sigma_z\}^T$ agente nell'intorno di un punto generico.

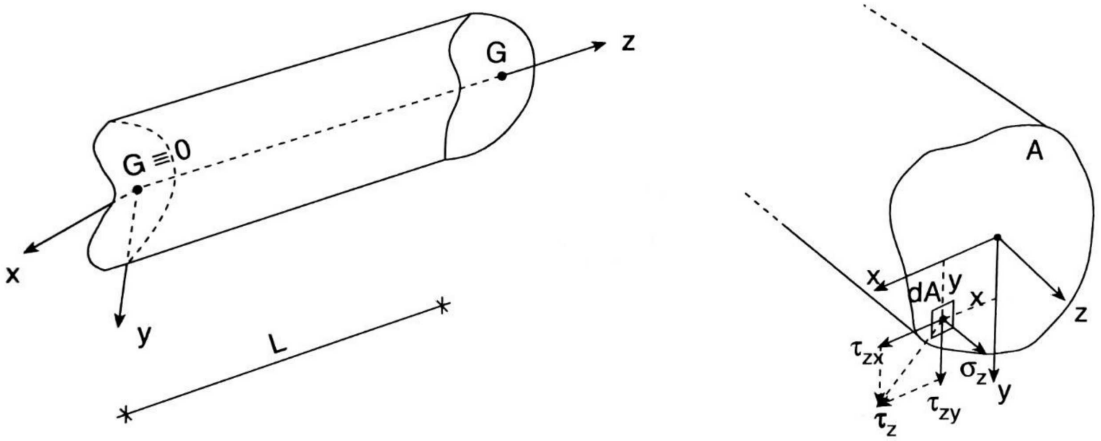


Figura 5.3: Trave di DSV in uno spazio 3D e componenti del vettore tensione agente nell'intorno dA di un generico punto della sezione [31]

Le forze risultanti sulla sezione, ovvero le azioni interne, sono quindi

$$
\begin{aligned}
N &= \int_A \sigma_z \, dA \quad \text{(forza normale)}, \\
T_x &= \int_A \tau_{zx} \, dA \quad \text{(taglio lungo } x\text{)}, \\
T_y &= \int_A \tau_{zy} \, dA \quad \text{(taglio lungo } y\text{)}, \\
M_x &= \int_A y\sigma_z \, dA \quad \text{(momento flettente attorno a } x\text{)}, \\
M_y &= \int_A x\sigma_z \, dA \quad \text{(momento flettente attorno a } y\text{)}, \\
M_t &= \int_A (x\,\tau_{zy} - y\,\tau_{zx}) \, dA \quad \text{(momento torcente)}.
\end{aligned}
\tag{5.8}
$$

La soluzione del problema di DSV prevede di trovare le componenti del vettore tensione in funzione delle risultanti, invertendo così le Eq. 5.8. Le tensioni sono proprietà del punto di applicazione, e quindi dipendono da x, y e z; d'altra parte invece le risultanti sono proprietà di una sezione, quindi dipendono solo da z. In questo tipo di problema il materiale della trave viene assunto come linearmente elastico, isotropo ed omogeneo. Le condizioni al contorno della trave in questo caso non sono importanti, quindi non è necessario porre alcun vincolo particolare. Per quanto riguarda i carichi, non sono presenti forze di volume e i carichi vengono applicati solo sulle sezioni alle estremità della trave; con il postulato di DSV si considerano solo le risultanti e non la distribuzione dei carichi applicati alle basi. Si assume l'area laterale della trave scarica, considerando $\mathbf{n}$ come il versore normale alla superficie uscente

$$
\begin{aligned}
\mathbf{t}_n = \{\sigma_n, \tau_{nz}, \tau_{nm}\}^T = \mathbf{0} \\
\mathbf{n} = \{\alpha_x, \alpha_y, 0\}^T
\end{aligned}
\tag{5.9}
$$

Tramite la relazione $\tau_{nz} = \tau_{zn}$ si evince quindi che anche $\tau_{zn} = 0$. Ne segue che il vettore tensione tangenziale relativo ad un taglio perpendicolare a z, ovvero $\tau_z = \{\tau_{zn}, \tau_{zm}\}^T = \{0, \tau_{zm}\}^T$ è parallelo al bordo stesso (Fig. 5.4).

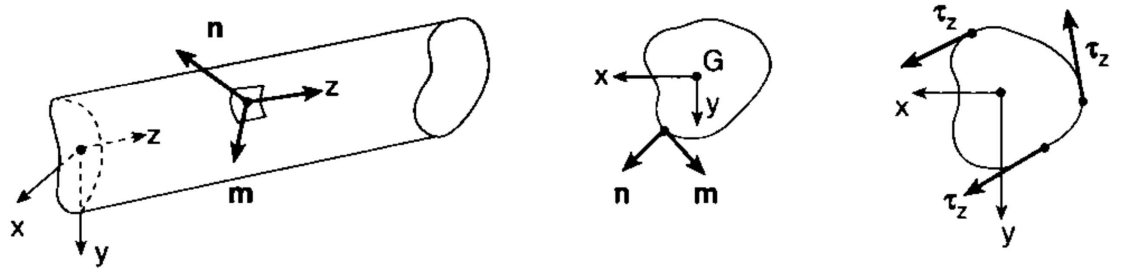


Figura 5.4: Vettore τ_z parallelo al bordo della sezione della trave [31]

Il postulato di DSV, su cui si basa il problema stesso, afferma che *"Se sulle basi si applicano distribuzioni di tensioni diverse ma aventi le stesse risultanti, oltre ad una certa distanza dalle basi stesse, detta 'distanza di estinzione', lo stato tensionale è sostanzialmente identico"* [31].

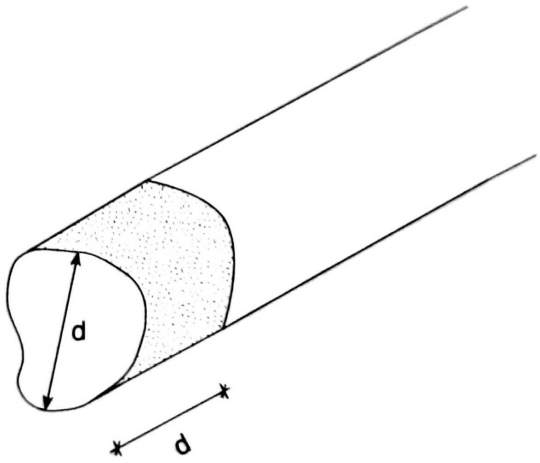


Figura 5.5: Zona di estinzione per una trave di materiale isotropo [31]

Questo postulato è molto importante, poiché garantisce che la soluzione del problema di DSV, per una data risultante di sforzo applicato, sia valida per qualsiasi distribuzione di stress avente la medesima risultante; si può dire che ciò è una diretta conseguenza del fatto che la distribuzione delle tensioni applicate influenza solo le zone della trave vicine alle estremità (Fig. 5.5). Per travi isotrope, la distanza di estinzione viene generalmente approssimata in modo soddisfacente assumendo che sia pari alla dimensione caratteristica massima della sezione trasversale. Pertanto, maggiore è la lunghezza della trave, maggiore sarà anche la porzione di trave coperta dalla soluzione del problema di DSV, minimizzando le zone di estinzione in prossimità delle basi.

5.2.1 Ipotesi semi-inversa

Per giungere alla soluzione del problema bisogna applicare la cosiddetta *ipotesi semi-inversa* che impone di cercare la soluzione nella forma

$$\sigma_x = \sigma_y = \tau_{xy} = 0 \tag{5.10}$$

Questa ipotesi annulla tutte le componenti di $\mathbf{T}$ ad esclusione di $\mathbf{t}_z$, dando maggior importanza al comportamento longitudinale rispetto a quello trasversale. Fisicamente, ciò che detta l'ipotesi semi-inversa è che per piani di sezione paralleli all'asse della trave, il vettore tensione sia diretto lungo z. Applicata l'ipotesi, il tensore della tensione risulta

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & \tau_{zx} \\ 0 & 0 & \tau_{zy} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \tag{5.11}$$

Lo stato deformativo del problema si riduce quindi nella forma

$$\begin{gathered} \varepsilon_{zz} = \frac{\sigma_z}{E}, \quad \varepsilon_{xx} = \varepsilon_{yy} = -\nu\,\varepsilon_z = -\nu\,\frac{\sigma_z}{E}, \\ \varepsilon_{xy} = 0, \quad \varepsilon_{xz} = \frac{\tau_{xz}}{2G}, \quad \varepsilon_{yz} = \frac{\tau_{yz}}{2G}. \end{gathered} \tag{5.12}$$

Di seguito all'introduzione del problema fin qui descritto, la soluzione dipenderà solo dalle risultanti applicate alle basi. I due tagli T_x e T_y meccanicamente analoghi sono riassunti nel problema del taglio e flessione; i due momenti M_x e M_y, anch'essi meccanicamente analoghi, sono riassunti nel problema della flessione; la forza assiale N è descritta nel problema della forza normale e il momento torcente M_t nel problema della torsione. Considerando la linearità dei problemi appena presentati, e quindi dell'applicabilità del principio di sovrapposizione, di seguito verranno studiati separatamente. Al fine di testare le funzionalità di una PINN, per praticità, si tratteranno solo i casi della forza normale e della flessione.

5.3 Trave di DSV - Forza Normale

Il caso della forza normale non presenta tensioni tangenziali e la N si distribuisce uniformemente sulla sezione di area A, ovvero

$$\tau_{xz} = \tau_{yz} = 0, \quad \sigma_z = \frac{N}{A} \tag{5.13}$$

La σ_z è costante lungo tutta la trave, quindi non dipende né da x, né da y e neanche da z. Inserendo le Eq. 5.13 nelle Eq. 5.8 otteniamo

$$\begin{gathered} T_x = T_y = M_t = 0, \\ M_x = \frac{N}{A} S_x = 0, \\ M_y = -\frac{N}{A} S_y = 0, \\ N(z) = \int_A \sigma_z \, dA = \frac{N}{A} \int_A dA = \frac{N}{A} A = N, \end{gathered} \tag{5.14}$$

dove $S_x = S_y = 0$ perchè x e y sono direzioni principali d'inerzia. In conclusione quindi otteniamo come risultante solo la forza normale.

Combinando le Eq. 5.12 e 5.13 otteniamo le componenti del tensore della deformazione

$$\varepsilon_{zz} = \frac{N}{EA}, \quad \varepsilon_{xx} = \varepsilon_{yy} = -\nu \frac{N}{EA}, \qquad \varepsilon_{xy} = \varepsilon_{zx} = \varepsilon_{zy} = 0 \tag{5.15}$$

che ci suggerisce la deformazione che subirà la trave (Fig. 5.6)

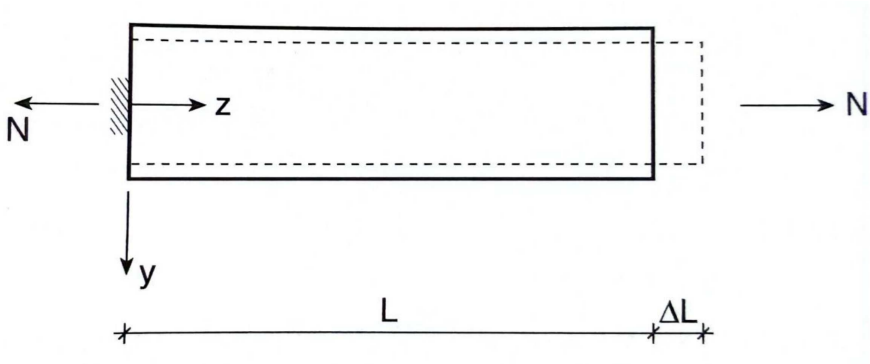


Figura 5.6: Deformazione della trave dovuta ad uno sforzo assiale [31]

Isolando un concio infinitesimo di trave, si possono notare l'allungamento assiale e la contrazione trasversale dovuta all'effetto di Poisson che compongono la deformazione nel pezzo. Essendo dipendenti da ε_z, ed essendo quest'ultimo costante, ne risulta che tutti i punti di una sezione subiscono lo stesso spostamento, mantenendo quindi la sezione piana e parallela a quella iniziale.

Per ottenere una soluzione ingegneristicamente accettabile ad un problema più generale di quello appena presentato, si estende la soluzione di DSV a partire dai suoi elementi caratterizzanti.

In primo luogo si continuano a ritenere valide le relazioni dell'Eq. 5.13 anche in situazioni in cui N ed A variano lungo z:

$$\tau_{xz} = \tau_{xy} = 0, \quad \sigma_z(z) = \frac{N(z)}{A(z)} \tag{5.16}$$

In secondo luogo si tratta il problema facendo riferimento all'asse della trave ma tenendo conto della deformabilità, definendo grandezze medie sulle sezioni capaci di rappresentare il comportamento della trave. Per la statica questa grandezza è $N(z)$, mentre per la cinematica sono la deformazione longitudinale e lo spostamento longitudinale medi

$$\varepsilon_m(z) = \frac{1}{A}\int_A \varepsilon_{zz}(x,y,z)\,dA \qquad w(z) = \frac{1}{A}\int_A w(x,y,z)\,dA \tag{5.17}$$

Integrando sull'area l'equazione di congruenza $\varepsilon_{zz} = \partial w/\partial z$ si ottiene

$$\varepsilon_m(z) = \frac{dw}{dz}(z) = w^{'}(z) \tag{5.18}$$

5.3.1 Problema elastico

Sono stati presentati, quindi, tutti gli strumenti utili a ricavare le soluzioni al problema della forza assiale appena descritto.

Dati	Incognite	Equazioni	
L		$w^{'} = \varepsilon_m$	congruenza
A(z)	$w(z)$	$N = EA\varepsilon_m$	legame costitutivo
E(z)	$\varepsilon_m(z)$	$N^{'} = -q$	equilibrio
q(z)	$N(z)$	$w = w_o$	b.c. cinematica
b.c.		$N = N_o$	b.c. statica

Tabella 5.1: Dati, incognite ed equazioni del problema elastico nel caso di forza assiale

La relazione $N^{'} = -q$ deriva direttamente dallo studio dell'equilibrio delle travi [31]; N_o e w_o si riferiscono a carichi e a spostamenti assiali imposti alle estremità.

Dalle equazioni di congruenza e legame costitutivo in Tab. 5.1 si ottiene la relazione

$$N = EAw^{'}, \tag{5.19}$$

cui si associano le condizioni al contorno $w = w_o$. Inserendo l'Eq. 5.19 all'interno dell'equazione di equilibrio si ottiene

$$(EAw^{'})^{'} = -q, \tag{5.20}$$

cui si associano le condizioni al contorno cinematiche e statiche, $w = w_o$ e $EAw^{'} = N_o$, che consentono di determinare w anche in strutture iperstatiche.

Considerando EA costante, il problema diventa

$$\begin{gathered} w^{''} = -\frac{q}{EA}, \\ w = w_o \quad \text{e/o} \quad w^{'} = \frac{N_o}{EA} \quad \text{al contorno} \end{gathered} \tag{5.21}$$

5.3.2 Soluzione al problema elastico

Si prenda in considerazione una trave a sbalzo con un carico assiale distribuito lungo la coordinata z e costante ed una forza assiale concentrata F all'estremità, tutte a trazione. Dato che q è costante, si integra l'Eq. 5.21 due volte, ottenendo

$$w(z) = -\frac{q}{EA}\frac{z^2}{2} + c_1 z + c_2. \tag{5.22}$$

In $z = 0$ lo spostamento è nullo, da cui si determina la costante c_2

$$0 = w(0) = c_2 \tag{5.23}$$

mentre in $z = L$ la forza normale è F, perciò

$$\frac{F}{EA} = w^{'}(L) = -\frac{q}{EA}L + c_1 \quad \Rightarrow \quad c_1 = \frac{F}{EA} + \frac{q}{EA}L. \tag{5.24}$$

La soluzione risulta quindi:

$$w(z) = \frac{q}{EA}\left(Lz - \frac{z^2}{2}\right) + \frac{F}{EA}z \tag{5.25}$$

5.4 Trave di DSV - Flessione semplice retta

Prendendo in esame la trave di DSV, se a questa vengono applicati due momenti M_x uguali ed opposti alle estremità, risulta la deformata illustrata in Fig. 5.7, ove si nota che a causa della simmetria la sezione di mezzeria non si sposta lungo z rimanendo piana.

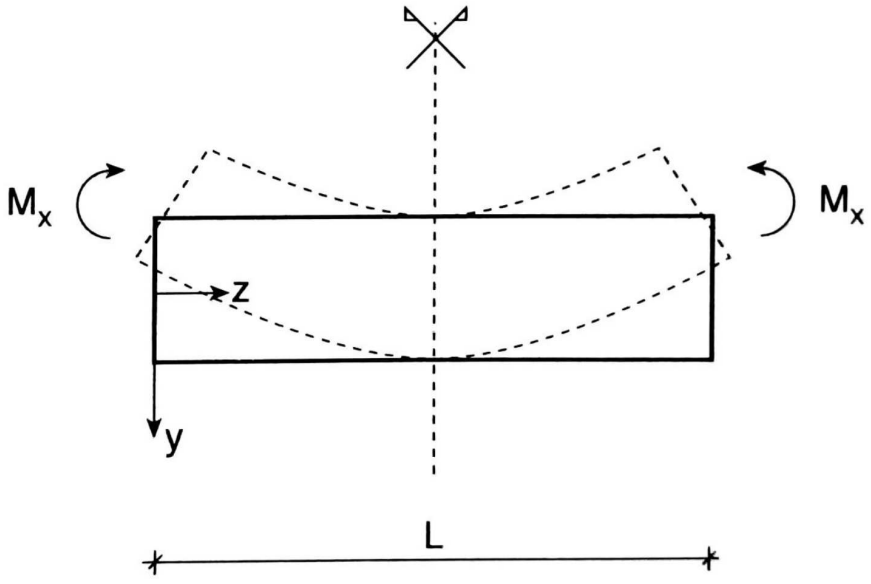


Figura 5.7: Trave sottoposta a flessione [31]

Alla deformazione precedente si può aggiungere un moto rigido, che per definizione non altera la deformazione, in modo da verificare la condizione di incastro all'origine (Fig. 5.8).

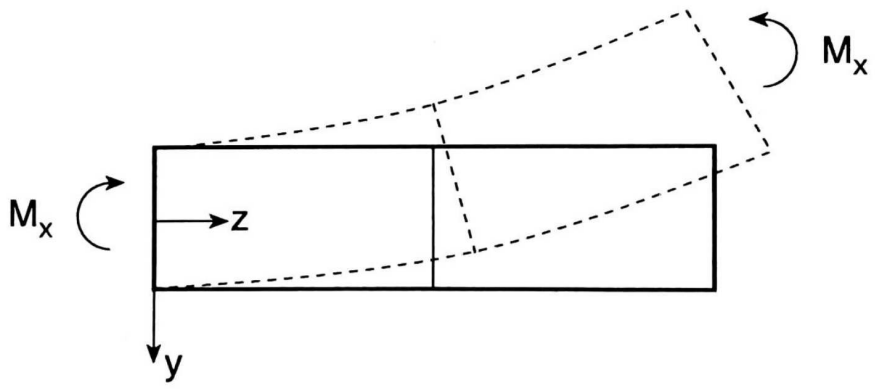


Figura 5.8: Trave sottoposta a flessione e rotazione rigida [31]

Ora si pensi di dividere a metà la trave ed applicare la medesima coppia di momenti M_x ad entrambe le semi-travi $0 < z < L/2$ e $L/2 < z < L$: il comportamento di questi tronchi sarà il medesimo di Fig. 5.7. Aggiungendovi traslazioni e rotazioni rigide, si uniscono le estremità dei due tronchi e si ottiene la configurazione rappresentata in Fig. 5.9.

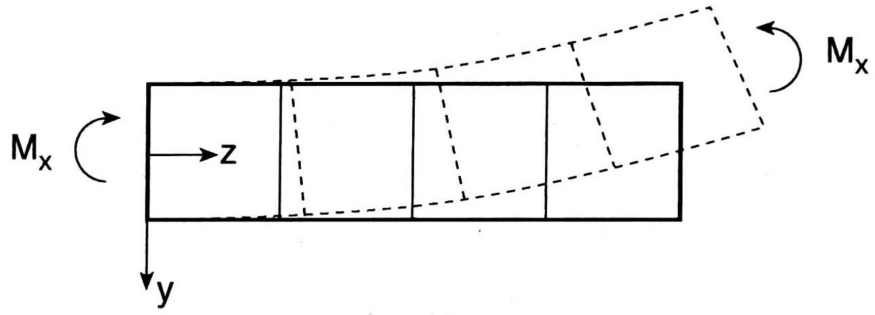


Figura 5.9: Due semitravi sottoposte a flessione e rotazione rigida [31]

Iterando il ragionamento si può concludere che tutte le sezioni rimangono piane ad ogni suddivisione della trave originaria, traslando e ruotando rigidamente i pezzi a deformazione avvenuta. Questo porta a pensare che un concio di trave si deforma alla stessa maniera (Fig. 5.10).

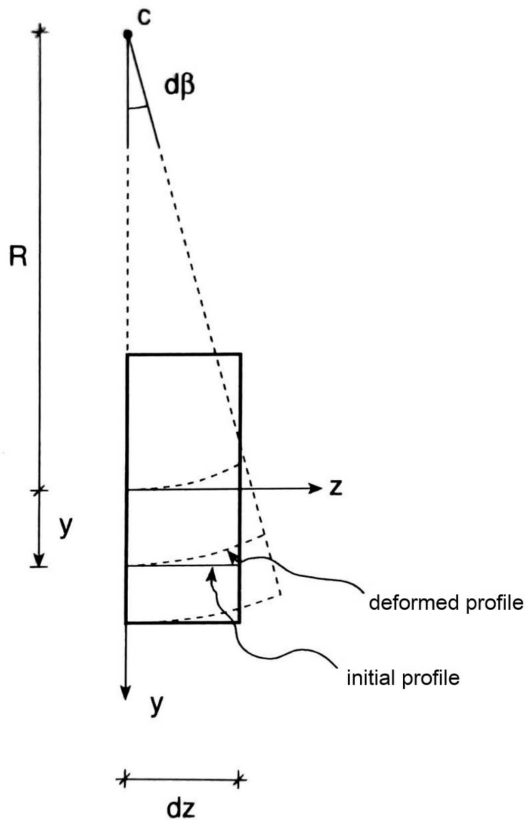


Figura 5.10: Deformazione dovuta alla flessione di un concio di trave [31]

L'asse inizialmente rettilineo si trasforma in un arco di circonferenza di raggio R con centro in C. La Fig. 5.10 mostra che $dz = R\,d\beta$ ed è quindi possibile identificare l'allungamento assiale tramite le lunghezze iniziali l_i e finali l_f delle fibre

$$\begin{aligned} l_i &= dz = R\,d\beta \\ l_f &= (R+y)\,d\beta \end{aligned} \quad \Longrightarrow \varepsilon_{zz} = \frac{l_f - l_i}{l_i} = \frac{(R+y)\,d\beta - R\,d\beta}{R d\beta} = \frac{y}{R} = \kappa_x\, y \tag{5.26}$$

dove $\kappa_x = 1/R$ è la curvatura nel piano perpendicolare a x. Tramite l'Eq. 5.12 si definisce quindi la tensione normale

$$\sigma_z = E\,\kappa_x\, y \tag{5.27}$$

e le tensioni tangenziali che, come nel casi di forza assiale, sono nulle. Di conseguenza utilizzando le Eq. 5.8 si ha $T_x = T_y = M_t = 0$, mentre

$$N = \int_A \sigma_z \, dA = E \, \kappa_x \int_A y dA = E \, \kappa_x \, S_x = 0$$

$$M_x = \int_A y \sigma_z \, dA = E \, \kappa_x \int_A y^2 dA = E \, \kappa_x \, I_x \tag{5.28}$$

$$M_y = -\int_A x \sigma_z \, dA = E \, \kappa_x \int_A xy dA = E \, \kappa_x \, I_{xy} = 0$$

dove $S_x = S_y = 0$ perchè x e y sono direzioni principali d'inerzia. Ne evince quindi che

$$\kappa_x = \frac{1}{R} = \frac{M_x}{EI_x} \quad \Longrightarrow \quad \sigma_z = \frac{M_x}{I_x} y \tag{5.29}$$

da cui è possibile determinare lo stato tensionale in funzione dell'azione interna. Inoltre questo dimostra che applicare una curvatura è più facile per travi poco rigide (con un valore di EI_x basso), mentre per travi più rigide è necessario applicare un momento più elevato che può portare a rottura la trave.

L'Eq. 5.29 dice che tutti i conci della trave si incurvano della stessa quantità, dovuto dal fatto che M_x, E ed I_x sono costanti lungo z: quindi la deformata è un arco di circonferenza di raggio R.

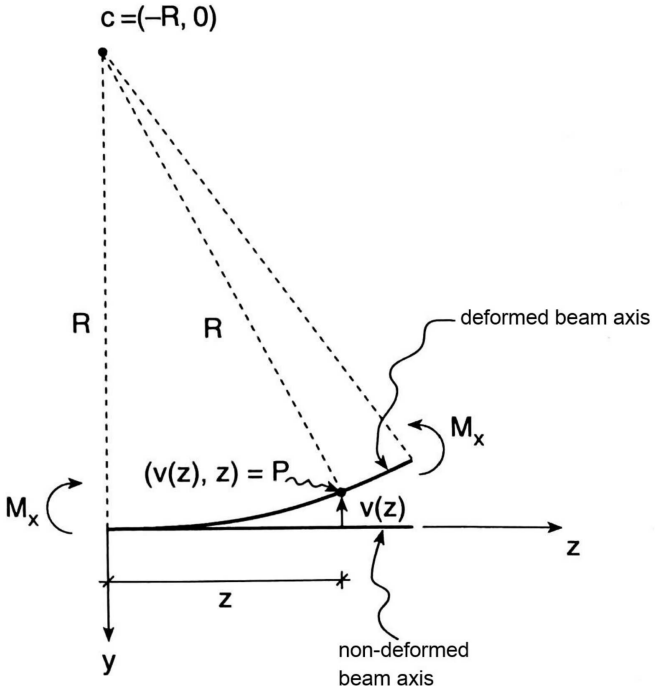


Figura 5.11: Deformata flessionale della trave [31]

Imponendo che la distanza tra C e il generico punto P sia pari a R, l'espressione analitica della deformata risulta

$$R^2 = d^2(P,C) = (v(z)+R)^2 + z^2 \quad \Longrightarrow \quad 0 = v^2(z) + 2\,v(z)\,R + z^2. \tag{5.30}$$

Secondo l'ipotesi di piccoli spostamenti il termine $v^2(z)$ risulta trascurabile, che porta a definire $v(z)$ come

$$v(z) = -\frac{z^2}{2R} = -\frac{1}{2}\,\kappa_x\,z^2 = -\frac{1}{2}\frac{M_x}{EI_x}z^2 \tag{5.31}$$

Derivando la precedente relazione si ottiene l'angolo di inclinazione che si intende positivo se antiorario,

$$\phi(z) = -v^{'}(z) = \kappa_x\,z = \frac{M_x}{EI_x}z. \tag{5.32}$$

da cui, derivando un'ulteriore volta, si ottiene l'espressione che lega la curvatura agli spostamenti

$$v^{''}(z) = -\kappa_x. \tag{5.33}$$

Anche per il problema del caso flessionale la soluzione di DSV va estesa con lo stesso criterio del caso di forza assiale. Sempre riferendosi all'asse della trave, viene mantenuto il medesimo stato tensionale, in modo da poter determinare il tensore della tensione conoscendo l'azione interna

$$\tau_{xz} = \tau_{yz} = 0, \quad \sigma_z(z) = \frac{M(z)}{I(z)}y \tag{5.34}$$

A livello deformativo il momento flettente incurva la trave, perciò il parametro descrittivo del problema è la curvatura $\kappa(z)$. Continuando a ritenere valida l'Eq. 5.29 anche quando tutte le grandezze variano lungo z si ha l'equazione di legame costitutivo

$$\kappa(z) = \frac{M(z)}{E(z)I(z)}, \tag{5.35}$$

mentre l'equazione di congruenza è estesa al caso di curvatura variabile

$$v^{''}(z) = -\kappa(z). \tag{5.36}$$

L'equazione di equilibrio deriva direttamente dallo studio dell'equilibrio delle travi [31],

$$M^{''}(z) = -p(z) - c^{'}(z) \tag{5.37}$$

ove $p(z)$ rappresenta il carico trasversale distribuito mentre $c(z)$ i momenti per unità di lunghezza applicati.

5.4.1 Problema elastico

Sono stati presentati, quindi, tutti gli strumenti utili a ricavare le soluzioni al problema della flessione semplice retta.

Dati	Incognite	Equazioni	
L		$v'' = -\kappa$	congruenza
I(z)	$v(z)$	$M = EI\,\kappa$	legame costitutivo
E(z)	$\kappa(z)$	$M'' = -p - c'$	equilibrio
p(z) e c(z)	$M(z)$	$v = v_o; \quad v' = -\phi_o$	b.c. cinematica
b.c.		$T = T_o; \quad M = M_o$	b.c. statica

Tabella 5.2: Dati, incognite ed equazioni del problema elastico nel caso di flessione semplice retta

Dalle equazioni di congruenza e legame costitutivo in Tab. 5.2 si ottiene la relazione

$$-M = EIv'' \tag{5.38}$$

alla quale si associano le condizioni al contorno $v = v_o$ e $v' = -\phi_o$. Inserendo l'Eq. 5.38 all'interno dell'equazione di equilibrio si ottiene quella che viene definita l'*equazione differenziale della linea elastica flessionale del 4° ordine*:

$$(EIv'')'' = p + c' \tag{5.39}$$

cui si associano sia le condizioni al contorno cinematice scritte in precedenza che le condizioni al contorno statiche $-EIv'' = M_o$ e $-(EIv'')' = T_o - c$ che permette di determinare la $v(z)$ anche per strutture iperstatiche.

Considerando EI costante e $c = 0$, il problema elastico diventa

$$\begin{gathered} v'''' = \frac{p}{EI}, \\ v = v_o \quad \text{e/o} \quad v' = -\phi_o \quad \text{e/o} \quad v'' = -\frac{M_o}{EI} \quad \text{e/o} \quad v''' = -\frac{T_o}{EI} \quad \text{al contorno.} \end{gathered} \tag{5.40}$$

5.4.2 Soluzione al problema elastico

Si consideri un caso estremamente generico in cui ad una trave a sbalzo viene applicato un taglio T ed un momento M all'estremità, mentre in tutta la trave vi è una distribuzione costante di forze trasversali p (si utilizza $c = 0$ in quanto è solitamente assente nei casi pratici). Dato che p è costante, si integra l'Eq. 5.40 quattro volte, ottenendo

$$v(z) = \frac{p}{EI}\frac{z^4}{24} + c_1\frac{z^3}{6} + c_2\frac{z^2}{2} + c_3 z + c_4. \tag{5.41}$$

In $z = 0$ lo spostamento e la rotazione sono nulli, da cui si determinano le costanti c_4 e c_3

$$\begin{aligned} 0 &= v(0) = c_4, \\ 0 &= v^{'}(0) = c_3. \end{aligned} \tag{5.42}$$

Mentre in $z = L$ la forza trasversale è pari a T e il momento è pari a M

$$\begin{aligned} -\frac{M}{EI} &= v^{''}(L) = \frac{p}{EI}\frac{L^2}{2} + c_1 L + c_2, \\ -\frac{T}{EI} &= v^{'''}(L) = \frac{p}{EI}L + c_1 \end{aligned} \tag{5.43}$$

da cui si determinano le costanti c_1 e c_2

$$\begin{aligned} c_1 &= -\frac{(T + pL)}{EI}, \\ c_2 &= -\frac{(M - TL - pL^2/2)}{EI}. \end{aligned} \tag{5.44}$$

La soluzione risulta quindi

$$v(z) = \frac{p}{EI}\frac{z^4}{24} - \frac{(T + pL)}{EI}\frac{z^3}{6} - \frac{(M - TL - pL^2/2)}{EI}\frac{z^2}{2}. \tag{5.45}$$

Capitolo 6

Non linearità geometriche, grandi spostamenti

Nella presentazione del problema di DSV viene considerata una trave che risponde linearmente ai carichi applicati, in quanto vi è l'ipotesi di piccoli spostamenti che esclude la non linearità del problema. Estendendo lo studio della trave a sbalzo, possiamo formulare delle equazioni descrittive per il problema dei grandi spostamenti.

