

# POLITECNICO DI TORINO

Collegio di Ingegneria Chimica e dei Materiali

Corso di Laurea Magistrale  
in Ingegneria Chimica e dei Processi Sostenibili



Tesi di Laurea Magistrale

## Modellazione termo - fluidodinamica di schiume solide: un approccio integrato di CFD e Machine Learning

Relatori

Prof. Daniele MARCHISIO

Prof. Gianluca BOCCARDO

Dott.ssa Agnese MARCATO

Candidato

Alessio BOCCA

Novembre 2023



*Ars longa, vita brevis.*

*Seneca*



# Indice

<b>Elenco delle figure</b>	VIII
<b>Elenco delle tabelle</b>	XI
<b>Acronimi</b>	XII
<b>Elenco dei simboli</b>	XIV
<b>I Introduzione</b>	<b>1</b>
<b>1 Stato dell'arte</b>	<b>7</b>
1.1 Geometria e caratterizzazione . . . . .	7
1.2 Stato dell'arte: OCF metalliche . . . . .	9
1.2.1 Classificazione basata su materiali e struttura . . . . .	9
1.2.2 Vie processistiche . . . . .	10
1.2.3 Applicazioni tecnologiche . . . . .	12
1.3 Stato dell'arte: OCF ceramiche . . . . .	13
1.3.1 Applicazioni tecnologiche . . . . .	14
<b>II Elementi Teorici e Metodi Computazionali</b>	<b>17</b>
<b>2 Elementi Teorici</b>	<b>21</b>
2.1 Geometrical model . . . . .	21
2.1.1 L'algoritmo . . . . .	21
2.1.2 Parametri geometrici . . . . .	23
2.2 Physical models . . . . .	24
2.2.1 Ipotesi adottate . . . . .	24
2.2.2 Equazioni di governo . . . . .	24
2.3 Data-driven models . . . . .	28
2.3.1 Modelli di regressione . . . . .	29

2.3.2	Modelli di classificazione . . . . .	34
<b>3</b>	<b>Metodi Computazionali</b>	<b>37</b>
3.1	The Finite Volume Method . . . . .	37
3.1.1	Equazioni semi-discretizzate . . . . .	39
3.1.2	Equazioni algebriche . . . . .	41
3.1.3	Condizioni di bordo . . . . .	42
3.1.4	Risoluzione del sistema di equazioni algebriche . . . . .	43
3.2	The Finite Volume Mesh . . . . .	44
3.3	Discretizzazione spaziale . . . . .	46
3.3.1	Termine diffusivo . . . . .	47
3.3.2	Termine convettivo . . . . .	48
3.4	Discretizzazione temporale . . . . .	51
3.5	OpenFoam tools . . . . .	52
3.5.1	Impostazione foamCases . . . . .	52
3.5.2	Meshing tools . . . . .	54
3.5.3	Solvers . . . . .	54
3.5.4	Postprocessing . . . . .	55
3.6	Cenni HPC . . . . .	56
<b>III</b>	<b>Impostazione Workflow e Dettagli Numerici</b>	<b>59</b>
<b>4</b>	<b>Simulazioni di flusso</b>	<b>63</b>
4.1	Setup delle simulazioni . . . . .	63
4.1.1	Grid Independency . . . . .	64
4.2	Analisi del REV . . . . .	66
4.2.1	Effetto di bordo . . . . .	69
4.2.2	Analisi CFD del REV . . . . .	69
4.3	Studio preliminare . . . . .	72
4.3.1	Geometrie a 3 5 DOF . . . . .	72
4.4	Dataset esteso . . . . .	75
4.4.1	Algoritmo di automazione . . . . .	75
4.4.2	Struttura del dataset . . . . .	77
4.4.3	Caratterizzazione delle geometrie . . . . .	79
4.5	Metodologia Machine Learning . . . . .	81
4.5.1	Classificatore SVM: pipeline . . . . .	81
4.5.2	Regressor FFNN: pipeline . . . . .	85

<b>5</b>	<b>Seconda fisica: trasporto di calore</b>	<b>87</b>
5.1	Setup delle simulazioni di scambio termico . . . . .	87
5.2	Simulazioni di interesse . . . . .	88
5.2.1	Esperimenti numerici . . . . .	88
<b>IV</b>	<b>Risultati e Conclusioni</b>	<b>91</b>
<b>6</b>	<b>Sistema complesso</b>	<b>93</b>
6.1	Fallimento dell'approccio classico . . . . .	94
6.2	Risultati dell'approccio machine learning . . . . .	96
6.3	Sistema non isoterma . . . . .	102
6.4	Conclusioni . . . . .	104
<b>A</b>	<b>Python scripts nell'analisi del REV</b>	<b>107</b>
A.1	Porespy script . . . . .	107
A.2	Concentric volume script . . . . .	109
<b>B</b>	<b>Algoritmo di automazione</b>	<b>111</b>
B.1	Input script . . . . .	111
B.2	Building script . . . . .	112
B.3	Launcher script . . . . .	118
B.4	Postprocessing script . . . . .	122
	<b>Bibliografia</b>	<b>127</b>

# Elenco delle figure

1	Progetto AIMFOAM. . . . .	6
1.1	Le caratteristiche geometriche delle OCF sono evidenziate a colori; in verde la superficie di una <i>strut</i> , in rosso due <i>windows</i> adiacenti alla <i>strut</i> , in giallo il volume delimitato da una cella [8]. . . . .	8
1.2	Processo INCO: schema a stadi [10]. . . . .	11
1.3	Processo Duocell: schema a stadi [10]. . . . .	12
1.4	Applicazioni per schiume metalliche e <i>cellular metallic materials</i> in funzione del grado di apertura [10]. . . . .	12
1.5	Simulazione di steam reforming del metano in reattori catalitici tubolari in condizioni operative tipiche dell'industria di processo [9].	15
2.1	Fasi dell'algoritmo di generazione delle geometrie basato sul software <b>Plugim</b> . . . . .	22
2.2	Il più semplice percettrone, la Threshold Logic Unit [28]. . . . .	32
2.3	Semplice rete MLP caratterizzata da due hidden layer oltre ai livelli di input ed output [28]. . . . .	33
2.4	Algoritmo SVM applicato ad un generico caso bidimensionale, ovvero dove ogni data-object è caratterizzato da due features [27]. . . . .	35
2.5	Effetto della regolarizzazione (parametro C) nell'algoritmo SVM di classificazione [27]. . . . .	35
3.1	Processo di discretizzazione nel FVM [31]. . . . .	38
3.2	Flusso attraverso le $nb(C)$ facce $f$ della cella C [30]. . . . .	40
3.3	Rappresentazione grafica del metodo di Gauss-Siedel [32]. . . . .	43
3.4	Topologia delle celle: owners, neighbors, faces [33]. . . . .	45
3.5	Tipi di elementi in mesh tridimensionali [33]. . . . .	45
3.6	Griglia cartesiana uniforme [34]. . . . .	46
3.7	Profilo di interpolazione alla faccia $e$ [34]. . . . .	47
3.8	Griglia monodimensionale nel problema advezione-diffusione [35]. . . . .	48
3.9	Termine convettivo: schema upwind [35]. . . . .	49
3.10	Struttura di una generica foamCase. . . . .	53

3.11	Tempo computazionale di meshing e solver per una generica geometria parallelizzata su più <i>cores</i> . . . . .	56
4.1	Bounding box simulata in <b>OpenFOAM</b> . Sono evidenziate le patch di inlet ed outlet del fluido [8]. . . . .	63
4.2	Porzione di mesh computazionale con $R_{Layer} = 2$ e $BM_{cells}/pore = 20$ , costruita su una generica geometria di open cell foam. . . . .	64
4.3	Risultati della grid independence analysis. . . . .	65
4.4	<i>Representative elementary volume</i> in un mezzo poroso [38]. . . . .	67
4.5	Identificazione del REV con la libreria python <i>porespy</i> . L'asse delle ascisse è discretizzato in venti classi di volume, i tratti rossi e gialli identificano rispettivamente il valor medio e la deviazione standard della porosità per ogni classe. . . . .	68
4.6	Identificazione del REV con lo script dei volumi concentrici. Aumentando la dimensione della box il profilo di porosità si stabilizza. Un'effetto di bordo è presente ai confini della bounding box. . . . .	68
4.7	Immagini <b>PlugIm</b> di tre slices in una geometria di OCF: una iniziale, una intermedia ed una finale. La prima e l'ultima presentano evidenti effetti di bordo. . . . .	69
4.8	Volumi progressivi simulati nell'analisi CFD del REV. . . . .	70
4.9	Campo di moto, geometria caratterizzata da $L_{REV} = 350$ . . . . .	71
4.10	Dataset casuale di trenta geometrie con valori variabili di $\alpha$ , $\beta$ , $REP$ ; quest'ultimo rappresentato dalla dimensione dei cerchi. Le geometrie arancioni presentano maggiore instabilità del REV per via di un impaccamento delle sfere meno compatto, dovuto a bassi valori di $\alpha$ , $\beta$ . . . . .	72
4.11	Summary table del dataset. . . . .	76
4.12	File dei risultati delle simulazioni di flusso. . . . .	78
4.13	Distribuzione della feature $\alpha$ nel dataset esteso. . . . .	79
4.14	Distribuzione delle features caratterizzanti le geometrie appartenenti al dataset esteso. Tutte i possibili valori delle features sono campionati; questo aspetto assumerà particolare importanza nel contesto del modello data-driven. . . . .	80
4.15	Matrice di correlazione (metodo di Pearson) applicata alla features ed alla class label. . . . .	83
4.16	Distribuzione delle class labels per il dataset di train/test del modello SVM e per il dataset esteso. Le geometrie etichettate come non fisiche saranno rimosse dal pool di training del modello di regressione. . . . .	83
5.1	Sezione y-normal delle due geometrie utilizzate nel primo esperimento numerico. . . . .	89

5.2	Sezione y-normal delle nove geometrie utilizzate nel secondo e nel terzo esperimento numerico. . . . .	90
6.1	Deterioramento delle classiche relazioni <i>power law</i> tra variabili chiave, all'aumentare della dimensionalità del problema. . . . .	94
6.2	Sono stati svolti esperimenti per diversi valori del parametro C. Ogni esperimento è caratterizzato da un valore di recall ed un valore di F1-Score. Le due metriche sono massimizzate in corrispondenza del range $C = 10, 11, \dots, 16$ . . . . .	96
6.3	Sono stati svolti esperimenti per architetture della rete neurale. Ogni esperimento è caratterizzato da un valore medio di $R^2$ e dalla sua deviazione standard $\sigma$ . Maggiore è il valore di $R^2$ e migliore è l'accuratezza del modello. . . . .	97
6.4	Focus della Fig.(6.3). E' possibile osservare come un minimo numero di neuroni sono sufficienti per superare le prestazioni del modello di regressione lineare multivariabile, Eq.(6.4). . . . .	98
6.5	<i>Parity plot</i> della permeabilità per le geometrie OCF risolte nel dataset esteso. Confronto tra valori CFD e valori predetti dal modello data-driven, approccio LOO. L'errore medio è del 3.05%. . . . .	99
6.6	Slice bidimensionale della mappa di permeabilità ottenuta dal modello data-driven di regressione. I valori dei parametri sono normalizzati con <i>MinMax-Scaler</i> nell'intervallo (0,1). . . . .	100
6.7	Distribuzione della permeabilità $\kappa$ , calcolata risolvendo le simulazioni fluidodinamiche sulle istanze del dataset esteso. . . . .	100
6.8	Distribuzione della porosità $\epsilon$ delle istanze appartenenti al dataset esteso. Il valore di $\epsilon$ è stato calcolato utilizzando le informazioni della mesh computazionale. . . . .	101
6.9	Distribuzione della superficie specifica $S_v$ delle istanze appartenenti al dataset esteso. Il valore di $S_v$ è stato calcolato utilizzando le informazioni della mesh computazionale. . . . .	101
6.10	Evoluzione transitoria del profilo di temperatura nella fase fluida per una generica geometria OCF in una simulazione di scambio termico. . . . .	102
6.11	Perdite di carico in funzione di $Re$ per le geometrie A,B. . . . .	103
6.12	$Nu$ in funzione di $Re$ per le geometrie A,B. . . . .	103

# Elenco delle tabelle

1.1	Dimensione delle celle e densità relativa per differenti vie processistiche nella produzione di schiume metalliche [10]. . . . .	10
1.2	Metanazione dell'anidride carbonica, confronto cinetiche [9]. . . . .	14
2.1	Parametri in ingresso all'algoritmo di generazione delle geometrie. . . . .	23
4.1	Dettagli numerici su geometrie e mesh. . . . .	71
4.2	Risultati delle simulazioni su porzioni crescenti di schiuma. . . . .	71
4.3	Dimensione media delle celle negli esperimenti DS-3DOS e DS-5DOS. . . . .	74
4.4	Parametri in ingresso all'algoritmo di automazione. . . . .	75
4.5	Dataframe delle features $\mathcal{F}$ , per il dataset di train-test del modello di classificazione SVM. . . . .	82
4.6	Dataframe delle class labels $\mathcal{T}$ , per il dataset di train-test del modello di classificazione SVM. . . . .	82
4.7	Dataframe delle features $\mathcal{F}^*$ , per il dataset di train-test del modello di regressione MLP. . . . .	86
4.8	Dataframe dei numerical targets $\mathcal{T}^*$ , per il dataset di train-test del modello di regressione MLP. . . . .	86
5.1	Caratterizzazione delle due geometrie impiegate per il primo esperimento numerico. . . . .	88
5.2	Caratterizzazione delle nove geometrie impiegate per il secondo ed il terzo esperimento numerico. . . . .	89
6.1	Architetture FFNN indagate. . . . .	97

