POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Chimica e dei Processi Sostenibili

Politecnico di Torino

Tesi di Laurea Magistrale

Simulazioni CFD e Machine Learning sull'impatto della struttura dei tessuti sulla loro permeabilità all'aria

Relatore

Prof. Gianluca BOCCARDO

Co-relatori

Prof.ssa Ada FERRI

Dott.ssa Eleonora BIANCA

Candidato

Marco LIPRANDI

Ottobre 2025

Sommario

Il mercato dei tessuti è in continua crescita, sia per quanto riguarda i tessuti convenzionali che per i tessuti tecnici e la permeabilità dei tessuti all'aria risulta di primaria importanza in diverse applicazioni tecniche. L'approccio basato su fluidodinamica computazionale (CFD) permette di valutare la permeabilità già in fase di progetto, diminuendo i costi economici derivanti dalle prove sperimentali, mentre un modello surrogato basato su machine learning puo' ridurre il costo computazionale rispetto all'approccio CFD, permettendo operazioni di esplorazione ed ottimizzazione di diversi set di parametri.

In quest'ottica l'obiettivo di questo lavoro di tesi è stato la creazione di un modello basato sul machine learning, utilizzando una rete neurale fully connected, per lo studio della permeabilità all'aria dei tessuti intrecciati, analizzati tramite CFD (usata per la creazione del dataset) al variare di alcuni parametri di design fondamentali come armatura e fabric count.

Utilizzando il software CFD open source OpenFOAM si è valutata la permeabilità dei tessuti all'aria in modo da creare un dataset robusto e validato su dati sperimentali presenti in letteratura. I tessuti sono stati modellati come mezzi porosi trascurando, con scelte supportate dalla letteratura, la micro-permeabilità dei fili per i campioni analizzati.

Le geometrie analizzate sono state generate tramite il software TexGen a partire da parametri geometrici presenti in letteratura, e le geometrie virtuali generate sono state validate tramite la variazione di un parametro geometrico di fitting, il reduction factor, in modo da minimizzare l'errore tra i risultati delle simulazioni CFD e quelli sperimentali. I risultati presentano l'individuazione del parametro di fitting per ogni geometria e armatura studiate.

Questa metodologia ha consentito la generazione di un dataset utilizzato per l'allenamento di una rete neurale capace di prevedere i valori di permeabilità a partire da combinazioni inesplorate dei parametri di design del tessuto.

Indice

Elenco delle tabelle

Elenco delle figure

Lista dei Simboli

Simbolo	Descrizione	Unità di misura
D_1	densità di fili di ordito	$fili/cm$
D_2	densità di fili di trama	cm
S_1	spacing fili di ordito	cm
S_2	spacing fili di trama	$fili/cm$
Δx	allungamento filo	m
x	lunghezza iniziale del filo	m
C_f	cover factor	$-$
O_a	open area	$-$
φ	packing fraction	$-$
v_{fibre}	volume fibre	m^3
v_{filo}	volume filo	m^3
w	larghezza filo	cm
t	altezza filo	cm
r_f	reduction factor	$-$
Δt	differenza altezza filo	cm
ν	Poisson ratio	$-$
ϕ	generica grandezza nell'equazione di trasporto	?
ρ	densità del fluido	kg/m^3
U	velocità del fluido	m/s
Γ^ϕ	coefficiente di diffusione di ϕ	m^2/s
Q^ϕ	termine sorgente ϕ	$?/m^3$
V_C	volume di controllo della cella V_C	m^3
$\mathbf{S}$	vettore superficie	$-$
$\mathbf{J}^{\phi,C}$	flusso convettivo	$?/m^2$
$\mathbf{J}^{\phi,D}$	flusso diffusivo	$?/m^2$
$\mathbf{J}^\phi$	flusso totale	$?/m^2$
ω_{ip}	peso del metodo di quadratura	$-$
a_C	coefficiente di linearizzazione del flusso	$-$
a_F	coefficiente di linearizzazione del flusso	$-$
b_C	coefficiente di linearizzazione del flusso	$-$

δx	distanza tra due centroidi	m
$\dot{m}_e$	portata massica nella faccia e	kg/s
$\dot{m}_w$	portata massica nella faccia w	kg/s
ϕ_e	valore di ϕ nella faccia e	?
ϕ_w	valore di ϕ nella faccia w	?
ϕ_E	valore di ϕ nel centroide E	?
ϕ_C	valore di ϕ nel centroide C	?
ϕ_W	valore di ϕ nel centroide W	?
R	permeabilità (ISO 9237:1995)	mm/s
$\bar{q}_v$	media aritmetica della portata d'aria	L/min
A	area del provino	cm^2
$K_\parallel$	permeabilità del filo (parallela)	m^2
R	raggio della fibra	m
c	costante di Kozeny	$-$
V_f	frazione volumica filo	$-$
C_1	parametro	$-$
$V_{f,max}$	frazione volumica massima filo	$-$
π	pi greco	3.14
D_h	diametro idraulico	μm
A_{poro}	Area poro	cm^2
P_{poro}	Perimetro poro	cm^2
a	dimensione poro lungo l'ordito	cm
b	dimensione poro lungo la trama	cm
n_x	numero di celle lungo x	$-$
CpD	celle per diametro	$-$
l_x	lunghezza del tessuto lungo x	cm
n_y	numero di celle lungo y	$-$
l_y	lunghezza del tessuto lungo y	cm
n_z	numero di celle lungo z	$-$
l_z	lunghezza del tessuto lungo z	cm
v	velocità	m/s
P	pressione	$Pa/m/s^2$
ν	viscosità cinematica	m^2/s
K_f	permeabilità calcolata con Darcy	m^2
μ	viscosità dinamica	$Pa \cdot s$
ΔP	differenza di pressione	Pa
L	spessore del tessuto	m
Re	numero di Reynolds	$-$
R_h	raggio idraulico	μm
MSE	mean square error	$-$
v_{cfd}	velocità CFD	m/s

v_{exp}	velocità sperimentale	m/s
$RMSE$	root mean square error	$-$

Capitolo 1

Introduzione

Il valore del mercato mondiale del settore tessile è stimato in 1.07 trilioni di dollari nel 2024, con un aumento previsto a 1.48 trilioni entro il 2033, trainato dalla crescente richiesta di tessuti eco-friendly in ambiti come l'abbigliamento o l'arredamento e di tessuti tecnici utilizzabili in diversi comparti industriali [1].

Proprio i tessuti tecnici rappresentano il segmento caratterizzato dal maggior interesse per le prestazioni e le caratteristiche innovative e alcune stime valutano il mercato dei tessuti tecnici attribuendogli un valore di circa 239 miliardi di dollari nel 2024, con predizioni in crescita fino a circa 392 miliardi di dollari nel 2032. [2]

La principale distinzione nei prodotti tessili avviene dividendo i tessuti convenzionali, utilizzati per uso comune o decorativo ed estetico, da quelli tecnici, dei quali si fa un uso funzionale sfruttando le proprietà dei prodotti. I tessuti tecnici a loro volta si dividono in dodici categorie a seconda della loro area di applicazione, individuate per la prima volta da Techtextil, la principale esposizione mondiale di tessuti tecnici [3]:

1. Agrotech: agricoltura, acquacoltura, orticoltura e silvicoltura
2. Buildtech: edilizia e costruzioni
3. Clothech: componenti tecnici per calzature e abbigliamento
4. Geotech: geotessili e ingegneria civile
5. Hometexh: componenti tecnici per mobili, tessili per la casa e rivestimenti per pavimenti
6. Indutech: filtrazione, trasporto, pulizia e altri impieghi industriali
7. Medtech: igiene e ambito medico
8. Mobiltech: automotive, navale, ferroviario e aerospaziale

9. Oekotech: protezione ambientale
10. Packtech: imballaggio
11. Protech: protezione personale e dei beni
12. Sporttech: sport e tempo libero

Sebbene i confini tra le diverse categorie non siano spesso definiti univocamente e spesso alcuni ambiti sembrino sovrapporsi (come per Geotech ed Agrotech, o per Packtech e Protech) questa è ancora oggi la definizione più utilizzata per i tessuti tecnici.

Tra le prestazioni e le proprietà ricercate nei tessuti tecnici vi è la permeabilità all'aria dei tessuti. Questa caratteristica è infatti fondamentale in diverse applicazioni tecnico-industriali: nella filtrazione dell'aria contaminata da inquinanti o particelle solide, ad esempio, si utilizzano materiali fibrosi per costruire filtri realizzati in tessuti "non tessuti", cioè realizzati per diretta compattazione delle fibre che facilitino il passaggio dell'aria pur rimuovendo i corpi esterni. In questo caso perciò la permeabilità all'aria è uno dei parametri principali, insieme all'efficienza di rimozione delle particelle, che determinano le prestazioni di un filtro [4].

Un'altra applicazione dei tessuti tecnici in cui la permeabilità dell'aria è cruciale per il corretto funzionamento sono gli airbag realizzati in tessuti intrecciati di Nylon ai quali viene richiesta una bassa permeabilità all'aria per garantire una corretta fase di gonfiaggio ed evitare perdite di gas. Trattandosi di tessuti intrecciati, la permeabilità in questa particolare applicazione può essere controllata non solo tramite il coating del tessuto, generalmente realizzato in silicone, ma ancora prima a partire da parametri di design come l'armatura del tessuto e il titolo del filo, cioè la sua massa per unità di lunghezza [5].

Nei capi d'abbigliamento utilizzati per lo sportswear la permeabilità all'aria è una proprietà di fondamentale importanza in quanto da essa dipendono le caratteristiche di termoregolazione e comfort dei prodotti indossati, soprattutto se a diretto contatto con la pelle, dove la permeabilità all'aria influenza anche l'asciugatura del tessuto. Un alto valore di permeabilità del tessuto infatti aumenta lo scambio termico tra il corpo e l'ambiente tramite convezione ed evaporazione, aumentando anche la velocità di evaporazione del sudore durante l'esercizio fisico [6].

Diversi studi hanno cercato di stimare questa proprietà dei tessuti tramite CFD. Nel loro studio Puszkarz e Krucińska [7] analizzano tessuti a maglia a doppio strato utilizzando SolidWorks per modellare i tessuti a partire da immagini ottiche e parametri noti sperimentalmente, trovando però differenze significative tra i valori sperimentali di permeabilità e quelli simulati attribuite alla semplificazione che deriva dal modellare i tessuti come strutture perfettamente periodiche che non riflette quindi la loro reale eterogeneità.

Zeng et al. [8] introducono una descrizione più realistica in questo senso, investigando un tessuto multistrato con armatura Twill in fibra di carbonio e un tessuto multistrato con armatura Plain Weave in fibra di vetro. Entrambi vengono modellati con TexGen applicando rotazioni e traslazioni locali delle fibre per simulare il reale comportamento dei tessuti, dovuto principalmente a sforzi di taglio e compressione. Simulando il flusso attraverso il tessuto con ANSYS CFX in condizioni laminari viene calcolato il tensore della permeabilità attraverso la legge di Darcy e i valori di permeabilità vengono comparati con i dati sperimentali indicando una sovrastima per quanto riguarda le geometrie idealizzate e una miglior rappresentazione della realtà quando si considerano le deformazioni subite dai tessuti.

Xiao et al. [9] concentrano invece l'analisi sui tessuti intrecciati, implementando un processo integrato che sfrutta il software TexGen per esportare le geometrie e analizzarle successivamente tramite CFD, sottoponendo i tessuti a differenze di pressione incrementali che però comportano l'utilizzo di diverse leggi per la valutazione della permeabilità, a seconda del regime fluidodinamico che si instaura. Lo studio, inoltre, si focalizza su un solo tipo di armatura, il Plain Weave, trascurando le altre possibili disposizioni dei fili nei tessuti.

La permeabilità è influenzata dalla struttura del tessuto. Lo studio di Zupin et al. [10] analizza infatti diverse geometrie di tessuti in cotone variando armature e densità di fili di trama e di ordito concentrandosi sui parametri geometrici dei pori che si creano tra i fili, come il loro diametro idraulico, mostrando la possibilità di analizzare queste strutture con strumenti e terminologie tipici della fluidodinamica.

In questa tesi l'obiettivo finale è stato lo sviluppo di un modello basato sul machine learning che fosse in grado di prevedere la permeabilità dei tessuti all'aria. Si è quindi allenata una rete neurale fully connected utilizzando come dataset i risultati delle simulazioni CFD validati attraverso i risultati sperimentali di Zupin et. al [10], ottenuti seguendo la norma ISO 9237:1995 [11].

I dati sul diametro dei fili misurati al microscopio ottico derivanti dallo stesso studio hanno infatti costituito il punto di partenza per generare le varie geometrie analizzate. Questi sono necessari per sfruttare le capacità del software TexGen di generare file .stl rappresentanti le superfici dei tessuti in modo da poter essere esportate e analizzate tramite CFD. Si è quindi utilizzato il software OpenFOAM per simulare le stesse condizioni che avvengono durante lo studio sperimentale della permeabilità dei tessuti.

Tra i parametri geometrici utilizzati per la descrizione delle geometrie si è utilizzato il reduction factor (r_f), non noto a priori, come parametro di fitting per individuare la geometria che restituisse il risultato di permeabilità più vicino a quello sperimentale. Questo parametro descrive il comportamento del tessuto quando sottoposto ad un carico, indicando la sua tendenza allo schiacciamento e aggiunge una descrizione ulteriore del comportamento del tessuto, evitando le problematiche

individuate da altri studi quando i modelli utilizzati per rappresentare i tessuti risultano troppo ideali. In questo lavoro i risultati delle simulazioni sono comparabili con i valori sperimentali, rendendo lo strumento della rete neurale utilizzabile, ad esempio, in fase di progetto nell'ingegneria di processo.

Capitolo 2

Elementi Teorici

2.1 Tessuti

Nel linguaggio tecnico dell'industria i tessuti sono strutture nelle quali il rapporto tra lo spessore e la superficie è molto basso e sono quindi approssimabili a strutture bidimensionali. Dal momento che questa definizione include strutture e materiali non necessariamente affini ai prodotti tessili, la prima classificazione più generale avviene dividendo i tessuti realizzati a partire da fibre dai tessuti realizzati in altri materiali. Nei tessuti fibrosi si inseriscono i tessuti intrecciati, a maglia e i tessuti "non tessuti", mentre tra i tessuti non fibrosi si possono classificare materiali come film plastici e sottili lastre metalliche. [12]

I tessuti formati da fibre sono perciò la categoria più vicina all'idea che l'immaginario comune possiede per queste strutture. All'interno di questa categoria è ancora possibile effettuare un'ulteriore distinzione, separando i tessuti ottenuti a partire dai fili, cioè formati dall'intreccio di fili per quanto riguarda i tessuti intrecciati o dalla loro interconnessione se si parla invece di tessuti a maglia dai tessuti "non-tessuti", che sono invece ottenuti per compattazione diretta delle fibre, ma non presentano nessuna organizzazione delle fibre in fili.

Sia che si tratti di tessuti convenzionali o di tessuti tecnici le caratteristiche e le prestazioni finali sono influenzate, secondo un approccio top-bottom, dalla costruzione del tessuto, dal tipo di filo e dal tipo di fibra.

2.1.1 Tessuti intrecciati

In questa tesi si è posta l'attenzione sui tessuti fibrosi costruiti a partire dai fili, in particolare sui tessuti intrecciati, che rappresentano la categoria caratterizzata da durabilità e stabilità dimensionale più elevata, le cui caratteristiche erano apprezzate già in antichità e che risulta tuttora la più utilizzata. [12] [13]

Quando si cercano determinate caratteristiche o proprietà in un tessuto intrecciato si agisce su alcuni fattori costruttivi e geometrici che più le influenzano, come ad esempio la densità superficiale, la densità volumica, lo spessore, il cover factor e l'utilizzo combinato di diversi tipo di filo e di fibre. [14]

2.1.2 Le armature

Un tessuto intrecciato è caratterizzato da due set di fili ortogonali tra loro. I fili di ordito sono verticali e sono disposti lungo la lunghezza del tessuto, e intrecciano quelli di trama che attraversano il tessuto orizzontalmente lungo la sua larghezza.

Il pattern secondo il quale i fili di trama e di ordito sono disposti nella faccia del tessuto definisce la sua costruzione, cioè la sua l'armatura, che è il principale fattore che determina le prestazioni e le caratteristiche del tessuto. Per definire un'armatura di un tessuto è sufficiente descrivere la posizione reciproca dei fili di trama e di ordito per la più piccola unità ripetitiva, che verrà ripetuta per tutta l'estensione superficiale del tessuto. [12]

È possibile quindi rappresentare schematicamente l'unità ripetitiva utilizzando uno schema a due colori che rappresenta l'intreccio dei fili, come in figura 2.1.

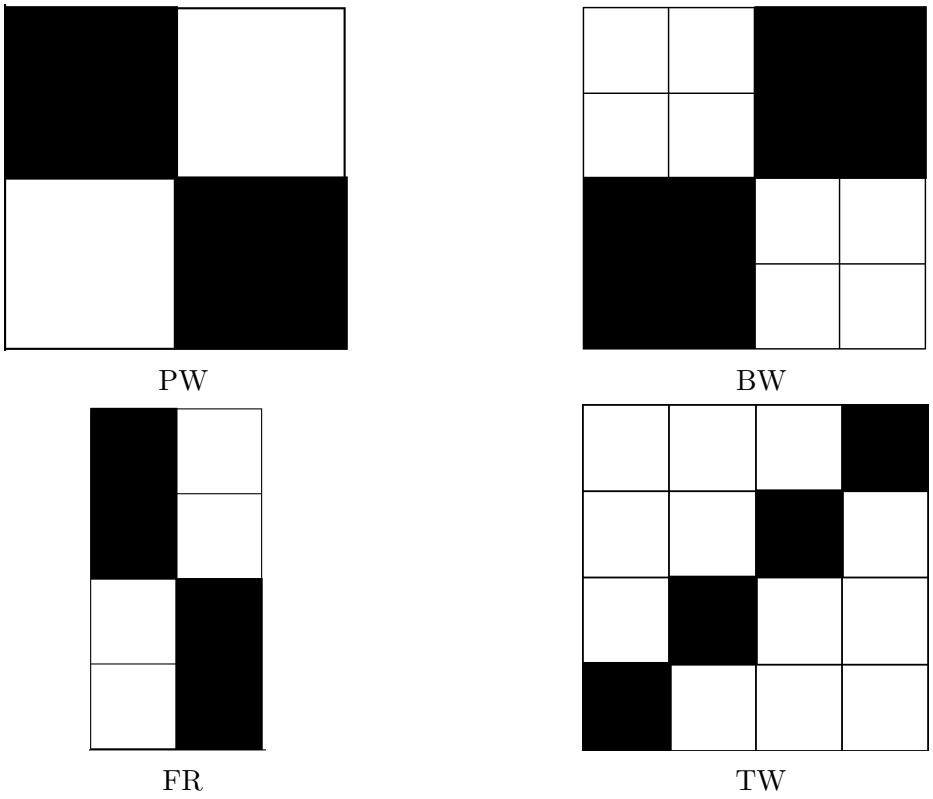


Figura 2.1: Rappresentazione schematica delle armature utilizzate

L'armatura più semplice e con un utilizzo trasversale per tessuti sia tecnici che convenzionali è detta Plain Weave (PW): per questo tipo di costruzione i fili di ordito interlacciano i fili di trama passando una volta sopra di essi e una volta sotto, rendendo di fatto visivamente equivalenti le due facce del tessuto, che risultano entrambe molto piatte e lisce. L'unità ripetitiva di questa armatura è costituita da due fili di trama e due di ordito. [12]

L'armatura Basket Weave (BW) descrive tessuti in cui più di un filo di ordito si interlaccia con più di un filo di trama, ad esempio se presenta due fili di ordito che passano in modo alternato due volte sopra e due volte sotto due fili di trama, in questo caso si parla di un'armatura BW 2/2. Per un BW 2/2 l'unità ripetitiva è formata da quattro fili di trama e quattro di ordito. [12]

Una costruzione simile al Basket Weave è il Filling Rib (FR), in cui invece che avere due fili di ordito che in modo alternato passano sotto a due fili di trama si trova un solo filo di ordito che passa in modo alternato sotto a due fili di trama: è il caso del FR 4/2, la cui unità ripetitiva è univocamente definita con quattro fili di trama e due di ordito. [12]

Il Twill (TW) è un'armatura che prevede un disegno diagonale: nello specifico caso del Twill 1/3 il primo filo di ordito dell'unità ripetitiva passa sopra al primo filo di trama e sotto ai successivi, mentre spostandosi verso destra il filo di ordito passa sopra al secondo filo di trama, al terzo ed infine al quarto, passando sotto a tutti gli altri. Le linee diagonali che caratterizzano questo tipo di armatura sono apprezzabili, ad esempio, nei jeans e nei tessuti in denim dove le caratteristiche dell'armatura e l'utilizzo di fili grezzi donano estrema durabilità e resistenza al tessuto. Osservando il tessuto nel verso dell'ordito si osserva quindi questa linea diagonale che corre verso destra, definendo un tessuto "Z Twill" o "destrorso". Il caso opposto, in cui la linea diagonale si sviluppa verso sinistra, è chiamato "S Twill" o "sinistrorso". [12]

2.1.3 Parametri geometrici

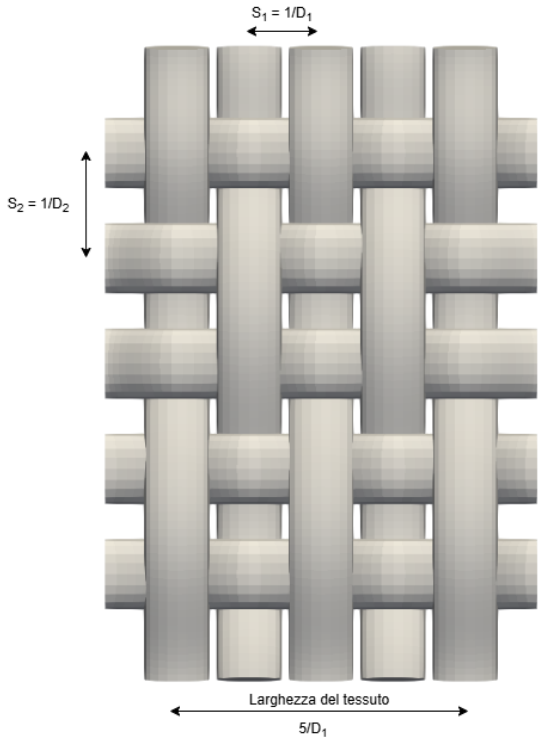
La sola armatura non è abbastanza per descrivere in modo dettagliato un tessuto e poter prevedere le sue caratteristiche e prestazioni.

Alcuni dei parametri che più influenzano le specifiche dei tessuti intrecciati, oltre alla disposizione dei fili che definisce l'armatura, sono: [12]

1. Densità di fili di trama e di ordito (fabric count)
2. Larghezza del tessuto
3. Spessore del tessuto
4. Densità superficiale

5. Crimp
6. Cover factor

Densità di fili di trama e di ordito (fabric count) e larghezza del tessuto



Larghezza e distanza tra fili

Spessore

Figura 2.2: Dimensioni e parametri geometrici del tessuto

La densità di fili (o fabric count) rappresenta il numero di fili per unità di lunghezza sia nella direzione dell'ordito che in quella della trama. Questo è solitamente misurato in $fili/cm$ o $fili/pollice$ ed il suo valore per i fili di trama e di ordito definisce rispettivamente un tessuto a faccia di trama se il fabric count è maggiore per i fili di trama, un tessuto a faccia di ordito se il valore di questo parametro è maggiore per i fili verticali e un tessuto equi-fronte se il numero di fili per unità di lunghezza è uguale per entrambi i set di fili.

Il fabric count descrive la "densità" di un tessuto, dove con "denso" si intende un tessuto in cui i fili sono molto ravvicinati tra loro. Il reciproco del fabric count infatti indica la distanza tra due fili adiacenti di trama o di ordito, permettendo di descrivere il secondo parametro elencato, la larghezza del tessuto, come il prodotto tra il reciproco della densità di fili di ordito (D_1) e il numero di fili di ordito da cui è composto un tessuto, come indicato in figura 2.2.

Spessore del tessuto

Lo spessore del tessuto è un parametro tanto importante quanto difficile da misurare. Nonostante i tessuti siano approssimabili come strutture bidimensionali, in molte applicazioni lo spessore rappresenta un fattore determinante per il comportamento del tessuto. Lo spessore non è un parametro stabile in quanto è fortemente dipendente dalla distorsione che il filo subisce durante il processo di tessitura e per questo si determina sperimentalmente attraverso la norma BS EN ISO 5084:1997 [15]. In questa norma lo spessore del tessuto è definito come la distanza perpendicolare tra due piatti di riferimento, uno fisso e uno che esercita sul tessuto una specifica pressione, uguale a $(1 \pm 0.01)\ kPa$ (o $(0.1 \pm 0.001)\ kPa$, se si intende testare tessuti a rilievo, pile o alcuni tessuti a maglia) dopo un intervallo di 30 ± 5 s.

Densità superficiale

La densità superficiale è una grandezza fisica misurata in g/m^2 o in oz/yd^2 e distingue i tessuti pesanti da quelli leggeri. Questa distinzione è soprattutto utilizzata nel campo dell'abbigliamento, dove i tessuti leggeri sono utilizzati per indumenti come camicie e magliette, mentre i tessuti pesanti sono tipici di capi come pantaloni o gonne. Il peso di un tessuto è controllabile agendo su parametri costruttivi e progettuali come la scelta del tipo di fibra, il titolo del filo (ovvero il rapporto tra la lunghezza del filo e il suo peso), il fabric count e l'armatura del tessuto. Inoltre alcuni processi post tessitura come trattamenti meccanici o chimici possono influire sul peso del tessuto. [12]

Crimp

Il Crimp è un concetto derivante dal processo stesso di tessitura: i fili, una volta che fanno parte del tessuto presentano una lunghezza minore di quella che possedevano originalmente per via delle pieghe che sono costretti a fare per determinare la struttura dell'armatura. Il Crimp è calcolato come:

$$Crimp(\%) = \frac{\Delta x}{x} \cdot 100 \tag{2.1}$$

Dove x è la lunghezza del filo all'interno del tessuto e $x+\Delta x$ è la lunghezza del filo quando viene estratto dal tessuto. Anche questo parametro è difficile da misurare e il suo valore è soggetto a fluttuazioni in quanto la pratica stessa dell'estrarre il filo dal tessuto ne determina un suo allungamento ogni volta differente.

Cover factor

Il Cover factor misura la percentuale dell'area coperta dal tessuto. Si calcola con l'equazione: [10]

$$C_f = (D_1 d_1 + D_2 d_2 - D_1 d_1, D_2 d_2) \cdot 100 \tag{2.2}$$

Dove $d_{1,2}$ è il diametro dei fili di trama e ordito e $D_{1,2}$ è il fabric count (il pedice 1 fa riferimento all'ordito, mentre il pedice 2 indica i fili di trama).

Il parametro complementare al cover factor, l'Open area, misura la superficie dei fori del tessuto e si calcola come: [10]

$$O_a = 100 - C_f \tag{2.3}$$

2.1.4 I fili

I fili rappresentano l'unità costitutiva dei tessuti. Alcune qualità del tessuto sono quindi direttamente correlate alle caratteristiche dei filati e vanno perciò progettati unitamente ai parametri geometrici indicati in precedenza se si è alla ricerca di specifiche prestazioni dei prodotti tessili. [16]

I fili per tessuti sono strutture lineari costituite da un insieme di fibre corte ritorte (spun yarns) o da un insieme di lunghi filamenti disposti in parallelo tra loro (continuous-filament yarn), tra i fili a filamento continuo è possibile distinguere ulteriormente quelli monofilamento da quelli multifilamento.

I filati ritorti sono tipici di fibre naturali come il cotone, dove si parte da fibre corte (staple fibres) e attraverso alcuni processi in serie che comprendono la preparazione della fibra stessa e il metodo di filatura scelto si giunge al filo allineando e consolidando le fibre attraverso una torsione delle stesse fibre.

La distinzione tra fili costituiti da filamenti continui e fili ottenuti a partire da fibre discontinue non è l'unica possibile, per specificare interamente un filo infatti è necessario indicare anche le caratteristiche della struttura di base, l'integrità superficiale e di bulk e le specifiche speciali come le caratteristiche di resistenza alle fiamme o i trattamenti con enzimi. [16]

La descrizione della struttura di base di un filo si basa su due importanti parametri: il titolo del filo (yarn count) e il numero di torsioni (twisting).

Lo Yarn Count esprime la finezza del filato, cioè la sua densità lineare. Questa grandezza è utilizzata al posto del diametro del filo dal momento che il diametro è una grandezza poco stabile e non uniforme lungo il filo. La densità lineare è esprimibile, per fibre corte e per tessuti da esse ottenuti, attraverso due sistemi: il sistema diretto e il sistema indiretto. Nel primo viene espressa la massa per unità di lunghezza, mentre per il secondo sistema viene indicato il reciproco, cioè la lunghezza per unità di massa. Le unità di misura usate per il sistema diretto sono il "tex" (misurato in g/km) con i suoi multipli e sottomultipli o il "denier" (misurato in $g/9km$) e sono normalmente più utilizzate rispetto a quelle indirette come il cotton count o il worsted count.