Consideriamo una lunga e sottile trave a sbalzo di sezione rettangolare costante, costituita da un materiale lineare elastico, omogeneo ed isotropo, che risponde alla legge di Hooke [31]. La deflessione di una trave a sbalzo è essenzialmente un problema tridimensionale: uno stiramento elastico in una direzione è accompagnato da una compressione nelle direzioni perpendicolari. Tuttavia, possiamo ignorare questo effetto in quanto è tanto trascurabile quanto più è elevata la snellezza della trave. In questo studio assumiamo che la trave non sia estensibile, che le deformazioni rimangano piccole e che l'ipotesi di Eulero-Bernoulli sia valida, ovvero che le sezioni trasversali piane che sono perpendicolari all'asse neutro prima della deformazione vi rimangano a deformazione avvenuta. Inoltre, assumiamo che le sezioni piane non cambino forma o area [6]. É possibile scrivere la relazione che lega il momento flettente e la curvatura di Eulero-Bernoulli come segue:

$$EI\,\frac{d\varphi}{ds} = M \tag{6.1}$$

dove M è il momento flettente (positivo in senso orario), $\kappa = d\varphi/ds$ è la curvatura in ogni punto della trave ed EI è la rigidezza flessionale considerata costante lungo la trave. Considerando s l'ascissa curvilinea, viene definito $\psi_{in}(s)$ l'angolo che fornisce la direzione iniziale della tangente in un punto qualsiasi rispetto all'asse x e $\psi(s)$ l'angolo a deflessione avvenuta, così da considerare l'angolo di curvatura come

$$\varphi(s) = \psi(s) - \psi_{in}(s) \tag{6.2}$$

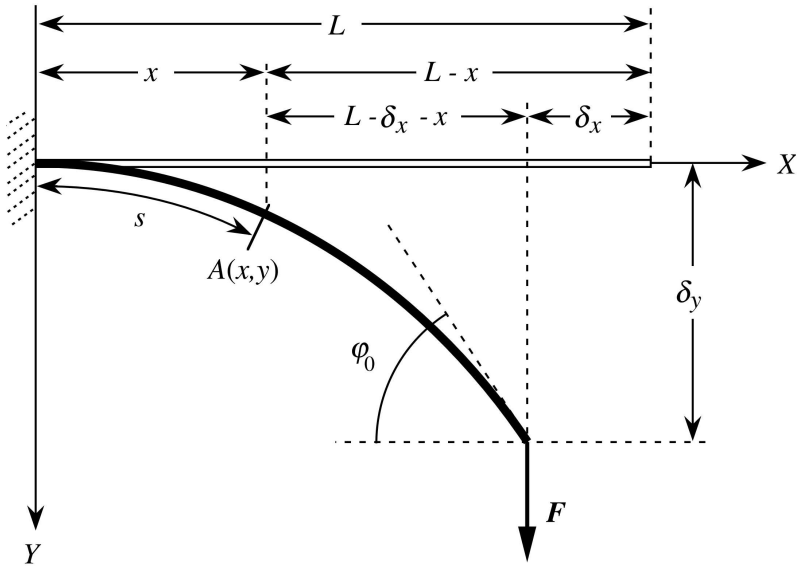


Figura 6.1: Deformata di una trave a sbalzo rettilinea a sezione costante [6]

La Fig. 6.1 mostra una trave a sbalzo rettilinea di lunghezza L con un carico concentrato F applicato all'estremità libera della trave. Qui i valori δ_x e δ_y sono rispettivamente gli spostamenti orizzontali e verticali all'estremità libera e φ_0 tiene conto della pendenza massima della trave [6]. Lo studio dell'equilibrio statico della trave viene rappresentato dalle equazioni che sfruttano la rappresentazione geometrica fornita in Fig. 6.2.

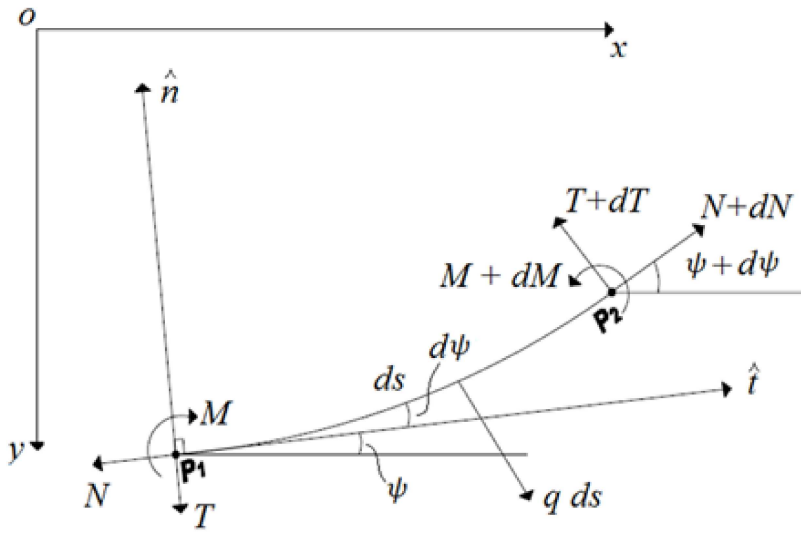


Figura 6.2: Concio di trave infinitesimo [21]

Isolando un tratto di curva tra i punti P_1 e P_2, ove $\|\overline{P_1P_2}\| = ds$, ed essendo: N la forza normale, T la forza di taglio, M il momento flettente e q il vettore dei carichi distribuiti, l'equilibrio statico alla traslazione sulle direzioni x e y e alla rotazione attorno a z assume la forma:

$$\begin{cases} \dfrac{d}{ds}[N\cos(\psi)] - T\sin(\psi)] + q_x = 0 \\ \dfrac{d}{ds}[N\sin(\psi)] + T\cos(\psi)] - q_y = 0 \\ \dfrac{dM}{ds} + T = 0 \end{cases} \tag{6.3}$$

Integrando le prime due equazioni da 0 alla generica ascissa curvilinea s, moltiplicando la prima per $\sin(\psi)$ e la seconda per $\cos(\psi)$, sommandole e sostituendole alla terza si ottiene una nuova equazione di equilibrio flessionale:

$$\begin{aligned} \frac{dM}{ds} + \Big[(T\sin\psi)\Big|_{s=0} - (N\cos\psi)\Big|_{s=0} + q_x s\Big]\sin(\psi) + \\ + \Big[(T\cos\psi)\Big|_{s=0} + (N\sin\psi)\Big|_{s=0} + q_y s\Big]\cos(\psi) = 0 \end{aligned} \tag{6.4}$$

Unendo le Eq. 6.1 e 6.4 si ottiene la generica equazione differenziale dell'equilibrio statico; nel caso di una trave a sbalzo $\psi|_{s=0} = 0$, soggetta ad un sistema di carichi concentrati all'estremità libera $\{F_x, F_y, M\}$ e carichi distribuiti lungo la trave $\{q_x, q_y\}$, è

$$EI\left(\frac{d^2\psi}{ds^2} - \frac{d^2\psi_{in}}{ds^2}\right) + [\pm F_x \pm q_x s]\sin(\psi) + [\pm F_y \pm q_y s]\cos(\psi) = 0, \tag{6.5}$$

dove F_x e q_x sono considerati positivi se diretti in direzione opposta all'asse x, mentre F_y e q_y sono considerati positivi se diretti come l'asse y (Fig. 6.1 per il riferimento degli assi). Trattandosi di una trave rettilinea, ovvero $\psi_{in}(s) = 0$, abbiamo che $\varphi(s) = \psi(s)$, da cui risulta

$$EI\frac{d^2\varphi}{ds^2} + [\pm F_x \pm q_x s]\sin(\varphi) + [\pm F_y \pm q_y s]\cos(\varphi) = 0, \tag{6.6}$$

Le condizioni al contorno dell'Eq. 6.6 sono date dalla conoscenza di φ e φ' in punti specifici: in questo caso l'incastro impone $\varphi(0) = 0$ mentre la seconda condizione è lineare all'estremità libera e impone $\varphi'(L) = M$ con M il momento flettente applicato. La rislouzione dell'Eq. 6.6 permette di ottenere l'angolo φ, ovvero di determinare le componenti del versore tangente lungo tutta la curva $\mathbf{t}(s) = [\cos(\varphi(s)), \sin(\varphi(s))]$. Considerando la geometria della trave in questione incastrata nell'origine $x(0) = y(0) = 0$, è possibile integrare le componenti del versore tangente, ottenendo così la nuova geometria parametrica:

$$x(s) = \int_0^s \cos(\varphi)\, ds \tag{6.7}$$

$$y(s) = \int_0^s \sin(\varphi)\, ds \tag{6.8}$$

Capitolo 7

Implementazione numerica

In questo capitolo vengono esposte le implementazioni per i singoli problemi descritti, nonchè il linguaggio di programmazione scelto per lo studio, ovvero Python, e l'ambiente di sviluppo utilizzato, ovvero Google Colaboratoty.

7.1 Python

Python è un linguaggio di programmazione di alto livello e di uso generale, la cui filosofia di progettazione enfatizza la leggibilità del codice con l'uso di identazioni significative [28]. Viene costantemente considerato uno dei linguaggi di programmazione più popolari e diffusi [37], e ha guadagnato notorietà nella facilità d'uso nella comunità del ML. L'ampia libreria standard di Python fornisce strumenti adatti a molti compiti ed è comunemente citata come uno dei suoi maggiori punti di forza. Nonostante ciò, lo sviluppo di librerie di terze parti è in ampia espansione, infatti il Python Package Index (PyPI), il repository ufficiale per il software Python di terze parti, contiene centinaia di migliaia di pacchetti con un'ampia gamma di funzionalità [22].

Nell'ambito del ML, vi sono innumerevoli strumenti basati sul linguaggio Python, creati ad hoc per operare in tutto il campo dell'intelligenza artificiale. Sono noti i più famosi framework open-source creati e sviluppati nel corso degli anni: Keras, TensorFlow e PyTorch. Per l'utilizzo che ne viene fatto nel corso di questo studio, il framework scelto non influisce i risultati o le performance del codice: pertanto, secondo un parere puramente personale, per una maggiore comprensione della documentazione proposta, PyTorch è il framework che verrà utilizzato per l'implementazione degli algoritmi.

7.2 PyTorch

PyTorch è un framework di apprendimento automatico basato sulla libreria Torch, utilizzata per applicazioni come la visione artificiale e l'elaborazione del linguaggio naturale, originariamente sviluppata da Meta AI. È un software gratuito e open source, su cui sono basati numerosi software di deep learning, tra cui Tesla Autopilot [25] e Uber's Pyro [18]. Inoltre PyTorch offre due funzionalità importanti:

1. Calcolo tensoriale con una potente accelerazione harware tramite le unità di elaborazione grafica, o Graphic Processing Unit (GPU)

2. Reti neurali profonde, o Deep Neural Network (DNN), costruite su un sistema di differenziazione automatica basate sui grafi e sulla "regola della catena" [12]

All'interno della libreria vi sono già definiti i modelli per la creazione di reti neurali multilayers; sono presenti molteplici funzioni di attivazione tra cui quelle presentate in Tab. 2.1; sono già implementati i principali algoritmi di ottimizzazione oltre a tutti i molteplici strumenti utili ad operare con i tensori [13].

Per la scrittura e l'esecuzione del codice è stato preso in considerazione di utilizzare il servizio di Google Colab, in quanto usufruendo del servizio non è necessaria l'installazione di un editor e di un compilatore. Colab è un servizio notebook Jupyter, ovvero un'applicazione web che consente di creare e condividere documenti che contengono codice live, equazioni e altre risorse multimediali, di Google che non richiede alcuna configurazione per l'utilizzo e fornisce accesso gratuito alle risorse di calcolo, incluse GPU. La semplicità e la potenzialità del servizio rendono Colab la soluzione ottimale per lo sviluppo del progetto.

7.3 Impostazione dei problemi

Ogni problema che si andrà ad affrontare presenterà delle caratteristiche similari e i medesimi approcci nell'ottimizzazione delle reti neurali. Partendo da quest'ultime, tutte le reti neurali sono pensate alla stessa maniera:

1. L'input layer presenta coordinate spaziali (z nei casi monodimensionali, mentre x, y e z nei casi tridimensionali). É un layer esente da funzioni di attivazione;

2. L'output layer presenta le variabili del problema, quindi spostamenti e/o sforzi (w nel problema assiale di DSV, v nel problema flessionale di DSV, φ nel problema flessionale di grandi spostamenti, mentre u, v, w, σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{yz} e σ_{xz} per il problema tridimensionale). É un layer esente da funzioni di attivazione;

3. Gli hidden layers presentano tutti la tangente iperbolica (tanh) come funzione di attivazione. Questi layers variano in quantità e in numero di neuroni presenti in base al problema in studio.

Lo schema che le rappresenta è composto come segue

$$i - h - \ldots - o \tag{7.1}$$

dove i definisce il numero di input, o definisce il numero di output e h definisce il numero di neuroni presenti nell'hidden layer. Per esempio $1 - 5 - 1$ definisce una rete con 1 input, 1 output e 5 neuroni nell'unico hidden layer, mentre $3 - 20 - 20 - 6$ definisce una rete con 3 input, 6 output e 20 neuroni per 2 hidden layer. I parametri di tutte le reti sono inizializzati con valori random utilizzando il medesimo *seed*; il *seed* è un valore utilizzato per inizializzare un generatore di numeri random. Il training dataset è composto da diversi tipi di dati:

- I collocation points, (cp) distribuiti all'interno di tutto il dominio del problema: questi punti rappresentano delle coordinate spaziali (per problemi 1D, $cp \in \mathcal{R}$; per problemi 3D, $cp \in \mathcal{R}^3$);

- Le condizioni al contorno, ovvero specifici e ben noti collocation points nei quali si conosce il valore della funzione incognita o le relative derivate parziali;

- Le misurazioni, pensate come "classici" data points, formati da coordinate del punto in questione e relativo risultato numerico ottenuto tramite simulazione FEM.

Tutti i problemi in studio utilizzano una funzione di perdita basata sul MSE, mentre l'optimizer utilizzato sarà Adam con un lr $\eta = 0.001$, se non diversamente specificato. Le funzioni di costo sono formate da 2 componenti per il caso 1D, mentre ci saranno 2 varianti per quello 3D:

$$\mathcal{L} = \lambda_b \mathcal{L}_b + \lambda_p \mathcal{L}_p \qquad \text{per il caso 1D,}$$

$$\left.\begin{aligned} \mathcal{L} &= \lambda_d \mathcal{L}_d + \lambda_b \mathcal{L}_b \\ \mathcal{L} &= \lambda_d \mathcal{L}_d + \lambda_b \mathcal{L}_b + \lambda_p \mathcal{L}_p \end{aligned}\right\} \quad \text{per il caso 3D}$$

Nel caso 3D si tenta eseguire un training in un primo tentativo utilizzando solo i datapoint e le condizioni al contorno (denominato "Prova NN"), mentre in un secondo tentativo si aggiunge il contributo dato dalla fisica (denominato "Prova PINN") e si confrontano i due. I coefficienti λ servono per ridimensionare, a livello di ordini di grandezza, le varie porzioni della funzione di costo, stabilizzando e/o incentivando il training della rete. I valori utilizzati non sono frutto di una funzione o di una legge, ma di ripetuti tentativi di rendere più performante il training. Se non diversamente specificato nel corso della tesi, vale per tutti $\lambda = 1$. Quindi, come è intuibile, il termine $\mathcal{L}_b$ fornisce la perdita rispetto alle condizioni al contorno del problema, $\mathcal{L}_p$ fornisce il residuo delle equazioni fisiche che governano il comportamento della soluzione, mentre $\mathcal{L}_d$ fornisce la perdita rispetto ai datapoint ottenuti da simulazioni numeriche.

Per ottenere questi ultimi dati sono effettuate delle analisi FEM tramite la coppia di software Patran-Nastran della suite MSC per i casi 3D lineari e non lineari che verranno affrontati.

7.4 Problema assiale di DSV

Il problema assiale di DSV è un problema semplice, dovuto alla monodimensionalità e alla linearità: note le condizioni al contorno, la soluzione analitica è determinata (Eq. 5.25). Sono effettuate 3 diverse prove con dati del problema differenti:

Caso	L (m)	D (mm)	A (m^2)	E (Pa)	q (N/m)	N_L (N)
1	0.8	20	$3.142 \cdot 10^{-4}$	$7 \cdot 10^{10}$	300	800
2	1.9	40	$1.257 \cdot 10^{-3}$	$7 \cdot 10^{10}$	400	900
3	3.4	70	$3.848 \cdot 10^{-3}$	$7 \cdot 10^{10}$	250	600

Tabella 7.1: Dati di 3 differenti casi del problema assiale di DSV

I primi 4 descrivono la geometria e il materiale in studio, mentre gli ultimi due si riferiscono ai carichi applicati: q è il carico assiale distribuito lungo la trave e N_L è la forza assiale applicata all'estremità libera in $z = L$. In tutte le prove ci si riferisce ad una trave a sbalzo, quindi $w(0) = 0$. L'obiettivo di questo studio è far si che la soluzione $w(z)$ sia rappresentata da una rete neurale, ovvero

$$w(z) \approx NN_w(z,\, \theta). \tag{7.2}$$

Lo schema della rete è 1-10-1 e questa si allena su un training set composto da 100 collocation points equispaziati tra 0 e L, e 2 condizioni al contorno definite sui due punti di estremità presenti nel set. La funzione di costo è definita come

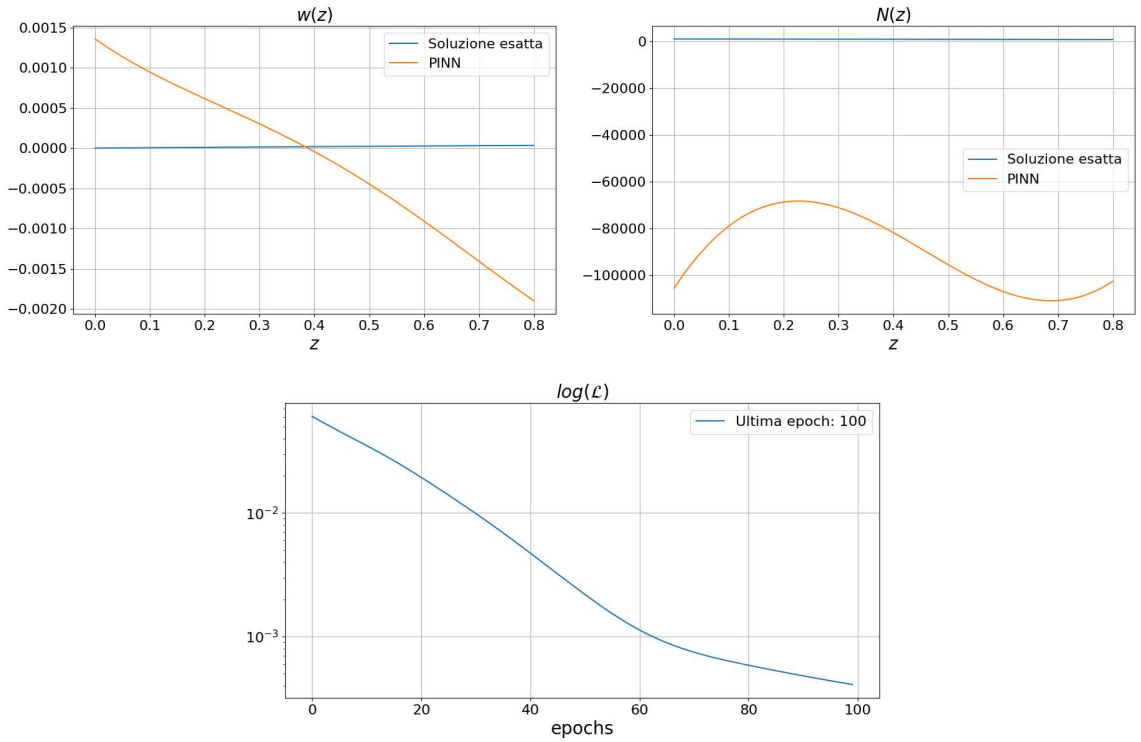
$$\begin{aligned} \mathcal{L} &= \mathcal{L}_b + \mathcal{L}_p, \\ \mathcal{L}_b &= w(0)^2 + \left(w^{'}(L) - \frac{N_L}{EA} \right)^2 \\ \mathcal{L}_p &= \frac{1}{N_p} \sum \left(w^{''}(z) + \frac{q}{EA} \right)^2 \end{aligned} \tag{7.3}$$

dove N_p è il numero di collocation points utilizzati durante la fase di training. La rete viene allenata in un massimo di 10000 epochs e si interrompe quando $\mathcal{L} < 10^{-4}$. I risultati sono poi ottenuti applicando la rete neurale su di un testing dataset di 1000 punti, così da renderne più accurata la rappresentazione. L'output fornisce il valore w in metri, mentre il valore di N, misurato in Newton, deriva dalla relazione

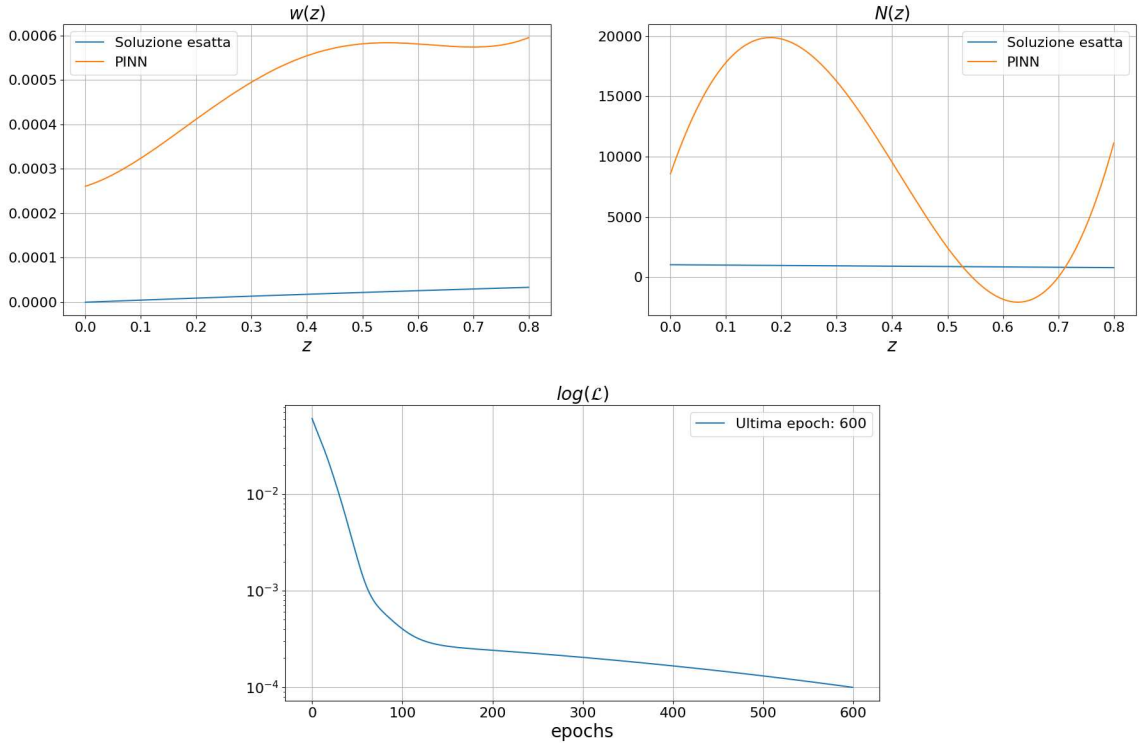
$$N = EAw^{'} \tag{7.4}$$

Come si può notare dalle Fig. 7.1, 7.2 e 7.3 i risultati forniti dalla rete neurale non sono corretti ed è possibile attribuire la causa alla disparità di ordini di grandezza tra input e output che rende sbilanciata la rete. Dal momento che l'ordine di grandezza della soluzione che si cerca è molto piccolo, è doveroso imporre una condizione di fine training più restrittiva, rischiando di entrare nel campo del rumore numerico e di precisione della macchina (il valore più basso che possono contenere i tensori di Pytorch è di 10^{-16}).

Quindi si procede ad una ricerca delle soluzioni adimensionalizzando il problema e introducendo un nuovo parametro N_0. Questo parametro rappresenta il valore della forza assiale all'incastro ed è definito come

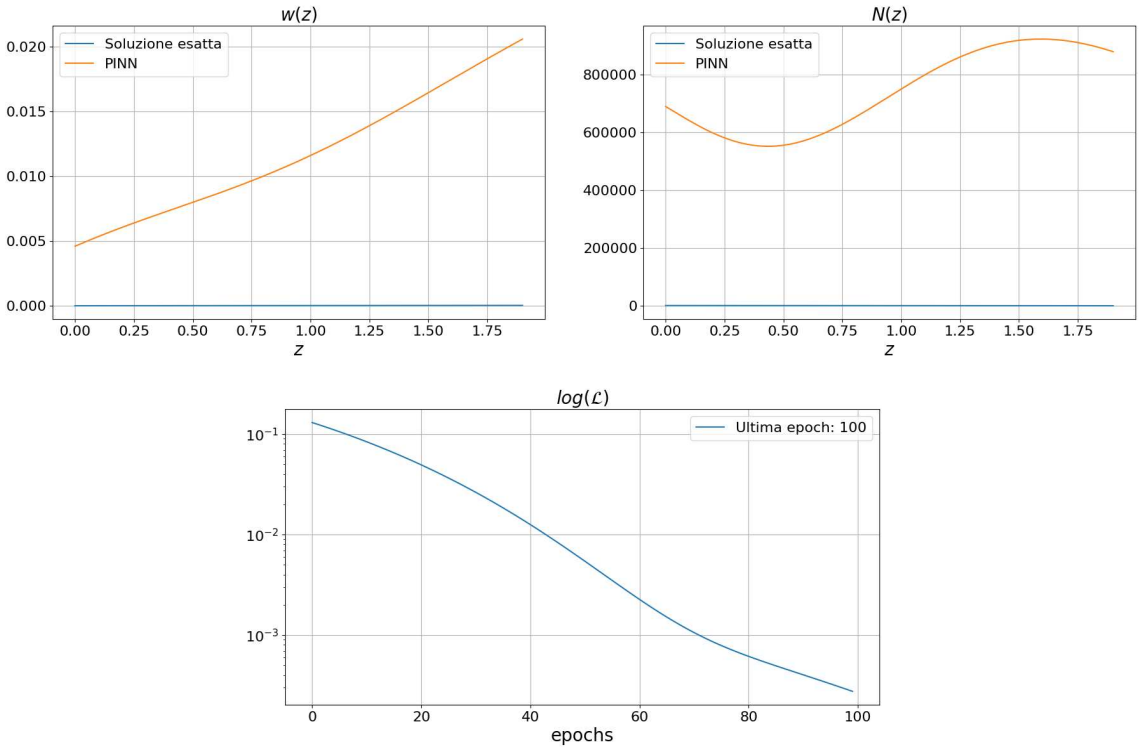


(a)

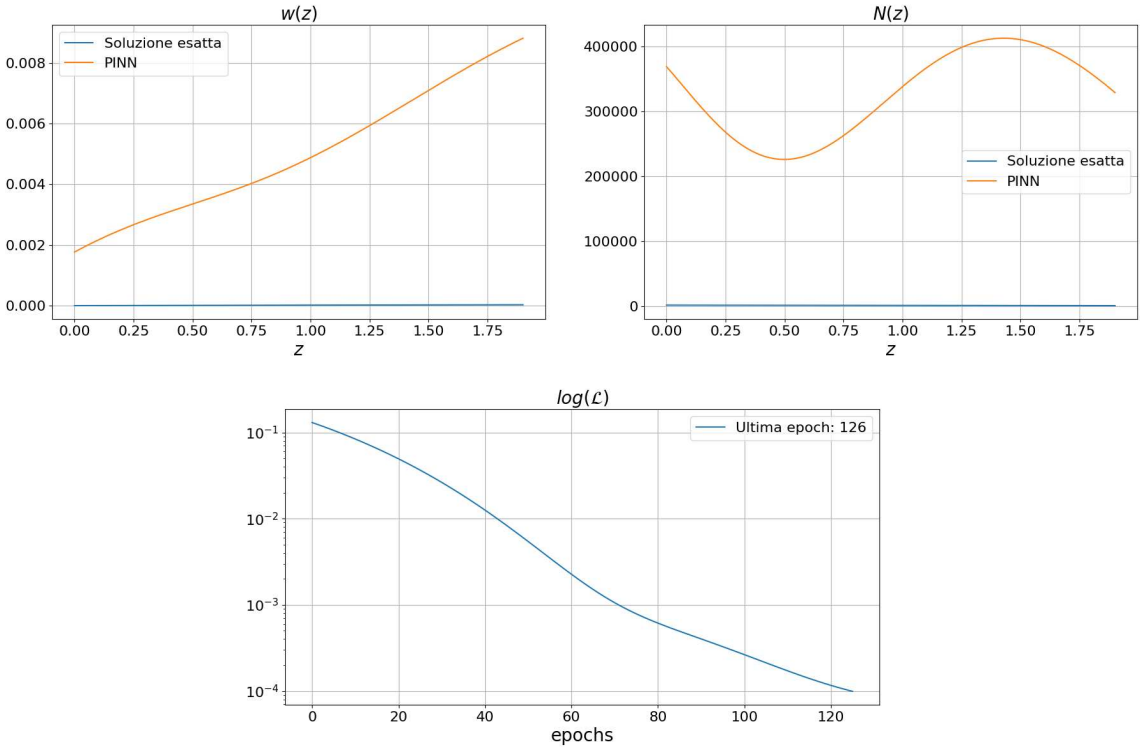


(b)

Figura 7.1: Risultati del caso 1 del problema assiale di DSV dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

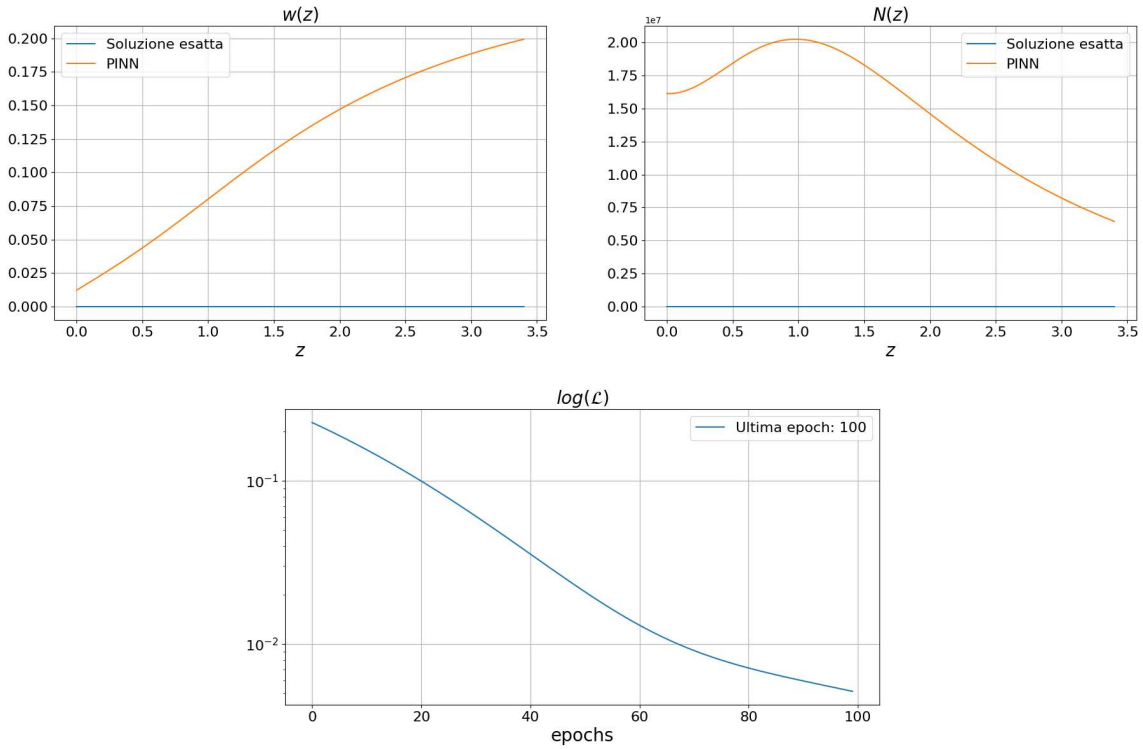


(a)

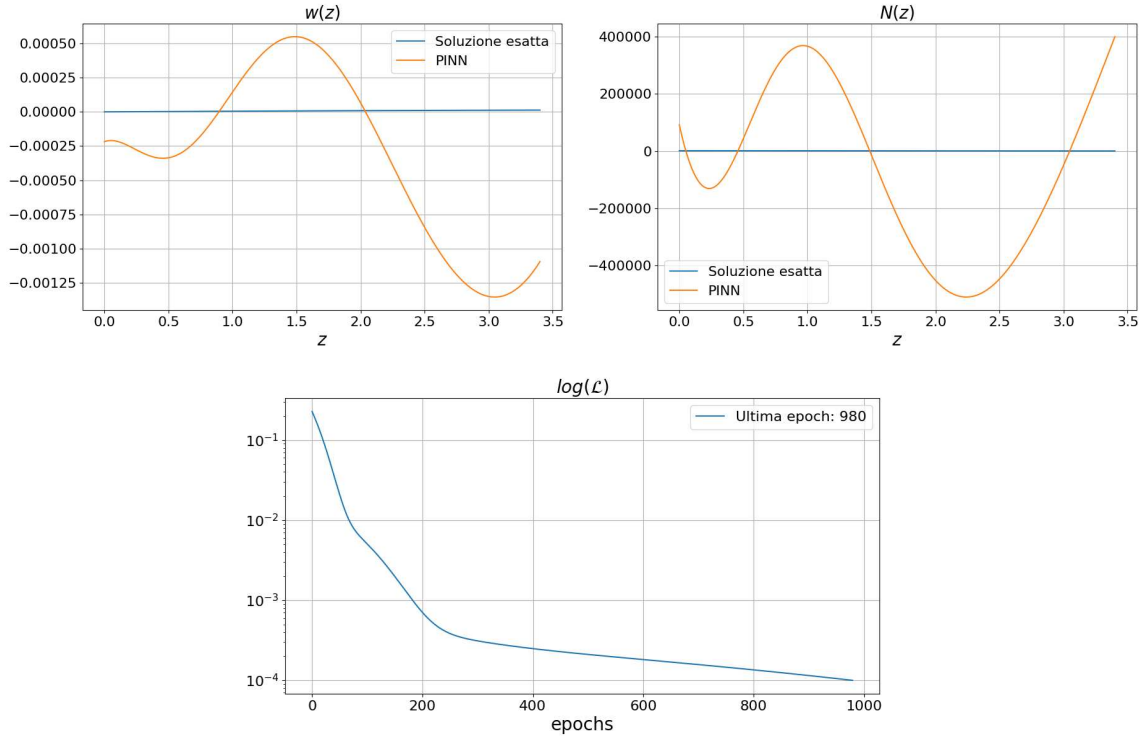


(b)

Figura 7.2: Risultati del caso 2 del problema assiale di DSV dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)



(a)



(b)

Figura 7.3: Risultati del caso 3 del problema assiale di DSV dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

$$N_0 = qL + N_L. \tag{7.5}$$

Vengono quindi adimensionalizzate le variabili del problema come segue

$$\begin{aligned} z =& L\, z^*, \\ w =& \frac{LN_0}{EA}\, w^* \end{aligned} \tag{7.6}$$

ove i valori con * sono le variabili adimensionalizzate. Segue quindi che le derivate di $w(z)$ risultano

$$\begin{aligned} w^{'} &= \frac{dw}{dz} = \frac{LN_0}{EA}\frac{1}{L}\frac{dw^*}{dz^*} = \frac{N_0}{EA} w^{'*} \\ w^{''} &= \frac{d^2w}{dz^2} = \frac{LN_0}{EA}\frac{1}{L^2}\frac{d^2w^*}{dz^{*2}} = \frac{N_0}{EAL} w^{''*} \end{aligned} \tag{7.7}$$

Questo porta il problema rappresentato in Eq. 5.21 ad essere

$$\begin{aligned} w^{''*} &= -\frac{qL}{N_0}, \\ w^* = w_0^* \quad \text{e/o} \quad w^{'*} &= \frac{N_L}{N_0} \quad \text{al contorno} \end{aligned} \tag{7.8}$$

che tramite la rete neurale cerca una soluzione adimensionalizzata del tipo

$$w^*(z) \approx NN_{w^*}(z,\, \theta). \tag{7.9}$$

Allo stesso modo lo schema della rete è 1-10-1 e si allena sempre su un training set di 100 collocation points, in questo caso equispaziati tra 0 e 1, e 2 condizioni al contorno definite sui due punti di estremità presenti nel set. La funzione di costo è quindi definita come

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_b + \mathcal{L}_p, \\ \mathcal{L}_b &= w^*(0)^2 + \left(w^{'*}(1) - \frac{N_L}{N_0} \right)^2 \\ \mathcal{L}_p &= \frac{1}{N_p} \sum \left(w^{''*}(z) + \frac{qL}{N_0} \right)^2 \end{aligned} \tag{7.10}$$

dove N_p è il numero di collocation points utilizzati durante la fase di training. La rete viene allenata in un massimo di 10000 epochs e si interrompe sempre a $\mathcal{L} < 10^{-4}$. I risultati sono poi ottenuti applicando la rete neurale su di un testing dataset di 1000 punti, così da renderne più accurata la rappresentazione, e verranno moltiplicati per

i propri fattori di adimensionalizzazione, in modo tale da poterli confrontare con le soluzioni analitiche.