# Acronimi

**BFGS** Broyden–Fletcher–Goldfarb–Shanno Algorithm

**CFD** Computational Fluid Dynamics

**CSV** Comma Separated Values

**CVD** Chemical Vapor Deposition

**DNN** Deep Neural Network

**DOC** Diesel Oxidation Catalyst

**DT** Decision Tree

**EEA** European Environment Agency

**EV** Electric Vehicles

**FFNN** Feed Forward Neural Networks

**FN** False Negative

**FP** False Positive

**FVM** Finite Volume Method

**HPC** High Performance Computing

**HX** Heat Exchanger

**IEA** International Energy Agency

**LASSO** Least Absolute Shrinkage and Selection Operator

**LBFGS** Limited-memory BFGS Algorithm

**LES** Large Eddy Simulation

**LOO** Leave One Out  
**LSM** Least Squares Method  
**MAE** Mean Absolute Error  
**MHS** Metallic Hollow Spheres  
**ML** Machine Learning  
**MLP** Multi Layers Perceptrons  
**MLPL** Machine Learning Pipeline  
**MSE** Mean Squared Error  
**NN** Rete neurale  
**OCF** Open Cell Foam  
**OMS** Organizzazione Mondiale della Sanità  
**POC** Particulate Oxidation Catalyst  
**PPI** Pore Per Inch  
**PU** Poliuretano  
**RAM** Random Access Memory  
**RANS** Reynolds Averaged Navier-Stokes equations  
**RELU** Rectified Linear Unit  
**REV** Representative Elementary Volume  
**RF** Random Forest  
**RSS** Residual Sum of Squares  
**SCR** Selective Catalytic Reduction  
**SEM** Scanning Electron Microscope  
**SIMPLE** Semi-Implicit Method for Pressure Linked Equations  
**SMR** Steam Methane Reforming  
**SVM** Support Vector Machine

**TN** True Negative

**TP** True Positive

**VOC** Volatile Organic Compounds

# Elenco dei simboli

$\alpha$	Primo fattore di compattezza	(-)
$\alpha_{fluid}$	Diffusività termica del fluido	(m <sup>2</sup> s <sup>-1</sup> )
$\beta$	Secondo fattore di compattezza	(-)
$\beta_i$	Coefficienti del modello di regressione	(-)
$\Delta T$	Forza spingente, scambio termico	(K)
$\Delta t$	Passo di discretizzazione temporale	(s)
$\delta x$	Distanza tra due centroidi di due celle distinte della mesh	(m)
$\Delta y$	Dimensione lineare di una generica cella C della mesh	(m)
$\delta_{Eucl}$	Distanza euclidea tra due punti nello spazio	(-)
$\delta_{geod}$	Distanza effettiva tra due punti nello spazio	(-)
$\dot{m}_f$	Portata massica attraverso una generica faccia $f$	(kg s <sup>-1</sup> )
$\dot{Q}$	Portata termica	(W)
$\dot{q}$	Portata volumica	(m <sup>3</sup> s <sup>-1</sup> )
$\epsilon$	Porosità	(-)
$\Gamma^\phi$	Generico coefficiente di trasporto dello scalare	(m <sup>2</sup> s <sup>-1</sup> )
$\hat{y}$	Variabile target stimata dal modello di regressione	(-)
$\kappa$	Permeabilità	(m <sup>2</sup> )
$\lambda$	Conducibilità termica del fluido	(W m <sup>-1</sup> K <sup>-1</sup> )
$\bar{\mathbf{x}}$	Media aritmetica dei valori della feature $\mathbf{x}$	(-)

$\bar{\mathbf{y}}$	Media aritmetica dei valori della feature $\mathbf{y}$	(-)
$\phi$	Vettore delle incognite (campo di $\phi$ )	(?)
$\mathbf{A}$	Matrice dei coefficienti	(-)
$\mathbf{B}$	Matrice di iterazione	
$\mathbf{b}$	Vettore dei bias, nel modello MLP	(-)
$\mathbf{b}$	Vettore dei termini noti, nella trattazione FVM	(?)
$\mathbf{D}$	Matrice diagonale	
$\mathbf{J}^{\phi,C}$	Flusso convettivo dello scalare $\phi$	(?/m <sup>2</sup> )
$\mathbf{J}_f^{\phi,C}$	Flusso convettivo dello scalare $\phi$ , sulla faccia $f$	(?/m <sup>2</sup> )
$\mathbf{J}^{\phi,D}$	Flusso diffusivo dello scalare $\phi$	(?/m <sup>2</sup> )
$\mathbf{J}_f^{\phi,D}$	Flusso diffusivo dello scalare $\phi$ , sulla faccia $f$	(?/m <sup>2</sup> )
$\mathbf{J}^{\phi}$	Flusso totale dello scalare $\phi$	(?/m <sup>2</sup> )
$\mathbf{J}_f^{\phi}$	Flusso totale dello scalare $\phi$ , sulla faccia $f$	(?/m <sup>2</sup> )
$\mathbf{L}$	Matrice triangolare inferiore	
$\mathbf{P}$	Matrice di preconditionamento	
$\mathbf{S}_f$	Vettore superficie della faccia $f$ appartenente alla cella $C$	(m <sup>2</sup> )
$\mathbf{U}$	Velocità istantanea, vettoriale	(m s <sup>-1</sup> )
$\mathbf{W}$	Matrice dei pesi, nel modello MLP	(-)
$\mathbf{w}$	Vettore dei pesi, nel modello MLP	(-)
$\mathbf{x}'$	Generica feature $\mathbf{x}$ scalata con un generico <i>scaler</i>	(-)
$\mathbf{x}$	Vettore delle features, nel modello MLP	(-)
$\mathbf{z}$	Vettore degli output di un layer di una rete MLP	(-)
$\mathcal{F}$	Dataframe delle features nel modello SVM	(-)
$\mathcal{F}^*$	Dataframe delle features nel modello di regressione	(-)
$\mathcal{M}_{border}$	Magnitudo dell'effetto di bordo	(voxel)

$\mathcal{N}$	Cardinalità del dataset da generare con <code>autoWorkflow</code>	(#)
$\mathcal{P}$	Pressione cinematica	( $\text{m}^2 \text{s}^{-2}$ )
$\mathcal{T}$	Dataframe delle labels nel modello SVM	( $\text{m}^2$ )
$\mathcal{T}^*$	Dataframe delle labels nel modello di regressione	( $\text{m}^2$ )
$\nu$	Viscosità cinematica	( $\text{m}^2 \text{s}^{-1}$ )
$\phi$	Generico scalare, nella trattazione FVM	(?)
$\phi_b$	Valore di $\phi$ su una faccia di bordo	(?)
$\phi_C$	Valore di $\phi$ nel centroide della cella C	(?)
$\phi_E$	Valore di $\phi$ nel centroide della cella E	(?)
$\phi_F$	Valore di $\phi$ nel centroide della generica cella F, adiacente a C	(?)
$\rho$	Densità del fluido	( $\text{kg m}^{-3}$ )
$\sigma_X$	Deviazione standard della feature X	(-)
$\sigma_Y$	Deviazione standard della feature Y	(-)
$\tau$	Tortuosità	(-)
$\varrho$	Raggio spettrale	
$a_C$	Coefficiente algebrico di linearizzazione del flusso	(-)
$a_F$	Coefficiente algebrico di linearizzazione del flusso	(-)
$a_W$	Coefficiente algebrico di linearizzazione del flusso	(-)
$b_C$	Coefficiente algebrico di linearizzazione del flusso	(-)
$c_p$	Calore specifico massico	( $\text{kJ kg}^{-1} \text{K}^{-1}$ )
$c_{p,in}$	Calore specifico massico, alla temperatura media di inlet	( $\text{kJ kg}^{-1} \text{K}^{-1}$ )
$c_{p,out}$	Calore specifico massico, alla temperatura media di outlet	( $\text{kJ kg}^{-1} \text{K}^{-1}$ )
$D_h$	Diametro idraulico	(m)
$F_1$	Metrica del classificatore SVM: $F_1$ score	(-)
$h$	Coefficiente di scambio termico	( $\text{W m}^{-2} \text{K}^{-1}$ )

$L$	Lunghezza della bounding box	(m)
$L_{REV}$	Dimensione lineare della bounding box del REV	(voxel)
$N_{sph}^*$	Numero di sfere effettivamente impaccate nella bounding box	(#)
$N_{sph}$	Numero di sfere da impaccare	(#)
$Nu$	Numero di Nusselt	(-)
$P$	Pressione del fluido	(Pa)
$Pe_{cella}$	Numero di Peclet di cella	(-)
$Pr$	Numero di Prandtl	(-)
$prc$	Metrica del classificatore SVM: precision	(-)
$Q^\phi$	Termine sorgente della generica proprietà $\phi$	( $?$ /m <sup>3</sup> )
$q_b$	Flusso di $\phi$ su una faccia di bordo	( $?$ /m <sup>2</sup> )
$R$	Raggio delle sfere impaccate	(voxel)
$r_i$	Residui nel modello di regressione	(-)
$R_{layer}$	Numero di refinement layers nella mesh computazionale	(#)
$R_{node}$	Dimensione dei nodi	(voxel)
$R_{strut}$	Dimensione delle struts	(voxel)
$r_{xy}$	Coefficiente di correlazione di Pearson	(-)
$Re$	Numero di Reynolds	(-)
$rec$	Metrica del classificatore SVM: recall	(-)
$REP$	Parametro di repulsione	(voxel)
$S_v$	Superficie specifica	(m <sup>2</sup> m <sup>-3</sup> )
$s_x$	Deviazione standard campionaria della generica feature $\mathbf{x}$	(-)
$S_{foam}$	Superficie della schiuma	(m <sup>2</sup> )
$T$	Temperatura	(K)
$t$	Tempo	(s)

$T_{fluid}$	Temperatura media del fluido	(K)
$T_{foam}$	Temperatura sulla superficie della matrice solida	(K)
$T_{in}$	Temperatura media di inlet	(K)
$T_{out}$	Temperatura media di outlet	(K)
$tv$	Magnitudo dell'operazione morfologica	(voxel)
$U_x$	Componente scalare della velocità istantanea lungo l'asse $x$	(m s <sup>-1</sup> )
$U_y$	Componente scalare della velocità istantanea lungo l'asse $y$	(m s <sup>-1</sup> )
$U_z$	Componente scalare della velocità istantanea lungo l'asse $z$	(m s <sup>-1</sup> )
$U_{avg}$	Velocità media in-pore, nella direzione del gradiente di pressione	(m s <sup>-1</sup> )
$u_s$	Velocità superficiale	(m s <sup>-1</sup> )
$V_C$	Volume di una generica cella C della mesh	(m <sup>3</sup> )
$V_{bb}$	Volume della bounding box	(m <sup>3</sup> )
$V_{tot}$	Volume totale	(m <sup>3</sup> )
$V_{void}$	Volume di vuoto	(m <sup>3</sup> )
$W$	Dimensione lineare della bounding box	(-)
$w_i$	Pesi dei collegamenti della rete neurale	(-)
$x_i$	Features del modello di regressione	(-)
$y$	Variabile target nel modello di regressione	(-)
$z$	Output di un neurone	(-)
$\Delta H_r^\circ$	Entalpia standard di reazione	(kJ mol <sup>-1</sup> )
$BM_{cells}$	Numero di celle per lato della bounding box, in <code>blockMesh</code>	(#)
$d_c$	Diametro medio delle <i>foam cell</i>	(m)
$f_{scale}$	Fattore di scaling delle geometrie	(-)
$NFC_{edge}$	Numero di <i>foam cells</i> per lato	(#)

**Parte I**

**Introduzione**



---

Negli ultimi decenni lo sviluppo dei paesi emergenti, il desiderio sempre maggiore di benessere ed avanguardia delle grandi egemonie e l'irrefrenabile consumismo che contraddistingue l'epoca recente hanno alimentato ininterrottamente il conto, in termini energetici, della nostra società. Il più recente report della IEA (International Energy Agency) stima un consumo energetico globale, nel solo anno 2019, pari a 418 EJ, corrispondente a circa  $1.16 \times 10^{11}$  MWh. Il dato risulta in crescita del 31% rispetto all'anno 1999 [1]. Lo sviluppo industriale ed il conseguente fabbisogno di materie prime, acqua, potenza elettrica e potenza termica hanno incrementato l'intensità con la quale le attività antropiche impattano sugli equilibri naturali. Lo stesso report della IEA evidenzia come le emissioni dirette di anidride carbonica dovute al consumo di combustibili siano aumentate da 22.2 a 33.6 Gt dal 1999 al 2019 [2]. I processi produttivi industriali sono inoltre co-responsabili dell'immissione in atmosfera di significative quantità dei cosiddetti gas-clima-alteranti o *greenhouse gases*; tra questi figurano, oltre la  $CO_2$ , il metano ( $CH_4$ ), gli F-gases (Fluorinated gases), gli idrofluorocarburi o gas freon ( $HFCs$ ), i perfluorocarburi ( $PFCs$ ), l'esaffluoruro di zolfo ( $SF_6$ ) ed il trifluoruro di azoto ( $NF_3$ ). Per questi gas è stimata un'emissione complessiva di 59 Gt di  $CO_{2eq}$  [3]. Un recente report sulla qualità dell'aria nel continente europeo, prodotto dalla EEA (European Environment Agency), evidenzia come nel 2021 il 96% della popolazione urbana è stata esposta a livelli di particolato molto al di sopra dei valori indicati nelle linee guida sanitarie stabilite dall'OMS (Organizzazione Mondiale della Sanità). Lo stesso documento stima in 238000 le morti premature dovute alle elevate concentrazioni di polveri sottili ed agenti inquinanti dispersi in atmosfera nel solo anno 2020, elevando l'inquinamento atmosferico, specialmente nelle estese aree urbane, a grave fattore di rischio per la salute pubblica [4].

L'ultimo decennio trascorso è stato positivamente caratterizzato da una crescente consapevolezza sull'importanza di concetti quali le economie circolari, l'efficientamento energetico, l'impatto in termini di *carbon footprint* delle attività sia personali che aziendali e della responsabilità che tali soggetti hanno rispetto l'intera società. In questo contesto, nel 2015, l'assemblea generale delle Nazioni Unite definisce i *Sustainable Development Goals*, una lista di 17 punti cardine per tracciare le nuove linee guida dello sviluppo globale.

Recentemente, sembra ormai accreditata l'ipotesi che non esista una tecnologia *perfetta* in grado di affrontare e risolvere le grandi tematiche di interesse globale, dipinte nel precedente paragrafo. E' tuttavia nota la necessità di dover investire in un paniere di tecnologie, la cui integrazione può realmente fare la differenza nelle grandi sfide del XXI secolo. Nel ventaglio delle tecnologie innovative, il presente lavoro intende dedicarsi allo studio di una di esse: le schiume a celle aperte, OCF. E' evidente come questa tecnologia possa mettere in campo un'importante contributo verso le sfide precedentemente discusse, a partire dal dramma dell'inquinamento

---

atmosferico che richiede una rapida risposta fondata su tecnologie capaci di abbattere prodotti di combustione e particolato atmosferico. Segue la necessità di sviluppare tecnologie per la cattura o la neutralizzazione dei gas clima-impattanti mediante adsorbimento o abbattimento catalitico. Infine la crescita del fabbisogno energetico globale deve necessariamente essere accompagnata da politiche incentrate sull'ottimizzazione energetica dei processi produttivi, l'utilizzo consapevole dell'energia elettrica e termica e la messa in campo di tecnologie per l'accelerazione della transizione ecologico-energetica; principale *milestone* della Comunità Internazionale nel medio lungo termine. Le tecnologie basate sulle OCF hanno dimostrato e dimostrano tuttora elevata efficacia e reale potenziale economico per essere pienamente considerate uno strumento opportuno per affrontare queste sfide.