Il Twisting è il meccanismo attraverso il quale le fibre vengono consolidate tra loro per formare il filo e conferirgli la resistenza necessaria. Nel caso di fili creati a partire da fibre corte, il processo di twisting è necessario proprio per garantire l'integrità del filo, mentre per i filamenti continui questo processo non è necessario per la resistenza, ma è spesso utilizzato ugualmente per aumentare la coesione nei fili multifilamento.

La torsione dei fili è descritta attraverso due parametri: il "twist level" e il "twist direction". Il livello di twist rappresenta il numero di torsioni effettuate per unità di lunghezza, mentre la direzione di twist specifica se la torsione è avvenuta in senso orario o in senso antiorario, definendo rispettivamente uno "Z twist" e un "S twist".

2.1.5 Le fibre

L'unità di base di un filo, con i quali si costruisce poi un tessuto, è la fibra. che con le sue caratteristiche e le sue proprietà superficiali determina le proprietà meccaniche e fisiche del filo e quindi le qualità del tessuto [17]. La fibra tessile può essere naturale o artificiale ed è spesso un materiale polimerico a struttura lineare. All'interno delle fibre naturali è possibile distinguere ulteriormente quelle cellulosiche come il cotone o lino da quelle proteiche (come la lana o la seta) e dalle minerali, come la lana di roccia, mentre le fibre "man-made" si distinguono in fibre rigenerate e sintetiche, come il poliestere, che rappresenta la fibra con maggiore quota di mercato, seguita dal cotone. Le fibre di cotone, dalle quali si ottengono i fili che, intrecciati, producono i tessuti trattati in questa tesi, rientrano nella categoria delle fibre naturali, provenienti quindi da fonti vegetali o animali.

Il cotone è una fibra naturale cellulosica, la più importante tra quelle naturali. La sua catena polimerica è formata da più di 10000 monomeri di cellulosa è costituito perciò da carbonio, idrogeno e ossigeno. Le fibre naturali variano in lunghezza (quelle di cotone rientrano in un range tra gli 1.5 e i 5 cm) e proprietà ed infatti necessitano di alcuni processi per uniformarne la grandezza prima di essere ritorte per formare il filo.

La compattezza delle fibre è il parametro di maggior interesse per il lavoro sviluppato in questa tesi, in quanto da questo deriva l'integrità del filo e la possibilità di poter essere descritto come un mezzo poroso, con eventuali semplificazioni riguardanti la sua micro-permeabilità. Infatti, è possibile immaginare un filo spun come una struttura formata da materiale fibroso e da vuoti d'aria determinati dalla discontinuità delle fibre lungo l'asse del filo. Nel caso specifico di fili spun le fibre corte sono compattate tra loro grazie a forze laterali generate dalla torsione imposta alle fibre, e la natura discreta di queste fibre naturali determina una compattezza variabile delle fibre nel filo che lascia perciò spazio a numerosi vuoti. Una bassa compattezza delle fibre è quindi sinonimo di un filo comprimibile, flessibile e con alta porosità. [16]

La compattezza delle fibre è esprimibile, alternativamente, attraverso due parametri: la densità di bulk del filo (o il suo reciproco, il volume specifico) e il volume di aria presente all'interno del filo. Il volume specifico del filo dipende quindi dal volume delle fibre e da quello dei vuoti tra di esse, che è riempito di aria. il volume del filo e quello delle fibre permettono di introdurre il parametro adimensionale φ, chiamato packing fraction, che quantifica la frazione di volume occupata dalle fibre rispetto al volume totale del filato ed è definito come:

$$\varphi = \frac{v_{\text{fibre}}}{v_{\text{filo}}} = \frac{v_{\text{fibre}}}{v_{\text{fibre}} + v_{\text{aria}}} \tag{2.4}$$

L'arrangiamento delle fibre è un parametro che, per fibre corte, risulta poco controllabile e prevedibile. Di norma si fa riferimento ad una struttura idealizzata del filo, ma questo per fili spun come quelli di cotone è spesso non riproducibile e di scarso interesse. L'arrangiamento delle fibre in un filo composto da fibre corte è quindi in prima analisi idealizzato come per fili composti da filamenti ritorti e poi interpretato secondo le deviazioni che l'utilizzo di fibre corte comportano.

Per quanto riguarda la mobilità delle fibre, invece, questo parametro determina la stabilità dimensionale sia del filo che del tessuto. È un parametro sicuramente importante soprattutto per fibre staple, data la loro natura discreta, mentre risulta meno critico nel caso di fibre continue, che mostrano un minor grado di mobilità. Tuttavia per quanto riguarda la permeabilità all'aria questo parametro non incide tanto quanto la compattezza delle fibre.

2.1.6 Geometrie utilizzate

Per confrontare i risultati di permeabilità ottenuti dalle simulazioni CFD con quelli ottenuti da Zupin et al. [10] e poter quindi validare il modello sono state utilizzate le quattro armature descritte: Plain Weave (PW), Basket Weave (BW), Filling Rib (4/2) (FR) e Twill (1/3) (TW) e due valori ciascuno per quanto riguarda le densità di fili di trama D_1 e di ordito D_2, individuando quindi quattro combinazioni (22-15, 22-20, 29.3-15 e 29,3-20) per ciascuna delle quattro armature per un totale di sedici geometrie differenti.

Dal momento che i tessuti intrecciati sono descrivibili tramite le loro unità ripetitive si potrebbero, in linea teorica, utilizzare quattro fili (due di ordito e due di trama) per descrivere il PW, sei fili per il FR (due di ordito e quattro di trama) e otto fili (quattro di ordito e quattro di trama) per BW e TW. Per rendere uniforme la trattazione e mantenere una simmetria nelle geometrie si è però deciso in questa tesi di utilizzare dieci fili (cinque di ordito e cinque di trama) per ognuna delle quattro armature.

I risultati sperimentali riportati da Zupin et al. provengono da tessuti realizzati con fili di cotone aventi tutti la stessa densità lineare. I dati relativi al diametro dei fili di trama e di ordito misurati tramite il microscopio ottico forniti dallo stesso paper sono particolarmente interessanti e utili per definire le caratteristiche del filo e poterlo modellare utilizzando il software TexGen, come meglio indicato nel capitolo 2.2.

I fili, una volta intrecciati per formare il tessuto, vengono deformati in seguito all'interlacciamento e allo stiramento. La geometria finale dei fili è quindi descritta introducendo due ulteriori parametri per il filo: w e t, in aggiunta al diametro iniziale d che non è più in grado di rappresentare accuratamente i fili nel tessuto. Il filo deformato presenta infatti una sezione ellittica in cui w rappresenta la larghezza del filo e t la sua nuova altezza, come rappresentato in figura 2.3.

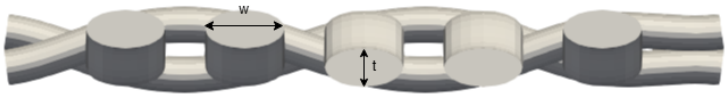


Figura 2.3: Sezione trasversale del filo in un tessuto

Queste due nuove dimensioni che descrivono la sezione trasversale del filo compaiono nella definizione di due nuovi parametri geometrici: il reduction factor

r_f e il poisson ratio ν.

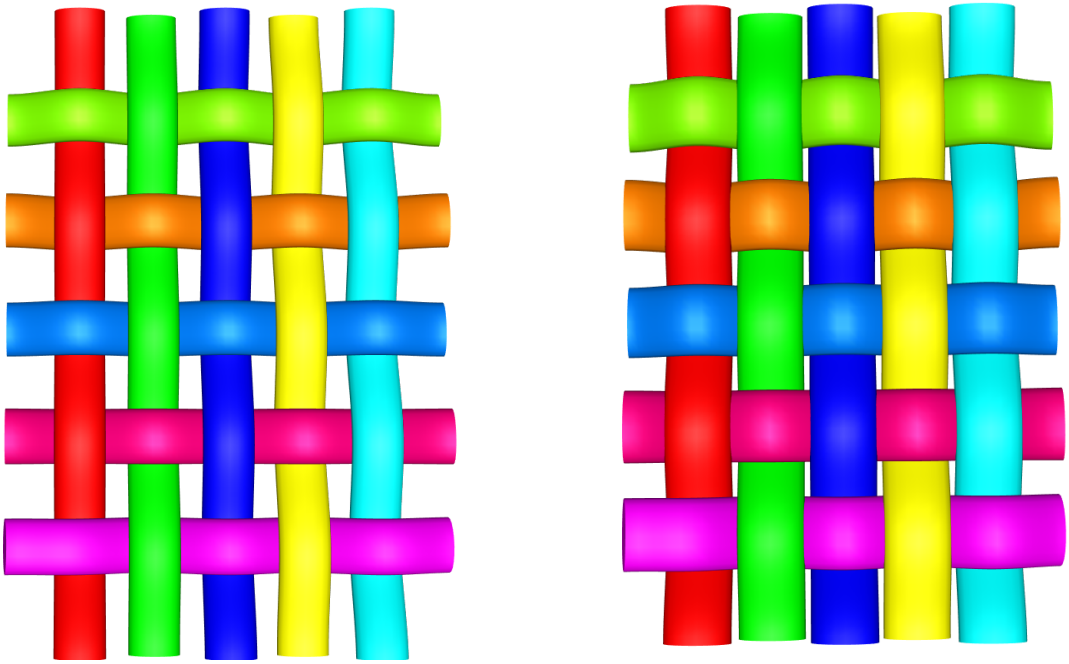
$$r_f = \frac{d-t}{d} = \frac{\Delta t}{d} \tag{2.5}$$

$$\nu = \frac{d-w}{d-t} \tag{2.6}$$

Il reduction factor è dato dal rapporto tra la differenza di altezza tra il filo iniziale e quello deformato e il diametro iniziale del filo. Questo parametro verrà utilizzato come parametro di fitting durante le simulazioni per descrivere il comportamento del filo, e quindi del tessuto, quando viene imposta una differenza di pressione ai capi del tessuto che determina il passaggio dell'aria.

Il Poisson ratio rappresenta il rapporto tra la deformazione trasversale e quella assiale ed infatti è calcolato come il rapporto tra la differenza tra il diametro iniziale d e la larghezza del filo w e la differenza tra il diametro iniziale e la sua altezza t.

Si è scelto in questa tesi di utilizzare un Poisson ratio pari a 0.4, valore tipico per fili di cotone [18].



Tessuto PW-22-15 Tessuto PW-29.3-20

Figura 2.4: Due tessuti PW rappresentanti gli estremi delle possibili combinazioni di fabric count

Poiché il diametro dei fili di trama e di ordito ed il Poisson ratio sono fissati, a parità di armatura e di r_f, la geometria con combinazione di densità 22-15 risulta

la meno densa (ovvero con i fili più lontani), mentre quella 29.3-20 è la più densa e quindi con dimensioni del tessuto minori, in quanto i cinque fili di trama e cinque di fili di ordito che la descrivono saranno più vicini tra loro. La distanza tra due fili consecutivi di trama o di ordito è infatti definita dall'equazione:

$$S = \frac{1}{D} \tag{2.7}$$

Dove D è rispettivamente pari a D_1 o D_2 nel caso di fili di ordito o di trama, individuando perciò le due distanze S_1 ed S_2. La distanza tra due fili, siano essi di trama o di ordito, è quindi inversamente proporzionale alla rispettiva densità dei fili.

Per visualizzare questo dato è possibile calcolare il volume del parallelepipedo che include il tessuto, cioè avente come dimensioni le massime estensioni nelle tre dimensioni del tessuto.

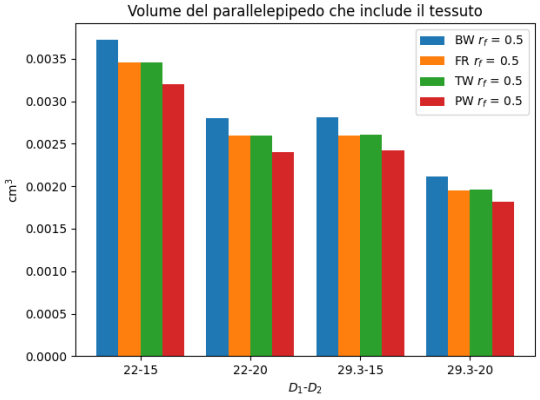


Figura 2.5: Dimensioni del parallelepipedo che include il tessuto al variare di armatura e fabric count

Il grafico 2.5 riporta come, per tessuti realizzati con cinque fili di trama e ordito e con reduction factor posto pari a 0.5, le densità 22-15 siano quelle con dimensioni maggiori per qualsiasi armatura. Le densità 22-20 e 29.3-15 risultano comparabili per stesse armature, mentre la combinazione 29.3-20 realizza tessuti con volume minore qualsiasi per qualsiasi armatura.

Per stesse densità invece l'armatura BW risulta sempre la più voluminosa, seguita da TW e FR con valori simili, mentre il PW è sempre la meno estesa. Questa caratteristica dipende principalmente dallo spessore del tessuto, quindi dalla sua estensione lungo l'asse z. Le dimensioni del tessuto lungo l'asse x e y sono infatti definite dalla densità di fili di ordito e trama rispettivamente e non sono influenzate dal pattern di intreccio dei tessuti. Viceversa, il modo in cui i fili sono intrecciati influenza lo spessore rendendo il tessuto più sottile per il PW dove i fili passano continuamente sopra e sotto i fili adiacenti e più spesso per il

BW, dove invece nell'unità ripetitiva i fili hanno una frequenza di interlacciamento minore dal momento che due fili di ordito passano contemporaneamente sopra e sotto due fili di trama.

2.2 TexGen

Il software open source TexGen è un programma sviluppato dall'Università di Nottingham per modellare in 3D tessuti e rinforzi per materiali compositi. Realizzando delle geometrie realistiche è possibile modellarne le proprietà sia meccaniche che fisiche senza campagne sperimentali che possono risultare costose o di difficile realizzazione. [19]

Il programma è disponibile per piattaforme Windows e Linux e crea modelli per i tessuti che possono essere esportati in formati utilizzabili per ulteriori indagini, sia tramite analisi agli elementi finiti, per investigare le proprietà meccaniche dei tessuti, che tramite fluidodinamica computazionale se si è interessati a proprietà come, ad esempio, la permeabilità.

TexGen è stato utilizzato nel workflow di questa tesi per generare le geometrie dei tessuti ed esportarle in formato .stl in modo da inserirle in un case openFOAM che permettesse di studiarne la permeabilità all'aria tramite CFD. I tessuti vengono modellati da Texgen a partire dai fili, che sono trattati come volumi solidi.

La creazione di un tessuto può avvenire attraverso tre principali metodi: la modalità Wizard, il menu Modeler attraverso l'interfaccia grafica (GUI) oppure tramite un codice Python. La prima modalità è quella più intuitiva e in quanto tale pone alcune limitazioni alla creazione di geometrie complesse, mentre le altre due permettono di sfruttare completamente il Core del programma arrivando al medesimo risultato in termini di geometrie del tessuto, ma sfruttando vie differenti. In tutti e tre i casi l'approccio per la creazione di un tessuto parte comunque dalla definizione delle caratteristiche dei fili che lo andranno a comporre.

Per definire un filo e quindi a un tessuto è necessario specificare alcune proprietà geometriche del filo: Il percorso del filo viene specificato tramite una serie di "Master Nodes" che individuano le coordinate dei punti in cui passerà il centro del filo. In tessuti che possiedono un'unità ripetitiva, ad esempio, si rende necessario indicare come Master Nodes almeno tutti punti in cui passa il filo per definire la stessa unità. Una volta definiti i Master Nodes TexGen calcola la posizione del filo tra questi nodi utilizzando una funzione di interpolazione (che può essere cubica o lineare) in alcune posizioni intermedie chiamate "Slave Nodes". Il numero di questi nodi secondari è definito dal parametro di risoluzione del filo.

Definito il percorso si passa a descrivere le sezioni trasversali, che vanno specificate per tutta la lunghezza del filo, ma spesso vengono mantenute costanti lungo tutta la sua lunghezza: questa è infatti l'impostazione di default, altrimenti è possibile definire specifiche sezioni 2D nei Master Nodes o in posizioni specifiche lungo il percorso del filo. TexGen permette di specificare sezioni 2D di forma ellittica, lenticolare, rettangolare e poligonale o una sezione ibrida combinazione delle precedenti. La sezione definita è posizionata nei Master Nodes o nelle posizioni specificate e orientata secondo un piano perpendicolare alla tangente al filo in quel punto. La

funzione di interpolazione a questo punto posiziona delle sezioni trasversali nei nodi secondari attraverso le quali si genera la mesh superficiale unendo i punti sul bordo delle sezioni adiacenti.

Il filo così definito è generato solo per la lunghezza specificata attraverso i Master Nodes, ma per creare tessuti che abbiano un'estensione maggiore della loro unità ripetitiva è possibile utilizzare dei vettori che ripetano i nodi dei fili nelle direzioni desiderate, cioè lungo x e y per tessuti 2D.

La porzione del tessuto che si desidera esportare ed analizzare ulteriormente è definita dal dominio, definito come un insieme di piani che formano un poligono complesso. Questo può includere l'unità ripetitiva o una sezione maggiore, a seconda dello studio che si desidera condurre sulla geometria creata.

2.2.1 Weave Wizard

La modalità Wizard è la modalità più rapida ed intuitiva da utilizzare per generare geometrie semplici automaticamente. [20]

Attraverso questa modalità è possibile generare tessuti 2D o 3D automaticamente utilizzando la classe CTextile, che contiene le informazioni necessarie per caratterizzare il filo (CYarn).

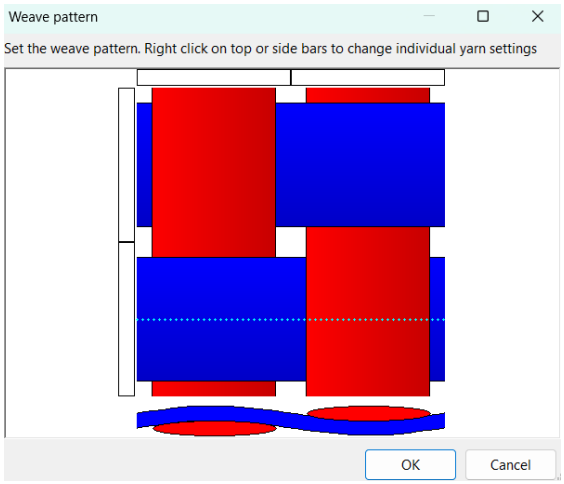


Figura 2.6: Schermata di definizione del pattern del tessuto in modalità Wizard

Per un tessuto 2D la modalità Wizard definisce una griglia, mostrata in figura 2.6, nella quale è possibile definire la posizione reciproca dei fili di trama e di ordito in ogni punto in cui questi si incontrano: questi punti non sono altro che i Master Nodes, nei quali i fili di ordito possono passare sopra o sotto i fili di trama. Nella modalità Wizard, perciò, i Master Nodes sono sempre posti in corrispondenza degli incroci dei fili. Le coordinate dei Master Nodes dipendono dallo spessore del filo e dalla distanza tra due fili adiacenti, cioè dalla densità di fili di trama e di

ordito. Tutte queste informazioni vengono definite in una finestra iniziale insieme al numero di fili e ad altre impostazioni necessarie per includere un dominio o per creare tessuti 3D, mostrata in figura 2.7.

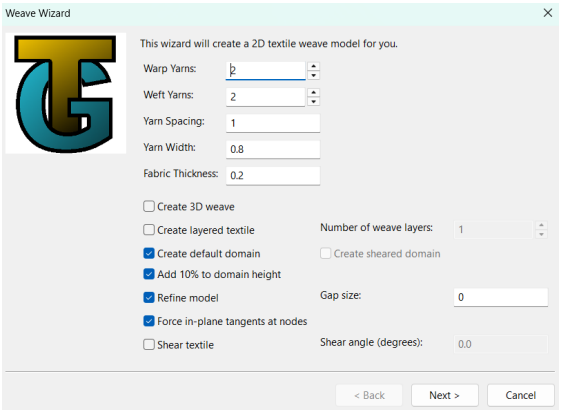


Figura 2.7: Schermata iniziale per la creazione di un tessuto in modalità Wizard

Come detto, la modalità permette di creare tessuti in modo rapido, ma a scapito della complessità realizzabile: ad esempio non è possibile definire tessuti con distanze tra fili di trama e di ordito differenti, ma solo indicare una distanza tra fili che sarà la stessa per entrambi i set. La sezione trasversale, inoltre, può essere definita solo come costante e di forma ellittica attraverso i parametri di larghezza e spessore del filo.

In figura 2.8 è rappresentato una semplice unità ripetitiva di un tessuto con trama PW generato in modalità Wizard con le caratteristiche definite nelle figure 2.6 e 2.7.

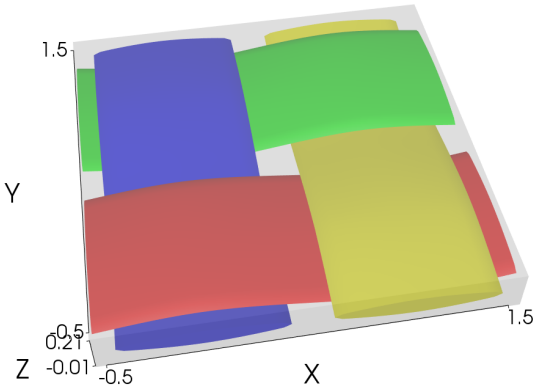


Figura 2.8: Tessuto PW creato in modalità Wizard

2.2.2 Menu Modeler della Graphic User Interface (GUI)

L'utilizzo del menu Modeler dell'interfaccia grafica garantisce maggiore libertà e manualità nella realizzazione di tessuti più complessi. [20]

In questa modalità, una volta definito un tessuto vuoto, questo viene poi creato partendo dalla definizione del singolo filo. Si definiscono il numero di nodi in cui il filo deve passare e la lunghezza del filo attraverso la finestra mostrata in figura 2.9 e si modificano le proprietà del filo, come la sua sezione, la funzione interpolatoria e i vettori ripetizione attraverso le rispettive finestre all'interno del menu Modeler.

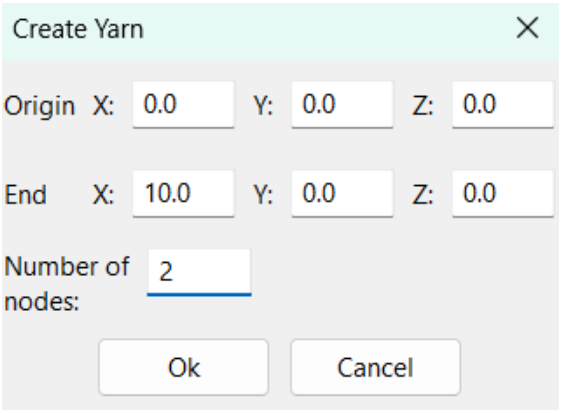


Figura 2.9: Definizione di un filo

Un esempio di filo creato con la modalità Modeler è mostrato in figura 2.10.

A questo punto si ripete l'operazione per tutti i fili che compongono il tessuto, o almeno la sua unità ripetitiva.

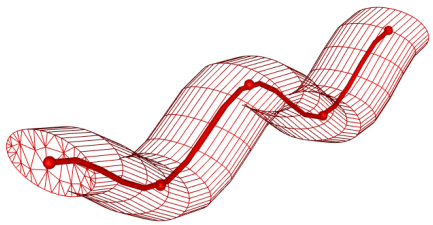


Figura 2.10: Filo creato attraverso la GUI

Utilizzando il Modeler è possibile sfruttare una console Python che permette di utilizzare alcuni comandi per modificare la geometri dei fili oppure di registrare come comando Python le operazioni eseguite tramite i menu dell'interfaccia grafica.

Codice Python

La creazione di un tessuto a partire dai fili può avvenire anche attraverso un codice Python, oppure si possono modificare le proprietà di un tessuto già generato utilizzando dei comandi Python. [20]

Questo è possibile perché TexGen, pur essendo scritto in C++, utilizza un'interfaccia grafica (GUI) i cui elementi possono essere eseguiti anche utilizzando uno o più comandi Python. Richiamando i vari moduli in cui è suddiviso il software si può quindi generare un tessuto seguendo le stesse modalità definite nella modalità Modeler, ma con maggiore flessibilità nel caso si lavori con geometrie più complesse.

Una volta scritto il codice in un ambiente separato è possibile caricarlo su TexGen nell'apposita sezione e il tessuto generato verrà visualizzato insieme all'eventuale dominio, se specificato nel codice.

Questa è stata la modalità scelta in questa tesi per generare le geometrie dei tessuti. Il codice completo per la generazione di tutti i tessuti è fornito nell'appendice A.

Il codice permette di creare geometrie di tessuti formati da cinque fili di trama e cinque fili di ordito a partire dai dati geometrici forniti, come densità di fili di trama e ordito, diametro dei fili, Poisson ratio e reduction factor. Questi dati sono necessari per definire la sezione trasversale costante dei fili, la spaziatura e la posizione reciproca degli stessi nel tessuto.

Viene anche definito un dominio, i cui piani tagliano i fili più esterni di trama e di ordito a metà, e viene assegnato un nome al tessuto. Per distinguere ogni armatura, combinazione di densità di fili di trama e di ordito, Poisson ratio e reduction factor si fa riferimento ad ogni geometria tramite un nome che indica "Armatura-D_1-D_2-ν-r_f".

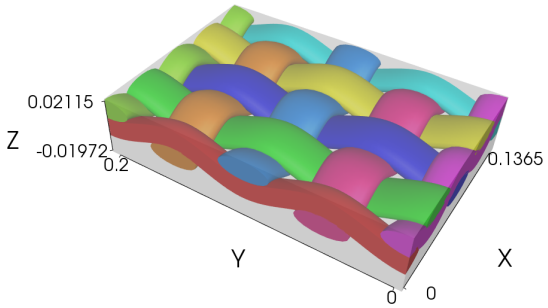


Figura 2.11: Tessuto PW generato dal codice

In Figura 2.11viene mostrato un tessuto PW generato dal codice riportato in appendice.

L'esportazione della superficie della geometria è stata gestita tramite codice per automatizzare il processo. Questo rappresenta uno dei vantaggi principali dello scripting rispetto all'utilizzo dell'interfaccia grafica.

Il formato scelto in questa tesi per esportare i tessuti ed inserirli in un caso OpenFOAM è il formato .stl, derivante dalla mesh superficiale, rappresentata tramite elementi poligonali.

2.3 CFD - OpenFOAM

2.3.1 Metodo ai volumi finiti

Il Metodo ai Volumi Finiti (FVM) è un metodo numerico ampiamente utilizzato per la discretizzazione delle equazioni differenziali alle derivate parziali che esprimono le leggi di conservazione su un volume differenziale. Nei problemi di fluidodinamica computazionale le equazioni da risolvere derivano dall'equazione di trasporto per una determinata grandezza ϕ, formulata come di seguito[21]:

$$\rho\frac{\partial\phi}{\partial t} + \rho\nabla(\mathbf{U}\phi) = \nabla\cdot(\Gamma^{\phi}\nabla\phi) + Q^{\phi} \tag{2.8}$$

Nell'equazione riportata ρ è la densità del fluido, U è la velocità del fluido, Γ^{ϕ} è il coefficiente di diffusione e Q^{ϕ} il termine sorgente. Questa equazione non è risolvibile analticamente per la presenza di derivate sia temporali (nulle nel caso stazionario) che spaziali.

Il metodo FVM consente di riformulare le equazioni continue in un sistema di equazioni algebriche discrete definite su un dominio suddiviso in sotto-volumi finiti, detti celle, in cui viene suddiviso lo spazio. All'interno di ciascuna cella, una generica proprietà ϕ mantiene il suo valore costante e pari a quello che assume nel centroide della cella stessa. Ogni cella assume quindi il ruolo di volume di controllo, all'interno del quale si applicano in forma integrale le leggi di conservazione. Ottenuto un set di equazioni algebriche, questo viene risolto per ogni cella in modo da trovare il valore che la variabile dipendente assume in ogni sotto-volume in cui è stato discretizzato lo spazio. Proseguendo dall'equazione 4.1, possiamo ottenere l'equazione per il caso stazionario in cui il termine transitorio è nullo:

$$\rho\nabla(\mathbf{U}\phi) = \nabla\cdot(\Gamma^{\phi}\nabla\phi) + Q^{\phi} \tag{2.9}$$

A questo punto, proseguendo con il metodo FVM si integra l'equazione 2.9 sul volume di controllo della cella V_C. Vengono quindi integrati il termine convettivo $\rho\nabla(U\phi)$, quello diffusivo $\nabla\cdot(\Gamma^{\phi}\nabla\phi)$ e quello sorgente Q^{ϕ} ottenendo:

$$\int_{V_c}\rho\nabla\cdot(\mathbf{U}\phi)\,\mathrm{d}V = \int_{V_c}\nabla\cdot(\Gamma^{\phi}\nabla\phi)\,\mathrm{d}V + \int_{V_c}Q^{\phi}\,\mathrm{d}V \tag{2.10}$$

L'elemento C sul quale si integra l'equazione 2.9 è rappresentato in figura 2.12.