Come si può notare dalle Fig. 7.4, 7.5 e 7.6, già dopo poche centinaia di iterazioni la soluzione proposta dalla PINN inizia ad avvicinarsi a quella analitica, ma per il raggiungimento della precisione richiesta sono necessarie più epochs. Si può notare come, rispetto ai tentativi non adimensionalizzati, la soluzione converga e risulti quasi identica a quella esatta.

7.5 Problema flessionale di DSV

Il problema flessionale di DSV, come quello assiale, è un problema semplice, dovuto alla monodimensionalità e alla linearità: note le condizioni al contorno, la soluzione analitica è nota (Eq. 5.45). Sono effettuate 3 diverse prove con dati del problema differenti:

Caso	L (m)	D (mm)	I (m^4)	E (Pa)	p (N/m)	T_L (N)	M_L (Nm)
1	0.8	80	$2.011 \cdot 10^{-6}$	$7 \cdot 10^{10}$	150	100	200
2	1.9	150	$2.485 \cdot 10^{-5}$	$7 \cdot 10^{10}$	200	150	300
3	3.4	210	$9.547 \cdot 10^{-5}$	$7 \cdot 10^{10}$	100	200	175

Tabella 7.2: Dati di 3 differenti casi del problema flessionale di DSV

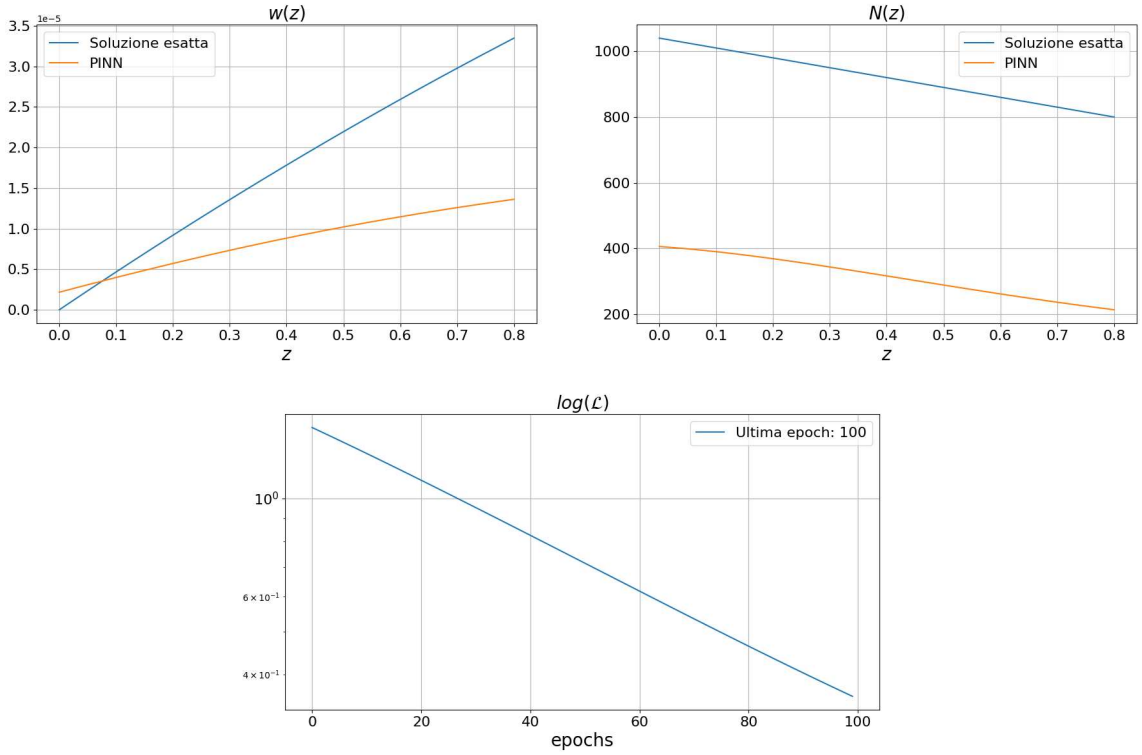
I primi 4 descrivono la geometria e il materiale in studio, mentre gli ultimi tre si riferiscono ai carichi applicati: p è il carico trasversale distribuito lungo la trave e T_L e M_L sono la forza trasversale e il momento flettente applicati all'estremità libera. In tutte le prove ci si riferisce ad una trave a sbalzo, quindi $v(0) = v^{'}(0) = 0$. L'obiettivo di questo studio è far si che la soluzione $v(z)$ sia rappresentata da una rete neurale, ovvero

$$v(z) \approx NN_v(z,\, \theta). \tag{7.11}$$

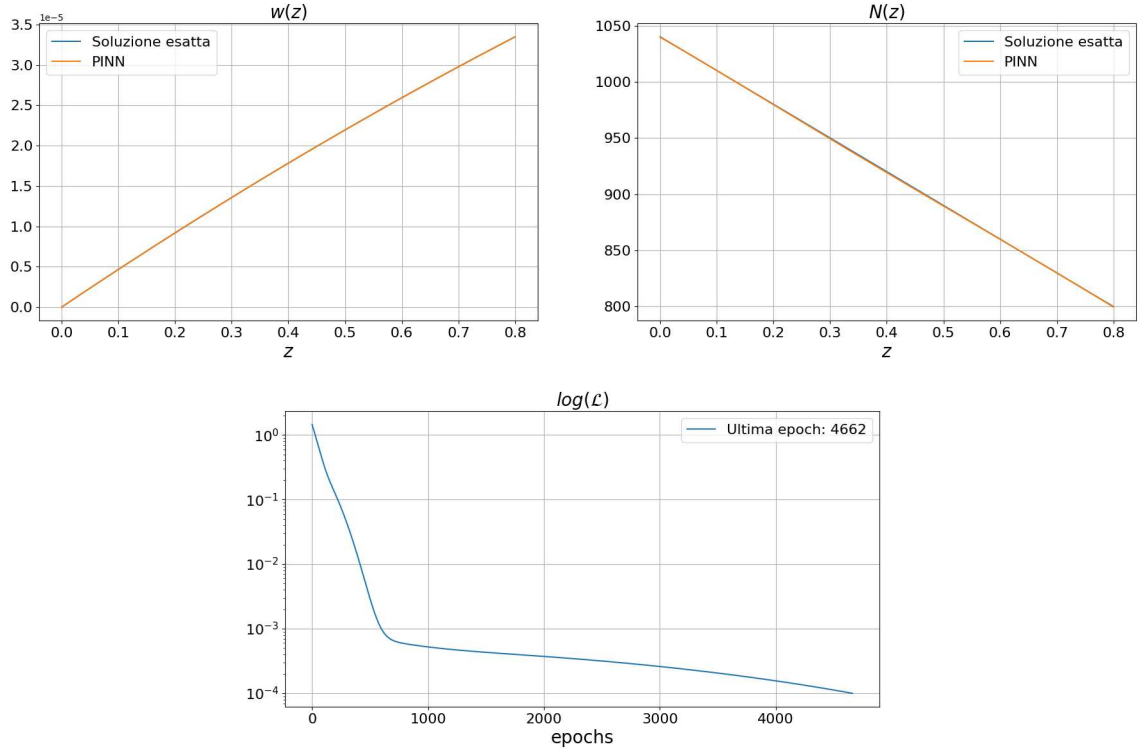
Lo schema della rete è 1-10-1 e questa si allena su un training set di 100 collocation points equispaziati tra 0 e L, e 4 condizioni al contorno definite sui due punti di estremità presenti nel set. La funzione di costo è definita come

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_b + \mathcal{L}_p, \\ \mathcal{L}_b &= v(0)^2 + v^{'}(0)^2 + \\ &\quad + \left(v^{''}(L) + \frac{M_L}{EI}\right)^2 + \left(v^{'''}(L) + \frac{T_L}{EI}\right)^2 \\ \mathcal{L}_p &= \frac{1}{N_p} \sum \left(v^{''''}(z) - \frac{p}{EI}\right)^2, \end{aligned} \tag{7.12}$$

dove N_p è il numero di collocation points utilizzati durante la fase di training. La rete viene allenata in un massimo di 10000 epochs e si interrompe a $\mathcal{L} < 10^{-4}$. I

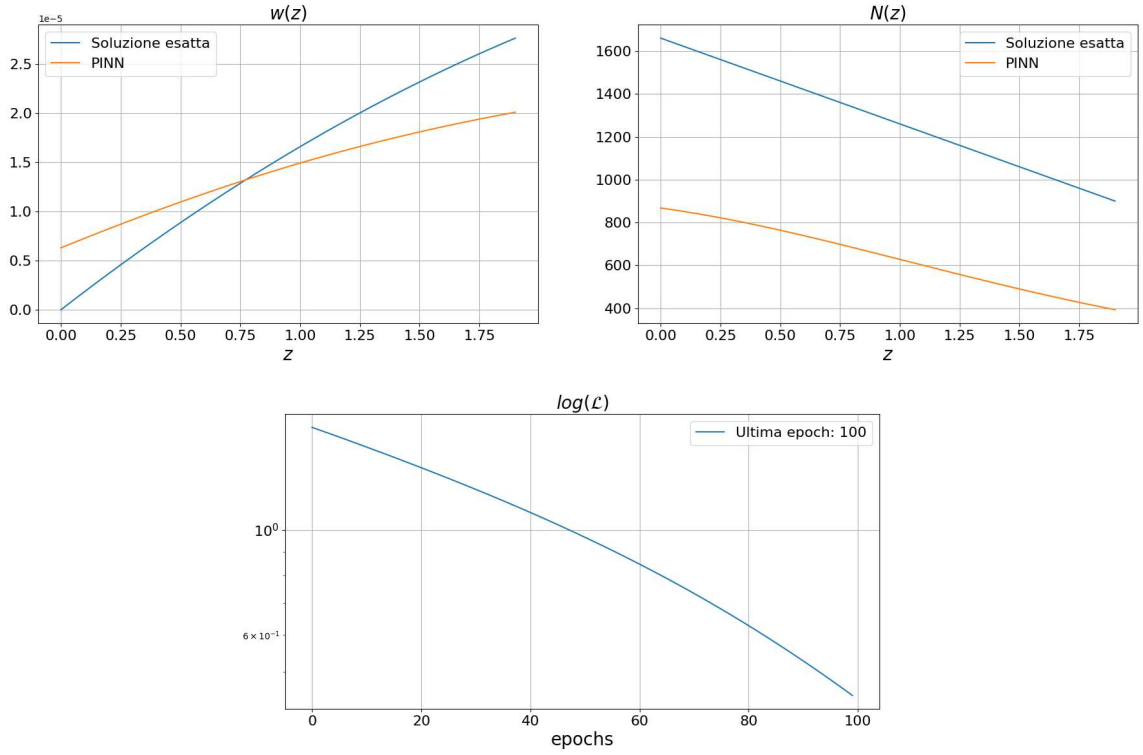


(a)

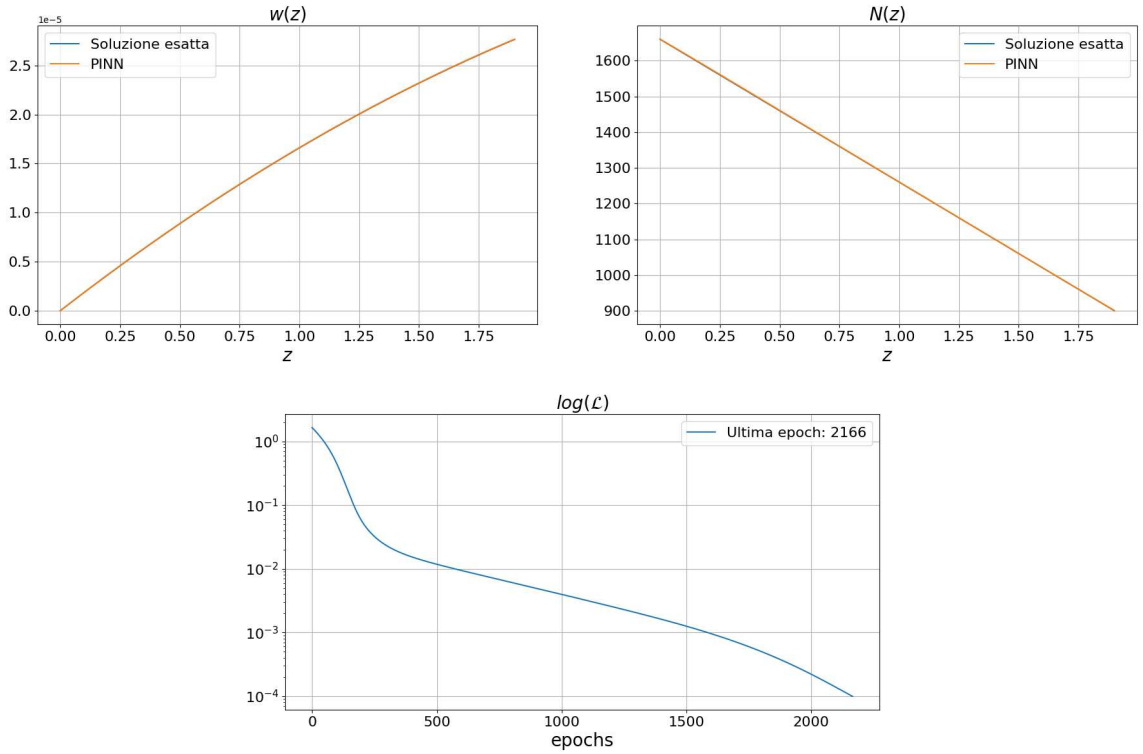


(b)

Figura 7.4: Risultati del caso 1 del problema assiale di DSV adimensionalizzato dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

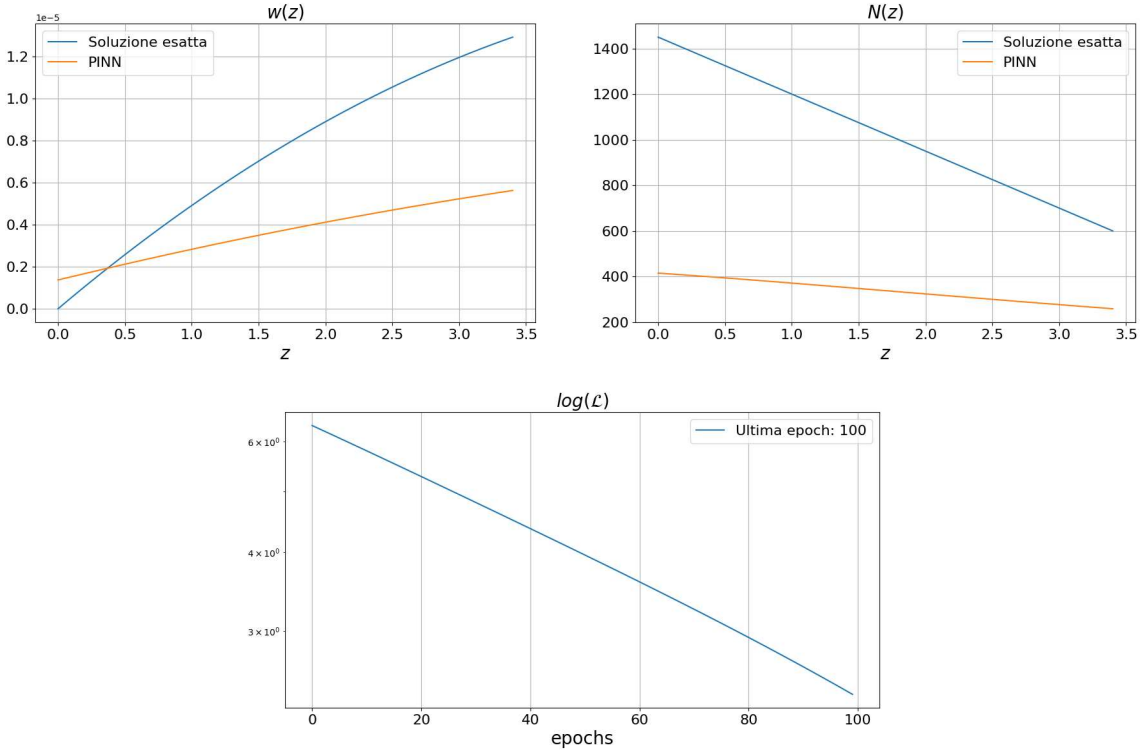


(a)

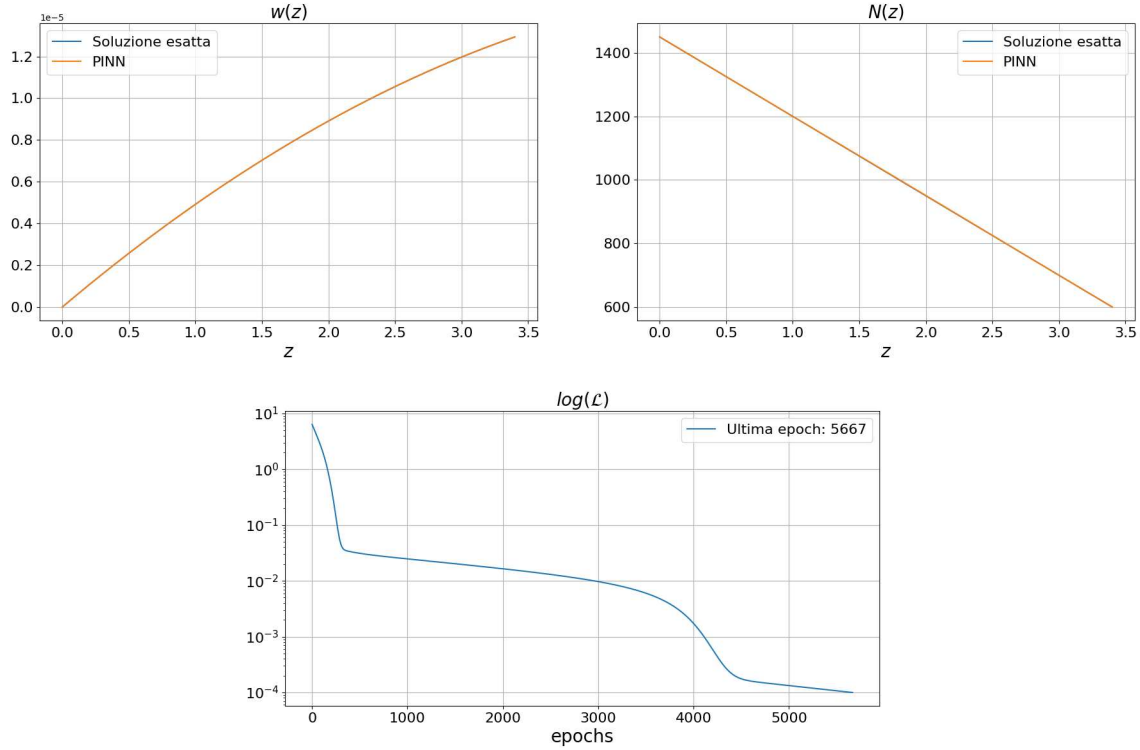


(b)

Figura 7.5: Risultati del caso 2 del problema assiale di DSV adimensionalizzato dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)



(a)



(b)

Figura 7.6: Risultati del caso 3 del problema assiale di DSV adimensionalizzato dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

risultati sono poi ottenuti applicando la rete neurale su di un testing dataset di 1000 punti, così da renderne più accurata la rappresentazione.

Similmente al caso assiale, dalle Fig. 7.7, 7.8 e 7.9 si nota che i risultati forniti dalla rete neurale non sono in grado di rappresentare la soluzione esatta. Differentemente dal problema precedente, siamo di fronte ad una equazione differenziale del quarto ordine mentre il problema assiale rappresenta un problema del secondo ordine. Per migliorare l'apprendimento si può agire affrontando i problemi che si ripresentano: è bene quindi definire dei fattori moltiplicativi tramite i quali ridimensionare il problema, cercando così delle soluzioni adimensionalizzate.

Quindi, per una ricerca delle soluzioni adimensionalizzate, si introduce un nuovo parametro M_0. Questo parametro rappresenta il valore del momento flettente all'incastro ed è definito come

$$M_0 = M_L - T_L L - \frac{pL^2}{2}. \tag{7.13}$$

Vengono quindi adimensionalizzate le variabili del problema come segue

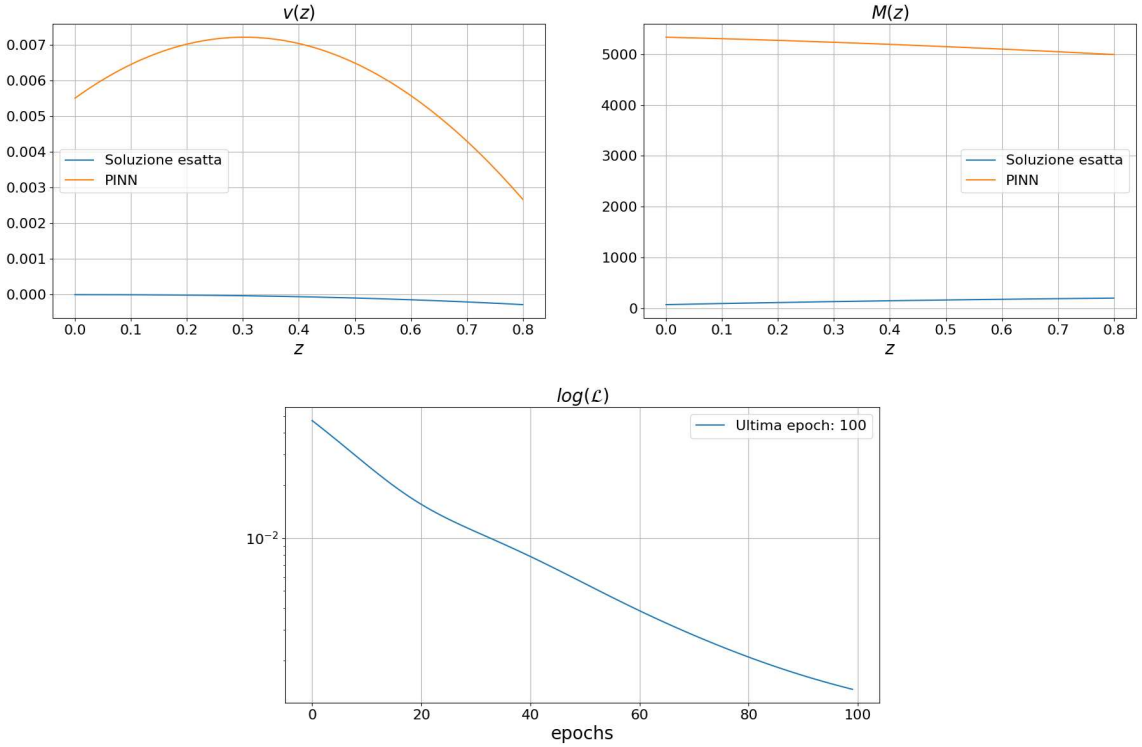
$$\begin{aligned} z =& L\, z^*, \\ v =& \frac{LM_0}{EI}\, v^* \end{aligned} \tag{7.14}$$

ove i valori con * sono le variabili adimensionalizzate. Segue quindi che le derivate di $v(z)$ risultano

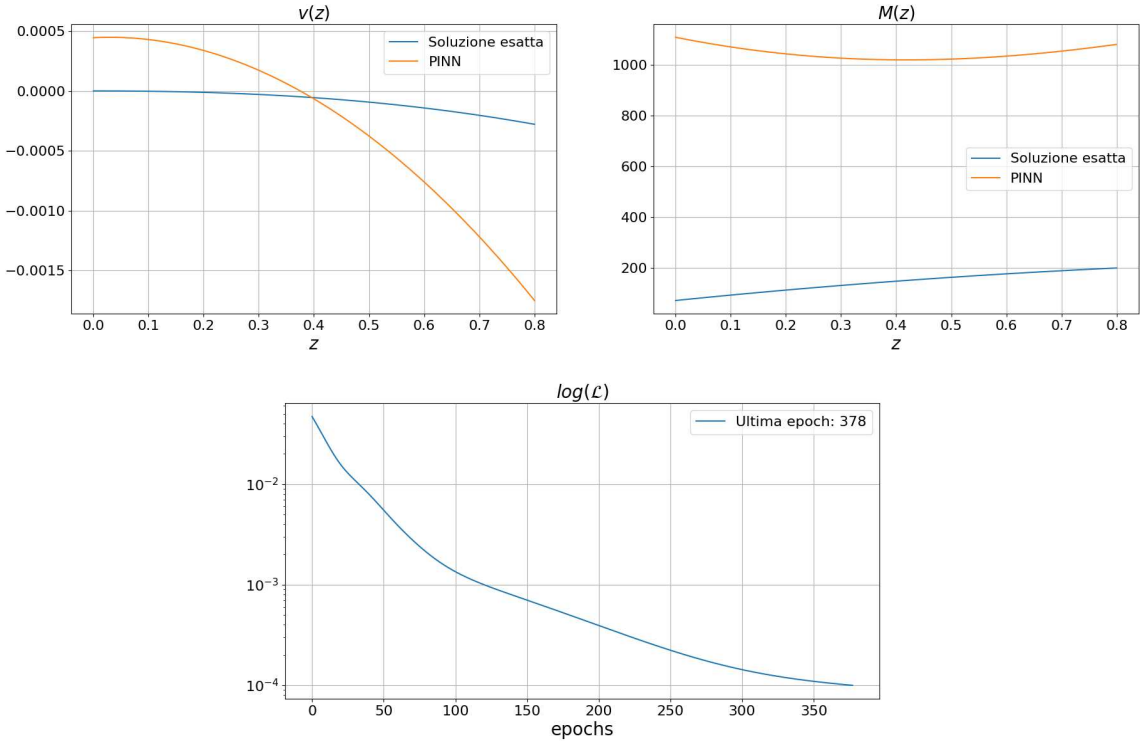
$$\begin{aligned} v^{'} =& \frac{dv}{dz} = \frac{LM_0}{EI}\frac{1}{L}\frac{dv^*}{dz^*} = \frac{M_0}{EI} v^{'*} \\ v^{''} =& \frac{d^2v}{dz^2} = \frac{LM_0}{EI}\frac{1}{L^2}\frac{d^2v^*}{dz^{*2}} = \frac{M_0}{EIL} v^{''*} \\ v^{'''} =& \frac{d^3v}{dz^3} = \frac{LM_0}{EI}\frac{1}{L^3}\frac{d^3v^*}{dz^{*3}} = \frac{M_0}{EIL^2} v^{'''*} \\ v^{''''} =& \frac{d^4v}{dz^4} = \frac{LM_0}{EI}\frac{1}{L^4}\frac{d^4v^*}{dz^{*4}} = \frac{M_0}{EIL^3} v^{''''*} \end{aligned} \tag{7.15}$$

Questo porta il problema rappresentato in Eq. 5.40 ad essere

$$\begin{aligned} v^{''''*} &= \frac{pL^3}{M_0}, \\ v^* = v_o \quad \text{e/o} \quad v^{'*} &= -\frac{\phi_o EI}{M_0} \quad \text{e/o} \\ v^{''*} = -\frac{M_L L}{M_0} \quad \text{e/o} \quad v^{'''*} &= -\frac{T_L L^2}{M_0} \quad \text{al contorno} \end{aligned} \tag{7.16}$$

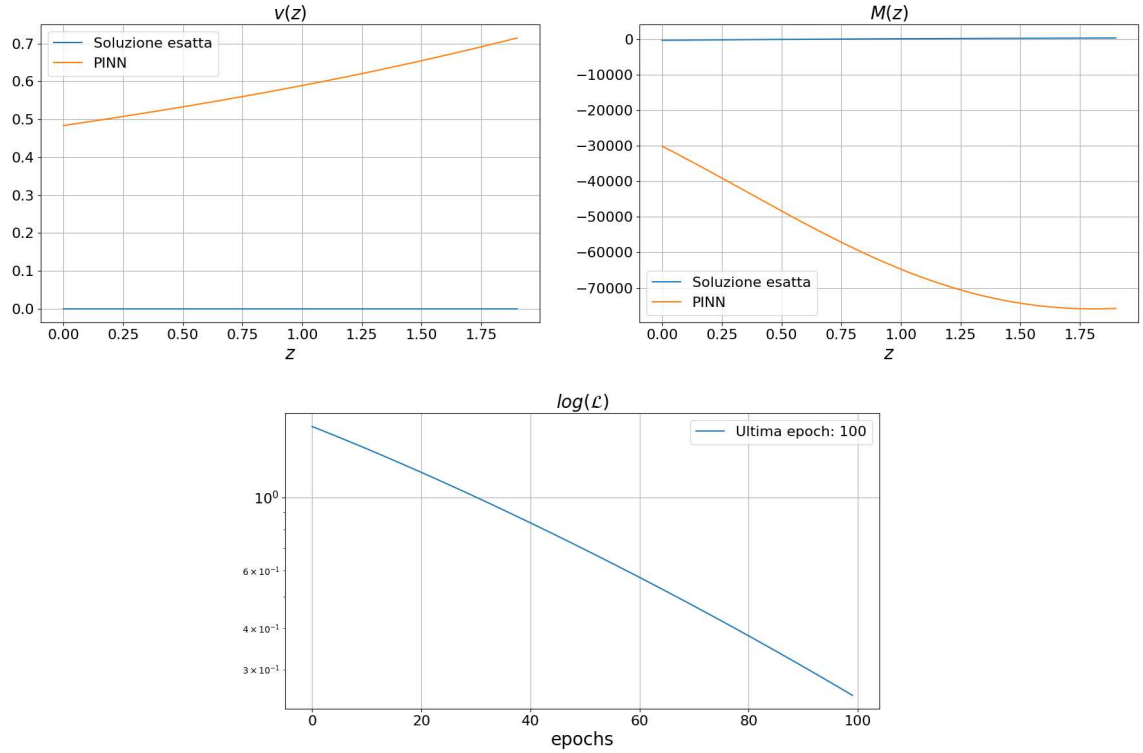


(a)