Nell'ambito dell'ingegneria chimica, i mezzi porosi rivestono un ruolo chiave in molte applicazioni per via delle loro peculiari proprietà come l'elevata superficie specifica e l'elevato grado di vuoto, che rendono queste strutture particolarmente interessanti dal punto di vista di molti processi industriali: dal trattamento di effluenti gassosi, all'abbattimento di inquinanti, al ruolo chiave di supporto per catalizzatori in processi chimici, ad applicazioni di scambio termico e come componenti chiave in batterie e fuel cells. Le schiume solide a celle aperte appartengono alla classe dei *cellular materials* e si contraddistinguono dalle schiume a celle chiuse per via dell'esteso *network* di pori interconnessi, che ne caratterizza la loro unicità. Presentano elevati gradi di vuoto, nel range 75% - 95%, e sono disponibili in materiali ceramici, metalli o polimerici in base alla specifica applicazione.

Le schiume ceramiche trovano impiego in ambienti estremi, ad elevate temperature, riuscendo a garantire eccellenti proprietà strutturali. Le schiume metalliche risultano interessanti per applicazioni di scambio termico, per via dell'elevata conducibilità termica e della vasta area superficiale. Le schiume metalliche presentano una minore conducibilità elettrica rispetto al materiale originale ma il loro impiego, come elettrodi per batterie, è in crescita per via della elevata area superficiale [5, 6].

Alantum è una azienda globale operante in Corea, Cina, Germania e Stati Uniti, specializzata nella produzione e nella commercializzazione di OCF [7]. I loro prodotti incontrano le esigenze delle industrie e le loro applicazioni possono essere organizzate in tre vaste aree:

- (i) Tecnologie per il trattamento dell'aria
  - Diesel Oxidation Catalyst (DOC)
  - Particulate Oxidation Catalyst (POC)
  - Selective Catalytic Reduction (SCR)

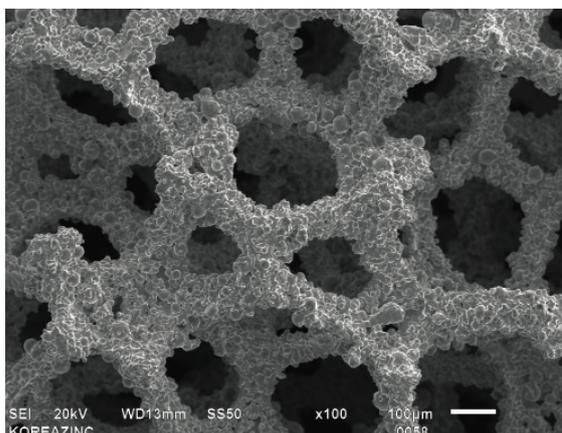
---

(ii) Tecnologie per i processi chimici

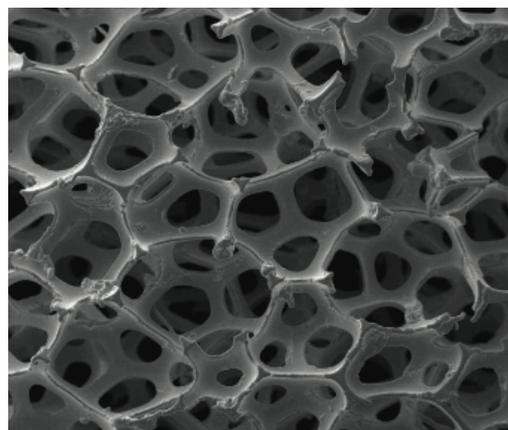
- Steam Methane Reforming (SMR)
- Methanation
- Dehydrogenation
- Ethylene Oxide

(iii) Prodotti speciali

- Industrial gas filters (SCR and VOC system)
- Gas and liquid mixers
- Fuel cell components
- Battery components



(a) Micrografia SEM di una schiuma *washcoated* per applicazioni DOC [7].



(b) Micrografia SEM di una schiuma Alantum a celle aperte, in *Ni* puro [7].

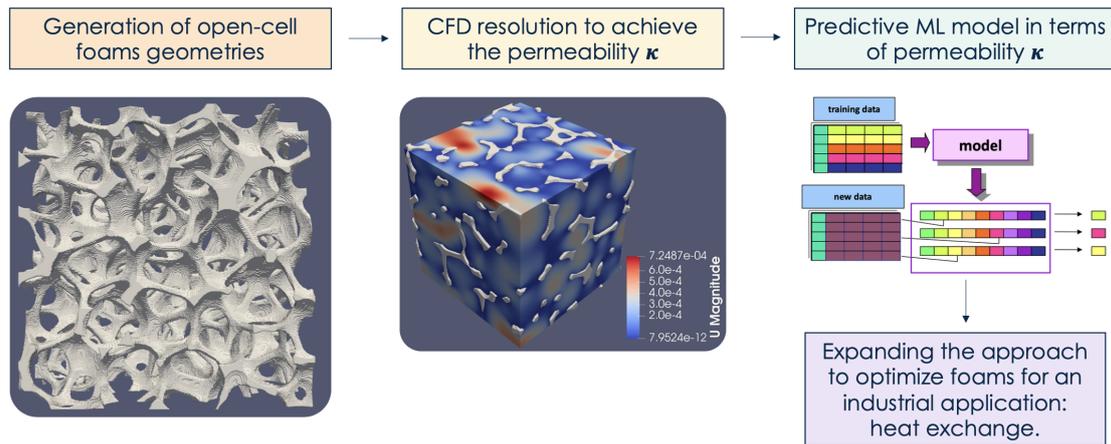
Partendo da questi presupposti, nasce qui la volontà di approfondire lo studio e la modellazione di questi mezzi porosi in un lavoro di tesi focalizzato sull'ottimizzazione della microstruttura di OCF mediante analisi fluidodinamiche e sfruttando le innovative opportunità offerte dagli algoritmi di ML (Machine Learning).

Il presente lavoro è collocato in un vasto progetto che vede come *mission* l'ottimizzazione delle geometrie di OCF sfruttando modellazioni fluidodinamiche e tecniche di machine learning. Storicamente, il ramo dell'ingegneria chimica non risulta essere particolarmente avvezzo all'utilizzo di algoritmi avanzati di machine learning (al di fuori dell'analisi statistica dei dati per il controllo di processo, dove vi è solida esperienza) a causa dell'esiguo numero di esperimenti

condotti ed il ristretto *pool* di dati disponibili. I cosiddetti *big data* (i.e. largo pool di informazioni, ampio dataset) risultano infatti una condizione imprescindibile per la proficua applicazione di modelli data-driven predittivi, tuttavia l'avvento della fluidodinamica computazionale ha parzialmente sopperito a tale requisito, fornendo la possibilità di disporre di ingenti quantità di dati; basti pensare alle sole informazioni del campo di moto per una mesh composta da milioni di celle.

L'impiego di algoritmi di machine learning predittivi è giustificato dalle difficoltà riscontrate nel definire relazioni funzionali tra le principali variabili indagate [8], e.g. pressione, velocità o grandezze adimensionali come  $Re$ ,  $Pr$ ,  $Nu$ . In questo lavoro saranno perseguiti diversi obiettivi nell'ambito della modellazione fluidodinamica della microstruttura di geometrie OCF, con particolare riguardo all'utilizzo di *neural networks* per l'individuazione di relazioni funzionali secondo un'approccio di tipo data-driven. A seguire, le *milestones* raggiunte nel presente lavoro.

- (i) Indagine CFD del campo di moto in geometrie OCF costruite con un modello geometrico basato sul software PlugIm.
- (ii) Sviluppo di un workflow automatico per la generazione e la risoluzione CFD di un vasto numero di geometrie (i.e. costruzione del dataset).
- (iii) Training di un modello data-driven per estendere l'analisi del campo di moto ad una vasta gamma di geometrie senza ricorrere ulteriormente dispendiose analisi CFD.
- (iv) Indagine CFD di una secondo problema fisico: un semplice modello di scambio termico.



**Figura 1:** Progetto AIMFOAM.

# Capitolo 1

## Stato dell'arte

### 1.1 Geometria e caratterizzazione

Le Open Cell Foams appartengono alla classe dei mezzi porosi e sono oggetto di studio dalle ultime decadi del XX secolo. In forma macroscopica si presentano sotto forma di pellets di dimensione variabile, solitamente dell'ordine dell'unità o di alcune decine di millimetri. Attualmente i processi di manifattura sono in grado di sviluppare schiume solide a base metallica, a base ceramica o di materiale polimerico; quest'ultime usate prevalentemente come stampo tridimensionale per la produzione delle prime. La microstruttura delle OCF è caratterizzata da tre elementi principali [9]:

- *Struts*: Sono le asticelle che connettono la struttura della schiuma, determinano il volume delle celle e possono assumere forme più o meno affusolate in funzione del particolare processo produttivo impiegato per la realizzazione.
- *Windows*: Sono le aperture che mettono in comunicazione celle adiacenti e sono contornate dalla superficie delle struts.
- *Cells*: Rappresentano il volume racchiuso nei vuoti della matrice solida.

Le capacità della geometria di opporre resistenza al passaggio del fluido, di interfacciarsi con esso, di promuovere il trasferimento di massa e di calore sono misurate utilizzando alcuni macrodescrittori.

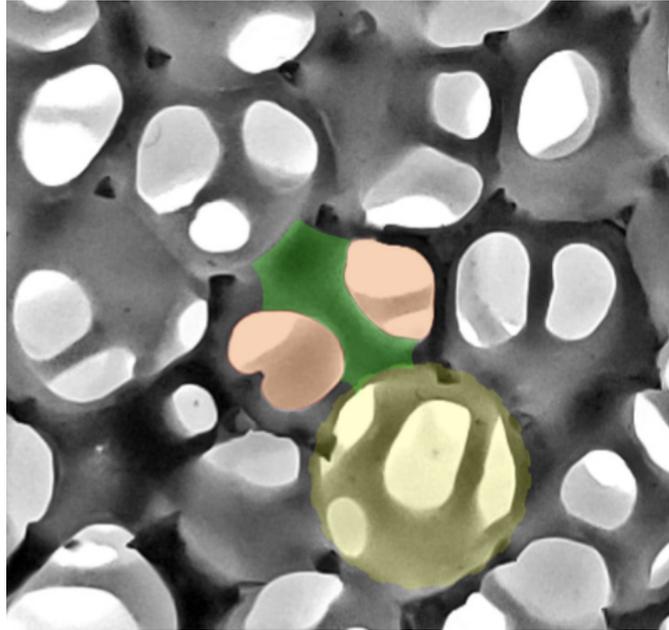
- (i) *Porosità*: Solitamente è indicata con la lettera greca  $\epsilon$ , rappresenta il grado di vuoto della geometria. La porosità può essere misurata a diverse scale di grandezza, dalla nanoscala delle particelle catalitiche fino alla scala millimetrica se si fa riferimento a pellets macroscopici.

$$\epsilon = \frac{V_{void}}{V_{tot}} \quad (1.1)$$

- (ii) *Superficie specifica*: Indicata come  $S_v$  questa proprietà misura il rapporto tra la superficie della matrice solida ed il volume del campione considerato. Maggiore è la superficie specifica, maggiore è la superficie disponibile per i fenomeni di trasporto, a parità di volume.

$$S_v = \frac{S_{foam}}{V_{tot}} \quad (1.2)$$

- (iii) *Permeabilità*: Indicata con la lettera greca  $\kappa$ , questa proprietà misura la capacità intrinseca di un mezzo poroso di opporre o meno resistenza al passaggio di un fluido nelle sue cavità. Tale concetto verrà dettagliato meglio nella sezione 2.2; in particolare la permeabilità sarà definita in Eq.(2.12).
- (iv) *Tortuosità*: Indicata con la lettera greca  $\tau$ , è definita come  $\tau = \delta_{geod}/\delta_{Eucl}$ . Dati due punti A, B nello spazio, la tortuosità misura il rapporto tra la distanza effettivamente percorsa dal fluido per muoversi da A e raggiungere B ( $\delta_{geod}$ ) e la distanza euclidea misurata tra i punti ( $\delta_{Eucl}$ ) [8].



**Figura 1.1:** Le caratteristiche geometriche delle OCF sono evidenziate a colori; in verde la superficie di una *strut*, in rosso due *windows* adiacenti alla strut, in giallo il volume delimitato da una cella [8].

## 1.2 Stato dell’arte: OCF metalliche

Le schiume solide metalliche sono ottenute a partire da un metallo fuso o una lega, lavorato secondo ben definite vie processistiche. L’alluminio è il materiale d’eccellenza ma, nella produzione, ne sono impiegati anche molti altri come rame, nickel, titanio e varie leghe. Le schiume metalliche sono contraddistinte da elevati livelli di porosità e strutture leggere, inoltre possiedono proprietà fisiche peculiari derivanti dai materiali impiegati nel processo produttivo [10].