Gli integrali di volume dei termini convettivi e diffusivi possono essere sostituiti con degli integrali di superficie utilizzando il teorema della divergenza e giungendo a:

$$\oint_{\partial V_C}(\rho\mathbf{U}\phi)\cdot d\mathbf{S} = \oint_{\partial V_C}\left(\Gamma^{\phi}\nabla\phi\right)\cdot d\mathbf{S} + \int_{V_C}Q^{\phi}dV \tag{2.11}$$

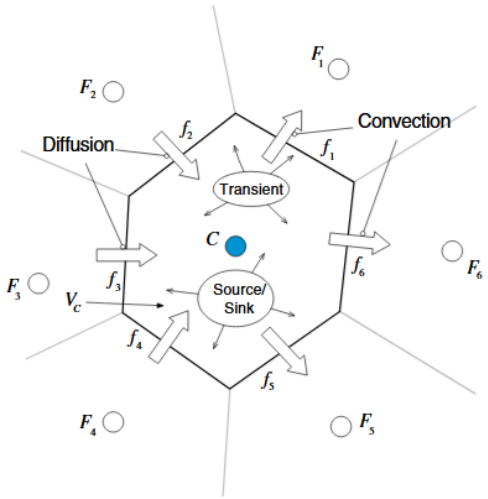


Figura 2.12: Elemento discreto

I flussi convettivi $\mathbf{J}^{\phi,C}$ e diffusivi $\mathbf{J}^{\phi,D}$ mostrati in figura 2.12 possono essere trattati come un unico flusso totale $\mathbf{J}^{\phi}$

$$\mathbf{J}^{\phi,C} = \rho \mathbf{U} \phi \tag{2.12}$$

$$\mathbf{J}^{\phi,D} = -\Gamma^{\phi} \nabla \phi \tag{2.13}$$

$$\mathbf{J}^{\phi} = \mathbf{J}^{\phi,C} + \mathbf{J}^{\phi,D} \tag{2.14}$$

I primi due termini dell'equazione 2.11 possono quindi essere espressi come:

$$\oint_{\partial V_C} (\rho \mathbf{U} \phi) \cdot d\mathbf{S} = \oint_{\partial V_C} \mathbf{J}^{\phi,C} \cdot d\mathbf{S} = \sum_{f \sim \text{ faces } (V_C)} \left(\int_f (\rho \mathbf{U} \phi) \cdot d\mathbf{S} \right) \tag{2.15}$$

$$\oint_{\partial V_C} \left(\Gamma^{\phi} \nabla \phi \right) \cdot d\mathbf{S} = \oint_{\partial V_C} \mathbf{J}^{\phi,D} \cdot d\mathbf{S} = \sum_{f \sim \text{ faces } (V_C)} \left(\int_f \left(\Gamma^{\phi} \nabla \phi \right) \cdot d\mathbf{S} \right) \tag{2.16}$$

$$\oint_{\partial V_C} \mathbf{J}^{\phi} \cdot d\mathbf{S} = \sum_{f \sim \text{ faces } (V_C)} \left(\int_f \mathbf{J}_f^{\phi} \cdot d\mathbf{S} \right) \tag{2.17}$$

Si nota come i flussi di superficie siano calcolati sulle facce dell'elemento C e non integrati al suo interno. Questo garantisce al metodo ai volumi finiti un'importante proprietà: l'essere conservativo. È importante che il medodo sia conservativo in quanto la proprietà ϕ è generalmente una quantità conservativa (ad esempio la massa o l'energia) ed è fondamentale che questa si conservi anche nel dominio discretizzato. Nel metodo ai volumi finiti questo è garantito dal momento che i flussi integrati su una faccia dipendono dai valori che la proprietà possiede negli elementi adiacenti alla faccia stessa: il flusso uscente da un elemento è pari a quello

entrante nell'elemento adiacente attraverso la stessa faccia, e quando un metodo possiede questa proprietà è detto conservativo. A questo punto, per continuare con la discretizzazione, per ogni faccia che l'elemento C possiede è necessario calcolare l'integrale di superficie, così come è necessario calcolare l'integrale di volume per il termine sorgente. Per gli integrali di superficie si ricorre ad un metodo di quadratura Gaussiano. Per una generica faccia f dell'elemento C si ottiene:

$$\int_f \mathbf{J}^\phi \cdot d\mathbf{S} = \int_f \left(\mathbf{J}^\phi \cdot \mathbf{n}\right) dS = \sum_{ip \sim ip(f)} \left(\mathbf{J}^\phi \cdot \mathbf{n}\right)_{ip} \omega_{ip} S_f \tag{2.18}$$

Dove ip è un punto di integrazione tra quelli scelti sulla superficie $ip(f)$ e w_{ip} è il peso. La scelta più semplice è quella che sfrutta un solo punto di integrazione collocato nel centroide della cella e $w_{ip} = 1$. Si ottiene così un'approsimazione con accuratezza del secondo ordine utilizzabile in una o tre dimensioni. Una maggiore accuratezza è ottenibile aumentando il numero di punti di integrazione: questo aumenta però il costo computazionale. Nel metodo ai volumi finiti si è soliti utilizzare un solo punto di integrazione sia per gli integrali di superficie che per quelli di volume necessari per il termine sorgente:

$$\int_V Q^\phi dV = \sum_{ip \sim ip(V)} \left(Q^\phi_{ip} \omega_{ip} V\right) \tag{2.19}$$

In questo caso si nota come il punto di integrazione ip è scelto all'interno del volume ($ip(V)$) e non sulla faccia ($ip(f)$), come invece accade nell'equazione 2.18.

Si giunge infine all'equazione stazionaria semi-discreta per l'elemento C, ottenendo un secondo ordine di accuratezza e un giusto compromesso tra accuratezza e flessibilità, mantenendo un costo computazionale relativamente basso:

$$\sum_{f \sim nb(C)} \left(\rho \mathbf{U}\phi - \Gamma^\phi \nabla \phi\right)_f \cdot \mathbf{S}_f = Q^\phi_C V_C \tag{2.20}$$

L'equazione 2.20 deve ancora essere trasformata in un'equazione algebrica: si vuole infatti giungere ad una formulazione che esprima i flussi in funzione dei valori che la variabile assume nelle celle adiacenti, in particolare nei centroidi.

La figura 2.14 mostra schematicamente come il flusso totale $FluxT_f$ attraverso una generica faccia f possa essere espresso come come la somma di due contributi lineari che dipendono dal valore di ϕ calcolati nei centroidi delle celle che condividono la faccia (C ed F) e di un contributo non lineare $FluxV_f$ che non può essere espresso in termini di ϕ_C e ϕ_F:

$$\mathbf{J}^\phi_f \cdot \mathbf{S}_f = FluxT_f = FluxC_f \phi_C + FluxF_f \phi_F + FluxV_f \tag{2.21}$$

$FluxC_f$ e $FluxF_f$ sono i coefficienti di linearizzazione e dipendono (così come $FluxV_f$) dal termine discretizzato e dallo schema di discretizzazione utilizzato.

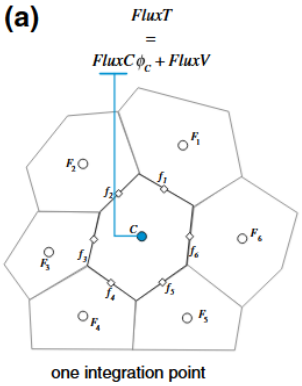


Figura 2.13: Integrazione volumica del termine sorgente con un punto di integrazione

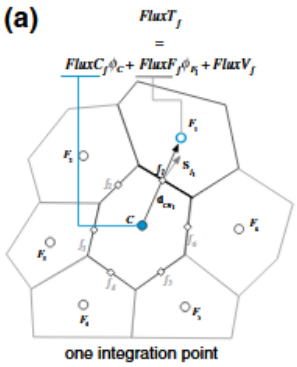


Figura 2.14: Integrazione superficiale dei flussi con un punto di integrazione

Sostituendo l'equazione 2.21 nel primo membro dell'equazione 2.20 e ripetendo l'operazione per tutti le facce $nb(C)$ della cella si ottiene:

$$\sum_{f\sim nb(C)} \left(\mathbf{J}_f^{\phi} \cdot \mathbf{S}_f\right) = \sum_{f\sim nb(C)} (FluxT_f) = \sum_{f\sim nb(C)} (FluxC_f\phi_C + FluxF_f\phi_F + FluxV_f) \tag{2.22}$$

Anche il flusso volumico va riscritto ed espresso anche in questo caso come la somma di un contributo lineare funzione del valore di ϕ nel centroide della cella C e di un termine non lineare:

$$Q_C^{\phi} V_C = FluxT = FluxC\phi_C + FluxV \tag{2.23}$$

Inserendo quest'ultima equazione, in cui $FluxC$ è nullo nel caso di termini sorgenti costanti, nell'equazione 2.20 insieme all'equazione si ottiene:

$$a_C\phi_C + \sum_{F\sim NB(C)} (a_F\phi_F) = b_C \tag{2.24}$$

Che rappresenta l'equazione algebrica in funzione del valore della proprietà ϕ nei centroidi delle celle adiacenti. I coefficienti a_C, a_F, b_C dell'equazione 2.24 sono espressi in funzione dei coefficienti di linearizzazione secondo le equazioni:

$$a_C = \sum_{f\sim nb(C)} (FluxC_f - FluxC) \tag{2.25}$$

$$a_F = FluxF_f \tag{2.26}$$

$$b_C = -\sum_{f\sim nb(C)} (FluxV_f + FluxV) \tag{2.27}$$

Estendendo questo processo a tutto il dominio computazionale discretizzato è possibile ottenere il valore della proprietà ϕ in ogni cella risolvendo un sistema di equazioni algebriche.

2.3.2 Discretizzazione spaziale

Con il metodo dei volumi finiti si è quindi scelto di integrare le equazioni di trasporto sui volumi in cui è suddiviso il dominio computazionale, andando a calcolare i flussi attraverso le facce di questi volumi e ottenendo un'equazione semi-discretizzata. All'interno dell'equazione di trasporto è necessario discretizzare sia il termine diffusivo, rappresentato dal Laplaciano, che il termine convettivo, rappresentato dal gradiente nello spazio. In un'equazione di trasporto si possono trovare entrambi i termini, che vanno trattati separatamente in quanto rappresentano ciascuno un fenomeno fisico a sè stante. Ognuno necessita perciò di considerazioni particolari per definire i corretti profili di interpolazione.[22]

Termine diffusivo

L'equazione stazionaria di diffusione è data da:

$$-\nabla \cdot \left(\Gamma^\phi \nabla\phi\right) = Q^\phi \tag{2.28}$$

Nella sezione precedente si è giunti, eliminando il primo gradiente, ad una prima forma discretizzata (equazione 2.20). Trattando solo il termine diffusivo, questa diventa:

$$\sum_{f\sim nb(C)} \left(-\Gamma^\phi \nabla\phi\right)_f \cdot \mathbf{S}_f = Q_C^\phi V_C \tag{2.29}$$

Espandendo la sommatoria, la precedente equazione può essere scritta come:

$$(-\Gamma^{\phi}\nabla\phi)_e \cdot \mathbf{S}_e + (-\Gamma^{\phi}\nabla\phi)_w \cdot \mathbf{S}_w + (-\Gamma^{\phi}\nabla\phi)_n \cdot \mathbf{S}_n + (-\Gamma^{\phi}\nabla\phi)_s \cdot \mathbf{S}_s = Q_C^{\phi} V_C \quad (2.30)$$

In cui per ogni cella il primo membro è dato dalla somma dei contributi dei flussi per le facce ad est, ovest, nord e sud.

Per comodità la trattazione verrà fatta utilizzando una griglia cartesiana uniforme per un caso in 2D, rappresentato in figura 2.15

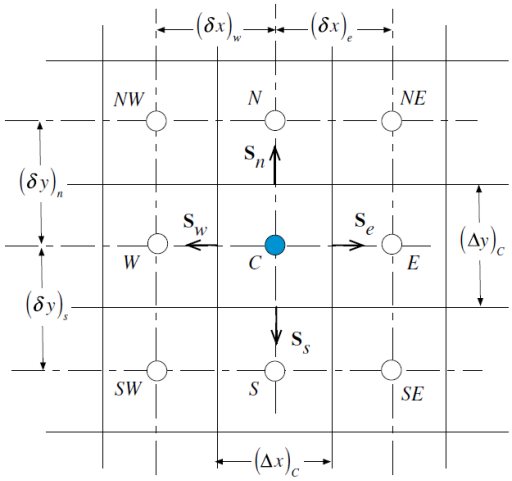


Figura 2.15: Griglia cartesiana uniforme

I flussi diffusivi attraverso le quattro facce di una cella bidimensionali sono calcolati nello stesso modo. Per la faccia est, ad esempio, dal momento che il vettore normale alla faccia è esprimibile come:

$$\mathbf{S}_e = +(\Delta y)_e \, \mathbf{i} = \|\mathbf{S}_e\| \, \mathbf{i} = S_e \, \mathbf{i} \quad (2.31)$$

Allora il flusso diffusivo attraerso la faccia sarà:

$$J_e^{\phi,D} = -(\Gamma^{\phi}\nabla\phi)_e \cdot \mathbf{S}_e = -\Gamma_e^{\phi}\mathbf{S}_e \left(\frac{\partial\phi}{\partial x}\mathbf{i} + \frac{\partial\phi}{\partial y}\mathbf{j}\right)_e \cdot \mathbf{i} = -\Gamma_e^{\phi}(\Delta y)_e \left(\frac{\partial\phi}{\partial x}\right)_e \quad (2.32)$$

In cui è stato esplicitato il gradiente ed è stato svolto il prodotto scalare. Per calcolare la derivata parziale della proprietà si considera una variazione lineare di ϕ tra i centroidi di due celle tra loro adiacenti. Definiti ϕ_C e ϕ_E rispettivamente i valori della generica proprietà nel centroide della cella di riferimento e il valore nel centroide immediatamente ad est allora $\left(\frac{\partial\phi}{\partial x}\right)_e$ non è altro che la pendenza, definita

dal rapporto tra la differenza dei valori di ϕ nelle due celle e la distanza tra i due centroidi adiacenti $(\delta x)_e$:

$$\left(\frac{\partial \phi}{\partial x}\right)_e = \frac{\phi_E - \phi_C}{(\delta x)_e} \tag{2.33}$$

Come per l'equazione 2.21 nel paragrafo precedente, è possibile esprimere il flusso diffusivo attraverso la faccia est come $FluxT_e$, cioè il flusso totale dato da tre contributi:

$$J_e^{\phi,D} = FluxT_e = FluxC_e\,\phi_C + FluxF_e\,\phi_F + FluxV_e \tag{2.34}$$

Utilizzando le equazioni 2.34, 2.32 e 2.33 si giunge a:

$$\begin{aligned} FluxT_e &= -\Gamma_e^{\phi}(\Delta y)_e \frac{(\phi_E - \phi_C)}{\delta x_e} \\ &= \Gamma_e^{\phi}\frac{(\Delta y)_e}{\delta x_e}\,(\phi_C - \phi_E) \\ &= FluxC_e\,\phi_C + FluxF_e\,\phi_E + FluxV_e \end{aligned} \tag{2.35}$$

Definendo:

$$gDiff_e = \frac{(\Delta y)_e}{\delta x_e} = \frac{\|\mathbf{S}_e\|}{\|\mathbf{d}_{CE}\|} = \frac{S_e}{d_{CE}} \tag{2.36}$$

Dove d_{CE} è la distanza tra i centroidi delle due celle. Adesso è possibile calcolare i valori di $FluxC_E$, $FluxF_E$, $FluxV_E$ in funzione di parametri noti:

$$\begin{aligned} FluxC_e &= \Gamma_e^{\phi} gDiff_e \\ Flux_e &= -\Gamma_e^{\phi} gDiff_e \\ FluxV_e &= 0 \end{aligned} \tag{2.37}$$

Questo procedimento svolto per la faccia est è svolto analogamente per le altre tre facce (i risultati saranno identici a quelli riportati, a meno dei pedici w, n, s e W, N, S da sostituire ad e ed E rispettivamente). L'equazione 2.30 è ora rappresentabile in forma algebrica:

$$a_C\phi_C + \sum_{F \sim NB(C)} a_F\phi_F = b_C \tag{2.38}$$

con

$$\begin{aligned} a_F &= FluxF_f = -\Gamma_f^\phi g Diff_f \\ a_C &= \sum_{f \sim nb(C)} FluxC_f \\ b_C &= Q_C^\phi V_C - \sum_{f \sim nb(C)} FluxV_f \end{aligned} \tag{2.39}$$

Dove F indica gli elementi confinanti con l'elemento C (quindi le celle E, W, N e S) e f indica le facce dell'elemento C (e, w, n e s).

L'approsimazione necessaria per calcolare la derivata con l'equazione 2.33 rende necessario valutare che la nuova equazione mantenga il significato fisico iniziale. Se si considera nullo il termine sorgente il trasporto della proprietà ϕ avviene solo per diffusione. Considerando per semplicità una sola dimensione, allora l'equazione discretizzata diventa:

$$a_C \phi_C + a_E \phi_E + a_W \phi_W = 0 \tag{2.40}$$

dove

$$a_E = -\Gamma_e^\phi g \text{ Diff }_e \quad a_W = -\Gamma_w^\phi g \text{ Diff }_w \quad a_C = -\left(a_E + a_W\right) \tag{2.41}$$

Per la legge di Fourier il trasporto della proprietà ϕ avviene lungo la direzione in cui la proprietà stessa decresce: il valore della proprietà sulle facce deve essere perciò compreso tra i valori della proprietà nei centroidi e questo è sempre vero nel caso di profili lineari, mentre potrebbe non essere vero, ad esempio, nel caso di profili parabolici in cui il valore alla faccia potrebbe essere maggiore o minore di quello in uno dei due centroidi consecutivi. Questo legittima l'approssimazione della derivata con la pendenza della retta, e l'utilizzo di un profilo lineare per la variazione della proprietà ϕ tra le celle.

Termine convettivo

Il termine convettivo è rappresentato, in una generica equazione di trasporto, dall'operatore divergenza.[23] Questo termine può essere discretizzato considerando un profilo simmetrico lineare, in modo simile a quando è stato fatto per il termine diffusivo, ma generalmente questo porta a risultati scadenti. Il profilo solitamente utilizzato è detto *Upwind* e presenta un'accuratezza del primo ordine. Per aumentare l'accuratezza è possibile utilizzare profili di ordine superiore che diminuiscono l'errore di discretizzazione rispetto ad un profilo lineare, ma possono dare origine ad un altro tipo di errore, detto errore di dispersione.

Per semplicità viene trattato il caso di un problema stazionario con convezione e diffusione in una dimensione. In questo caso la generica equazione di trasporto può essere espressa come:

$$\frac{d(\rho U\phi)}{dx} - \frac{d}{dx}\left(\Gamma^{\phi}\frac{d\phi}{dx}\right) = 0 \tag{2.42}$$

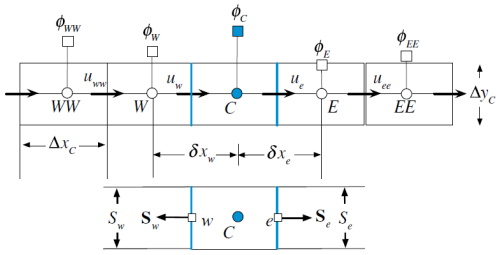


Figura 2.16: Griglia monodimensionale

Questa viene risolta numericamente integrando sull'elemento monodimensionale rappresentato in figura 2.16. Utilizzando il teorema della divergenza e riscrivendo l'equazione 2.42 in termini di flussi convettivi e diffusivi si ottiene:

$$\sum_{f\sim nb(C)} \left(\rho U\phi\mathbf{i} - \Gamma^{\phi}\frac{d\phi}{dx}\mathbf{i}\right)_f \cdot \mathbf{S}_f = 0 \tag{2.43}$$

Siccome le superfici opposte hanno normali opposte, nel caso monodimensionale la precedente equazione può essere riscritta come:

$$\left[(\rho U\Delta y\phi)_e - \left(\Gamma^{\phi}\frac{d\phi}{dx}\Delta y\right)_e\right] - \left[(\rho U\Delta y\phi)_w - \left(\Gamma^{\phi}\frac{d\phi}{dx}\Delta y\right)_w\right] = 0 \tag{2.44}$$

Considerando noto il valore di U nelle facce delle celle e che le derivate possono essere discretizzate secondo i metodi descritti in precedenza, rimangono da calcolare i valori della proprietà ϕ nelle facce e e w in funzione dei valori nei nodi adiacenti, cioè nei centroidi.

Lo schema *Upwind* è lo schema di advezione maggiormente utilizzato quando si ha una direzione principale del flusso. Il valore della proprietà nella faccia dipende in questo caso dalla direzione del flusso. Facendo riferimento alla figura 2.16 il valore di ϕ nelle facce e e w sono quindi date da:

$$\phi_e = \begin{cases} \phi_C & \text{se } \dot{m}_e > 0 \\ \phi_E & \text{se } \dot{m}_e < 0 \end{cases} \qquad \phi_w = \begin{cases} \phi_C & \text{se } \dot{m}_w > 0 \\ \phi_W & \text{se } \dot{m}_w < 0 \end{cases} \tag{2.45}$$

Dove $\dot{m}_e$ ed $\dot{m}_w$ sono le portate massiche nelle facce e ed w

2.3.3 Risolvere il campo di moto e di pressione

Generalmente il campo di velocità non è noto a priori in un problema di fluidodinamica computazionale, ma è proprio ciò che si vuole risolvere. In particolare se si intende risolvere il campo di velocità e di pressione è necessario risolvere l'equazione di continuità e quelle di Navier-Stokes. [24] Per fluidi incomprimibili vi è poi una forte correlazione tra velocità e pressione, e la pressione non può essere calcolata analiticamente a partire dalle equazioni di continuità o di Navier-Stokes, riportate di seguito:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{2.46}$$

$$\frac{\partial}{\partial t}[\rho \mathbf{U}] + \nabla \cdot \{\rho \mathbf{U}\mathbf{U}\} = -\nabla p + \nabla \cdot \left\{\mu \left[\nabla \mathbf{U} + (\nabla \mathbf{U})^{\mathrm{T}}\right]\right\} + \mathbf{f}_b \tag{2.47}$$

L'algoritmo utilizzato per risolvere il campo di moto e di pressione è detto $SIMPLE$ (Semi Implicit Method for Pressure Linked Equations). A partire dalle equazioni di continuità e di Navier-Stokes espresse nella seguente forma:

$$\sum_{f \sim nb(C)} \dot{m}_f = 0 \tag{2.48}$$

$$a_e^u U_e + \sum_{f \sim NB(e)} a_f^u U_f = b_e^u - V_e \left(\frac{\partial p}{\partial x}\right)_e \tag{2.49}$$

Questo metodo inizia da una stima iniziale dei campi di pressione e velocità ($u^{(n)}$ e $p^{(n)}$), di solito posti nulli o pari alle condizioni iniziali, e ad ogni iterata risolve le Navier-Stokes per trovare il campo di velocità denotato con u^*. Essendo u^* soluzione dell'equazione 2.49 allora vale:

$$a_e^u U_e^* + \sum_{f \sim NB(e)} a_f^u U_f^* = b_e^u - V_e \left(\frac{\partial p^{(n)}}{\partial x}\right)_e \tag{2.50}$$

Si nota però che il valore di pressione è $p^{(n)}$, cioè quello della iterata precedente. U^* infatti potrebbe non soddisfare l'equazione di continuità e per fare in modo che questa risulti invece sempre soddisfatta si introduce una correzione nei campi di velocità e pressione (U' e p').

Iterativamente il metodo genera dei campi di moto e di pressione che soddisfano in modo consecutivo le Navier-Stokes e l'equazione di continuità. Le soluzioni finali alle quali il metodo tende all'aumentare delle iterate sono un campo di moto e di pressione che soddisfano contemporaneamente entrambe le equazioni. In particolare il metodo:

1. Inizia con una stima dei campi $U^{(n)}$ e $p^{(n)}$

2. Risolve l'equazione 2.49 per ottenere un nuovo campo di velocità U_f^*
3. Aggiorna le portate massiche utilizzando U_f^* ed ottenendo $\dot{m}_f^*$
4. Risolve l'equazione di correzione della pressione utilizzando la nuova portata massica e ottenendo il campo di pressione corretto p'
5. Aggiorna i campi di pressione e di velocità così che questi possano ora soddisfare l'equazione di continuità:

$$\begin{aligned} U_f^{**} &= U_f^* + U_f' \quad U_f' = -D_f^u \left(\frac{\partial p'}{\partial x}\right)_f \\ p_C^* &= p_C^{(n)} + p_C' \\ \dot{m}_f^{**} &= \dot{m}_f^* + \dot{m}_f' \quad \dot{m}_f' = -\rho_f D_f^u \Delta y_f \left(\frac{\partial p'}{\partial x}\right)_f \end{aligned} \tag{2.51}$$

6. imposta U^{**} e p^* come i nuovi valori di $U^{(n)}$ e $p^{(n)}$
7. ripete dal punto 2 fino a quando non viene raggiunta la convergenza

2.4 Calcolo della permeabilità

2.4.1 Misura della permeabilità all'aria nei tessuti

La permeabilità all'aria dei tessuti è una proprietà fondamentale sia per quanto riguarda il comfort termico dell'abbigliamento che per applicazioni dei tessuti utilizzati per realizzare, ad esempio, filtri o airbag.

Sperimentalmente la permeabilità è misurata in accordo con la norma ISO 9237:1995. [11]

Questa norma è applicabile a diverse tipologie di tessuto (compresi i tessuti "non tessuti") e definisce la permeabilità come la velocità di un flusso di aria che attraversa perpendicolarmente un tessuto sotto specifiche condizioni di area del campione, caduta di pressione e tempo. L'area del provino è fissata per standardizzare le misure e viene imposta una differenza di pressione a seconda della tipologia di tessuto analizzata: l'aria, passando attraverso il tessuto, determinerà una differenza di pressione ai suoi capi. Più precisamente, la norma impone:

1. Un'area di prova di 20 cm^2.
2. Una caduta di pressione di 100 Pa nel caso di tessuti destinati all'abbigliamento e di 200 Pa nel caso di tessuti industriali.
3. Atomsfera come definito nella norma ISO 139 durante le fasi di precondizionamento, condizionamento e test.

Qualora non fosse possibile seguire le condizioni indicate è possibile modificare l'area del provino a nuovi valori pari a 5, 50 o 100 cm^2 e utilizzzare valori di ΔP pari a 50 o 500 Pa.

Nell'articolo di Zupin et al. [10] viene utilizzato un Air Permeability tester FX 3300 Labotester III per misurare sperimentalmente la permeabilità dei tessuti. Questa apparecchiatura, mostrata in figura 2.17, risulta in accordo con la normativa, che richiede:

1. Un portacampione circolare in frado di supportare le dimensioni specificate.
2. Un sistema di bloccaggio che non deformi il campione.
3. Un manometro per misurare la caduta di pressione.
4. Un flussometro, o uno strumento analogo in grado di misurare la portata di aria che determina la differenza di pressione desiderata attraverso il tessuto.

Sia il manometro che il flussometro devono avere un'accuratezza del 2%.

La norma prevede il calcolo della permeabilità R attraverso l'equazione:

Figura 2.17: Air Permeability Tester Model FX 3300-IV [25]

$$R = \frac{\bar{q}_v}{A} \times 167 \tag{2.52}$$

Dove $\bar{q}_v$ è la media aritmetica della portata di aria, misurata il L/min, A è l'area del provino, misurata in cm^2 e 167 è un fattore di conversione per passare dai $L/(min \cdot cm^2)$ ai mm/s.

I risultati sperimentali di permeabilità per campioni di 20 cm^2 e ΔP pari a 200 Pa sono tuttavia riportati nell'articolo di Zupin et al. ed espressi in $L/(s \cdot m^2)$. Va notato che la velocità del flusso d'aria è spesso espressa in mm/s. La conversione è diretta, poiché 1 $L/(s \cdot m^2)$ equivale a 1 mm/s, o a 0.001 m/s.