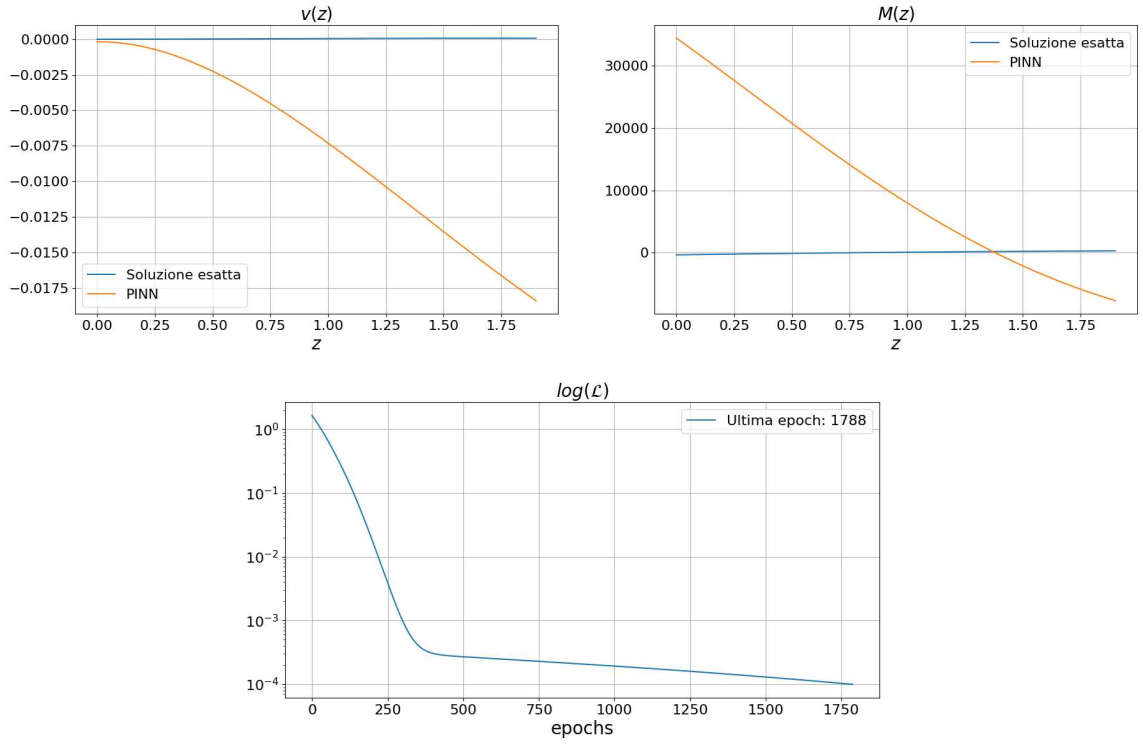


(b)

Figura 7.7: Risultati del caso 1 del problema flessionale di DSV dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

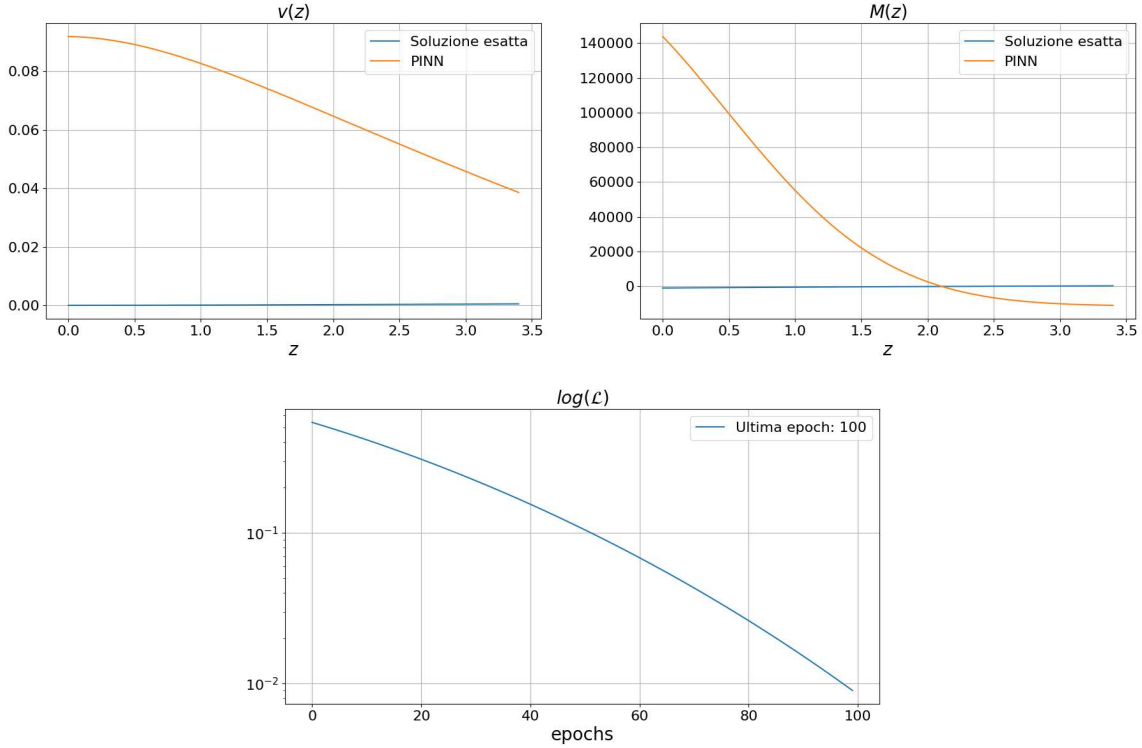


(a)

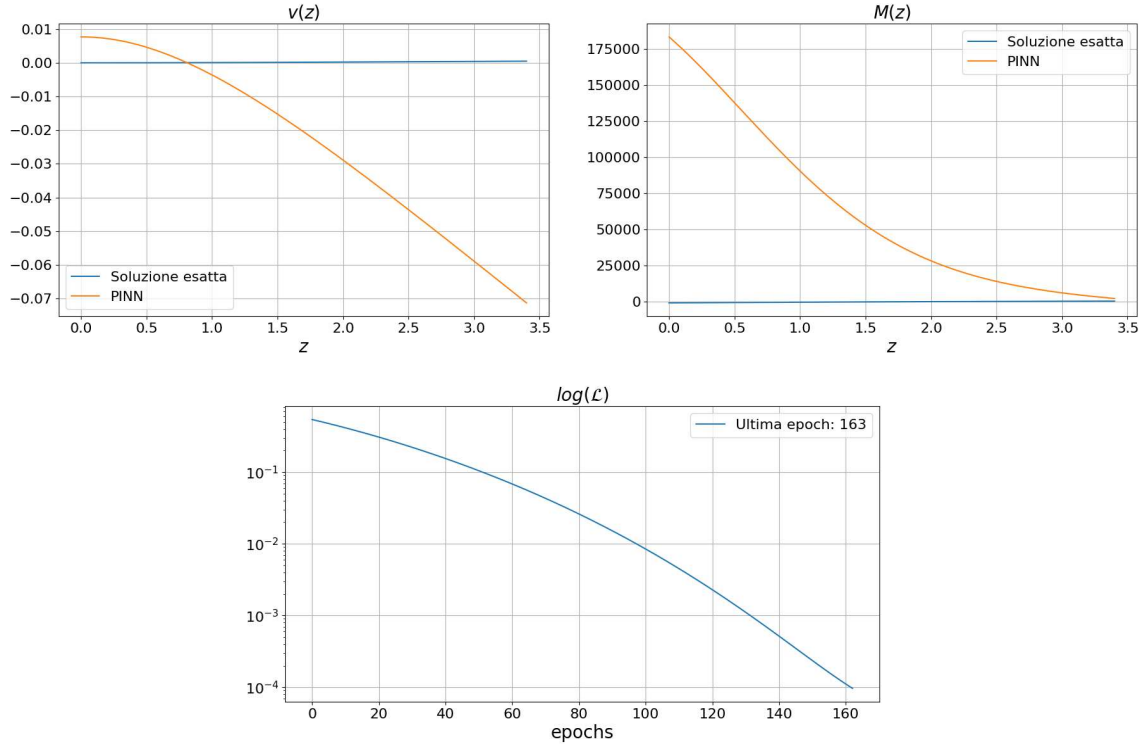


(b)

Figura 7.8: Risultati del caso 2 del problema flessionale di DSV dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)



(a)



(b)

Figura 7.9: Risultati del caso 3 del problema flessionale di DSV dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

che tramite la rete neurale cerca una soluzione adimensionalizzata del tipo

$$v^*(z) \approx NN_{v^*}(z,\, \theta). \tag{7.17}$$

Lo schema della rete è sempre 1-10-1 e si allena su un training set di 100 collocation points, ma questa volta equispaziati tra 0 e 1, e 4 condizioni al contorno definite sui due punti di estremità presenti nel set. La funzione di costo è definita come

$$\begin{aligned}
\mathcal{L} &= \mathcal{L}_b + \mathcal{L}_p, \\
\mathcal{L}_b &= v^*(0)^2 + v'^*(0)^2 + \\
&\quad + \left(v''(1) + \frac{M_L L}{M_0}\right)^2 + \left(v'''^*(1) + \frac{T_L L^2}{M_0}\right)^2 \\
\mathcal{L}_p &= \frac{1}{N_p} \sum \left(v''''^* - \frac{pL^3}{M_0}\right)^2,
\end{aligned} \tag{7.18}$$

dove N_p è il numero di collocation points utilizzati durante la fase di training. La rete viene allenata in un massimo di 10000 epochs e si interrompe a $\mathcal{L} < 10^{-4}$. I risultati sono poi ottenuti applicando la rete neurale su di un testing dataset di 1000 punti, così da renderne più accurata la rappresentazione, e verranno moltiplicati per i propri fattori di adimensionalizzazione in modo tale da poterli confrontare con le soluzioni analitiche.

Si può notare come in Fig. 7.10, 7.11 e 7.12, rispetto ai tentativi non adimensionalizzati, la soluzione converga e risulti quasi identica a quella esatta. Sono necessarie più iterazioni rispetto al caso assiale in quanto questa soluzione è di quarto grado e quindi più complessa da trovare.

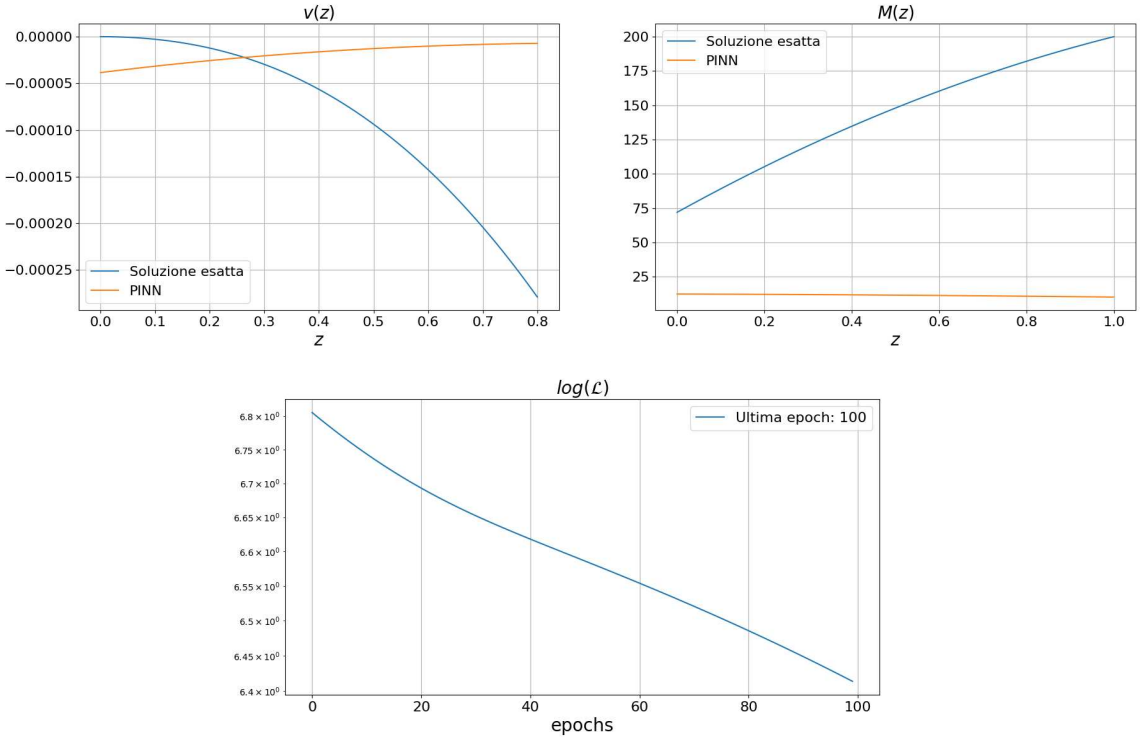
7.6 Problema flessionale non lineare

Quando si tratta di grandi spostamenti, viene meno l'ipotesi di linearità che rende semplice la determinazione di soluzioni analitiche. In questo caso si tratta il problema di trave a sbalzo sottoposta a carichi che la deflettono, il cui asse è allineato lungo x ($s = x$). Sono effettuate 3 diverse prove con dati del problema differenti:

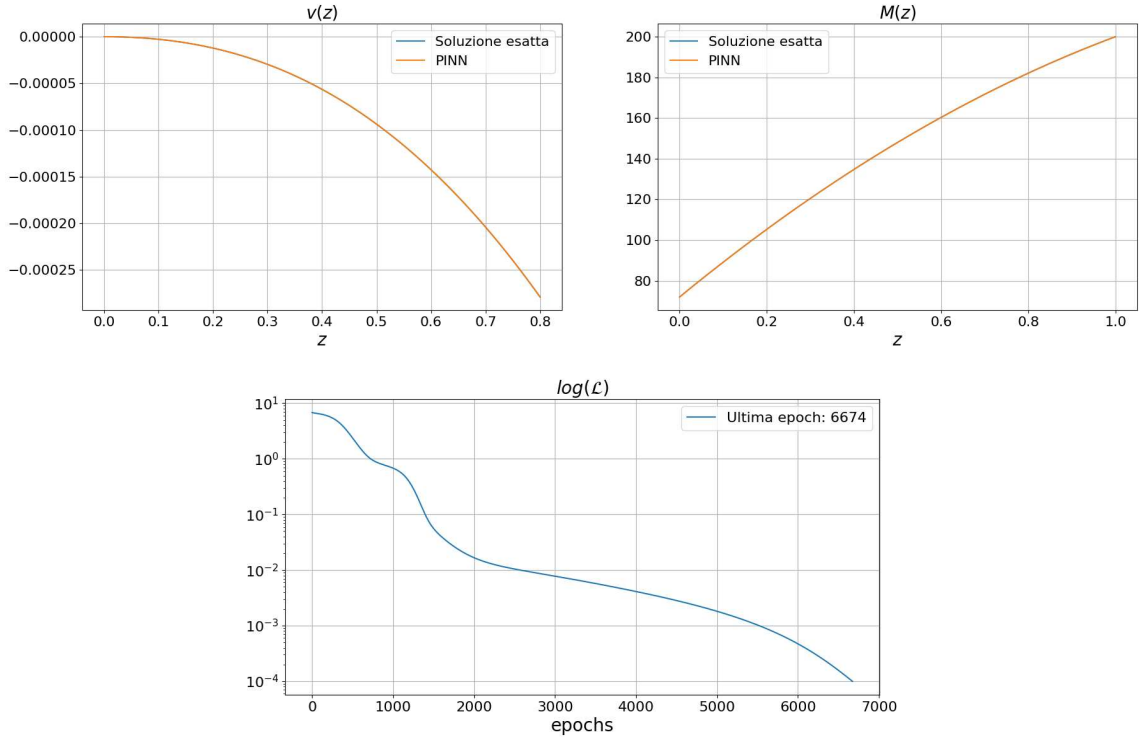
Caso	L (m)	D (mm)	I (m^4)	E (Pa)	F_x (N)	F_y (N)	M_L (Nm)
1	0.8	20	$7.854 \cdot 10^{-9}$	$7 \cdot 10^{10}$	-700	900	-1100
2	1.9	30	$3.976 \cdot 10^{-8}$	$7 \cdot 10^{10}$	1200	2400	1400
3	3.4	50	$3.068 \cdot 10^{-7}$	$7 \cdot 10^{10}$	3500	0	4000

Tabella 7.3: Dati di 3 differenti prove del problema flessionale non lineare

Come nel problema flessionale di DSV, i primi 4 descrivono la geometria e il materiale in studio, mentre gli ultimi tre si riferiscono ai carichi applicati all'estremità libera: F_x e F_y sono i carichi concentrati applicati lungo x e y mentre M_L è il momento

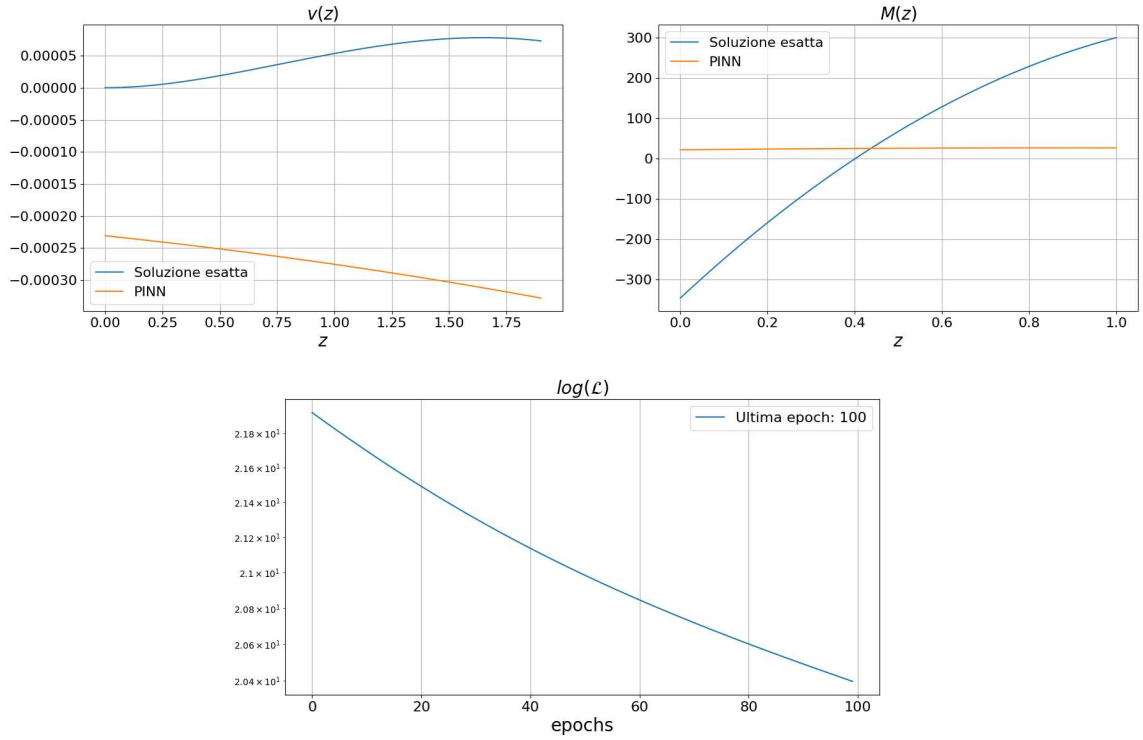


(a)

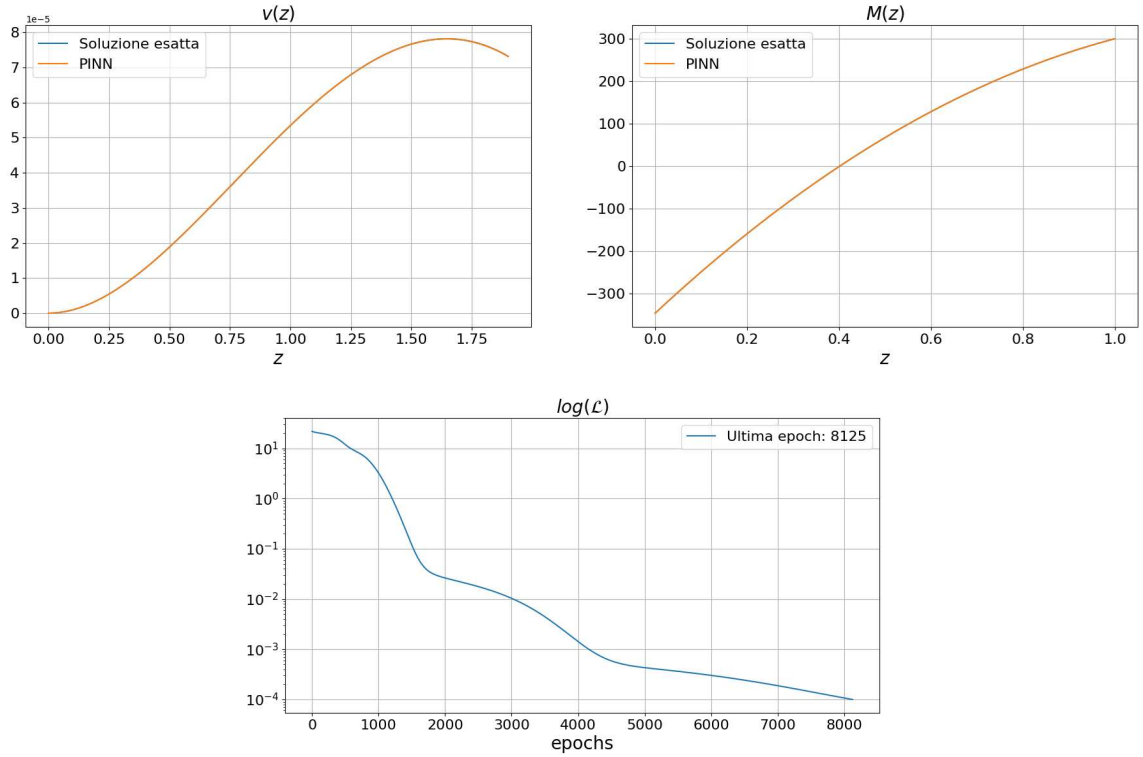


(b)

Figura 7.10: Risultati della prova 1 del problema flessionale di DSV adimensionalizzato dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

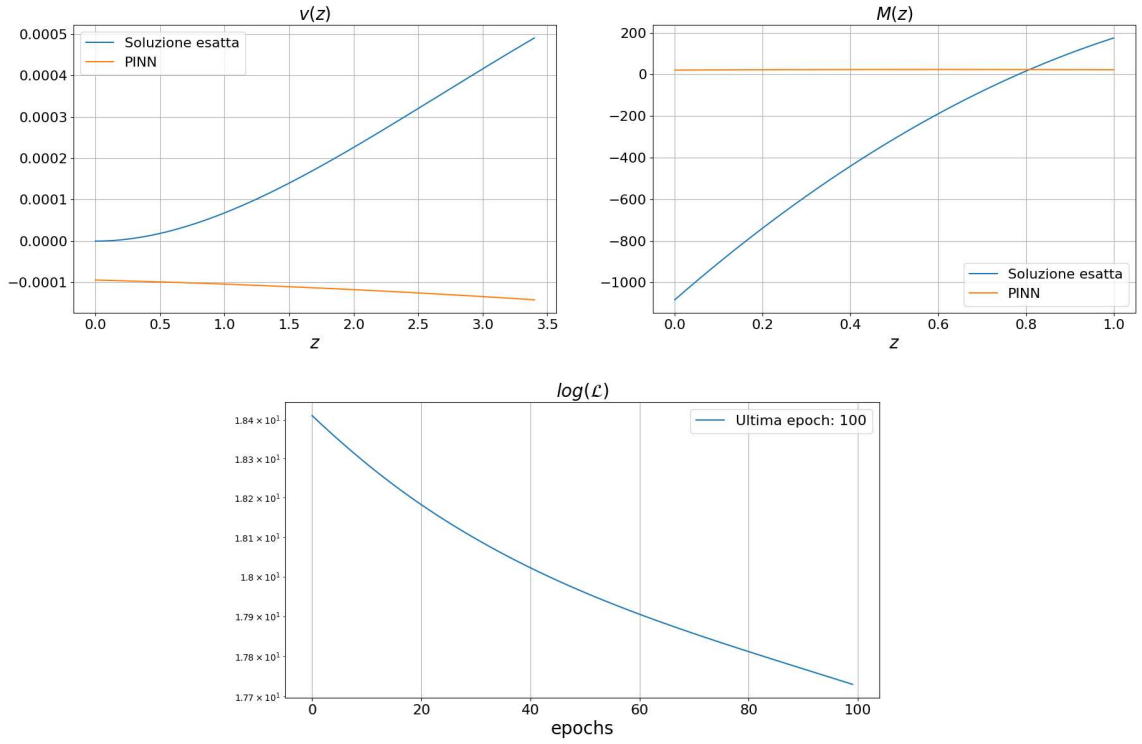


(a)

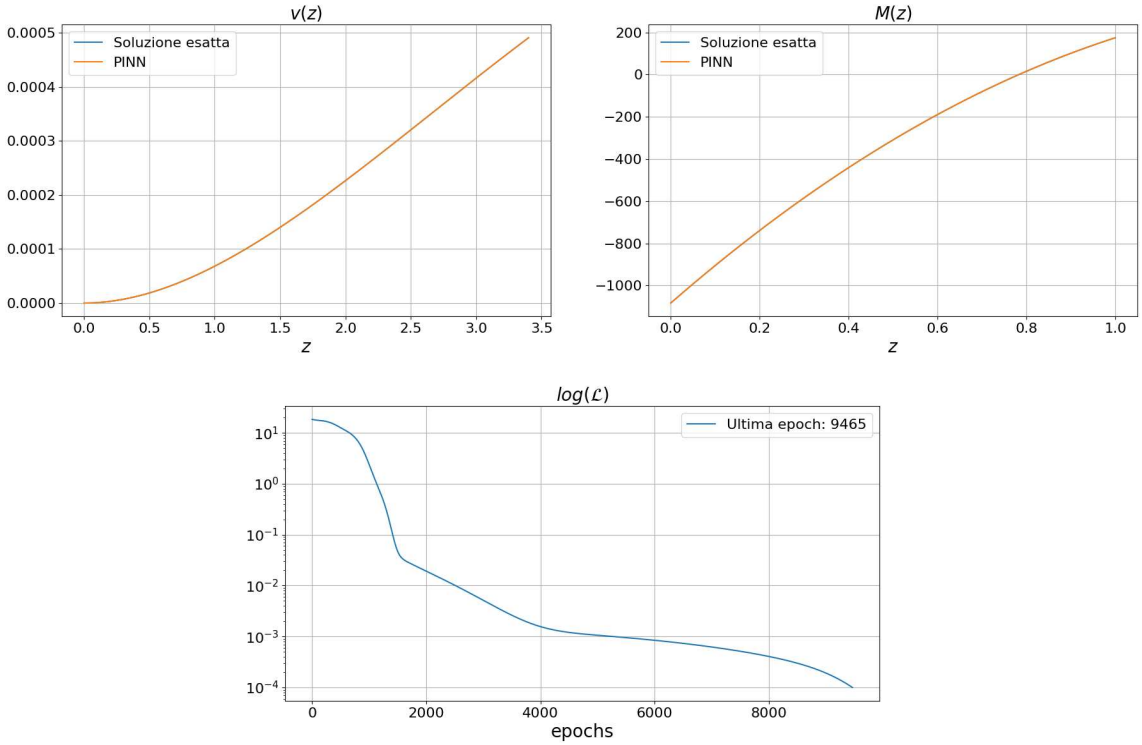


(b)

Figura 7.11: Risultati della prova 2 del problema flessionale di DSV adimensionalizzato dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)



(a)



(b)

Figura 7.12: Risultati della prova 3 del problema flessionale di DSV adimensionalizzato dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

flettente attorno a z (Fig. 6.1 per il riferimento degli assi). Come descritto nell'Eq. 6.6, si cerca una soluzione per la variabile φ che, a seguito degli integrali di linea lungo x (Eq. 6.7), fornirà la nuova geometria deformata della trave. L'obiettivo di questo studio è far si che la soluzione $\varphi(x)$ sia rappresentata da una rete neurale, ovvero

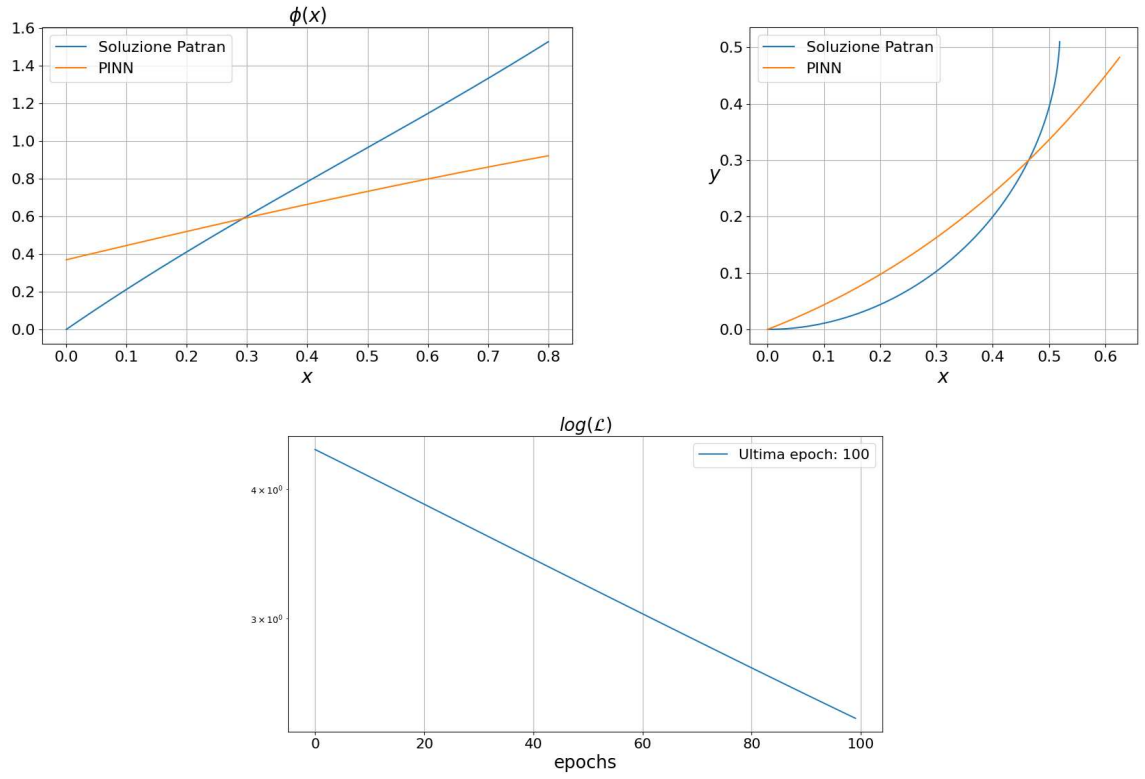
$$\varphi(x) \approx NN_{\varphi}(x, \theta). \tag{7.19}$$

Lo schema della rete è 1-10-1 e questa si allena su un training set di 100 collocation points equispaziati tra 0 e L, e 2 condizioni al contorno definite sui due punti di estremità presenti nel set. Trattandosi di una trave a sbalzo si avrà $\varphi(0) = 0$. La funzione di costo è definita come

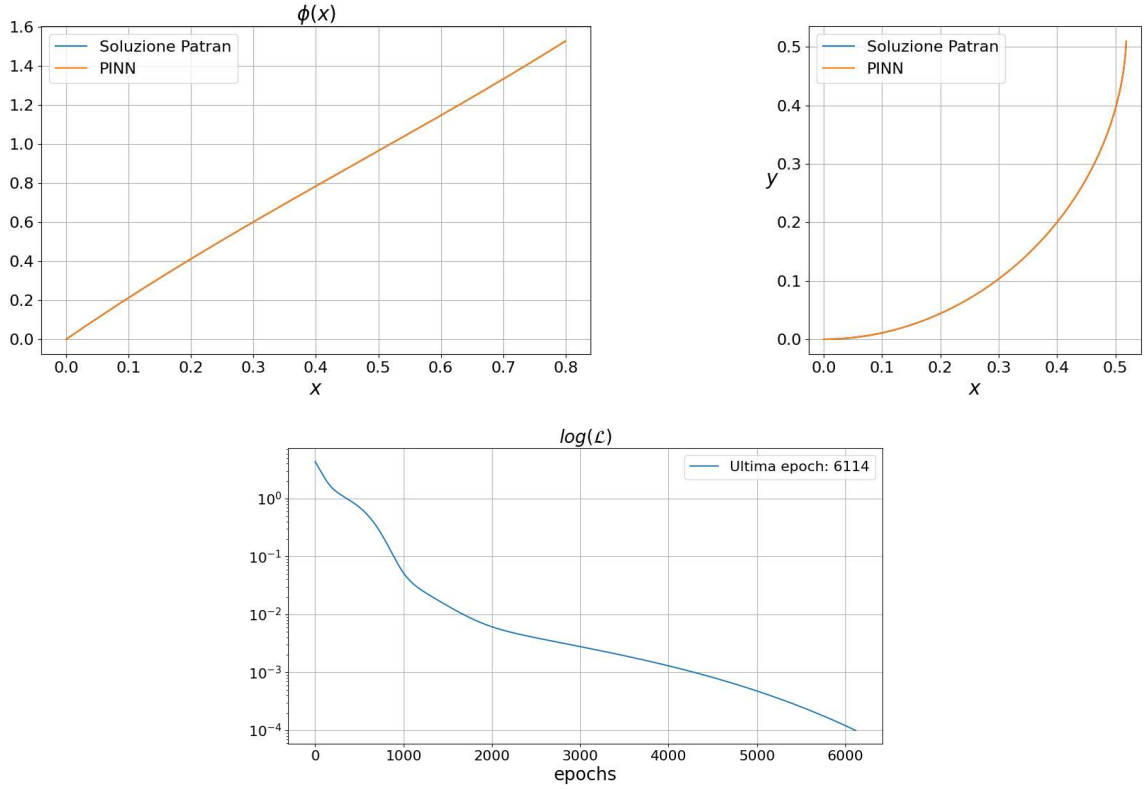
$$\begin{aligned} \mathcal{L} &= \mathcal{L}_b + \mathcal{L}_p, \\ \mathcal{L}_b &= \varphi(0)^2 + \left(\varphi^{'}(L) + \frac{M_L}{EI}\right)^2 \\ \mathcal{L}_p &= \frac{1}{N_p} \sum \left(\varphi^{''}(x) + \frac{F_x \ \sin \ \varphi(x) + F_y \ \cos \ \varphi(x)}{EI}\right)^2, \end{aligned} \tag{7.20}$$

dove N_p è il numero di collocation points utilizzati durante la fase di training. La rete viene allenata in un massimo di 10000 epochs e si interrompe a $\mathcal{L} < 10^{-4}$. I risultati sono poi ottenuti eseguendo la rete neurale su un testing dataset di 1000 punti, così da renderne più accurata la rappresentazione.