### 1.2.1 Classificazione basata su materiali e struttura

Le schiume metalliche sono attualmente prodotte utilizzando metalli e leghe allo scopo di raggiungere le prestazioni richieste dalle specifiche applicazioni. Le OCF più comunemente impiegate sono:

- (i) Alluminio: La produzione di schiume a base di alluminio è fondata sull’uso di agenti espandenti come il carbonato di calcio ( $CaCO_3$ ), il carbonato di magnesio ( $MgCO_3$ ) o alcuni idruri metallici come  $ZrH_2$  e  $TiH_2$ . Durante il processo produttivo si raggiungono elevate temperature allo scopo di fondere il metallo mentre gli agenti espandenti, ad alta temperatura, si decompongono liberando gas responsabile della formazione dei pori nella struttura finale [5, 11].
- (ii) Rame e nickel: La produzione di schiume a base di rame e nickel è sostenuta dalla società Alantum mediante un processo che vede l’impiego di una soluzione legante organica che permette l’applicazione di polveri metalliche (o in lega) sulla superficie di schiume solide poliuretaniche (PU). Il principale impiego di questo prodotto è la produzione di elettrodi per batterie di EV [11, 12].
- (iii) Ferro: Le schiume a base di ferro presentano caratteristiche differenti dalle medesime realizzate in alluminio; esse manifestano una maggiore resistenza meccanica, una maggiore capacità di assorbimento dell’energia, costi inferiori e migliore saldabilità. In questo caso, per via delle elevate temperature di fusione del ferro, i convenzionali agenti espandenti sono inefficaci. La lavorazione delle schiume a base ferro prevede l’insufflaggio di  $CO_2$  come agente espandente non convenzionale. Possono essere raggiunte porosità fino al 57% ad una temperatura di 1350°C.
- (iv) Metallic hollow spheres (MHS): Sono schiume metalliche caratterizzate da una distribuzione omogenea di celle a forma sferica. Sono principalmente utilizzate nella progettazione di scambiatori di calore ed in applicazioni di *thermal energy storage*.

E' possibile inoltre classificare le OCF metalliche in base alla via processistica adottata per ottenerne la caratteristica micro-struttura. Le tecniche più diffuse includono l'uso di:

- (i) Agenti espandenti: In questa classe di composti si trovano gli idruri metallici, i carbonati ed alcuni ossidi. La peculiare caratteristica che li accomuna è la capacità di rilasciare gas quando vengono sottoposti a riscaldamento. Un'esempio dei più comuni agenti espandenti comprende  $TiH_2$ ,  $MgCO_3$ ,  $CaCO_3$ ,  $ZrH_2$  [5, 13].
- (ii) Spaziatori: Talvolta sono utilizzate strutture inerti (ceramiche o polimeriche) per definire la microstruttura delle schiume solide. Il metallo fuso viene iniettato nel *filler* ed il risultato finale è una geometria caratterizzata da un network omogeneo di celle e pori. La struttura solida inerte viene rimossa al termine delle operazioni [14].
- (iii) Modello di stampo: Al fine di ottenere dettagliate microstrutture il seguente approccio è adottato nella produzione di OCF. Lo stampo è inizialmente costruito in materiale differente, successivamente è iniettato il metallo fuso ed infine, una volta solidificato, lo stampo è rimosso [15].

## 1.2.2 Vie processistiche

La produzione di OCF puo' avvenire per molteplici vie processistiche

**Tabella 1.1:** Dimensione delle celle e densità relativa per differenti vie processistiche nella produzione di schiume metalliche [10].

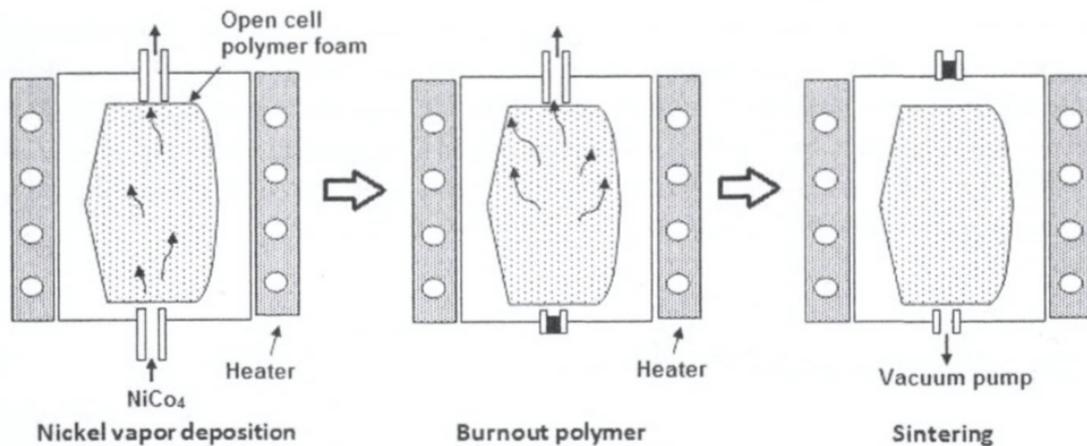
Dimensione celle (mm)	Densità relativa (-)	Vie processistiche (-)
0.01 - 0.1	0.01 - 0.1	Celle chiuse   HSC, PDISS, GME, EGE, PDM
0.1 - 1.00	0.1 - 1.00	Celle aperte   VEDCP, Parzialmente aperte   MGI, PDM
1 - 10	0.1 - 1.00	Celle chiuse   MGI

**dove**

*HSC*: hollow sphere consolidation, *PDISS*: particle decomposition in semi-solid, *GME*: gas-metal eutectic, *EGE*: entrapped gas expansion, *PDM*: particle decomposition in melt, *VEDCP*: vapor of electro-deposition on open-cell polymer, *MGI*: melt gas injection.

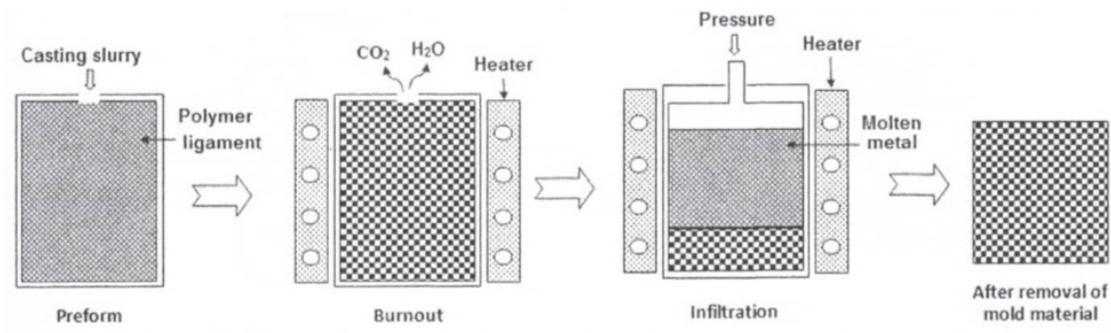
ed i possibili prodotti spaziano dalle schiume a celle aperte (oggetto di studio per questo lavoro) alle schiume a celle semi-aperte o chiuse. Caratteristiche e metodi di produzione sono elencati in Tab.(1.1). I processi di riferimento per la produzione di OCF metalliche sono *INCO* e *Duocell* ed entrambi sono basati sull'impiego del metallo fuso e di una struttura in altro materiale che funge da *template* per la schiuma metallica.

- (i) Processo *INCO*: Diversi metalli vengono depositati sulla superficie di una struttura polimerica con la funzione di stampo tridimensionale. Il processo è di tipo CVD (Chemical Vapor Deposition) e solitamente viene utilizzato il nickel tetracarbonile per rilasciare il metallo sulla superficie dello stampo. Per temperature superiori a  $100\text{ }^{\circ}\text{C}$ , il  $\text{Ni}(\text{CO})_4$  si decompone in monossido di carbonio e nickel e quest'ultimo ricopre completamente la superficie del polimero. La principale limitazione al processo deriva dalla tossicità del  $\text{Ni}(\text{CO})_4$  che è sottoposto a stringenti regolamentazioni o talvolta, in alcune giurisdizioni, completamente messo al bando. Il vantaggio dei prodotti risiede in una scarsa resistenza al passaggio di corrente se confrontati con prodotti simili ottenuti da differenti vie processistiche. La dimensione dei pori varia da  $100$  a  $300\text{ }\mu\text{m}$  e la densità relativa può raggiungere un minimo di  $0.02 - 0.05$  [10].



**Figura 1.2:** Processo INCO: schema a stadi [10].

- (ii) Processo *Duocell*: Inizialmente è costruita una struttura *template* in materiale polimerico o in cera, successivamente la struttura è ricoperta di materiale ceramico in modo da ottenere un negativo della schiuma solida. Infine, rimosso il *template*, il negativo è lentamente riempito con il metallo fuso o la lega. Un gradiente di pressione è applicato per superare la resistenza al passaggio del fuso, riuscendo ad ottenere una struttura finale omogenea e compatta.

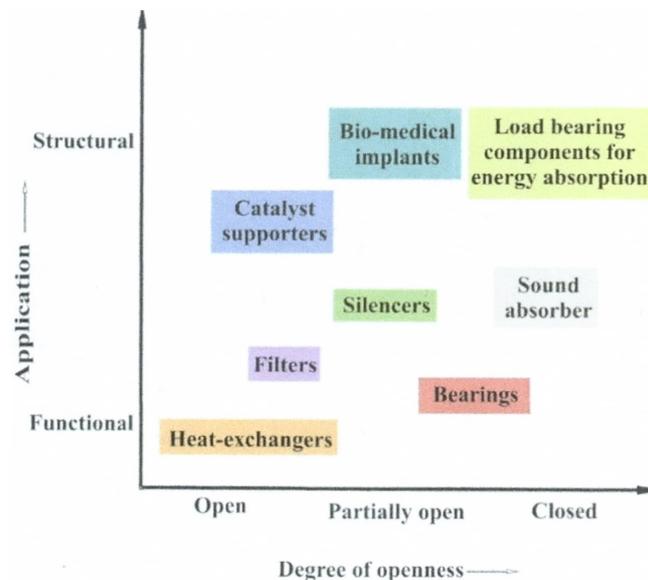


**Figura 1.3:** Processo Duocell: schema a stadi [10].

Il prodotto presenta una dimensione dei pori nel range 1 - 5 mm ed una densità relativa intorno a 0.05 [10].

### 1.2.3 Applicazioni tecnologiche

Le OCF spaziano in due ampi gradi di libertà: la funzionalità e la porosità. Differenti combinazioni di queste caratteristiche identificano i principali prodotti o le principali applicazioni ingegneristiche delle schiume solide.



**Figura 1.4:** Applicazioni per schiume metalliche e *cellular metallic materials* in funzione del grado di apertura [10].

Sono di seguito riportate alcune applicazioni per OCF metalliche:

- (i) HX (Heat Exchanger) e macchine termiche: L'utilizzo di schiume solide a base di alluminio e rame è particolarmente indicato per ottenere componenti ad alta conducibilità termica. Lo scambio termico è conseguito iniettando un fluido nelle porosità della schiuma [16]. L'ottimizzazione di tali strutture prevede di ridurre al minimo la resistenza al flusso ed aumentare il più possibile il trasporto di calore. L'elevata area di scambio, l'elevata porosità e l'eccellente conducibilità termica rendono le OCF ottime per tali impieghi. Le schiume dimostrano inoltre eccellenti performance se utilizzate come *heat sink* in microelettronica, in presenza di sistemi che necessitano di una significativa capacità di dissipazione termica [7].
- (ii) *Support* per catalizzatori: L'efficienza dei catalizzatori nei processi chimici industriali dipende fortemente dall'affinità con il substrato, dalle condizioni operative e dall'intimità del contatto tra substrato e particelle catalitiche attive, pertanto l'elevata porosità e la vasta superficie messa a disposizione dalle OCF offrono un'eccellente supporto su cui disperdere le particelle catalitiche attive. Questa soluzione è attualmente impiegata nella riduzione catalitica di ossidi di azoto  $NO_x$  emessi dalle centrali elettriche e dai motori termici delle automobili [17].
- (iii) Miscelatori statici: Il complesso network di pori che caratterizza la microstruttura è talvolta utilizzato per favorire l'intimo contatto in caso di reazioni chimiche che necessitano la pre-miscelazione dei reagenti.
- (iv) Elettrodi per batterie: Le OCF hanno dimostrato di essere una valida alternativa ai supporti porosi convenzionali utilizzati nelle *lead-acid batteries* come elettrodi. L'elevata porosità e la conducibilità elettrica giustificano l'applicazione di questa tecnologia anche nel campo delle batterie e delle fuel cell. Lo stesso principio è adottato nelle batterie *Ni-Cd*.

## 1.3 Stato dell'arte: OCF ceramiche

Schiume solide estremamente porose, sviluppate in materiale ceramico, ricoprono un'importante ruolo in applicazioni differenti rispetto le omologhe OCF metalliche. La similarità della struttura geometrica ne permette una caratterizzazione che solca il sentiero già tracciato dalle schiume metalliche. La presenza di un *network* di celle aperte garantisce bassa resistenza al passaggio di fluidi e l'elevata area di contatto tra le fasi rende questa tecnologia interessante per applicazioni di catalisi, anche in condizioni molto spinte di temperatura e pressione. La maglia reticolare, risultato della microstruttura delle schiume, incide sulla fluidodinamica del sistema

aumentando i fenomeni convettivi e, di conseguenza, migliorando il trasporto di massa e di calore [9]. Il materiale ceramico garantisce elevata resistenza meccanica e termica permettendone l'utilizzo laddove le condizioni operative fossero proibitive per materiali polimerici, metallici o compositi.

### 1.3.1 Applicazioni tecnologiche

Sono di seguito riportate due applicazioni per le schiume ceramiche; entrambe verranno brevemente descritte e dettagliate con alcuni esempi.

- (i) Processi catalitici: Rappresentano il miglior target attuale per l'impiego di schiume a celle aperte in materiale ceramico. Tali dispositivi possono essere commercializzati sottoforma di pellets o cartucce e presentano molteplici vantaggi se confrontati con i classici riempimenti impaccati: le cartucce possono essere dimensionate su misura per accomodarsi in reattori di varie forme e dimensioni, le schiume permettono di ridurre le perdite di carico per via dell'elevata porosità e di conseguenza i costi energetici (e.g. pompaggio e compressione) ad esse associati. Inoltre le vaste aree di contatto comportano una migliore efficienza in termini di trasporto di materia e calore ed una migliore miscelazione assiale e radiale contribuiscono ad evitare la formazione di *hot-spot* in alcune configurazioni reattoristiche [9].

La metanazione dell'anidride carbonica è un processo altamente esotermico che prevede l'idrogenazione della  $CO_2$  con formazione di metano ed acqua secondo la seguente reazione chimica:



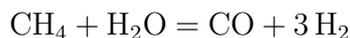
**Tabella 1.2:** Metanazione dell'anidride carbonica, confronto cinetiche [9].