2.4.2 Modelli predittivi

La permeabilità nei tessuti è analizzabile su due diverse scale. Un tessuto infatti è un mezzo poroso a doppia scala, dal momento che presenta micro-pori tra le fibre che compongono il filo e meso-pori tra i fili. Il fluido può quindi fluire con continuità tra i canali costituiti dai pori del tessuto e attraverso i fili, incontrando minor resistenza al passaggio nel primo caso e maggior resistenza nel caso debba passare attraverso le fibre del filo. [26]

Descrivendo i tessuti, e quindi i fili, come mezzi porosi, la micro-permeabilità del filo potrebbe essere valutata in linea teorica attraverso l'equazione di Kozeny-Carman, ma questa è valida solo mezzi porosi isotropi e quindi non utilizzabile per rinforzi per compositi come i tessuti, come evidenziato nello studio di B.R. Gebart. [27]

Lo stesso Gebart è quindi giunto alla formulazione di due equazioni empiriche che permettono di valutare la permeabilità dei fili, considerati come composti

da fibre disposte in modo parallelo tra loro, a seconda se il flusso sia parallelo o perpendicolare ad essi.

$$K_{\parallel} = \frac{8R^2}{c} \frac{(1 - V_f)^3}{V_f^2} \tag{2.53}$$

$$K_{\perp} = C_1 \left(\sqrt{\frac{V_{f,\max}}{V_f}} - 1 \right)^{\frac{5}{2}} R^2 \tag{2.54}$$

Dove R è il raggio della fibra, V_f è la frazione volumica e $V_{f,max}$ è il suo massimo valore raggiungibile in base all'impaccamento scelto dalle fibre, mentre c e C_1 sono costanti anch'esse dipendenti dal tipo di impaccamento.

Per un arrangiamento quadratico delle fibre si ottengono i seguenti valori:

Arrangiamento quadratico delle fibre	
C_1	$\frac{16}{9\pi\sqrt{2}}$
$V_{f,\max}$	$\frac{\pi}{4}$

Tabella 2.1: Valori globali delle costanti per il calcolo di $K_{\perp}$

Per quanto riguarda il valore di frazione volumica, in letteratura si trovano valori che variano tra 0.4 e 0.6 [28], che inseriti nell'equazione 2.54 con i valori della tabella 2.1, insieme ad un valore di 8 µm per il raggio della fibra, restituiscono due valori estremi di micropermeabilità $K_{\perp}$, pari a $2.61 \cdot 10^{-12}$ m^2 e $2.02 \cdot 10^{-13}$ m^2.

L'articolo di Zupin et al. fornisce, oltre ai valori dei diametri dei fili utilizzati per generare le geometrie, anche i valori di diametro idraulico per i pori generati dalla disposizione dei fili nelle diverse armature. [10]

In figura 2.18 è rappresentato un poro per un tessuto PW.

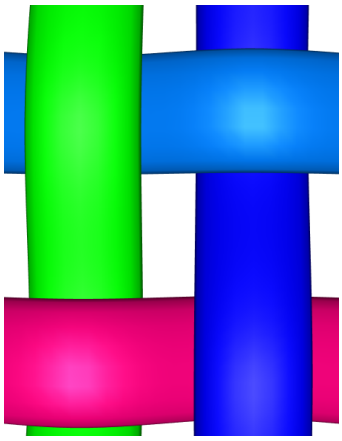


Figura 2.18: Dettaglio di un poro per un tessuto PW

Per le armature utilizzate e le combinazioni di fabric count utilizzate i valori di diametro idraulico risultano:

Armatura	D_1/D_2	D_h (µm)
PW	22/15	274.25
PW	22/20	215.41
PW	29.3/15	151.24
PW	29.3/20	125.21
BW	22/15	274.25
BW	22/20	203.77
BW	29.3/15	134.93
BW	29.3/20	118.94
FR	22/15	293.20
FR	22/20	215.41
FR	29.3/15	151.24
FR	29.3/20	131.44
TW	22/15	255.67
TW	22/20	215.41
TW	29.3/15	126.88
TW	29.3/20	110.94

Tabella 2.2: Diametro idraulico D_h dei fori per le geometrie utilizzate, da Zupin et al. [10]

Il diametro idraulico è definito dal quoziente tra l'area del poro e il perimetro bagnato secondo l'equazione:

$$D_h = \frac{4\,A_{\text{poro}}}{P_{\text{poro}}} \tag{2.55}$$

Nelle quattro armature dei tessuti analizzati i pori sono sempre di forma rettangolare, ma con diverso aspect ratio, cioè diversi rapporti tra le due dimensioni a e b del poro, e perciò l'equazione 2.55 si può esprimere come:

$$D_h = \frac{4\,A_{\text{pori}}}{P_{\text{pori}}} = \frac{4ab}{2(a+b)} = \frac{2ab}{a+b} \tag{2.56}$$

Entrambe le dimensioni del poro sono poi calcolabili a partire da parametri geometrici (i diametri) e costruttivi (le densità di fili di trama e di ordito) del tessuto, secondo le equazioni:

$$a = \frac{1}{D_1} - d_1 \tag{2.57}$$

$$b = \frac{1}{D_2} - d_2 \tag{2.58}$$

Le due geometrie che individuano i valori estremi di diametro idraulico risultano perciò essere il TW-29.3-20 con 110.94 μm e il PW-22-15 insieme al BW-22-15 con 274.25 μm.

I due valori di micro-permeabilità individuati e i due valori estremi di diametro idraulico trovati permettono di posizionarsi all'interno del rettangolo mostrato in figura 2.21, interamente contenuto nella zona in cui il flusso alla micro-scala risulta trascurabile nel grafico ottenuto da Syerko et al [26].

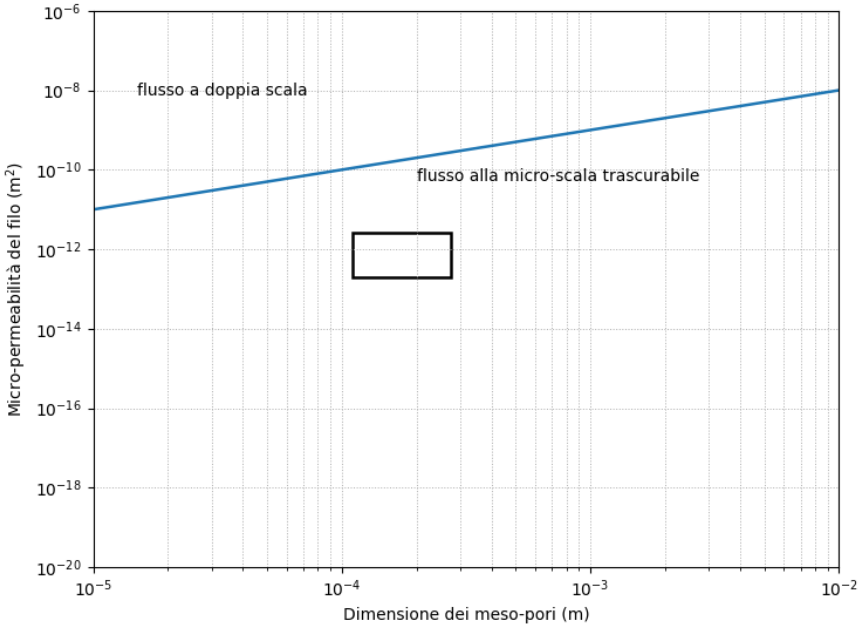


Figura 2.19: Scala del flusso in materiali fibrosi [26]

I tessuti trattati in questa tesi possono perciò essere considerati come composti da fili impermeabili all'aria, che passerà solo attraverso i meso-pori definiti dall'intreccio dei fili.

Capitolo 3

Dettagli Numerici

3.1 Impostazione delle simulazioni con OpenFOAM

3.1.1 Generazione della mesh

In OpenFOAM il dominio computazionale è descritto discretizzandolo con blocchi di celle poliedriche. Le caratteristiche della mesh sono visualizzabili nella directory `constant/polyMesh`, all'interno della quale è possibile leggere i file che descrivono la connettività delle facce, ovvero: `points`, `faces`, `owner`, `neighbour` e `boundary`. [29]

Utilizzando l'utility `blockMesh` si generano delle celle esaedriche e successivamente tramite l'utility `snappyHexMesh` vengono create delle split-hexaedral cells.

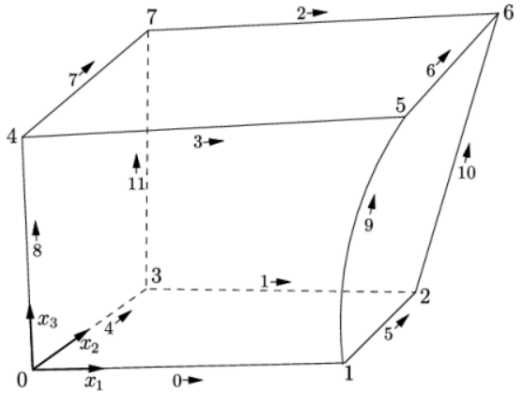


Figura 3.1: Esempio di una cella esaedrica realizzabile con `blockMesh` [29]

`blockMesh`

L'utility `blockMesh` è la prima che viene utilizzata per la divisione del dominio computazionale in celle esaedriche tridimensionali. Un esempio di una cella esaedrica è fornito in figura 3.1.

Il comando legge un dizionario chiamato `blockMeshDict` situato nella directory `system` del caso in analisi. I bordi dei blocchi esaedrici (o di blocchi con meno di otto vertici, qualora si facciano collassare una o più coppie di vertici) creati possono essere definiti da linee rette, archi o spline e la definizione della mesh è data dal numero di celle posizionate in ogni direzione all'interno dei blocchi. Le direzioni degli assi sono definite da una terna destrorsa che segue l'ordine di definizione dei vertici.

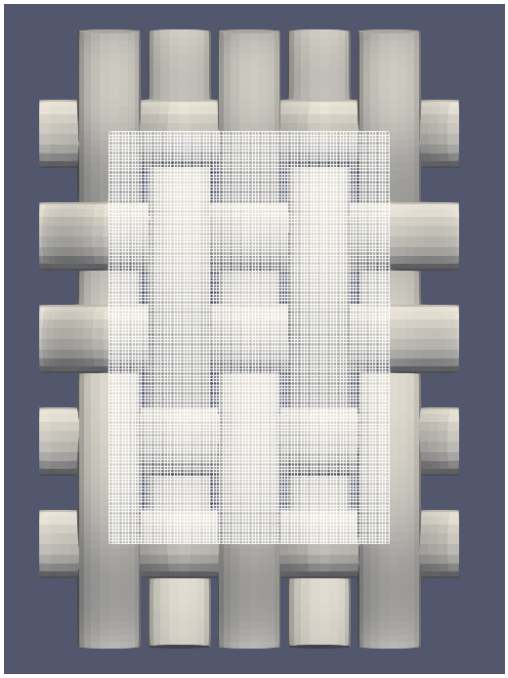
I vertici sono scelti in modo da definire il dominio di controllo dato da un canale a forma di parallelepipedo nel quale passa l'aria che attraversa il tessuto, che è posto trasversalmente e centrato rispetto al tessuto stesso. La lunghezza del canale è scelta pari al doppio dello spessore del tessuto, per non appesantire eccessivamente le simulazioni, ma garantendo comunque il completo sviluppo del profilo di velocità.

Come descritto nel capitolo 2.1.6, ogni tessuto è definito da cinque fili di trama e cinque fili di ordito per descrivere in modo accurato le quattro armature trattate e mantenere una simmetria in tutte le geometrie. La scelta dei vertici è tale da tagliare a metà i fili più esterni sia di trama che di ordito, in modo da permettere il passaggio dell'aria solo all'interno dei fori e non in canali preferenziali ai lati del parallelepipedo definito con `blockMesh`: le dimensioni del canale e quindi i vertici del parallelepipedo dipendono perciò interamente dalle dimensioni del tessuto.

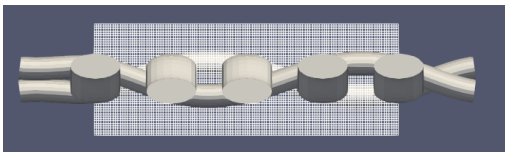
Dal momento che la distanza tra due successivi fili di trama o di ordito è definito come il reciproco della densità dei fili di trama e di ordito (equazione 2.7 nel capitolo 2.1.6) ed essendo tutte le armature descritte da dieci fili in totale è evidente come le dimensioni del tessuto e quindi della faccia trasversale del dominio computazionale siano inversamente proporzionali alle densità D_1 e D_2: tessuti con densità diverse determineranno un canale con dimensioni lungo x e y differenti. La dimensione lungo z del tessuto, cioè il suo spessore, determinerà inoltre la lunghezza del dominio computazionale in quanto impostato pari al doppio dello spessore: al variare della geometria e del reduction factor anch'esso varierà, come indicato in figura 3.7 nel capitolo 3.1.2. Le coordinate degli estremi del dominio computazionale sono quindi legate strettamente alla geometria del tessuto che si analizza per ogni caso.

Per calcolare le dimensioni si è utilizzato quindi uno script Python che, a partire dai diversi file .stl descriventi le singole geometrie, scrive le coordinate dei vertici nel file `caseSetup`, in modo da automatizzare questo passaggio.

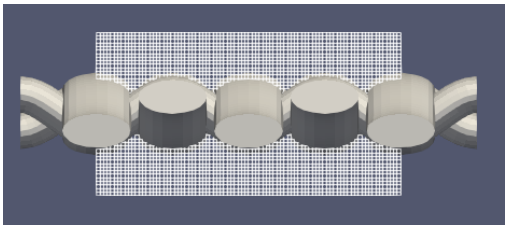
Per scegliere il numero di celle da inserire nelle tre dimensioni del dominio e per fare in modo che queste rimangano costanti rispetto al diametro del filo si utilizza l'equazione 3.1, che verrà discussa nel capitolo 3.1.2.



Piano xy



Piano zy



Piano zx

Figura 3.2: Mesh esaedrica creata da `blockMesh` per un tessuto FR ($D_1 = 29.3$, $D_2 = 20$ e $r_f = 0.58$)

In figura 3.2 vengono mostrate tre viste della mesh esaedrica definita da `blockMesh` per un tessuto FR in cui è possibile apprezzare le dimensioni del dominio computazionale rispetto alle dimensioni del tessuto.

`snappyHexMesh`

Una volta definito il dominio computazionale regolare con `blockMesh` è necessario adattare la mesh attorno a strutture complesse come i tessuti, che vengono letti da OpenFOAM tramite un file .stl. Questo processo iterativo è implementato attraverso l'utility `snappyHexMesh` [30] a partire dalla mesh esaedrica creata in precedenza da `blockMesh`. L'utility legge un dizionario chiamato `snappyHexMeshDict`, situato nella directory `system` del caso e che permette di avanzare nelle tre fasi della definizione della mesh non strutturata finale:

1. Generazione della mesh frastagliata: in questa fase vengono identificate le celle esaedriche (con aspect ratio il più possibile vicino all'unità) che intersecano l'oggetto solido. Queste celle vengono suddivise e alcune vengono rimosse per eliminare il solido dalla descrizione del dominio computazionale. La zona del dominio computazionale nella quale le celle vengono mantenute, cioè la

zona fluida, è definita dalle coordinate di un punto chiamato `locationInMesh` specificato nel sub-dizionario `castellatedMeshControls` all'interno del dizionario `snappyHexMeshDict`. All'interno dello stesso sub-dizionario viene anche definito il parametro `refinementSurfaces` che individua delle zone di rifinizione in prossimità della superficie solida, dove viene aumentato il numero di celle. Il parametro in questione è stato variato durante lo studio di grid independence e la scelta del suo valore è indicata nel capitolo 3.1.2.

2. Snapping superficiale: in questa fase la mesh ricopia più fedelmente la superficie solida rispetto alla mesh a gradini lasciata dal primo step. Vengono infatti proiettati i vertici dei blocchi sulla superficie stessa, rendendo la mesh più liscia. Anche questa fase presenta il suo sub-dizionario `snapControl` con il quale si può controllare il suo comportamento.

3. L'ultima fase è opzionale e non è stata utilizzata nelle simulazioni per questa tesi. Si tratta dell'aggiunta di strati di celle in prossimità della superficie solida per risolvere meglio lo strato limite. Questa fase è gestita da alcuni parametri impostati nel sub-dizionario `addLayersControl`, che vengono però ignorati se `addLayers` è impostato su `false` all'inizio del dizionario.

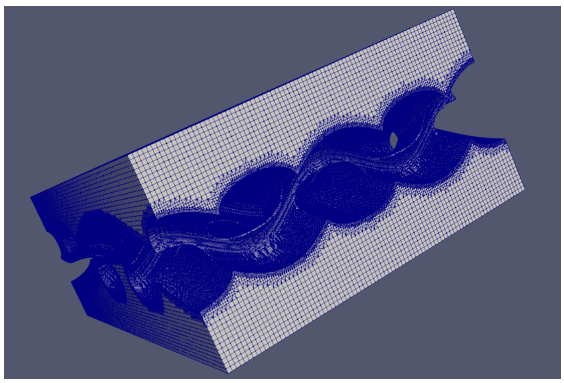


Figura 3.3: Mesh finale per lo stesso tessuto mostrato in figura 3.2 al termine di `snappyHexMesh`

Il dizionario, oltre a definire i parametri per le tre fasi di creazione della mesh, definisce anche i parametri di qualità della mesh finale attraverso il sub-dizionario `meshQualityControls`. I parametri definiti all'interno (o all'interno di un file esterno chiamato `meshQualityDict` richiamato nel sub-dizionario) definiscono alcuni criteri come ad esempio i limiti di skewness, non ortogonalità e aspect ratio delle celle, in modo che se troppe celle non rispettassero questi limiti durante uno dei processi iterativi di `snappyHexMesh` si possa annulare l'ultima modifica effettuata alla mesh.

Fino a questo punto il software ha trattato ogni dimensione inserita come se si trattasse di metri: il dizionario `blockMeshDict` riporta infatti il valore 1 alla voce `convertToMeters` così come le diverse geometrie sono state create con TexGen utilizzando uno script che definisce ogni misura fornita in metri. Questo è utile per la generazione della mesh finale perché le varie operazioni di trimming and fitting per generare la mesh vengono fatte su una geometria che è di fatto ingrandita rispetto alla realtà: queste operazioni fatte su dimensioni troppo piccole potrebbero portare infatti ad alcuni problemi che influenzerebbero la qualità della mesh.

Al termine della creazione della mesh è necessario perciò riscalare di un fattore 100 utilizzando il comando `transformPoints "scale=(1e-2 1e-2 1e-2)"`, riportando di fatto tutte le unità di misura delle dimensioni fornite ai centimetri e arrivando in questo modo ad una descrizione del tessuto del tutto realistica.

La mesh creata a questo punto deve essere controllata e validata per valutarne la qualità: questo avviene tramite l'utility `checkMesh`, che anche in questo caso legge un dizionario chiamato `checkMeshDict` contenente le condizioni che la mesh deve rispettare, ad esempio la minima e massima area delle facce, l'aspect ratio e la non ortogonalità.

Durante l'esecuzione di `snappyHexMesh` può capitare che l'utility crei alcune celle isolate: la risoluzione delle equazioni differenziali con le modalità descritte nel capitolo 2.3.1 rende indispensabile che le celle siano connesse a quelle adiacenti. Il comando `subsetMesh region0 -overwrite` permette di eliminare queste celle che impedirebbero il corretto lancio della simulazione. Un secondo utilizzo di `checkMesh` permette di valutare la qualità della mesh finale utilizzata durante la simulazione.

A questo punto è possibile lanciare la simulazione. Il solver utilizzato per risolvere problemi allo steady state in cui si vuole risolvere il campo di moto di moto e di pressione, come in questo caso, è simpleFoam. L'obiettivo è risolvere tramite il metodo dei volumi finiti l'equazione di continuità e quella di Navier-Stokes una volta che quest'ultime sono state trasformate in equazioni algebriche in seguito all'applicazione di una discretizzazione spaziale.

3.1.2 Grid Independence

Nelle simulazioni CFD è fondamentale assicurarsi che i risultati ottenuti non dipendano dalla definizione della griglia con la quale viene suddiviso il dominio computazionale, e cioè che i risultati non dipendano dal numero di celle impostato nelle varie dimensioni.

Una mesh più fitta è in generale sinonimo di risultati più accurati, ma aumentare troppo il numero di celle potrebbe portare ad un aumento esorbitante dei costi computazionali, poiché in ogni cella dovranno essere risolte le equazioni differenziali che descrivono i bilanci, come quelli di conservazione della massa, della quantità di

moto e dell'energia, trasformandole in un sistema di equazioni algebriche secondo il metodo dei volumi finiti descritto nel capitolo 2.3.1.

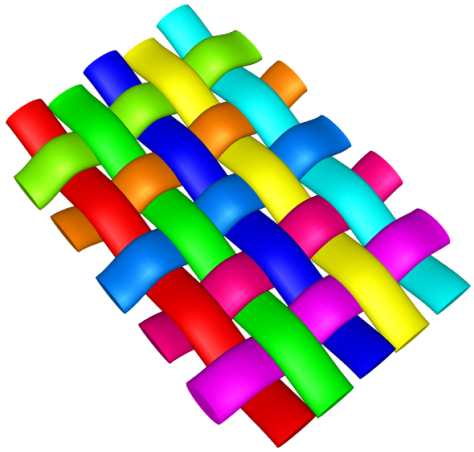


Figura 3.4: Tessuto PW con fabric count pari a 29.3-20 e reduction factor pari a 0.5 usato per lo studio di grid independence

Si è quindi deciso di analizzare un tessuto con armatura Plain Weave e densità di fili in trama e ordito pari a 29.3-20 mostrato in figura 3.4: si è scelto infatti l'armatura che, pur essendo nella pratica la più semplice da realizzare tra tutti i tessuti intrecciati, risulta quella in cui i fili si interlacciano maggiormente e di utilizzare la combinazione di fabric count più elevata in modo da avere la geometria più densa tra quelle studiate e che perciò risulta la più difficile da riprodurre durante il processo di meshing. Durante lo studio di grid independence il reduction factor è stato considerato pari a 0.5: il valore del reduction factor sarà successivamente utilizzato come parametro di fitting, come già accennato, per validare i risultati delle simulazioni con quelli presenti in letteratura, ma in prima analisi si è scelto questo valore medio per descrivere la tendenza dei fili allo stiramento, cioè del loro schiacciarsi durante il processo di interlacciamento per formare un tessuto.

Dal momento che ogni combinazione di armatura e di fabric count individua un dominio computazionale differente il parametro scelto per individuare in modo univoco il livello di raffinamento della mesh è stato il numero di celle per diametro reale del filo di ordito, cioè il diametro che tiene conto del reduction factor, chiamato w e rappresentato in figura 2.3.

Durante le simulazioni uno script Python individua a partire dal file .stl della geometria in esame i diversi parametri necessari a definire il dominio computazionale, scrivendoli in `caseSetup`. In `caseSetup` viene anche definito il numero di celle per diametro, utilizzato per lo studio di grid independence.

Con i dati che legge da `caseSetup`, il dizionario `blockMeshDict` è in grado di generare il dominio computazionale e dividerlo in un determinato numero di celle

in ogni direzione, secondo queste equazioni:

$$n_x = \frac{CpD}{w_1} \cdot l_x \tag{3.1}$$

Dove n_x è il numero di celle lungo la direzione x, l_x è la lunghezza del lato della mesh lungo la direzione x e w_1 è il diametro del filo di ordito che tiene conto della deformazione del filo ed è quindi funzione del Poisson ratio e del reduction factor secondo le equazioni:

$$w_1 = d_1 + v \cdot (d_1 - t_1) \tag{3.2}$$

$$t_1 = r_f \cdot d_1 \tag{3.3}$$

Per ottenere una griglia uniforme, a partire da n_x si definiscono i numeri di celle nelle altre due direzioni secondo due semplici proporzioni

$$n_x : l_x = n_y : l_y \tag{3.4}$$

$$n_x : l_x = n_z : l_z \tag{3.5}$$

Dove l_y ed l_z sono rispettivamente le lunghezze del dominio computazionale lungo y e z (Queste sono tutte informazioni salvate da bounding.py in `caseSetup` e lette da `blockMeshDict`).

Una volta individuato il valore di CpD che rende i risultati indipendenti dalla definizione della griglia per il PW-29.3-20 allora questo determinerà le celle necessarie in ogni direzione per ogni armatura e ogni combinazione di densità di trama e ordito.

Il secondo passo nella generazione della mesh è l'utility `snappyHexMesh`, che a partire dalla mesh a blocchi definita da `blockMesh` elimina le celle in cui è presente il solido. Nello studio di grid independence si è scelto di variare il parametro `level` nel sub-dizionario `refinementSurfaces` di `snappyHexMeshDict`.

`refinementSurfaces` imposta il raffinamento della mesh attorno alle superfici solide definite dall'.stl. Il parametro `min` all'interno di `level (min max)` imposta il livello minimo di suddivisione delle celle, mentre max imposta il livello massimo. Sono state testate tre diverse configurazioni: `level (0 0)`, `level (1 1)` e `level (2 2)`.

In definitiva nello studio di grid independence si è indagato il ruolo di due parametri: CpD e `RefLe` (da inserire in `level (RefLe RefLe)`) sul valore della velocità lungo z. Questa velocità, definita velocità Darcyana, è una velocità media calcolata sulla faccia di inlet, ed è un valore direttamente fornito da OpenFOAM al termine della simulazione.

refinementSurfaces level (0 0)

CpD	numero di celle totali	celle totali dopo snappyHexMesh	aumento di celle	aumento %	v [m/s]	variazione v %
14	139 568	98 288			0.039 509 8	
16	205 030	144 226	45 938	32	0.293 243	86.53
18	297 198	209 172	64 946	31	0.456 620	35.78
20	413 512	291 294	82 122	28	0.475 372	3.94
22	533 760	375 986	84 692	23	0.466 812	1.83
24	702 240	494 445	118 459	24	0.471 299	0.95
26	902 880	636 484	142 039	22	0.457 758	2.96
28	1 138 488	802 368	165 884	21	0.452 358	1.19
30	1 361 745	960 384	158 016	16	0.447 248	1.14
32	1 668 520	1 176 576	216 192	18	0.441 900	1.21

Tabella 3.1: Confronto celle e velocità al variare di *CpD* per `refinementSurfaces level (0 0)`.

refinementSurfaces level (1 1)

CpD	numero di celle totali	celle totali dopo snappyHexMesh	aumento di celle	aumento %	v [m/s]	variazione v %
14	139 568	209 997			0.458 720	
16	205 030	293 477	83 480	28	0.445 534	2.96
18	297 198	400 058	106 581	27	0.435 949	2.20
20	413 512	529 642	129 584	24	0.426 892	2.12
22	533 760	661 481	131 839	20	0.421 906	1.18
24	702 240	838 898	177 417	21	0.417 593	1.03
26	902 880	1 042 985	204 087	20	0.411 339	1.52
28	1 138 488	1 279 070	236 085	18	0.406 670	1.15

Tabella 3.2: Confronto celle e velocità al variare di *CpD* per `refinementSurfaces level (1 1)`.

refinementSurfaces level (2 2)

CpD	numero di celle totali	celle totali dopo snappyHexMesh	aumento di celle	aumento %	v [m/s]	variazione v %
14	139 568	733 626			0.408 126	
16	205 030	976 053	242 427	25	0.403 317	1.19
18	297 198	1 277 045	300 992	24	0.398 612	1.18
20	413 512	1 623 539	346 494	21	0.395 841	0.70
22	533 760	1 966 410	342 871	17	0.394 171	0.42
24	702 240	2 409 608	443 198	18	0.392 544	0.41
26	902 880	2 896 837	487 229	17	0.391 851	0.18
28	1 138 488	3 435 440	538 603	16	0.391 026	0.21

Tabella 3.3: Confronto celle e velocità al variare di *CpD* per `refinementSurfaces level (2 2)`.

Le simulazioni hanno restituito diversi valori di velocità, riportati in figura 3.5. Per `level (0 0)`, soprattutto per bassi valori di CpD, i risultati risultano di bassa qualità oscillando molto e stabilizzandosi molto lentamente: questo perché la combinazione di bassi CpD e l'assenza del raffinamento vicino alle superfici descrive

molto grossolanamente la superficie del tessuto.

