POLITECNICO DI TORINO

Collegio di Ingegneria Chimica e dei Materiali

Corso di Laurea Magistrale
in Ingegneria Chimica e dei Processi Sostenibili

Accoppiamento di CFD e Machine Learning: due casi studio nell'ingegneria di processo

Relatori
Prof. Daniele Marchisio
Prof. Gianluca Boccardo

Candidato
Agnese Marcato

Anno Accademico 2018/2019

Indice

Elenco delle figure

Elenco delle tabelle

1 | Introduzione

Il termine *Machine Learning* (ML) è stato coniato nel 1959 da Arthur Samuel che ne diede la seguente definizione:
[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. (Arthur Samuel, 1959).
Una definizione più ingegneristica che permette di comprendere le azioni base di un algoritmo di Machine Larning è stata proposta, invece, dal Prof. Tom Mitchell nel 1997:
A computer program is said to learn from experience E with respect to some task T and some performance measue P, if its performance on T, as measured by P, improves with experience E. (Tom Mitchell, 1997) [1].
La particolarità di queste tecniche risiede nella capacità di imparare oltre che applicare decisioni pre-programmate. Uno dei primi obiettivi dei ricercatori in questo ambito è stato mimare alcuni processi di apprendimento della mente umana automatizzando delle routine spontanee ed immediate per degli esseri umani; infatti, il Machine Learning è strettamente legato al concetto di Intelligenza Artificiale (*Artificial Intelligence*, AI). Uno degli esempi più famosi è il riconoscimento delle cifre scritte a mano: è un compito semplice per una persona ma non immediato per un algoritmo che deve essere in grado di prescindere dalle minime variazioni con cui possono essere scritti dei numeri. Altri esempi celebri riguardano delle vere e proprie sfide tra umani e computer in giochi da tavolo. AlphaGo, software per il gioco del go sviluppato da Google DeepMind, fu il primo computer in grado di battere nel 2016 alcuni dei giocatori più forti del mondo. Il programma, basato su reti neurali profonde, è stato allenato dapprima attraverso un database di partite reali per poi passare a giocare contro se stesso per migliorare le proprie prestazioni. Altre applicazioni più quotidiane delle tecniche di Machine Learning sono gli algoritmi di profilazione (suggerimenti personalizzati in piattaforme come Youtube, Netflix), gli algoritmi di tagging (riconoscimento automatico di oggetti e persone in foto), gli algoritmi per la predizione di parametri finanziari (previsione dell'affidabilità creditizia) e medici (aiuto nella diagnosi per immagini). Innumerevoli sono le possibili applicazioni future come le automobili a guida autonoma [2].

L'impatto dell'Intelligenza Artificiale è considerato una rivoluzione all'altezza della rivoluzione industriale e digitale. Se la rivoluzione industriale ha sfruttato l'introduzione delle macchine per potenziare e sostituire il lavoro manuale e la rivoluzione digitale ha tratto profitto dalla potenza computazionale dei computer per sostituire e amplificare i lavori intellettuali ripetitivi eseguiti dalle persone, l'Intelligenza Artificiale potrebbe sostituire e potenziare molti compiti intellettuali non ripetitivi attualmente svolti da degli umani diventando in effetti, per la prima volta, un concorrente ad essi [3].

Per prevedere realisticamente l'impatto degli imminenti sviluppi nel campo dell'Intelligenza Artificiale è necessario prendere coscienza della forte non linearità dell'avanzamento delle nuove tecnologie. Il lasso di tempo che scorre tra l'invenzione di una nuova tecnologia e la sua applicazione pratica risulta essere sempre più breve: sono trascorsi, infatti, 200 anni tra lo sviluppo del primo motore realizzabile (Newcomen, 1707) e la costruzione del primo autoveicolo commerciale (Ford, 1908); sono passati 90 anni dall'introduzione dell'elettricità al suo sfruttamento intensivo nelle industrie; sono, invece, stati sufficienti 20 anni per la commercializzazione del primo computer (IBM 360) a partire da ENIAC, il primo in assoluto; sono trascorsi 10 anni dalla prima chiamata da telefono mobile (Dr. Martin Cooper, 1973) e il lancio della Motorola; gli smartphones, introdotti nel 2002, sono soggetti a crescita tecnologica continua ed includono strumenti che sfruttano l'Intelligenza Artificiale. Non ci sono dubbi sul fatto che lo sviluppo di tecnologie AI accoppiate con la crescita esponenziale di Internet influenzerà nel breve termine come le aziende operano, come vendono i propri prodotti e come sono gestite [3].

Le tecniche di Machine Learning ed Intelligenza Artificiale hanno la capacità di influenzare direttamente sia la produzione che le caratteristiche di un ampio spettro di prodotti e servizi con importanti implicazioni per la produttività, l'impiego e la concorrenza. In parallelo però esse hanno il potenziale di cambiare il processo stesso di innovazione con conseguenze che possono essere ugualmente profonde e, nel tempo, potrebbero dominare gli effetti diretti. "Inventare un metodo di invenzione", infatti, ha un impatto economico ancora più grande dello sviluppo di un singolo nuovo prodotto [4].

Se la robotica nell'ambito dell'AI è un settore trainante ma trova applicazione in contesti finalizzati a uno scopo ben preciso, solitamente industriale, le reti neurali sono uno strumento che rientra nelle tecnologie *general-purpose* (GPT) in quanto sono applicabili in diversi settori, sono capaci di introdurre innovazione in essi e hanno la possibilità di evolversi nel tempo. Le reti neurali, cuore del Deep Learning, sono un programma che utilizza una combinazione di pesi per tradurre un set di dati in ingresso in un set di dati in uscita, misura la qualità della predizione ed infine rivaluta i pesi per minimizzare la distanza dei dati in uscita dalla realtà. Le reti neurali sono uno strumento promettente per la predizione non strutturata di eventi fisici e logici in cui gli algoritmi tradizionali basati su un insieme finito di istruzioni operano con difficoltà. Lo sviluppo di questo nuovo approccio di predizione potrebbe essere in grado di rinnovare la ricerca tecnica e scientifica: piuttosto che focalizzarsi su piccoli dataset ottimamente caratterizzati, è possibile ora procedere identificando grandi insiemi di dati non strutturati che possono essere utilizzati per sviluppare dinamicamente predizioni accurate sui fenomeni in studio [4].

I primi tentativi di impiego di questo approccio innovativo alla ricerca risalgono al decennio corrente e coprono molti campi scientifici e tecnici. Ad esempio sono state utilizzate delle reti neurali convolutive (CNN) per lo sviluppo di nuovi materiali di impiego fotovoltaico. Attraverso un database contenente prove sperimentali di diffrazioni a raggi X e dati cristallografici è possibile prevedere quali nuovi materiali avranno potenzialmente le caratteristiche più opportune per tale impiego diminuendo il numero di

procedure sperimentali necessarie, quindi velocizzando il processo di ricerca [5]. Le CNN sono sfruttate anche da Atomwise, startup fondata nel 2012 in California, per l'identificazione della bioattvità di piccole molecole nell'ambito della scoperta di nuovi farmaci, con gli stessi vantaggi in termini di tempo e sperimentazione del lavoro precedente [6]. I ricercatori che si occupano di scienze della Terra, ed in particolare di meteorologia, auspicano lo sfruttamento di tecniche di Machine Learning in parallelo ai modelli fisici tradizionali per poter gestire ed estrarre informazioni dall'enormità di dati disponibili quotidianamente in questo settore in modo tale da poter produrre delle predizioni più accurate e a lungo termine [7].

La ricerca nel campo dell'ingegneria chimica ha iniziato a sfruttare i mezzi propri dell'Intelligenza Artificiale a partire dagli anni 80 del secolo scorso, tuttavia il loro impatto non è stato determinate a tal punto da consolidare tale sinergia. La modellazione è l'ambito dell'ingegneria chimica entro cui potrebbero inserirsi le tecniche di Machine Learning. Essa ha visto il suo sviluppo attraverso due strumenti chiave: dapprima l'introduzione di metodi matematici per la modellazione delle operazioni unitarie ed in secondo luogo l'introduzione della programmazione matematica. Lo step successivo potrebbe prevedere l'introduzione delle tecniche di AI al fine di potenziare le capacità umane specialistiche di elaborazione delle informazioni e decisionali per risolvere problemi nell'ambito della sintesi, del design, del controllo, della programmazione, dell'ottimizzazione e dell'analisi del rischio [8]. Potenzialmente l'introduzione di tecniche di Machine Learning potrebbe cambiare radicalmente l'approccio con cui è stato condotta la ricerca finora, infatti, si potrebbe passare dall'approccio *top-down* tipico della scienza del 20esimo secolo ad un approccio *bottom-up*. Il tradizionale riduzionismo consiste nell'analizzare un caso studio dal livello macroscopico a quello microscopico, cioè dal tutto alle sue parti. La fisica e la chimica hanno tratto importanti progressi da questo approccio, come lo sviluppo della meccanica statistica, la teoria generale della relatività e la teoria delle stringhe. Sfruttando le tecniche di ML nel 21esimo secolo si potrebbe condurre un processo opposto, detto costruzionista, ossia sfruttare i dati a disposizione per costruire modelli per la risoluzione di nuovi e più complessi problemi tecnologici. Se al momento gli strumenti più potenti del Machine Learning, le reti neurali, possono essere sfruttati come scatole nere (*black box*) ciò che potrebbe rivoluzionare profondamente l'ingegneria chimica sarebbe l'accoppiamento di modelli costruiti a partire da dati con i principi fondamentali della fisica e della chimica, in quanto sarebbe possibile coniugare la ricerca teorica con delle predizioni accurate [8].

Affinché si possano applicare le tecniche di ML è necessario possedere un'enorme quantità di dati. L'ingegneria chimica non è di per se un classico settore di *big data* come lo sono la finanza o il riconoscimento vocale e visivo, tuttavia negli ambiti che prevedono l'utilizzo delle simulazioni al computer è possibile generare una voluminosa quantità di dati [8]. La fluidodinamica computazionale (CFD) è un ramo della meccanica dei fluidi che utilizza delle tecniche di risoluzione numerica per analizzare e risolvere problemi che coinvolgono il flusso di fluidi, quindi dato un sistema proprio dell'ingegneria chimica è possibile eseguire delle simulazioni al computer per ottenere il campo di moto, di pressione, i profili di concentrazione o qualsiasi altra variabile oggetto di studio.

La fluidodinamica computazionale può essere utilizzata per ampliare il dataset neces-

sario partendo dai dati ottenuti attraverso attività sperimentali, che per motivi pratici richiederebbero un tempo tale da rendere laboriosa e costosa la creazione di una quantità sufficiente di risultati. Questa attività è definita *simulation-on-the-loop*: essa prevede un'attività sperimentale iniziale, la messa a punto di simulazioni CFD che modellino con accuratezza la realtà e la risoluzione di altre simulazioni che esplorino uno spettro di casistiche nuove per ampliare il dataset iniziale. I modelli *data-driven*, ottenuti con questo processo, sono in grado di fornire delle predizioni pressoché istantanee, quindi nel campo dell'ingegneria chimica potrebbero trovare applicazione nella modellazione multiscala per tenere conto di un elevato numero di sotto-modelli senza appesantire computazionalmente le simulazioni, oppure nel controllo industriale in linea.

Il presente lavoro ha l'obiettivo di sviluppare un *workflow* per la creazione del set di dati necessari alla creazione di modelli ML e valutare la capacità predittiva di tali modelli, anch'essi costruiti in questo lavoro, rispetto ai risultati ottenuti dalle simulazioni CFD. Tale metodologia è stata testata su due casi studio: la produzione di emulsioni alimentari e il flusso e trasporto attraverso mezzi porosi. Il primo caso studio riguarda la produzione della maionese e di un'ampia gamma di condimenti, essi consistono nell'emulsione di una fase organica in acqua opportunamente stabilizzata. Il processo di emulsificazione prevede la miscelazione di tutti gli ingredienti, in seguito alla quale il fluido viene fatto scorrere attraverso un'apparecchiatura rotante in grado di imprimere elevati sforzi di taglio, in questo modo la distribuzione dimensionale delle gocce di fase organica, che influenza il sapore, il colore e la struttura, raggiunge le specifiche di prodotto. Il secondo caso studio riguarda il trasporto e la deposizione di particelle colloidali in mezzi porosi. La comprensione di questi fenomeni è basilare per lo studio di molti problemi ambientali ed ingegneristici, come ad esempio la filtrazione, i processi catalitici, la separazione cromatografica e la bonifica delle falde acquifere per mezzo di iniezione di colloidi. La creazione di modelli data-driven per i casi studio presentati potrebbe trovare diretta applicazione industriale per il primo mentre per il secondo i modelli potrebbero inserirsi in un contesto più ampio di modellazione multiscala.

Il lavoro svolto in questa tesi è strutturato come segue:

- esposizione dei concetti teorici fondamentali per la comprensione dei modelli a principi primi e delle principali tecniche di ML, con particolare attenzione alle reti neurali (Capitolo 2);
- descrizione dei metodi utilizzati per la risoluzione numerica delle simulazioni CFD, per la redazione dei programmi finalizzati alla creazione del dataset, per la realizzazione del modelli ML (Capitolo 3);
- presentazione dei dettagli riguardanti le simulazioni dei casi studio (Capitolo 4);
- analisi e presentazione dei risultati (Capitolo 5);
- conclusioni (Capitolo 6).

2 | Elementi teorici e aspetti fondamentali

In questo capitolo sono esposte le nozioni teoriche che sono alla base di questo lavoro. In primo luogo, vengono presentate le equazioni la cui risoluzione permette di descrivere fisicamente i sistemi considerati (sezione 2.1). In secondo luogo, è proposta una panoramica sugli algoritmi principali di machine learning (sezione 2.2), con particolare enfasi sulle reti neurali e su alcune tecniche di regressione lineare che sono state utilizzate per costruire i modelli predittivi.

2.1 Physics-based models

I modelli detti *physics-based* sono ottenuti dalla risoluzione di equazioni a loro volta derivate da conoscenze fisiche, per questo sono anche detti modelli a principi primi [9]. Nel caso in esame la fisica del sistema è descritta da equazioni differenziali alle derivate parziali la cui soluzione viene facilitata dall'assunzione di ipotesi semplificative.

2.1.1 Ipotesi

Di seguito sono presentate le ipotesi che sono state adottate per la risoluzione delle simulazioni del *cone mill* per la produzione di emulsioni alimentari:

- il fluido è considerato incomprimibile e con densità costante;
- il fluido è considerato continuo, infatti, lo spazio in cui esso scorre è molto più grande del libero cammino medio delle molecole;
- la dispersione liquido-liquido pur essendo un sistema multi-fase è considerato come un fluido monofase (*pseudo-single-phase*);
- il comportamento del fluido è assunto non-Newtoniano e per la determinazione della viscosità è stata implementata una legge reologica basata su dati sperimentali [10].

Per quanto riguarda le simulazioni del flusso attraverso mezzi porosi sono state effettuate le seguenti ipotesi semplificative [11]:

- il fluido è considerato incomprimibile e con densità costante;

- il mezzo poroso è saturo, quindi è presente una singola fase fluida;

- il fluido è considerato un continuo, infatti, sebbene le lunghezze caratteristiche delle geometrie (i.e.: la dimensione dei pori) prese in esame siano dell'ordine delle centinaia di micrometri, esse sono sufficientemente maggiori del libero cammino medio delle molecole;

- l'approccio del continuo è stato adottato anche per le particelle di colloide poiché la loro dimensione è inferiore alla dimensione dei grani di almeno quattro ordini di grandezza, ne è stata quindi espressa la loro concentrazione;

- si assume che la concentrazione delle particelle sia sufficientemente bassa da considerare il sistema diluito, così che il flusso diffusivo possa essere descritto attraverso la legge di Fick:

$$J_i = -\mathcal{D}\frac{\partial C}{\partial x_i}, \tag{2.1}$$

  in cui J_i è il flusso diffusivo (mol m^{-3} s^{-1}), C è la concentrazione del colloide (mol m^{-3}), $\mathcal{D}$ è il coefficiente di diffusione (m^2 s^{-1});

- il fluido è Newtoniano, quindi la sua viscosità è costante al variare dello *strain rate*, tale ipotesi è valida se si considera il fluido sufficientemente diluito;

- si considera che le particelle seguano il flusso del fluido, questa ipotesi è verificata per particelle solide (caratterizzate da densità compresa tra 5000 e 10000 kg m^{-3} e dimensione inferiore a 1 μm) disperse in acqua a temperatura ambiente. Più propriamente a queste condizioni corrisponde un numero di Stokes, definito come il rapporto tra il tempo caratteristico di rilassamento delle particelle e del fluido, molto inferiore all'unità.

- si considera che le particelle si muovano con una velocità uguale a quella del fluido e che il coefficiente di diffusione sia indipendente dalla distanza tra la particella e il grano. Le interazioni dovute alle forze di Van der Waals, al doppio strato elettronico e di tipo idrodinamico non sono considerate nella modellazione, infatti, si assume che l'effetto di ritardo idrodinamico, risultante in un incremento delle forze viscose cui è soggetta la particella in prossimità dei grani, sia controbilanciato dalle forze attrattive di London (approssimazione di Smoluchowski-Levich).

2.1.2 Equazioni di governo

Le equazioni, riportate in forma indiciale o di Einstein, che devono essere risolte per descrivere il flusso nel cone mill e nei mezzi porosi sono:

- equazione di continuità, riportata per un fluido incomprimibile:

$$\frac{\partial U_i}{\partial x_i} = 0, \tag{2.2}$$

dove U_i è la componente i-esima della velocità del fluido (m s^{-1});

- equazione di Navier Stokes, valida per un fluido incomprimibile e newtoniano:

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j^2}, \tag{2.3}$$

dove ρ è la densità del fluido (kg m^{-3}), ν è la viscosità cinematica del fluido (m^2 s^{-1});

Per il caso del flusso attraverso mezzi porosi, in cui si descrive la deposizione di un colloide disperso nel fluido, è necessario risolvere l'equazione di advezione-diffusione:

$$\frac{\partial C}{\partial t} + U_j \frac{\partial C}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\mathcal{D}\frac{\partial C}{\partial x_j}\right). \tag{2.4}$$

Il coefficiente di diffusione dovuto al moto Browniano è valutato attraverso l'equazione di Einstein:

$$\mathcal{D} = \frac{k_B T}{3\pi\mu d_P}, \tag{2.5}$$

dove k_B è la costante di Boltzmann (1.38×10^{-23} (J K^{-1})), T è la temperatura del fluido (K), μ è la viscosità dinamica del fluido (kg m^{-1} s^{-1}), d_P è la dimensione caratteristica del colloide (m).

Al fine di fornire come obiettivo agli algoritmi di machine learning una grandezza che fosse rappresentativa dei risultati ottenuti dalle simulazioni sono state scelte per il flusso attraverso mezzi porosi:

- la permeabilità, in quanto grandezza rappresentativa della soluzione del campo di moto, valutata attraverso la legge di Darcy:

$$\frac{\Delta P}{L} = \frac{\mu}{k} q, \tag{2.6}$$

dove ΔP sono le perdite di carico attraverso il letto (Pa), L è la lunghezza del letto nella direzione principale del flusso (m), k è la permeabilità (m^2), q è la velocità superficiale del fluido (m s^{-1});

- l'efficienza di deposizione, in quanto grandezza rappresentativa della soluzione dell'equazione di advezione e diffusione, valutata attraverso la seguente definizione [11]:

$$\eta = \frac{\ln\left(\frac{C}{C_0}\right)}{-\frac{3}{2}\frac{1-\varepsilon}{\varepsilon}\frac{L}{d_g}}, \tag{2.7}$$

dove C è la concentrazione del colloide nel fluido in uscita dal mezzo poroso (mol m^{-3}), C_0 è la concentrazione del colloide nel fluido in ingresso al mezzo poroso (mol m−3), ε è la porosità del letto, d_g è la dimensione dei grani del mezzo poroso (m).

Per il caso delle simulazioni del flusso attraverso il cone mill è stata scelta come feature rappresentativa lo strain rate medio volumico, in quanto esso è una grandezza che caratterizza l'apparecchiatura che ha, infatti, la funzione di emulsionare gli ingredienti. Lo strain rate è definito come la parte simmetrica del tensore gradiente di velocità:

$$\mathrm{S}_{ij} = \frac{1}{2}\left(\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}\right). \tag{2.8}$$

2.2 Data-driven models

I modelli *data-driven* sono ottenuti a partire da un set di dati e la loro formulazione non richiede una conoscenza a priori della fisica che governa il sistema o, generalmente, delle relazioni che legano le variabili in ingresso e in uscita del modello [9].