Si può notare come in Fig. 7.13, 7.14 e 7.15 il problema converga molto bene e la soluzione sia pressoché coincidente con le soluzioni ottenute con le analisi FEM, senza bisogno di ridimensionare il problema. Questo è dovuto al fatto che la soluzione e le sue derivate sono dell'ordine dell'unità. Per raggiungere una maggiore precisione nella prova 3 è stato sufficiente aumentare la precisione richiesta al training imponendo $\mathcal{L} < 10^{-5}$ (Fig. 7.16), rendendo più restrittiva la condizione di uscita dalla fase di training.

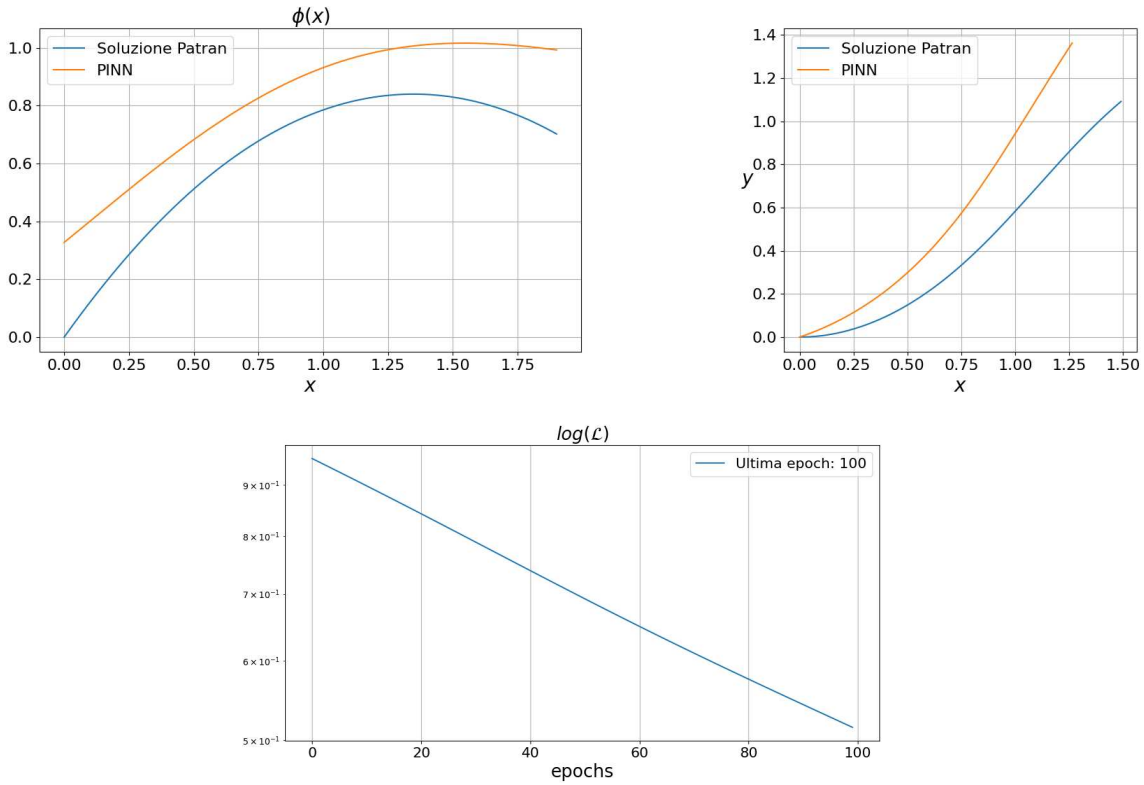


(a)

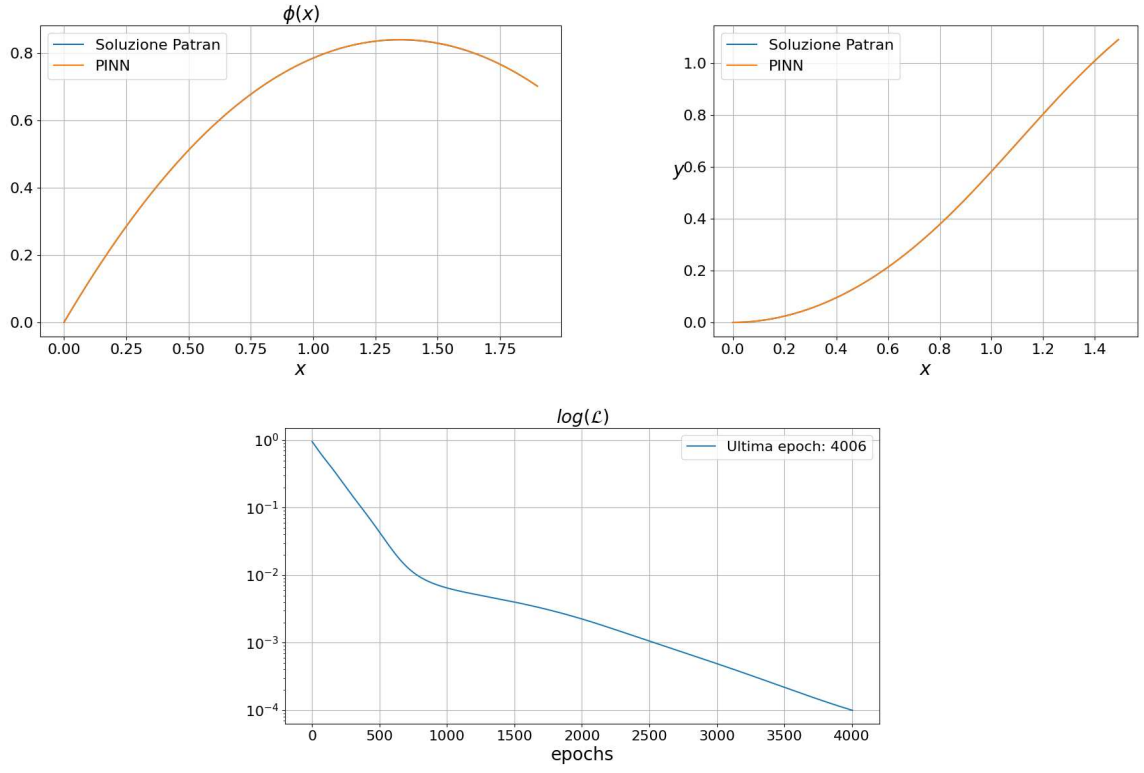


(b)

Figura 7.13: Risultati del caso 1 del problema dei grandi spostamenti dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

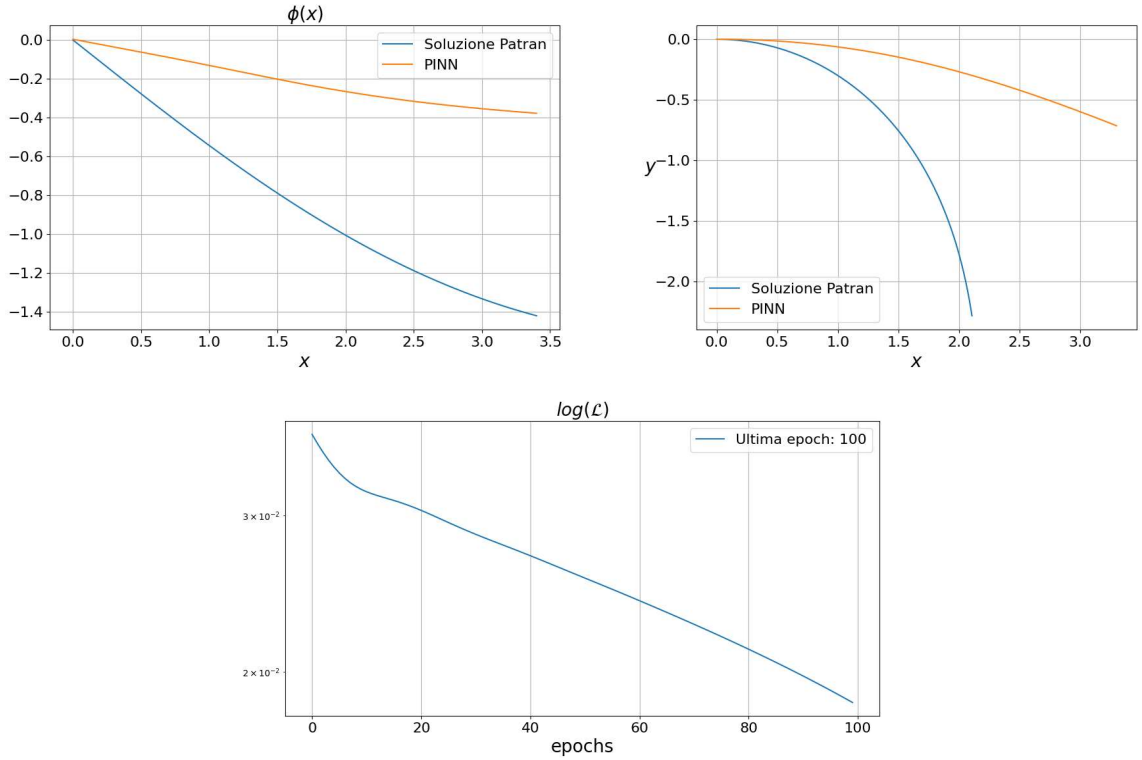


(a)

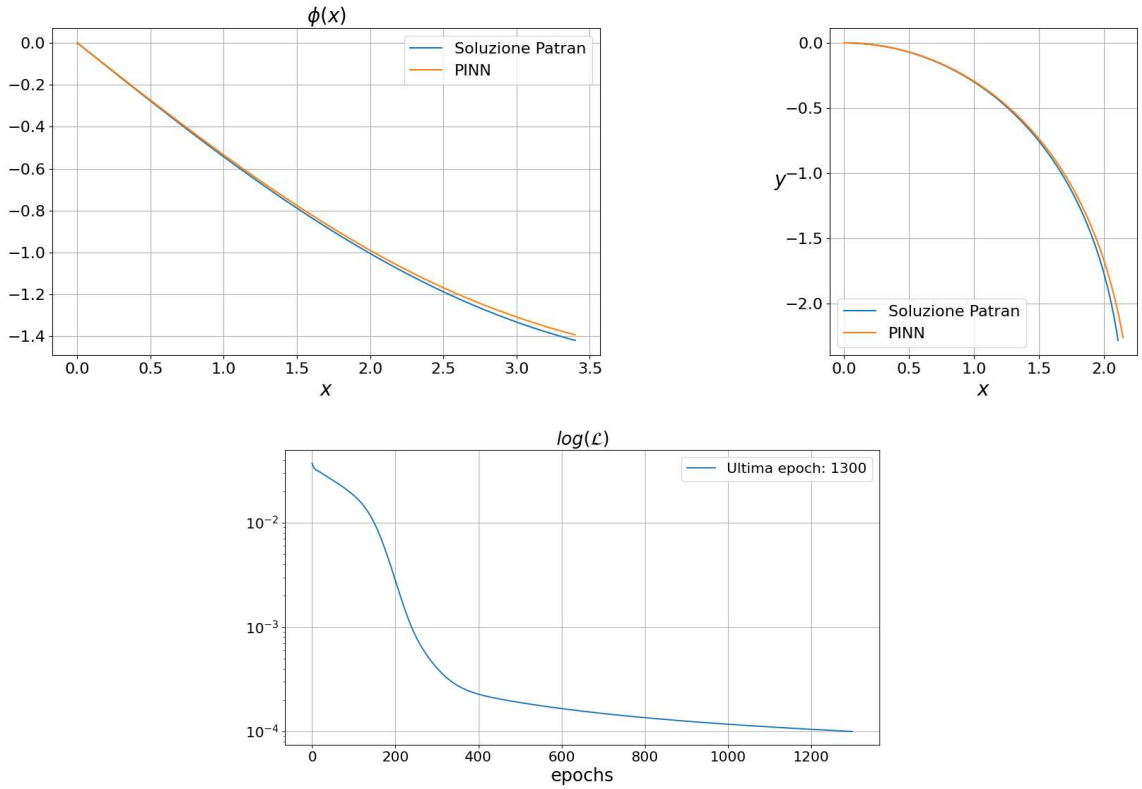


(b)

Figura 7.14: Risultati del caso 2 del problema dei grandi spostamenti dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)

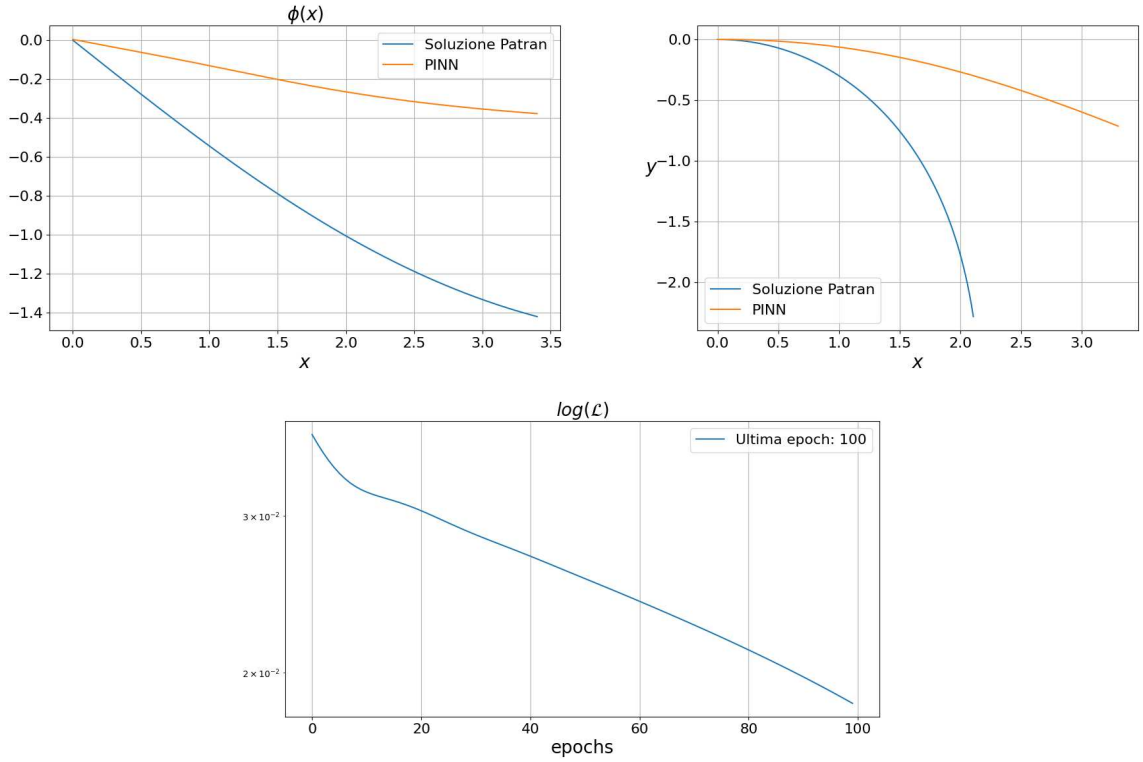


(a)

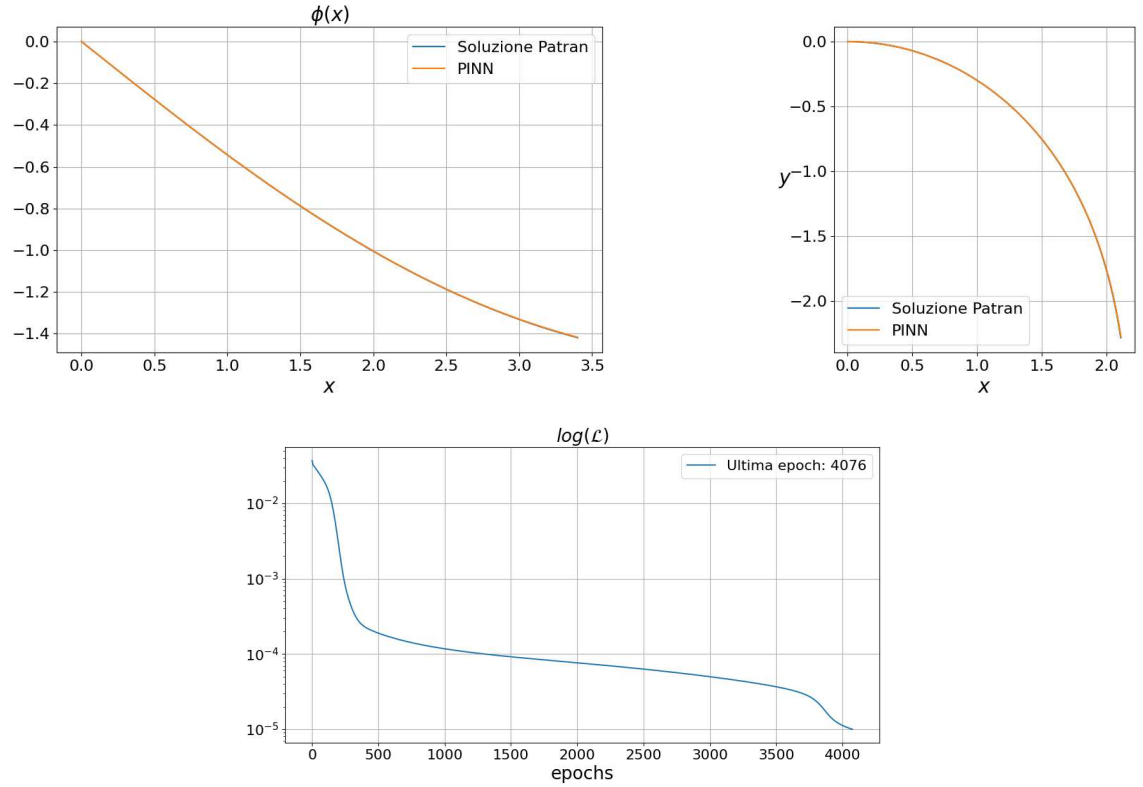


(b)

Figura 7.15: Risultati del caso 3 del problema dei grandi spostamenti dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-4}$ (b)



(a)



(b)

Figura 7.16: Risultati del caso 3 del problema dei grandi spostamenti dopo 100 epochs (a) e a fine training quando $\mathcal{L} < 10^{-5}$ (b)

7.7 Problema lineare 3D

Nel problema tridimensionale lineare si vuole studiare il comportamento di un corpo soggetto a condizioni di piccoli spostamenti imposti. Si tratta sempre di una trave a sbalzo, a forma di parallelepipedo a base quadrata che misura 10 cm lungo x e y, mentre misura 1 m lungo z (Fig. 7.17)

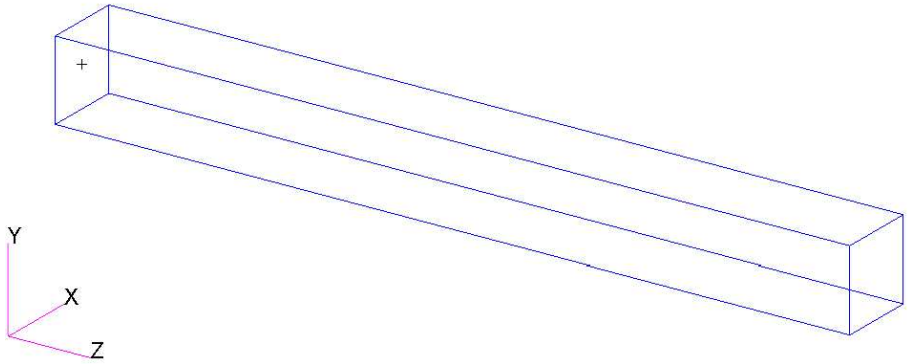


Figura 7.17: Rappresentazione del corpo modellato nel software Patran

Il materiale utilizzato è l'alluminio (Tab. 4.1), ovvero $E = 70 \cdot 10^9\, Pa$ e $\nu = 0.35$. L'associazione dei parametri utilizzati ad un materiale presente nel mondo reale aiuta solo a figurarsi il problema, ma non è un limite per la ricerca di una soluzione. Infatti, nel caso insorgessero tensioni superiori a quelle di snervamento o rottura, ciò non comprometterebbe la validità delle equazioni: sarebbe solo un caso non replicabile nella realtà ma matematicamente corretto. Tramite le equazioni di congruenza (Eq. 4.33)

$$\begin{aligned}
\varepsilon_{xx} &= \frac{\partial u}{\partial x}, \quad \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \\
\varepsilon_{yy} &= \frac{\partial v}{\partial y}, \quad \varepsilon_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right), \\
\varepsilon_{zz} &= \frac{\partial w}{\partial z}, \quad \varepsilon_{xz} = \frac{1}{2}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right),
\end{aligned}$$

le equazioni di legame costitutivo (Eq. 4.48)

$$\begin{aligned}
\sigma_{xx} &= (2\mu + \lambda)\,\varepsilon_{xx} + \lambda\,(\varepsilon_{yy} + \varepsilon_{zz}), \quad \sigma_{xy} = 2\mu\,\varepsilon_{xy}, \\
\sigma_{yy} &= (2\mu + \lambda)\,\varepsilon_{yy} + \lambda\,(\varepsilon_{xx} + \varepsilon_{zz}), \quad \sigma_{xz} = 2\mu\,\varepsilon_{xz}, \\
\sigma_{zz} &= (2\mu + \lambda)\,\varepsilon_{zz} + \lambda\,(\varepsilon_{xx} + \varepsilon_{yy}), \quad \sigma_{yz} = 2\mu\,\varepsilon_{yz}
\end{aligned}$$

e le equazioni di equilibrio (Eq. 4.11)

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} + X = 0,$$
$$\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} + Y = 0,$$
$$\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + Z = 0$$

si descrive matematicamente il problema lineare elastico 3D, a cui vanno aggiunte le condizioni al contorno. Nel problema in questione non vi sono forze di volume, ovvero $X = Y = Z = 0$, mentre le condizioni al contorno sono rappresentate unicamente da spostamenti imposti, quindi senza imporre alcuno stato tensionale. Si tratta sempre di trave a sbalzo, il che implica che una faccia quadrata abbia $u = v = w = 0$, mentre, per rimanere in campo lineare, si è imposto all'estremità opposta piccoli spostamenti lungo y e lungo x che misurano rispettivamente $v = 0.001\,m$ e $u = 0.0005\,m$.

Sulla base dei casi studiati in precedenza, e considerando gli ordini di grandezza presenti all'interno del problema, si è optato direttamente per una soluzione ridimensionalizzata. In questo caso non si cercano le soluzioni adimensionalizzando le grandezze incognite, ma cambiando l'unità di misura con le quali si vogliono rappresentare. Nelle equazioni della meccanica le varie grandezze sono considerate utilizzando il sistema internazionale, ovvero spostamenti in metri (m), forze in Newton (N) e tensioni in Pascal (Pa). La strategia escogitata vuole trovare, invece, una soluzione che abbia gli spostamenti in millimetri $(mm = 10^{-3}m)$ e le tensioni in megapascal $(MPa = 10^6 Pa = N/mm^2)$, mentre gli input rappresentati dalle coordinate spaziali rimangono sempre in metri (m). Questo implica che le equazioni appena presentate forniscono i risultati negli ordini di grandezza attesi a meno di fattori moltiplicativi.

Si trattano quindi le proprietà del materiale in megapascal (MPa)

$$E = 7 \cdot 10^4\,\text{MPa}, \quad G = \frac{E}{2(1+\nu)} = \frac{7 \cdot 10^4}{2.7}\,\text{MPa},$$

mantenendo $\nu = 0.35$ in quanto adimensionale, che andranno poi a ridefinire i parametri μ e λ. Inoltre, nelle equazioni di congruenza e di equilibrio, ove sono presenti derivazioni rispetto alle coordinate x, y, z, compare il fattore moltiplicativo 10^{-3} per tutte le singole derivate parziali, in modo da riequilibrare gli ordini di grandezza tra input (metri) e output (millimetri e megapascal)

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} \cdot 10^{-3}, \quad \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \cdot 10^{-3},$$

$$\varepsilon_{yy} = \frac{\partial v}{\partial y} \cdot 10^{-3}, \quad \varepsilon_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \cdot 10^{-3},$$

$$\varepsilon_{zz} = \frac{\partial w}{\partial z} \cdot 10^{-3}, \quad \varepsilon_{xz} = \frac{1}{2}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) \cdot 10^{-3},$$

$$\left(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z}\right) \cdot 10^{-3} = 0,$$

$$\left(\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z}\right) \cdot 10^{-3} = 0,$$

$$\left(\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z}\right) \cdot 10^{-3} = 0.$$

Nonostante nelle equazioni di equilibrio il fattore 10^{-3} si possa semplificare, questo in realtà risulta molto utile perché aiuta a stabilizzare il training della rete. Si ridefiniscono poi i valori delle condizioni al contorno ridimensionati

$$u = 0.5\,mm \quad \text{e} \quad v = 1\,mm.$$

7.7.1 Modello FEM

Per allenare la rete neurale e validarne i risultati ottenuti, viene costruito un modello sul quale verrà effettuata un'analisi FEM con i dati appena forniti. Viene creato il materiale alluminio e viene generata la geometria descritta in precedenza. La mesh applicata è frutto di ripetuti tentativi: a seguito di più analisi si è dimostrata necessaria una maggiore densità di elemento lungo l'asse z rispetto agli assi x e y. La dimensione scelta è quindi di 18x18x1200 elementi rispettivamente lungo x, y e z, generando un totale di 388800 elementi e 433561 nodi. Lungo la faccia contrassegnata con $+$, posta a $z = 0$, viene applicato un incastro che impedisce ai nodi che vi appartengono di muoversi; alla faccia opposta vengono imposti gli spostamenti $u = 0.0005$ e $v = 0.001$ (Fig. 7.18)

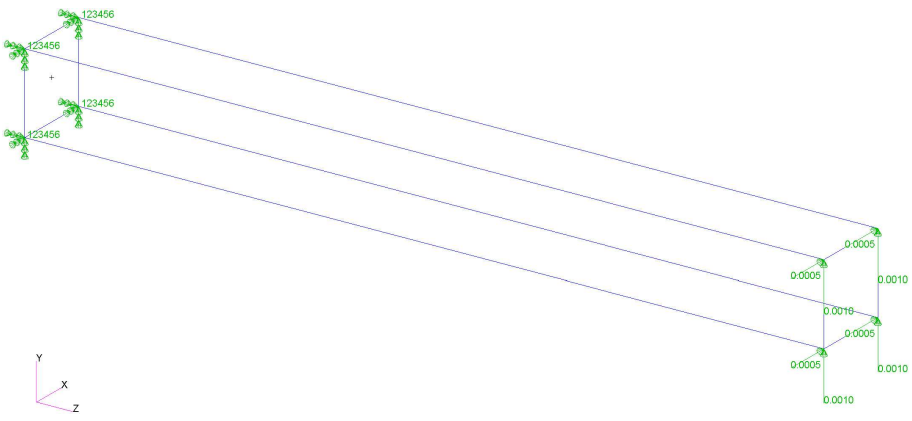


Figura 7.18: Rappresentazione della geometria e dei vincoli imposti

Viene eseguita un'analisi statica lineare (in Patran chiamata *SOL 101*) e si caricano i risultati all'interno del modello così da poterli rappresentare (Fig. 7.19, 7.20)

Si nota come è presente un picco di stress in prossimità dell'incastro di 27.8 MPa e sulla maggior parte della parte esterna della trave vi è la tensione minima registrata di 61.2 kPa. La grandezza degli spostamenti ha un andamento simil lineare che va da 0 all'incastro fino a 1.12 mm sul lato opposto.

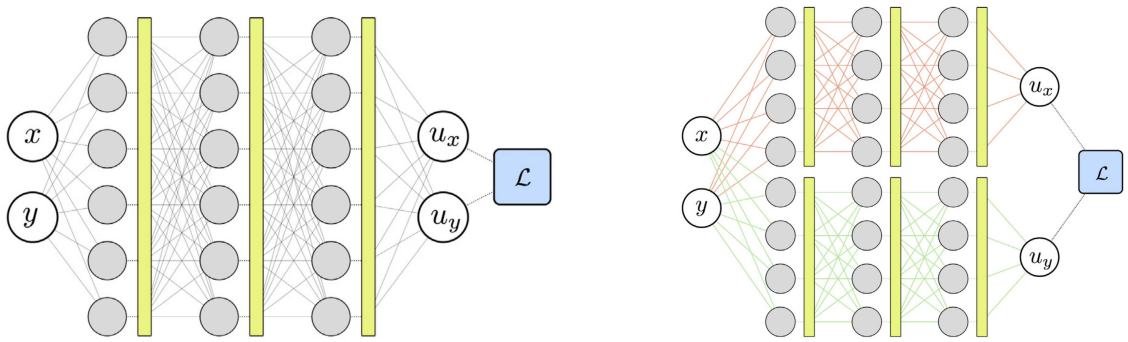
I risultati presentano le unità di misura nel sistema internazionale, perciò una volta che verranno estratti i datapoints per i training delle reti bisognerà ridimensionarne l'ordine di grandezza. Vengono estratti in tutto 24 datapoint dai 433561 nodi totali. Questi punti sono disposti 4 per piano, in 6 piani perpendicolari all'asse z e paralleli tra loro, equispaziati tra $z = 0$ e $z = 1$ come rappresentato in Fig. 7.21.

I dati estratti sono poi inseriti in un file *.csv* (*comma separated value*) e utilizzati nel codice python.