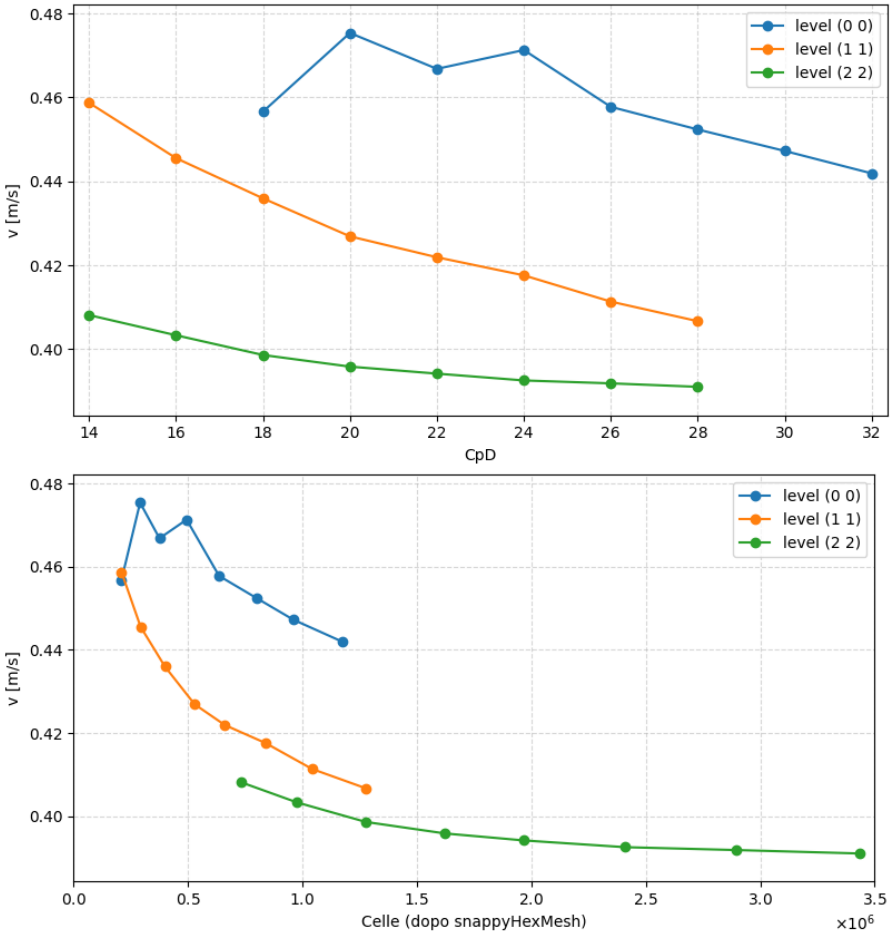


Figura 3.5: Andamento della velocità v in funzione di CpD e del numero totale di celle dopo `snappyHexMesh`

Gli andamenti di `level (1 1)` e di `level (2 2)` invece risultano migliori e l'aumentare di CpD determina valori di velocità sempre più bassi, che sembrano raggiungere l'asintoto soprattutto nel caso di `level (2 2)`. Il grafico di `level (2 2)` mostra una convergenza migliore verso un valore stabile, il che ha portato alla scelta di questi parametri per quanto riguarda il livello minimo e massimo di raffinamento. Per quanto riguarda la scelta del valore di CpD minimo per descrivere accuratamente il problema e cioè per il quale l'aumento ulteriore del numero di celle non determina una variazione marcata dei risultati, si è osservato che per CpD pari a 20 la velocità diminuiva dello 0.70% rispetto al caso precedente (CpD pari a 18) con un aumento di celle del 21%. Un ulteriore raffinamento (CpD pari a 22) avrebbe comportato una diminuzione della velocità dello 0.42% a fronte di un ulteriore aumento del 17% per quanto riguarda il numero di celle, passando da

1623539 a quasi due milioni di celle. Quest'ultimo aumento di celle risulta perciò ingiustificato in relazione all'esigua diminuzione del valore della velocità. La scelta dei due parametri che determinano la definizione della griglia di calcolo è quindi stata di 20 celle per diametro e di `level (2 2)` per quanto riguarda la raffinatezza vicino alle superfici solide.

Va però specificato che, dal momento che il dominio computazionale cambia in funzione della geometria del tessuto trattato, il numero di celle varia continuamente al variare dei parametri l_x, l_y e soprattutto di l_z, che essendo funzione dello spessore del tessuto dipende fortemente dal reduction factor, come mostrato in figura 3.6.

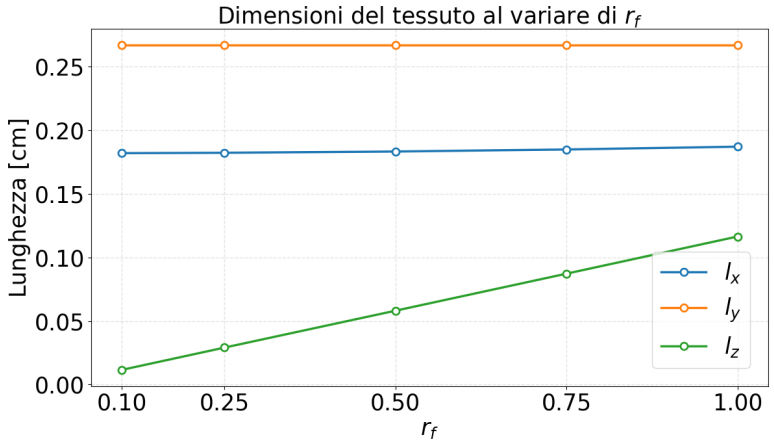


Figura 3.6: Dimensioni nelle tre direzioni al variare del reduction factor

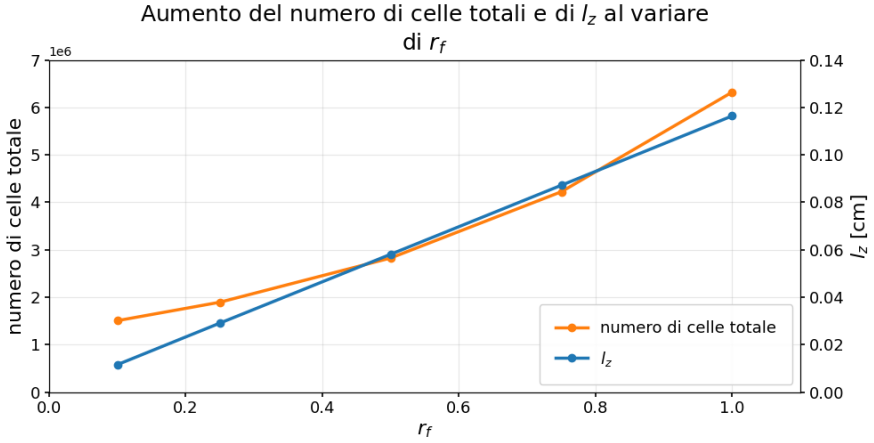


Figura 3.7: Aumento di celle e di l_z al variare del reduction factor

In generale, per la stessa armatura e combinazione di densità di fili di trama e ordito il numero totale di celle aumenta all'aumentare del reduction factor in

quanto aumenta, soprattutto, la dimensione lungo z del dominio computazionale. Simulazioni che hanno reduction factor maggiori avranno perciò, a parità di CpD e di livelli di raffinamento, un numero di celle maggiore, come mostrato in figura 3.7.

Dal momento che il valore del parametro CpD individua il numero di celle con cui `blockMesh` suddivide il dominio e che `snappyHexMesh` elimina le celle dove si trova la superficie solida, il profilo di velocità verrà risolto effettivamente solo tra gli interstizi creati dai fili. Il reduction factor, che verrà variato in quanto verrà utilizzato come parametro di fitting, incide molto sul valore di w_1 e perciò su quelli di n_x, n_y ed n_z: per alcuni valori di reduction factor il profilo di velocità potrebbe non venire descritto in modo soddisfacente, in quanto gli interstizi potrebbero essere descritti da troppe poche celle. Si è perciò scelto di mantenere il valore di w_1 pari a quello ottenuto nello studio di grid independence, cioè quello ottenuto con un valore di reduction factor pari a 0.5, come valore da inserire nell'equazione 3.1 per ogni geometria, anche al variare del reduction factor.

3.1.3 Condizioni iniziali e condizioni al contorno

All'interno del `blockMeshDict`, subito dopo la definizione del numero di celle nelle tre dimensioni, vi è la lista `boundary` [29]. Questa sezione definisce sei patch del blocco iniziale, i cui vertici sono definiti da uno script Python come descritto nel capitolo 3.1.1, ai quali viene assegnato un nome.

Le informazioni per ciascuna patch sono poi contenute nel rispettivo subdizionario, che è composto da due entrate:

1. `type` definisce il tipo di patch, che può essere una semplice `patch` alla quale si applicano le condizioni al contorno oppure una condizione geometrica. [31]
2. `faces` definisce la faccia del blocco attribuita a quella patch utilizzando quattro vertici alla volta tra quelli definiti all'inizio di `blockMesh`.

Nelle simulazioni effettuate il blocco principale esaedrico è stato diviso in sei patch chiamate `inlet`, `outlet`, `top`, `bottom`, `left` e `right`. Le prime due costituiscono le facce in cui entra ed esce il flusso di aria e sono definite come `type patch` in quanto a queste verranno applicate le condizioni al contorno. Le restanti patch sono invece tutte del tipo `type simmetry`.

L'utilizzo delle patch `type simmetry` è decisamente utile poiché permette di risolvere la simulazione solo per una porzione del dominio, in questo caso una porzione di tessuto definita da dieci fili in totale, tenendo in considerazione che la stessa soluzione sarà ripetuta, per simmetria appunto, nelle direzioni delle tre dimensioni.

Durante l'esecuzione di `snappyHexMesh` la mesh viene adattata attorno al tessuto e questo determina la creazione di una nuova patch che prende il nome del tessuto in analisi.

Queste sette patch vengono utilizzate per definire le condizioni al contorno all'interno dei file `p` e `U` che descrivono rispettivamente le condizioni per la pressione e per la velocità e si trovano all'interno della directory `0`, cioè quella relativa al tempo iniziale.

Per quanto riguarda la pressione, questa è impostata in modo da definire una differenza di pressione di 163.27 m^2/s^2 equivalente a 200 Pa tra la faccia di inlet e quella di outlet, così da replicare le stesse condizioni utilizzate da Zupin et al. per ricavare i risultati sperimentali [10]. OpenFOAM utilizza i m^2/s^2 come unità di misura per la pressione, definita come pressione cinetica o pressione divisa per densità:

$$P[m^2/s^2] = \frac{P[\text{Pa}]}{\rho} \tag{3.6}$$

Dal momento che il fluido in considerazione è l'aria, descritta come un fluido incomprimibile newtoniano con densità pari a $\rho = 1.225\ kg/m^3$ e viscosità cinematica pari a $\nu = 1.46 \cdot 10^{-5}\ m^2/s$ allora i valori di pressione sono stati impostati pari a 163.27 m^2/s^2 nella faccia di inlet, mentre è stato impostato come nullo il valore nella faccia di outlet. I valori appena calcolati sono considerati uniformi e costanti sulle facce, infatti sono stati utilizzati i comandi `type fixedValue` e `value uniform`.

Per le patch `simmetry` è stato specificato il tipo `type simmetry` in virtù delle simmetrie geometriche che ogni tessuto possiede, mentre per il tessuto è stato specificato il tipo `type zeroGradient`, impostando cioè il gradiente normale della pressione nullo in prossimità del tessuto.

La velocità è invece definita nel rispettivo file `U`, in cui le facce di inlet ed outlet sono descritte da un `type zeroGradient`, mentre le altre patch definite in `blockMesh` sono, anche in questo caso, impostate con `type simmetry`. Per quanto riguarda il tessuto è stato definito il `noSlip` che impone che il profilo di velocità sia nullo a contatto con la parete solida.

Per quanto riguarda le condizioni iniziali, queste sono definite all'inizio dei file `p` e `U` attraverso l'entrata `internalField`. Questa stringa imposta per la pressione il valore `uniform 80` posizionandosi circa a metà dei valori impostati nelle patch inlet ed outlet, mentre imposta come nulla la velocità, che essendo una grandezza vettoriale sarà descritta come `uniform (0 0 0)`.

Queste due condizioni rappresentano l'ipotesi iniziale per i due campi utilizzati dal solver stazionario `simpleFoam`, che utilizza l'algoritmo $SIMPLE$ descritto nel capitolo 2.3.3.

3.1.4 Estrapolazione dei risultati

L'obiettivo delle simulazioni è di indagare tramite CFD la permeabilità dei tessuti all'aria. Trattando i tessuti come mezzi porosi si può pensare di utilizzare la legge di Darcy, che mette in relazione lineare la velocità del fluido al gradiente di pressione, per arrivare a questo dato. [9].

La permeabilità K_f è data, secondo questa legge, da:

$$K_f = \mu \frac{Lv}{\Delta P} \tag{3.7}$$

Dove μ è la viscosità dinamica dell'aria, L è lo spessore del tessuto, v è la velocità del fluido e ΔP è la differenza di pressione imposta ai capi del tessuto.

Questa legge è però valida esclusivamente in regime laminare. La natura del moto di un fluido è investigata attraverso il numero di Reynolds, un numero adimensionale definito come:

$$Re = \frac{\rho v L}{\mu} \tag{3.8}$$

Dove ρ è la densità del fluido, v è la sua velocità, μ la sua viscosità dinamica e L è una lunghezza caratteristica.

A seconda della geometria in esame vengono definiti dei valori di Re che identificano i diversi regimi fluidodinamici: laminare, di transizione e turbolento.

Per i tessuti una valida definizione di lunghezza caratteristica L può essere quella di raggio idraulico dei pori. Il valore del raggio idraulico R_h è direttamente collegato a quello di diametro idraulico D_h già descritto nel capitolo 2.4.2, più precisamente vale:

$$R_h = \frac{D_h}{4} \tag{3.9}$$

La legge di Darcy perde di validità per valori di Re elevati. Dal momento che i risultati delle simulazioni mostrano un regime di flusso non-laminare, si è reso necessario un approccio alternativo.

In un approccio molto più simile alla misurazione pratica della permeabilità descritta nel capitolo 2.4.1 si è quindi deciso di valutare la velocità del flusso di aria attraverso il tessuto.

Questo valore di velocità, chiamata velocità Darcyana e definita come il rapporto tra la portata volumica del fluido e l'area del tessuto, è un dato che è possibile ottenere direttamente dal software OpenFOAM al termine delle simulazioni, come InletAvarageSpeed. Questa corrisponde ad una media pesata sull'inlet della velocità ed è una Function Object definita nel `controlDict`.

Capitolo 4

Risultati

4.1 Identificazione del parametro di fitting r_f

Le geometrie descritte nel capitolo 2.1.6 sono quindi state analizzate con il software OpenFOAM impostando ogni simulazione secondo quanto descritto nel capitolo 3.1.

Ogni geometria è descritta dai parametri geometrici definiti nel capitolo 2.1.6. Tra questi il parametro r_f, cioè il reduction factor, è stato scelto come parametro di fitting. Il reduction factor è stato infatti definito come indicatore del comportamento dei diversi tessuti al flusso di aria generato dalla differenza di pressione imposta, che risulta difficile da prevedere a priori, a differenza degli altri parametri di design, noti da Zupin et al. [10].

Il reduction factor è definito secondo l'equazione 2.5, e può perciò assumere valori compresi tra 0 e 1. Per mantenere un significato fisico ed evitare tessuti senza spessore r_f è stato quindi fatto variare tra 0.1 e 1 con passo 0.01 per ogni geometria analizzata. Per ognuna delle 16 geometrie (quattro combinazioni di densità di fili di trama e di ordito per ciascuna delle quattro armature) sono state perciò condotte 91 simulazioni volte ad individuare il valore di r_f che restituisse il valore di velocità Darcyana più vicino a quello sperimentale, fornito, come per i parametri geometrici del filo, da Zupin et al [10]. Il valore fornito da OpenFOAM è espresso in m/s, mentre l'articolo di riferimento fornisce i risultati in termini di $L/(s \cdot m^2)$, che esprime comunque una velocità in unità di misura del sistema internazionale, a meno di un fattore di conversione di 10^{-3}. Per individuare il valore ottimale del reduction factor per ogni densità di ogni armatura è stato calcolo l'errore quadratico medio (mean squared error) tra la velocità sperimentale v_{exp} e la velocità ottenuta dalle simulazioni v_{cfd}.

$$MSE = (v_{cfd} - v_{exp})^2 \tag{4.1}$$

4.1.1 Plain Weave

Nella figura 4.1 sono riportati i valori della velocità ottenuta dalle simulazioni al variare di r_f insieme al valore sperimentale della velocità. Quando la velocità restituita da OpenFOAM interseca la velocità sperimentale, allora si avrà un minimo nella curva del MSE, come riportato nei grafici, che indicherà il valore ottimale di reduction factor per quella specifica geometria.

Osservando i risultati è possibile notare come per valori di r_f prossimi agli estremi le simulazioni possono restituire valori di velocità poco realistici o molto fluttuanti. Addirittura nel caso del PW-29.3-20 le simulazioni con reduction factor da 0.1 a 0.18 non arrivavano a convergenza. Si nota però come per valori centrali di reduction factor, che corrispondono effettivamente a valori con maggiore significato fisico, le velocità assumano una tendenza monotona crescente.

I risultati per i valori ottimali di r_f sono riassunti nella tabella 4.1.

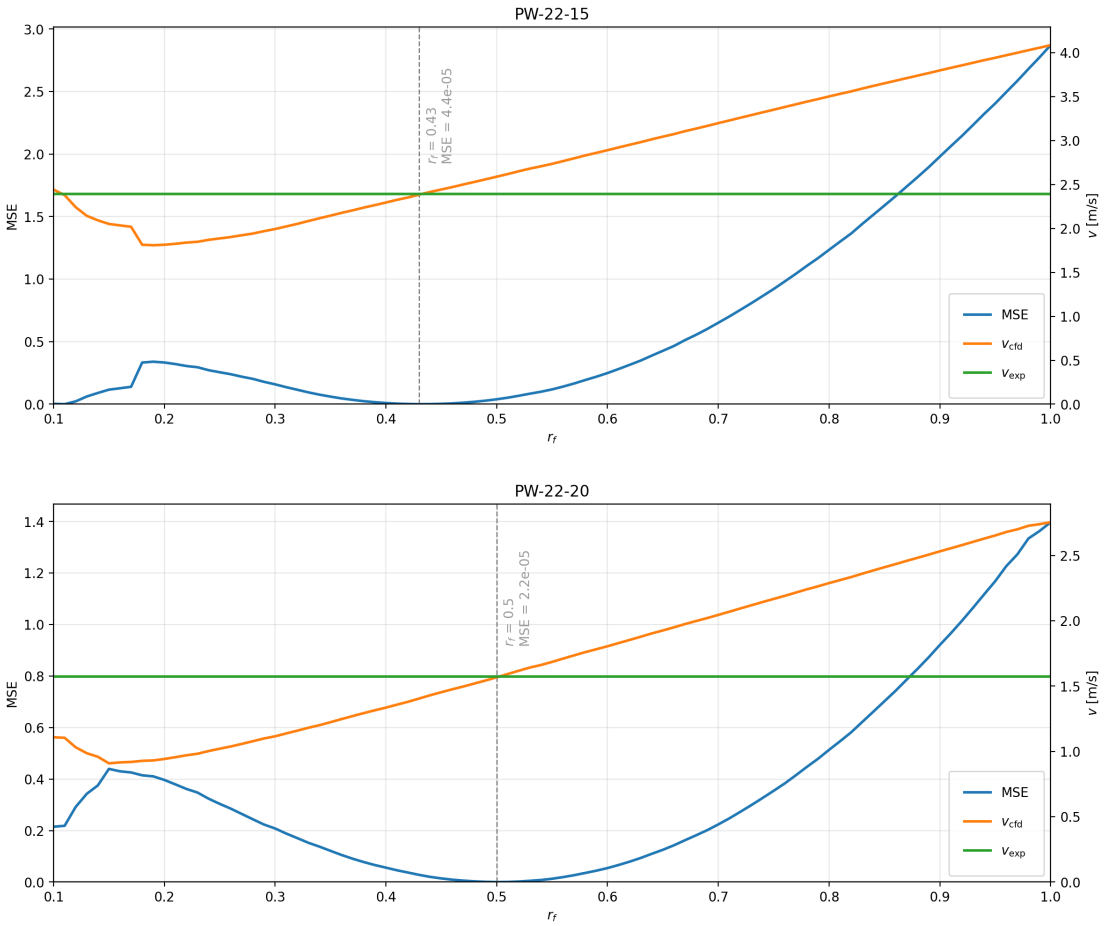


Figura 4.1: Confronto dei grafici PW con r_f ottimale evidenziato.

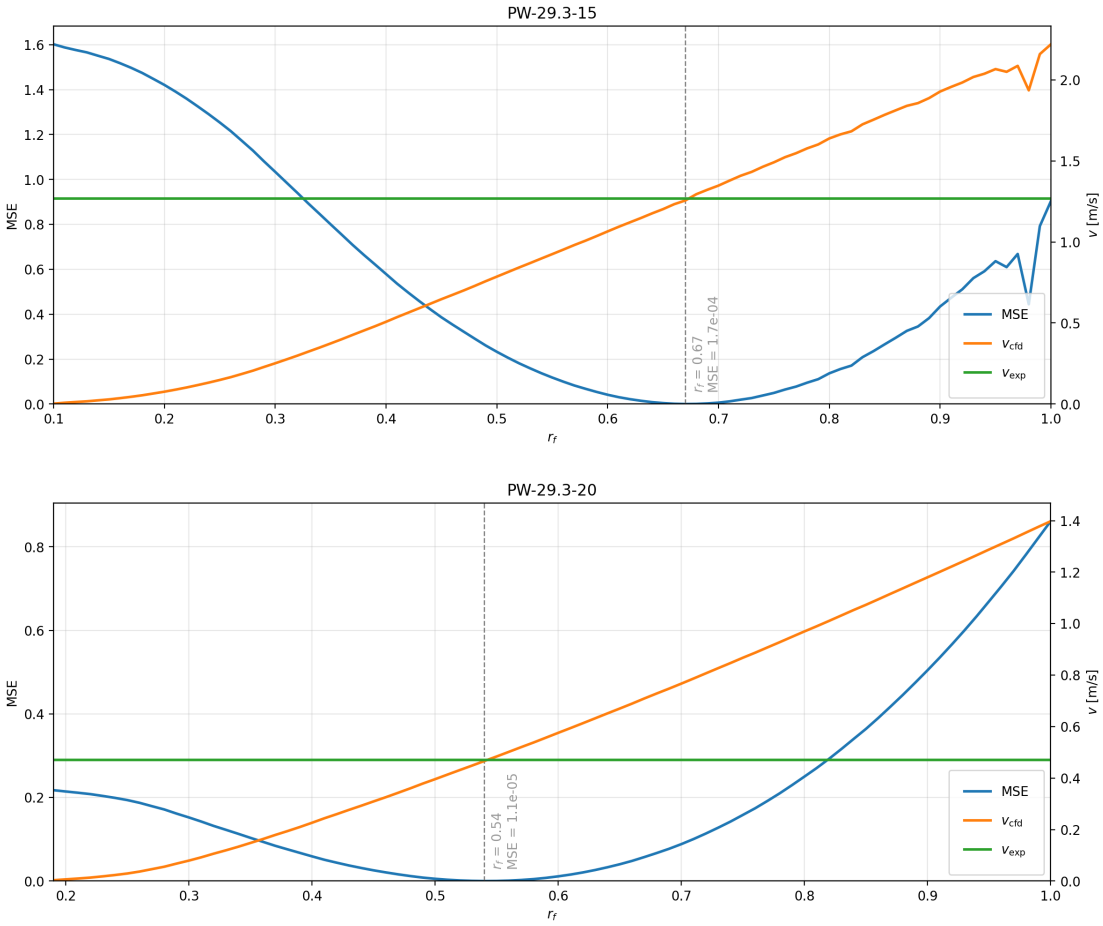


Figura 4.1: Confronto dei grafici PW con r_f ottimale evidenziato.

Simulazioni ottimali

PW	v_{exp} [m/s]	v_{CFD} [m/s]	MSE	r_f ottimale
PW-22-15	2.392	2.385	4.4×10^{-5}	0.43
PW-22-20	1.572	1.567	2.2×10^{-5}	0.50
PW-29.3-15	1.268	1.255	1.7×10^{-4}	0.67
PW-29.3-20	0.470	0.466	1.1×10^{-5}	0.54

Tabella 4.1: Valori di velocità sperimentale e CFD, MSE e r_f ottimale per i tessuti PW.

Individuare un valore ottimale di reduction factor per ogni geometria potrebbe causare overfitting durante il processo di allenamento della rete neurale. L'obiettivo risulta perciò quello di ottenere un parametro descrittivo dell'intera armatura, che possa descrivere il comportamento di tutte le geometrie al suo interno e che permetta quindi di simulare con soddisfacente accuratezza anche geometrie con densità diverse da quelle sperimentali.

Questo parametro, chiamato reduction factor globale, è individuato all'interno dell'intervallo definito dai r_f ottimali ottenuti per le geometrie sperimentali. Per ogni valore di r_f compreso tra il valore minimo e massimo di r_f ottimale individuati in precedenza e indicati in tabella 4.1 si valuta la radice dell'errore quadratico medio ($RMSE$: rooted mean square error) per le quattro combinazioni di densità, come nell'equazione 4.2.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{4} \left(v_{cfd_i} - v_{exp_i}\right)^2}{4}} = \sqrt{\frac{\sum_{i=1}^{4} \left(MSE_i\right)}{4}} \tag{4.2}$$

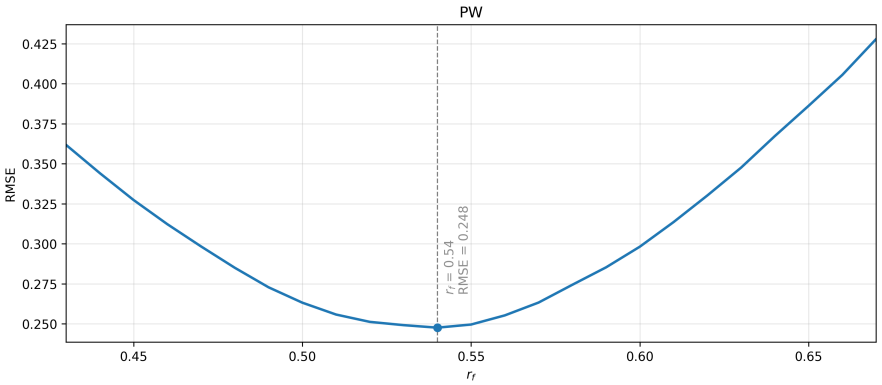


Figura 4.2: Radice dell'errore quadratico medio al variare di r_f.

La figura 4.2 riporta il grafico del $RMSE$ al variare di r_f e individua, in corrispondenza del minimo della curva, il valore ottimale del reduction factor per l'armatura Plain Weave, definito come r_f globale.

4.1.2 Basket Weave

Procedendo con l'armatura Basket Weave, sono state effettuate le stesse considerazioni del caso precedente. Rispetto al PW, questa armatura presenta valori di velocità molto più oscillanti per valori di r_f crescenti, come è possibile osservare dalla figura 4.3. Questo comportamento risulta in linea con le considerazioni del capitolo 2.1.6, infatti nella figura 2.5 si nota come l'armatura BW sia la più voluminosa delle quattro, soprattutto per quanto riguarda la combinazione di densità di fili di trama e di ordito 22-15.

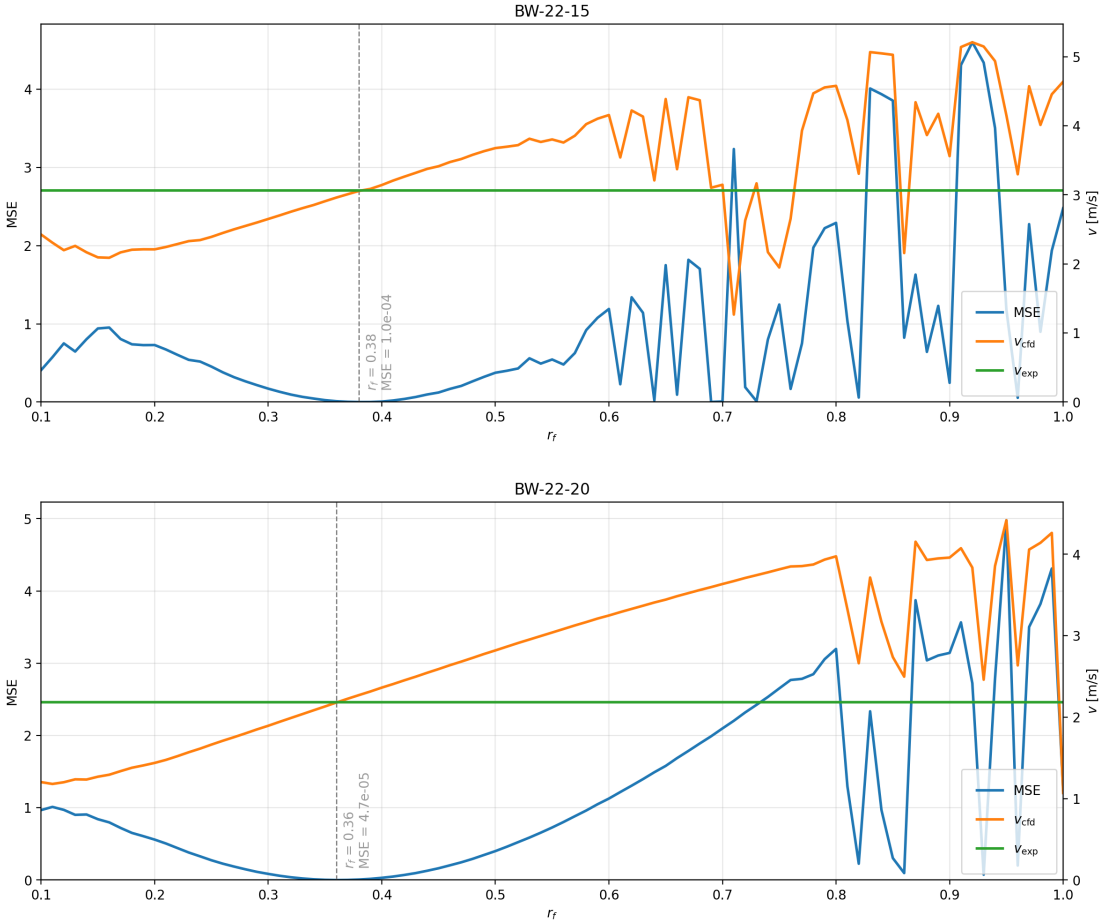


Figura 4.3: Confronto dei grafici BW con r_f ottimale evidenziato.