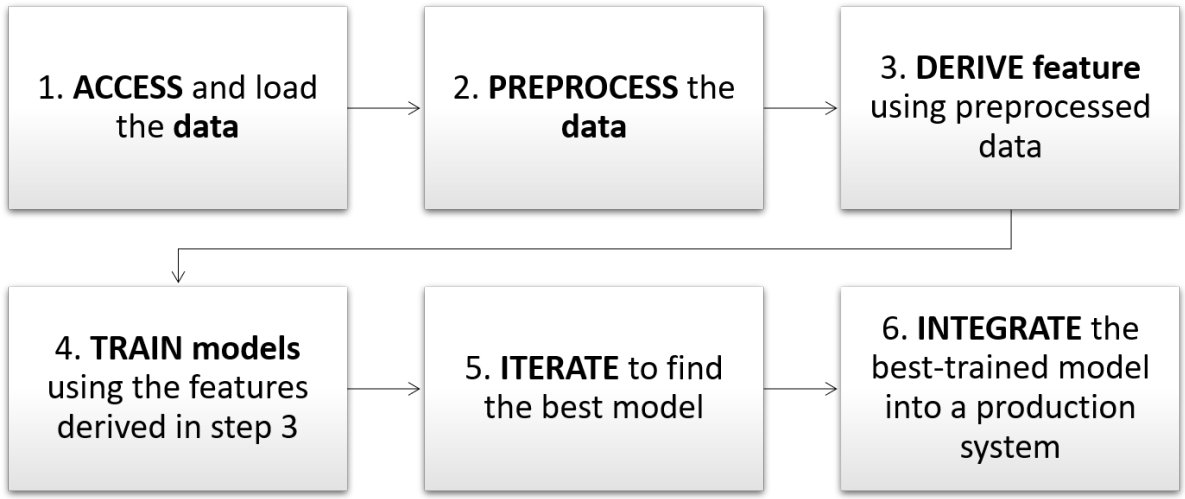


Figura 2.1: Workflow per l'applicazione di tecniche di machine learning

In Figura (2.1) è proposto un *workflow* per l'applicazione di algoritmi di machine learning. In primo luogo, è necessario possedere un dataset contenente tutte le informazioni che si vogliono utilizzare per l'ottenimento del modello surrogato; verrà chiamato *sample* un dato avente un certo numero di proprietà di input (i.e.:*feature*) ed eventualmente degli output, detti *label* o *target*. In seguito, viene eseguito un *pre-processing* dei dati, al fine di valutare la presenza di *outlier*, controllare la mancanza di alcune feature, e effettuare un eventuale *scaling* dei dati. Dopodiché è necessario scegliere quali feature utilizzare durante l'allenamento dell'algoritmo; questo passaggio è importante in quanto l'individuazione di feature adeguate, indipendenti l'una dall'altra e causali con gli output è fondamentale per ottenere un modello accurato. La fase successiva consiste nell'allenamento del modello, il cosiddetto *training*, durante il quale vengono forniti in

input all'algoritmo di machine learning i sample. Infine, è possibile migliorare le prestazioni del modello ottenuto attuando delle semplificazioni, come ad esempio valutare la possibilità di ridurre il numero di feature in input a parità di accuratezza oppure aggiungendo delle complessità, come ad esempio, suddividere il modello in più sotto modelli oppure ampliare il dataset in input [12].

È possibile suddividere gli algoritmi di machine learning in base all'entità della supervisione cui sono sottoposti durante la fase di training. Si distingue quindi tra [1]:

- *supervised learning*: ogni dato inviato all'algoritmo è provvisto di un'etichetta o di un label, che è l'obiettivo di predizione del modello. Si usano queste tecniche per eseguire compiti di classificazione (distinzione tra un numero finito di categorie) oppure di regressione (predizione di un valore numerico) [13];

- *unsupervised learning*: i dati di input non possiedono label. Si usano queste tecniche per eseguire compiti di clustering (suddivisione dei dati in sottogruppi contraddistinti da caratteristiche comuni), di visualizzazione (per avere una visione di insieme dei dati in modo da identificare pattern imprevedibili), di riduzione di dimensione (in modo da semplificare il dataset) oppure di determinazione di dati anomali. Spesso queste tecniche vengono sfruttate per eseguire il pre-processing del dataset [14];

- *semisupervised learning*: soltanto alcuni dei dati di input possiedono labels;

- *reinforcement learning*: il sistema di apprendimento, detto agente, osserva l'ambiente che lo circonda ed esegue delle azioni così da ottenere dei *reward* positivi o negativi, in modo da imparare da solo la migliore strategia, detta *policy*.

Nell'ambito del lavoro di tesi sono state utilizzate delle tecniche di tipo supervised learning; nei paragrafi successivi vengono approfondite le strategie disponibili per eseguire lo scaling delle feature, i concetti alla base delle reti neurali e alcune tecniche di regressione lineare.

2.2.1 Scaling delle feature in input

Lo step di scaling delle feature in input è fondamentale nell'ambito del pre-processing dei dati, infatti, la normalizzazione fa in modo che ogni feature contribuisca in modo rilevante alla predizione dell'algoritmo. Ad esempio, alcuni classificatori sfruttano la definizione di distanza euclidea per valutare l'appartenenza di un dato (punto) ad una categoria, quindi se una feature ha un range di variabilità più alto rispetto ad un'altra, sarà la prima a governare la distanza del punto dalle categorie, quindi a determinare la classificazione. Inoltre, è stato verificato che lo scaling delle feature velocizza la convergenza di alcuni algoritmi di machine learning, come il gradient descent [15]. Di seguito sono riassunte alcune delle principali tecniche di scaling, è chiamato x il vettore contenente i valori di una certa feature per ogni sample, mentre x' il vettore normalizzato:

- normalizzazione min-max - tutti valori del vettore vengono scalati in modo che ricadano tra un valore minimo a e un valore massimo b

$$x' = a + \frac{(x - \min(x))\ (b-a)}{\max(x) - \min(x)}, \tag{2.9}$$

dove $\min(x)$ è il valore minimo del vettore x, mentre $\max(x)$ è il valore massimo del vettore x;

- normalizzazione media - i valori vengono scalati del valore medio e normalizzati dalla differenza tra il massimo e il minimo del vettore x

$$x' = \frac{x - \bar{x}}{\max(x) - \min(x)}, \tag{2.10}$$

dove $\bar{x}$ è il valore medio del vettore x;

- standardizzazione - a differenza del metodo precedente la normalizzazione viene eseguita dividendo per la deviazione standard

$$x' = \frac{x - \bar{x}}{\sigma}. \tag{2.11}$$

La deviazione standard si ricorda essere definita come:

$$\sigma = \frac{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2}}{N}, \tag{2.12}$$

dove N è il numero di dati;

- scaling all'unità di lunghezza - tutti i valori sono scalati in modo tale che la lunghezza del vettore normalizzato x' sia pari a uno, ogni componente del vettore è quindi diviso per la lunghezza del vettore x:

$$x' = \frac{x}{\|x\|_2} = \frac{x}{\sqrt{x_1^2 + x_2^2 + \ \ldots\ + x_N^2}}. \tag{2.13}$$

2.2.2 Reti neurali

Le reti neurali (*artificial neural networks*, ANNs) consistono in una vasta categoria di algoritmi di machine learning e devono il proprio nome alla struttura del tessuto nervoso da cui si è tratta ispirazione per la loro progettazione. Tale tecnica è il cuore del deep learning e permette di affrontare obiettivi complessi in modo versatile e scalabile. È utile presentare l'analogia con i sistemi biologici al fine di comprendere il funzionamento delle reti neurali; in Figura 2.2 è schematizzato un neurone.

I neuroni sono delle cellule presenti nella corteccia cerebrale degli animali e sono costituiti da un corpo cellulare contenente il nucleo, da delle ramificazioni chiamate

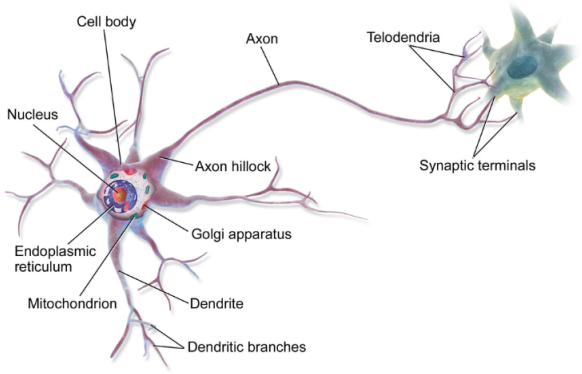


Figura 2.2: Schema di un neurone [16]

dendriti e da un prolungamento più lungo detto assone, la cui parte terminale si dirama in telodendri aventi minuscole strutture dette sinapsi che sono connesse ai dendriti di altri neuroni. I neuroni scambiano attraverso le sinapsi degli impulsi elettrici, detti segnali, quando uno di essi riceve un numero sufficiente di segnali nell'unità di tempo sarà in grado di trasmettere anch'esso un segnale elettrico ai neuroni cui è collegato. I neuroni, sebbene siano delle unità piuttosto semplici, sono in grado di eseguire operazioni complesse in quanto sono organizzati in strutture molto articolate costituite da miliardi di unità, infatti, ogni neurone è collegato a migliaia di altri [17].

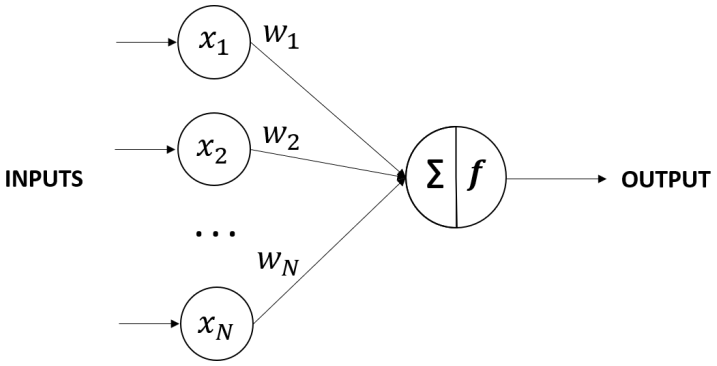


Figura 2.3: Schema di una threshold logic unit (TLU)

Se il neurone è l'elemento costitutivo del tessuto nervoso, la *Threshold Logic Unit* (TLU, Figura 2.3) è l'unità base di una rete neurale. Un neurone artificiale di questo tipo è connesso a dei valori numerici in input $(x_1, x_2, \ldots, x_N)$ attraverso dei pesi $(w_1, w_2, \ldots, w_N)$, le operazioni che esegue sono prima una somma pesata degli input ed in seguito l'applicazione di una funzione di attivazione, quindi complessivamente [1]:

$$z = f\left(w_1 x_1 + w_2 x_2 + \ldots + w_N x_N\right) = f\left(\mathbf{w}^T \mathbf{x}\right). \tag{2.14}$$

Le principali funzioni di attivazione utilizzate, rappresentate in Figura 2.4, sono:

- la funzione logistica

$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}}; \tag{2.15}$$

- la tangente iperbolica

$$\tanh(z) = 2\sigma(2z) \; - \; 1; \tag{2.16}$$

- la funzione ReLU (i.e.: *Rectified Linear Unit*)

$$ReLU(z) = \max(0, z). \tag{2.17}$$

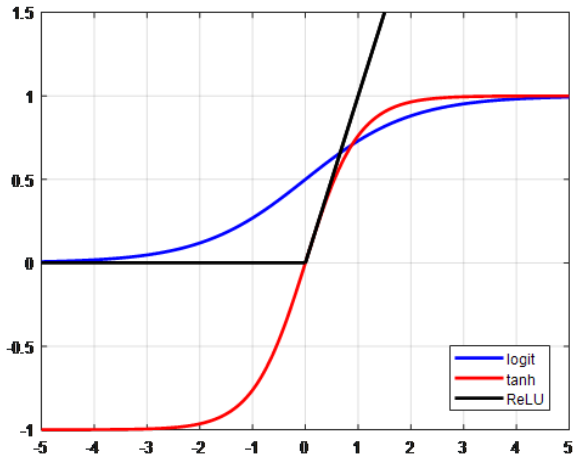


Figura 2.4: Principali funzioni di attivazione

Più TLU disposte a formare un unico layer permettono di formare una struttura detta percettrone (Figura 2.5). Ogni TLU è connessa a ciascun neurone di input, questo tipo di neuroni fornisce in output qualsiasi valore gli venga dato in ingresso. È presente, inoltre, una feature aggiuntiva ($x_0 = 1$) rappresentata attraverso un neurone *bias* che viene aggiunto in ogni layer, a differenza degli altri tipi di neurone non è fornito di alcun valore in ingresso ed esso, a livello concettuale, ha la stessa funzione dell'intercetta in una relazione lineare in quanto è indipendente dalle feature scelte per la predizione del modello [1].

I calcoli che esegue un percettrone possono essere così riassunti:

$$z_1 = f\left(w_{1,1}x_1 + \ldots + w_{1,j}x_j + \ldots + b_1\right); \tag{2.18}$$

$$z_j = f\left(w_{i,1}x_1 + \ldots + w_{i,j}x_j + \ldots + b_i\right); \tag{2.19}$$

$$z_N = f(w_{N,1}x_1 + \ldots + w_{N,j}x_j + \ldots + b_N); \tag{2.20}$$

Quindi:

$$\mathbf{z} = \; f(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{2.21}$$

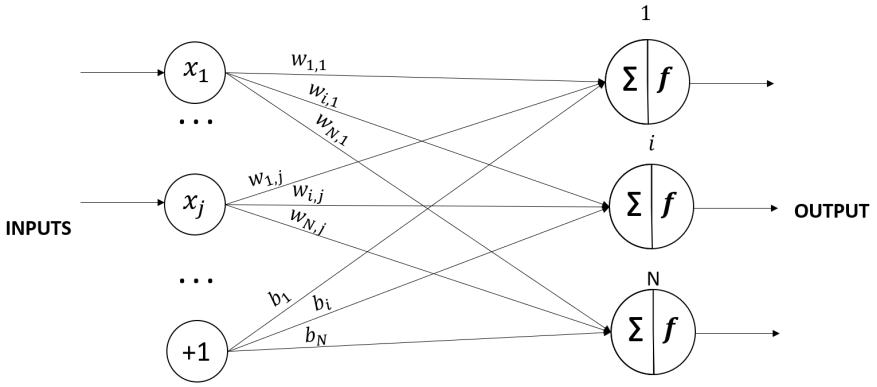


Figura 2.5: Schema di un percettrone

dove **W** è la matrice dei pesi, **x** è il vettore delle feature, **b** è il vettore dei bias, **z** è il vettore degli output. Lo sviluppo di algoritmi per il training di un percettrone affondano le proprie radici nelle teorie neuroscientifiche, in particolare nella legge di Hebb (o *Hebbian learning*) secondo la quale quando un neurone biologico interagisce spesso con un altro neurone allora la connessione tra i due diventa più forte, questa idea è stata sintetizzata nella celebre frase di Siegrid Löwel: "*Cells that fire together, wire together*", cioè le cellule che trasmettono segnali nello stesso momento sono legate tra loro. Nell'ambito delle reti neurali, due neuroni sono connessi in modo tanto più forte quanto maggiore è il valore del peso che li lega, inoltre, l'algoritmo di training tiene conto dell'errore commesso dalla rete, in modo da non rafforzare connessioni che portano a predizioni errate. In particolare, viene somministrato al percettrone un sample per volta; per ciascuno di essi viene predetto l'output; se la predizione è sbagliata, l'algoritmo aumenta i pesi degli input che avrebbero contribuito a una previsione più corretta. Questo semplice algoritmo di training può essere riassunto dall'Eq.(2.22) [1].

$$w_{i,j}^{k+1} = w_{i,j}^{k} + \lambda \left(t_i - z_i\right) x_j, \tag{2.22}$$

dove $w_{i,j}$ è il peso che lega l'i-esimo neurone di input e il j-esimo neurone di output, x_j è il j-esimo valore di input del sample corrente,z_i è l'output dell'i-esimo neurone per il sample corrente, t_i è l'output target per il sample corrente. Nell'Eq.(2.22) appare il learning rate λ, esso è un importante parametro che indica quanto velocemente la rete adatta i suoi parametri ai nuovi dati. Se viene scelto un valore troppo elevato allora il sistema si adatterà velocemente ai nuovi dati ma tenderà anche a ignorare l'effetto dei dati più vecchi; al contrario se il valore del learning rate è basso il sistema sarà soggetto ad una maggiore inerzia ma sarà meno influenzato dal rumore caratteristico dei dati o da dati non rappresentativi [1]. Al fine di riuscire a risolvere problemi sempre più complessi e non lineari sono state progettate delle reti neurali aventi più strati di neuroni (Figura 2.6, Multi-Layer Perceptron (MLP)).

Questo tipo di reti neurali è costituito da un layer di input, uno di output e diversi hidden layer, ossia layer nascosti, un'ANN costituita da due o più hidden layer è detta *deep neural network* (DNN) [1]. Si tenga conto che il numero di neuroni per layer e il numero di hidden layer può essere scelto arbitrariamente e il tipo di funzione di

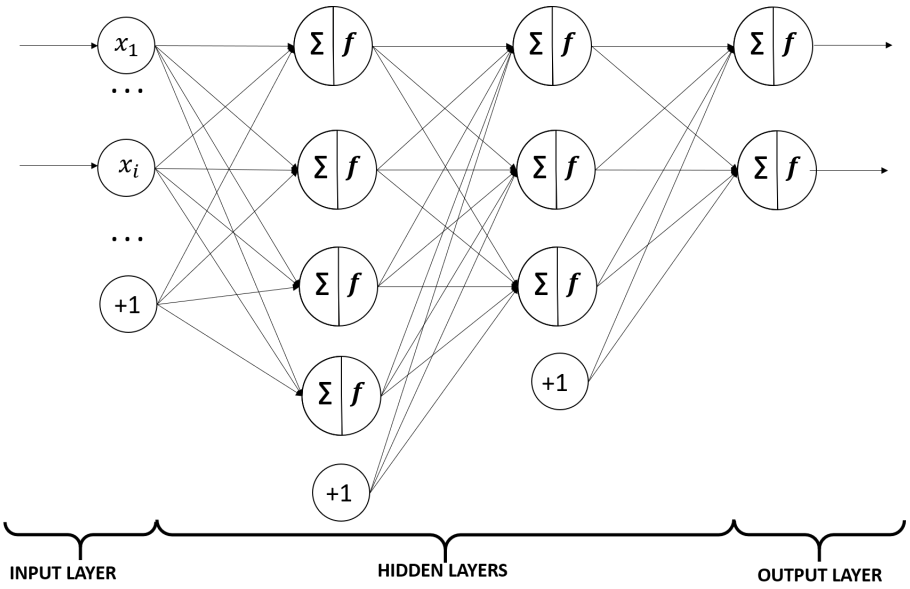


Figura 2.6: Schema di un multi-layer perceptron (MLP)

attivazione può variare tra i diversi layer. L'output layer può essere costituito da un singolo neurone se si tratta di un problema di regressione o un numero di neuroni pari al numero di categorie possibili nel caso di problemi di classificazione. Gli output di ciascun layer di neuroni sono gli input del layer successivo, quindi i calcoli che la DNN esegue, prendendo come esempio la rete schematizzata in Figura 2.6 avente tre layer, possono essere riassunti in questo modo

$$\mathbf{z}^3 = f^3\{\mathbf{W}^3 f^2\left[\mathbf{W}^2 f^1\left(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1\right) + \mathbf{b}^2\right] + \mathbf{b}^3\}, \tag{2.23}$$

dove l'apice indica il numero del layer corrispondente. Il training di una DNN è eseguito attraverso un algoritmo di retropropagazione (backpropagation algorithm). Per ogni sample del dataset, dapprima l'algoritmo ne calcola una predizione e ne valuta l'errore, in seguito valuta il contributo di ciascuna connessione neuronale all'errore ed infine aggiorna i pesi in modo tale da minimizzare l'errore. Uno degli algoritmi generici di ottimizzazione utilizzati nel campo del machine learning è il *gradient descent*, l'idea che sta alla base della tecnica è variare iterativamente i parametri al fine di minimizzare una funzione di costo. L'analogia utilizzata per spiegare questa tecnica è la seguente: si immagini di trovarsi sulla cima di una montagna e di voler raggiungere la base nel minor tempo possibile, allora sarà necessario scegliere il percorso con la pendenza maggiore; allo stesso modo l'algoritmo misura il gradiente locale della funzione errore rispetto ai pesi e ai bias e segue tale direzione fintanto che il gradiente è nullo, cioè errore minimo. Nella pratica inizialmente vengono introdotti dei valori random dei pesi e dei bias, in seguito vengono aggiornati ad ogni iterazione al fine di minimizzare la funzione di costo (Figura 2.7).

Il metodo di retropropagazione, in particolare della steepest descent backpropagation, consiste nei seguenti passaggi, si rimanda all'ALLEGATO A per la descrizione dettagliata del metodo e per la descrizione di ogni termine:

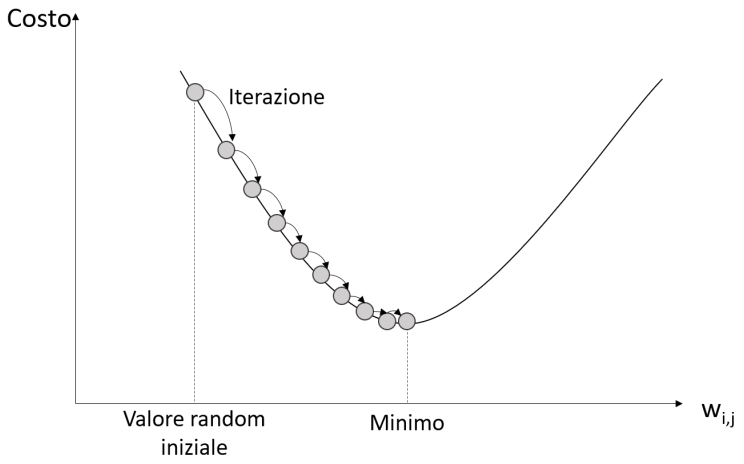


Figura 2.7: Rappresentazione concreta del gradient descent

- propagazione "in avanti" (*forward*), valutazione dell'output di ciascun layer:

$$\mathbf{z}^0 = \mathbf{x}, \tag{2.24}$$

$$\mathbf{z}^{m+1} = f^{m-1}(\mathbf{W}^{m+1}\mathbf{x^m} + \mathbf{b}^{m+1}) \text{ per m = 0, 1,..., M - 1,} \tag{2.25}$$

$$\mathbf{z} = \mathbf{x}^M, \tag{2.26}$$

  in cui M è il numero di layer della rete;

- propagazione "all'indietro" (*backward*), valutazione per ciascun layer della sensitività, una grandezza legata al gradiente della funzione errore rispetto ai pesi o ai bias. La sensitività dell'ultimo layer M è valutata nel seguente modo:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{z}), \tag{2.27}$$

  data la sensitività nel layer M è possibile calcolare le sensitività dei layer precedenti:

$$\mathbf{s}^m = -2\dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1} \text{ per m = M - 1,..., 2, 1,} \tag{2.28}$$

  dove $\dot{\mathbf{F}}$ è la matrice diagonale:

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{\mathbf{f}}^m(n_1^m) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \dot{\mathbf{f}}^m(n_{S^m}^m) \end{bmatrix}, \tag{2.29}$$

  in cui S^m è il numero di neuroni nel layer m e $\dot{\mathbf{f}}^m\left(n_j^m\right)$ è definita come

$$\dot{\mathbf{f}}^m\left(n_j^m\right) = \frac{\partial f^m\left(n_j^m\right)}{\partial n_j^m}, \tag{2.30}$$

n è un vettore il cui elemento i-esimo rappresenta l'input dovuto a tutti i neuroni e il bias del layer precedente:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m; \tag{2.31}$$

- Aggiornamento dei pesi e dei bias, attraverso la definizione del gradient descent:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \lambda \mathbf{s}^m \left(\mathbf{z}^{m-1}\right)^T, \tag{2.32}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \lambda \mathbf{s}^m \ . \tag{2.33}$$

2.2.3 Algoritmi di regressione

Altri strumenti che rientrano tra le tecniche di Machine Learning sono gli algoritmi di regressione, che si possono suddividere nelle seguenti categorie [1]:

- algoritmi di regressione lineare: i parametri del modello sono lineari rispetto alle feature in ingresso, sono semplici da interpretare e veloci nelle predizioni. Per questo motivo essi sono uno dei primi modelli da provare, tuttavia la loro accuratezza è inferiore a causa dei forti vincoli legati alla linearità;

- alberi di regressione: vengono utilizzati degli alberi decisionali per legare le feature in ingresso alle conseguenze che ne derivano. Sebbene siano uno strumento tipico della classificazione, è possibile costruire degli alberi in grado di ottenere in output una variazione continua per problemi di regressione. Questi algoritmi sono di semplice interpretazione e veloci;

- macchine a vettori di supporto: sono degli algoritmi non parametrici basati sui kernel, la cui interpretabilità dipende dalla linearità dell'algoritmo scelto;

- gaussian process regression models: sono degli algoritmi probabilistici non parametrici basati sulla definizione di kernel appropriati, sono molto accurati ma di difficile interpretazione.