7.7.2 Architettura della rete

La fase di training è quella più complessa di tutto il processo, poiché è quella che definisce se il modello creato è in grado di approssimare le soluzioni che si stanno cercando. Inizialmente è stata pensata una singola rete neurale con 3 input e 3 output, rappresentativi rispettivamente le coordinate spaziali x, y, z e gli spostamenti u, v, w. Con questa rete è quindi possibile ricavare i valori delle tensioni (σ) e delle deformazioni (ε) tramite il gradiente degli spostamenti rispetto alle coordinate e i parametri del materiale secondo le equazioni di congruenza e di legame costitutivo. L'utilizzo di uno strumento basato su differenziazione automatica consente di ottenere derivate esatte, senza approssimazioni numeriche, trattando la rete neurale come una vera e propria funzione matematica.

Un esempio è presentato da Raissi et al. [20] dove viene studiato il comportamento di una piastra soggetta a determinate forze di volume. L'esempio riportato studia una piastra nel dominio 2D, ove le coordinate spaziali sono rappresentate da x e y mentre gli spostamenti da $u = u_x$ e $v = u_y$.



(a) Singola rete che presenta input multipli e output multipli

(b) Più reti con input multipli e singoli output

Figura 7.22: Proposta di differenti architetture delle reti [20]

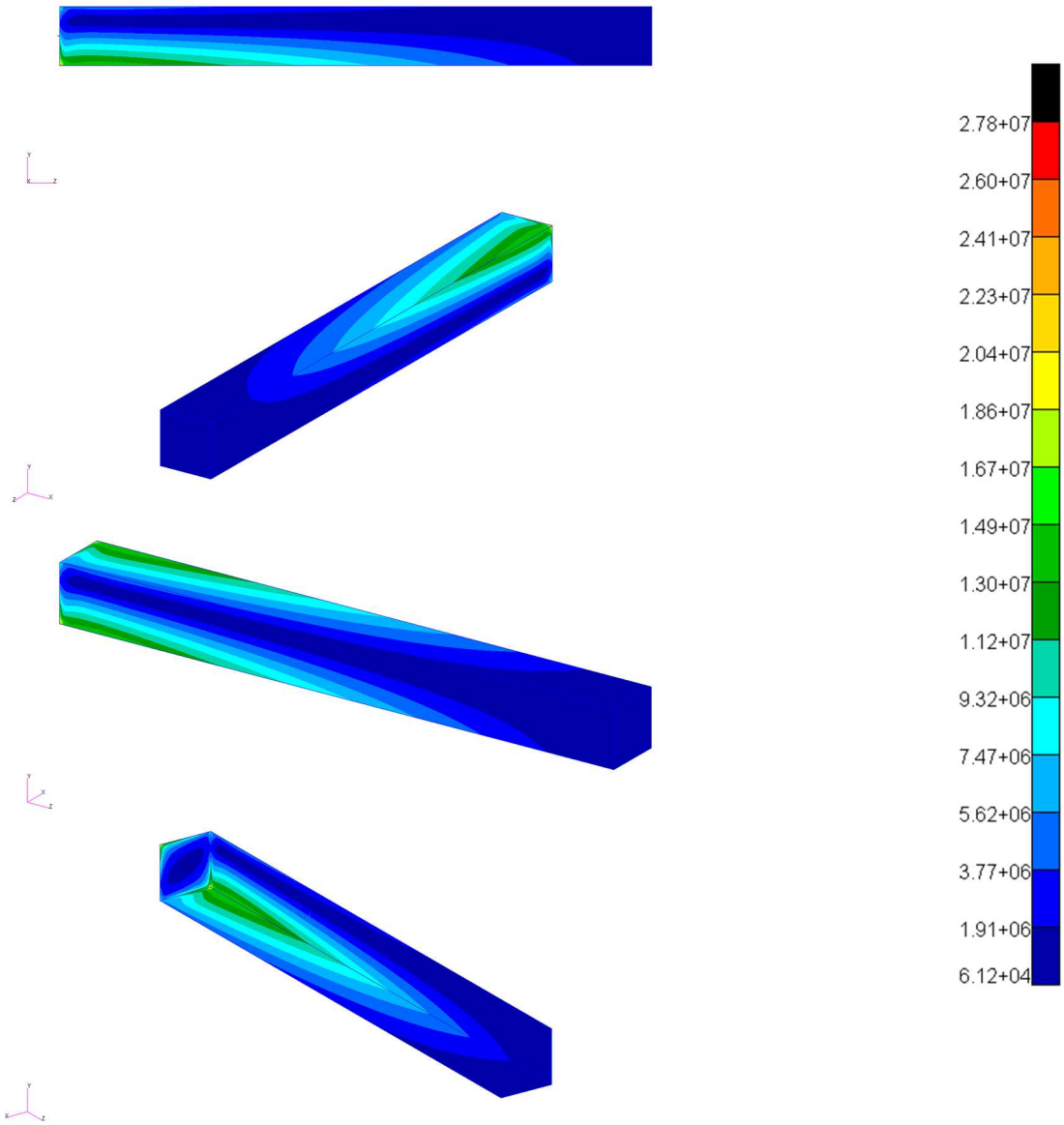


Figura 7.19: Tensioni di von Mises (Pa) rappresentate tramite una scala a colori su tutto il solido visto da più prospettive

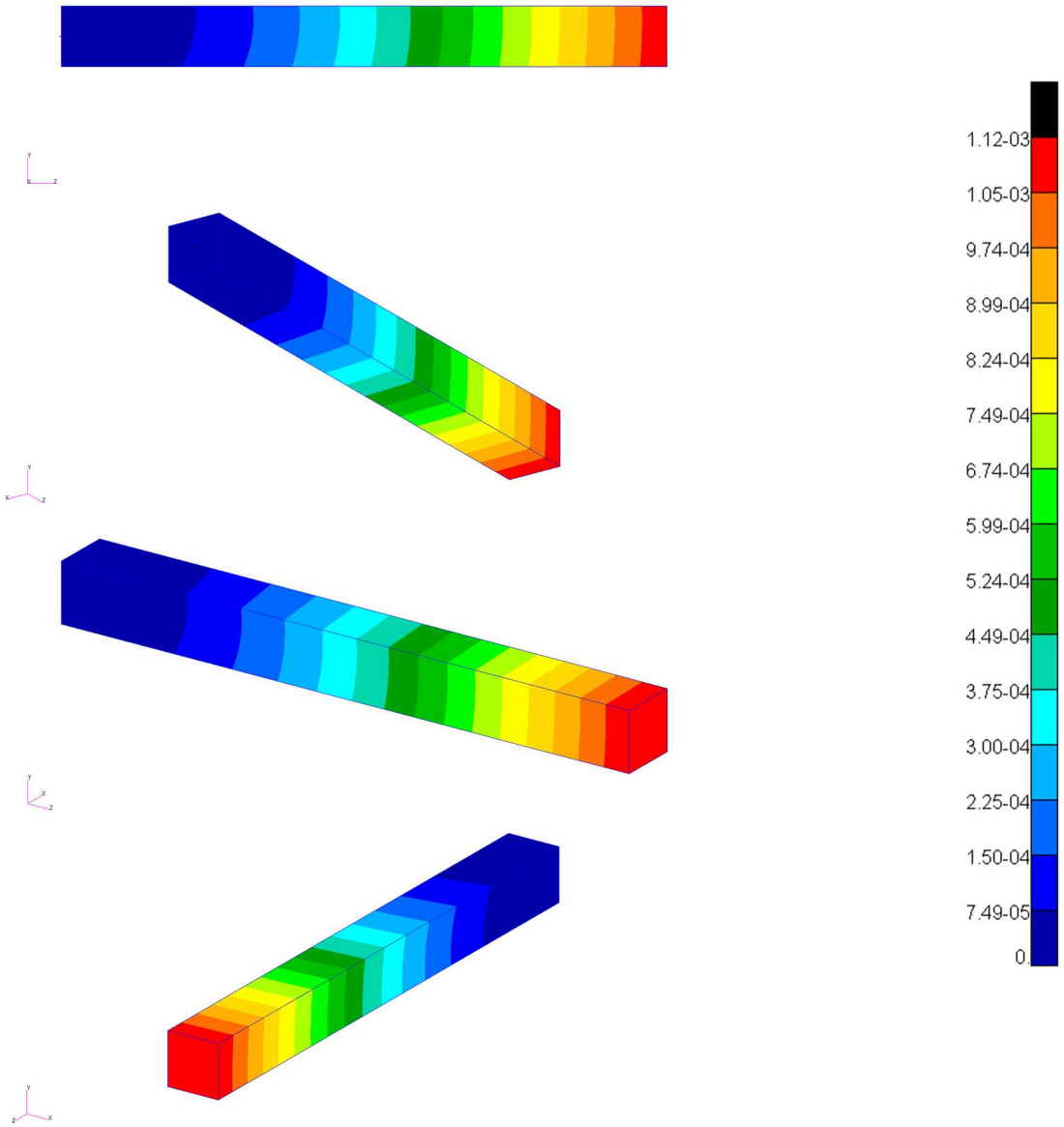


Figura 7.20: Modulo degli spostamenti (m) rappresentati tramite una scala a colori su tutto il solido visto da più prospettive

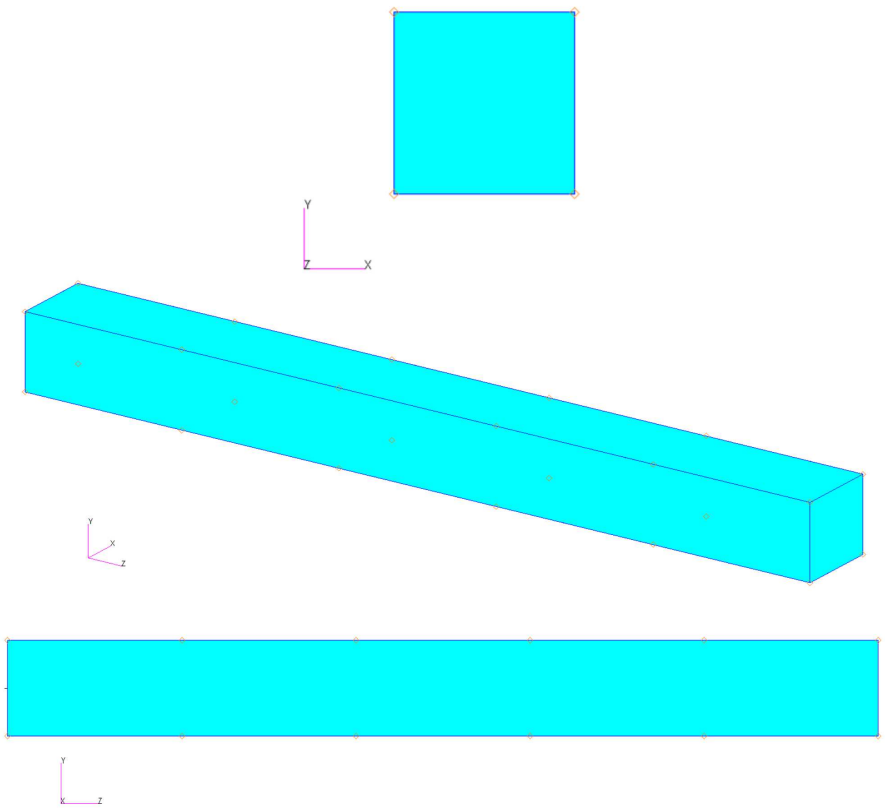


Figura 7.21: Nodi sulla superficie del solido dai quali vengono estratti i datapoints

Riferendosi all'esempio appena presentato, in Fig. 7.22 sono proposte due differenti scelte a livello di architettura della rete che possono essere utilizzate.

Per quanto utilizzare molteplici reti neurali possa sembrare una soluzione più macchinosa, questo le porta a rappresentare funzioni più semplici rispetto ad una più funzione più ampia e complessa con output multipli. In conclusione, la scelta adottata nel problema 2D riportato come esempio [20] verte sul definire 5 reti neurali (Fig. 7.23), ciascuna delle quali rappresenta una singola variabile di interesse del problema (u, v, σ_{xx}, σ_{yy}, σ_{xy}).

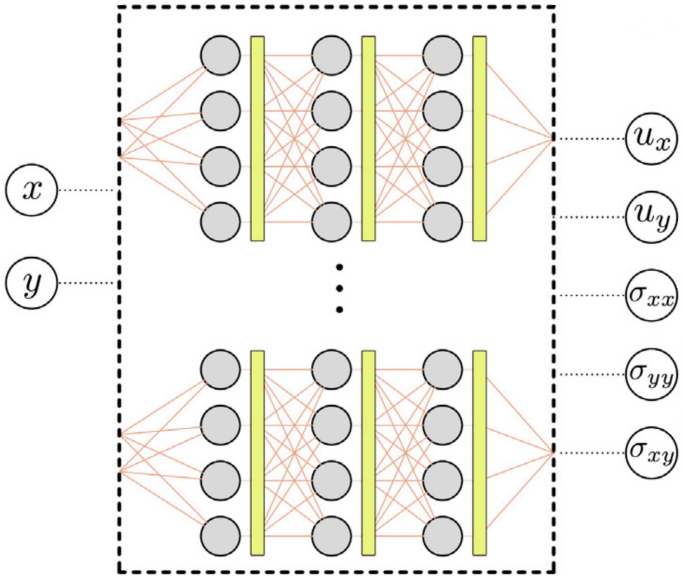


Figura 7.23: Architettura utilizzata nell'esempio presentato da Raissi et al. [20]

In un'analisi preliminare si è deciso di impiegare lo stesso approccio proposto nello studio [20], ma ampliandolo al caso 3D di interesse per questa tesi. Dopo molteplici tentativi, si è giunti alla conclusione che l'utilizzo di reti multiple era la scelta ottimale. Si è notato che il costo computazionale della fase di training passando da una a più reti diminuiva drasticamente, consentendo di allenarle in tempi ragionevoli (decine di minuti). Nella fase di training, l'utilizzo di una singola rete richiede un gran numero di derivazioni per la costruzione della funzione di costo sulle molteplici equazioni che governano la fisica ($\mathcal{L}_p$) e ciò portava a tempi di attesa troppo ingenti (si tratta di training da decine di ore).

Infine, ispirandosi al metodo proposto da Raissi et al. [20], la strategia adottata in questa tesi è virata verso l'utilizzo di 9 reti neurali, ciascuna associata ad una singola variabile di interesse del problema (u, v, w, σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{yz}, σ_{xz}) secondo lo schema 3-50-50-50-1. Le deformazioni (ε_{xx}, ε_{yy}, ε_{zz}, ε_{xy}, ε_{yz}, ε_{xz}) vengono poi ricavate per derivazione delle funzioni incognite u, v, w.

7.7.3 Dataset

Per il training delle 9 reti designate a descrivere il comportamento lineare della trave 3D, è stato pensato un dataset, come detto in precedenza, composto da diverse tipologie di elementi.

Collocation points

In primo luogo è stata creata una fitta rete di collocation points equispaziati all'interno di tutto il dominio 3D della trave. In Fig. 7.24 è rappresentata una rete di 11x11x101 punti ad emulare una mesh di 10x10x100 elementi cubici (in Patran chiamati Hex8) utilizzati in un'analisi FEM.

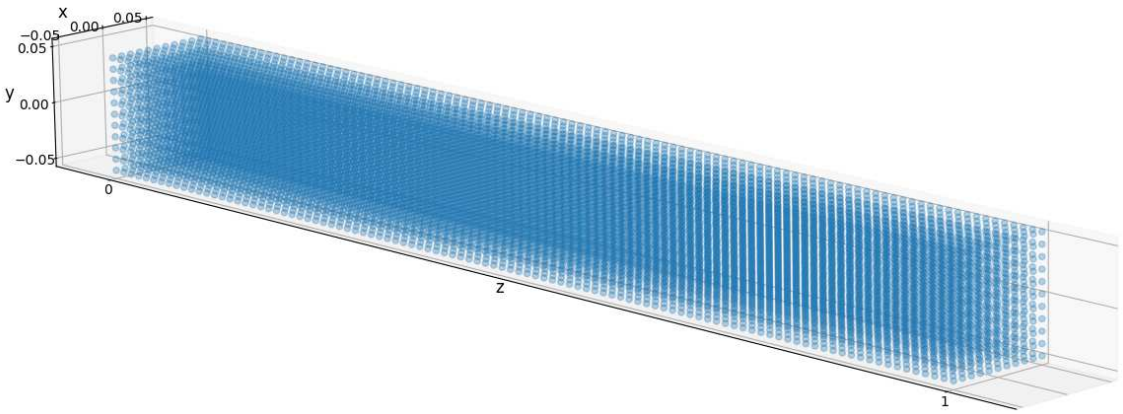


Figura 7.24: Rete di collocation points di 11 punti lungo x, 11 punti lungo y, 101 punti lungo z

Sono stati effettuati diversi tentativi con differenti dimensioni di reti di collocation points (11x11x101, 13x13x121, 21x21x201, 19x19x1201). Ciò che risulta è che maggiore è il numero di collocation points utilizzati durante il training e maggiore è la precisione della rete, con conseguente aumento del tempo di training. La memoria disponibile nella macchina fisica su cui viene eseguito il codice è un elemento importante da tenere in considerazione, poiché l'utilizzo di un eccessivo numero di collocation points obbliga a suddividere il dataset in più batch.

Per il problema analizzato in questo esempio sono stati considerati 11x11x101 collocation points, ovvero i 12221 nodi rappresentati in Fig. 7.24, in ognuno dei quali sarà valutata la physics loss

$$\begin{aligned}
\mathcal{L}_p = &\left(\sigma_{xx,x} + \sigma_{xy,y} + \sigma_{xz,z}\right)^2 + \\
&+ \left(\sigma_{xy,x} + \sigma_{yy,y} + \sigma_{yz,z}\right)^2 + \\
&+ \left(\sigma_{xz,x} + \sigma_{yz,y} + \sigma_{zz,z}\right)^2 + \\
&+ \left(2\mu\varepsilon_{xy} - \sigma_{xy}\right)^2 + \\
&+ \left(2\mu\varepsilon_{xz} - \sigma_{xz}\right)^2 + \\
&+ \left(2\mu\varepsilon_{yz} - \sigma_{yz}\right)^2 + \\
&+ \left(2\mu\varepsilon_{xx} + \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) - \sigma_{xx}\right)^2 + \\
&+ \left(2\mu\varepsilon_{yy} + \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) - \sigma_{yy}\right)^2 + \\
&+ \left(2\mu\varepsilon_{zz} + \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) - \sigma_{zz}\right)^2.
\end{aligned} \tag{7.21}$$

Vanno poi sommati i contributi di tutti i collocation points e successivamente divisi per il numero di punti N_p (notazione presa dall'Eq. 3.6), che in questo caso vale 12221.

Boundary points

Si conoscono a priori le condizioni che stanno al contorno del solido, da una parte per l'incastro e dall'altra per gli spostamenti imposti. I collocation points creati sulla superficie posta a $z = 0$ e su quella opposta a $z = 1$ (rispettivamente in blu e in arancione in Fig. 7.25) sono utilizzati anche per la parte di data loss

$$\mathcal{L}_b = (u - u^*)^2 + (v - v^*)^2 + (w - w^*)^2 \tag{7.22}$$

dove i termini con l'asterisco rappresentano i valori imposti.

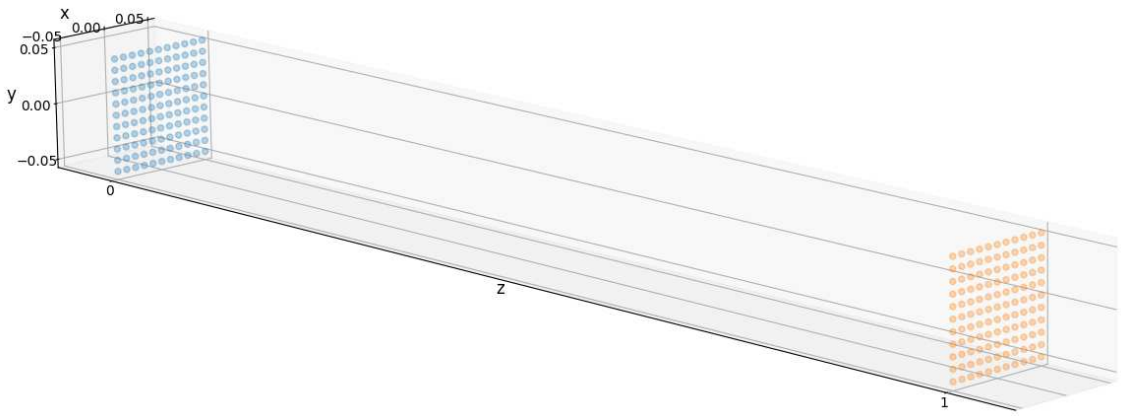


Figura 7.25: Collocation points posti a $z = 0$ (blu) e $z = 1$ (arancioni)

Per i punti a $z = 0$ vale

$$\mathcal{L}_b = u^2 + v^2 + w^2 \tag{7.23}$$

mentre per quelli a $z = 1$ vale

$$\mathcal{L}_b = (u - 0.5)^2 + (v - 1)^2 \tag{7.24}$$

nell'ottica di allenare le reti a fornire risultati in millimetri. Per ogni superficie vanno quindi sommati i contributi di tutti i punti che vi appartengono e successivamente divisi per il numero di punti N_b (notazione presa dall'Eq. 3.6), che in questo caso vale 121.

Data points

Nella definizione dei collocation points sono estratti 24 punti in cui vengono definiti i datapoints, tramite i risultati forniti dall'analisi FEM. In ognuno dei punti designati rappresentati in Fig. 7.26 viene valutato il data loss

$$\begin{aligned}\mathcal{L}_p = {} & (\sigma_{xx} - \sigma^*_{xx})^2 + (\sigma_{yy} - \sigma^*_{yy})^2 + (\sigma_{zz} - \sigma^*_{zz})^2 + \\ & + (\sigma_{xy} - \sigma^*_{xy})^2 + (\sigma_{yz} - \sigma^*_{yz})^2 + (\sigma_{xz} - \sigma^*_{xz})^2 + \\ & + (u - u^*)^2 + (v - v^*)^2 + (w - w^*)^2\end{aligned} \tag{7.25}$$

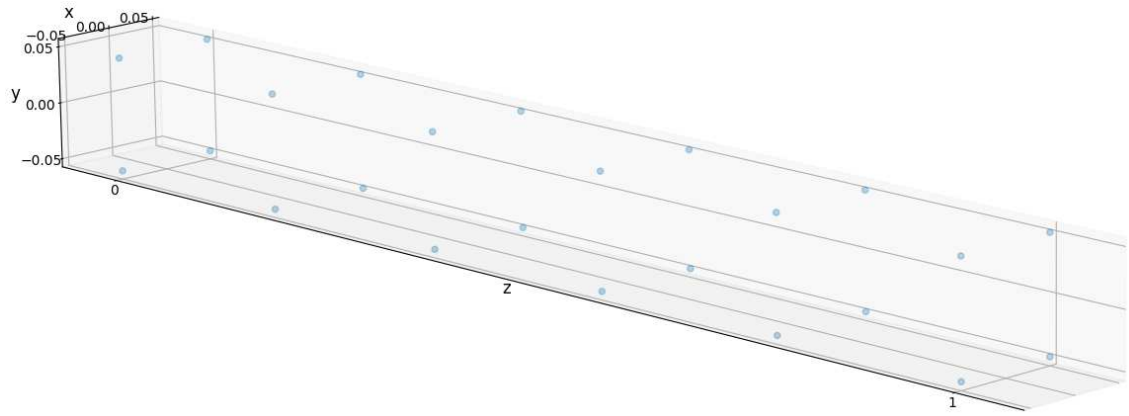


Figura 7.26: Collocation points utilizzati per valutare il data loss

dove i termini con l'asterisco rappresentano i risultati dell'analisi FEM ridimensionati in millimetri (mm) e in megapascal (MPa). Vanno quindi sommati i contributi di tutti e 24 i punti e successivamente divisi per il numero di data points N_b (notazione presa dall'Eq. 3.6), che in questo caso vale 24.

7.7.4 Training

Tutte e 9 le reti neurali vengono allenate con l'algoritmo Adam utilizzando i parametri di default della libreria Pytorch [11] ($\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$). Ad ogni epoch viene valutata la loss function sull'intero training dataset e si tiene traccia dell'epoch in cui raggiunge il suo minimo: a questo punto vengono eseguite le reti su un testing dataset, ovvero un set di collocation points che replicano i nodi presenti nel modello FEM. Viene creato quindi un set di collocation points simile a quello rappresentato in Fig. 7.24 ma di dimensione 19x19x1201, per un totale di 433561 punti, e si confrontano i valori ottenuti da una parte con le reti neurali e dall'altra con l'analisi FEM.

Seguendo lo stesso procedimento, sono state considerate due strategie distinte: nella prima si allenano le reti con una loss function priva della componente di physics loss denominata Prova NN,

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_b,$$

mentre la seconda con la formulazione completa denominata Prova PINN

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_b + \mathcal{L}_p.$$

Questo permetterà di confrontare quanto la componente fisica cambia la fase di training in termini di tempo impiegato e precisione finale.

7.7.5 Risultati

Di seguito sono rappresentati i risultati forniti dalle reti neurali e il confronto con i valori forniti dall'analisi FEM sul training dataset che va a formare il parallelepipedo simulato.

In Fig. 7.27 sono rappresentate le componenti degli spostamenti, incolonnati in modo da suddividere u, v e w nelle singole colonne. Lungo ogni colonna viene prima rappresentato il risultato dato dall'analisi lineare eseguita su Patran, a seguire i risultati forniti dalle reti neurali allenate prima per la Prova NN e poi per la Prova PINN, ognuno seguito dall'errore rispetto ai valori nominali estratti dall'analisi FEM. Gli errori rappresentano la differenza in valore assoluto tra i valori ottenuti dalla rete neurale e i valori nominali, quindi espressi in millimetri. Viene rappresentata la media degli errori accumulati in tutti i collocation points per le componenti u, v e w in Tab. 7.4

Lineare	**Prova NN**	**Prova PINN**
Errore u	0.00082	0.01213
Errore v	0.19127	0.19614
Errore w	0.18813	0.18856

Tabella 7.4: Medie degli errori misurate in millimetri (mm) delle componenti u, v, w valutati sui 433561 collocation points per la Prova NN e la Prova PINN del problema lineare

Successivamente sono rappresentate le 6 componenti delle tensioni, in Fig. 7.28 le componenti σ_{xx}, σ_{yy} e σ_{zz}, mentre in Fig. 7.29 le componenti σ_{xy}, σ_{yz} e σ_{xz}, secondo lo schema presentato in precedenza per gli spostamenti. Anche in questo caso gli errori rappresentano la differenza espressa in valore assoluto tra i valori ottenuti dalla rete neurale e i valori nominali, quindi espressi in megapascal. Viene rappresentata la media degli errori accumulati in tutti i collocation points per ogni componente σ in Tab. 7.5

Lineare	**Prova NN**	**Prova PINN**
Errore σ_{xx}	0.0759	0.32807
Errore σ_{yy}	0.09637	0.24731
Errore σ_{zz}	1.2017	0.61731
Errore σ_{xy}	0.00451	0.08842
Errore σ_{yz}	0.37307	0.58658
Errore σ_{xz}	0.35776	0.31144

Tabella 7.5: Medie degli errori misurate in megapascal (MPa) delle componenti σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{yz}, σ_{xz} valutati sui 433561 collocation points per la Prova NN e la Prova PINN del problema lineare

A seguire vengono rappresentate in Fig. 7.30 il valore di spostamento in modulo (s) e il valore della tensione di von Mises σ_{vM}, calcolati su ogni nodo secondo le equazioni Eq. 7.26, seguendo lo stesso schema utilizzato in precedenza.

$$s = \sqrt{u^2 + v^2 + w^2}$$

$$\sigma_{vM} = \sqrt{\frac{1}{2}\Big((\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2\Big) + 3\,(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{xz}^2)} \tag{7.26}$$

Viene rappresentata la media degli errori accumulati in tutti i collocation points sia per il modulo dello spostamento che per la tensione di von Mises in Tab. 7.6

Lineare	**Prova NN**	**Prova PINN**
Errore s	0.00721	0.01502
Errore σ_{vM}	1.44615	0.52422

Tabella 7.6: Medie degli errori misurate in millimetri (mm) e in megapascal (MPa) rispettivamente del modulo degli spostamenti s e del valore di tensione di von Mises σ_{vM} valutati sui 433561 collocation points per la Prova NN e la Prova PINN del problema lineare

Infine in Fig. 7.31 viene rappresentato l'andamento della loss function e il relativo tempo di esecuzione del training espresso in secondi (s). Quest'ultimo dato viene raggruppato in Tab. 7.7.

Lineare	**Prova NN**	**Prova PINN**
Training time t	536.93	1185.05

Tabella 7.7: Tempo di training espresso in secondi (s) per la Prova NN e la Prova PINN del problema lineare

7.8 Problema non lineare 3D

Nel problema tridimensionale non lineare, sulla base del problema lineare 3D, si vuole studiare il comportamento di un corpo soggetto a grandi spostamenti.