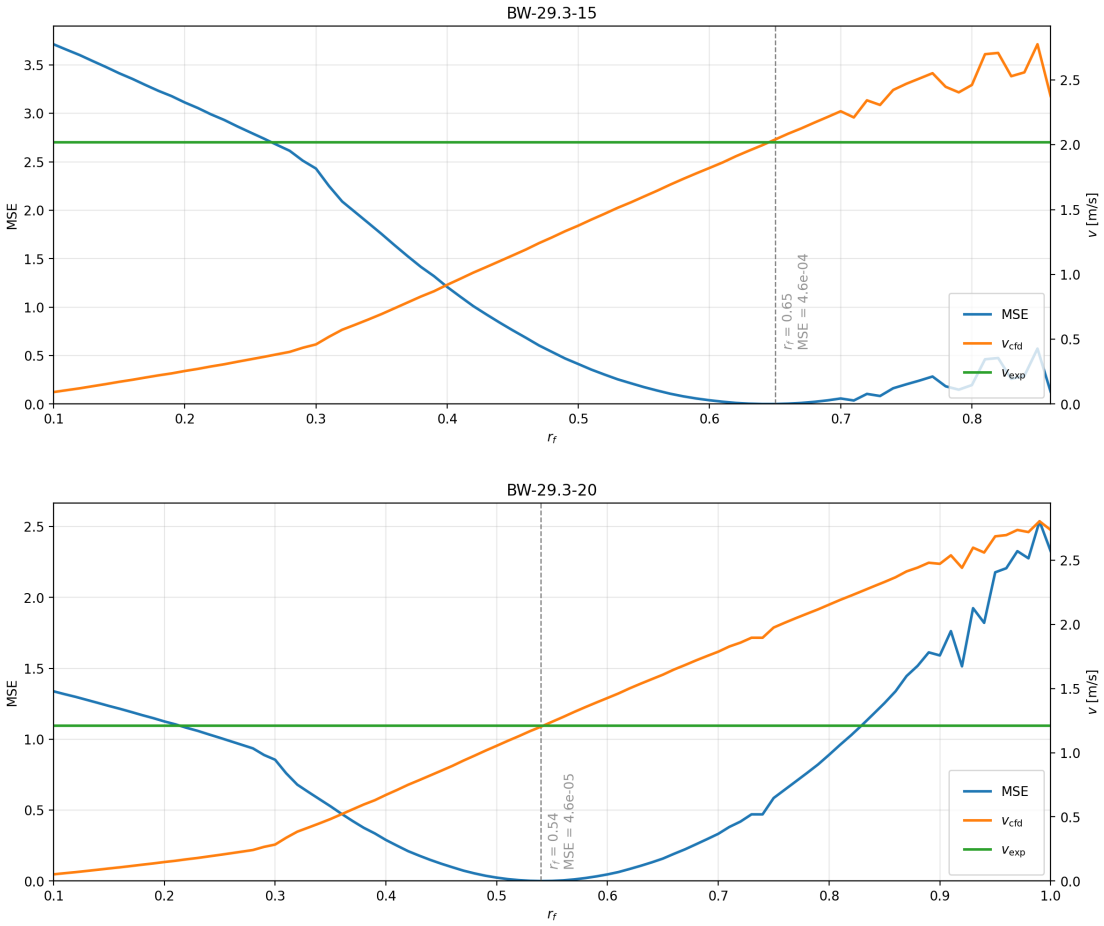


Figura 4.3: Confronto dei grafici BW con r_f ottimale evidenziato.

Il valore di reduction factor per il quale la v_{cfd} interseca la v_{exp} minimizzando il MSE è individuato, per ogni geometria, in un intorno di valori per i quali non si notano oscillazioni della velocità estratta dalle simulazioni.

Procedendo all'individuazione del valore globale di r_f per il Basket Weave però mette di fronte a risultati poco affidabili, come è possibile notare in figura 4.4, dove sono presenti più minimi relativi della curva dell'$RMSE$.

Il motivo di questo comportamento anomalo risiede nella rumorosità dei risultati della geometria con fabric count pari a 22-15, che presenta valori di velocità oscillanti e non più monotoni a partire da un valore di r_f ottimale pari a 0.54. Sì e costretti ad includere i risultati da questa geometria in quanto, come riassunto in tabella 4.2, l'intervallo in cui si ricerca il valore globale di r_f è delimitato da valore minimo di 0.36 al valore massimo di 0.65, comprendendo quindi i risultati oscillanti di BW-22-15 per valori di r_f superiori a 0.54.

Simulazioni ottimali

BW	v_{exp} [m/s]	v_{CFD} [m/s]	MSE	r_f ottimale
BW-22-15	3.065	3.057	1.00×10^{-4}	0.38
BW-22-20	2.187	2.180	4.75×10^{-5}	0.36
BW-29.3-15	2.019	2.041	4.63×10^{-4}	0.65
BW-29.3-20	1.209	1.202	4.58×10^{-5}	0.54

Tabella 4.2: Valori di velocità sperimentale e CFD, MSE e r_f ottimale per i tessuti BW.

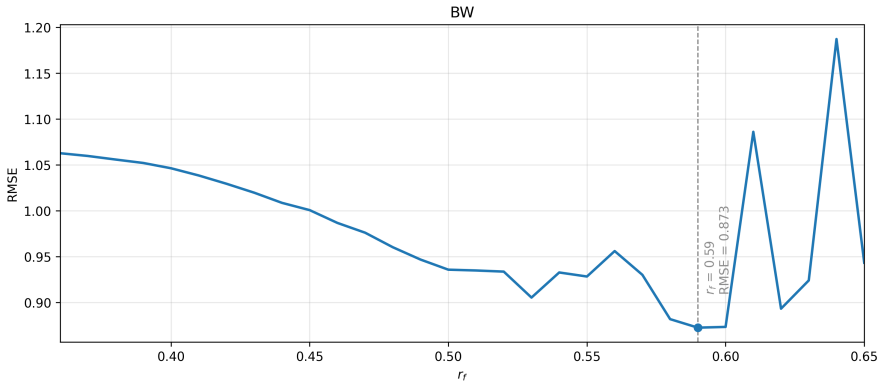


Figura 4.4: Radice dell'errore quadratico medio al variare di r_f.

Per evitare di scegliere un valore di r_f globale che fosse poco rappresentativo per via di alcuni risultati poco affidabili provenienti da una sola geometria si è quindi deciso di aumentare il numero di celle nelle tre dimensioni definito da `blockMesh`

di un fattore pari a 1.6 per le simulazioni con reduction factor tra 0.54 e 0.65 della geometria BW-22-15. Lo stesso dominio computazionale viene ora descritto da un numero di celle pari a circa quattro volte quello iniziale, che porta perciò a nuovi risultati per quanto riguarda la geometria BW-22-15.

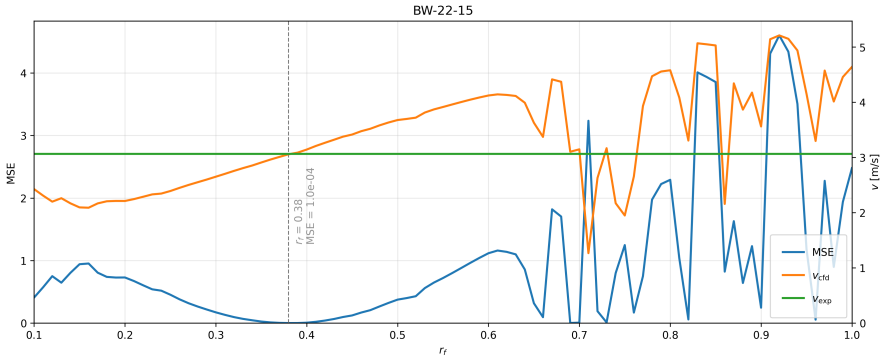


Figura 4.5: Risultati per BW-22-15 dopo l'aumento del numero di celle.

Come ci si poteva aspettare, osservando la figura 4.5 il valore di reduction factor ottimale per questa armatura non varia rispetto ai primi risultati, che già risultavano accurati.

L'andamento della velocità, che si mantiene monotono crescente per valori di r_f più alti dopo l'aumento delle celle determina però una nuova curva dell'$RMSE$ che quindi individua un nuovo valore, più affidabile, dell'r_f globale per il Baset Weave.

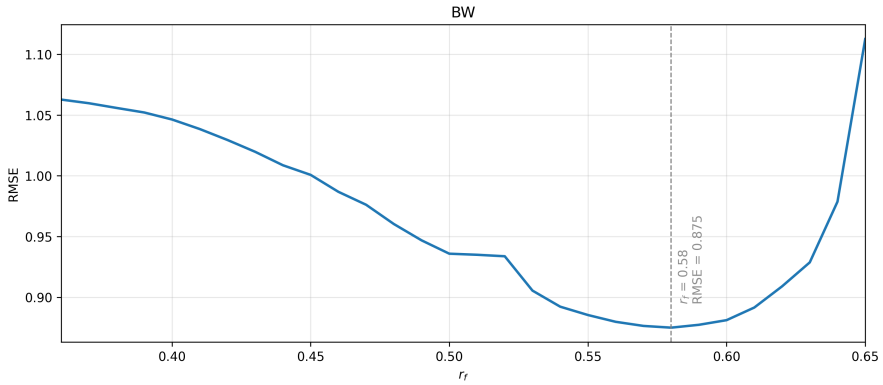


Figura 4.6: Radice dell'errore quadratico medio al variare di r_f dopo l'aumento del numero di celle.

Confrontando le figure 4.4 e 4.6 si nota come il valore dell'r_f globale passi da 0.59 a 0.58. L'aumento di celle per le simulazioni che presentavano oscillazioni della velocità non determina una variazione marcata del reduction factor globale per il BW, ma permettono di individuarlo nel minimo assoluto della curva dell'$RMSE$, invece che in un minimo relativo come in figura 4.4.

4.1.3 Filling Rib

L'armatura FR, pur essendo più voluminosa del PW, è comunque sempre meno estesa rispetto al BW, come osservabile in figura 2.5. Per le geometrie di questa armatura infatti non è stato necessario aumentare le celle come per il Basket Weave e i risultati, seppure con qualche oscillazione per bassi valori di r_f (FR-22-15) o per alti valori di r_f (FR-29.3-15) individuano sempre il valore di r_f ottimale quando la curva della v_{exp} è intersecata dalla curva della v_{CFD} nel suo tratto monotono crescente, indice di simulazioni affidabili.

Per il FR-29.3-20 non si osservano oscillazioni né per valori bassi, né per valori alti di reduction factor.

Nella tabella 4.3 sono riassunti i risultati per le simulazioni ottimali, che individuano il range di valori in cui verrà cercato il r_f globale per l'armatura FR.

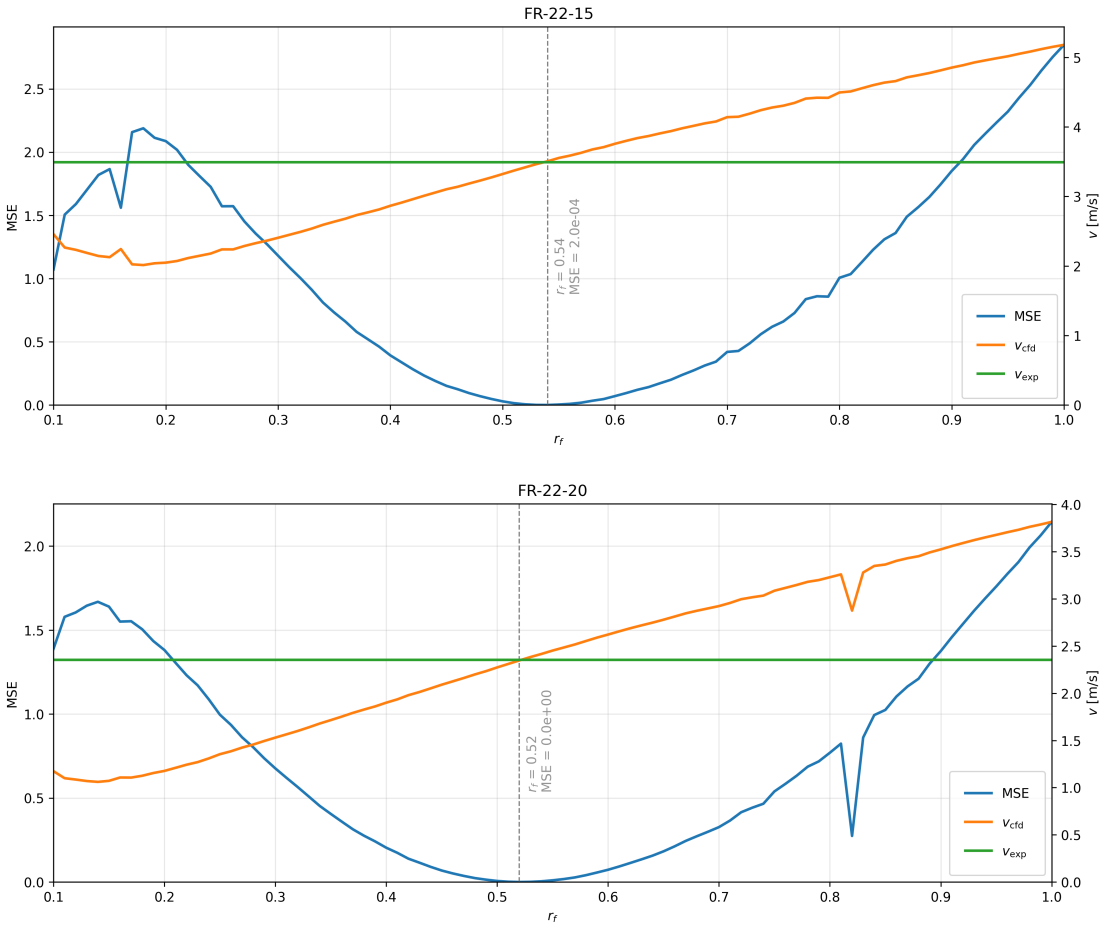


Figura 4.7: Confronto dei grafici FR con r_f ottimale evidenziato.

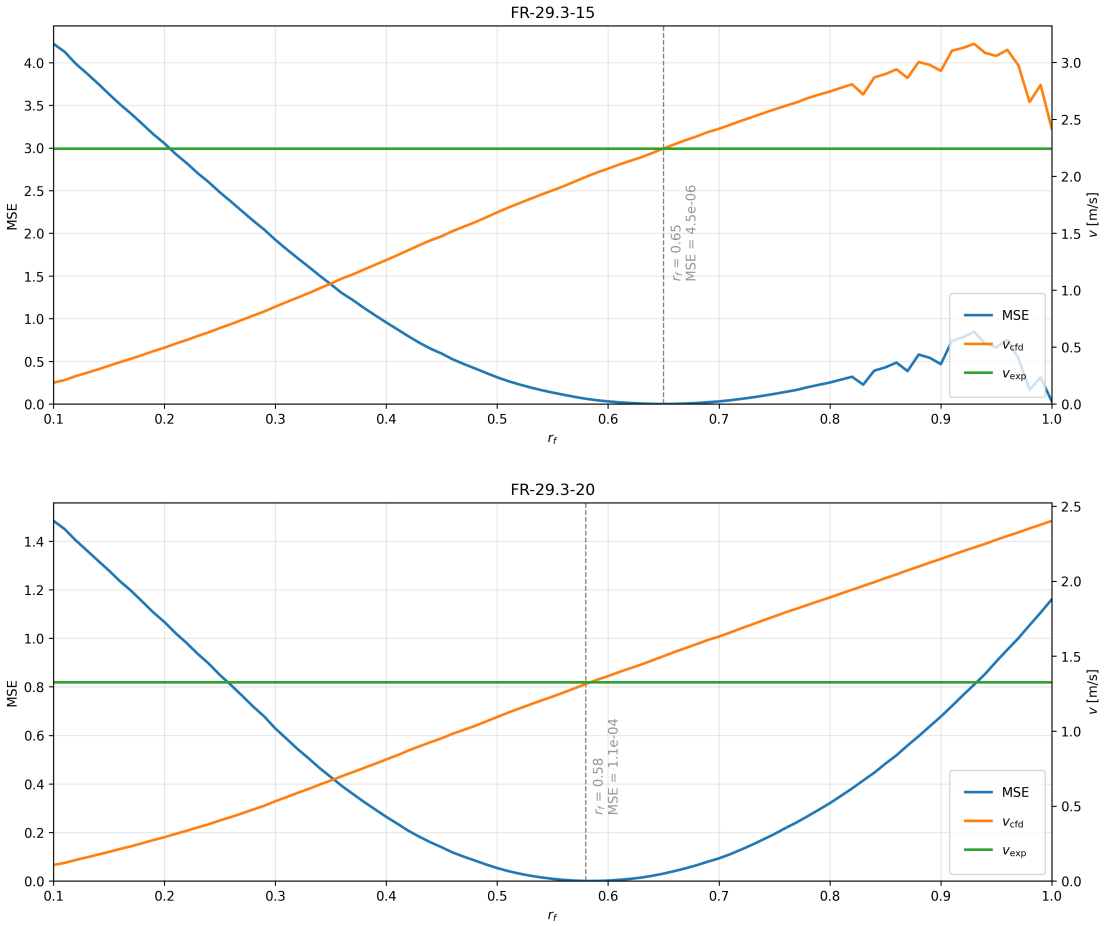


Figura 4.7: Confronto dei grafici FR con r_f ottimale evidenziato.

Simulazioni ottimali

FR	v_{exp} [m/s]	v_{CFD} [m/s]	MSE	r_f ottimale
FR-22-15-0.4-0.54	3.492	3.505	2.00×10^{-4}	0.54
FR-22-20-0.4-0.52	2.353	2.348	2.17×10^{-5}	0.52
FR-29.3-15-0.4-0.65	2.242	2.244	4.49×10^{-6}	0.65
FR-29.3-20-0.4-0.58	1.325	1.314	1.11×10^{-4}	0.58

Tabella 4.3: Valori di velocità sperimentale e CFD, MSE e r_f ottimale per i tessuti FR.

Nella figura 4.8 è invece indicato il valore di reduction factor globale per il Filling Rib, in corrispondenza del minimo del $RMSE$.

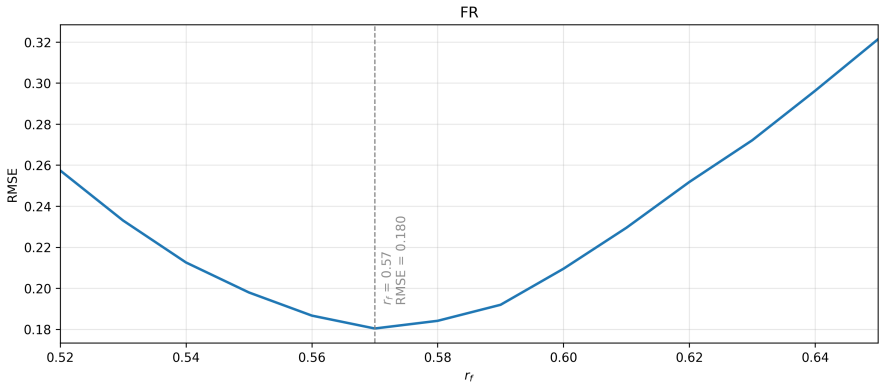


Figura 4.8: Radice dell'errore quadratico medio al variare di r_f.

4.1.4 Twill

Le quattro geometrie per l'armatura Twill sono state analizzate in modo analogo alle precedenti. Questa armatura ha dimensioni paragonabili al FR, come è possibile notare osservando la figura 2.5, e anche in questo caso non è stato necessario aumentare il numero di celle per nessuna simulazione.

Come si evince dalla tabella 4.4 il range di valori di reduction factor da analizzare per individuare il valore globale per l'armatura oscilla tra un valore minimo di 0.38 e un valore massimo di 0.58.

Questo intervallo include il tratto monotono crescente per ogni geometria e questo determina una curva dell'$RMSE$ che individua chiaramente, in corrispondenza del minimo, il valore di r_f globale, mostrato in figura 4.10 e pari a 0.48.

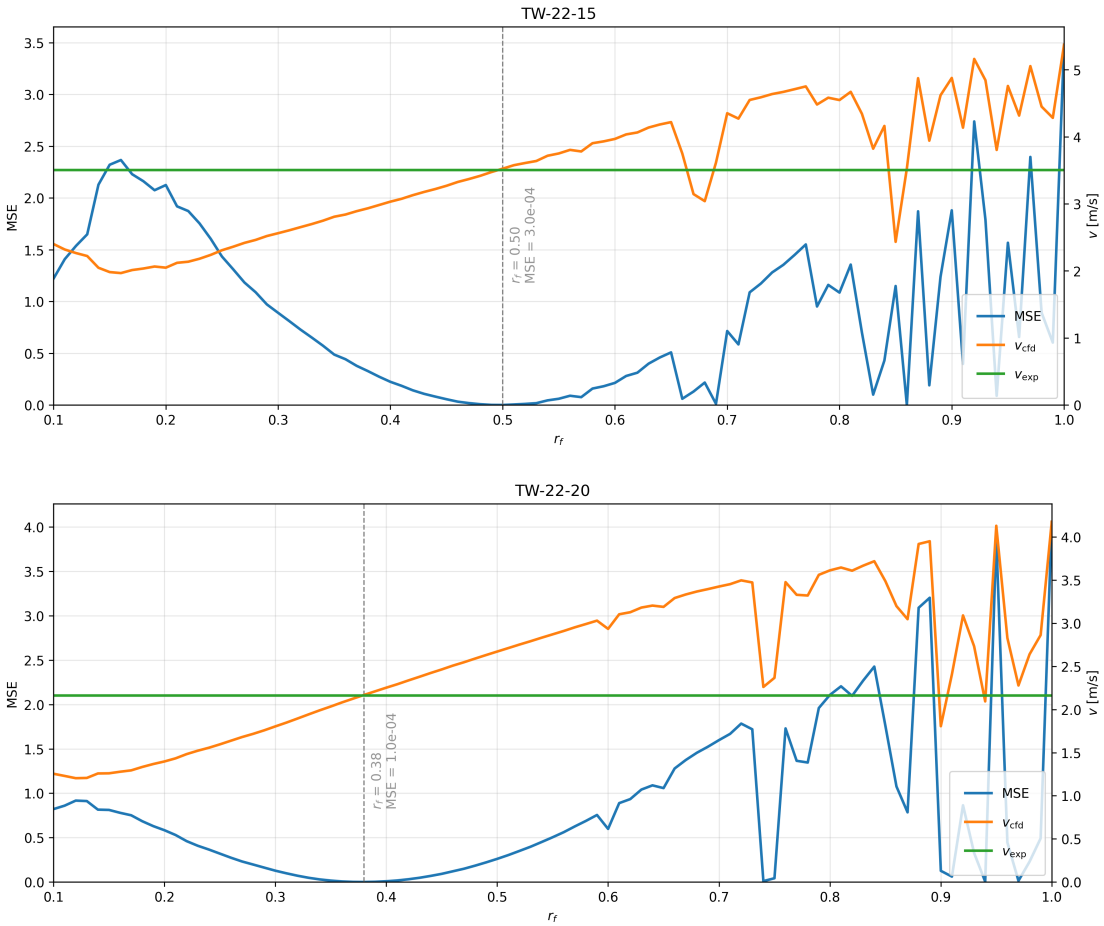


Figura 4.9: Confronto dei grafici TW con r_f ottimale evidenziato.

Simulazioni ottimali (TW)

TW	v_{exp} [m/s]	v_{CFD} [m/s]	MSE	r_f ottimale
TW-22-15	3.506	3.524	3.00×10^{-4}	0.5
TW-22-20	2.162	2.169	1.00×10^{-4}	0.38
TW-29.3-15	1.579	1.590	1.10×10^{-4}	0.58
TW-29.3-20	0.833	0.830	1.16×10^{-5}	0.46

Tabella 4.4: Valori di v_{exp}, v_{CFD}, MSE e r_f ottimale per i tessuti TW.

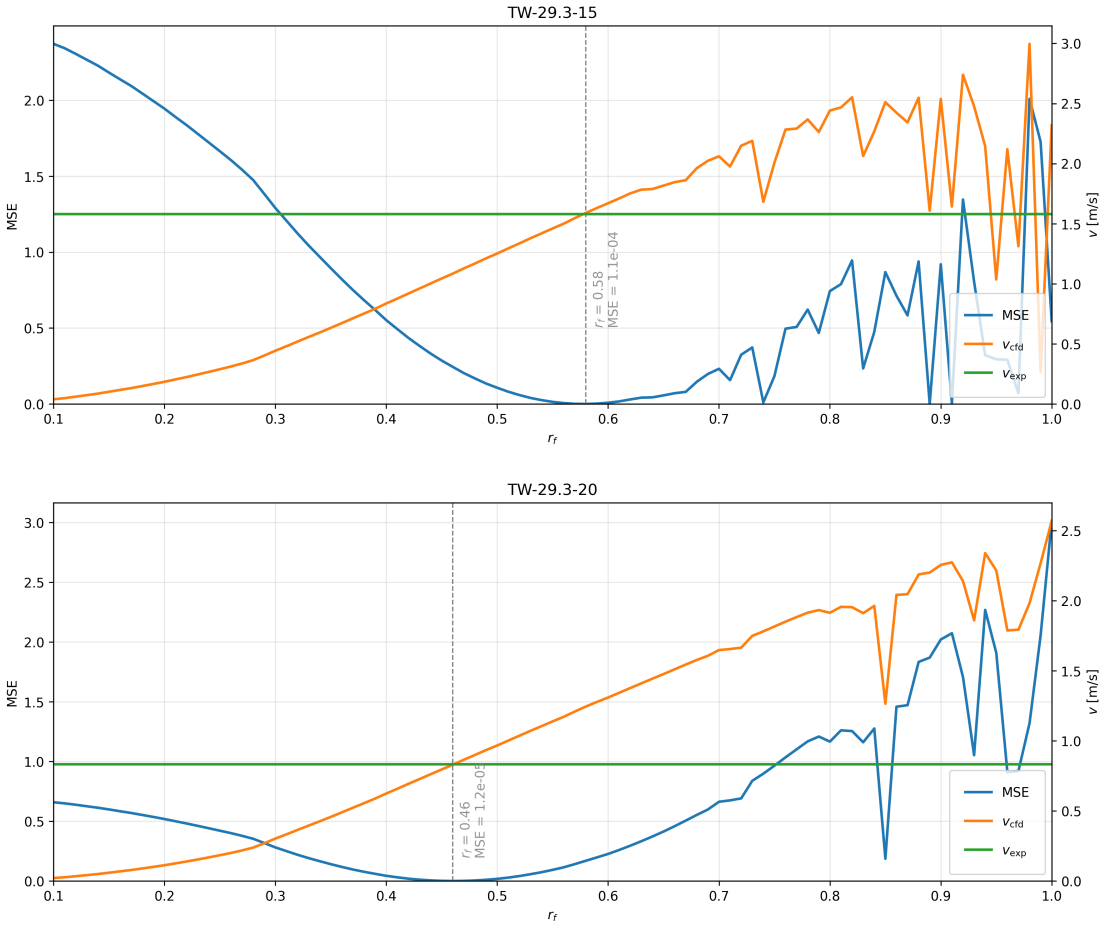


Figura 4.9: Confronto dei grafici TW con r_f ottimale evidenziato.

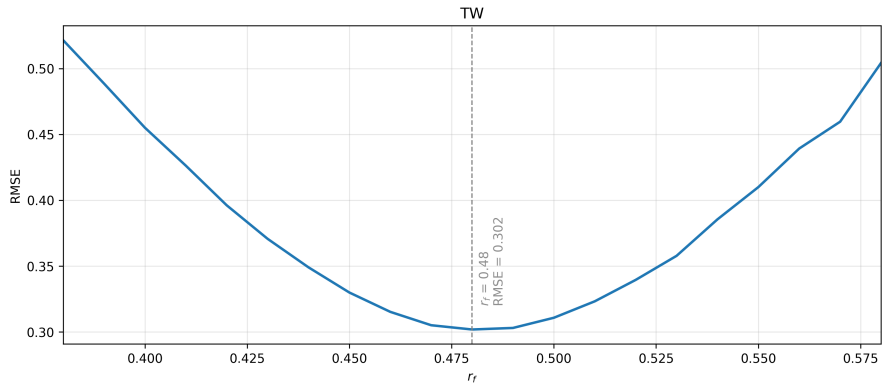


Figura 4.10: Radice dell'errore quadratico medio al variare di r_f.

4.1.5 Influenza di armatura e fabric count sulla permeabilità all'aria

La permeabilità all'aria dei diversi tessuti investigati può essere comparata valutando la velocità v_{cfd} derivante dalle quattro combinazioni di densità di fili di trama e di ordito per le quattro armature. I risultati presentati in figura 4.11 provengono dalle simulazioni con r_f ottimale per ogni geometria.