3 | Dettagli numerici

In questo capitolo verranno approfonditi i dettagli numerici relativi alle simulazioni di fluidodinamica computazionale (3.1), alla produzione del dataset di input (3.2) e alla creazione dei modelli data-driven (3.3).

Le simulazioni CFD sono state effettuate con il programma *open-source* `OpenFOAM` (*Open-source Field Operation And Manipulation*) versione 6, il *toolbox* utilizza il linguaggio di programmazione `C++` e applica il metodo dei volumi finiti per risolvere le equazioni di trasporto. Da una singola simulazione risolta è possibile ottenere un sample, tuttavia per ottenere un intero dataset ne sono necessari migliaia, a tal fine sono stati prodotti degli *script* in `Python` che permettono di impostare, risolvere ed estrarre i risultati di un numero elevato di simulazioni. Una volta ottenuto il dataset sono state progettate e allenate le reti neurali usando un opportuno algoritmo disponibile nell'ambiente di calcolo `MATLAB`.

3.1 CFD - `OpenFOAM`

Nel capitolo precedente sono state presentate le equazioni che descrivono la fisica dei sistemi fluidi studiati nel lavoro di tesi, una generica equazione di trasporto può essere formulata in notazione indiciale nel seguente modo:

$$\rho\frac{\partial\phi}{\partial t} + \rho\nabla\cdot(\mathbf{U}\phi) = \nabla\cdot(\Gamma\nabla\phi) + S_\phi, \tag{3.1}$$

dove ρ è la densità, ϕ è la quantità trasportata e Γ è il coefficiente di trasporto molecolare (ad es. coefficiente di viscosità, diffusione o conducibilità).

I termini rappresentano l'accumulo della quantità ϕ nel tempo, il trasporto convettivo di ϕ dovuto al moto d'insieme del fluido, il trasporto diffusivo di ϕ e il termine sorgente. È possibile ricavare le diverse equazioni di trasporto sostituendo i rispettivi termini:

- equazione di continuità - $\phi = 1$, $\Gamma = 0$, $S_\phi = 0$;
- equazione di Navier Stokes - $\phi = U_i$, $\Gamma = \mu$, $S_\phi = -\frac{\partial p}{\partial x_i}$;
- equazione di advezione e diffusione - $\phi = \mathcal{C}$, $\Gamma = \mathcal{D}$, $S_\phi = 0$.

In genere non è possibile risolvere queste equazioni differenziali analiticamente in quanto non sono lineari e spesso contengono derivate sia spaziali che temporali, dunque è necessario sfruttare dei metodi di risoluzione numerica al fine di ottenere delle soluzioni approssimate.

3.1.1 Metodo dei volumi finiti

La tecnica più utilizzata per risolvere numericamente le equazioni del tipo della Eq.(3.1) è il metodo dei volumi finiti [18], il nome si riferisce alla necessità di suddividere l'intero dominio computazionale in sottovolumi, detti celle. Si assume che ogni cella sia come un piccolo CSTR perfettamente miscelato in cui la proprietà ϕ permane costante. L'insieme delle celle dà luogo alla griglia, o mesh, la quantità ϕ è valutata nel centroide di ogni cella. In Figura 3.1 è riportato un esempio con relativa nomenclatura di una cella tridimensionale, per geometrie 3D, e di una cella bidimensionale, per geometrie 2D. L'equazione differenziale di trasporto è riformulata per ogni cella in modo da ottenere un set di equazioni algebriche lineari, risolvibili in modo iterativo, in questo modo è garantita la conservazione delle quantità a livello locale di cella, di conseguenza la conservazione a livello globale. L'equazione di trasporto (3.1) viene dapprima integrata

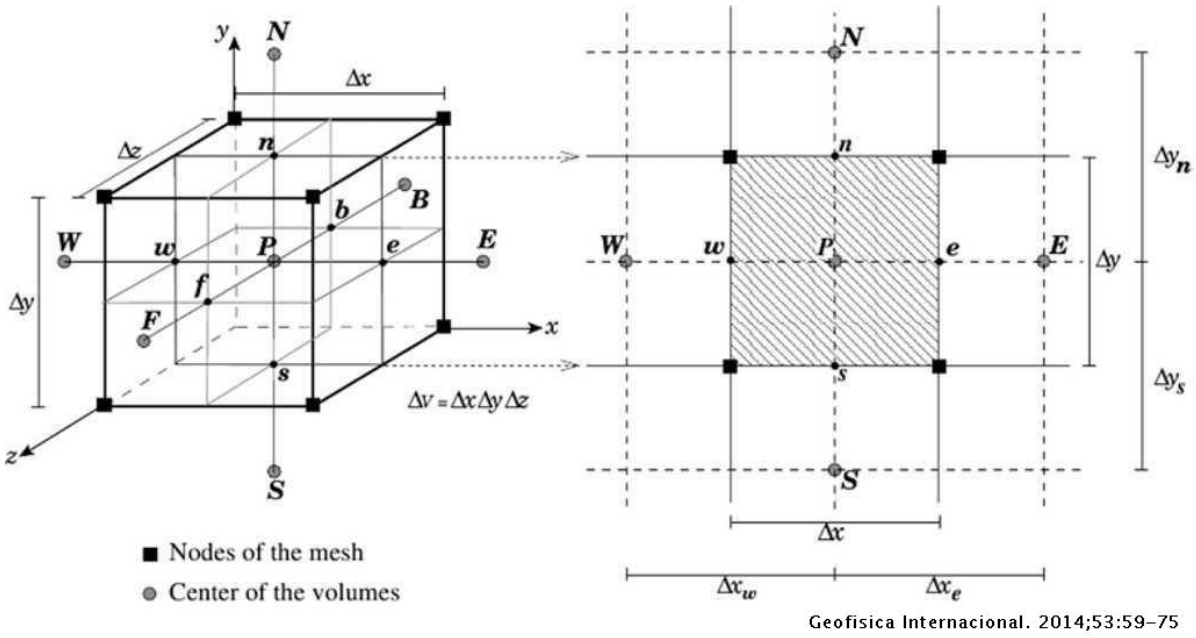


Figura 3.1: Esempi di celle tridimensionali e bidimensionali con nomenclatura [19]

su un volume di controllo Ω:

$$\int_\Omega \rho \frac{\partial \phi}{\partial t} \mathrm{d}V + \int_\Omega \rho \nabla \cdot (\mathbf{U}\phi)\, \mathrm{d}V = \int_\Omega \nabla \cdot (\Gamma \nabla \phi)\, \mathrm{d}V + \int_\Omega S_\phi \mathrm{d}V. \tag{3.2}$$

In seguito, viene applicato il teorema di Gauss al termine convettivo e al termine diffusivo:

$$\int_\Omega \rho \frac{\partial \phi}{\partial t} \mathrm{d}V + \int_S \rho \mathbf{U} \cdot \mathbf{n} \phi \mathrm{d}S = \int_\mathrm{S} \Gamma \nabla \phi \mathbf{n} \mathrm{d}S + \int_\Omega S_\phi \mathrm{d}V, \tag{3.3}$$

dove S è la superficie che racchiude il volume di controllo e $\mathbf{n}$ è il vettore normale alla superficie. Al fine di ricondurre l'Eq.(3.3) ad un'equazione algebrica è necessario approssimare gli integrali.

Per quanto riguarda il termine convettivo, il prodotto $\mathbf{U} \cdot \mathbf{n}$ rappresenta la componente della velocità perpendicolare alla superficie dS, allora per una cella tridimensionale (Figura 3.1):

$$\begin{aligned}\int_S \rho \mathbf{U} \cdot \mathbf{n} \phi \mathrm{d}S = &- \rho \left[(SU\phi)_\mathrm{w} - (SU\phi)_\mathrm{c} + (SV\phi)_\mathrm{s} \right] + \\ &+ \left[-(SV\phi)_\mathrm{n} + (SW\phi)_\mathrm{f} - (SW\phi)_\mathrm{b} \right],\end{aligned} \tag{3.4}$$

dove U, V, W sono le velocità nelle direzioni x, y, z, gli indici w, e, s, n, f, t denotano le facce ovest, est, sud, nord, front e back.
Il termine diffusivo può essere approssimato in modo analogo a quanto fatto per il termine precedente:

$$\begin{aligned}\int_\mathrm{S} \Gamma \nabla \phi \mathbf{n} \mathrm{d}S = &- \left[\left(S\Gamma \frac{\partial \phi}{\partial x} \right)_\mathrm{w} - \left(S\Gamma \frac{\partial \phi}{\partial x} \right)_\mathrm{e} + \left(S\Gamma \frac{\partial \phi}{\partial y} \right)_\mathrm{s} \right] + \\ &+ \left[- \left(S\Gamma \frac{\partial \phi}{\partial y} \right)_\mathrm{n} + \left(S\Gamma \frac{\partial \phi}{\partial z} \right)_\mathrm{f} \left(S\Gamma \frac{\partial \phi}{\partial z} \right)_\mathrm{b} \right].\end{aligned} \tag{3.5}$$

Infine, per quanto riguarda il termine di generazione, viene semplicemente valutato un valore medio del termine sorgente nel volume di controllo:

$$\int_\Omega S_\phi \mathrm{d}V \approx \overline{S}_\phi V. \tag{3.6}$$

In conclusione, l'equazione differenziale di trasporto (3.1) è stata ora trasformata in equazioni che possono essere risolte algebricamente (3.4), (3.5), (3.6), tuttavia i valori di ϕ, U_j e Γ sulle facce delle celle devono essere interpolati, in quanto le equazioni non sono risolte sulle facce [18].

3.1.2 Metodi di discretizzazione spaziale - `OpenFOAM`

Al fine di valutare gli integrali è necessario interpolare i valori di ϕ e del suo gradiente sulle facce delle celle a partire dai valori di ϕ nei centri delle celle. In altre parole è necessario valutare i flussi convettivi e diffusivi della proprietà ϕ sulle facce a partire dal valore al centro della cella, che è l'unico noto e calcolato nel metodo ai volumi finiti.

Per quanto riguarda le simulazioni per la soluzione del campo di moto in `OpenFOAM` si riporta a titolo di esempio il metodo LUDS (*Linear Upwind Difference Scheme*) utilizzato per l'interpolazione del valore della proprietà ϕ (ad esempio la velocità) del termine convettivo [20]. Si immagini di voler determinare il valore di ϕ nella faccia est (un caso ancora più semplificato di quello raffigurato nel caso bidimensionale della Figura 3.1, in quanto si considerano solo le celle ad est e ovest, caso monodimensionale), i nodi computazionali utilizzati per l'interpolazione dipendono dalla direzione della velocità del fluido, ossia dalla direzione del flusso convettivo, in particolare se $\mathbf{U} \cdot \mathbf{n} > 0$:

$$\phi_e = \phi_P \lambda_e + \phi_W \left(1 - \lambda_e \right), \tag{3.7}$$

dove

$$\lambda_e = \frac{x_e - x_W}{x_E - x_W}; \tag{3.8}$$

È possibile dimostrare che questo schema di discretizzazione è del secondo ordine, cioè il suo errore associato è proporzionale al quadrato della dimensione delle celle. Si considerino gli sviluppi in serie di Taylor di ϕ_e e ϕ_W dove H rappresenta i termini di ordine superiore:

$$\phi_W = \phi_P + (x_W - x_P)\left(\frac{\partial\phi}{\partial x}\right)_P + \frac{1}{2}(x_W - x_P)^2\left(\frac{\partial^2\phi}{\partial x^2}\right)_P + H, \tag{3.9}$$

$$\phi_e = \phi_P + (x_e - x_P)\left(\frac{\partial\phi}{\partial x}\right)_P + \frac{1}{2}(x_e - x_P)^2\left(\frac{\partial^2\phi}{\partial x^2}\right)_P + H, \tag{3.10}$$

dall'Eq.(3.9) è possibile esplicitare il gradiente di ϕ in P:

$$\left(\frac{\partial\phi}{\partial x}\right)_P = \frac{\phi_E}{x_E - x_P} - \frac{\phi_P}{x_E - x_P} - \frac{1}{2}(x_E - x_P)\left(\frac{\partial^2\phi}{\partial x^2}\right)_P - \frac{H}{x_E - x_P}, \tag{3.11}$$

che sostituito nell'Eq.(3.10) permette di verificare l'ordine dell'errore associato alla discretizzazione:

$$\begin{aligned}\phi_e =& \phi_P + \frac{(x_e - x_P)\,\phi_E}{x_E - x_P} - \frac{(x_e - x_P)\,\phi_P}{x_E - x_P} + \frac{(x_e - x_P)(x_E - x_P)}{2}\left(\frac{\partial^2\phi}{\partial x^2}\right)_P + \\ &+ \frac{(x_e - x_P)^2}{2}\left(\frac{\partial^2\phi}{\partial x^2}\right)_P + H\frac{(x_e - x_P)}{(x_E - x_P)} = \phi_E\lambda_e + \phi_P(1 - \lambda_e) + O\left(\Delta x^2\right).\end{aligned} \tag{3.12}$$

Si consideri una problema monodimensionale (schematizzato in Figura 3.2) in cui i termini di accumulo e generazione siano nulli, la densità e il coefficiente di trasporto siano costanti, allora è necessario risolvere la seguente equazione per ciascuna cella:

$$(\rho U\phi)_e\, S_e - (\rho U\phi)_w\, S_w = \left(\Gamma\frac{\partial\phi}{\partial x}\right)_e S_e - \left(\Gamma\frac{\partial\phi}{\partial x}\right)_w S_w. \tag{3.13}$$

Applicando le formule di derivazione numerica del secondo ordine:

$$\phi_e - \phi_w - \frac{\Gamma}{\rho U}\frac{\phi_{EE} - \phi_P}{2\Delta x} + \frac{\Gamma}{\rho U}\frac{\phi_E - \phi_W}{2\Delta x} = 0. \tag{3.14}$$

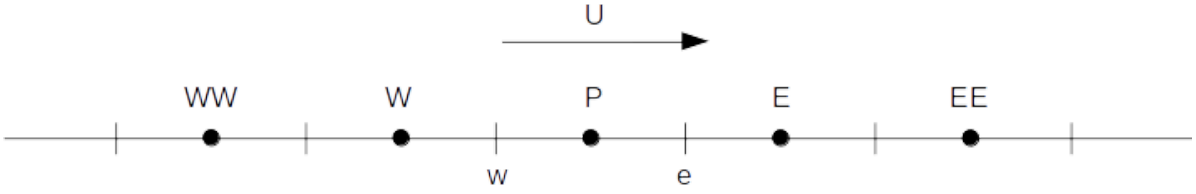


Figura 3.2: Definizione delle celle per un problema monodimensionale

La proprietà ϕ è valutata sulle facce delle celle a partire dal suo valore nel centro attraverso lo schema LUDS, si considera una griglia regolare per cui $\lambda_e = \frac{3}{4}$:

$$\frac{3}{4}\phi_P + \frac{1}{4}\phi_W - \frac{3}{4}\phi_W - \frac{1}{4}\phi_{WW} - \frac{\Gamma}{\rho U}\frac{\phi_{EE} - \phi_P}{2\Delta x} + \frac{\Gamma}{\rho U}\frac{\phi_E - \phi_W}{2\Delta x} = 0. \tag{3.15}$$

É possibile arrangiare i termini nel seguente modo:

$$-\frac{\mathrm{Pe_c}}{4}\phi_{WW} - (\mathrm{Pe_c}+1)\,\phi_W + \left(\frac{3\mathrm{Pe_c}+2}{2}\right)\phi_P + \phi_E + \phi_{EE} = 0. \tag{3.16}$$

Se si considera una discretizzazione spaziale costituita da N celle sarà necessario risolvere un sistema lineare del seguente tipo:

$$\begin{bmatrix} \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & -\frac{\mathrm{Pe_c}}{4} & -(\mathrm{Pe_c}+1) & \left(\frac{3\mathrm{Pe_c}+2}{2}\right) & 1 & 1 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_N \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{3.17}$$

in cui $\mathrm{Pe_c}$ è il numero di Peclet di cella che esprime il rapporto tra il tempo associato al fenomeno diffusivo e il tempo di permanenza nella cella:

$$\mathrm{Pe_c} = \frac{\Delta x^2/(\Gamma/\rho)}{\Delta x/U} = \frac{\rho U \Delta x}{\Gamma}, \tag{3.18}$$

la soluzione è limitata per ogni valore di $\mathrm{Pe_c}$.

É stata ottenuta una matrice sparsa contenente cinque diagonali di dimensione pari al numero di celle della discretizzazione spaziale. Per risolvere il sistema lineare è possibile usare dei metodi iterativi, come il metodo di Gauss-Siedel, che ben si presta alla risoluzione di matrici di questo tipo.

In conclusione, con il metodo dei volumi finiti è sufficiente risolvere sistemi lineari analoghi a quello di Eq.3.17, invece di risolvere equazioni differenziali come Eq.3.1.

Questo metodo presenta numerosi vantaggi, ma soffre di un problema molto serio. Non è in grado, a meno che non vengano rispettate alcune condizioni, di garantire la limitatezza della soluzione. Nel caso delle simulazioni transitorie del trasporto di colloidi attraverso mezzi porosi è, infatti, necessario introdurre degli schemi di discretizzazione limitati in modo da evitare che la concentrazione locale del colloide assuma valori al di fuori del range di validità, come ad esempio valori superiori alla concentrazione in ingresso o negativi. Fra gli schemi introdotti è utile citare i TVD (*Total Variation Diminishing*). Essi sono in grado di mantenere l'elevata accuratezza degli schemi ad alto ordine eliminando le oscillazioni caratteristiche dell'instabilità di questi metodi [20]. Gli schemi TVD sono in grado quindi di preservare la monotonicità della soluzione evitando la formazione di nuovi massimi e minimi o l'accentuazione di quelli già esistenti. Affinché la monotonicità sia soddisfatta è necessario che la variazione totale della soluzione riportata nell'Eq.(3.19) in riferimento alla Figura 3.3 non aumenti nel tempo:

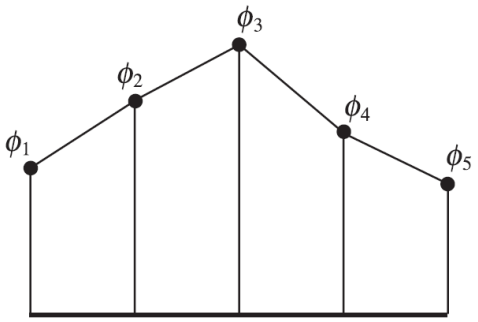


Figura 3.3: Esempio di discretizzazione per illustrare la variazione totale [20]

$$\begin{aligned} TV(\phi) &= |\phi_2 - \phi_1| + |\phi_3 - \phi_2| + |\phi_4 - \phi_3| + |\phi_5 - \phi_4| \\ &= |\phi_3 - \phi_1| + |\phi_5 - \phi_3| \,. \end{aligned} \tag{3.19}$$

La proprietà ϕ sulla faccia est per un generico schema di discretizzazione può essere espressa nel seguente modo:

$$\phi_e = \phi_P + \frac{1}{2}\psi(r)\left(\phi_E - \phi_P\right). \tag{3.20}$$

dove:

$$r = \left(\frac{\phi_P - \phi_W}{\phi_E - \phi_P}\right) \tag{3.21}$$

Si può verificare confrontando con l'Eq.(3.7) che per uno schema LUDS $\psi(r) = r$. In Figura 3.4 è riportato il diagramma r-ψ in cui è possibile confrontare diversi schemi di discretizzazione per griglie regolari.

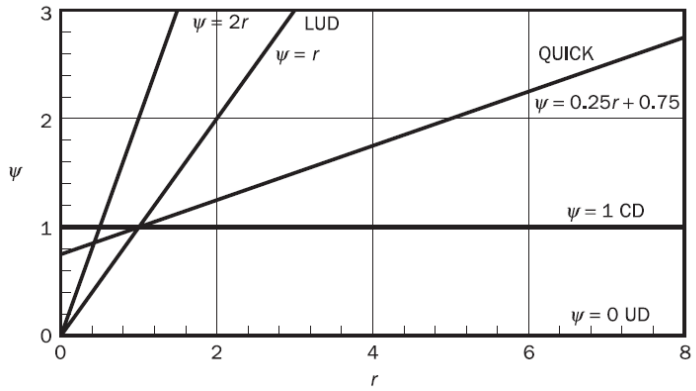


Figura 3.4: Diagramma r-ψ per diversi schemi di discretizzazione (LUD: *Linear Upwind Difference scheme*, QUICK: *Quadratic Upstream Interpolation for Convective Kinematics*, CD: *Central Difference scheme*, UD: *Upwind Difference scheme*) [20]

L'idea alla base degli schemi TVD è di introdurre una modifica agli schemi classici in modo da forzare la relazione r-ψ a rimanere nelle zone del diagramma concesse dalle ipotesi elencate di seguito:

- se $0 < r < 1$ il limite superiore è $\psi(r) = 2r$, allora $\psi(r) \leqslant r$;
- se $r \geqslant 1$ il limite superiore è $\psi(r) = 2$, allora $\psi(r) \leqslant 2$;
- se lo schema è del secondo ordine è necessario che la funzione $\psi(r)$ passi nel punto $(1, 1)$.