7.8.1 Differenze con il caso lineare

Le differenze rispetto al caso 3D lineare sono relativamente poche, ma di una notevole rilevanza. Innanzitutto si tratta sempre di una trave a sbalzo, della medesima forma e misure (Fig. 7.17). Il materiale utilizzato è leggermente modificato, ovvero viene considerato $E = 70 \cdot 10^6\, Pa$ e $\nu = 0.35$, mantenendo sempre le considerazioni fatte sulle tensioni e la validità matematica delle soluzioni. Il problema elastico è il medesimo, con la differenza che in questo caso viene considerato il tensore delle deformazioni di Green (Eq. 4.42) le cui componenti sono riportate di seguito

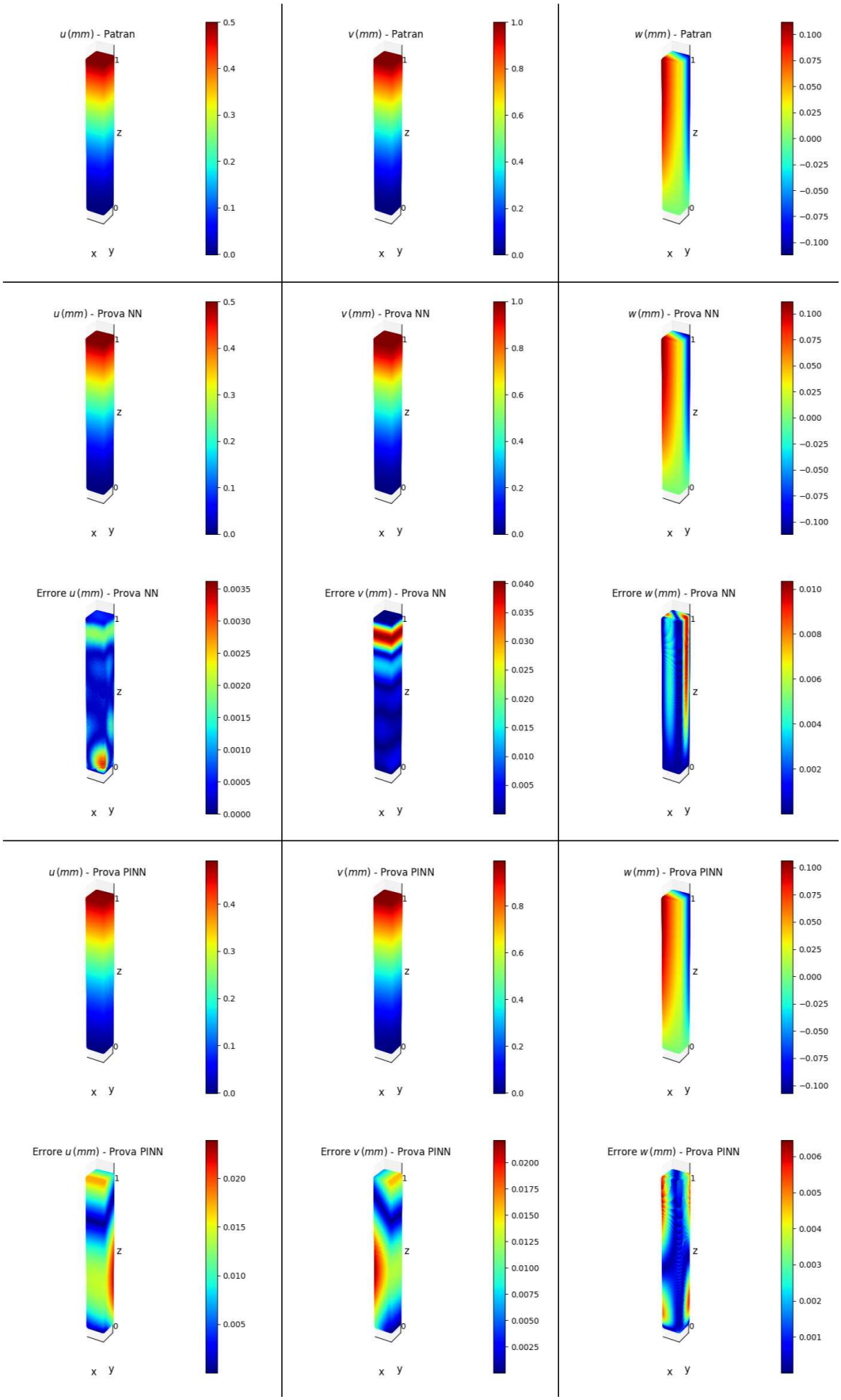


Figura 7.27: Valori delle componenti u, v, w in mm ottenuti dall'analisi FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN del problema lineare

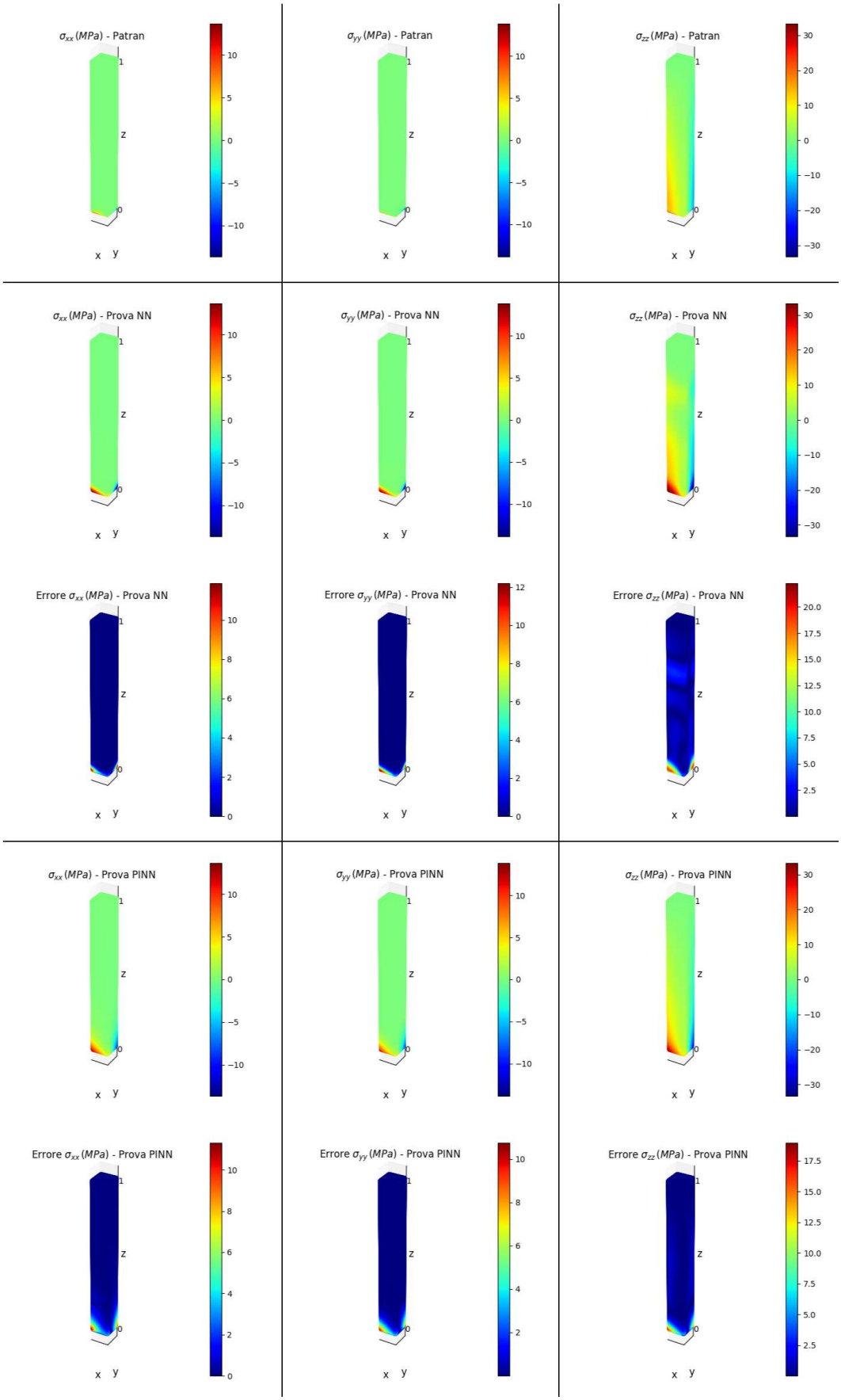


Figura 7.28: Valori delle componenti σ_{xx}, σ_{yy}, σ_{zz} in MPa ottenuti dall'analisi FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN del problema lineare

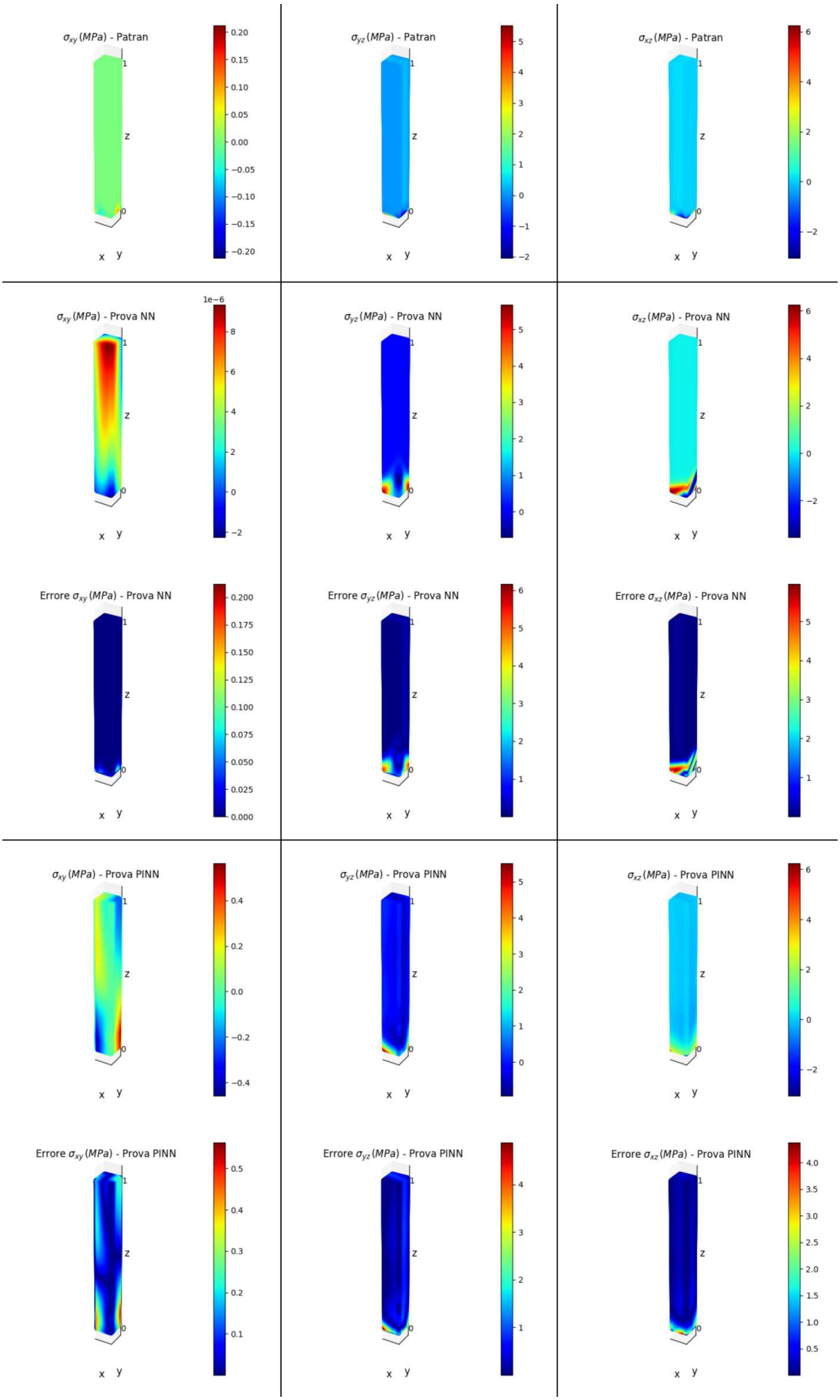


Figura 7.29: Valori delle componenti σ_{xy}, σ_{yz}, σ_{xz} in MPa ottenuti dall'analisi FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN del problema lineare

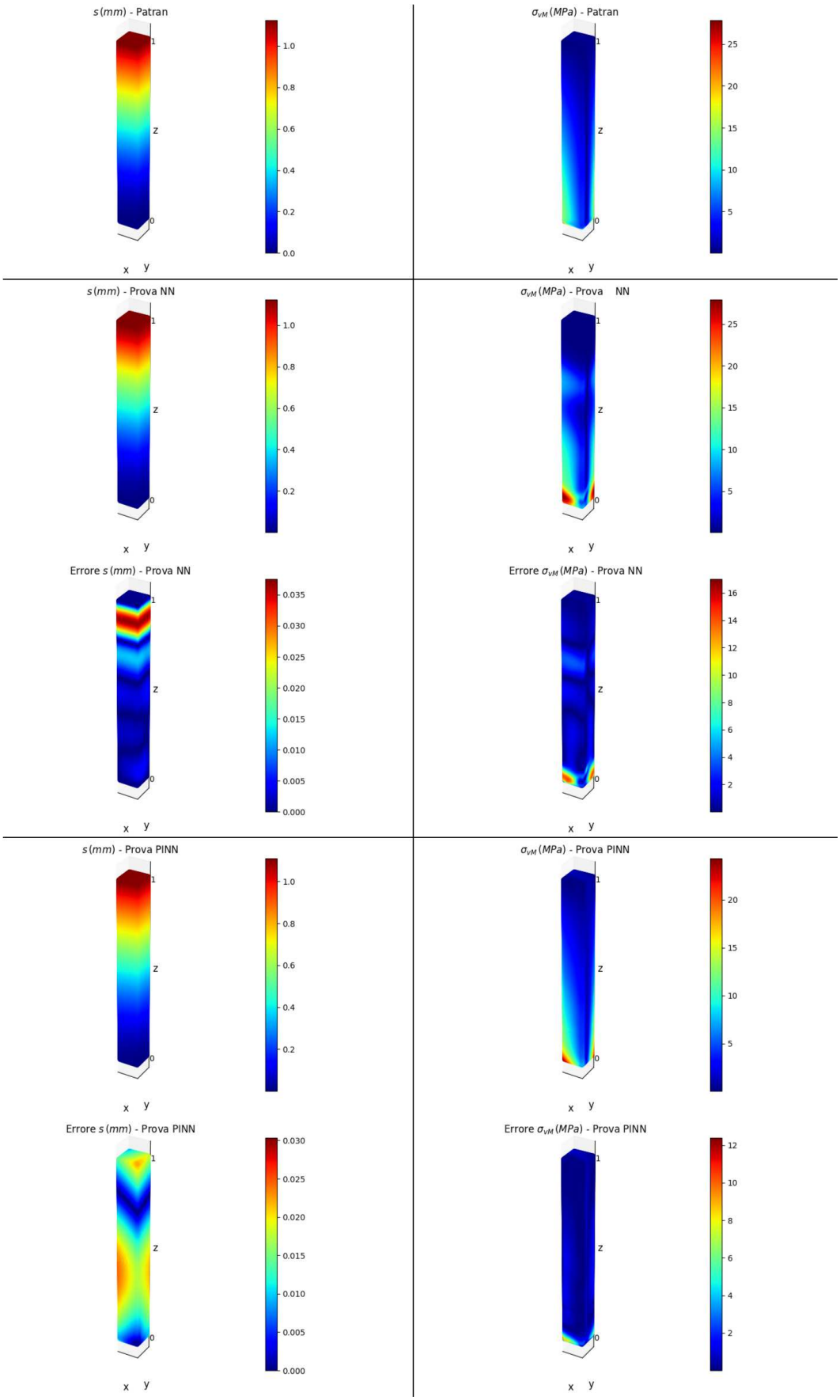
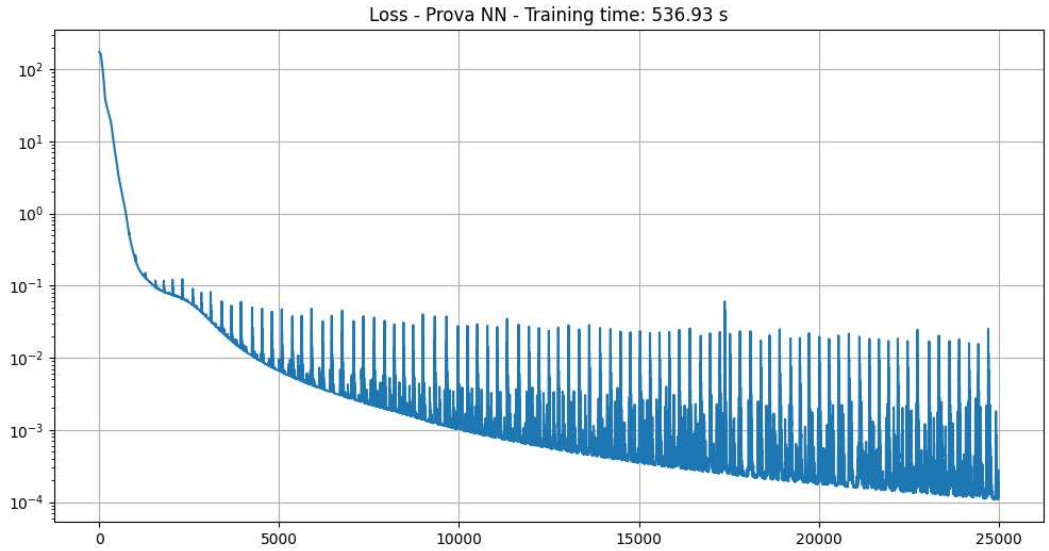
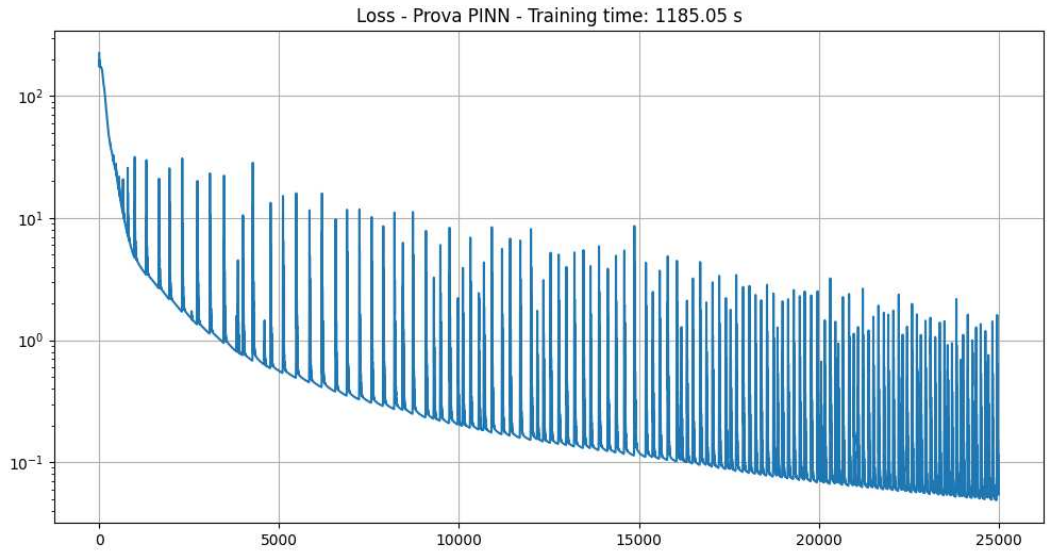


Figura 7.30: Valori di s e σ_{vM} calcolati dai risultati ottenuti dall'analisi FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN del problema lineare



(a)



(b)

Figura 7.31: Andamento della loss function nelle 25000 epochs di training della Prova NN (a) e della Prova PINN (b) per il problema 3D lineare

$$E_{xx} = \frac{\partial u}{\partial X} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial X}\right)^2 + \left(\frac{\partial v}{\partial X}\right)^2 + \left(\frac{\partial w}{\partial X}\right)^2\right]$$

$$E_{yy} = \frac{\partial v}{\partial Y} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial Y}\right)^2 + \left(\frac{\partial v}{\partial Y}\right)^2 + \left(\frac{\partial w}{\partial Y}\right)^2\right]$$

$$E_{zz} = \frac{\partial w}{\partial Z} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial Z}\right)^2 + \left(\frac{\partial v}{\partial Z}\right)^2 + \left(\frac{\partial w}{\partial Z}\right)^2\right]$$

$$E_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial X}\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X}\frac{\partial v}{\partial Y} + \frac{\partial w}{\partial X}\frac{\partial w}{\partial Y}\right]$$

$$E_{xz} = \frac{1}{2}\left(\frac{\partial u}{\partial Z} + \frac{\partial w}{\partial X}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial X}\frac{\partial u}{\partial Z} + \frac{\partial v}{\partial X}\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial X}\frac{\partial w}{\partial Z}\right]$$

$$E_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial Y}\frac{\partial u}{\partial Z} + \frac{\partial v}{\partial Y}\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y}\frac{\partial w}{\partial Z}\right]$$

Sono presenti le medesime forze di volume e il medesimo incastro, ma variano gli spostamenti imposti. Infatti sulla superficie opposta all'incastro sono applicati spostamenti lungo x e y che misurano rispettivamente $u = 0.1\,m$ e $v = 0.3\,m$. In questo caso le reti vengono allenate al fine di fornire risultati in metri (m) e megapascal (MPa), variando unicamente le proprietà del materiale

$$E = 70\,\text{MPa}, \quad G = \frac{E}{2(1+\nu)} = \frac{70}{2.7}\,\text{MPa},$$

ma senza modificare gli spostamenti imposti e le equazioni di congruenza e di equilibrio che definiscono il problema.

Allo stesso modo viene creato un equivalente modello FEM che riporta le caratteristiche del problema non lineare appena presentato. In questo caso l'analisi eseguita è un'analisi non lineare (in Patran chiamata *SOL 106*) e a seguito di questa si caricano i risultati all'interno del modello così da poterli rappresentare (Fig. 7.32, 7.33)

7.8.2 Risultati

Di seguito sono rappresentati i risultati forniti dalle reti neurali e il confronto con i valori forniti dall'analisi FEM sul training dataset che va a formare il parallelepipedo simulato.

In Fig. 7.35 sono rappresentati i valori delle componenti degli spostamenti, incolonnati come nel problema lineare. Viene rappresentata la media degli errori accumulati in tutti i collocation points per le componenti u, v e w in Tab. 7.8

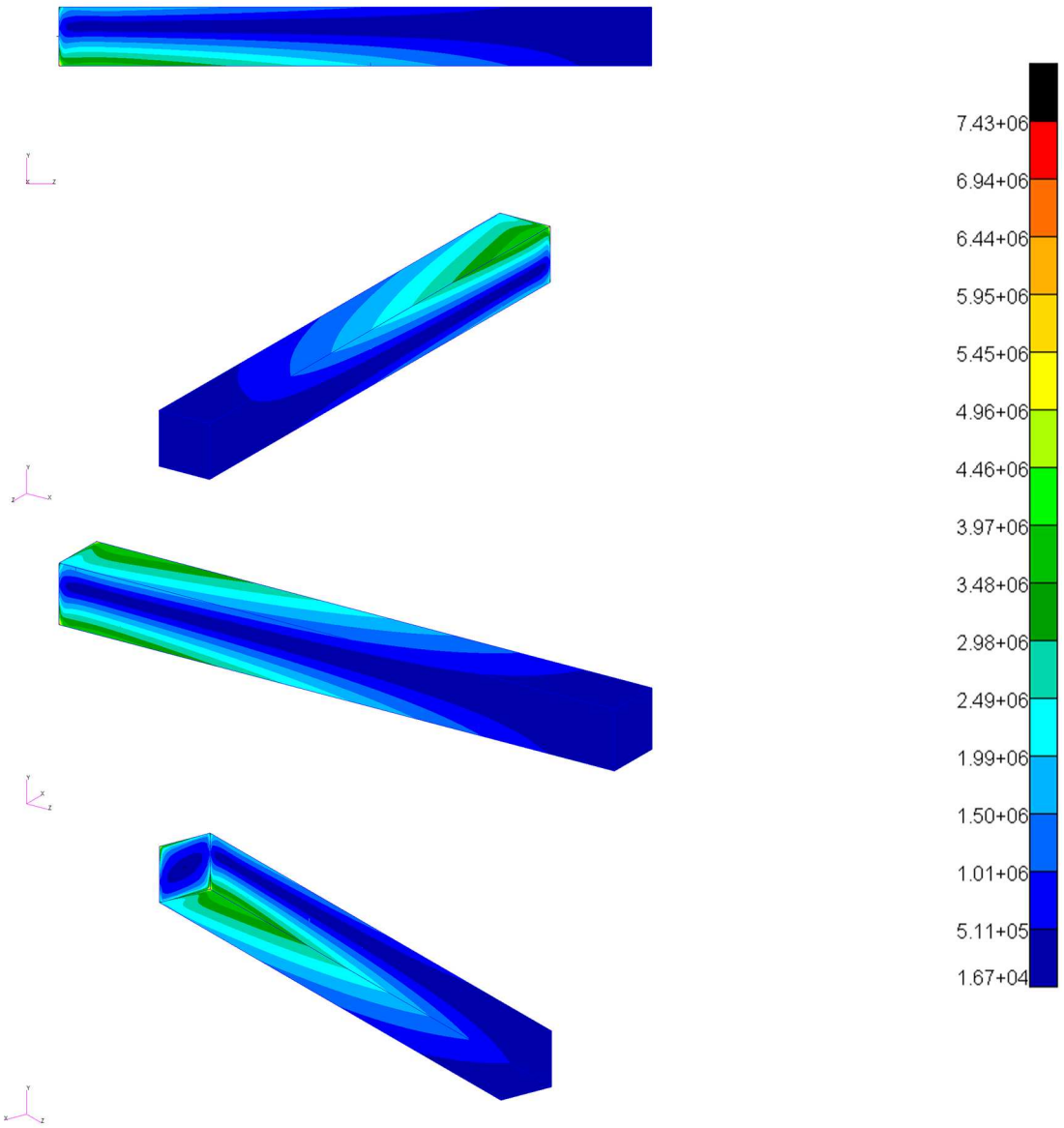


Figura 7.32: Tensioni di von Mises rappresentate tramite una scala a colori su tutto il solido visto da più prospettive

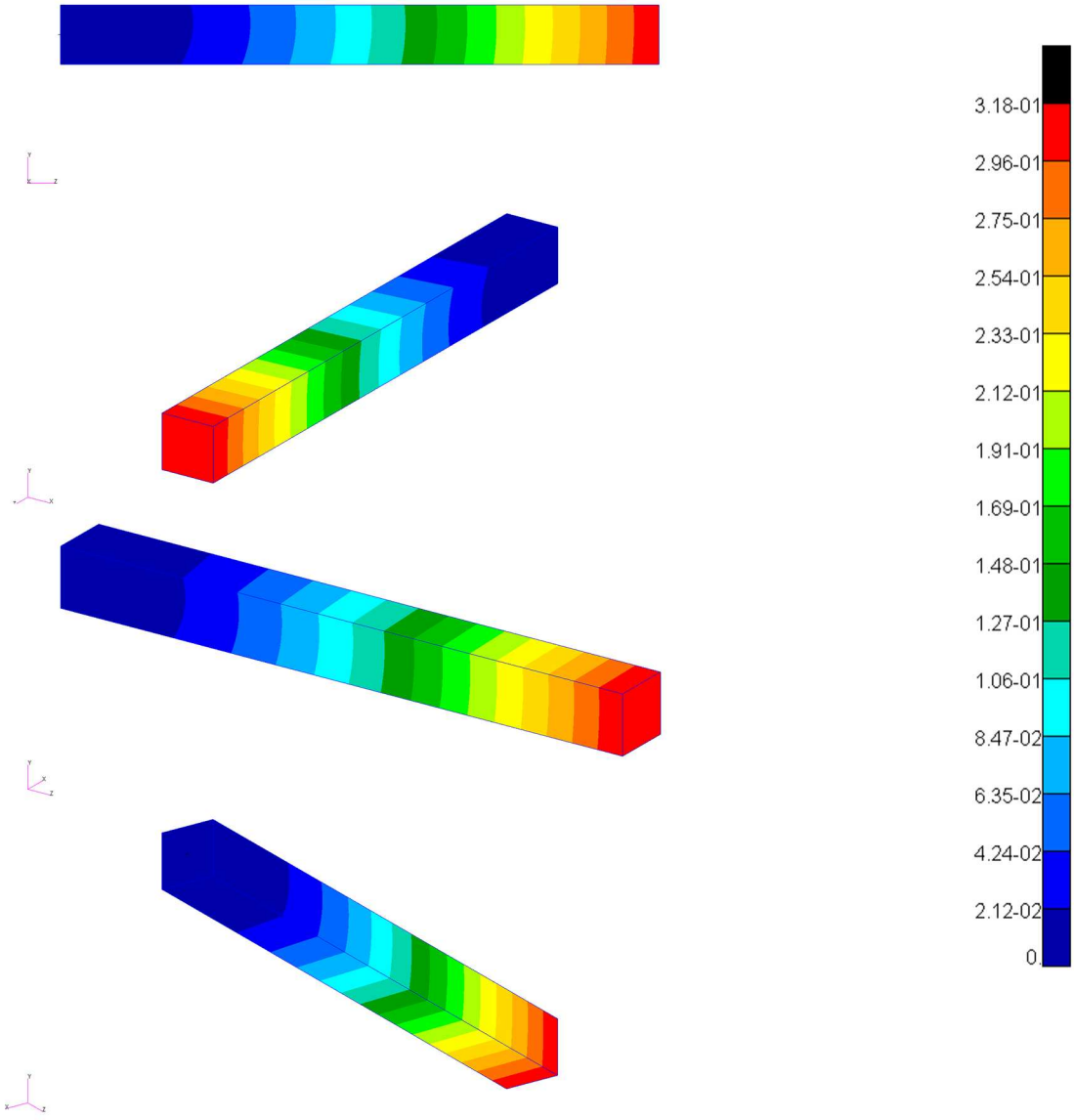


Figura 7.33: Modulo degli spostamenti rappresentati tramite una scala a colori su tutto il solido visto da più prospettive

Non Lineare	**Prova NN**	**Prova PINN**
Errore u	0.00041	0.00639
Errore v	0.07507	0.079
Errore w	0.03761	0.04786

Tabella 7.8: Medie degli errori misurate in metri (m) delle componenti u, v, w valutati sui 433561 collocation points per la Prova NN e la Prova PINN del problema non lineare

Successivamente sono rappresentate le 6 componenti delle tensioni, in Fig. 7.36 le componenti σ_{xx}, σ_{yy} e σ_{zz}, mentre in Fig. 7.37 le componenti σ_{xy}, σ_{yz} e σ_{xz}, secondo lo schema presentato in precedenza per gli spostamenti. Anche in questo caso gli errori rappresentano la differenza espressa in valore assoluto tra i valori ottenuti dalla rete neurale e i valori nominali, quindi espressi in megapascal. Viene rappresentata la media degli errori accumulati in tutti i collocation points per ogni componente σ in Tab. 7.9

Non Lineare	**Prova NN**	**Prova PINN**
Errore σ_{xx}	0.03867	0.07016
Errore σ_{yy}	0.03498	0.05021
Errore σ_{zz}	0.23558	0.16646
Errore σ_{xy}	0.00126	0.07165
Errore σ_{yz}	0.0878	0.21252
Errore σ_{xz}	0.12538	0.09005

Tabella 7.9: Medie degli errori misurate in megapascal (MPa) delle componenti σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{yz}, σ_{xz} valutati sui 433561 collocation points per la Prova NN e la Prova PINN del problema non lineare

Vengono rappresentate in Fig. 7.38 il valore di spostamento in modulo (s) e il valore della tensione di von Mises σ_{vm}, calcolati su ogni nodo secondo le equazioni Eq. 7.26, seguendo lo stesso schema utilizzato in precedenza. Viene rappresentata la media degli errori accumulati in tutti i collocation points sia per il modulo dello spostamento che per la tensione di von Mises in Tab. 7.10

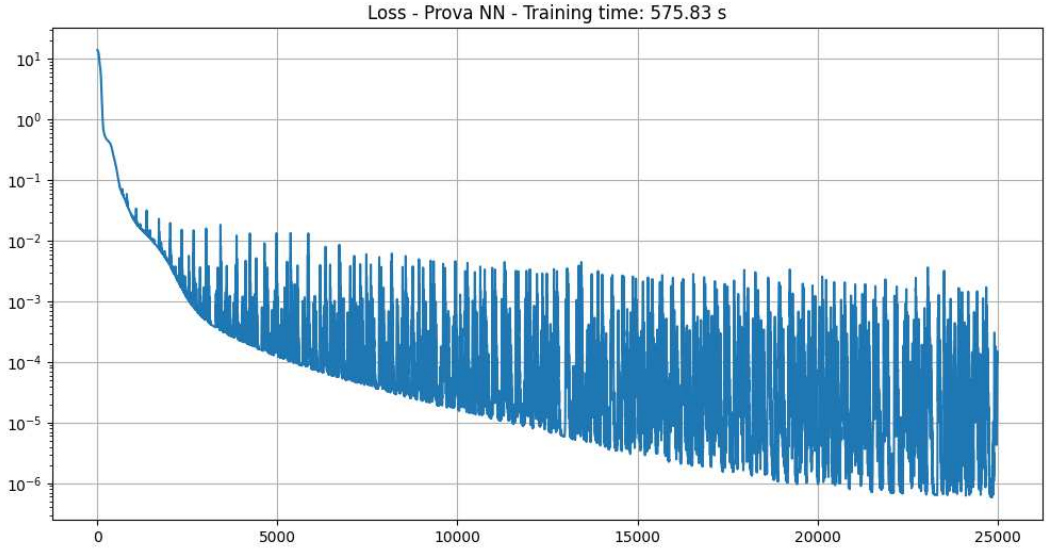
Non Lineare	**Prova NN**	**Prova PINN**
Errore s	0.00032	0.29779
Errore σ_{vM}	0.28136	2.04878

Tabella 7.10: Medie degli errori misurate in metri (m) e in megapascal (MPa) rispettivamente del modulo degli spostamenti s e del valore di tensione di von Mises σ_{vM} valutati sui 433561 collocation points per la Prova NN e la Prova PINN del problema non lineare

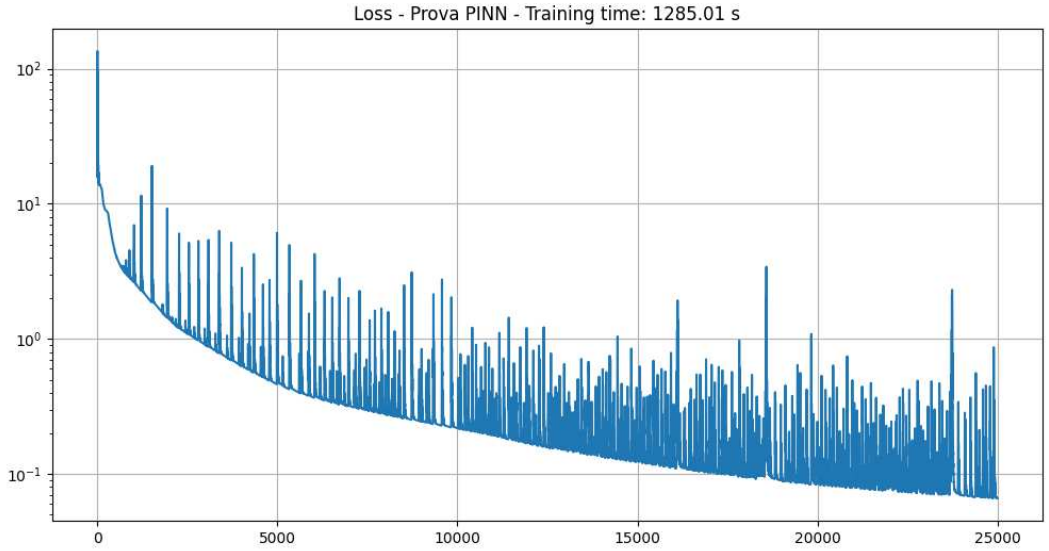
Infine in Fig. 7.34 viene rappresentato l'andamento della loss function e il relativo tempo di esecuzione del training espresso in secondi (s). Quest'ultimo dato viene raggruppato in Tab. 7.11.