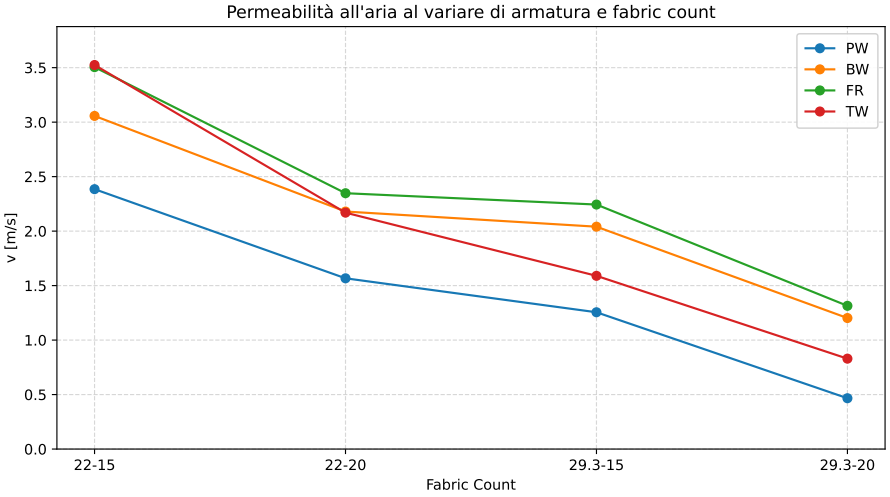


Figura 4.11: Andamento di v_{cfd} al variare di armatura e fabric count

I parametri di design investigati, come armatura e densità di fili di trama e di ordito influenzano la permeabilità del prodotto.

In particolare si nota come geometrie di tessuti meno densi all'interno della stessa armatura determinino una permeabilità maggiore rispetto a geometrie con fabric count più alto. Questo comportamento è determinato dalla presenza di pori con dimensioni maggiori tra i fili che costituiscono il tessuto. In tutte le geometrie

analizzate infatti, come riscontrabile nell'appendice A, il numero di pori creati dai tessuti è sempre pari a sedici, dato dall'intreccio di cinque fili di trama e di cinque fili di ordito, e la loro dimensione gioca un ruolo fondamentale per il passaggio dell'aria attraverso il tessuto, dal momento che i fili sono considerati impermeabili in virtù delle considerazioni del capitolo 2.4.2.

Le dimensioni di questi pori, a parità di diametro del filo, risultano inversamente proporzionali alla densità dei fili, come riscontrabile nelle equazioni 2.57 e 2.58, e questo è in grado di spiegare il comportamento dei tessuti al variare del fabric count.

L'influenza della dimensione dei pori sulla permeabilità valida l'assunzione presentata nel capitolo 2.4.2 e basata sullo studio di Syerko et al. [26] che ha portato a considerare trascurabile la micro-permeabilità dei fili in questo lavoro di tesi.

Analizzando invece i risultati in base alle armature il Plain Weave risulta sempre il meno permeabile per tutte le combinazioni di fabric count, mentre il Filling Rib risulta in generale l'armatura con valori di velocità dell'aria più alta, tranne che per la densità minore, dove il Twill risulta leggermente più permeabile.

Risulta possibile individuare un andamento della velocità simile nel BW e nel FR al variare del fabric count. Questa similitudine deriva dal fatto che entrambi presentano fili flottanti, cioè fili che passano sopra ad altri fili senza intrecciarsi ad ogni intersezione, e che nella loro unità ripetitiva il numero di intrecci, cioè il numero di volte in cui i fili cambiano posizione, risulta identico. Il PW e il TW rappresentano armature particolari sotto questo punto di vista, dato che il Plain Weave non presenta fili flottanti, mentre il Twill è l'armatura che ne presenta il maggior numero. Questo, unito all'effetto della dimensione del poro, può favorire il passaggio dell'aria attraverso il tessuto.

4.2 Rete neurale

Analizzando i dati ottenuti dalle simulazioni CFD è stata studiata la relazione tra la permeabilità dei tessuti all'aria (espressa in termini di velocità) e alcuni parametri di design (armatura e fabric count) e geometrici (reduction factor).

Per ottenere ulteriori risultati andando ad esplorare più valori di fabric count e reduction factor per le quattro geometrie in analisi si è allenata una rete neurale fully connected feed forward, cioè un modello in cui ogni neurone di uno strato è connesso a tutti i neuroni dello strato precedente, come mostrato in figura 4.13, e in cui il segnale scorre dagli input all'output senza cicli o ricorrenze.

4.2.1 Costruzione del dataset

Nel capitolo 4.1 è stato individuato il valore del parametro r_f globale, cioè il valore per il quale ogni geometria risultava univocamente definita.
Le simulazioni risultano quindi validate dai risultati sperimentali identificando, per le quattro armature, quattro valori di reduction factor globale indicati nella tabella 4.5:

r_f globale per ogni armatura

Armatura	r_f globale
PW	0.54
BW	0.58
FR	0.57
TW	0.48

Tabella 4.5: Valori globali di r_f (minimo dell'RMSE) per le quattro armature.

Rappresentando un valore globale, questo r_f descrive l'intera armatura, cioè per ogni combinazione di densità di fili di trama e di ordito. Nella figura 4.12 a sinistra sono mostrate le quattro combinazioni di fabric count utilizzate nelle simulazioni CFD. Per ottenere un dataset di dimensioni adeguate per allenare la rete neurale si sono definiti tre ulteriori valori di densità di fili di trama all'interno dei due valori utilizzati in precedenza e tre ulteriori valori di densità di fili di ordito, anch'essi compresi tra i due valori già noti. Ogni armatura avrà quindi 25 geometrie diverse, con combinazioni di fabric count rappresentate a destra nella figura 4.12, per la descrizione delle quali verranno utilizzati i valori di r_f globale trovati.

Il training e il test della rete neurale sono stati quindi effettuati con queste 100 simulazioni: le quattro armature avranno venticinque combinazioni di fabric count tutte interne al dominio definito dalle quattro densità analizzate nel capitolo 4.1.

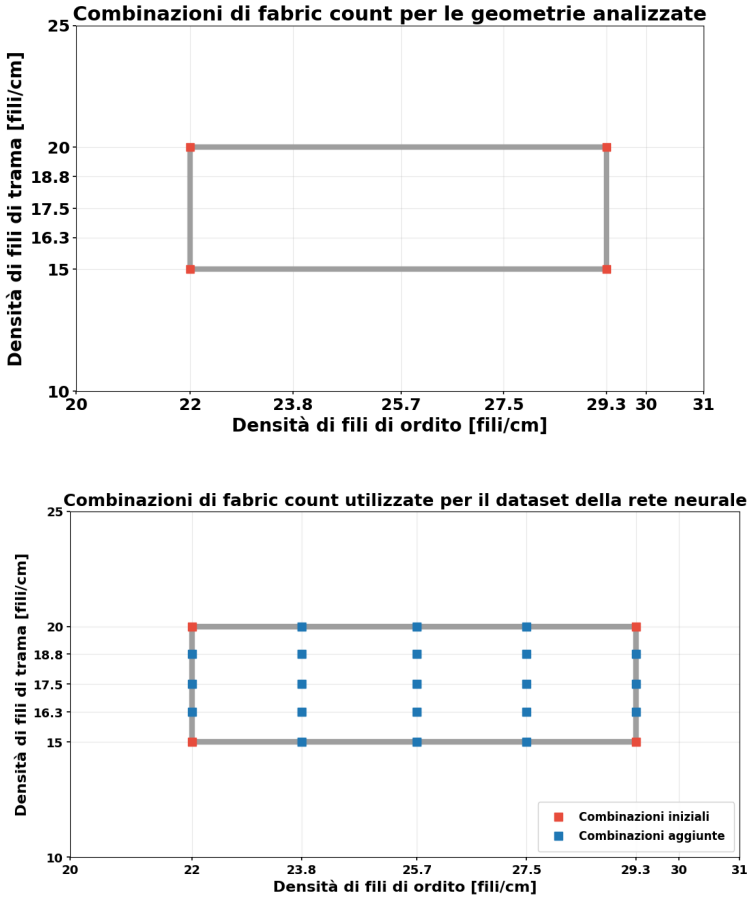


Figura 4.12: Combinazioni di fabric count

4.2.2 Preprocessing

Il dataset è salvato come file .csv ed è composto da cinque variabili: tre numeriche (D_1, D_2 e r_f), una categorica (*strct*) rappresentante l'armatura e per ogni combinazione di queste quattro variabili è riportata la velocità ottenuta dalle simulazioni, che è la variabile target. Il dataset viene letto tramite pandas, che permette di leggere .csv con colonne miste mantenendo la colonna della variabile categorica come testo ed è quindi composto da cinque colonne: *strct*, D_1, D_2, r_f e il target *vel*. [32]

Prima di passare i dati alla rete è necessario effettuare il preprocessing degli stessi utilizzando la libreria scikit-learn. [33]

La variabile categorica che che rappresenta il tipo di armatura è data da una stringa di testo (ad esempio "PW", "BW", "FR", o "TW"). Per utilizzare questi dati in una rete neurale è necessario convertirli in un formato numerico. Ciò si ottiene

attraverso un processo noto come one-hot encoding, un trasformatore importato dal modulo preprocessing di scikit-learn [34], che trasforma ogni valore di testo unico in un vettore binario.

L'etichetta viene trasformata in un vettore binario a quattro colonne, avente entrate tutte pari a 0 tranne per una componente ciascuno, pari a 1, che individua l'armatura. L'aver trasformato l'etichetta in un vettore piuttosto che in un numero è fondamentale per il corretto funzionamento della rete in quanto evita l'instaurarsi di un ordine e di una gerarchia fittizia tra le diverse armature.

Le tre variabili numeriche, insieme al target, vengono poi scalate per ottenere un intervallo comune, compreso tra -1 e 1, attraverso MinMaxScaler: questo viene fatto per omogeneizzare i range delle diverse variabili numeriche e stabilizzare l'ottimizzazione della rete.

Le variabili trattate dalla rete sono quindi sette, che costituiscono l'input finale: le tre variabili numeriche scalate più la variabile categorica trattata con one-hot encoder, che vengono concatenate in un array.

Una volta terminato il preprocessing, gli array di input e target vengono convertiti in tensori PyTorch, in modo che questi dati possano essere utilizzati dalla rete. [35]

4.2.3 Training della rete e architettura

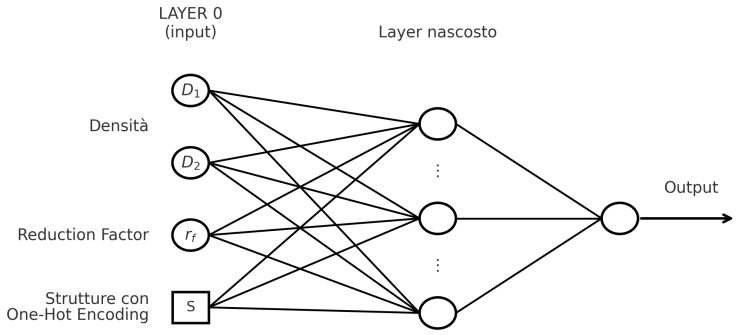


Figura 4.13: Architettura della rete

I parametri di una funzione di predizione vanno appresi su parte del dataset e la loro bontà va valutata sull'altra porzione. Allenare la rete e testarla sugli stessi dati porterebbe infatti ad ottenere un match perfetto, ma informazioni poco utili sulla previsioni di dati non ancora osservati.

Per evitare questo problema si è quindi diviso il dataset destinando l'80% dei dati per il training della rete e il restante 20% per testare le sue prestazioni.

Il modello utilizzato è una rete neurale fully connected (FCNN) con 16 neuroni nell'unico layer nascosto, implementata in PyTorch. Il primo layer lineare proietta

l'array con le variabili concatenate nel layer nascosto, dove viene utilizzata la funzione di attivazione ReLu, mentre l'ultimo layer lineare restituisce una stima scalare della variabile target, cioè la velocità.

Per addestrare il modello, è stato utilizzato il Mean Squared Error (MSE) come loss function, la cui minimizzazione è stata ottenuta tramite l'ottimizzatore Adam. Il learning rate, fissato a 0.001, determina la grandezza del passo di aggiornamento dei pesi del modello a ogni iterazione, guidando la convergenza verso il minimo della funzione di perdita.

Il modello lavora con batch di 10 valori del training set. Per ogni batch vengono aggiornati i pesi. Il numero massimo di epoche, cioè di passaggi completi su tutto il training set, è fissato a 1000. Al termine di ogni epoca si calcola la perdita sul set di validazione, se questa non migliora per 30 epoche consecutive il training della rete si ferma: questa procedura è nota come early stopping.

4.2.4 Valutazione

Al termine dell'allenamento sono state realizzati il diagramma di parità (parity plot) e le funzioni di perdita di training e di test.

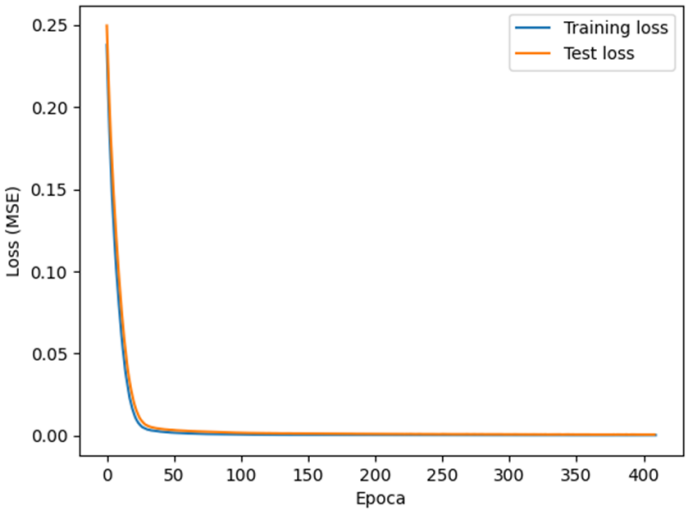


Figura 4.14: Loss function

Le funzioni di perdita sono utili per valutare un possibile overfitting, underfitting e se il modello è in grado di generalizzare i dati che riceve.

Il modello evidenzia una rapida diminuzione delle curve di perdita dopo poche epoche, mostrate in figura 4.14, che raggiungono un asintoto molto basso: questo è indice di una buona generalizzazione e di assenza del fenomeno di overfitting o di

underfitting, cioè quando il modello riporta un errore molto basso sul training set, ma non riesce a mantenere buone performance con il set di validazione o quando il modello presenta un'architettura troppo semplice. Il criterio di early stopping è raggiunto intorno alla quattrocentesima epoca.

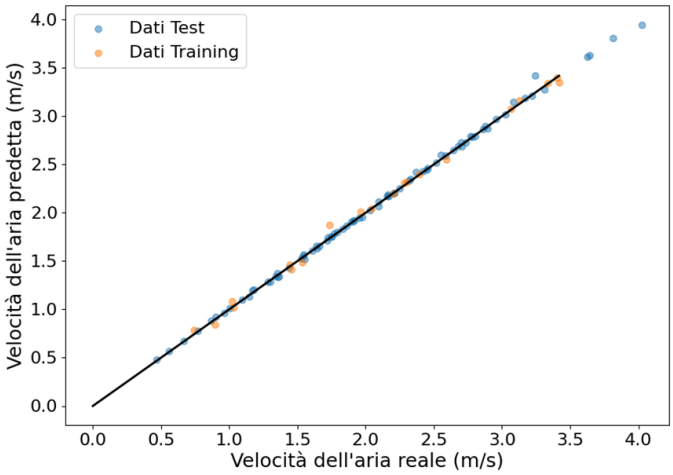


Figura 4.15: Diagramma di parità

Analizzando il diagramma di parità mostrato in figura 4.15, che mette in relazione la velocità predetta dal modello con quella calcolata dalle simulazioni CFD, si nota come entrambi i set di dati (training e test) si addensino lungo la bisettrice, indice di un bias trascurabile e di un comportamento dei dati di test coerente con quelli di training.

Capitolo 5

Conclusioni

L'obiettivo principale del lavoro di questa tesi è stato quello di costruire un modello basato sul machine learning per studiare la permeabilità all'aria dei tessuti, allenando una rete neurale con i risultati ottenuti tramite CFD e validati da prove sperimentali presenti in letteratura.

L'attenzione è stata posta sui tessuti intrecciati, analizzando quattro armature tra le più utilizzate nel mercato dei tessili: Plain Weave (PW), Basket Weave (BW), Filling Rib (FR) e Twill (TW). I risultati ottenuti dalle simulazioni CFD effettuate con OpenFOAM sono stati confrontati con i valori di permeabilità provenienti da prove sperimentali disponibili in letteratura che seguivano la norma ISO 9237:1995.

Sono stati investigati valori di reduction factor compresi tra 0.1 (massimo schiacciamento) e 1 (filo inalterato) ed è stato individuato, per ciascuna geometria, un valore, denominato r_f ottimale, che minimizzasse il MSE tra la velocità calcolata tramite CFD e la velocità calcolata sperimentalmente e disponibile in letteratura.

Analizzando i risultati ottenuti in questa prima fase è stato possibile valutare la correlazione tra la permeabilità e i parametri di design come armatura e fabric count, individuando un andamento decrescente della velocità rispetto alla densità di fili di trama e ordito. Questo comportamento è stato attribuito alla dimensione dei pori, dal momento che ogni geometria è descritta dallo stesso numero di pori e l'aria è costretta a passare al loro interno poiché i fili del tessuto sono stati considerati impermeabili. Un possibile sviluppo futuro potrebbe concentrarsi sul comportamento del filo investigando la sua micro-permeabilità e i fattori che la influenzano, come il tipo di fibre e il loro arrangiamento. La forte correlazione tra i valori di fabric count, dai quali dipendono le dimensioni dei pori del tessuto, e i risultati di permeabilità avvalora inoltre la scelta intrapresa di modellare i fili dei tessuti come impermeabili, trascurando l'apporto della micro-porosità.

L'approccio che porta alla definizione di un reduction factor globale invece ha aperto ad un possibile ulteriore studio del comportamento della specifica armatura. Per ampliare l'indagine e variare quindi altri parametri di design, è infatti necessario

trovare una descrizione meno univoca rispetto al r_f ottimale. Per ogni armatura si è quindi successivamente individuato un valore di reduction factor, denominato r_f globale, che fosse in grado di descrivere il suo comportamento al passaggio dell'aria al variare delle densità di fili di trama e di ordito. Questo valore di reduction factor si è individuato tra il valore massimo e minimo dell'r_f ottimale per ogni combinazione di fabric count all'interno dell'armatura, in modo tale che minimizzasse il $RMSE$ tra le velocità.

Disponendo di questo valore di r_f si è ampliato il range di combinazioni di fabric count. La scelta di mantenersi all'interno dei valori minimi e massimi di D_1 e D_2 utilizzati nella prima fase di validazione del modello permette di sfruttare anche questi nuovi risultati, seppur non validati direttamente dalle corrispondenti prove sperimentali.

L'allenamento della rete neurale fully connected ha restituito risultati soddisfacenti, analizzati sia tramite la loss function dei due set di dati, sia tramite il diagramma di parità che hanno indicato una corretta stima della permeabilità in relazione alla complessità dell'architettura della rete. Il vantaggio dell'utilizzo della rete neurale per la stima della permeabilità dei tessuti risiede nella possibilità di calcolare, con costi computazionali trascurabili rispetto all'analisi tramite CFD, il valore di permeabilità a partire dai parametri geometrici e di design individuati, cioè armatura, densità del tessuto e reduction factor, offrendo uno strumento di indagine ancora prima della fase produttiva, che risulterebbe molto utile nell'ingegneria di processo.

Appendice A

Codici Python per generare tessuti con TexGen

In questa appendice viene mostrato il codice Python per generare tessuti utilizzando TexGen