Inoltre il range dei possibili schemi del secondo ordine che possono essere degli schemi TVD sono limitati dagli schemi CDS e LUDS:

- se $0 < r < 1$ il limite inferiore è $\psi(r) = r$ e il limite superiore è $\psi(r) = 1$, allora $r \leqslant \psi(r) \leqslant 1$;
- se $r \geqslant 1$ il limite inferiore è $\psi(r) = 1$ e il limite superiore è $\psi(r) = 2$, allora $1 \leqslant \psi(r) \leqslant r$.

In conclusione, in Figura 3.5 sono riportate le zone disponibili del diagramma r-ψ per un schema TVD del secondo ordine.

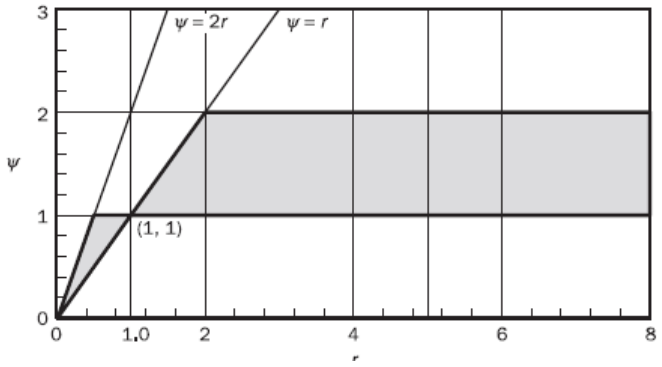


Figura 3.5: Diagramma r-ψ con le regioni accessibili a schemi del secondo ordine [20]

3.1.3 Metodi di discretizzazione temporale - `OpenFOAM`

Per la risoluzione delle simulazioni transitorie è stato utilizzato come schema di discretizzazione temporale il metodo di Eulero implicito, che è uno schema del primo ordine incondizionatamente stabile:

$$\phi_i^{n+1} = \phi_i^n + \Delta t \left(-U \frac{\phi_{i+1}^{n+1} - \phi_{i-1}^{n+1}}{2\Delta x} + \frac{\Gamma \phi_{i+1}^{n+1} - 2\phi_i^{n+1} + \phi_{i-1}^{n+1}}{\Delta x^2} \right). \tag{3.22}$$

Se si definiscono il numero di Courant (c) e il tempo caratteristico di convezione (d):

$$\mathrm{c} = \frac{U\Delta t}{\Delta x}, \qquad \mathrm{d} = \frac{\Gamma \Delta t}{\rho \Delta x^2}, \tag{3.23}$$

è possibile rielaborare l'Eq.(3.22):

$$\phi_i^{n+1} = (1-2d)\phi_i^n + \left(d - \frac{c}{2}\right)\phi_{i+1}^n + \left(d + \frac{c}{2}\right)\phi_{i-1}^n, \tag{3.24}$$

che in forma vettoriale:

$$\underline{\phi}^{n+1} = A\underline{\phi}^n, \tag{3.25}$$

dove $\underline{\phi}^{n+1}$ è il vettore della proprietà ϕ nelle celle della discretizzazione spaziale all'iterata $n+1$, $\underline{\phi}^n$ è il corrispettivo vettore all'iterata n e A è una matrice tri-diagonale. Questo sistema di equazioni può essere risolto con il metodo di Gauss-Seidel al fine di determinare come varia la proprietà ϕ nelle celle del dominio spaziale nel tempo.

3.1.4 Algoritmo SIMPLE

Per ottenere il campo di moto e di pressione di un sistema fluido è necessario risolvere l'equazione di continuità e l'equazione di Navier Stokes, che possono essere trasformate attraverso il metodo dei volumi finiti in equazioni algebriche introducendo un'opportuna discretizzazione spaziale, tuttavia la soluzione contemporanea delle due equazioni è problematica in quanto non è presente un'equazione per ricavare esplicitamente la pressione.

Se il fluido è comprimibile è possibile ricavare dall'equazione di continuità la densità, da un'opportuna equazione di stato la pressione ed infine il campo di velocità attraverso l'equazione di trasporto della quantità di moto.

Se il fluido è incomprimibile invece non è possibile utilizzare in modo diretto l'equazione di continuità, per risolvere le equazioni si applica l'algoritmo SIMPLE (*Semi-Implicit Method for Pressure Linked Equations*), esso è un metodo iterativo che necessita dell'iniziale assunzione di un campo di pressione e consta dei seguenti step:

1. dato il campo di pressione viene risolta l'equazione del momento in modo da ottenere il campo di moto;

2. dato il campo di moto si risolve l'equazione di Poisson (3.26) in modo da ottenere il campo di pressione aggiornato:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left(\frac{\partial\left(\rho U_i U_j\right)}{\partial x_j}\right). \tag{3.26}$$

   L'equazione di Poisson è ottenuta applicando la divergenza all'equazione di Navier-Stokes e sostituendo l'equazione di continuità.

3. si confrontano il campo di pressione utilizzato nello step 1 e quello ottenuto nello step 2, se non si è raggiunta la convergenza si itera nuovamente utilizzando il nuovo campo di pressione nello step 1.

3.2 Gestione dei dati e orchestrazione delle simulazioni

Al fine di produrre un numero sufficiente di sample per allenare una rete neurale è necessario automatizzare la risoluzione di un elevato numero di simulazioni. In Figura 3.6 è schematizzato il workflow per la risoluzione delle simulazioni di flusso e trasporto attraverso mezzi porosi, tuttavia è possibile adattarlo alle simulazioni di flusso attraverso il cone mill se si escludono i punti 5 e 6.

Figura 3.6: Workflow per la creazione del dataset per il training della rete neurale

Nei paragrafi successivi verranno discussi i diversi punti del workflow facendo riferimento alla struttura degli script redatti per ciascun punto.

3.2.1 Preparazione del template e creazione delle cartelle di simulazione

In primo luogo è opportuno stabilire quali siano gli obiettivi di predizione della rete neurale e quali siano le feature di input sfruttate per tale scopo, esse possono essere delle variabili geometriche oppure delle condizioni operative. Dopodiché viene creata una cartella di simulazione, detta template, contenente dei file di impostazione che vengono modificati ad ogni variazione delle feature stabilite. Per ogni feature in input è necessario scegliere un range di variabilità che caratterizza l'ampiezza di esplorazione del caso, tenendo conto che le predizioni dei modelli data-driven saranno accurate se si investiga una condizione ricadente nel range.

Una volta creato il template, per ottenere il numero di cartelle di simulazione corrispondente ai casi scelti, è stato scritto uno script in linguaggio `Python` che esegue le seguenti operazioni:

- dato il range di variabilità delle feature per ciascuna di esse crea un vettore di valori random uniformi avente lunghezza pari al numero di simulazioni volute;
- dato il numero di simulazioni volute copia la cartella di template e la incolla rinominandola in modo progressivo;

- per ciascuna nuova cartella crea un file di testo in cui sono riportati i valori delle feature estratti dalla componente dei vettori corrispondente alla numerazione della cartella;

- crea un file di sommario in cui si riassumono i valori delle feature per ciascuna simulazione.

3.2.2 Risoluzione delle simulazioni del campo di moto

Grazie al passaggio precedente sono disponibili tutte le cartelle di simulazioni impostate pronte per la risoluzione che prevede:

- creazione della mesh (esecuzione di `blockMesh` ed, eventualmente, `snappyHexMesh`);

- controllo della mesh (esecuzione di `checkMesh`);

- risoluzione del campo di moto (esecuzione di `simpleFoam` oppure `SRFSimpleFoam`);

- funzioni di post processing per l'estrazione dei risultati.

Poiché il numero delle simulazioni da risolvere è dell'ordine delle migliaia, è opportuno utilizzare un sistema informatico in grado di risolvere in parallelo le diverse simulazioni in modo da diminuire notevolmente il tempo necessario per la creazione del dataset. A tal fine è stato usato un cluster composto di nodi con CPU Intel Xeon E5-2680 v3 2.50 GHz 12-cores, con 3.7 TB RAM totali [1]. In particolare, è stato realizzato uno script in `BASH` che sottomette allo scheduler del cluster uno script presente in ogni cartella di simulazione contenente tutti i comandi precedentemente esposti.

3.2.3 Controllo delle simulazioni

Dopo aver effettuato le simulazioni è necessario valutare se i risultati siano tutti utilizzabili, in particolare è possibile che per alcuni di essi non siano stati raggiunti i criteri di convergenza, che le simulazioni si siano interrotte in modo anomalo o che la mesh non sia stata generata correttamente. É stato creato uno script in linguaggio `Python` in grado di individuare tali simulazioni attraverso i seguenti criteri:

- eseguita la funzione di post-processing che salva in un file il numero dell'iterazione finale della simulazione, se il numero è pari al numero massimo di iterazioni impostate allora la simulazione non ha raggiunto la convergenza e viene esclusa;

- eseguita la funzione di post-processing che salva in un file il numero dell'iterazione finale della simulazione, se non è presente alcun valore allora la simulazione è stata interrotta e viene esclusa;

- aperto il file prodotto durante il controllo della mesh (comando `checkMesh`) si cerca la parola 'Failed' nel testo, se presente allora la discretizzazione spaziale è fallita, la mesh risulta quindi inutilizzabile.

[1] Risorse di calcolo fornite da hpc@polito, progetto di Academic Computing del Dipartimento di Automatica e Informatica presso il Politecnico di Torino (http://www.hpc.polito.it)

3.2.4 Predisposizione per le simulazioni successive

Nel caso studio del flusso attraverso mezzi porosi è stato anche simulato il trasporto di un colloide, si prevede quindi la risoluzione di un equazione di trasporto aggiuntiva in seguito all'ottenimento del campo di moto. Queste simulazioni sono state gestite inserendo la cartella di simulazione all'interno di quella relativa al campo di moto con un opportuno file di testo contenente le feature di input. Tuttavia prima di avviare le nuove simulazioni è necessario copiare la cartella `polyMesh` contenente tutte le informazioni relative alla discretizzazione spaziale nella cartella `constant` della nuova simulazione, inoltre i risultati del campo di moto vengono copiati e incollati nella cartella contenente le condizioni al contorno della nuova simulazione. Queste operazioni sono state eseguite attraverso uno script in linguaggio `Python`.

3.2.5 Risoluzione delle simulazioni di trasporto

Impostate le nuove simulazioni è possibile eseguire i seguenti comandi:

- risoluzione del trasporto di concentrazione scalare (comando `scalarTransportFoam`);
- funzioni di post-processing per l'estrazione dei risultati.

Come nel caso delle simulazioni del campo di moto anche per queste è stato prodotto uno script `BASH` in grado di lanciare gli script presenti in ciascuna cartella di simulazione in modo tale da poter risolvere in parallelo un maggior numero di simulazioni attraverso il cluster.

---------- SUMMARY FILE ----------

exp	Inlet pressure	Sphere diameter	Porosity	Standard Deviation	Colloid diameter	Permeability	Deposition efficiency
1	0.000427526	0.000168	0.659243037	0.1	6.10E-08	5.10E-10	0.037003778
2	0.000377741	0.000173	0.633893043	0.3	4.00E-08	3.21E-10	0.084340332
3	0.00040068	0.000178	0.610949231	0.1	4.80E-08	2.84E-10	0.088034477
4	0.000489692	0.00019	0.53363515	0.1	8.00E-08	7.76E-11	0.080436551
5	0.000435061	0.000185	0.579721559	0.3	6.70E-08	1.94E-10	0.069349337
6	0.000351594	0.000114	0.622067701	0.3	9.70E-08	8.50E-11	0.09183354
7	0.000429125	0.000132	0.624420816	0.2	3.10E-08	1.69E-10	0.083580121
8	0.000383535	0.000101	0.56639726	0.2	7.30E-08	5.36E-11	0.117309091
9	0.000500941	0.000161	0.595412512	0.2	4.60E-08	2.01E-10	0.076443349
10	0.000412489	0.000169	0.527509059	0.3	9.30E-08	1.29E-10	0.053336019
11	0.000449003	0.00011	0.6628799	0.2	9.60E-08	1.83E-10	0.061270648
12	0.000334786	0.000102	0.508292038	0.2	9.90E-08	2.94E-11	0.122916658
13	0.000441709	0.000198	0.466460442	0.2	4.20E-08	5.38E-11	0.118550359
14	0.000339876	0.000153	0.60203687	0.3	6.90E-08	2.56E-10	0.050472892
15	0.000406892	0.000134	0.655782893	0.2	9.10E-08	2.72E-10	0.075755629
16	0.000478788	0.000129	0.501431843	0.1	5.30E-08	3.66E-11	0.120583601
17	0.000346602	0.000101	0.449811904	0.3	6.80E-08	1.91E-11	0.15230202

Feature di input Output Sample

Figura 3.7: Esempio di file di testo riassuntivo

3.2.6 Estrazione dei risultati

L'ultimo step del workflow è un passaggio fondamentale in quanto vengono preparati i sample che saranno forniti alla rete neurale durante la fase di training e viene eseguito attraverso uno script in `Python`. In particolare, si escludono dall'estrazione dei risultati tutte le simulazioni problematiche individuate con il punto 3, poi i risultati vengono

estratti, eventualmente elaborati ed infine salvati in un vettore. Lo step finale prevede la redazione di un file di testo (come da esempio in Figura 3.7) in cui ad ogni sample corrisponde una riga con una relativa numerazione, le feature di input (estratte dal file precedentemente creato nel punto 1) e i risultati.

3.3 Costruzione del modello data-driven: MATLAB

Per la realizzazione dei modelli data-driven è stato sfruttato l'ambiente di calcolo MATLAB R2019a. Le reti neurali utilizzate nel lavoro di tesi sono dei percettroni o dei percettroni multi layer, queste semplici architetture sono definite *shallow neural networks* in MATLAB e possono essere progettate con il seguente comando:
`feedforwardnet(hiddenSizes,trainFcn)`
dove `hiddenSizes` è un vettore riga contenente il numero di neuroni per layer e `trainFcn` indica l'algoritmo di training che si vuole utilizzare. Ad esempio il comando:
`net-prova = feedforwardnet([10 5])`
realizza una rete neurale avente due hidden layer, il primo avente 10 neuroni e il secondo 5; quando l'algoritmo di training non è specificato viene assunto di default l'algoritmo di Levenberg-Marquardt (`trainlm`) descritto in seguito. Dopodiché la rete viene allenata attraverso l'algoritmo di training scelto:
`[net,tr] = train(net,...)`
dove `net` è la rete progettata e poi allenata, `tr` è una struttura contenente le informazioni riguardanti il training. Ad esempio con il comando:
`[net-trained,tr] = train(net-prova,x,t)`
si allena la rete net-prova fornendogli i sample caratterizzati da una matrice di feature in input x e una matrice di target in output t. Una volta ottenuta la rete neurale allenata è possibile utilizzarla per eseguire delle predizioni, ad esempio:
`predizione = net-trained(input)`
dove `input` ha lo stesso numero di feature di x.

3.3.1 Algoritmo di Levenberg-Marquardt

Nel lavoro di tesi è stato utilizzato l'algoritmo di training di Levenberg-Marquardt implementato di default in MATLAB. Questo algoritmo discende da quello presentato nel capitolo precedente e nell'ALLEGATO A (steepest descent backpropagation) al quale sono state apportate delle modifiche al fine di diminuirne i problemi di convergenza [21], assumendo un learning rate variabile. L'idea che sta alla base di questa scelta è che quando la funzione da minimizzare, il mean square error, è piatta è opportuno usare un learning rate alto per spostarsi dalla zona, mentre se la curvatura è elevata è meglio procedere a piccoli step con un learning rate più basso. Il learning rate è legato all'inverso della matrice Hessiana della funzione errore, al fine di rendere la matrice invertibile essa viene approssimata sfruttando un parametro che possa essere adattato durante il training (per una descrizione precisa dell'algoritmo si rimanda al capitolo 12 di [21]).

In MATLAB il dataset fornito all'algoritmo viene suddiviso in tre set:

- *training set*: questi sample vengono utilizzati per aggiornare i pesi e i bias della rete neurale, sono il 70% dei totali;

- *validation set*: questi sample servono per determinare il momento in cui il training è completo, sono il 15% dei totali;

- *test set*: questi sample servono per valutare la performance della rete allenata, sono il 15% dei totali.

Le funzioni di attivazione utilizzate di default in `MATLAB` sono delle tangenti iperboliche per i neuroni degli hidden layer, mentre delle ReLU per i neuroni dell'output layer.

4 | Simulazioni effettuate e condizioni operative

In questo capitolo viene descritta l'impostazione delle simulazioni eseguite su `OpenFOAM`, in particolare sono discussi in dettaglio le geometrie create e la loro discretizzazione spaziale nella sezione 4.1, i solver utilizzati e le condizioni al contorno implementate nella sezione 4.2. Vengono presentate in parallelo le impostazioni dei due casi studio considerati: la produzione di emulsioni alimentari e il flusso attraverso mezzi porosi. Nella sezione 4.3 sono definiti gli input e gli output scelti per la realizzazione dei modelli data-driven per i diversi casi studio, in particolare sono presentati i range di esplorazione delle feature scelti per la realizzazione delle simulazioni finalizzate alla creazione del dataset.

4.1 Geometria e griglia di calcolo

In questa sezione verranno descritte le geometrie adottate per la simulazione dei due casi studio.

4.1.1 Emulsioni alimentari

Come accennato nell'introduzione la produzione della maionese e di un'ampia gamma di condimenti alimentari prevede un'iniziale miscelazione in cui l'olio è aggiunto lentamente alla fase acquosa e ai tuorli d'uovo, in seguito l'emulsione viene completata in un secondo miscelatore, detto cone mill, in cui si raggiungono elevati sforzi di taglio in modo che le gocce d'olio raggiungano le dimensioni desiderate: il processo è schematizzato in Figura 4.1 [10]. L'obiettivo delle simulazioni è stato modellare il secondo step di miscelazione, in Figura 4.2 è illustrata l'intera apparecchiatura del cone mill utilizzato per condurre il lavoro sperimentale presentato in *Dubbelboer et al.* [10], a destra, e il particolare oggetto delle simulazioni, a sinistra. L'emulsione scorre attraverso i due coni tronchi assialsimmetrici e concentrici, quello esterno è fisso mentre quello interno ruota: in questo lavoro sono stati esplorati sistemi con valori diversi del gap tra rotore e statore. In `OpenFOAM`, tramite l'utility `blockMesh`, è stata creata una geometria assialsimmetrica, in Figura 4.3 sono riportate le dimensioni prinicpali nella vista frontale (a destra) e le relazioni geometriche utilizzate per la realizzazione dell'assialsimmetria (a sinistra). Il fluido scorre all'interno della corona conica, entra dalla base superiore ed esce da quella

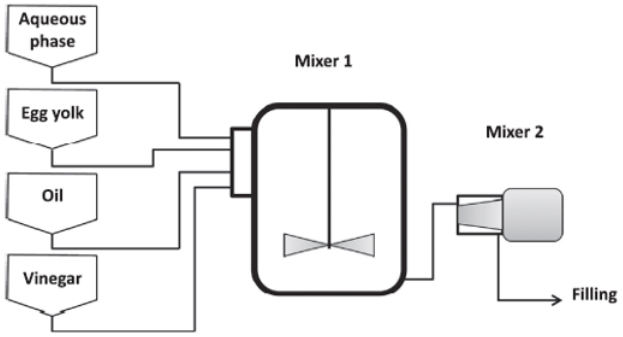


Figura 4.1: Schema del processo di produzione di emulsioni alimentari [10]

Figura 4.2: cone mill utilizzato per condurre gli studi sperimentali [22]

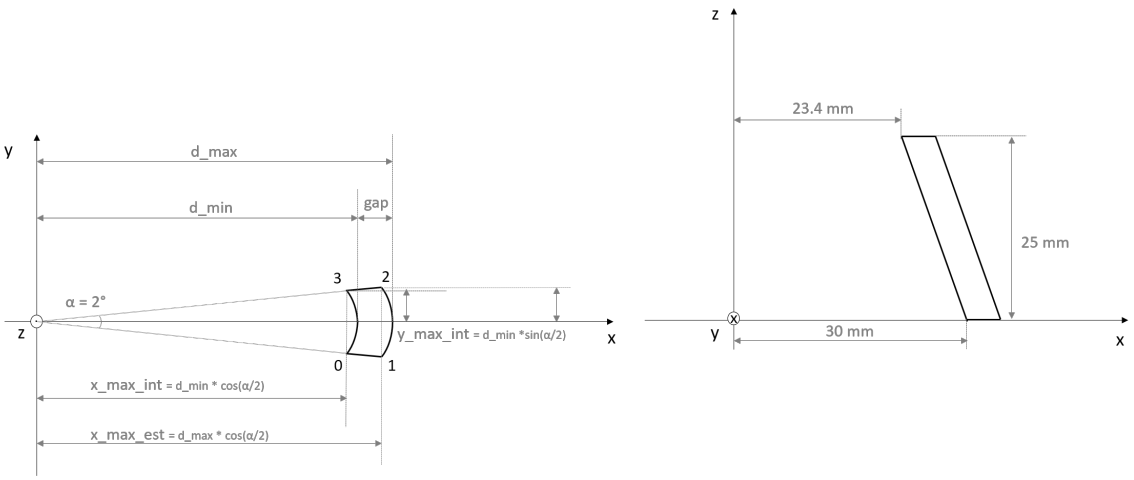


Figura 4.3: Geometria del cone mill implementata in `OpenFOAM`, vista dall'alto (a sinistra) e vista frontale (a destra)

inferiore, la parete cilindrica esterna ed interna sono dei *wall*, mentre la faccia frontale e posteriore sono denominate *wedge* in quanto sono le sezioni che per estrusione danno luogo alla geometria assialsimmetrica (si veda la prossima sezione per la definizione matematica delle condizioni al contorno).