	18 mm pellets	foam
wt % Ru	0.90	0.85
Effectiveness factor	0.23	$\sim 1$
Rate ( $\text{mol s}^{-1} \text{g}^{-1}$ )	$2.44 \times 10^{-3}$	$4.47 \times 10^{-3}$

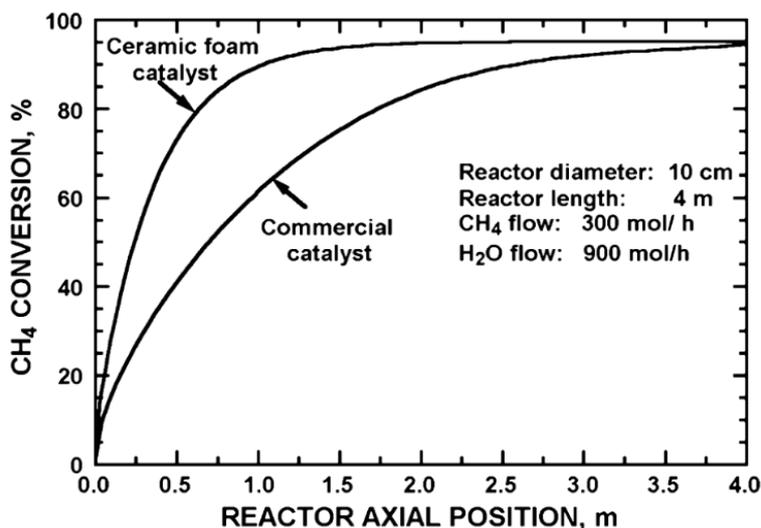
La reazione è catalizzata da rutenio disperso su  $\gamma-Al_2O_3$ . Martyn V. Twigg e James T. Richardson [9] hanno condotto un confronto fra l'uso di classici pellets di allumina da 18 mm ed una schiuma ceramica equivalente di  $\alpha-Al_2O_3$  ricoperta con  $\gamma-Al_2O_3$  al 10 wt %. A parità di condizioni operative l'attività della schiuma ricoperta di catalizzatore è stata circa il doppio dell'attività dei classici pellets da 18 mm.

Un esempio di reazione endotermica è dato dal processo di reforming del metano per il quale Richardson et al. [18] hanno dimostrato un miglioramento della cinetica di reazione ricorrendo alle OCF ceramiche come supporto per il catalizzatore al rutenio.

Il processo industriale è descritto dalla seguente reazione chimica.



Simulazioni bidimensionali hanno evidenziato come l’utilizzo di schiume ceramiche disperse di materiale catalitico comporti il raggiungimento di una conversione totale dei reagenti ad una lunghezza pari a circa la metà rispetto allo stesso reattore tubolare catalitico equipaggiato con pellets di allumina (Fig.1.5).



**Figura 1.5:** Simulazione di steam reforming del metano in reattori catalitici tubolari in condizioni operative tipiche dell’industria di processo [9].

- (ii) Filtrazione: Le schiume ceramiche trovano largo impiego come filtri per via della tortuosità della microstruttura e dell’elevata area superficiale. E.Meloni et al. [19] mostrano come sia possibile impiegare OCF ceramiche come filtri di fuliggine in caldaie alimentate a biomassa. Il loro lavoro dettaglia un’applicazione nella quale i filtri raggiungono un’efficienza di filtrazione superiore al 50% ed una eccellente efficienza di rigenerazione pari al 96%.



**Parte II**

**Elementi Teorici e Metodi  
Computazionali**



---

La seconda parte di questo lavoro riporta, nel 2° Capitolo, le fondamenta teoriche utilizzate per modellare e descrivere i fenomeni di trasporto nelle geometrie di schiume solide. In un primo momento la modellazione geometrica delle schiume verrà discussa; sarà trattato l'algoritmo di generazione delle strutture ed i suoi parametri. Successivamente saranno riportate nella sezione 2.2.2 le equazioni di governo dei fenomeni di trasporto indagati in questo lavoro. Tali equazioni descrivono il trasporto di quantità di moto ed il trasporto di calore nel fluido. Infine la sezione 2.3 sarà dedicata alla descrizione degli algoritmi di ML utilizzati in questo lavoro, il loro funzionamento e le operazioni indispensabili per la corretta esecuzione degli stessi.

Il Capitolo 3 sarà interamente dedicato alla discussione del ruolo del *Finite Volume Method* come strumento di fluidodinamica computazionale, utilizzato per risolvere le equazioni di governo su di una geometria tridimensionale. Saranno trattati i concetti di *mesh* computazionale, equazioni semi-discretizzate, discretizzazione spaziale e temporale. Le fondamenta della CFD saranno riprese ed evidenziate al fine di fornire una robusta inquadratura teorico-computazionale al presente lavoro.

La sezione 3.5 sarà dedicata all'approfondimento delle equazioni e dei solver implementati nel software di fluidodinamica computazionale `OpenFOAM`, utilizzati per redigere questo lavoro. Saranno inoltre fornite brevi descrizioni dei principali tools del software utilizzati nei processi di meshing della geometria come *blockMesh*, *snappyHexMesh* e *checkMesh*.

Un breve cenno al meccanismo di HPC (High Performance Computing), utilizzato per la risoluzione delle simulazioni OCF, sarà fornito nella sezione 3.6.



# Capitolo 2

## Elementi Teorici

### 2.1 Geometrical model

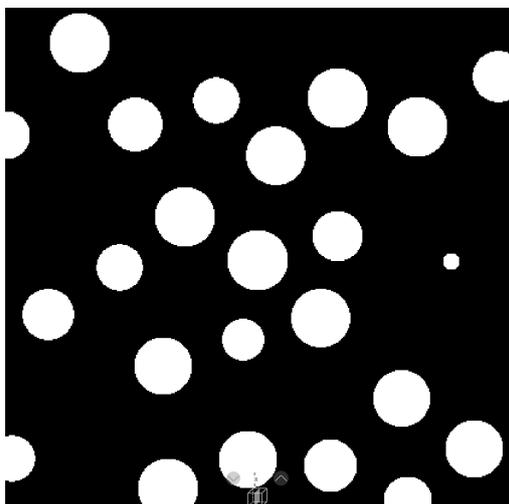
Le schiume solide generate in questo lavoro sono ottenute a partire da un modello non fisico, stocastico, sulla base di una serie di parametri geometrici forniti in ingresso. Il modello è implementato con codice Python 3.9 e fa uso del software open-source `PlugIm`, rilasciato dall'istituto IFPEN, Lione [20]. Il software `PlugIm` è utilizzato sotto forma di applicativo eseguibile e permette l'analisi e lo studio dettagliato delle geometrie in formato `.fda` o `.tif`, risultando uno strumento flessibile ed indicato per condurre la campagna esplorativa obiettivo di questo lavoro.

#### 2.1.1 L'algoritmo

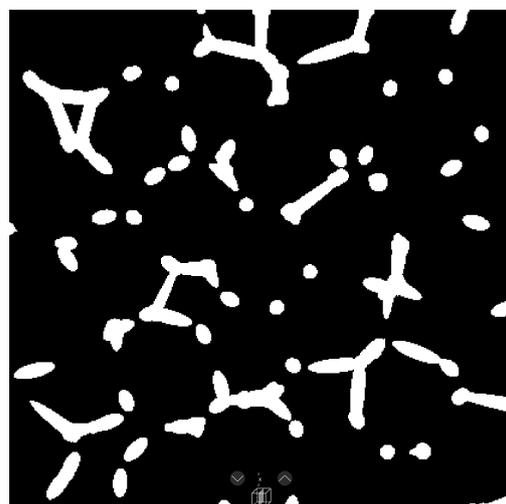
La descrizione completa del funzionamento dell'algoritmo è presente in una recente tesi di Dottorato [8]. Questo paragrafo riassume le principali operazioni eseguite dall'algoritmo ed il risultato finale. Il codice impiega l'algoritmo di tassellazione di Voronoi [21] per costruire lo scheletro tridimensionale della schiuma utilizzando come *seeds* le coordinate spaziali dei centri di  $N_{sph}$  sfere impaccate in una bounding box cubica di lato  $W$ . L'impaccamento delle sfere è strutturato su di un modello di aggregazione stocastico proposto da Ferri, Moreaud et al. [22] nel quale è possibile gestire indirettamente la compattezza dell'aggregato attraverso la regolazioni di parametri geometrici ( $\alpha$ ,  $\beta$ ,  $REP$ ) che verranno approfonditi in seguito.

La struttura ottenuta dalla tassellazione di Voronoi viene infine sottoposta ad una operazione morfologica di chiusura che ne determina il caratteristico aspetto tipico delle schiume solide (Fig.1.1). L'output dell'algoritmo consiste in un file `.tif` che racchiude l'insieme di tutte le *slices* (tagli bidimensionali della bounding box); questa struttura prende il nome di *stack*. Il file `.tif` viene infine utilizzato per costruire un modello tridimensionale della schiuma ricorrendo all'algoritmo

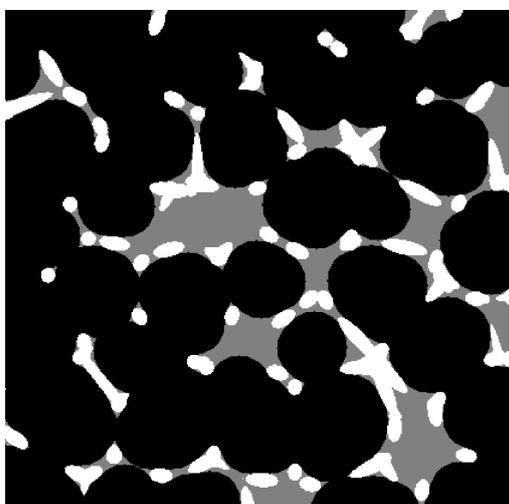
*marching cubes* [23]. Il risultato finale è un file `.stl` (stereolitografico) che può essere facilmente interpretato e visualizzato da programmi di rendering tridimensionale. Le geometrie generate nell'ambito di questo lavoro sono state studiate e modificate ricorrendo al software open-soure Paraview v5.11.0.



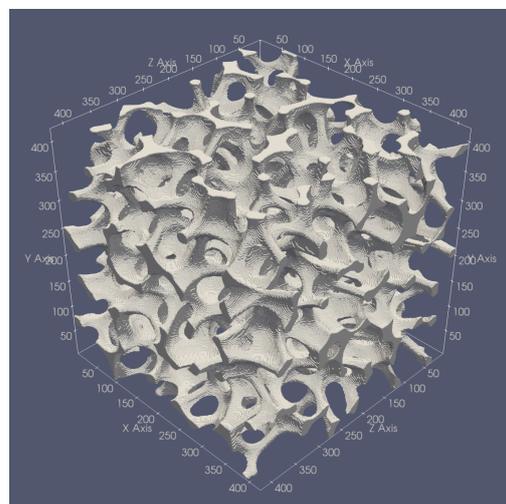
(a) Slice della bounding box contenente l'aggregato di sfere.



(b) Slice della bounding box contenente lo scheletro della schiuma ottenuto applicando la tessellazione di Voronoi.



(c) Applicazione dell'operazione morfologica di chiusura, in evidenza le zone interessate.



(d) Geometria 3D ottenuta dal file `.tif` applicando l'algoritmo *marching cubes*.

**Figura 2.1:** Fasi dell'algoritmo di generazione delle geometrie basato sul software Plugim.

## 2.1.2 Parametri geometrici

I parametri in ingresso definiscono lo spazio di lavoro e determinano le caratteristiche geometriche delle schiume generate mediante l’algoritmo che, tuttavia, essendo di tipo stocastico, restituisce *realizzazioni* sempre differenti, casuali.

**Tabella 2.1:** Parametri in ingresso all’algoritmo di generazione delle geometrie.

Descrizione	Parametro	Dimensione
Primo fattore di compattezza	$\alpha$	[-]
Secondo fattore di compattezza	$\beta$	[-]
Repulsione	$REP$	[voxel]
Dimensione dei nodi	$R_{node}$	[voxel]
Dimensione delle struts	$R_{strut}$	[voxel]
Magnitudo operazione morfologica	$tv$	[voxel]
N° di sfere da impaccare	$N_{sph}$	[#]
Raggio delle sfere impaccate	$R$	[voxel]
Dimensione della bounding box	$W$	[voxel]

La dimensione della bounding box è determinata dal parametro  $W$  che definisce la lunghezza del lato mentre la dimensione delle sfere aggregate è definita dal parametro  $R$  che ne misura il raggio. Il parametro  $N_{sph}$  determina la cardinalità delle sfere da impaccare nella bounding box rispettando i vincoli imposti dagli altri parametri; nel caso in cui, durante la fase di impaccamento, si esaurisca lo spazio disponibile per ulteriori sfere il valore  $N_{sph}$  non verrà raggiunto.

I due parametri  $\alpha$  e  $\beta$  gestiscono la compattezza dell’aggregato; essi definiscono rispettivamente la probabilità con la quale i nuovi elementi sferici vengono collocati nelle concavità dell’aggregato già esistente nella bounding box e la probabilità con la quale i nuovi elementi sferici vengono collocati in prossimità del centro di massa dell’aggregato. [22]

La minima distanza tra le sfere impaccate è garantita dal parametro di repulsione  $REP$  mentre la magnitudine dell’operazione morfologica di chiusura (in termini di voxel) è regolata dal parametro  $tv$ . Le dimensioni dei nodi e delle struts generati dalla tessellazione di Voronoi sono determinati da  $R_{node}$  ed  $R_{strut}$ .

## 2.2 Physical models

Lo studio fluidodinamico delle geometrie è fondato su modelli fisici che descrivono l'evoluzione del sistema nel tempo. Tali modelli sono costituiti da singole equazioni o set di equazioni ricavate da principi primi quali la conservazione della massa e dell'energia all'interno del sistema. Il presente lavoro si pone l'obiettivo di studiare il trasporto di quantità di moto ed il trasporto di calore all'interno delle schiume solide. Tali fenomeni sono descritti dalle equazioni di Navier-Stokes-Fourier, set di equazioni differenziali alle derivate parziali per le quali è possibile facilitarne la risoluzione assumendo opportune ipotesi semplificative.

### 2.2.1 Ipotesi adottate

La scelta delle opportune ipotesi semplificative è di cruciale importanza per ottenere un sistema il più semplice possibile, senza rinunciare ad elevati standard di accuratezza e verosimiglianza con la realtà. Le seguenti ipotesi sono state adottate:

- Il fluido è considerato un mezzo continuo, la scala alla quale sono stati studiati i fenomeni è dell'ordine della centinaia di micrometri, dimensioni ben superiori rispetto al cammino libero delle molecole del fluido.
- Il fluido è considerato incomprimibile e newtoniano.
- Il mezzo poroso è saturo della fase liquida, essa è l'unica fase presente all'interno dei pori della geometria.
- Alcune proprietà del fluido come la viscosità cinematica e la conducibilità termica sono assunte costanti.