Non Lineare	**Prova NN**	**Prova PINN**
Training time t	575.83	1285.01

Tabella 7.11: Tempo di training espresso in secondi (s) per la Prova NN e la Prova PINN del problema non lineare



(a)



(b)

Figura 7.34: Andamento della loss function nelle 25000 epochs di training della Prova NN (a) e della Prova PINN (b) per il problema 3D non lineare

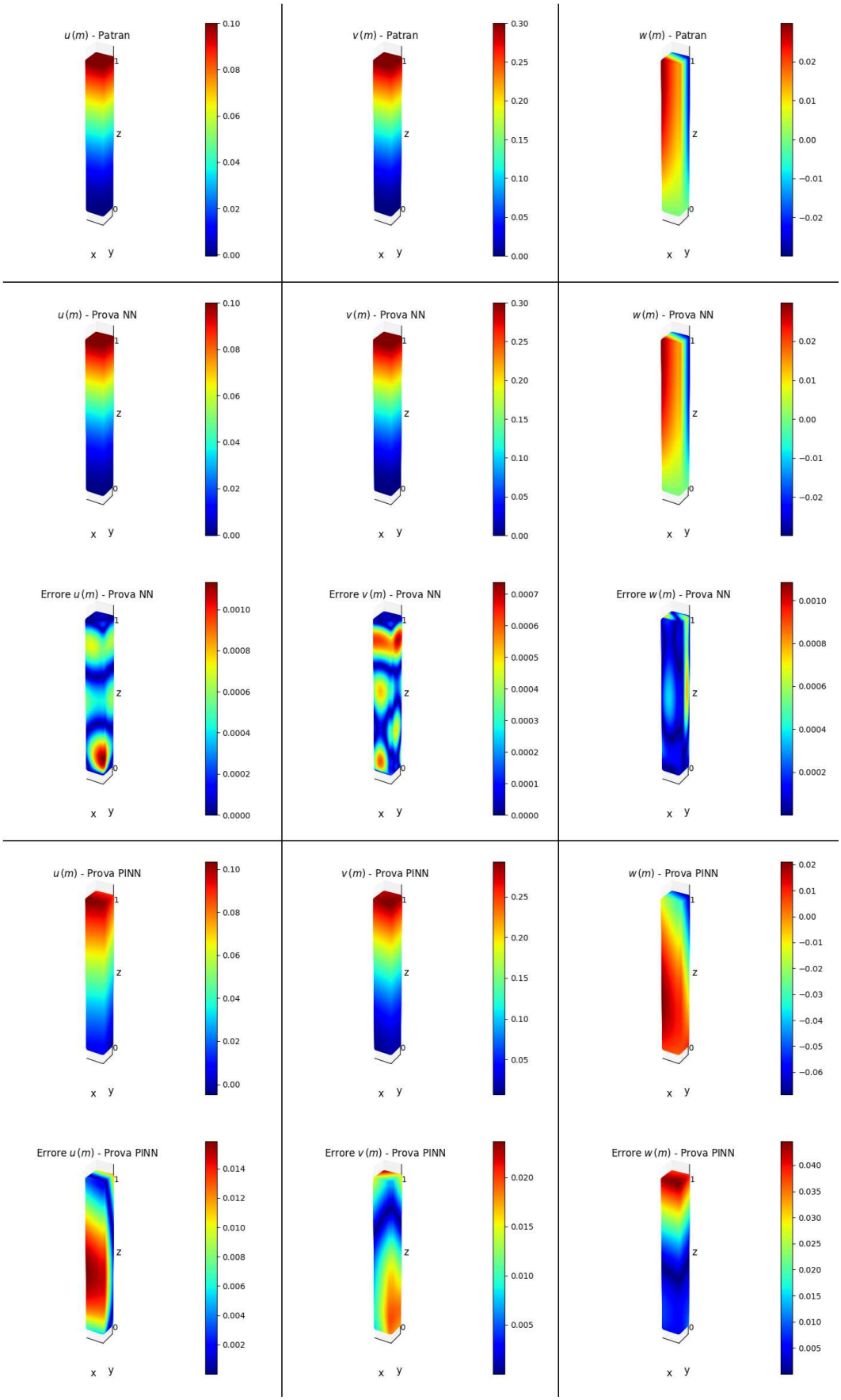


Figura 7.35: Valori delle componenti u, v e w in m ottenuti dall'analisi non lineare FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN non lineare

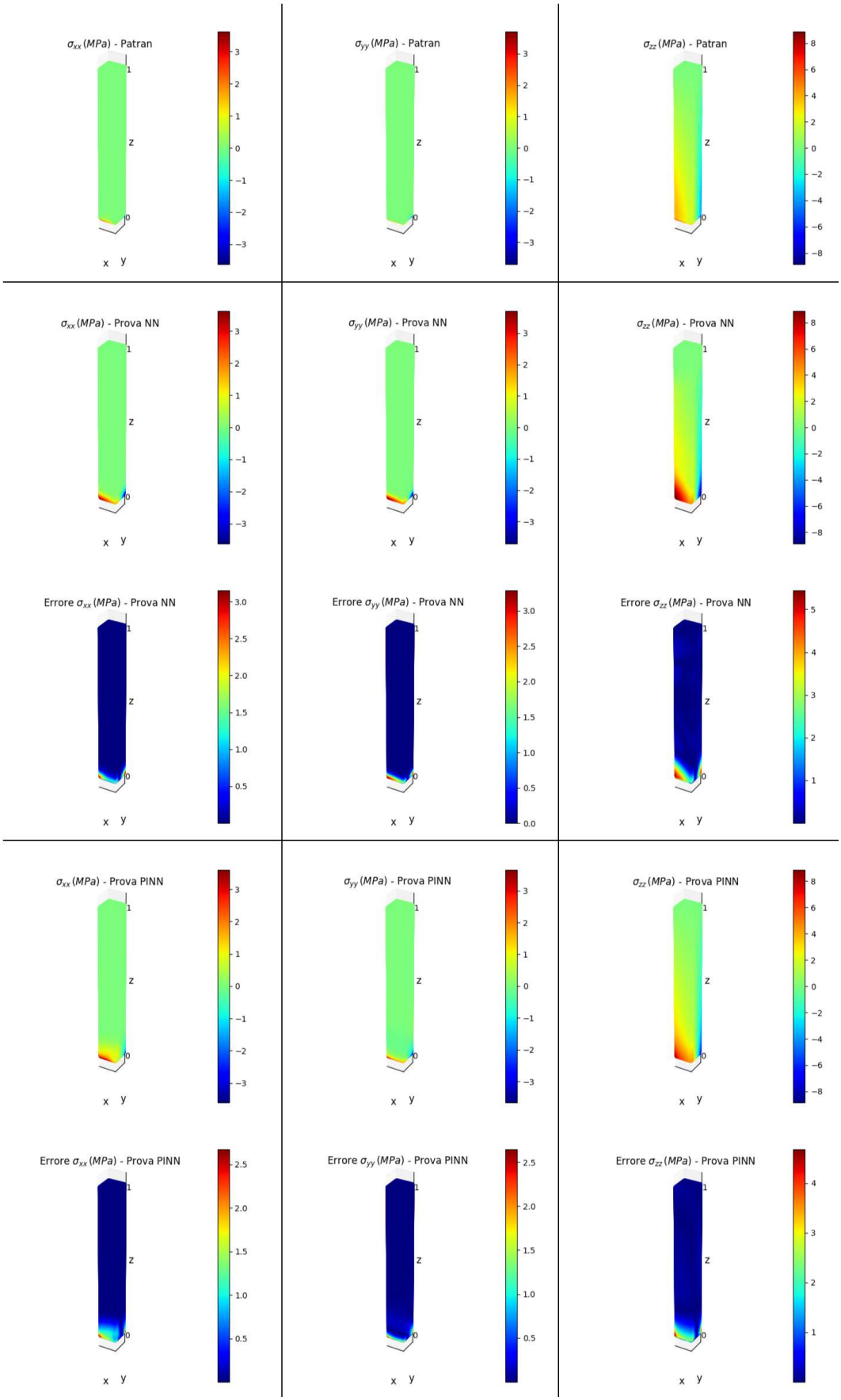


Figura 7.36: Valori delle componenti σ_{xx}, σ_{yy} e σ_{zz} in MPa ottenuti dall'analisi non lineare FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN non lineare

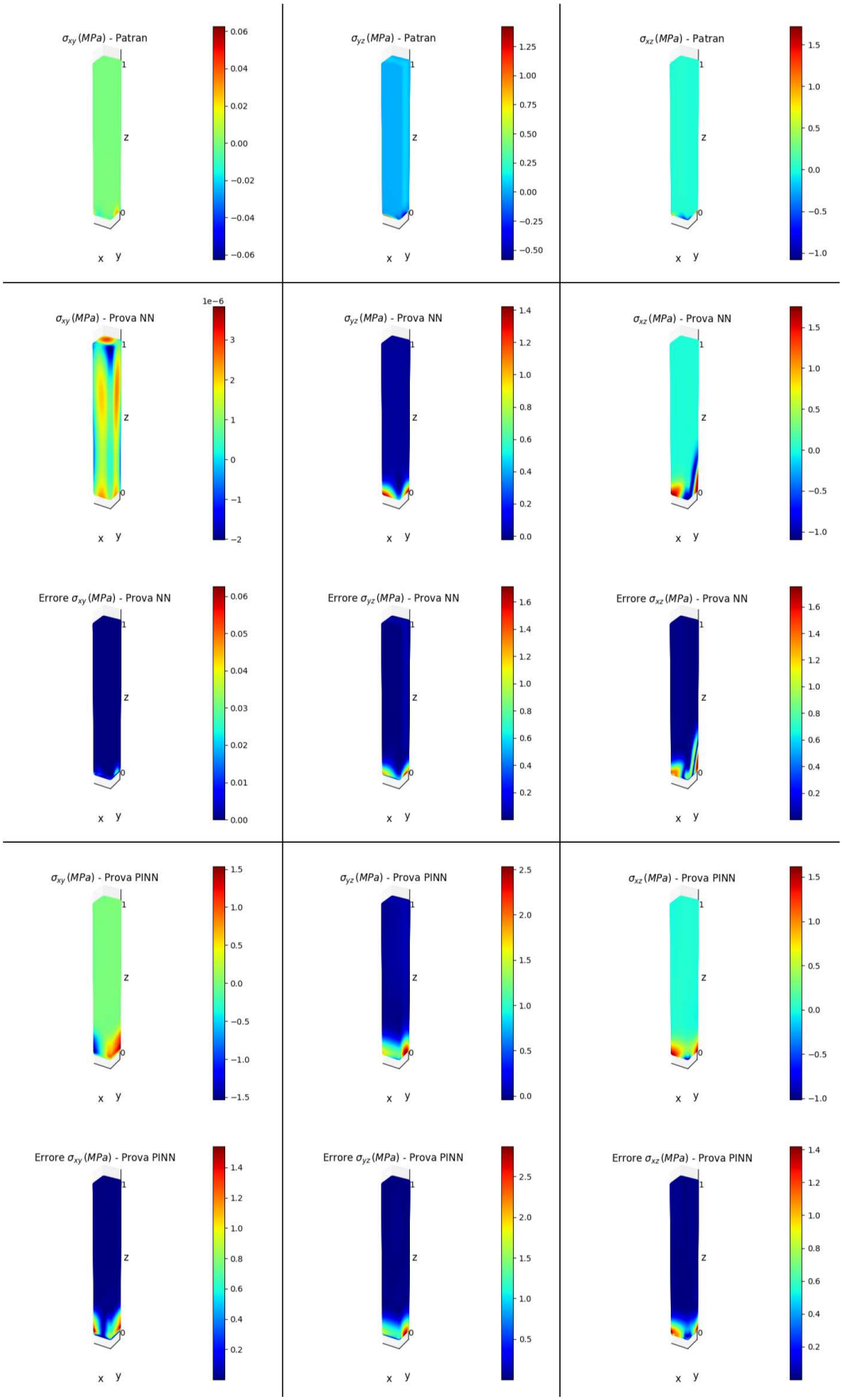


Figura 7.37: Valori delle componenti σ_{xy}, σ_{yz} e σ_{xz} in MPa ottenuti dall'analisi non lineare FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN non lineare

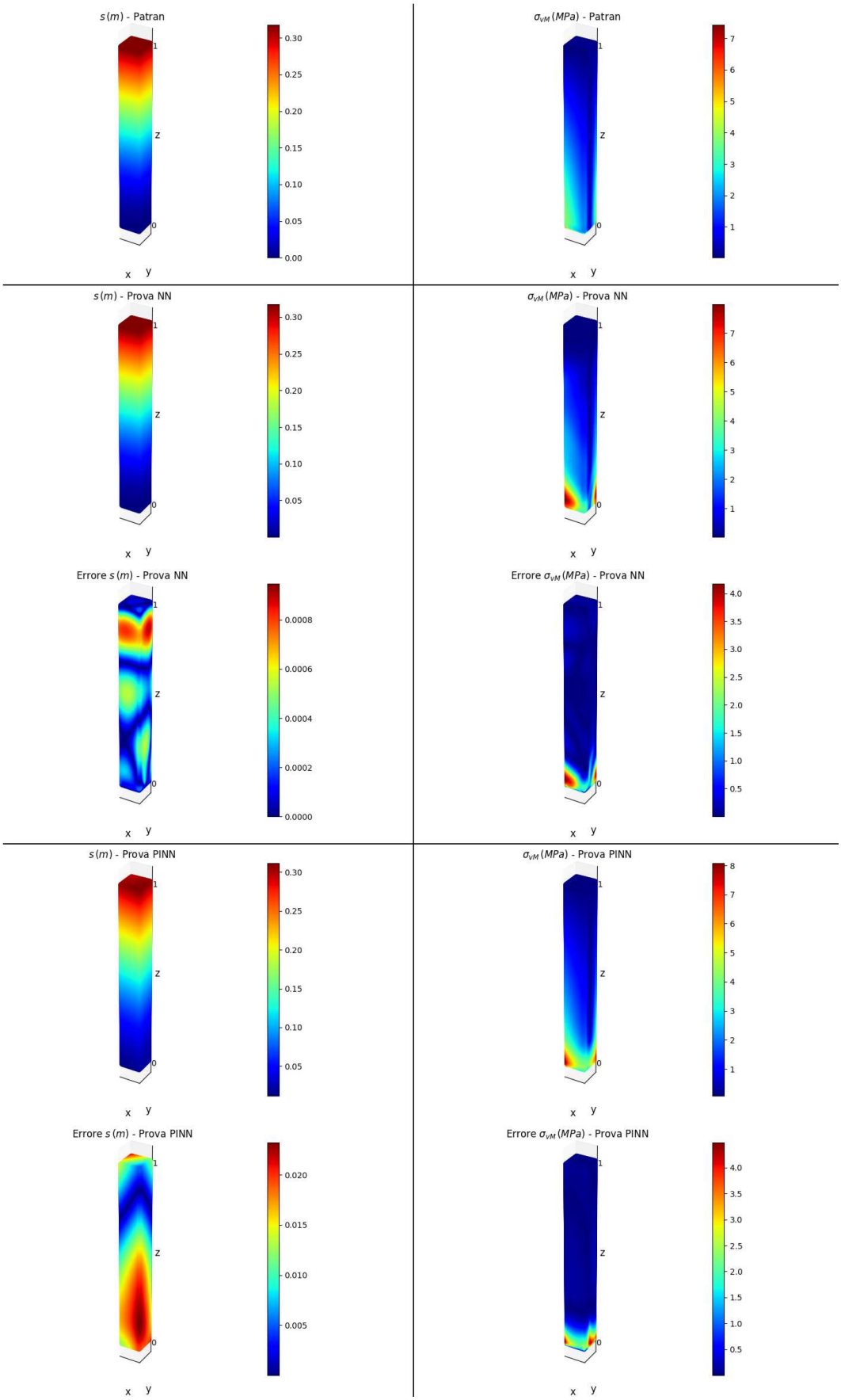


Figura 7.38: Valori del modulo dello spostamento e della tensione di von Mises rispettivamente in m e MPa calcolati dai risultati ottenuti dall'analisi non lineare FEM su Patran e dalle reti neurali nella Prova NN e nella Prova PINN non lineare

Capitolo 8

Conclusioni

In questo elaborato sono state prese in esame strategie per l'uso del machine learning alla meccanica computazionale. Nello specifico si è implementata la componente fisica nella funzione di costo durante la fase di training, trasformando così una rete neurale standard in una PINN particolarizzata al problema in esame. In generale è stato raggiunto l'obiettivo dello studio, ovvero quello di creare modelli surrogati tramite l'utilizzo di reti neurali in modo fornire un'alternativa ai software standard utilizzati per le simulazioni FEM. A training effettuato, il vantaggio consiste nel ridotto tempo di risposta che presenta una rete neurale rispetto ad un'analisi FEM tradizionale.

Nella risoluzione dei problemi 1D la rete neurale ha raggiunto l'obiettivo, ovvero approssimare una soluzione di un sistema di equazioni differenziali in un dominio monodimensionale. D'altra parte si può notare nei problemi 3D vi siano differenze minime tra la Prova NN e la Prova PINN. Le tabelle dimostrano che l'aggiunta della componente di physics loss nella fase di training di una PINN raddoppia il tempo di training, a causa dell'onerosità computazionale che ne consegue. Per entrambi i casi 3D vi sono situazioni in cui le soluzioni sono meglio rappresentate nella Prova PINN rispetto alla Prova NN e altri in cui accade l'opposto. L'aggiunta della componente fisica alla funzione di perdita non ha comportato un miglioramento dei risultati assoluto, ma solo in parte. Questo può essere attribuito a diverse cause: assenza di un problema adimensionalizzato, fattori λ non ottimali, condizioni al contorno scarse, numero di collocation points troppo basso, architettura delle reti poco efficace o errata scelta dell'algoritmo di ottimizzazione.

Non è stata trattata in maniera approfondita la stabilità numerica dei problemi, che incide in maniera significativa sul training di una rete. Le accortezze attuate in termini di ridimensionalizzazione dei problemi hanno consentito di ottenere risultati sicuramente più accurati, tanto che per le equazioni adimensionalizzate dei casi 1D i risultati sono quasi indistinguibili dalle soluzioni esatte. Si può pensare di estendere l'adimensionalizzazione delle equazioni anche ai casi tridimensionali, ad esempio emulando il lavoro fatto sulle equazioni della fluidodinamica di Navier Stokes tramite l'introduzione di opportuni fattori adimensionali [2]. Può essere un'idea vincente la creazione di un analogo sistema adimensionale capace di descrivere il problema.

Il campo del Scientific Machine Learning è vasto e ancora molto inesplorato, pieno

di potenzialità che si stanno scoprendo solo ora. Con questo studio si è cercato di far si che le reti neurali si modellassero a tal punto di replicare la funzione matematica incognita di un problema noto, allenandole per ogni caso singolare. Questa tesi si è limitata ad applicare le reti neurali su pochi casi di diverso genere, ma un possibile sviluppo può essere quello di generalizzare la rete, fornendo come input anche una serie di parametri del problema. In questo modo sarà possibile creare uno strumento più generale, che simuli in maniera più completa il funzionamento di un software standard utilizzato per le simulazioni FEM.

Appendice A

Analisi delle deformazioni

Nello studio delle deformazioni è possibile partire da un concetto più generale, utilizzando una funzione chiamata *funzione di trasposizione*, con proprietà simili a quelle della funzione spostamento, ma pensata in un'ottica più generale. Si consideri un corpo deformabile arbitrario in una configurazione iniziale (*c.i.*) che subisce uno spostamento generico verso una configurazione deformata (*c.d.*). L'equazione che lega le posizioni del corpo in *c.i.* e in *c.d.* è definita come

$$\mathbf{x} = \boldsymbol{\chi}(\mathbf{X}), \qquad \mathbf{u}(\mathbf{X}) = \mathbf{x} - \mathbf{X} = \boldsymbol{\chi}(\mathbf{X}) - \mathbf{X} \tag{A.1}$$

dove $\mathbf{X}$ rappresenta la *c.i.*, $\mathbf{x}$ rappresenta la *c.d.*, $\mathbf{u}$ rappresenta la funzione spostamento e $\boldsymbol{\chi}$ rappresenta la funzione di trasposizione [45].

Data la relazione tra $\boldsymbol{\chi}$ e $\mathbf{u}$ descritta tramite l'Eq. A.1, le restrizioni applicate alla funzione di trasposizione valgono anche per la funzione di spostamento. Di conseguenza, l'analisi della deformazione di un corpo può essere effettuata riferendosi ad ambedue le funzioni, ottenendo in ogni caso il medesimo risultato.

É di fondamentale importanza lo studio del gradiente della funzione di trasposizione, in quanto ci consente di studiare gli spostamenti che avvengono nell'intorno di un generico punto materiale appartenente al corpo. Il gradiente della funzione di trasposizione viene rappresentato dal tensore $\mathbf{F}$

$$\mathbf{F} = \boldsymbol{\nabla}\boldsymbol{\chi}(\mathbf{X}) \tag{A.2}$$

oppure, tramite l'Eq. A.1,

$$\mathbf{F} = \mathbf{I} + \boldsymbol{\nabla}\mathbf{u}(\mathbf{X}). \tag{A.3}$$

dove $\mathbf{I}$ rappresenta la matrice identità. Il gradiente, per definizione, permette di rappresentare la trasformazione subita da un segmento orientato in *c.i.* $\mathbf{dX}$ alla *c.d.* $\mathbf{dx} = \boldsymbol{\chi}(\mathbf{X} + \mathbf{dX}) - \boldsymbol{\chi}(\mathbf{X})$ tramite

$$\mathbf{dx} = \mathbf{F}\,\mathbf{dX} \tag{A.4}$$

In uno spazio 3D, definite le quantità

$$\mathbf{x} = \left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\}, \quad \mathbf{X} = \left\{ \begin{array}{c} X \\ Y \\ Z \end{array} \right\}, \quad \mathbf{u} = \left\{ \begin{array}{c} u \\ v \\ w \end{array} \right\}, \tag{A.5}$$

l'Eq. A.1 diventa, in componenti scalari,

$$\begin{aligned} x(X,Y,Z) &= X + u(X,Y,Z) \\ y(X,Y,Z) &= Y + v(X,Y,Z) \\ z(X,Y,Z) &= Z + w(X,Y,Z) \end{aligned} \tag{A.6}$$

Di conseguenza, il tensore gradiente della funzione di trasposizione $\mathbf{F}$ equivale a

$$\mathbf{F} = \left[\begin{array}{ccc} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} & \frac{\partial z}{\partial Z} \end{array} \right] = \left[\begin{array}{ccc} 1 + \frac{\partial u}{\partial X} & \frac{\partial u}{\partial Y} & \frac{\partial u}{\partial Z} \\ \frac{\partial v}{\partial X} & 1 + \frac{\partial v}{\partial Y} & \frac{\partial v}{\partial Z} \\ \frac{\partial w}{\partial X} & \frac{\partial w}{\partial Y} & 1 + \frac{\partial w}{\partial Z} \end{array} \right] \tag{A.7}$$

In generale, lo spostamento di un corpo, o di punti materiali appartenenti ad esso, è composto da una componente di spostamento rigida e da una componente che ne descrive la deformazione. In particolare, il tensore $\mathbf{F}$ che descrive la trasformazione di un punto materiale e del suo intorno può essere decomposto ottenendo

$$\mathbf{F} = \mathbf{R}\mathbf{U} = \mathbf{V}\mathbf{R} \tag{A.8}$$

dove $\mathbf{R}$ è un tensore ortogonale che descrive la rotazione rigida, mentre $\mathbf{U}$ e $\mathbf{V}$ sono tensori simmetrici detti rispettivamente *tensore destro* e *tensore sinistro della deformazione*. Per studiare la deformazione pura occorre quindi separare il tensore $\mathbf{R}$ dal tensore $\mathbf{U}$, o $\mathbf{V}$.

A.1 Tensore della deformazione di Green

Il tensore della deformazione di Green viene definito come segue

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right) \tag{A.9}$$

dove $\mathbf{I}$ rappresenta la matrice identità [33]. La quantità $\mathbf{F}^T\mathbf{F}$ elimina la componente di rotazione rigida, in quanto

$$\mathbf{F}^T\,\mathbf{F} = (\mathbf{R}\,\mathbf{U})^T(\mathbf{R}\,\mathbf{U}) = \mathbf{U}^T\,\mathbf{R}^T\,\mathbf{R}\,\mathbf{U} = \mathbf{U}^T\,\mathbf{U} \tag{A.10}$$

data l'ortogonalità di $\mathbf{R}$. Proprio per questo, infatti, il tensore $\mathbf{E}$ viene anche chiamato *Tensore destro della deformazione di Green*. Considerando l'Eq. A.3, possiamo riscrivere il tensore di Green come

$$\mathbf{E} = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u}^T + \boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^T\boldsymbol{\nabla}\mathbf{u}\right). \tag{A.11}$$

Esplicitando le singole componenti risulta

$$E_{xx} = \frac{\partial u}{\partial X} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial X}\right)^2 + \left(\frac{\partial v}{\partial X}\right)^2 + \left(\frac{\partial w}{\partial X}\right)^2\right] \tag{A.12}$$

$$E_{yy} = \frac{\partial v}{\partial Y} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial Y}\right)^2 + \left(\frac{\partial v}{\partial Y}\right)^2 + \left(\frac{\partial w}{\partial Y}\right)^2\right] \tag{A.13}$$

$$E_{zz} = \frac{\partial w}{\partial Z} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial Z}\right)^2 + \left(\frac{\partial v}{\partial Z}\right)^2 + \left(\frac{\partial w}{\partial Z}\right)^2\right] \tag{A.14}$$

$$E_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial X}\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X}\frac{\partial v}{\partial Y} + \frac{\partial w}{\partial X}\frac{\partial w}{\partial Y}\right] \tag{A.15}$$

$$E_{xz} = \frac{1}{2}\left(\frac{\partial u}{\partial Z} + \frac{\partial w}{\partial X}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial X}\frac{\partial u}{\partial Z} + \frac{\partial v}{\partial X}\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial X}\frac{\partial w}{\partial Z}\right] \tag{A.16}$$

$$E_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y}\right) + \frac{1}{2}\left[\frac{\partial u}{\partial Y}\frac{\partial u}{\partial Z} + \frac{\partial v}{\partial Y}\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y}\frac{\partial w}{\partial Z}\right] \tag{A.17}$$

e, considerando che il tensore delle deformazioni è simmetrico, possiamo quindi comporlo come segue

$$\mathbf{E} = \begin{bmatrix} E_{xx} & E_{xy} & E_{xz} \\ E_{xy} & E_{yy} & E_{yz} \\ E_{xz} & E_{yz} & E_{zz} \end{bmatrix} \tag{A.18}$$

A.2 Tensore della deformazione infinitesima

Partendo dalla definizione del tensore delle deformazioni di Green viene introdotta l'ipotesi di piccoli spostamenti, ovvero si ipotizza che le posizioni iniziali e finali siano praticamente coincidenti. Si parla di teoria lineare della deformazione lo studio che considera il campo di spostamenti $\mathbf{u}(\mathbf{X})$ e il suo gradiente $\boldsymbol{\nabla}\mathbf{u}$ come degli *infinitesimi*. Si parla di piccoli spostamenti e di piccole deformazioni quando

$$\frac{||\mathbf{u}(\mathbf{X})||}{L} \ll 1, \quad ||\boldsymbol{\nabla}\mathbf{u}(\mathbf{X})|| \ll 1 \tag{A.19}$$

dove L rappresenta una dimensione rappresentativa del problema. É possibile ricavare la formulazione del tensore delle deformazioni infinitesime trascurando gli infinitesimi di ordine superiore al primo partendo dalla definizione del tensore di Green. Utilizzando le Eq. A.12 otteniamo quindi

$$\varepsilon_{xx} = \frac{\partial u}{\partial x}, \quad \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{A.20}$$

$$\varepsilon_{yy} = \frac{\partial v}{\partial y}, \quad \varepsilon_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{A.21}$$

$$\varepsilon_{zz} = \frac{\partial w}{\partial z}, \quad \varepsilon_{zx} = \frac{1}{2}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) \tag{A.22}$$

e, considerando che il tensore delle deformazioni è simmetrico, possiamo quindi comporlo come segue

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{zz} \end{bmatrix} \tag{A.23}$$

Bibliografia

[1] Abdulrahman Tosho Abdulahi, Roseline Oluwaseun Ogundokun, Ajiboye Raimot Adenike, Mohd Asif Shah, and Yusuf Kola Ahmed. PulmoNet: a novel deep learning based pulmonary diseases detection model. *BMC Medical Imaging*, 24(1):51, February 2024.

[2] Jr. Anderson, John D. *Fundamentals of Aerodynamics*. McGraw Hill, 2011.

[3] T. Ando. Majorization relations for hadamard products. *Linear Algebra and its Applications*, 223-224:57–64, 1995. Honoring Miroslav Fiedler and Vlastimil Ptak.

[4] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, Karen Willcox, and Steven Lee. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. 2 2019.

[5] Suzanna Becker and Yann Lecun. Improving the convergence of back-propagation learning with second-order methods. 01 1989.

[6] Tarsicio Beléndez, Cristian Neipp, and Augusto Beléndez. Large and small deflections of a cantilever beam. *European Journal of Physics*, 23(3):371, may 2002.

[7] Javier Bonet and Richard D. Wood. *INTRODUCTION*, page 1–21. Cambridge University Press, 2008.

[8] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1899–1903. IEEE, may 2017.

[9] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[10] Zichao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12(1):6136, oct 2021.

[11] PyTorch Contributors. Adam. `https://pytorch.org/docs/stable/generated/torch.optim.Adam.html`, 2023. [Online].

[12] PyTorch Contributors. Autograd mechanics. `https://pytorch.org/docs/stable/notes/autograd.html`, 2023. [Online].

[13] PyTorch Contributors. Torch.optim. `https://pytorch.org/docs/stable/optim.html`, 2023. [Online].

[14] PyTorch Contributors. Adadelta. `https://pytorch.org/docs/stable/generated/torch.optim.Adadelta.html`, 2024. [Online].

[15] PyTorch Contributors. Rmsprop. `https://pytorch.org/docs/stable/generated/torch.optim.RMSprop.html`, 2024. [Online].

[16] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[17] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, nov 2018. Special issue on deep reinforcement learning.

[18] Noah Goodman. Uber ai labs open sources pyro, a deep probabilistic programming language. `https://www.tiobe.com/tiobe-index/`, nov 2017. [Online].

[19] Bidhayak Goswami and Anindya Chatterjee. Balancing a stick with eyes shut: Inverted pendulum on a cart without angle measurement. *arXiv e-prints*, jan 2023.

[20] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

[21] Christian Iandiorio and Pietro Salvini. An analytical solution for large displacements of end-loaded beams. In Magd Abdel Wahab, editor, *Proceedings of the 1st International Conference on Numerical Modelling in Engineering*, pages 320–338, Singapore, 2019. Springer Singapore.

[22] The Python Package Index. `https://pypi.org/`, 2024. [Online].

[23] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, feb 2021.

[24] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, aug 2021.

[25] Andrej Karpathy. Pytorch at tesla - andrej karpathy, tesla. `https://www.youtube.com/watch?v=oBkllt KXtDE`, 2019. [Online].

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[27] Vladimir Krasnopolsky and Aleksei A Belochitski. Using machine learning for model physics: An overview. *arXiv preprint arXiv:2002.00416*, 2022.

[28] D. Kuhlman. *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. Platypus Global Media, 2011.

[29] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[30] Alexander Lavin, David Krakauer, Hector Zenil, Justin Gottschlich, Tim Mattson, Johann Brehmer, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, Carina Prunkl, Brooks Paige, Olexandr Isayev, Erik Peterson, Peter L. McMahon, Jakob Macke, Kyle Cranmer, Jiaxin Zhang, Haruko Wainwright, Adi Hanuka, Manuela Veloso, Samuel Assefa, Stephan Zheng, and Avi Pfeffer. Simulation intelligence: Towards a new generation of scientific methods. *arXiv e-prints*, dec 2021.

[31] S. Lenci. *Lezioni di meccanica strutturale*. Pitagora, 2009.

[32] Zichao Long, Yiping Lu, and Bin Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, dec 2019.

[33] Jacob Lubliner. *Plasticity theory*. Courier Corporation, 2008.

[34] B Moseley. *Physics-informed machine learning: from concepts to real-world applications*. PhD thesis, University of Oxford, 2022.

[35] Mohammad Motamed. Approximation power of deep neural networks: an explanatory mathematical survey, 2022.

[36] Goran Nakerst, John Brennan, and Masudul Haque. Gradient descent with momentum—to accelerate or to super-accelerate? *arXiv preprint arXiv:2001.06472*, 2020.

[37] TIOBE organization. Tiobe index for march 2024. `https://www.tiobe.com/tiobe-index/`, 2024. [Online].

[38] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[39] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, nov 2017.

[40] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

[41] Stephan Rasp, Michael S. Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39):9684–9689, 2018.

[42] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986.

[43] V.S. Ryaben'kii and S.V. Tsynkov. *A Theoretical Introduction to Numerical Analysis*. Taylor & Francis, 2006.

[44] Tensorflow. Adadelta. `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/experimental/Adadelta`, 2024. [Online].

[45] C. Truesdell. *A First Course in Rational Continuum Mechanics. Vol. 1: General Concepts*, volume 71 of *Pure and Applied Mathematics*. Academic Press, New York, first edition, 1977.

[46] Kiwon Um, Robert Brand, Yun, Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, 2021.

[47] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.