La mesh realizzata tramite l'utility `blockMesh` è uniforme, lo spessore del gap è stato suddiviso in 40 celle (i.e.: in direzione radiale) mentre la direzione verticale (z) in 100 celle; è stata adottata questa discretizzazione spaziale in quanto è stato verificato il raggiungimento dell'indipendenza dei risultati dalla griglia.

4.1.2 Mezzi porosi

Le simulazioni del flusso attraverso mezzi porosi sono state effettuate con diversi tipi di geometrie bidimensionali: dapprima è stato simulato il flusso attraverso un canale contenente due ostacoli circolari, per poi passare ad arrangiamenti casuali di 30 sfere (cerchi in 2D).

Per quanto riguarda le simulazioni del flusso attraverso il canale le geometrie realizzate sono semplici, il caso base è riportato in Figura 4.4. Il flusso scorre orizzontalmente, entra dalla faccia sinistra ed esce dalla faccia destra, i bordi delle sfere sono dei wall, sulla faccia superiore ed inferiore è stata imposta una condizione di simmetria. Per quanto riguarda la mesh, dapprima è stato creato il rettangolo attraverso l'utility `blockMesh`, in seguito le sfere sono state introdotte tramite l'utility `snappyHexMesh`, utilizzata per creare la mesh nello spazio interno così ricavato. La discretizzazione spaziale è stata scelta in modo tale che il numero di celle per diametro sia uguale a 28 [23] [24].

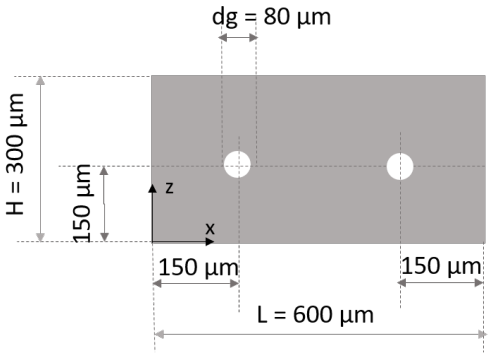
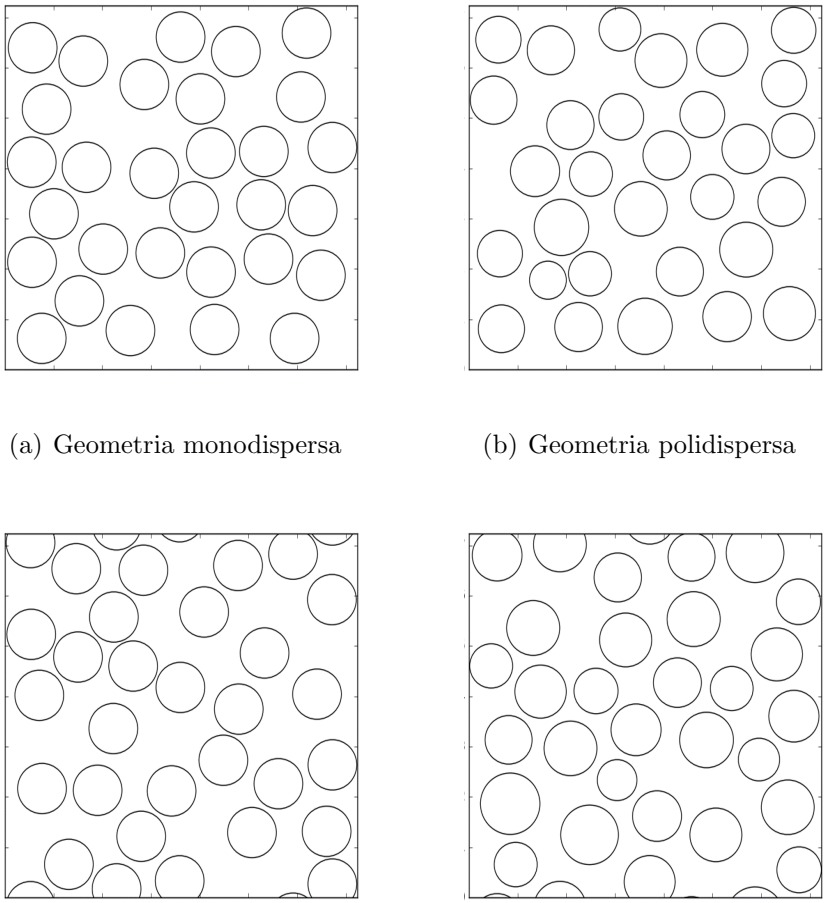


Figura 4.4: Canale contenente due ostacoli, geometria preliminare per la simulazione del flusso attraverso mezzi porosi

Per quanto riguarda le geometrie random sono state testate le seguenti tipologie:

- 30 sfere con dimensione identica, Figura 4.5(a);
- 30 sfere con diametro polidisperso, Figura 4.5(b);
- 30 sfere con dimensione identica e geometria periodica, Figura 4.5(c);
- 30 sfere con diametro polidisperso e geometria periodica, Figura 4.5(d).

Così come nel caso precedente il fluido entra dalla faccia sinistra ed esce dalla faccia destra, i bordi delle sfere sono wall. I casi a-b e c-d si distinguono per la condizione che è stata impostata sulla faccia superiore ed inferiore, infatti, per le geometrie a-b, in cui non ci sono sfere che attraversano il bordo del dominio spaziale, la condizione scelta è di simmetria, mentre per le geometrie c-d, in cui le sfere che fuoriescono dal bordo superiore sono completate da quelle fuoriuscenti dal bordo inferiore, la condizione scelta è di periodicità, in questo modo viene rappresentata una condizione di flusso attorno a sole geometrie circolari.



(a) Geometria monodispersa

(b) Geometria polidispersa

(c) Geometria monodispersa periodica

(d) Geometria polidispersa periodica

Figura 4.5: Tipi di geometri random costruite per la simulazione di mezzi porosi

Per la realizzazione delle diverse tipologie di geometrie random sono stati redatti altrettanti script in linguaggio `Python`, si riporta nell'ALLEGATO B a titolo di esempio il codice completo per la disposizione di sfere come in Figura 4.5(d), il caso più complesso. I dati in input al codice sono il numero di sfere che si vogliono posizionare, la porosità che si vuole ottenere, il diametro medio delle sfere e la deviazione standard della distribuzione dei diametri. In base a questi valori il codice valuta l'area totale del mezzo poroso,

calcolata come l'area occupata dalle sfere divisa per la porosità, e di conseguenza la lunghezza del lato del quadrato. Con la funzione `random.normal` del pacchetto `numpy` è possibile ottenere un vettore di lunghezza pari al numero di sfere desiderate e contenente i diametri delle sfere caratterizzate da una certa deviazione standard rispetto al valore medio. Le componenti di questo vettore vengono ordinate in modo decrescente così da dover posizionare dapprima le sfere più grandi e in seguito quelle più piccole, in questo modo è più facile ottenere dei mezzi porosi a bassa porosità (evitando molte sovrapposizioni tra sfere nelle iterate finali della procedura). Il passo successivo del codice prevede l'inserimento delle prime due sfere, la prima è posizionata per intero nella metà sinistra del dominio geometrico (Figura (4.6), sfera A), la seconda è posizionata sul bordo nella metà destra del dominio geometrico (Figura (4.6), sfera BU-BL). In seguito, il codice posiziona le restanti 28 sfere utilizzando il seguente procedimento iterativo:

- viene assegnata una posizione random;
- se la sfera ricade completamente all'interno del dominio spaziale si valuta la distanza rispetto alle altre sfere per evitare sovrapposizioni, in particolare se la sfera da posizionare è la D della Figura (4.6):
    - si valuta che la distanza tra la sfera D e tutte le sfere "intere", come A, sia maggiore di una distanza minima;
    - si valuta che la distanza tra la sfera D e tutte le sfere ripartite tra il bordo superiore ed inferiore sia maggiore di una distanza minima, in particolare è necessario verificare tale condizioni per entrambe le parti della sfera;
- se la sfera ricade parzialmente all'interno del dominio spaziale, la parte fuoriuscente è posizionata sulla faccia opposta e si valuta la distanza di entrambe le parti rispetto alle altre sfere per evitare sovrapposizioni, in particolare se la sfera da posizionare è la C (le due parti CU e CL) della Figura (4.6):
    - si valuta che la distanza tra le parti CL e CU e tutte le sfere "intere", come A, sia maggiore di una distanza minima;
    - si valuta che la distanza tra le parti CL e CU e tutte le sfere ripartite tra il bordo superiore ed inferiori sia maggiore di una distanza minima, in particolare è necessario verificare tale condizione per entrambe le parti della sfera;
- se tutte le condizioni sulle distanze sono verificate allora la posizione della sfera può essere salvata e si può procedere al posizionamento della sfera successiva;
- se non sono verificate tutte le condizioni sulle distanze allora si reitera questa procedura provando delle nuove coordinate casuali di posizione per un numero massimo di 100 volte oltre le quali la creazione della geometria risulta fallita.

Una volta che tutte le sfere sono state posizionate il codice crea un file di testo contenente le coordinate di ciascuna di esse, necessarie per la creazione della mesh.

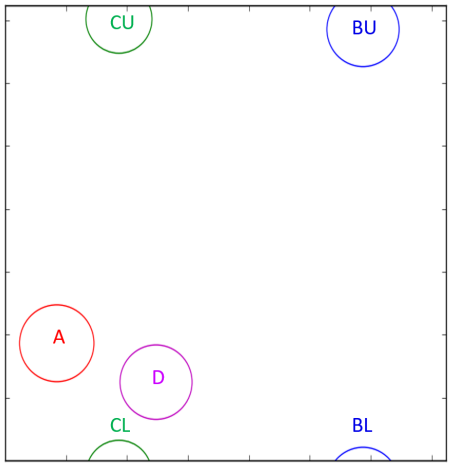


Figura 4.6: Esempio per la creazione di geometrie random periodiche polidisperse

Per quanto riguarda la mesh, dapprima è stato creato il quadrato attraverso l'utility `blockMesh` e la discretizzazione spaziale è stata scelta in modo tale che il numero di celle per diametro medio sia uguale a 35. É stato scelto un valore più alto rispetto al caso del canale in quanto il numero di celle si riferisce al diametro medio, quindi impostando un valore più elevato è possibile garantire un numero di celle sufficiente anche per sfere di diametro più piccolo. In seguito tramite l'utility `snappyHexMesh` sono state posizionate le sfere e la mesh è stata adattata agli oggetti mantenendo una griglia cartesiana.

4.2 Simulazioni effettuate e condizioni al contorno

In questa sezione sono riportate le proprietà dei fluidi utilizzati nelle simulazioni, eventuali modelli reologici adottati, i dettagli riguardanti i solver e le condizioni al contorno implementate in `OpenFOAM`.

4.2.1 Emulsioni alimentari

Una emulsione è ovviamente un sistema multifase costituito da una fase continua e una fase dispersa. In questo lavoro però il sistema multifase è stato descritto come un fluido monofasico, secondo l'ipotesi dello "pseudo-single-phase", tuttavia è stata implementata in `OpenFOAM` una legge reologica in modo da rappresentare in modo opportuno la reologia della emulsione. La legge reologica è stata ottenuta a partire da dati sperimentali [10] in letteratura:

$$\nu_r = \nu_{r,\infty} + K\dot{\gamma}^m, \tag{4.1}$$

dove ν_r è la viscosità relativa del fluido e per determinare la viscosità cinematica è sufficiente moltiplicare per la viscosità dell'acqua, assunta pari a 1 cP. I valori dei parametri della legge dipendono dalla frazione in peso di fase organica nell'emulsione: si è considerata un'emulsione al 70% in peso (72% in volume) di olio, in Tabella 4.1 sono riportati

i valori utilizzati. Data la densità della fase dispersa organica simulata (olio di semi di soia) pari a 917 kg m^{-3} [25], la frazione volumica, e la frazione in peso di olio è stato possibile valutare la densità dell'emulsione, che risulta essere pari a 943 (kg m^{-3}).

Parametro	**Valore**
$\nu_{r,\infty}$	5.9
K	1800
m	-0.66

Tabella 4.1: Parametri legge reologica per maionese al 70%$_{wt}$ in olio

Il solver di `OpenFOAM` impiegato per la risoluzione del campo di moto è `SRFSimpleFoam`, adatto per simulazioni stazionarie, fluidi incomprimibili, non Newtoniani, e soggetti a rotazione attorno a un singolo asse. Le equazioni del moto sono state risolte in regime laminare. La particolarità di questo solver risiede nel fatto che data una velocità di rotazione rispetto a un certo asse, l'asse dei due coni, tutte le condizioni al contorno sono fornite relativamente a tale rotazione. In base a ciò la parete interna ha associata una condizione di non scorrimento alla parete, in quanto si muove alla velocità di rotazione imposta, la parete esterna ha velocità nulla in quanto è ferma, in ingresso è imposta la velocità di ingresso del fluido e in uscita una condizione di gradiente nullo. Per quanto riguarda le condizioni al contorno della pressione è stato imposto un valore nullo in uscita dal cone mill e una condizione di gradiente nullo per la parete interna, esterna e per l'ingresso. Sia per la pressione sia per la velocità è stata impostata una tolleranza pari a 10^{-6}.

4.2.2 Mezzi porosi

Il fluido considerato per il flusso attraverso mezzi porosi è acqua a temperatura ambiente (298 K) con densità pari a 1000 (kg m^{-3}) e viscosità cinematica pari a 0.89 $\times 10^{-6}$ $m^2 s^{-1}$.

Il solver di `OpenFOAM` utilizzato per la risoluzione del campo di moto è `simpleFoam`, adatto per simulazioni stazionarie e fluidi incomprimibili, le equazioni del moto sono state risolte in regime laminare. In ingresso e in uscita dal mezzo sono imposti valori di pressione in modo tale che il profilo di pressione sia costante in ingresso ed in uscita, risultando in un profilo di velocità sviluppato. Per quanto riguarda le sfere è stata imposta una condizione di non scorrimento per la velocità e di gradiente nullo per la pressione. Sia per la pressione che per la velocità è stata impostata una tolleranza di 10^{-5}.

Il solver di `OpenFOAM` utilizzato per la risoluzione dell'equazione di advezione-diffusione è `scalarTransportFoam`, adatto per simulazioni transitorie e fluidi incomprimibili. Questo tipo di trasporto è detto scalare, in quanto la concentrazione è una grandezza scalare, e passivo, poiché le particelle di colloide non influenzano il campo di moto. É imposta una concentrazione di colloide unitaria in ingresso al mezzo, una condizione di gradiente nullo in uscita e una concentrazione nulla sui grani, non si considera quindi l'effetto di ingombro dovuto al colloide depositato e si suppone una deposizione irreversibile sui

grani, condizione nota come *clean-bed filtration* [26]. É stata impostata una tolleranza sulla concentrazione pari a 10^{-6}.

4.3 Impostazione del dataset

Come presentato nel capitolo 2, al fine di ottenere un modello data-driven è necessario stabilire quali siano le features caratteristiche del sistema per l'ottenimento di una certa predizione.

Nel caso delle simulazioni del cone mill per la produzione di emulsioni alimentari, in cui l'obiettivo è ottenere lo strain rate volumico, è stato scelto di utilizzare come feature in input al modello la distanza tra le due pareti del cone mill, la velocità di rotazione della parete interna e la velocità in ingresso del fluido. In Tabella 4.2 sono riportati i range di variazione delle feature entro cui sono state realizzate le simulazioni per l'ottenimento del dataset, gli estremi dei range esplorati sono stati determinati dalle condizioni cui sono state condotte le prove sperimentali [10].

Feature	**Range**
Gap	208 - 624 (μm)
Velocità di rotazione	3170 - 6784 (RPM)
Velocità in ingresso	0.047 - 0.460 (m s^{-1})

Tabella 4.2: Range di variazione delle feature per la creazione del dataset del caso emulsioni alimentari.

Per quanto riguarda le simulazioni del flusso attraverso mezzi porosi si distingue tra il caso del canale con due ostacoli ed il caso dei mezzi random. Per il primo caso è stato scelto di utilizzare come feature in input l'altezza del canale e il diametro delle sfere per la predizione della permeabilità, mentre per la predizione dell'efficienza di deposizione alle precedenti feature si aggiungono la pressione in ingresso e il diametro del colloide. Per i parametri geometrici sono stati scelti dei valori medi per l'altezza del canale (300 μm) e per il diametro dei grani (80 μm) e sono stati diminuiti e aumentati del 20% al fine di esplorare l'effetto della loro variazione. Dati questi valori medi è stata valutata la pressione in ingresso necessaria per avere una velocità media volumica pari a 0.0001 ms^{-1} così che, definito il numero di Reynolds:

$$\mathrm{Re} = \frac{\rho U d_g}{\mu}, \tag{4.2}$$

in cui si ricorda che d_g è il diametro dei grani, esso risulti essere sufficientemente inferiore a uno. Anche questo valore di pressione è stato aumentato e diminuito del 20% per l'ottenimento del range operativo. In Tabella 4.3 sono riportati i range di variazione delle feature entro cui sono state realizzate le simulazioni.

Nel caso dei mezzi random, per la determinazione della permeabilità sono stati utilizzati il diametro medio dei grani, l'eventuale deviazione standard, e la porosità del

mezzo; per la predizione dell'efficienza di deposizione alle precedenti feature si aggiungono la pressione in ingresso e il diametro del colloide. La scelta del valore medio della pressione in ingresso è stata effettuata analogamente al caso precedente, il valore degli altri parametri è stato imposto in base alle grandezze caratteristiche di questi mezzi. In Tabella 4.4 sono riportati i range di variazione delle feature entro cui sono state realizzate le simulazioni.

Feature	**Range**
Altezza canale	240 - 360 (μm)
Diametro grani	64 - 96 (μm)
Pressione in ingresso	7.0×10^{-6} - 10.5×10^{-6} (m^2s^{-2})
Diametro del colloide	30 - 100 (nm)

Tabella 4.3: Range di variazione delle feature per la creazione del dataset del caso canale con ostacoli.

Feature	**Range**
Diametro medio grani	100 - 200 (μm)
Deviazione standard	0.1 - 0.3
Porosità	0.5 - 0.65
Pressione in ingresso	3.3×10^{-4} - 5×10^{-4} (m^2s^{-2})
Diametro del colloide	30 - 100 (nm)

Tabella 4.4: Range di variazione delle feature per la creazione del dataset del caso mezzi random.

5 | Presentazione risultati

In questo capitolo sono dapprima presentati i risultati delle simulazioni di fluidodinamica computazionale eseguite su `OpenFOAM` per i due casi studio, per i casi base di trasporto attraverso mezzi porosi e i casi con le stesse condizioni operative degli esperimenti disponibili per la produzione di emulsioni alimentari (sezione 5.1). In seguito sono analizzate le prestazioni dei modelli data-driven (sezione 5.2)

5.1 Simulazioni di fluidodinamica computazionale

In questa sezione sono riportate le soluzioni principali delle simulazioni di fluidodinamica computazionale per le condizioni sperimentali del caso della produzione di emulsioni alimentari e per i casi base del flusso attraverso mezzi porosi.

5.1.1 Emulsioni alimentari

Le condizioni sperimentali presentate in uno studio in letteratura [10] sono riassunte nella Tabella 5.1, mentre nella Tabella 5.2 sono riportati i risultati della simulazione: la velocità media volumica e la viscosità cinematica media volumica, che hanno permesso di calcolare il numero di Reynolds. Per il caso in esame il numero di Reynolds è definito nel seguente modo:

$$\mathrm{Re} = \frac{\rho \Omega R_{in}}{\nu}, \tag{5.1}$$

dove Ω è la velocità di rotazione in (rad s^{-1}), R_{in} è il maggiore raggio del cono rotante.