Listing A.1: Script TexGen

```
1
2  import TexGen.Core as TG
3  from TexGen.Core import *
4  #import numpy as np
5  import os
6  import csv
7
8  # 1) Setting geometrical parameters
9  # NB @1 always refers to warp (ordito: verticale) yarns and @2 refer to weft (
   trama: orizzontale) yarns
10
11 # Setting yarns density, expressed in [picks/cm]
12 D1_values = [22, 23.8, 25.7, 27.5, 29.3]
13 D2_values = [15, 16.3, 17.5, 18.8, 20]
14
15 #[cm]
16 d1 = 0.0263
17 d2 = 0.0282
18
19 # Parametri variabili
20 v_values = [0.4]  # Poisson's Ratio variabile
21 #r_f_values = [round(x * 0.01, 2) for x in range(10, 101)]
22 r_f_values = [0.57]
23 #weave_type = ["PW", "BW", "TW", "FR"]
24 weave_type = ["FR"]
25
```

```
# Directory di output per salvare i file generati
output_dir = "Textiles"
os.makedirs(output_dir, exist_ok=True)  # Ensure the output directory exists

# Funzione per generare e salvare la geometria
def generate_and_save_textile(v, r_f, D1, D2, weave):
    s1 = 1 / D1
    s2 = 1 / D2
    t1 = r_f * d1
    t2 = r_f * d2
    w1 = d1 + v * (d1 - t1)
    w2 = d2 + v * (d2 - t2)

    Textile = TG.CTextile()

    # Creazione e definizione delle caratteristiche dei filati
    Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]

    if weave == "PW":
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, -s2, t1)))
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, 0)))
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, s2, t1)))
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 2 * s2, 0)))
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 3 * s2, t1)))
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 4 * s2, 0)))
        Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 5*s2, t1)))

        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, -s2, 0)))
        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 0, t1)))
        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, 0)))
        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 2 * s2, t1)))
        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 3 * s2, 0)))
        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 4 * s2, t1)))
        Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 5 * s2, 0)))

        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, -s2, t1)))
        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 0, 0)))
        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, s2, t1)))
        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 2 * s2, 0)))
        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 3 * s2, t1)))
        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 4 * s2, 0)))
        Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 5 * s2, t1)))

        Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, -s2, 0)))
        Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 0, t1)))
        Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, s2, 0)))
        Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 2 * s2, t1)))
        Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 3 * s2, 0)))
        Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 4 * s2, t1)))
```

```
75          Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 5 * s2, 0)))
76
77          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, -s2, t1)))
78          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 0, 0)))
79          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, s2, t1)))
80          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 2 * s2, 0)))
81          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 3 * s2, t1)))
82          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 4 * s2, 0)))
83          Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 5 * s2, t1)))
84
85          for Yarn in Yarns:
86              Yarn.AssignInterpolation(TG.CInterpolationCubic())
87              Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w1,
    t1)))
88              Yarn.SetResolution(20)
89              Yarn.AddRepeat(TG.XYZ(7*s1, 0, 0))
90              Yarn.AddRepeat(TG.XYZ(0, 7*s2, 0))
91              Textile.AddYarn(Yarn)
92
93          Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
94          Yarns[0].AddNode(TG.CNode(TG.XYZ(-s1, 0, t2)))
95          Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, t2)))
96          Yarns[0].AddNode(TG.CNode(TG.XYZ(s1, 0, 0)))
97          Yarns[0].AddNode(TG.CNode(TG.XYZ(2 * s1, 0, t2)))
98          Yarns[0].AddNode(TG.CNode(TG.XYZ(3 * s1, 0, 0)))
99          Yarns[0].AddNode(TG.CNode(TG.XYZ(4 * s1, 0, t2)))
100         Yarns[0].AddNode(TG.CNode(TG.XYZ(5 * s1, 0, 0)))
101
102         Yarns[1].AddNode(TG.CNode(TG.XYZ(-s1, s2, t2)))
103         Yarns[1].AddNode(TG.CNode(TG.XYZ(0, s2, 0)))
104         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, t2)))
105         Yarns[1].AddNode(TG.CNode(TG.XYZ(2 * s1, s2, 0)))
106         Yarns[1].AddNode(TG.CNode(TG.XYZ(3 * s1, s2, t2)))
107         Yarns[1].AddNode(TG.CNode(TG.XYZ(4 * s1, s2, 0)))
108         Yarns[1].AddNode(TG.CNode(TG.XYZ(5 * s1, s2, t2)))
109
110         Yarns[2].AddNode(TG.CNode(TG.XYZ(-s1, 2 * s2, 0)))
111         Yarns[2].AddNode(TG.CNode(TG.XYZ(0, 2 * s2, t2)))
112         Yarns[2].AddNode(TG.CNode(TG.XYZ(s1, 2 * s2, 0)))
113         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 2 * s2, t2)))
114         Yarns[2].AddNode(TG.CNode(TG.XYZ(3 * s1, 2 * s2, 0)))
115         Yarns[2].AddNode(TG.CNode(TG.XYZ(4 * s1, 2 * s2, t2)))
116         Yarns[2].AddNode(TG.CNode(TG.XYZ(5 * s1, 2 * s2, 0)))
117
118         Yarns[3].AddNode(TG.CNode(TG.XYZ(-s1, 3 * s2, t2)))
119         Yarns[3].AddNode(TG.CNode(TG.XYZ(0, 3 * s2, 0)))
120         Yarns[3].AddNode(TG.CNode(TG.XYZ(s1, 3 * s2, t2)))
121         Yarns[3].AddNode(TG.CNode(TG.XYZ(2 * s1, 3 * s2, 0)))
122         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 3 * s2, t2)))
```

```
123         Yarns[3].AddNode(TG.CNode(TG.XYZ(4 * s1, 3 * s2, 0)))
124         Yarns[3].AddNode(TG.CNode(TG.XYZ(5 * s1, 3 * s2, t2)))
125 
126         Yarns[4].AddNode(TG.CNode(TG.XYZ(-s1, 4 * s2, 0)))
127         Yarns[4].AddNode(TG.CNode(TG.XYZ(0, 4 * s2, t2)))
128         Yarns[4].AddNode(TG.CNode(TG.XYZ(s1, 4 * s2, 0)))
129         Yarns[4].AddNode(TG.CNode(TG.XYZ(2 * s1, 4 * s2, t2)))
130         Yarns[4].AddNode(TG.CNode(TG.XYZ(3 * s1, 4 * s2, 0)))
131         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 4 * s2, t2)))
132         Yarns[4].AddNode(TG.CNode(TG.XYZ(5 * s1, 4 * s2, 0)))
133 
134         for Yarn in Yarns:
135             Yarn.AssignInterpolation(TG.CInterpolationCubic())
136             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w2,
    t2)))
137             Yarn.SetResolution(20)
138             Yarn.AddRepeat(TG.XYZ(6 * s1, 0, 0))
139             Yarn.AddRepeat(TG.XYZ(0, 6 * s2, 0))
140             Textile.AddYarn(Yarn)
141 
142     if weave == "BW":
143         Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
144 
145         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, -s2, 0)))
146         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, 0)))
147         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, s2, t1)))
148         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 2 * s2, t1)))
149         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 3 * s2, 0)))
150         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 4 * s2, 0)))
151         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 5 * s2, t1)))
152 
153         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, -s2, 0)))
154         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 0, 0)))
155         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, t1)))
156         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 2 * s2, t1)))
157         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 3 * s2, 0)))
158         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 4 * s2, 0)))
159         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 5 * s2, t1)))
160 
161         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, -s2, t1)))
162         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 0, t1)))
163         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, s2, 0)))
164         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 2 * s2, 0)))
165         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 3 * s2, t1)))
166         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 4 * s2, t1)))
167         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 5 * s2, 0)))
168 
169         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, -s2, t1)))
170         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 0, t1)))
```

```
171         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, s2, 0)))
172         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 2 * s2, 0)))
173         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 3 * s2, t1)))
174         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 4 * s2, t1)))
175         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 5 * s2, 0)))
176 
177         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, -s2, 0)))
178         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 0, 0)))
179         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, s2, t1)))
180         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 2 * s2, t1)))
181         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 3 * s2, 0)))
182         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 4 * s2, 0)))
183         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 5 * s2, t1)))
184 
185         for Yarn in Yarns:
186             Yarn.AssignInterpolation(TG.CInterpolationCubic())
187             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w1,
    t1)))
188             Yarn.SetResolution(20)
189             Yarn.AddRepeat(TG.XYZ(5 * s1, 0, 0))
190             Yarn.AddRepeat(TG.XYZ(0, 5 * s2, 0))
191             Textile.AddYarn(Yarn)
192 
193         Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
194 
195         Yarns[0].AddNode(TG.CNode(TG.XYZ(-s1, 0, 0)))
196         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, t2)))
197         Yarns[0].AddNode(TG.CNode(TG.XYZ(s1, 0, t2)))
198         Yarns[0].AddNode(TG.CNode(TG.XYZ(2 * s1, 0, 0)))
199         Yarns[0].AddNode(TG.CNode(TG.XYZ(3 * s1, 0, 0)))
200         Yarns[0].AddNode(TG.CNode(TG.XYZ(4 * s1, 0, t2)))
201         Yarns[0].AddNode(TG.CNode(TG.XYZ(5 * s1, 0, t2)))
202 
203         Yarns[1].AddNode(TG.CNode(TG.XYZ(-s1, s2, t2)))
204         Yarns[1].AddNode(TG.CNode(TG.XYZ(0, s2, 0)))
205         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, 0)))
206         Yarns[1].AddNode(TG.CNode(TG.XYZ(2 * s1, s2, t2)))
207         Yarns[1].AddNode(TG.CNode(TG.XYZ(3 * s1, s2, t2)))
208         Yarns[1].AddNode(TG.CNode(TG.XYZ(4 * s1, s2, 0)))
209         Yarns[1].AddNode(TG.CNode(TG.XYZ(5 * s1, s2, 0)))
210 
211         Yarns[2].AddNode(TG.CNode(TG.XYZ(-s1, 2 * s2, t2)))
212         Yarns[2].AddNode(TG.CNode(TG.XYZ(0, 2 * s2, 0)))
213         Yarns[2].AddNode(TG.CNode(TG.XYZ(s1, 2 * s2, 0)))
214         Yarns[2].AddNode(TG.CNode(TG.XYZ(2 * s1, 2 * s2, t2)))
215         Yarns[2].AddNode(TG.CNode(TG.XYZ(3 * s1, 2 * s2, t2)))
216         Yarns[2].AddNode(TG.CNode(TG.XYZ(4 * s1, 2 * s2, 0)))
217         Yarns[2].AddNode(TG.CNode(TG.XYZ(5 * s1, 2 * s2, 0)))
218 
```

```
219         Yarns[3].AddNode(TG.CNode(TG.XYZ(-s1, 3 * s2, 0)))
220         Yarns[3].AddNode(TG.CNode(TG.XYZ(0, 3 * s2, t2)))
221         Yarns[3].AddNode(TG.CNode(TG.XYZ(s1, 3 * s2, t2)))
222         Yarns[3].AddNode(TG.CNode(TG.XYZ(2 * s1, 3 * s2, 0)))
223         Yarns[3].AddNode(TG.CNode(TG.XYZ(3 * s1, 3 * s2, 0)))
224         Yarns[3].AddNode(TG.CNode(TG.XYZ(4 * s1, 3 * s2, t2)))
225         Yarns[3].AddNode(TG.CNode(TG.XYZ(5 * s1, 3 * s2, t2)))
226 
227         Yarns[4].AddNode(TG.CNode(TG.XYZ(-s1, 4 * s2, 0)))
228         Yarns[4].AddNode(TG.CNode(TG.XYZ(0, 4 * s2, t2)))
229         Yarns[4].AddNode(TG.CNode(TG.XYZ(s1, 4 * s2, t2)))
230         Yarns[4].AddNode(TG.CNode(TG.XYZ(2 * s1, 4 * s2, 0)))
231         Yarns[4].AddNode(TG.CNode(TG.XYZ(3 * s1, 4 * s2, 0)))
232         Yarns[4].AddNode(TG.CNode(TG.XYZ(4 * s1, 4 * s2, t2)))
233         Yarns[4].AddNode(TG.CNode(TG.XYZ(5 * s1, 4 * s2, t2)))
234 
235         for Yarn in Yarns:
236             Yarn.AssignInterpolation(TG.CInterpolationCubic())
237             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w2,
    t2)))
238             Yarn.SetResolution(20)
239             Yarn.AddRepeat(TG.XYZ(5 * s1, 0, 0))
240             Yarn.AddRepeat(TG.XYZ(0, 5 * s2, 0))
241             Textile.AddYarn(Yarn)
242 
243     if weave == "FR":
244         Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
245 
246         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, -s2, 0)))
247         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0,  t1)))
248         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, s2, t1)))
249         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 2 * s2, 0)))
250         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 3 * s2, 0)))
251         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 4 * s2, t1)))
252         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 5 * s2, t1)))
253 
254         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, -s2, t1)))
255         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 0, 0)))
256         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, 0)))
257         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 2*s2, t1)))
258         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 3*s2, t1)))
259         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 4*s2, 0)))
260         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 5*s2, 0)))
261 
262         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, -s2, 0)))
263         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 0, t1)))
264         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, s2, t1)))
265         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 2*s2, 0)))
266         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 3*s2, 0)))
```

```
267         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 4*s2, t1)))
268         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 5*s2, t1)))
269 
270         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, -s2, t1)))
271         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 0, 0)))
272         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, s2, 0)))
273         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 2*s2, t1)))
274         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 3*s2, t1)))
275         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 4*s2, 0)))
276         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 5*s2, 0)))
277 
278         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, -s2, 0)))
279         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 0, t1)))
280         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, s2, t1)))
281         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 2*s2, 0)))
282         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 3*s2, 0)))
283         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 4*s2, t1)))
284         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 5*s2, t1)))
285 
286         for Yarn in Yarns:
287             Yarn.AssignInterpolation(TG.CInterpolationCubic())
288             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w1,
    t1)))
289             Yarn.SetResolution(20)
290             Yarn.AddRepeat(TG.XYZ(5 * s1, 0, 0))
291             Yarn.AddRepeat(TG.XYZ(0, 5 * s2, 0))
292             Textile.AddYarn(Yarn)
293 
294         Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
295 
296         Yarns[0].AddNode(TG.CNode(TG.XYZ(-s1, 0, t2)))
297         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, 0)))
298         Yarns[0].AddNode(TG.CNode(TG.XYZ(s1, 0, t2)))
299         Yarns[0].AddNode(TG.CNode(TG.XYZ(2*s1, 0, 0)))
300         Yarns[0].AddNode(TG.CNode(TG.XYZ(3*s1, 0, t2 )))
301         Yarns[0].AddNode(TG.CNode(TG.XYZ(4*s1, 0, 0)))
302         Yarns[0].AddNode(TG.CNode(TG.XYZ(5*s1, 0, t2)))
303 
304         Yarns[1].AddNode(TG.CNode(TG.XYZ(-s1, s2, t2)))
305         Yarns[1].AddNode(TG.CNode(TG.XYZ(0, s2, 0)))
306         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, t2)))
307         Yarns[1].AddNode(TG.CNode(TG.XYZ(2*s1, s2, 0)))
308         Yarns[1].AddNode(TG.CNode(TG.XYZ(3*s1, s2, t2)))
309         Yarns[1].AddNode(TG.CNode(TG.XYZ(4*s1, s2, 0)))
310         Yarns[1].AddNode(TG.CNode(TG.XYZ(5*s1, s2, t2)))
311 
312         Yarns[2].AddNode(TG.CNode(TG.XYZ(-s1, 2*s2, 0)))
313         Yarns[2].AddNode(TG.CNode(TG.XYZ(0, 2*s2, t2)))
314         Yarns[2].AddNode(TG.CNode(TG.XYZ(s1, 2*s2, 0)))
```

```
315         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 2*s2, t2)))
316         Yarns[2].AddNode(TG.CNode(TG.XYZ(3*s1, 2*s2, 0)))
317         Yarns[2].AddNode(TG.CNode(TG.XYZ(4*s1, 2*s2, t2)))
318         Yarns[2].AddNode(TG.CNode(TG.XYZ(5*s1, 2*s2, 0)))
319 
320         Yarns[3].AddNode(TG.CNode(TG.XYZ(-s1, 3*s2, 0)))
321         Yarns[3].AddNode(TG.CNode(TG.XYZ(0, 3*s2, t2)))
322         Yarns[3].AddNode(TG.CNode(TG.XYZ(s1, 3*s2, 0)))
323         Yarns[3].AddNode(TG.CNode(TG.XYZ(2*s1, 3*s2, t2)))
324         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 3*s2, 0)))
325         Yarns[3].AddNode(TG.CNode(TG.XYZ(4*s1, 3*s2, t2)))
326         Yarns[3].AddNode(TG.CNode(TG.XYZ(5*s1, 3*s2, 0)))
327 
328         Yarns[4].AddNode(TG.CNode(TG.XYZ(-s1, 4*s2, t2)))
329         Yarns[4].AddNode(TG.CNode(TG.XYZ(0, 4*s2, 0)))
330         Yarns[4].AddNode(TG.CNode(TG.XYZ(s1, 4*s2, t2)))
331         Yarns[4].AddNode(TG.CNode(TG.XYZ(2*s1, 4*s2, 0)))
332         Yarns[4].AddNode(TG.CNode(TG.XYZ(3*s1, 4*s2, t2)))
333         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 4*s2, 0)))
334         Yarns[4].AddNode(TG.CNode(TG.XYZ(5*s1, 4*s2, t2)))
335 
336         for Yarn in Yarns:
337             Yarn.AssignInterpolation(TG.CInterpolationCubic())
338             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w2,
    t2)))
339             Yarn.SetResolution(20)
340             Yarn.AddRepeat(TG.XYZ(5 * s1, 0, 0))
341             Yarn.AddRepeat(TG.XYZ(0, 5 * s2, 0))
342             Textile.AddYarn(Yarn)
343 
344     if weave_type == "TW":
345         Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
346 
347         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, -s2, 0)))
348         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, t1)))
349         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, s2, 0)))
350         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 2*s2, 0)))
351         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 3*s2, 0)))
352         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 4*s2, t1)))
353         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 5*s2, 0)))
354 
355         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, -s2, 0)))
356         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 0, 0)))
357         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, t1)))
358         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 2*s2, 0)))
359         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 3*s2, 0)))
360         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 4*s2, 0)))
361         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, 5*s2, t1)))
362 
```

```
363         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, -s2, 0)))
364         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 0, 0)))
365         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, s2, 0)))
366         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 2*s2, t1)))
367         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 3*s2, 0)))
368         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 4*s2, 0)))
369         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 5*s2, 0)))
370
371         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, -s2, t1)))
372         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 0, 0)))
373         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, s2, 0)))
374         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 2*s2, 0)))
375         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 3*s2, t1)))
376         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 4*s2, 0)))
377         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 5*s2, 0)))
378
379         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, -s2, 0)))
380         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 0, t1)))
381         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, s2, 0)))
382         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 2*s2, 0)))
383         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 3*s2, 0)))
384         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 4*s2, t1)))
385         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 5*s2, 0)))
386
387         for Yarn in Yarns:
388             Yarn.AssignInterpolation(TG.CInterpolationCubic())
389             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w1,
    t1)))
390             Yarn.SetResolution(20)
391             Yarn.AddRepeat(TG.XYZ(5 * s1, 0, 0))
392             Yarn.AddRepeat(TG.XYZ(0, 5 * s2, 0))
393             Textile.AddYarn(Yarn)
394
395         Yarns = [TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn(), TG.CYarn()]
396
397         Yarns[0].AddNode(TG.CNode(TG.XYZ(-s1, 0, t2)))
398         Yarns[0].AddNode(TG.CNode(TG.XYZ(0, 0, 0)))
399         Yarns[0].AddNode(TG.CNode(TG.XYZ(s1, 0, t2)))
400         Yarns[0].AddNode(TG.CNode(TG.XYZ(2*s1, 0, t2)))
401         Yarns[0].AddNode(TG.CNode(TG.XYZ(3*s1, 0, t2 )))
402         Yarns[0].AddNode(TG.CNode(TG.XYZ(4*s1, 0, 0)))
403         Yarns[0].AddNode(TG.CNode(TG.XYZ(5*s1, 0, t2)))
404
405         Yarns[1].AddNode(TG.CNode(TG.XYZ(-s1, s2, t2)))
406         Yarns[1].AddNode(TG.CNode(TG.XYZ(0, s2, t2)))
407         Yarns[1].AddNode(TG.CNode(TG.XYZ(s1, s2, 0)))
408         Yarns[1].AddNode(TG.CNode(TG.XYZ(2*s1, s2, t2)))
409         Yarns[1].AddNode(TG.CNode(TG.XYZ(3*s1, s2, t2)))
410         Yarns[1].AddNode(TG.CNode(TG.XYZ(4*s1, s2, t2)))
```

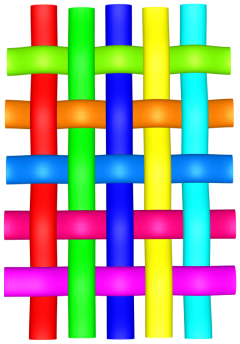
```
411         Yarns[1].AddNode(TG.CNode(TG.XYZ(5*s1, s2, 0)))
412 
413         Yarns[2].AddNode(TG.CNode(TG.XYZ(-s1, 2*s2, t2)))
414         Yarns[2].AddNode(TG.CNode(TG.XYZ(0, 2*s2, t2)))
415         Yarns[2].AddNode(TG.CNode(TG.XYZ(s1, 2*s2, t2)))
416         Yarns[2].AddNode(TG.CNode(TG.XYZ(2*s1, 2*s2, 0)))
417         Yarns[2].AddNode(TG.CNode(TG.XYZ(3*s1, 2*s2, t2)))
418         Yarns[2].AddNode(TG.CNode(TG.XYZ(4*s1, 2*s2, t2)))
419         Yarns[2].AddNode(TG.CNode(TG.XYZ(5*s1, 2*s2, t2)))
420 
421         Yarns[3].AddNode(TG.CNode(TG.XYZ(-s1, 3*s2, 0)))
422         Yarns[3].AddNode(TG.CNode(TG.XYZ(0, 3*s2, t2)))
423         Yarns[3].AddNode(TG.CNode(TG.XYZ(s1, 3*s2, t2)))
424         Yarns[3].AddNode(TG.CNode(TG.XYZ(2*s1, 3*s2, t2)))
425         Yarns[3].AddNode(TG.CNode(TG.XYZ(3*s1, 3*s2, 0)))
426         Yarns[3].AddNode(TG.CNode(TG.XYZ(4*s1, 3*s2, t2)))
427         Yarns[3].AddNode(TG.CNode(TG.XYZ(5*s1, 3*s2, t2)))
428 
429         Yarns[4].AddNode(TG.CNode(TG.XYZ(-s1, 4*s2, t2)))
430         Yarns[4].AddNode(TG.CNode(TG.XYZ(0, 4*s2, 0)))
431         Yarns[4].AddNode(TG.CNode(TG.XYZ(s1, 4*s2, t2)))
432         Yarns[4].AddNode(TG.CNode(TG.XYZ(2*s1, 4*s2, t2)))
433         Yarns[4].AddNode(TG.CNode(TG.XYZ(3*s1, 4*s2, t2)))
434         Yarns[4].AddNode(TG.CNode(TG.XYZ(4*s1, 4*s2, 0)))
435         Yarns[4].AddNode(TG.CNode(TG.XYZ(5*s1, 4*s2, t2)))
436 
437         for Yarn in Yarns:
438             Yarn.AssignInterpolation(TG.CInterpolationCubic())
439             Yarn.AssignSection(TG.CYarnSectionConstant(TG.CSectionEllipse(w2,
    t2)))
440             Yarn.SetResolution(20)
441             Yarn.AddRepeat(TG.XYZ(5 * s1, 0, 0))
442             Yarn.AddRepeat(TG.XYZ(0, 5 * s2, 0))
443             Textile.AddYarn(Yarn)
444 
445     Textile.AssignDomain(TG.CDomainPlanes(TG.XYZ(0, 0, -1.5*t1), TG.XYZ(4*s1,
    4*s2, 1.5*t2)))
446     textile_name = f"{weave_type}-{D1}-{D2}-{v}-{r_f}"
447     TG.AddTextile(textile_name, Textile)
448 
449     mesh = TG.CMesh()
450     TrimDomain = False
451     Textile.AddSurfaceToMesh(mesh, TrimDomain)
452     mesh.SaveToSTL(os.path.join(output_dir, textile_name + ".stl"))
453 
454     csv_path = os.path.join(output_dir, "features.csv")
455     file_exists = os.path.isfile(csv_path)
456 
457     with open(csv_path, mode='a', newline='') as csv_file:
```

```
458         writer = csv.writer(csv_file)
459         if not file_exists:
460             writer.writerow(['strct', 'D1', 'D2', 'v', 'rf'])
461         writer.writerow([weave_type, D1, D2, v, r_f])
462 
463 # Loop through all combinations of v and r_f to generate textiles
464 for weave_type in weave_type:
465     for D1 in D1_values:
466         for D2 in D2_values:
467             for v in v_values:
468                 for r_f in r_f_values:
469                     generate_and_save_textile(v, r_f, D1, D2, weave_type)
```

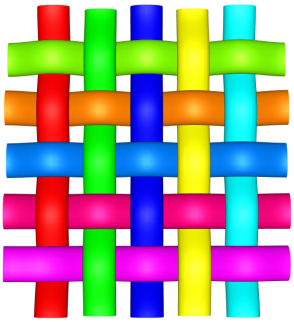
Appendice B

Immagini dei tessuti generati con TexGen

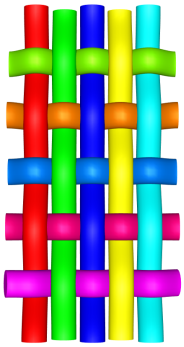
B.1 PW



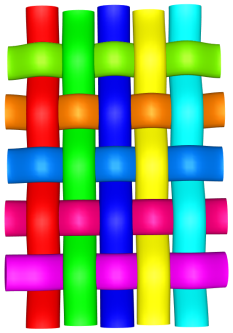
PW-22-15-0.4-0.43



PW-22-20-0.4-0.5



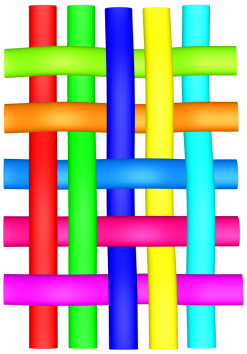
PW-29.3-15-0.4-0.67



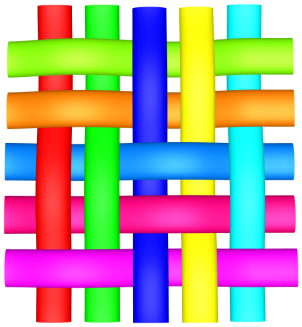
PW-29.3-20-0.4-0.54

Figura B.1: Immagini dei tessuti PW con r_f ottimale per ciascuna combinazione di fabric count

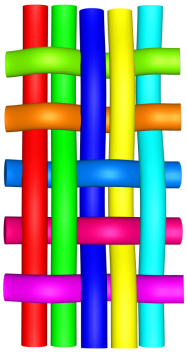
B.2 BW



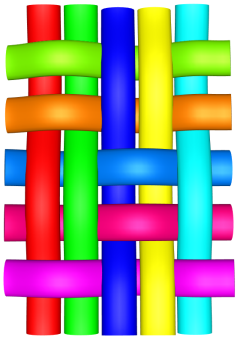
BW-22-15-0.4-0.38



BW-22-20-0.4-0.36



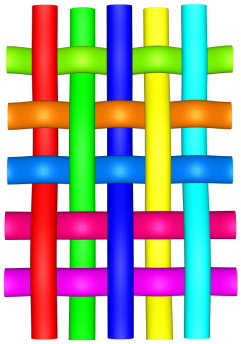
BW-29.3-15-0.4-0.65



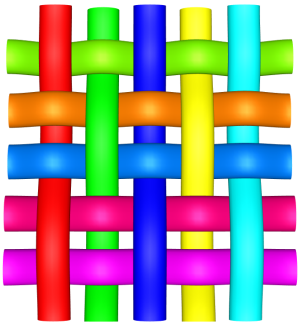
BW-29.3-20-0.4-0.54

Figura B.2: Immagini dei tessuti BW con r_f ottimale per ciascuna combinazione di fabric count

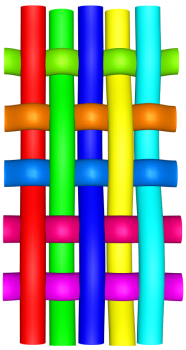
B.3 FR



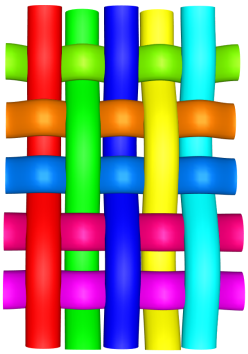
FR-22-15-0.4-0.54



FR-22-20-0.4-0.52



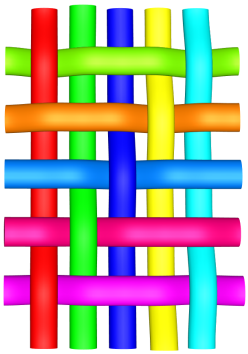
FR-29.3-15-0.4-0.65



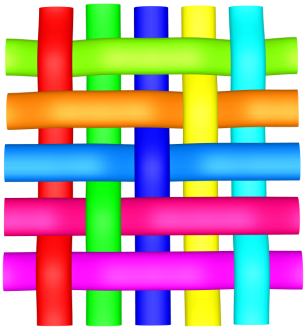
FR-29.3-20-0.4-0.58

Figura B.3: Immagini dei tessuti FR con r_f ottimale per ciascuna combinazione di fabric count

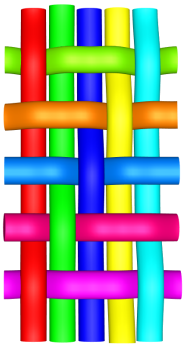
B.4 TW



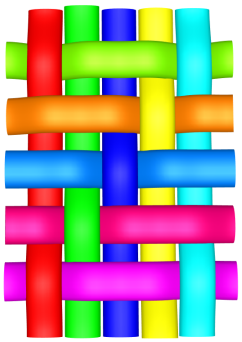
TW-22-15-0.4-0.50



TW-22-20-0.4-0.38



TW-29.3-15-0.4-0.58



TW-29.3-20-0.4-0.46

Figura B.4: Immagini dei tessuti TW con r_f ottimale per ciascuna combinazione di fabric count

Bibliografia

[1] IMARC Group. *Textile Market Size, Share, Trends and Forecast by Raw Material, Product, Application, and Region, 2025-2033*. Report ID: SR112025A3728; Accessed: 2025-09-03. 2025. URL: `https://www.imarcgroup.com/textile-market` (cit. a p. 1).

[2] Fortune Business Insights. *Technical Textiles Market Size, Share & Industry Analysis, By Product Type (Agrotech, Buildtech, Clothtech, Geotech, Hometech, Indutech, Meditech, Mobiltech, Packtech, Protech, Sportech, and Oekotech), By Fiber Type (Natural and Synthetic), and Regional Forecast, 2025-2032*. Report ID: FBI102716; Last updated: 18 Aug 2025; Accessed: 2025-09-04. 2025. URL: `https://www.fortunebusinessinsights.com/technical-textiles-market-102716` (cit. a p. 1).

[3] A. Richard Horrocks e Subhash C. Anand, cur. *Handbook of Technical Textiles*. Woodhead Publishing Series in Textiles. 1st ed. Cambridge: Woodhead Publishing, 2000 (cit. a p. 1).

[4] S. Han, J. Kim e S. H. Ko. «Advances in air filtration technologies: structure-based and interaction-based approaches». In: *Materials Today Advances* 9 (2021), p. 100134 (cit. a p. 2).

[5] M. S. Parvez, M. M. Rahman, M. Samykano e Mohammad Yeakub Ali. «Current advances in fabric-based airbag material selection, design and challenges for adoption in futuristic automobile applications». In: *Materials Today: Proceedings* 107 (2024), pp. 158–165 (cit. a p. 2).

[6] I. Di Domenico, S. Hoffmann e P. Collins. «The Role of Sports Clothing in Thermoregulation, Comfort, and Performance During Exercise in the Heat: A Narrative Review». In: *Sports Medicine - Open* 8.1 (dic. 2022) (cit. a p. 2).

[7] A. Puszkarz e I. Krucińska. «Modeling of Air Permeability of Knitted Fabric Using the Computational Fluid Dynamics». In: *Autex Research Journal* 18 (mag. 2018), pp. 365–376 (cit. a p. 2).

[8] X. Zeng, A. Endruweit, L. P. Brown e A. C. Long. «Numerical prediction of in-plane permeability for multilayer woven fabrics with manufacture-induced deformation». In: *Composites Part A: Applied Science and Manufacturing* 77 (2015), pp. 266–274 (cit. a p. 3).

[9] X. Xiao, A. Long, K. Qian, X. Zeng e T. Hua. «Through-thickness permeability of woven fabric under increasing air pressure: Theoretical framework and simulation». In: *Textile Research Journal* 87.13 (2017), pp. 1631–1642 (cit. alle pp. 3, 51).

[10] Z. Zupin, A. Hladnik e K. Dimitrovski. «Prediction of one-layer woven fabrics air permeability using porosity parameters». In: *Textile Research Journal* 82 (ott. 2011), pp. 117–128 (cit. alle pp. 3, 10, 13, 34, 36, 37, 50, 53).

[11] International Organization for Standardization. *ISO 9237:1995 – Textiles – Determination of permeability of fabrics to air*. Standard. Geneva, Switzerland, 1995 (cit. alle pp. 3, 34).

[12] Y. E. Elmogahzy. *Engineering Textiles (Second Edition)*. Woodhead Publishing, 2020. Cap. 10 (cit. alle pp. 5–7, 9).

[13] B. Kumar e J. Hu. «6 - Woven fabric structures and properties». In: *Engineering of High-Performance Textiles*. Woodhead Publishing, 2018, pp. 133–151 (cit. a p. 5).

[14] J. Hearle. «The structural mechanics of fibers». In: *Journal of Polymer Science Part C: Polymer Symposia* 20 (mar. 2007), pp. 215–251 (cit. a p. 6).

[15] *Textiles — Determination of thickness of textiles and textile products*. British Standards Institution (BSI), 1997 (cit. a p. 9).

[16] Y. E. Elmogahzy. *Engineering Textiles (Second Edition)*. Woodhead Publishing, 2020. Cap. 9 (cit. alle pp. 10–12).

[17] Y. E. Elmogahzy. *Engineering Textiles (Second Edition)*. Woodhead Publishing, 2020. Cap. 8 (cit. a p. 11).

[18] L. Guanzhi, Z. Qiang, W. Jun e R. Gong. «Modeling of Transverse Compression Behavior of Yarns». In: *Textile Research Journal* (2017), pp. 184–190 (cit. a p. 14).

[19] L.P. Brown e A.C. Long. «8 - Modeling the geometry of textile reinforcements for composites: TexGen». In: *Composite Reinforcements for Optimum Performance (Second Edition)*. A cura di Philippe Boisse. Second Edition. Woodhead Publishing, 2021, pp. 237–265 (cit. a p. 17).

[20] Louise P. Brown. «TexGen». Chapter manuscript for *Advanced Weaving Technology*, section 2.9. Composites Research Group, Faculty of Engineering, University of Nottingham (cit. alle pp. 18, 20, 21).

[21] F. Moukalled, L. Mangani e M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 2016 (cit. a p. 23).

[22] F. Moukalled, L. Mangani e M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 2016. Cap. 8 (cit. a p. 27).

[23] F. Moukalled, L. Mangani e M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 2016. Cap. 11 (cit. a p. 30).

[24] F. Moukalled, L. Mangani e M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 2016. Cap. 15 (cit. a p. 32).

[25] ATI Corporation / TEXTEST AG. *FX 3300-IV LabAir — Air Permeability Tester*. Accessed: 2025-08-12. 2025. URL: https://aticorporation.com/instruments/air-permeability-tester-model-fx-3300-iv (cit. a p. 35).

[26] E. Syerko, C. Binetruy, S. Comas-Cardona e A. Leygue. «A numerical approach to design dual-scale porosity composite reinforcements with enhanced permeability». In: *Materials & Design* 131 (2017), pp. 307–322 (cit. alle pp. 35, 38, 67).

[27] R. Gebart. «Permeability of Unidirectional Reinforcements for RTM». In: *Journal of Composite Materials* 26 (8 ago. 1992), pp. 1100–1133 (cit. a p. 35).

[28] M. El Messiry. «Theoretical determination of the fiber volume fraction distribution for natural fiber fabric reinforced polymer composite». In: *Journal of Industrial Textiles* 48 (nov. 2018) (cit. a p. 36).

[29] Chris Greenshields. *OpenFOAM v9 User Guide - 5.3 Mesh generation with the blockMesh utility*. Accessed: 2025-08-11. 2021. URL: https://doc.cfd.direct/openfoam/user-guide-v9/blockmesh#x26-1850005.3 (cit. alle pp. 39, 49).

[30] Chris Greenshields. *OpenFOAM v9 User Guide - 5.4 Mesh generation with the snappyHexMesh utility*. Accessed: 2025-08-11. 2021. URL: https://doc.cfd.direct/openfoam/user-guide-v9/snappyhexmesh#x27-1970005.4 (cit. a p. 41).

[31] Chris Greenshields. *OpenFOAM v9 User Guide - 5.2 Boundaries*. Accessed: 2025-08-15. 2021. URL: https://doc.cfd.direct/openfoam/user-guide-v9/boundaries#x25-1780005.2.1 (cit. a p. 49).

[32] The pandas development team. *pandas-dev/pandas: Pandas*. Ver. 2.3.0. 2025. DOI: 10.5281/zenodo.3509134 (cit. a p. 69).

[33] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. a p. 69).

[34] K. Potdar, T. Pardawala e C. Pai. «A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers». In: *International Journal of Computer Applications* 175 (2017), pp. 7–9 (cit. a p. 70).

[35] Sagar Imambi, Kolla Bhanu Prakash e G. R. Kanagachidambaresan. «PyTorch». In: *Programming with TensorFlow*. A cura di Kolla Bhanu Prakash e G. R. Kanagachidambaresan. EAI/Springer Innovations in Communication and Computing. Cham: Springer, 2021, pp. 87–104 (cit. a p. 70).