### 2.2.2 Equazioni di governo

Le seguenti equazioni governano la fisica del sistema, esse sono scritte tenendo in considerazione le ipotesi adottate e sono riportate in notazione vettoriale.

L'equazione di continuità descrive la conservazione della massa all'interno del sistema; può essere semplificata e ridotta all'Eq.(2.1) nell'ipotesi di fluidi incomprimibili, come i liquidi, per cui si può considerare trascurabile l'effetto della pressione sulla densità.

$$\nabla \cdot \mathbf{U} = 0 \tag{2.1}$$

$\mathbf{U}$  è il vettore velocità, in un sistema di coordinate cartesiane  $U_x, U_y, U_z$  rappresentano le componenti lungo i tre assi, Eq.(2.2). La velocità del fluido è misurata in  $(\text{m s}^{-1})$ .

$$\mathbf{U} = \begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix} \quad (2.2)$$

$\nabla$  è un'operatore differenziale, in coordinate cartesiane e notazione vettoriale:

$$\nabla = \left[ \frac{\partial}{\partial x}, \quad \frac{\partial}{\partial y}, \quad \frac{\partial}{\partial z} \right] \quad (2.3)$$

Le equazioni di Navier-Stokes sono tre equazioni differenziali alle derivate parziali, una per ogni dimensione dello spazio, che descrivono il trasporto di quantità di moto nel fluido. Sono composte da un primo termine che descrive l'accumulo di quantità di moto ( $qdm$ ) all'interno del sistema, un secondo termine detto di *advezione* che descrive il trasporto di  $qdm$  dovuto ai flussi massici in entrata ed uscita dal bordo del sistema. Al secondo membro dell'equazione figurano le forze superficiali e di volume che agiscono sul sistema. Per un fluido incomprimibile e newtoniano:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{U} \quad (2.4)$$

dove  $\rho$  è la densità del fluido ed è misurata in ( $\text{kg m}^{-3}$ ),  $p$  è la pressione (Pa),  $\nu$  è la viscosità cinematica del fluido ( $\text{m}^2 \text{s}^{-1}$ ).

$\nabla^2$  è un'operatore differenziale, in coordinate cartesiane e notazione vettoriale:

$$\nabla^2 = \left[ \frac{\partial^2}{\partial x^2}, \quad \frac{\partial^2}{\partial y^2}, \quad \frac{\partial^2}{\partial z^2} \right] \quad (2.5)$$

L'equazione di trasporto dell'energia permette di studiare la distribuzione di temperatura all'interno del sistema. In presenza di fluido incomprimibile l'equazione assume la forma semplificata, Eq.(2.6). Non sono considerati contributi dovuti all'attrito viscoso per via della ridotta velocità del fluido e non vi sono termini sorgente.

$$\frac{\partial T}{\partial t} + \mathbf{U} \cdot \nabla T = \alpha_{fluid} \nabla^2 T \quad (2.6)$$

dove  $T$  è la temperatura del fluido (K),  $\alpha_{fluid}$  è la diffusività termica del fluido ed è misurata in ( $\text{m}^2 \text{s}^{-1}$ ) ed è calcolata come:

$$\alpha_{fluid} = \frac{\lambda}{\rho c_p} \quad (2.7)$$

dove  $\lambda$  è la conducibilità termica del fluido ed è misurata in ( $\text{W m}^{-1} \text{K}^{-1}$ ),  $c_p$  è il calore specifico massico del fluido a pressione costante ed è misurato in ( $\text{J kg}^{-1} \text{K}^{-1}$ ).

L'utilizzo delle reti neurali richiede l'identificazione di una grandezza che sia rappresentativa dei risultati delle simulazioni per ogni fisica tenuta in considerazione nel sistema. Nel seguente lavoro due problemi fisici differenti sono considerati; inizialmente si analizza il flusso attraverso le geometrie, successivamente lo studio si focalizza sul trasporto di calore. I parametri individuati per caratterizzare le geometrie nelle simulazioni di flusso e nelle simulazioni di scambio termico sono rispettivamente:

- (i) la permeabilità:  $\kappa$  ( $\text{m}^2$ ).
- (ii) il numero di Nusselt:  $Nu$  (-).

### La permeabilità

Tutte le simulazioni di flusso eseguite durante lo studio della permeabilità nelle schiume solide sono condotte in presenza di moto Darciano (laminare), per il quale il numero di Reynolds è molto minore dell'unità.

$$Re \ll 1 \quad (2.8)$$

Il numero di Reynolds è un numero adimensionale che può essere interpretato come misura dell'importanza relativa di advezione e diffusione nei fenomeni di trasporto di  $qdm$ . In questo lavoro, è così definito:

$$Re = \frac{U_{avg} D_h}{\nu} \quad (2.9)$$

dove  $U_{avg}$  è la velocità media del fluido lungo la direzione del gradiente di pressione ( $\text{m s}^{-1}$ ),  $D_h$  è il diametro idraulico della schiuma solida misurato in (m).

$$D_h = \frac{4\epsilon}{S_v} \quad (2.10)$$

dove  $\epsilon$  è il grado di vuoto della geometria Eq.(1.1) (-),  $S_v$  è la superficie specifica ( $\text{m}^2 \text{m}^{-3}$ ). Per evitare ambiguità è riportata la definizione di superficie specifica adottata nel presente lavoro:

$$S_v = \frac{S_{foam}}{V_{bb}} \quad (2.11)$$

dove  $S_{foam}$  è la superficie effettiva della geometria esposta al contatto con il fluido ( $\text{m}^2$ ),  $V_{bb}$  è il volume totale della bounding box ( $\text{m}^3$ ).

La permeabilità è valutata mediante la legge di Darcy, valida per le condizioni operative descritte in precedenza (Eq.2.8):

$$\frac{\Delta\mathcal{P}}{L} = \frac{\nu}{\kappa}u_s \quad (2.12)$$

dove  $\Delta\mathcal{P}$  è il gradiente di pressione cinematica ( $\text{m}^2 \text{s}^{-2}$ ),  $L$  è la lunghezza del canale simulato (m),  $u_s$  è la velocità superficiale del fluido ( $\text{m s}^{-1}$ ).

### Il numero di Nusselt

Le simulazioni di scambio termico sono condotte in condizioni di moto laminare e di transizione ( $1 < Re < 60$ ). Le relazioni utilizzate per il calcolo di  $Nu$  sono riportate a seguito.

$$\dot{Q} = \rho\dot{q}(c_{p,out}T_{out} - c_{p,in}T_{in}) \quad (2.13)$$

dove  $\dot{Q}$  è il flusso di calore scambiato tra fase solida e fluido (W),  $\dot{q}$  è la portata volumica di fluido ( $\text{m}^3 \text{s}^{-1}$ ),  $c_{p,out}$  ( $\text{J kg}^{-1} \text{s}^{-1}$ ) è il calore specifico del fluido, per unità di massa, valutato alla temperatura media di uscita  $T_{out}$  (K),  $c_{p,in}$  è il calore specifico del fluido, per unità di massa, valutato alla temperatura media di ingresso  $T_{in}$  (K).

$$h = \frac{\dot{Q}}{S_{foam}\Delta T} \quad (2.14)$$

dove  $h$  è il coefficiente di scambio termico ( $\text{W m}^{-2} \text{K}^{-1}$ ),  $\Delta T$  è la forza spingente ovvero la differenza tra la temperatura sulla superficie della schiuma  $T_{foam}$  e la temperatura del fluido  $T_{fluid}$  (K).

$$\Delta T = T_{foam} - T_{fluid} \quad (2.15)$$

Seguendo un semplice modello già adottato da Guo et al. [24] la temperatura del fluido è valutata come media aritmetica tra la temperatura media in uscita e la temperatura media in ingresso al sistema.

$$T_{fluid} = \frac{1}{2}(T_{out} - T_{in}) \quad (2.16)$$

Infine il numero di Nusselt in questo lavoro è calcolato come:

$$Nu = \frac{h D_h}{\lambda} \quad (2.17)$$

## 2.3 Data-driven models

Un modello data-driven è un'approccio modellistico nel quale si cerca di estrarre relazioni funzionali direttamente dai dati. Si contrappone all'approccio classico nel quale si cercano delle equazioni esplicite tra le variabili indagate, tali da descrivere i fenomeni oggetto di studio. Nella costruzione di modelli data-driven assumono particolare importanza i concetti di istanza, *features* (i.e. caratteristiche, dati in ingresso) e *target* (i.e. target, obiettivo). Un'istanza è un singolo elemento del dataset, per esempio nell'ambito di questo lavoro il dataset sarà l'insieme delle simulazioni sulle schiume solide mentre le istanze saranno le singole simulazioni che compongono il dataset. Ogni istanza è proprietaria di alcune features che la caratterizzano; nel caso delle simulazioni di flusso le features (dati di input) sono i parametri generatori (Tab.2.1), utilizzati dal modello geometrico per costruire la schiuma indagata in quella specifica simulazione. Ad ogni istanza è associato un valore rappresentativo; nel caso precedente il valore rappresentativo delle simulazioni di flusso è stato individuato nella permeabilità ( $\kappa$ ). E' stato scelto il valore di permeabilità perchè risulta essere un ottimo descrittore del comportamento della schiuma, in regime laminare, in presenza di un flusso fluido nelle sue cavità, Eq.(2.12). L'obiettivo di un modello data-driven supervisionato, come quello impiegato in questo lavoro, è restituire il valore target (permeabilità) sulla base dei valori delle features (parametri del modello geometrico) di nuove istanze. Le nuove istanze, in questo specifico contesto, sono nuove geometrie di OCF non ancora sottoposte ad analisi fluidodinamica.

Gli step fondamentali per ottenere un modello data-driven sono due: il *training* del modello e l'interrogazione del modello, ossia il *testing*. La fase di training varia in base all'algoritmo di ottimizzazione scelto, in generale durante questo step l'algoritmo ottimizza i parametri del modello, migliorando la sua capacità predittiva su nuove istanze. La seconda fase è quella di interrogazione del modello ovvero vengono fornite le features di nuove istanze ed il modello restituisce le corrispondenti labels predette. La fase di training solitamente è la più dispendiosa in termini di risorse computazionali mentre la fase di testing è istantanea.

E' possibile classificare i modelli data-driven in quattro grandi classi [25] sulla base dell'impostazione assegnata al modello, sia in termini di *training* sia a livello di target. Di seguito saranno dettagliate meglio le classi dei modelli e le loro caratteristiche peculiari.

- Modelli supervisionati: Sono modelli caratterizzati da una fase di training nella quale sono fornite delle istanze (training set) complete di features e labels. In questo lavoro saranno impiegati modelli data-driven supervisionati ed il training set sarà composto da una serie di istanze (simulazioni) per le quali sono note sia le features (parametri generatori del modello geometrico) sia le

labels (permeabilità), calcolate a partire dai risultati delle simulazioni CFD. I modelli supervisionati permettono di condurre regressioni o classificazioni di nuove istanze, una volta conclusa la fase di training.

- Modelli non supervisionati: Sono modelli che vengono allenati a partire da un training set privo di labels. Solitamente questi modelli cercano relazioni e somiglianze nei dati; un classico esempio è dato dagli algoritmi di *clustering* che suddividono le istanze in classi cercando di raggruppare quelle simili tra di loro.
- Modelli semi-supervisionati: Questi modelli sono ibridi tra quelli precedentemente descritti. Spesso il loro obiettivo è utilizzare una grande quantità di dati non etichettati per migliorare le performance di modelli supervisionati.
- Reinforcement learning: Sono modelli basati su un meccanismo di benefici e perdite, le risposte che ricevono feedback positivi consolidano il "percorso decisionale" del modello mentre le risposte che ricevono feedback negativi inducono modifiche nel "percorso decisionale" del modello.

Nell'ambito di questo lavoro si porrà particolare attenzione ai modelli data-driven supervisionati, impiegati nella realizzazione di un modello predittivo di regressione, il cui obiettivo è la predizione di una variabile target numerica (permeabilità). Sarà anche sviluppato un modello di classificazione che permetterà di filtrare le geometrie in ingresso, rimuovendo a priori quelle che presentano caratteristiche non fisiche; questo step è necessario in quanto le OCF sono ottenute da un modello geometrico che non implementa nessuna fisica.

### 2.3.1 Modelli di regressione

I modelli di regressione sono modelli supervisionati che hanno l'obiettivo di predire una variabile target numerica, in funzione dei valori delle *explanatory variables*, ovvero le features delle nuove istanze.

$$y = f(x_1, x_2, \dots, x_n) \quad (2.18)$$

dove  $y$  è la variabile target numerica ed  $x_i$  è la generica feature. Il processo di training richiede una collezione di istanze, anche chiamate *data-object*, caratterizzate da un set di features (attributi) ed una target label (valore numerico). I modelli di regressione possono essere classificati in base al numero di *explanatory variables* o regressori in:

- Regressione semplice: Il valore target è predetto utilizzando una sola feature in input.

$$y = f(x_1) \quad (2.19)$$

- Regressione multipla: Il valore target è predetto utilizzando  $n$  features in input, Eq.(2.18).

Una seconda classificazione è invece costruita sul tipo di relazione funzionale tra la variabile target e le variabili di input:

- Regressione lineare: La relazione è lineare rispetto alle features
- Regressione non lineare: La relazione non è lineare; un esempio è dato da regressioni con relazione logaritmica tra la variabile target e le features.

Il più semplice esempio di regressione è dato dal modello lineare semplice, dove il valore target è legato linearmente ad una singola feature,  $x_1$ .

$$\hat{y} = \beta_0 + \beta_1 x_1 \quad (2.20)$$

$\beta_0, \beta_1$  sono i coefficienti del modello di regressione. L'obiettivo della fase di training del modello è determinare il valore dei coefficienti di regressione in modo da minimizzare l'errore tra il valore target reale e la predizione fornita dal modello stesso. Una tecnica popolare per raggiungere questo obiettivo è quella di minimizzare la somma dei quadrati dei residui; LSM (Least Squares Method).

$$\min(RSS) = \min\left(\sum_i (r_i)^2\right) = \min\left(\sum_i (y_i - (\beta_0 + \beta_1 x_1))^2\right) \quad (2.21)$$

dove i residui  $r_i$  sono così definiti:

$$r_i = y_i - \hat{y}_i \quad (2.22)$$

in particolare,  $y_i$  è il valore reale della variabile target mentre  $\hat{y}_i$  è il valore predetto dal modello di regressione.