Esperimento	**Velocità di rotazione (RPM)**	**Gap (mm)**	**Portata massica (kg s^{-1})**	**Re**
1	6039	0.624	31	138.50
2	6784	0.208	15	60.51
3	3170	0.624	64	63.90
4	6039	0.624	15	138.52
5	3170	0.208	48	26.01

Tabella 5.1: Condizioni operative delle prove sperimentali [10]

Esperimento	**Velocità di rotazione (RPM)**	**Gap (mm)**	**Viscosità cinematica media volumica (m^2s^{-1})**	**Re**
1	6039	0.624	8.5410^{-5}	138.50
2	6784	0.208	7.3210^{-5}	60.51
3	3170	0.624	9.7210^{-5}	63.90
4	6039	0.624	8.5410^{-5}	138.52
5	3170	0.208	7.9610^{-5}	26.01

Tabella 5.2: Risultati delle simulazioni di fluidodinamica computazionale per la valutazione del numero di Reynolds

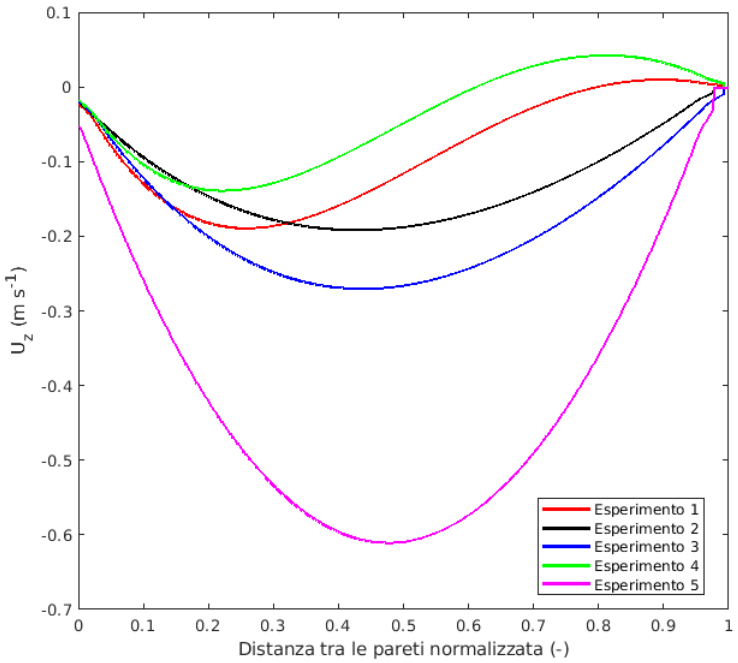
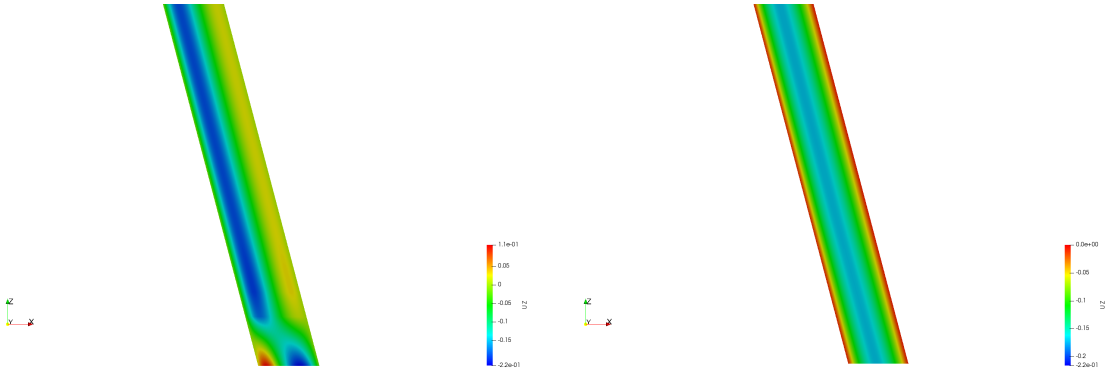
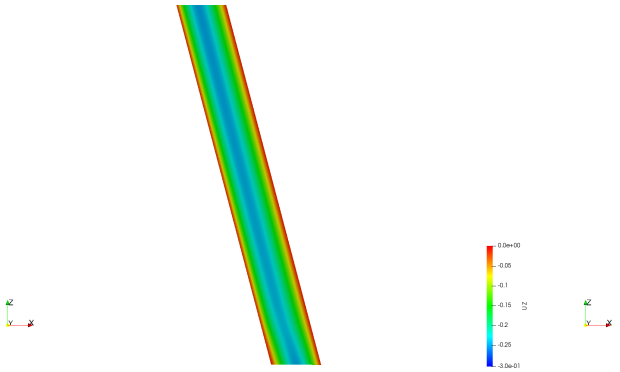


Figura 5.1: Profilo velocità z per gli esperimenti

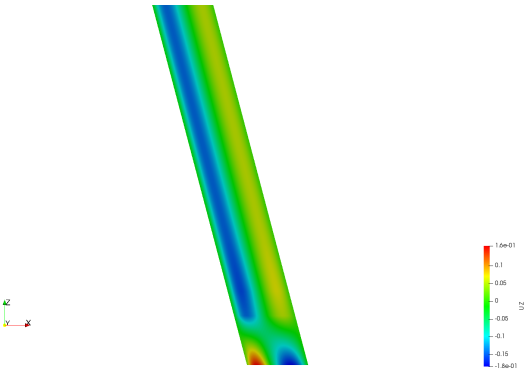


(a) Esperimento 1

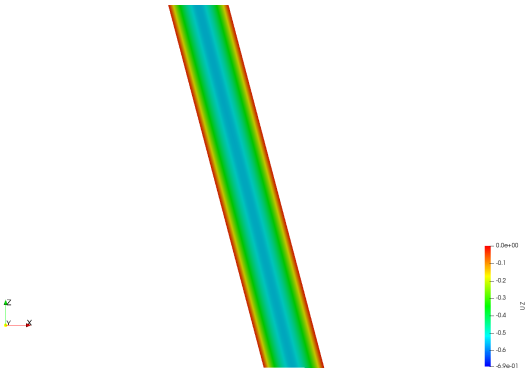
(b) Esperimento 2



(c) Esperimento 3



(d) Esperimento 4



(e) Esperimento 5

Figura 5.2: Contour plot della componente z della velocità nella parte terminale della geometria per le diverse prove sperimentli

É possibile notare che l'esperimento numero 1 e il numero 4 presentano un numero di Reynolds decisamente più alto rispetto agli altri, questa caratteristica si riflette nelle soluzioni del campo di moto, in Figura 5.2 sono riportati i *contour plot* per la componente z della velocità del fluido. Nella seconda, terza e quinta prova sperimentale il fluido scorre sempre nella direzione opposta dell'asse z (assiale) e i profili della componente z della velocità valutati nel gap a metà altezza della geometria sono parabolici, Figure 5.2(a), 5.2(c) e 5.2(d). Il flusso è quindi completamente in regime laminare e stabile. Il flusso nel primo e quarto esperimento è invece caratterizzato da un grande vortice che interessa quasi tutta la lunghezza del cone mill e un vortice di senso opposto all'uscita della geometria. L'andamento della componente z della velocità, valutato nel gap a metà altezza, Figura 5.2(b) e 5.2(e), non è più parabolico ma assume valori negativi in prossimità del rotore per poi diventare positivi verso la parete fissa a causa della presenza del vortice. Quindi, sebbene il flusso sia in regime laminare, sono presenti delle instabilità. I vortici che causano le instabilità sono detti vortici di Taylor e sono caratteristici di sistemi analoghi a quello studiato, come ad esempio la cella di Taylor-Couette. In questo dispositivo il fluido scorre in modo analogo al cone mill ma la geometria è costituita da due cilindri concentrici. Al variare del rapporto tra il raggio del cilindro esterno ed interno è possibile individuare il valore del numero di Reynolds critico a cui corrisponde l'inizio delle instabilità. In particolare si formano dei vortici di Taylor circolari con centro sull'asse comune dei due cilindri [27]. Geometrie analoghe a quelle realizzate sono state studiate da *Xue et al.* simulando un flusso attraverso dei tronchi di cono coassiali a pendenza positiva e negativa, in particolare è stato analizzato come varia la configurazione dei vortici al variare del numero di Reynolds. I risultati delle simulazioni sono in accordo con quanto riportato in letteratura, infatti, è stato possibile osservare la presenza di vortici di Taylor soltanto per i sistemi aventi un numero di Reynolds superiore a 75 [28].

5.1.2 Mezzi porosi

In Figura 5.3 è riportato il campo di moto del fluido che scorre nel canale attorno alle sfere. Si può notare che la velocità diminuisce fino a diventare nulla sulla superficie delle sfere, sulla faccia superiore ed inferiore si può verificare la condizione di simmetria impostata come condizione al contorno.

In Figura 5.4 sono riportati i contour plot della concentrazione nel dominio spaziale: all'inizio la concentrazione del colloide è nulla, Figura 5.4(a), in seguito è possibile osservare come varia la concentrazione, Figure 5.4(b) 5.4(c), fino al raggiungimento di uno stato stazionario, Figura 5.4(d). É possibile verificare che la concentrazione rimane nulla sulle sfere per poi crescere all'aumentare della distanza da esse.

Per quanto riguarda, invece, i mezzi porosi random, al fine di effettuare un confronto fra le diverse geometrie costruite è stato scelto di riportare i contour plot della velocità, in Figura 5.5, a parità di condizioni operative e caratteristiche geometriche macroscopiche. La differenza principale che si nota tra le geometrie non periodiche, Figure 5.5(a) e 5.5(b), e le geometrie periodiche, Figure 5.5(c) e 5.5(d), è la localizzazione dei picchi di velocità. Infatti, si nota come la velocità sia massima in corrispondenza dei contorni

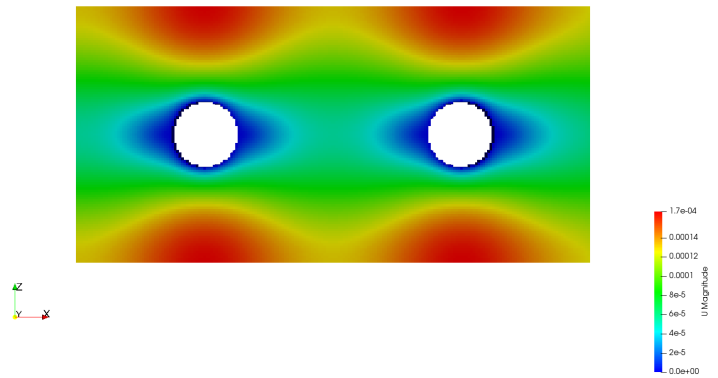
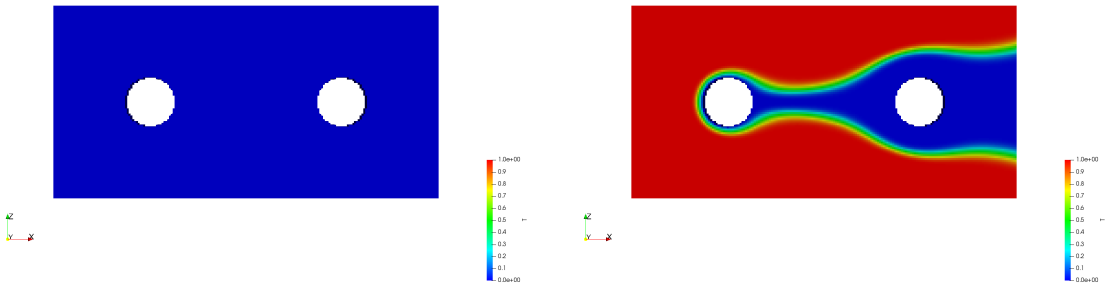
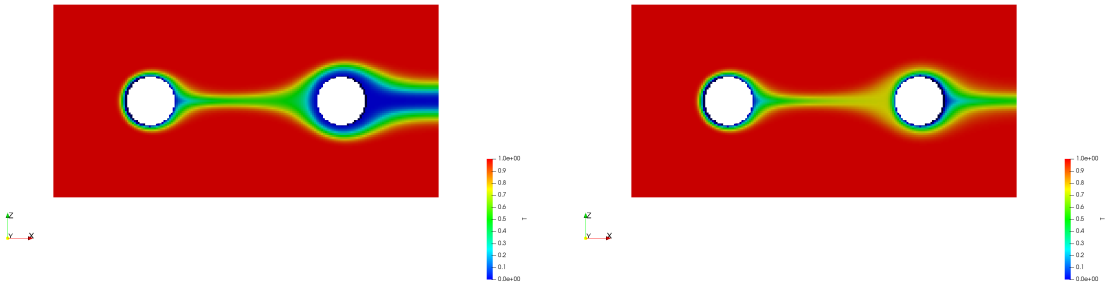


Figura 5.3: Contour plot della velocità per il caso del flusso attraverso un canale con ostacoli, Re = 0.0087.



(a) t = 0s

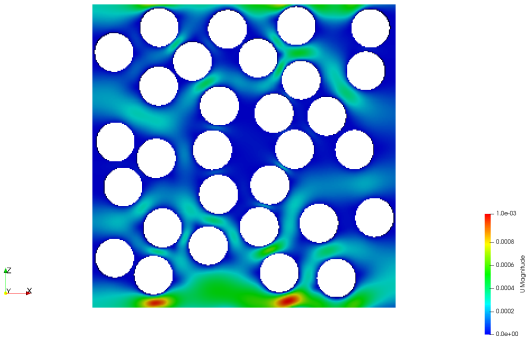
(b) t = 5s



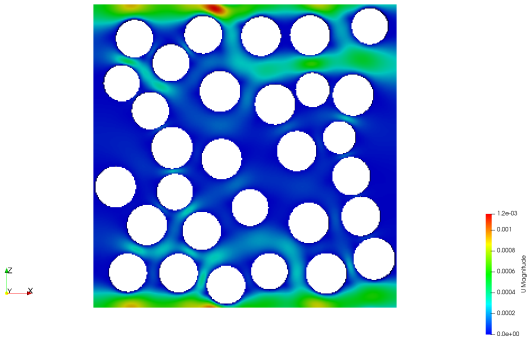
(c) t = 10s

(d) t = 20s

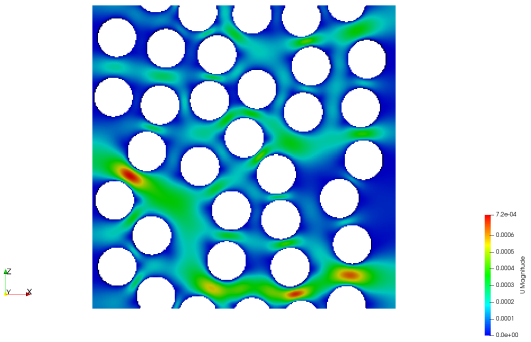
Figura 5.4: Contour plot della concentrazione per tempi crescenti, fino al raggiungimento dello stazionario, Sc = 70764, Pe = 615.



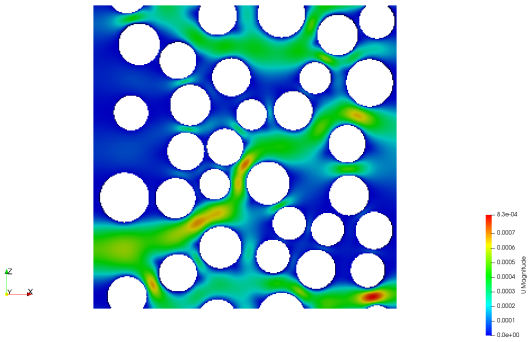
(a) Geometria monodispersa, Re=0.017



(b) Geometria polidispersa, Re=0.017

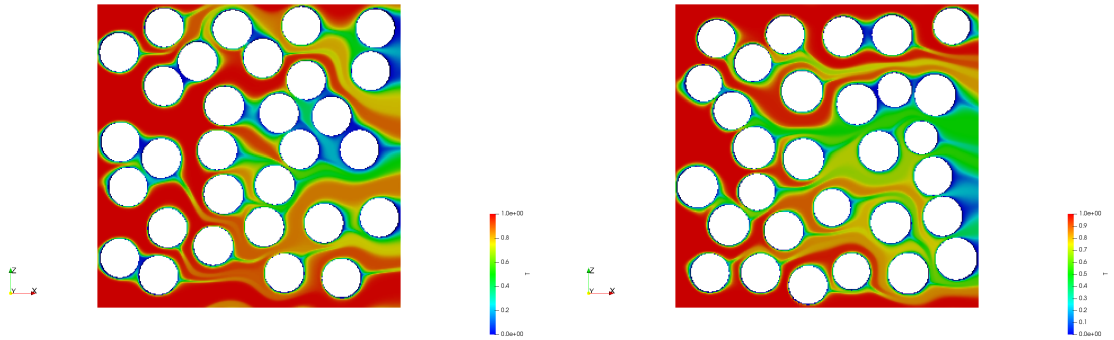


(c) Geometria monodispersa periodica, Re=0.014



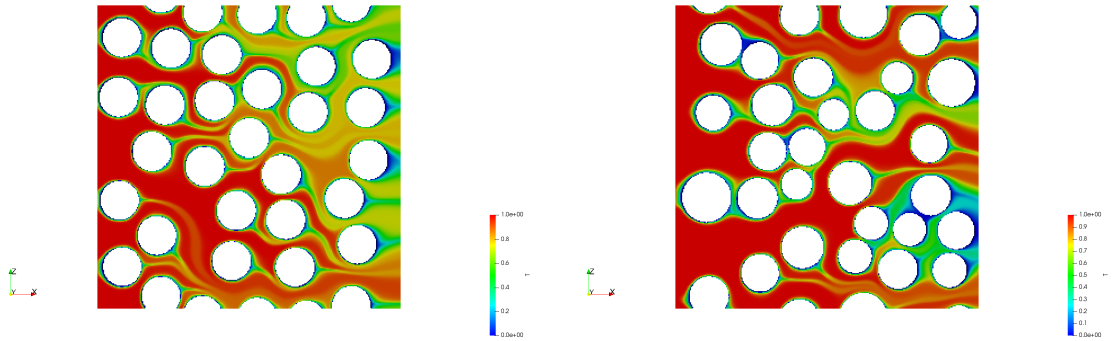
(d) Geometria polidispersa periodica, Re=0.019

Figura 5.5: Contour plot della velocità per le diverse geometrie proposte.



(a) Geometria monodispersa, Sc=87095, Pe=1449.

(b) Geometria polidispersa, Sc=87095, Pe=1523.



(c) Geometria monodispersa periodica, Sc=87095, Pe=1253.

(d) Geometria polidispersa periodica, Sc=87095, Pe=1629.

Figura 5.6: Contour plot della concentrazione del colloide per le diverse geometrie proposte.

preferenziali non fisici per il flusso.

Per questo motivo è stato creato un codice per la realizzazione di geometrie random periodiche: infatti, in questo caso la porosità locale è uniforme, inibendo il fenomeno di canalizzazione del flusso, e il sistema si comporta in modo più simile a quanto accada nella realtà. In particolare i sistemi più realistici risultano essere i mezzi porosi random periodici, per quanto appena descritto, e polidispersi, poiché raffigurano al meglio la natura del materiale costituente il mezzo, ad esempio la sabbia.

In Figura 5.6 sono riportati i contour plot della concentrazione del colloide per le diverse geometrie allo stato stazionario. Le stesse osservazioni riportate per il caso del canale possono essere verificate nei casi random. Per quanto riguarda le condizioni al contorno imposte sulla faccia superiore ed inferiore, per le geometrie non periodiche il profilo di concentrazione è simmetrico, Figure 5.6(a) e 5.6(b), per le geometrie non periodiche è, invece, ciclico, Figure 5.6(c) e 5.6(d). I numeri adimensionali di Peclet e di Schmidt sono definiti come segue:

$$\mathrm{Pe} = \frac{q d_g}{\mathcal{D}}, \quad \mathrm{Sc} = \frac{\nu}{\mathcal{D}}. \tag{5.2}$$

5.2 Prestazioni modello surrogato

In questa sezione verranno presentate le prestazioni delle reti neurali per la predizione della velocità di deformazione, per il caso della produzione dell'emulsione, e della permeabilità ed efficienza di deposizione, per il caso del flusso attraverso mezzi porosi, a partire dalle feature descritte nel capitolo precedente.

5.2.1 Emulsioni alimentari

La velocità di rotazione della parete interna, la velocità in ingresso del fluido e la distanza tra i coni sono le feature utilizzate dalla rete neurale per predire lo strain rate medio volumico. A tal fine è stata progettata ed allenata una rete neurale contenente un hidden layer costituito da 10 neuroni. La performance della rete è stata valutata considerando per ciascun sample l'errore percentuale della predizione (NN) rispetto al risultato della relativa simulazione di fluidodinamica computazionale (CFD) svolte con le stesse condizioni operative:

$$\text{errore} = \text{ass}\left(\frac{\mathrm{NN} - \mathrm{CFD}}{\mathrm{CFD}}\right). \tag{5.3}$$

Nel caso in esame, per valutare la performance, sono stati dati dapprima in input alla rete dei nuovi sample, non presenti nel dataset di training, in cui sono mantenuti fissi i valori della velocità in ingresso e del gap, e viene variata la velocità di rotazione in un range più ampio di quello di variazione dei parametri di training, Figura 5.7. É possibile osservare che l'errore è pressoché nullo all'interno del range, mentre al di fuori aumenta in modo più che lineare, tuttavia fino a +20% degli RPM l'errore si mantiene al di sotto del 3%.

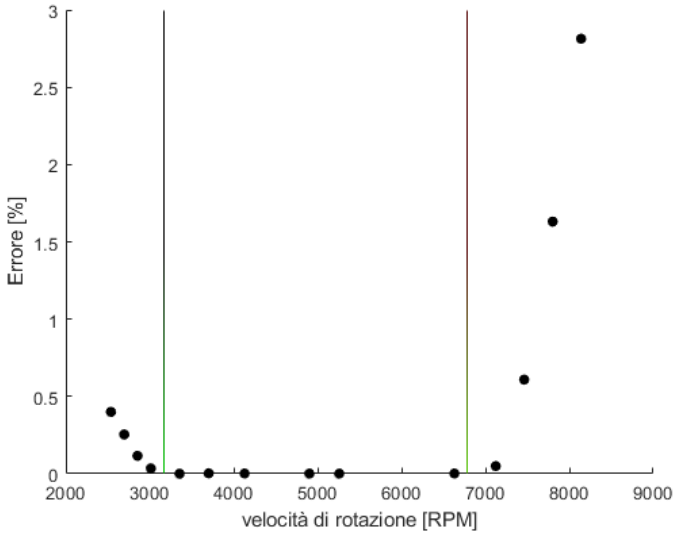


Figura 5.7: Errore associato alla predizione della velocità di deformazione per diversi valori della velocità di rotazione all'interno e all'esterno del range, rete neurale contenente 10 neuroni disposti in un hidden layer

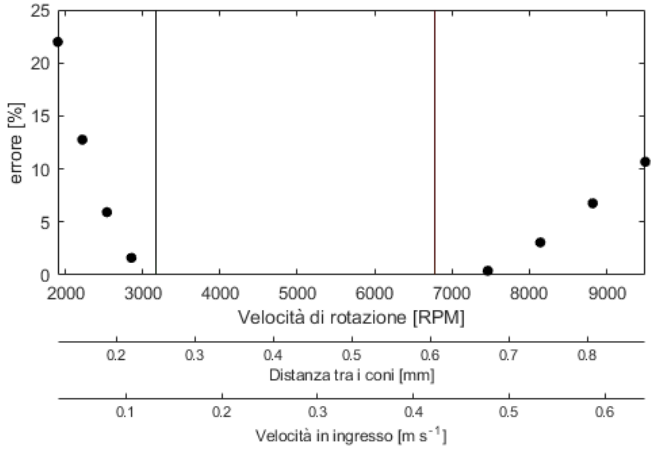


Figura 5.8: Errore associato alla predizione della velocità di deformazione per valori delle feature al di fuori del range, rete neurale contenente 10 neuroni disposti in un hidden layer

Nella seconda prova effettuata, in Figura 5.8, tutte le feature sono state variate in un range più ampio del range di training, e gli errori aumentano in modo più che lineare all'aumentare della distanza dai limiti del range, tuttavia gli errori risultano essere più elevati del caso precedente in quanto tutte le feature sono state variate contemporaneamente al di fuori dei limiti.

5.2.2 Mezzi porosi

Nel caso del flusso attraverso un canale contenente due ostacoli sferici la performance della rete è stata provata con 25 nuovi sample, non presenti nel dataset di training, ed è stata valutata la qualità della predizione della permeabilità e dell'efficienza di deposizione attraverso dei diagrammi di parità, Figura 5.9. É possibile osservare che la rete predice in modo ottimale sia la permeabilità, Figura 5.9(a), che l'efficienza di deposizione, Figura 5.9(b), con errori per entrambi i casi inferiori all'1%.

Viste le buone prestazioni, la rete è stato sottoposta anche a dei sample caratterizzati da feature al di fuori del range. Per quanto riguarda la permeabilità, l'altezza del canale e il diametro dei grani sono stati variati dapprima in modo concorde, Figura 5.10(a), poi in modo discorde, Figura 5.10(b). Per entrambi i casi gli errori, valutati come in eq. 5.3, aumentano all'incrementare della distanza dai limiti del range ed in entrambi i casi la loro entità è simile. Anche nel caso dell'efficienza di deposizione gli errori aumentano all'incrementare della distanza dai limiti del range, ma l'entità dell'errore è maggiore, Figura 5.10(c).

Poiché l'allenamento della rete neurale non è un procedimento deterministico, eseguendo il training più volte è possibile ottenere risultati di predizione diversi. Per questo motivo, dato il dataset, è stato eseguito il training della rete neurale per 10 volte. In seguito a ciascun allenamento, dati i sample aggiuntivi, sono state effettuate le predizioni e sono state salvate in una matrice avente 10 colonne e un numero di righe pari al numero di sample. Data la matrice, sono stati valutati gli errori medi per ciascuna predizione costruendo un vettore in cui ciascuna componente contiene la media dei valori della riga corrispondente della matrice. Per ciascuna riga della matrice è stata, inoltre, calcolata la deviazione standard, così da poter costruire le barre di errore presenti nei grafici.

Per il caso dei mezzi random sono stati prodotti, in maniera analoga al caso precedente, dei diagrammi di parità per valutare le prestazioni delle reti neurali nella predizione della permeabilità, Figura 5.11, e dell'efficienza di deposizione, Figura 5.12. Al fine di paragonare in modo equo le diverse geometrie il training è stato effettuato con una grandezza identica del dataset e lo stesso numero di samples aggiuntivi per il test finale. In Tabella 5.3 sono riportate queste informazioni e gli errori medi di predizione ottenuti.

In primo luogo è possibile osservare che per tutti i tipi di geometria l'errore associato a ciascun sample aggiuntivo non è trascurabile a differenza del caso del flusso attraverso un canale. In secondo luogo è possibile effettuare i seguenti confronti tra le geometrie proposte:

- la predizione della permeabilità è migliore per le geometrie monodisperse (Figure 5.11(a) e 5.11(b)) rispetto a quelle polidisperse (Figure 5.11(c) e 5.11(d)) in quanto le prime sono più semplici delle seconde, infatti, posseggono una feature in meno

Geometria	**Numero di sample nel dataset**	**Numero di sample aggiuntivi**	**Errore permeabilità .** [%]	**Errore efficienza di deposizione** [%]
Monodispersa	805	30	21.9	15.7
Polidispersa	805	29	30.9	14.1
Monodispersa periodica	805	35	20.2	16.9
Polidispersa periodica	805	30	28.8	17.1

Tabella 5.3: Riassunto delle prestazioni delle reti neurali

(i.e. deviazione standard della distribuzione dei diametri degli ostacoli) in ingresso alla rete;

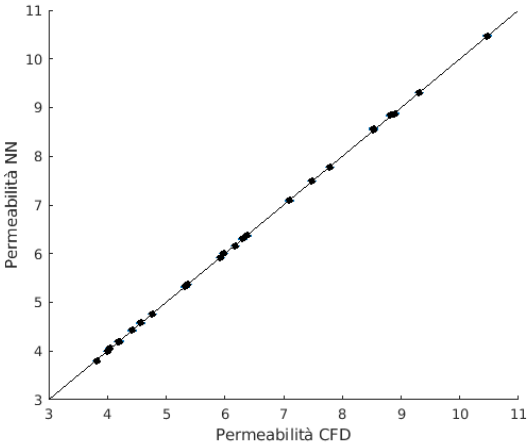
- la predizione della permeabilità è migliore per le geometrie periodiche (Figure 5.11(c) e 5.11(d));

- la predizione dell'efficienza di deposizione appare essere meno influenzata dal tipo di geometria, tuttavia è migliore per le geometrie non periodiche (Figure 5.12(a) e 5.12(b))

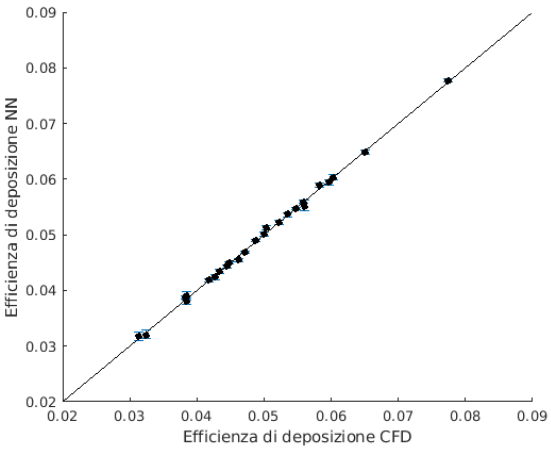
Sono state, inoltre, testate diverse architetture di reti neurali per valutare la loro influenza sulla qualità della predizione. A titolo di esempio sono stati riportati in Tabella 5.4 gli errori medi sui sample aggiuntivi per geometrie random polidisperse periodiche. Il numero di componenti del vettore rappresenta il numero di hidden layer mentre il loro valore il numero di neuroni per ciascun layer. É possibile concludere che l'architettura della rete influenza gli errori medi di predizione, fino a 50 neuroni gli errori non variano eccessivamente, mentre quando si aumentano i neuroni gli errori aumentano notevolmente a causa dell'*overfitting*. Poiché gli errori associati alle predizioni della permeabilità e dell'efficienza di deposizione non sono trascurabili, è stato aumentato il numero di sample presenti nel dataset di training per mezzi porosi random polidispersi periodici in modo da valutare la diminuzione dell'errore al variare dell'ampiezza del dataset, Figura 5.13. Sebbene l'andamento dell'errore sia decrescente, è possibile osservare che la sua diminuzione sia minima per entrambi gli obiettivi di predizione, ed il semplice aumento del dataset di training non costituisce un'opzione viabile per l'incremento di accuratezza del metodo, e si renderà necessario l'utilizzo di tecniche più sofisticate.

Architettura	**Errore permeabilità** [%]	**Errore efficienza di deposizione** [%]
[5]	24.9	15.9
[10]	28.8	17.1
[5 5]	28.3	15.3
[20]	26.8	16.2
[10 10]	27.2	16.4
[15 10 5]	31.4	17.4
[50]	28.9	18.4
[100]	34.5	21.1
[50 50]	35.6	23.1

Tabella 5.4: Errori sulla predizione della permeabilità e dell'efficienza di deposizione per mezzi porosi random polidispersi periodici per diverse architetture di reti neurali

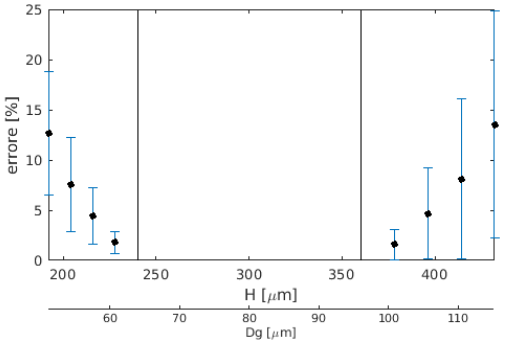


(a) Predizione della permeabilità, errore medio 0.19%

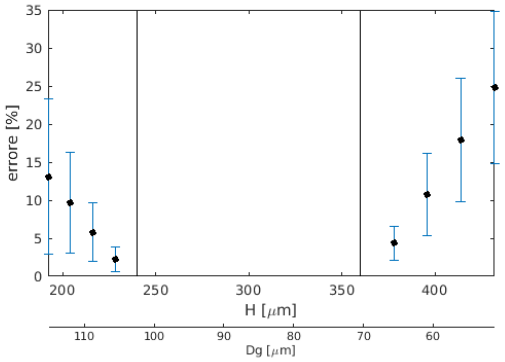


(b) Predizione dell'efficienza di rimozione, errore medio 0.83%

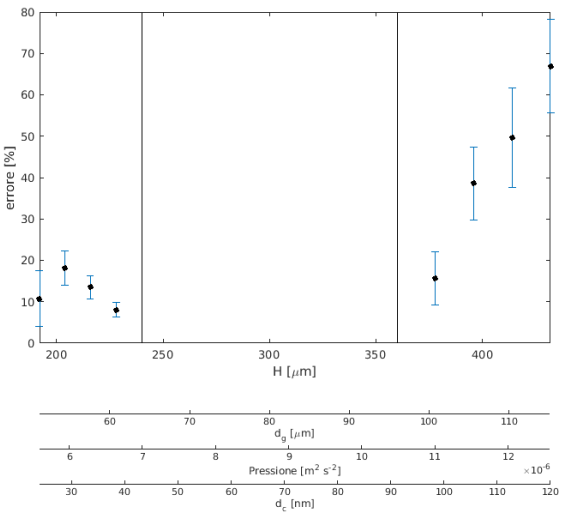
Figura 5.9: Diagramma di parità per il caso flusso attraverso canale su 25 nuovi casi, rete neurale contenente 10 neuroni disposti in un hidden layer



(a) Variazione dell'altezza del canale e del diametro dei grani concorde per la predizione della permeabilità



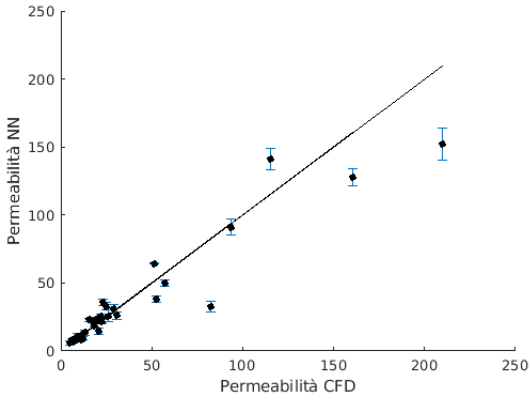
(b) Variazione dell'altezza del canale e del diametro dei grani discorde per la predizione della permeabilità



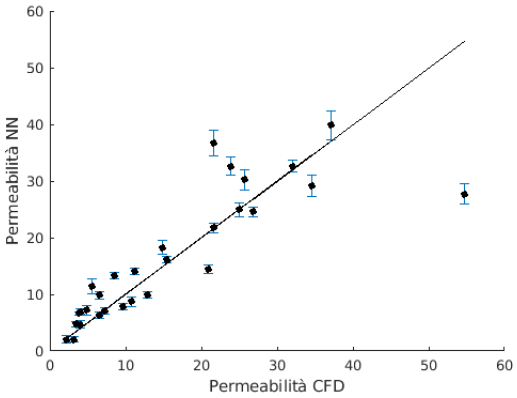
(c) Variazione dell'altezza del canale, del diametro dei grani, della pressione in ingresso e del diametro del colloide per la predizione dell'efficienza di deposizione

Figura 5.10: Errore associato alla predizione della permeabilità e dell'efficienza di deposizione nel caso del canale per valori delle feature al di fuori del range, rete neurale contenente 10 neuroni disposti in un hidden layer. Raffigurato: il valore medio degli errori (punto nero) e valore della deviazione standard (barra di errore)

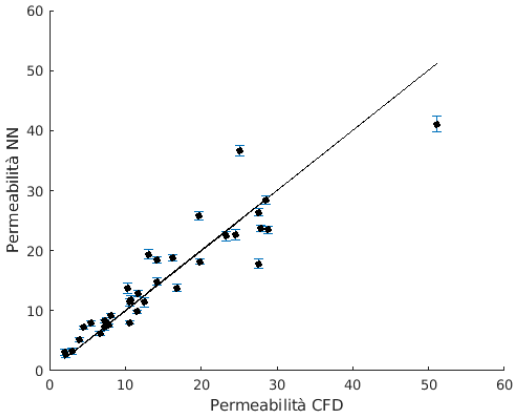
.



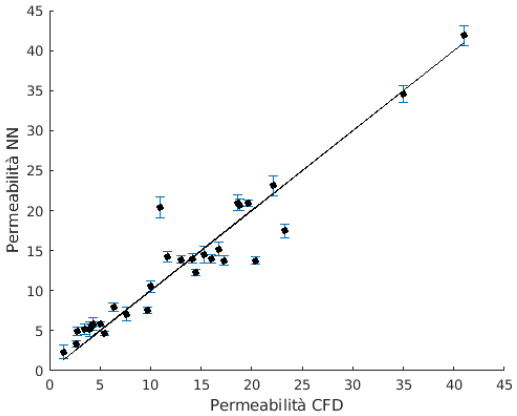
(a) Geometria monodispersa, errore medio 21.9%



(b) Geometria polidispersa, errore medio 30.9%

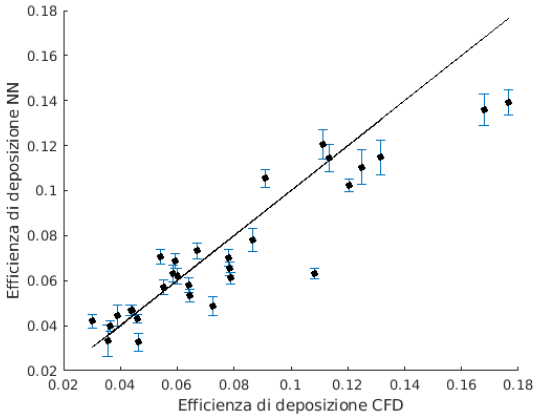


(c) Geometria monodispersa periodica, errore medio 20.2%

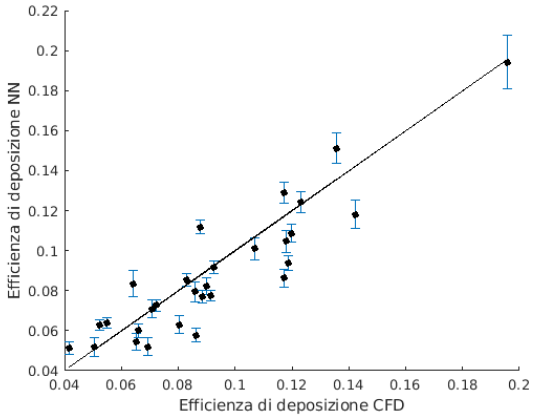


(d) Geometria polidispersa periodica, errore medio 23.4%

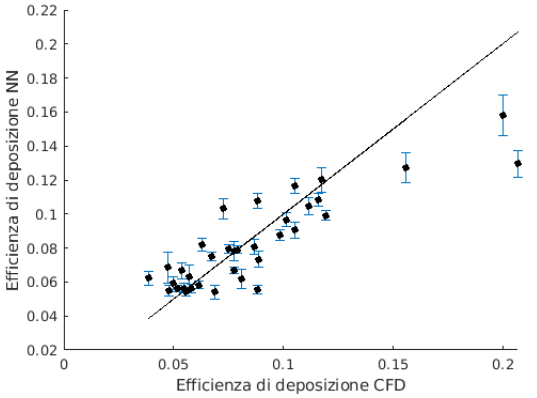
Figura 5.11: Diagramma di parità della permeabilità per il caso flusso attraverso un mezzo poroso su 25 nuovi casi, rete neurale contenente 10 neuroni disposti in un hidden layer. Raffigurato: il valore medio degli errori (punto nero) e valore della deviazione standard (barra di errore).



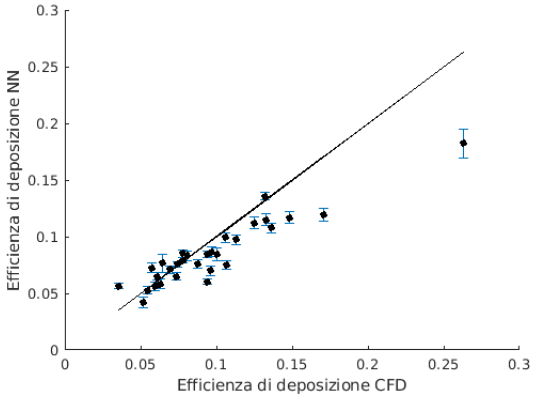
(a) Geometria monodispersa, errore medio 15.7%



(b) Geometria polidispersa, errore medio 14.1%

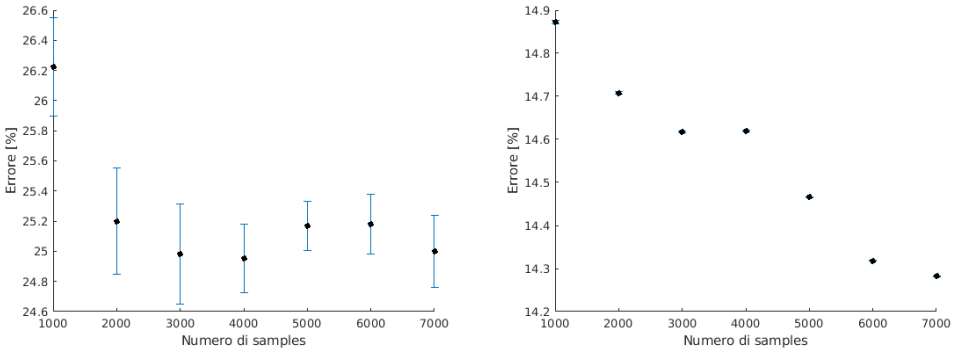


(c) Geometria monodispersa periodica, errore medio 16.9%



(d) Geometria polidispersa periodica, errore medio 16.1%

Figura 5.12: Diagramma di parità dell'efficienza di deposizione per il caso flusso attraverso canale su 25 nuovi casi, rete neurale contenente 10 neuroni disposti in un hidden layer. Raffigurato: il valore medio degli errori (punto nero) e valore della deviazione standard (barra di errore).



(a) Permeabilità (b) Efficienza di deposizione

Figura 5.13: Andamento dell'errore medio all'aumentare della dimensione del dataset di training della rete neurale (i.e.: numero di sample)

6 | Conclusioni

L'obiettivo del presente lavoro è stato valutare l'applicazione di tecniche di machine learning, in particolare di reti neurali, per la predizione di risultati tipici di simulazioni di fluidodinamica computazionale. Dato il caso studio è stato dapprima necessario preparare un dataset per eseguire il training delle reti, a tal fine è stato sviluppato un workflow i cui passaggi fondamentali sono:

- definizione degli obiettivi di predizione e delle feature causali da sfruttare a tal scopo;
- preparazione della cartella template di `OpenFOAM` impostata opportunamente (condizioni al contorno, geometria, discretizzazione spaziale, proprietà fisiche del fluido, solver, criteri di convergenza);
- creazione delle cartelle di simulazione impostate in modo che il valore delle feature scelte copra un range operativo utile per gli scopi applicativi del modello finale;
- risoluzione delle simulazioni;
- controllo ed estrazione dei risultati delle simulazioni.

Questo workflow è stato dapprima applicato alla produzione di emulsioni alimentari in cui è stato predetto lo strain rate volumico a partire da specifiche condizioni operative (velocità in ingresso del fluido e velocità di rotazione della parete interna) e parametri geometrici (distanza tra rotore e statore). Le prestazioni della rete hanno errori associati pressoché nulli all'interno del range di variazione delle feature scelte, mentre questi errori diventano apprezzabili quando si utilizza il modello data-driven come strumento di estrapolazione, cercando cioè predizioni di strain rate associate a set di parametri in ingresso al di fuori del range di training originale, ed aumentano più che linearmente allontanandosi da questi limiti.

Il secondo caso studio a cui è stato applicato il workflow è il flusso e il trasporto di un colloide attraverso mezzi porosi, in cui sono stati predetti la permeabilità del mezzo e l'efficienza di deposizione del colloide a partire da condizioni operative (pressione in ingresso e diametro del colloide) e parametri geometrici. Nel caso del flusso attraverso un canale con ostacoli regolari gli errori della rete neurale sono inferiori all'1% all'interno del range di variazione delle feature, al di fuori del range gli errori aumentano all'aumentare delle distanza dai limiti del range. Nel caso del flusso attraverso mezzi random sono state create delle geometrie costituite da 30 sfere disposte a formare geometrie polidisperse o

monodisperse ed eventualmente periodiche. Le predizioni delle reti in questo caso si sono rivelate problematiche anche all'interno del range di variazione delle feature con errori dell'ordine del 20-30% per la permeabilità e del 14-18% per l'efficienza di deposizione.

Questi risultati evidenziano come semplici reti neurali *"fully-connected"* (i.e.: percettroni multi-layer) siano sufficienti per la costruzione di modelli data-driven di elevata accuratezza e minimo costo computazionale basati su modelli CFD con un limitato numero di parametri, nel caso invece di strutture fluidodinamiche più complesse (e.g.: flusso in mezzi dispersi) questa soluzione risulta non ottimale ed è necessario l'utilizzo di reti neurali più sofisticate.