In generale l'Eq.(2.20) può essere generalizzata per  $n$  regressori, ottenendo l'equazione generica per un modello di regressione lineare multivariabile.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \quad (2.23)$$

E' importante osservare come i singoli coefficienti  $\beta_i$  vadano stimati e testati separatamente, ovvero  $\beta_i$  dovrebbe rappresentare l'effetto medio su  $\hat{y}$  dovuto ad una variazione unitaria su  $x_i$ , mantenendo il resto costante. Per tale motivo è di estrema importanza che i regressori (features) siano indipendenti e non correlati tra di loro; per esempio in caso di istanze con molti attributi (*high dimensional data-object*) alcuni di essi potrebbero contenere informazioni ridondanti, essendo pertanto correlati e determinando quindi problemi nella stima dei coefficienti di

regressione  $\beta_i$ . In questi casi è necessario procedere con una analisi di correlazione delle features ed una fase di *feature engineering*, ovvero di riformulazione delle features in ingresso riducendo la dimensionalità dell'istanza (i.e. riducendo il numero di features). Tale processo sarà dettagliato meglio nella sezione 4.5. Talvolta un problema riscontrato nelle operazioni di regressione è il cosiddetto *data overfitting*, ovvero la costruzione di un modello che descrive perfettamente i dati disponibili ma affetto da ridotta generalizzazione. L'uso di tecniche di regolarizzazione come Ridge e Lasso permettono di costruire modelli più generalizzati. La tecnica LASSO (Least Absolute Shrinkage and Selection Operator) consiste per esempio nell'assegnare valori pari a zero ai coefficienti  $\beta_i$  riferiti a features poco importanti ai fini della regressione; in questo modo la complessità del modello è ridotta e migliora la generalizzazione. Esistono altri metodi per costruire modelli di regressione come DT (Decision Tree) e RF (Random Forest), che risultano semplici e spesso interpretabili o, in alternativa, tecniche più elaborate con maggiori potenzialità come le NN (Rete neurale). Marcato et al. [26] mostrano un esempio di applicazione di reti neurali per lo studio del flusso e di fenomeni di trasporto di massa all'interno di geometrie di mezzi porosi. Il presente lavoro utilizzerà architetture MLP (Multi Layers Perceptrons) per la costruzione di modelli di regressione data-driven, in particolare saranno utilizzati i tools della libreria Python `sklearn` [27]. La valutazione della bontà dei modelli di regressione costruiti farà affidamento sulle seguenti metriche:

- MAE (Mean Absolute Error): La metrica definisce l'errore medio assoluto ed è calcolata come:

$$MAE = \frac{1}{n} \sum_i |r_i| = \frac{1}{n} \sum_i |y_i - \hat{y}_i| \quad (2.24)$$

- MSE (Mean Squared Error): La metrica definisce l'errore medio quadratico ed è calcolata come:

$$MSE = \frac{1}{n} \sum_i (r_i)^2 = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2 \quad (2.25)$$

- $R^2$ : Il coefficiente rappresenta la bontà del modello nel fitting dei dati. E' calcolato come:

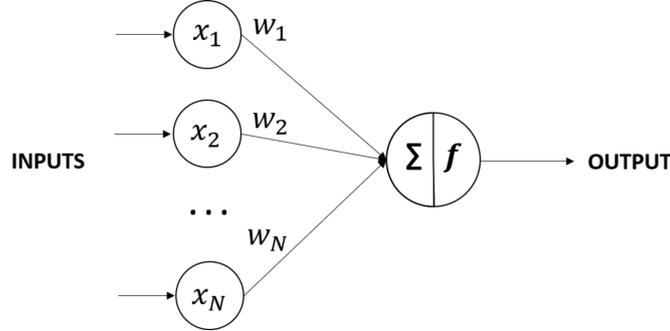
$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \quad (2.26)$$

con  $\bar{y} = \frac{1}{n} \sum_i y_i$

### Architettura MLP

L'uso delle reti neurali è particolarmente indicato per sistemi multidimensionali, non lineari. L'architettura MLP comprende una rete di unità elementari dette

perceptroni. Ognuna di queste unità elabora delle informazioni in ingresso e restituisce in uscita un valore che sarà l'input per l'unità successiva.



**Figura 2.2:** Il più semplice perceptrone, la Threshold Logic Unit [28].

Ogni perceptrone riceve una serie di input  $(x_1, x_2, \dots, x_N)$ , ne calcola la somma pesata introducendo un bias ed applica al risultato una activation function  $\varphi$ :

$$z = \varphi(w_1x_1 + w_2x_2 + \dots + w_Nx_N + b) = \varphi(\mathbf{w}^T \mathbf{x}) \quad (2.27)$$

dove  $w_i$  ed  $x_i$ , con  $i = 1, 2, \dots, N$ , sono rispettivamente i pesi e gli input del perceptrone mentre  $b$  è il bias. L'activation function utilizzata in questo lavoro è la ReLU (Rectified Linear Unit) ed è espressa come:

$$ReLU(z) = \max(0, z) \quad (2.28)$$

Più perceptron possono formare dei livelli e l'architettura MLP è caratterizzata dal numero di layers e dal numero di neuroni per layer. I calcoli condotti in un singolo layer di una rete neurale MLP sono descritti dalla seguente equazione:

$$\begin{aligned} z_1 &= \varphi(w_{1,1}x_1 + w_{1,2}x_2 + \dots + w_{1,N}x_N + b_1) \\ z_2 &= \varphi(w_{2,1}x_1 + w_{2,2}x_2 + \dots + w_{2,N}x_N + b_2) \\ &\vdots \\ z_M &= \varphi(w_{M,1}x_1 + w_{M,2}x_2 + \dots + w_{M,N}x_N + b_M) \end{aligned} \quad (2.29)$$

dove  $z_j$  con  $j = 1, 2, \dots, M$  sono gli output delle  $M$  unità elementari che compongono un generico layer in una architettura MLP. In forma matriciale:

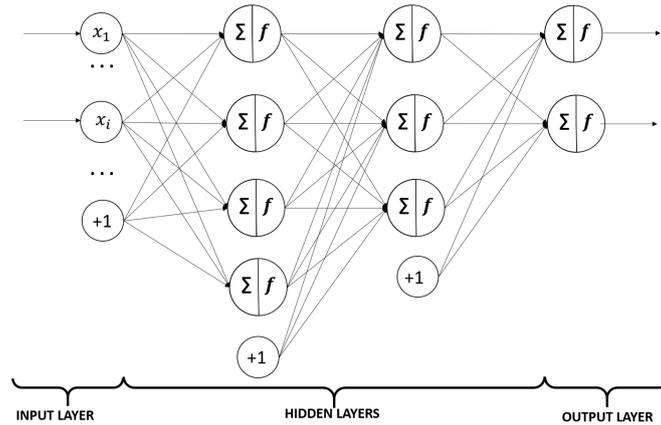
$$\mathbf{z} = \varphi(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (2.30)$$

dove  $\mathbf{W}$  è la matrice dei pesi,  $\mathbf{x}$  è il vettore delle features,  $\mathbf{b}$  è il vettore dei bias e  $\mathbf{z}$  è il vettore degli output per un generico layer.

La costruzione di architetture complesse su più layers definisce le cosiddette *deep neural network* o DNN. In una generica architettura MLP il grafo è costituito

da un input layer nel quale sono immessi i valori delle features, come definiti in sezione 2.1, una serie di hidden layer costituiti da un numero variabile di unità elementari ed un output layer che restituisce il valore numerico target, in questo lavoro la permeabilità. L'Eq. (2.31) descrive i calcoli eseguiti da una semplice rete MLP, dotata di input, output layers ed un singolo hidden layer.

$$z = \varphi^{II} \left[ \mathbf{W}^{II} \varphi^I \left( \mathbf{W}^I \mathbf{x} + \mathbf{b}^I \right) + \mathbf{b}^{II} \right] \quad (2.31)$$



**Figura 2.3:** Semplice rete MLP caratterizzata da due hidden layer oltre ai livelli di input ed output [28].

L'algoritmo di training scorre le istanze del training set una alla volta ed esegue le seguenti azioni:

- Iniziando dal primo layer è calcolato l'output del layer corrente utilizzando l'Eq. (2.30).
- Il calcolo procede per tutti i layers successivi della rete (forward propagation) finchè l'output finale è calcolato.
- L'output finale è confrontato con l'output atteso (label nota) ed è valutato l'errore.
- L'algoritmo di retropropagazione dell'errore scorre nuovamente la rete dall'output finale verso gli input aggiornando i valori di pesi ( $\mathbf{W}^i$ ) e bias ( $\mathbf{b}^i$ ) per ogni neurone.

L'algoritmo BFGS (Broyden–Fletcher–Goldfarb–Shanno Algorithm) è un metodo iterativo utilizzato nella risoluzione di problemi di ottimizzazione non lineari. L'algoritmo di ottimizzazione utilizzato è LBFGS (Limited-memory BFGS Algorithm),

una variante del metodo BFGS utilizzata in applicazioni Machine Learning. Il processo di ottimizzazione si conclude quando è raggiunta una determinata tolleranza dell'errore rispetto all'output atteso (labels note). Un secondo possibile criterio di stop è dato imponendo un limite al numero di iterazioni.

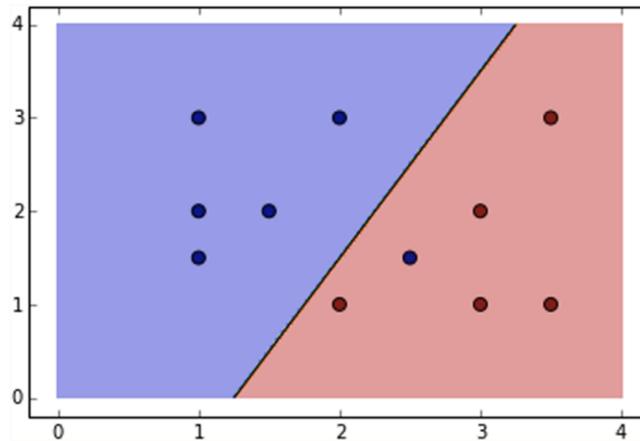
### 2.3.2 Modelli di classificazione

I modelli di classificazione hanno l'obiettivo di assegnare una class label (classe) a nuove istanze, previa fase di allenamento del modello che richiede un set di data-object per cui è nota la class label (supervised learning). Molteplici tecniche sono disponibili per la costruzione di classificatori tra cui i già citati DT (Decision Tree) e RF (Random Forest). Questi metodi sono caratterizzati da bassa complessità, elevata efficienza, interpretabilità del modello e sono robusti rispetto a *noise* ed *outliers*. Metodi più elaborati comprendono l'uso di reti neurali o modelli SVM (Support Vector Machine) per costruire il classificatore. In questo lavoro sarà costruito un modello SVM di classificazione per filtrare le geometrie non fisiche, considerabili come outliers per il modello di regressione. Le tecniche SVM presentano eccellenti performance nonostante siano modelli non interpretabili e complessi dal punto di vista del *tuning* dei parametri.

#### SVM Classifier

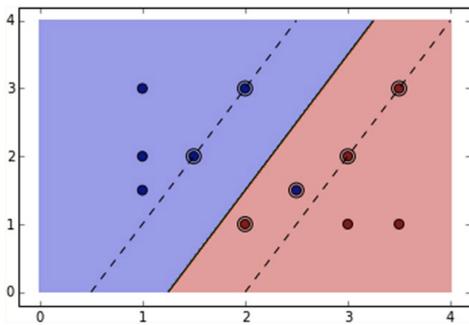
Durante la fase di apprendimento, questi classificatori proiettano le osservazioni in uno spazio multidimensionale chiamato spazio decisionale e costruiscono una superficie di separazione chiamata confine decisionale che divide questo spazio in due aree di appartenenza. Nel caso più semplice, cioè lineare, il confine decisionale sarà rappresentato da un piano (in 3D) o da una linea retta (in 2D). La linea retta verrà posizionata in modo da massimizzare la sua distanza dalle osservazioni più vicine contenute nel training set. Questa condizione dovrebbe produrre due porzioni diverse in cui dovrebbero essere contenuti tutti i punti di una stessa classe. Nei casi più complessi le superfici di separazione sono forme curve, articolate.

Considerando il caso più semplice, ovvero con dominio bidimensionale, la Fig. (2.4) mostra l'applicazione dell'algorithm SVM ad un dataset contenente 11 data-object assegnati a due class labels differenti (blu, rosso). L'algorithm divide lo spazio decisionale in due zone di appartenenza, le nuove istanze fornite al modello saranno pertanto classificate con la class label "rosso" se ricadono nella zona rossa o con la class label "blu" se ricadono nella zona blu.

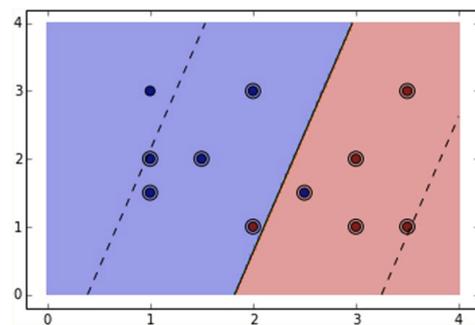


**Figura 2.4:** Algoritmo SVM applicato ad un generico caso bidimensionale, ovvero dove ogni data-object è caratterizzato da due features [27].

Un concetto importante nell'algoritmo SVM è dato dalla regolarizzazione. Essa dipende dal parametro  $C$ : un valore piccolo di  $C$  significa che il confine decisionale è calcolato utilizzando molte o tutte le osservazioni attorno alla linea di separazione (maggiore regolarizzazione), mentre un valore grande di  $C$  significa che il confine decisionale è calcolato sulle osservazioni più vicine alla separazione (regolarizzazione inferiore). Se non diversamente specificato, il valore predefinito di  $C$  è pari a 1.



(a) Algoritmo SVM caratterizzato da minore regolarizzazione,  $C = 1$ .



(b) Algoritmo SVM caratterizzato da maggiore regolarizzazione,  $C = 0.1$ .