A | Algoritmo di retropropagazione

Di seguito è presentata la derivazione completa dell'algoritmo di retropropagazione. In questa derivazione si considera il caso più completo in cui il target della rete neurale sia un vettore. Per ogni sample del dataset è dapprima necessario valutare la predizione della rete neurale:

$$\mathbf{z}^0 = \mathbf{x} \tag{A.1}$$

$$\mathbf{z}^{m+1} = f^{m+1}(\mathbf{W}^{m+1}\mathbf{x^m} + \mathbf{b}^{m+1}) \text{ per m = 0, 1,..., M - 1} \tag{A.2}$$

$$\mathbf{z} = \mathbf{x}^M \tag{A.3}$$

In cui z^m è l'output del layer m, x^m è l'input del layer m, $\mathbf{W}^m$ è la matrice dei pesi relativi al layer m e $\mathbf{b}^m$ è il vettore contenente i bias del layer m. Poiché l'obiettivo dell'algoritmo di retropropagazione è quello di determinare il valore dei pesi e dei bias in modo da minimizzare l'errore di predizione della rete si definisce la funzione errore, cioè il mean square error:

$$F(\mathbf{p}) = (\mathbf{t}(k) - \mathbf{z}(k))^T(\mathbf{t}(k) - \mathbf{z}(k)) = \mathbf{e}^T(k)\mathbf{e}(k) \tag{A.4}$$

dove $\mathbf{p}$ è il vettore contente i pesi e i bias e $\mathbf{t}(k)$ è il vettore dei target per un determinato sample k. Il più veloce, quindi il più "ripido", algoritmo di discesa per il mean square error (stochastic gradient descent) è:

$$\begin{aligned} w^m_{i,j}(k+1) &= w^m_{i,j}(k) - \lambda\frac{\partial F}{\partial w^m_{i,j}} \\ b^m_i(k+1) &= b^m_i(k) - \lambda\frac{\partial F}{\partial b^m_i} \end{aligned} \tag{A.5}$$

É necessario quindi determinare le derivate parziali della funzione errore rispetto ai pesi e ai bias, a tal fine si applica la regola della catena:

$$\begin{aligned} \frac{\partial F}{\partial w^m_{i,j}} &= \frac{\partial F}{\partial n^m_i} \times \frac{\partial n^m_i}{\partial w^m_{i,j}} \\ \frac{\partial F}{\partial b^m_i} &= \frac{\partial F}{\partial n^m_i} \times \frac{\partial n^m_i}{\partial b^m_i} \end{aligned} \tag{A.6}$$

si è introdotta dunque la funzione n_i che rappresenta l'input netto al neurone i-esimo dovuto ai contributi dei neuroni j-esimi del layer precedente:

$$n^m_i = \sum_{j=1}^{S^{m-1}} w^m_{i,j} z^{m-1}_j + b^m_i \tag{A.7}$$

dove S^m è il numero di neuroni del layer m. É possibile sviluppare le derivate presenti nell'equazione A.5:

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = z_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1 \tag{A.8}$$

definiamo la sensitività come la derivata parziale della funzione errore fatta rispetto alla funzione n_i, quindi è indice dell'impatto dei pesi relativi al neurone i-esimo sull'errore complessivo:

$$s_i^m \equiv \frac{\partial F}{\partial n_i^m} \tag{A.9}$$

Di conseguenza, i pesi possono essere aggiornati nel seguente modo:

$$\begin{aligned} w_{i,j}^m(k+1) &= w_{i,j}^m(k) - \lambda s_i^m z_j^{m-1} \\ b_i^m(k+1) &= b_i^m(k) - \lambda s_i^m \end{aligned} \tag{A.10}$$

in forma vettoriale:

$$\begin{aligned} \mathbf{W}^m(k+1) &= \mathbf{W}^m(k) - \lambda \mathbf{s}^m \left(\mathbf{z}^{m-1}\right)^T \\ \mathbf{b}^m(k+1) &= \mathbf{b}^m(k) - \lambda \mathbf{s}^m \end{aligned} \tag{A.11}$$

É necessario ora determinare il vettore delle sensitività, a tal fine si sfrutta nuovamente la regola della catena:

$$\mathbf{s}^m = \frac{\partial F}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial F}{\partial \mathbf{n}^{m+1}} \tag{A.12}$$

dove $\left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)$ è una matrice Jacobiana del seguente tipo:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{s^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{s^m}^m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix} \tag{A.13}$$

il suo elemento i-esimo può essere espresso come:

$$\begin{aligned} \frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} z_l^m + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial z_j^m}{\partial n_j^m} = \\ = w_{i,j}^{m+1} \frac{\partial f^m\left(n_j^m\right)}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m\left(n_j^m\right) \end{aligned} \tag{A.14}$$

allora la matrice Jacobana può essere rielaborata nella forma seguente:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \dot{\mathbf{F}}^m\left(\mathbf{n}^m\right) \tag{A.15}$$

dove $\dot{\mathbf{F}}$ è una matrice diagonale del tipo:

$$\dot{\mathbf{F}}^m\left(\mathbf{n}^m\right) = \begin{bmatrix} \dot{f}^m\left(n_1^m\right) & 0 & \dots & 0 \\ 0 & \dot{f}^m\left(n_2^m\right) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dot{f}^m\left(n_{S^m}^m\right) \end{bmatrix} \tag{A.16}$$

la sensitività A.12 può essere aggiornata in base ai risultati raggiunti:

$$\mathbf{s}^m = \dot{\mathbf{F}}^m\left(\mathbf{n}^m\right)\left(\mathbf{W}^{m+1}\right)^T \frac{\partial F}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m\left(\mathbf{n}^m\right)\left(\mathbf{W}^{m+1}\right)^T \mathbf{s}^{m+1} \tag{A.17}$$

É stato dimostrato che la sensitività del layer m-esimo dipende dalla sensitività del layer successivo, proprio per questo motivo l'algoritmo è detto di retropropagazione, quindi è necessario determinare la sensitività dell'ultimo layer M per determinare di conseguenza tutte le altre ($\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \ldots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1$):

$$\begin{aligned} s_i^M = \frac{\partial F}{\partial n_i^M} = \frac{\partial (\mathbf{t}-\mathbf{z})^T(\mathbf{t}-\mathbf{z})}{\partial n_i^M} = \frac{\partial \sum_{i=1}^{s^M}\left(t_j - z_j\right)^2}{\partial n_i^M} = -2\left(t_i - z_i\right)\frac{\partial z_i}{\partial n_i^M} = \\ = -2\left(t_i - z_i\right)\frac{\partial f^M\left(n_i^M\right)}{\partial n_i^M} = -2\left(t_i - z_i\right)\dot{f}^M\left(n_i^M\right) \end{aligned} \tag{A.18}$$

quindi in forma matriciale:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M\left(\mathbf{n}^M\right)(\mathbf{t}-\mathbf{z}) \tag{A.19}$$

Data questa sensitività è possibile aggiornare i pesi e i bias relativi all'ultimo layer, poi determinare la sensitività dei layer successivi, aggiornare i pesi e i bias e così fino al raggiungimento del primo layer.

B | Script per la creazione di geometrie in Python

Si riporta di seguito il codice in linguaggio `Python` per la creazione di geometrie random periodiche con diametro delle sfere polidisperso.

```
import math as m
import numpy
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from matplotlib.patches import Circle

# Dati
Num     = 30
eps     = 0.55
Dg      = 100e-6
# Calcoli
Acerchio = m.pi * Dg * Dg / 4
Asolido = Num * Acerchio
Abox = Asolido / (1 - eps)
L = m.sqrt(Abox)
Dg_list = numpy.random.normal(Dg * 1000000, 10, Num)
Dg_list=sorted(Dg_list, reverse=True)
for i in range (0,Num):
        Dg_list[i]= Dg_list[i] / 1000000
x_list_meta     = [0]
y_list_meta     = [0]
Dg_list_meta = [0]
# posizione random A
x_list = [None] * Num
y_list = [None] * Num
flag = 0
stop = 0
x_max = L / 2 - Dg_list[0] / 2
x_min = Dg_list[0] / 2
y_max = L - Dg_list[0] / 2
y_min = Dg_list[0] / 2
x_list[0] = numpy.random.uniform(x_min, x_max)
y_list[0] = numpy.random.uniform(y_min, y_max)
# posizione random BU e BL
x_max = L - Dg_list[1] / 2
x_min = L / 2 + Dg_list[1] / 2
y_min = L - Dg_list[1] / 2
```

```
y_min = L
x_list[1] = numpy.random.uniform(x_min, x_max)
y_list[1] = numpy.random.uniform(y_min, y_max)
x_list_meta[0] = x_list[1]
y_list_meta[0] = y_list[1] - L
Dg_list_meta[0] = Dg_list[1]
for i in range (2,Num):
 for k in range (0, 100000):
  flag_1 = 0
  flag_2 = 0
  flag_3 = 0
  flag_4 = 0
  y_meta = 0
  x_max = L - Dg_list[i] / 2
  x_min = Dg_list[i] / 2
  y_max = L
  y_min = 0
  x = numpy.random.uniform(x_min, x_max)
  y = numpy.random.uniform(y_min, y_max)
  if y < Dg_list[i] / 2:
        y_meta = y + L
  if y > (L-Dg_list[i] / 2):
        y_meta  = y - L
  if y_meta != 0:
        for l in range (0,len(Dg_list_meta)):
          distanza_BL_CU = m.sqrt(m.pow(x_list_meta[l] - x,2) + m.pow(
             y_list_meta[l] - y_meta,2))
          distanza_BL_CL = m.sqrt(m.pow(x_list_meta[l] - x,2) + m.pow(
             y_list_meta[l] - y,2))
          min_dist = (Dg_list_meta[l] / 2 + Dg_list[i] / 2) * 1.15
          if distanza_BL_CU <= min_dist or distanza_BL_CL <= min_dist:
                                                          flag_1 = 1
          elif l == len(Dg_list_meta) -1 and distanza_BL_CU > min_dist
              and distanza_BL_CL > min_dist:
                flag_2 = 1
          if flag_1 == 1 or flag_2 ==1:
                break
        if flag_1 == 1:
          None
        else:
          for l in range (0, i):
                distanza_BU_CU = m.sqrt(m.pow(x_list[l] - x,2) + m.pow
                   (y_list[l] - y_meta,2))
                distanza_BU_CL = m.sqrt(m.pow(x_list[l] - x,2) + m.pow
                   (y_list[l] - y,2))
                min_dist = (Dg_list[l] / 2 + Dg_list[i] / 2) * 1.15
                if  distanza_BU_CL      <= min_dist or distanza_BU_CU
                   <= min_dist:
                  break
                elif l == i-1 and distanza_BU_CL > min_dist and
                   distanza_BU_CU > min_dist:
                  x_list[i] = x
                  y_list[i] = y
```

```
                    x_list_meta.append(x)
                    y_list_meta.append(y_meta)
                    Dg_list_meta.append(Dg_list[i])
                    flag = 1
                    conto = conto + 1
            if flag == 1:
                flag = 0
                    break
        if k == (100000-1):
                    stop = 1
                    break
    else:
      for l in range (0,len(Dg_list_meta)):
            distanza_D_BL = m.sqrt(m.pow(x_list_meta[l] - x,2) + m.pow(
               y_list_meta[l] - y,2))
            min_dist = (Dg_list[l] / 2 + Dg_list[i] / 2) * 1.15
            if distanza_D_BL <= min_dist:
                  flag_3 = 1
                  break
            elif l == len(Dg_list_meta) - 1 and distanza_D_BL > min_dist
               :
                  flag_4 = 1
            if flag_3  == 1 or flag_4    == 1:
                  break
          if flag_3  == 1:
            None
          else:
            for l in range (0, i):
                  distanza_D_BU = m.sqrt(m.pow(x_list[l] - x,2) + m.pow(
                     y_list[l] - y,2))
                  min_dist = (Dg_list[l] / 2 + Dg_list[i] / 2) * 1.15
                  if  distanza_D_BU <= min_dist:
                    break
                  elif l == i-1 and distanza_D_BU > min_dist :
                x_list[i] = x
                    y_list[i] = y
                    flag = 1
                    break
            if flag == 1:
                  flag = 0
                  break
            if k == (100000-1):
                  stop = 1
                  break
    if stop == 1:
          print("Riprovare")
          break


if stop != 1:
    sfere = open("sfere_shm1.txt","w")
    for i in range (0,Num):
          sfera = i + 1
```

```
        sfere.write("sphere_%r\n" % sfera)
        sfere.write("{\n")
        sfere.write("type\tsearchableSphere;\n")
        sfere.write("centre\t(%r 0 %r);\n" %(x_list[i],y_list[i]))
        r = Dg_list[i] / 2
        sfere.write("radius %r;\n" % r)
        sfere.write("}\n\n" )
sfera = 30
for i in range (0, len(Dg_list_meta)):
        sfera = sfera + 1
        sfere.write("sphere_%r\n" % sfera)
        sfere.write("{\n")
        sfere.write("type\tsearchableSphere;\n")
        sfere.write("centre\t(%r 0 %r);\n" %(x_list_meta[i],
           y_list_meta[i]))
        r = Dg_list_meta[i]/2
        sfere.write("radius %r;\n" % r)
        sfere.write("}\n\n" )
        sfere.close()
```

Lista dei simboli

Simbolo	Descrizione	Unità di misura
$\mathbf{b}$	Bias della rete	-
C	Concentrazione del colloide	mol m^{-3}
C_0	Concentrazione del colloide in ingresso	mol m^{-3}
$\mathcal{D}$	Coefficiente di diffusione	m^2 s^{-1}
ΔP	Perdite di carico	Pa
d_P	Dimensione caratteristica del colloide	m
d_g	Dimensione dei grani	m
ε	Porosità del mezzo	-
J_i	Flusso diffusivo in direzione i	mol m^{-3} s^{-1}
k	Permeabilità	m^2
k_B	Costante di Boltzmann	1.38×10^{-23} J K^{-1}
Γ	Coefficiente di trasporto molecolare	m^2 s^{-1}
L	Lunghezza del dominio nella direzione principale del flusso	m
λ	Learning rate	-
M	numero di layer della rete	-
μ	Viscosità dinamica del fluido	kg m^{-1} s^{-1}
ν	Viscosità cinematica del fluido	m^2 s^{-1}
η	Efficienza di deposizione	-
Ω	Velocità di rotazione	RPM
q	Velocità superficiale del fluido	m s^{-1}
ρ	Densità del fluido	kg m^{-3}
$\mathbf{s}^m$	Sensitività del layer m	-
ϕ	Proprietà generica trasportata	var

S_{ij}	Strain rate	s^{-1}
σ	Deviazione standard	var
T	Temperatura del fluido	K
$\mathbf{t}$	Target della rete	var
U_i	Velocità del fluido in direzione i	m s^{-1}
$\mathbf{W}$	Pesi della rete	-
$\mathbf{x}$	Feature in ingresso alla rete	var
$\mathbf{z}$	Predizione in uscita dalla rete	var

Acronimi

AI Artficial Intelligence

ANN Artficial Neural Networks

CDS Central Difference Scheme

CFD Computational Fluid Dynamics

CNN Convolutional Neural Network

DNN Deep Neural Network

GPT General-Purpose Technologies

LUDS Linear Upwind Difference Scheme

ML Machine Learning

MLP Multi-Layer Perceptron

OpenFOAM Open-source Field Operation And Manipulation

SIMPLE Semi-Implicit Method for Pressure Linked Equations

TLU Threshold Logic Unit

TVD Total Variation Diminishing

Bibliografia

[1] A. Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

[2] G. Gottsegen. 15 examples of machine learning making established industries smarter. `https://builtin.com/artificial-intelligence/machine-learning-examples-applications`. [Online; accessed 20-Settembre-2019].

[3] S. Makridakis. The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms. *Futures*, 90:46–60, 2017.

[4] M. C. Iain, R. Henderson, and S. Stern. *The Impact of Artificial Intelligence on Innovation: An Exploratory Analysis*, pages 115–146. University of Chicago Press, January 2018.

[5] F. Oviedo, Z. Ren, S. Sun, C. Settens, Z. Liu, N. T. P. Hartono, S. Ramasamy, B. L. DeCost, S. I. P. Tian, G. Romano, et al. Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks. *npj Computational Materials*, 5(1):60, 2019.

[6] W. Izhar, D. Michael, and H. Abraham. Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *CoRR*, abs/1510.02855, 2015.

[7] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195, 2019.

[8] V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65(2):466–478, 2019.

[9] P. Czop, G. Kost, D. Sławik, and G. Wszołek. Formulation and identification of first-principle data-driven models. *Journal of Achievements in materials and manufacturing Engineering*, 44(2):179–186, 2011.

[10] A. Dubbelboer, J. J. Janssen, H. Hoogland, E. Zondervan, and J. Meuldijk. Pilot-scale production process for high internal phase emulsions: Experimentation and modeling. *Chemical Engineering Science*, 148:32–43, 2016.

[11] G. Boccardo, D. L. Marchisio, and R. Sethi. Microscale simulation of particle deposition in porous media. *Journal of colloid and interface science*, 417:227–237, 2014.

[12] Matlab. Getting started with machine learning. `https://it.mathworks.com/campaigns/offers/machine-learning-with-matlab.html`. [Online; accessed 28-Luglio-2019].

[13] Matlab. Applying supervised learning. `https://it.mathworks.com/campaigns/offers/machine-learning-with-matlab.htmlB`. [Online; accessed 28-Luglio-2019].

[14] Matlab. Applying unsupervised learning. `https://it.mathworks.com/campaigns/offers/machine-learning-with-matlab.htmlB`. [Online; accessed 28-Luglio-2019].

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Wikipedia. Multipolar neuron. `https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png`. [Online; accessed 3-Ottobre-2019].

[17] Enciclopedia Treccani. Neuroni e sinapsi. `http://www.treccani.it/enciclopedia/neuroni-e-sinapsi_28Enciclopedia-della-Scienza-e-della-Tecnica29/`. [Online; accessed 9-Agosto-2019].

[18] B. Andersson, R. Andersson, L. Håkansson, M. Mortensen, R. Sudiyo, and B. Van Wachem. *Computational fluid dynamics for engineers*. Cambridge University Press, 2011.

[19] L. M. de la Cruz and D. Monsivais. Parallel numerical simulation of two-phase flow model in porous media using distributed and shared memory architectures. *Geofísica internacional*, 53(1):59–75, 2014.

[20] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

[21] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan. *Neural network design*. Martin Hagan, 2014.

[22] IKA process. Ika mixing and processing technology. `https://www.ikaprocess.com/ika/pdf/flyer-catalog/201609_Process_Technology_IWS_EN_94000172_web.pdf`. [Online; accessed 01-Settembre-2019].

[23] G. Boccardo, F. Augier, Y. Haroun, D. Ferre, and D. L. Marchisio. Validation of a novel open-source work-flow for the simulation of packed-bed reactors. *Chemical Engineering Journal*, 279:809–820, 2015.

[24] M. Icardi, G. Boccardo, D. L. Marchisio, T. Tosco, and R. Sethi. Pore-scale simulation of fluid flow and solute dispersion in three-dimensional porous media. *Physical review E*, 90(1):013032, 2014.

[25] Chemical Book. Soybean oil. `https://www.chemicalbook.com/ProductChemicalPropertiesCB2703220_EN.htm`. [Online; accessed 6-Settembre-2019].

[26] K. Yao, M. T. Habibian, and C. R. O'Melia. Water and waste water filtration. concepts and applications. *Environmental science & technology*, 5(11):1105–1112, 1971.

[27] R.B. Bird, W.E. Stewart, and E.N. Lightfoot. *Transport Phenomena*. Wiley International edition. Wiley, 2006.

[28] X. Li, J. Zhang, and L. Xu. A numerical investigation of the flow between rotating conical cylinders of two different configurations. *Journal of Hydrodynamics*, 26(3):431–435, 2014.

Ringraziamenti

Giunta al termine della scrittura colgo l'occasione per ringraziare chi mi è stato vicino in questi mesi di lavoro e durante questi cinque anni di studio.

Ringrazio il Prof. Daniele Marchisio per avermi dato l'opportunità di lavorare a questo progetto di tesi, per la cura con cui mi ha guidata nella stesura e nella revisione di questo elaborato, la sua esperienza mi ha permesso di comprendere il significato profondo di ciò che stavo facendo. Ringrazio il Prof. Gianluca Boccardo per avermi seguita passo passo nella realizzazione dell'intero lavoro, per la sua disponibilità e il suo entusiasmo, che mi hanno permesso di fare cose di cui non mi sarei creduta capace fino a sette mesi fa. Vi ringrazio per avermi permesso di entrare nel campo del Machine Learning e per avermi lasciato piena libertà di espressione.

Grazie al mio gruppo olefine (Chiara, Giulio, Giorgio, Giovanni, Stefano) per avermi fatto comprendere il valore che ha avere un obiettivo comune e per aver condiviso momenti stressanti e divertenti. É grazie alla collaborazione di ognuno di voi che sono riuscita a dedicare tempo alla mia tesi e raggiungere questo traguardo.

Ringrazio le mie compagne di corso e amiche, Carola, Eleonora, Federica, Silvia, con cui ho condiviso ogni lezione, ogni sessione esame, ogni progetto e ogni decisione in questi ultimi quattro anni. Venire al poli ogni mattina sapendo che ci sareste state voi con cui parlare e scherzare non è stato (quasi) mai un peso!

Ringrazio Lorenzo per aver condiviso con me ogni istante della magistrale, per tutti gli esercizi e i temi d'esame fatti insieme, per tutte le ore passate a ripetere prima degli esami, per aver reso divertenti le ore interminabili di lezione e il laboratorio di fluidodinamica in questi mesi di tesi. Ma tutto questo è riduttivo, grazie per esserci stato sempre.

Grazie ai miei amici, Chiara, Alessandro, Martina per essere cresciuti con me in questi anni di università e per tutto ciò che abbiamo condiviso.

Grazie ad Alessandro per aver creduto sempre in me, per avermi sopportata in ogni sessione esame, per aver festeggiato con me ogni traguardo come se fosse anche suo.

Grazie ai miei zii adottivi, Claudio e Simonetta, per aver festeggiato con me la fine di ogni sessione, per aver sempre avuto stima e orgoglio del mio lavoro e dei miei studi.

Ringrazio i miei genitori per avermi permesso di dedicarmi completamente agli studi in questi anni e per avermi sostenuta in ogni decisione. Grazie a mia madre, Laura, per avermi sopportata nei momenti peggiori, apprezzo molto tutto ciò che fai per me. Grazie a mio padre, Giuseppe, per avermi insegnato ad affrontare ogni situazione con coraggio e determinazione, non potrei avere esempio migliore.

Infine, ringrazio mia nonna Maddalena, a cui dedico questa tesi. Da quando ero bambina e fino agli ultimi esami universitari il mio studio è stato accompagnato dal tuo cucito. Ho imparato da te ad essere paziente e mi hai insegnato che un lavoro ha valore solo se fatto bene dall'inizio alla fine, se sono riuscita a raggiungere questo traguardo lo devo ai tuoi insegnamenti.