**Figura 2.5:** Effetto della regolarizzazione (parametro  $C$ ) nell'algoritmo SVM di classificazione [27].



## Capitolo 3

# Metodi Computazionali

In questo capitolo saranno descritti i principali concetti di fluidodinamica computazionale. Una prima sezione tratterà il metodo dei volumi finiti e l'approccio utilizzato per studiare campi scalari e vettoriali risolvendo le equazioni di Navier-Stokes-Fourier. Successivamente una sezione sarà riservata alla definizione delle equazioni algebriche risolte nel FVM. Saranno poi descritti i metodi di discretizzazione spaziale e temporale del dominio computazionale. Infine alcuni cenni relativi al software open source `OpenFOAM`, utilizzato in questo lavoro per implementare il FVM. Le nozioni di CFD presentate in questo capitolo sono tratte dal libro "*The Finite Volume Method in Computational Fluid Dynamics*" scritto da F.Moukalled, L.Mangani, M.Darwish [29].

### 3.1 The Finite Volume Method

Il metodo dei volumi finiti, FVM, è stato adottato nel campo della fluidodinamica computazionale successivamente rispetto ad altri approcci numerici come le differenze finite o gli elementi finiti. L'adozione del FVM è supportata da grandi vantaggi, in primis l'intrinseca conservazione delle proprietà come massa ed energia. Inoltre l'ampia flessibilità del FVM è data dal fatto che il processo di discretizzazione può essere conseguito direttamente nello spazio fisico senza bisogno di alcuna trasformazione tra il dominio fisico e quello computazionale [30]. Il processo di discretizzazione procede in parallelo a livello geometrico ed a livello di equazioni di trasporto come schematizzato in Fig.(3.1)

La discretizzazione dell'intera geometria in celle avviene seguendo un processo detto *meshing* del dominio computazionale. Allo stesso modo le equazioni di trasporto devono essere discretizzate e risolte su ogni cella del dominio, di conseguenza le celle non possono essere indipendenti ma interconnesse, in modo da scambiarsi informazioni tra volumi adiacenti.

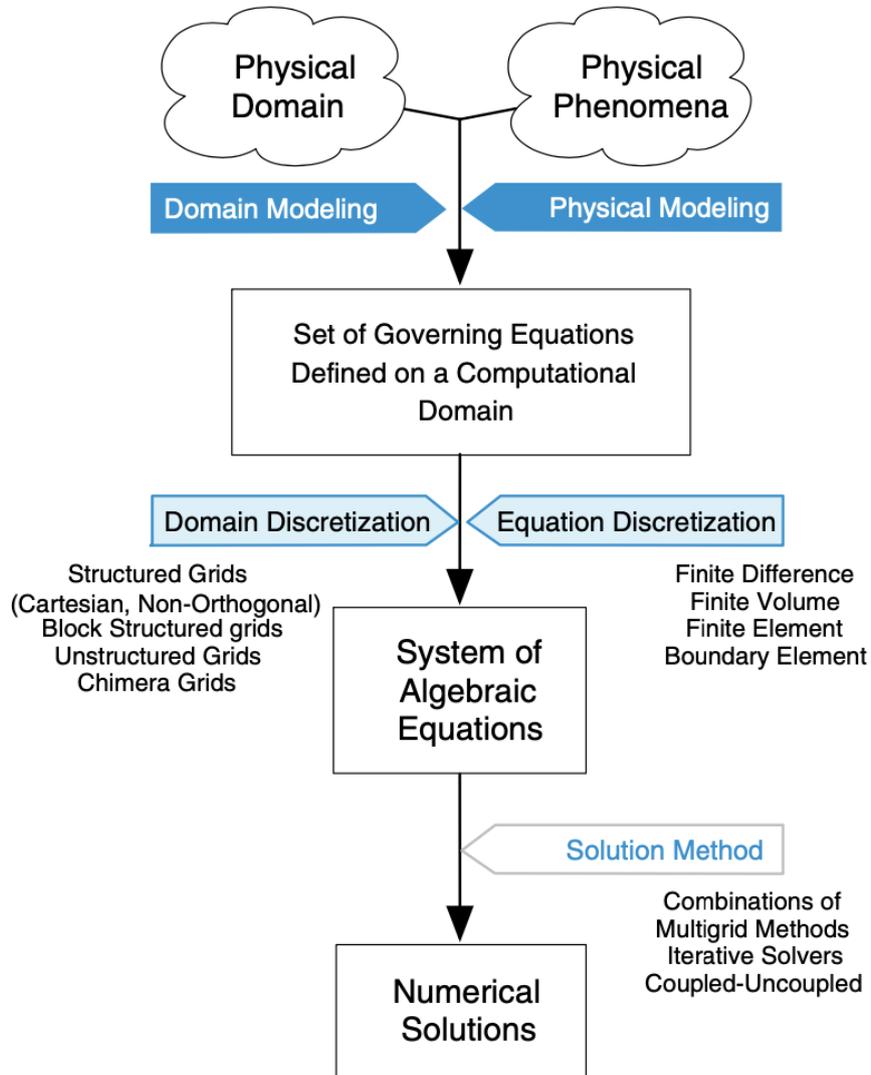


Figura 3.1: Processo di discretizzazione nel FVM [31].

### 3.1.1 Equazioni semi-discretizzate

Il primo step del FVM consiste nella definizione delle equazioni semi-discretizzate a partire dalle equazioni di trasporto che descrivono il sistema da simulare. In questa sezione, per semplicità, si farà riferimento alla generica equazione di conservazione di uno scalare ma le stesse considerazioni possono essere applicate alle equazioni di Navier-Stokes-Fourier.

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho\mathbf{U}\phi) = \nabla \cdot (\Gamma^\phi \nabla \phi) + Q^\phi \quad (3.1)$$

dove  $\phi$  rappresenta il generico scalare,  $\Gamma^\phi$  rappresenta il coefficiente di diffusività generico ed è misurato in  $(\text{m}^2 \text{s}^{-1})$ .  $Q^\phi$  è il termine sorgente nell'equazione di trasporto dello scalare. Se il sistema è stazionario l'equazione assume la seguente forma:

$$\nabla \cdot (\rho\mathbf{U}\phi) = \nabla \cdot (\Gamma^\phi \nabla \phi) + Q^\phi \quad (3.2)$$

Se consideriamo una cella C ed integriamo Eq.(3.1) sul volume della cella  $V_C$ , applicando il teorema di Gauss al termine convettivo e diffusivo si ottiene la seguente equazione:

$$\oint_{\partial V_C} (\rho\mathbf{U}\phi) \cdot d\mathbf{S} = \oint_{\partial V_C} (\Gamma^\phi \nabla \phi) \cdot d\mathbf{S} + \int_{V_C} Q^\phi dV \quad (3.3)$$

dove  $\mathbf{S}$  è il vettore superficie e  $\oint_{\partial V_C}$  è l'integrale superficiale sul volume  $V_C$ .

Si indica con  $\mathbf{J}^{\phi,C}$  e  $\mathbf{J}^{\phi,D}$  rispettivamente il flusso convettivo e diffusivo:

$$\mathbf{J}^{\phi,C} = \rho\mathbf{U}\phi \quad (3.4)$$

$$\mathbf{J}^{\phi,D} = -\Gamma^\phi \nabla \phi \quad (3.5)$$

$$\mathbf{J}^\phi = \mathbf{J}^{\phi,C} + \mathbf{J}^{\phi,D} \quad (3.6)$$

E' possibile ora sostituire l'integrale superficiale sulla cella C con la somma dei termini di flusso relativi alle facce della cella C. In questo modo l'integrale superficiale del flusso convettivo, diffusivo e totale diventano rispettivamente:

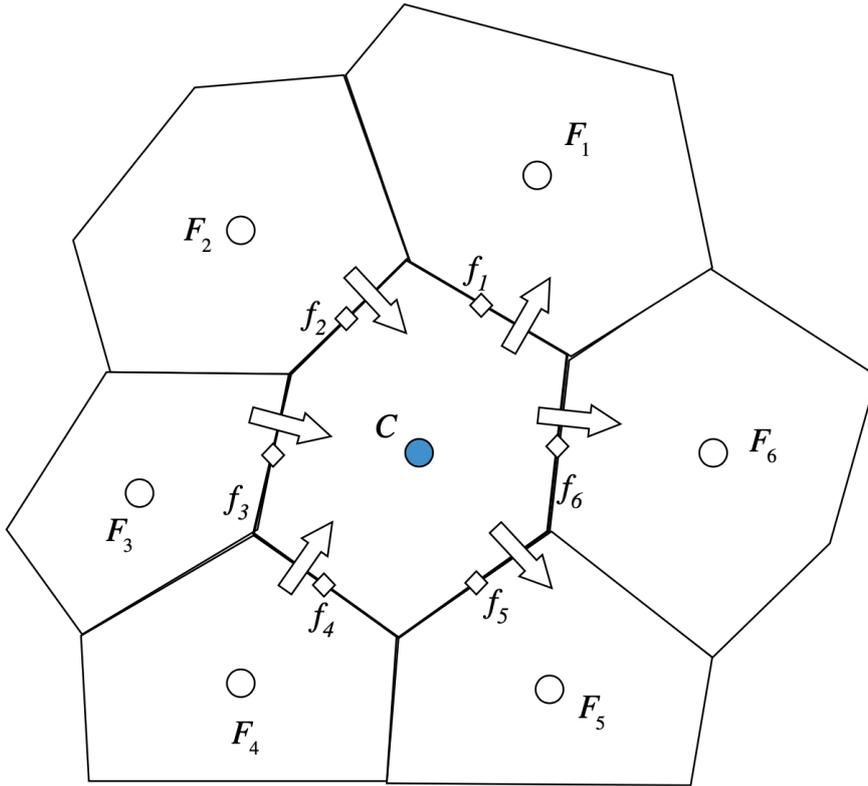
$$\oint_{\partial V_C} \mathbf{J}^{\phi,C} \cdot d\mathbf{S} = \sum_{f \sim \text{faces}(V_C)} \left( \int_f (\rho\mathbf{U}\phi) \cdot d\mathbf{S} \right) \quad (3.7)$$

$$\oint_{\partial V_C} \mathbf{J}^{\phi, D} \cdot d\mathbf{S} = \sum_{f \sim \text{faces}(V_C)} \left( \int_f (\Gamma^\phi \nabla \phi) \cdot d\mathbf{S} \right) \quad (3.8)$$

$$\oint_{\partial V_C} \mathbf{J}^\phi \cdot d\mathbf{S} = \sum_{f \sim \text{faces}(V_C)} \left( \int_f \mathbf{J}_f^\phi \cdot d\mathbf{S} \right) \quad (3.9)$$

Infine è necessario calcolare gli integrali sulle facce  $f$  della cella  $C$  e per farlo si introduce l'uso di una formula di quadratura. E' pratica comune nel FVM utilizzare un unico punto di integrazione (punto medio) ottenendo così un'accuratezza del secondo ordine [30]. Seguendo questo approccio l'Eq.(3.3) semi-discretizzata risulta essere:

$$\sum_{f \sim \text{nb}(C)} (\rho \mathbf{U} \phi - \Gamma^\phi \nabla \phi)_f \cdot \mathbf{S}_f = Q_C^\phi V_C \quad (3.10)$$



**Figura 3.2:** Flusso attraverso le  $\text{nb}(C)$  facce  $f$  della cella  $C$  [30].

### 3.1.2 Equazioni algebriche

L'obiettivo del secondo step nel processo di discretizzazione è quello di trasformare Eq.(3.10) in una equazione algebrica esprimendo il flusso superficiale e volumico in funzione dei valori delle variabili memorizzate nei *centroids* delle celle adiacenti (*neighboring cells*).

Focalizziamoci sul flusso superficiale  $f_1$  tra la cella C e la cella adiacente  $F_1$ ; lo stesso approccio è valido per ogni altra cella adiacente a C e per ogni cella C nel dominio computazionale. Il flusso può essere scomposto in un contributo lineare, proporzionale al valore della proprietà  $\phi$  nei centroidi della cella C ( $\phi_C$ ) e della cella adiacente  $F_1$  ( $\phi_F$ ), ed un contributo non lineare che non dipende dai valori  $\phi_C$  e  $\phi_F$ . Ovvero:

$$\mathbf{J}_f^\phi \cdot \mathbf{S}_f = FluxT_f = FluxC_f \phi_C + FluxF_f \phi_F + FluxV_f \quad (3.11)$$

dove  $FluxT_f$  è il flusso totale ed è decomposto in tre termini:

- (i)  $FluxC_f \phi_C$  : E' il prodotto tra il coefficiente di linearizzazione di C ed il valore della proprietà  $\phi$  nel centroide della medesima cella.
- (ii)  $FluxF_f \phi_F$  : E' il prodotto tra il coefficiente di linearizzazione di F ed il valore della proprietà  $\phi$  nel centroide della medesima cella.
- (iii)  $FluxV_f$  : E' il termine non lineare del flusso.

I valori dei coefficienti  $FluxC_f$ ,  $FluxF_f$ ,  $FluxV_f$  dipendono dal termine discretizzato e dallo schema di discretizzazione impiegato [30]. La linearizzazione del flusso è ottenuta sostituendo Eq.(3.11) in Eq.(3.10) ottenendo:

$$\sum_{f \sim nb(C)} (\mathbf{J}_f^\phi \cdot \mathbf{S}_f) = \sum_{f \sim nb(C)} (FluxT_f) = \sum_{f \sim nb(C)} (FluxC_f \phi_C + FluxF_f \phi_F + FluxV_f) \quad (3.12)$$

Anche il flusso volumico può essere scomposto in un termine lineare, proporzionale al valore della proprietà nel centroide della cella C, ed un termine non lineare:

$$Q_C^\phi V_C = FluxT = FluxC \phi_C + FluxV \quad (3.13)$$

Sostituendo Eq.(3.12) ed Eq.(3.13) in Eq.(3.10) si ottiene l'equazione algebrica:

$$a_C \phi_C + \sum_{F \sim NB(C)} (a_F \phi_F) = b_C \quad (3.14)$$

dove le seguenti relazioni legano i valori dei coefficienti algebrici con i valori dei coefficienti di linearizzazione del flusso:

$$a_C = \sum_{f \sim nb(C)} FluxC_f - FluxC \quad (3.15)$$

$$a_F = FluxF_f \quad (3.16)$$

$$b_C = \sum_{f \sim nb(C)} FluxV_f - FluxV \quad (3.17)$$

E' possibile definire un'equazione algebrica come Eq.(3.14) per ogni cella C nel dominio computazionale ed essendo le equazioni algebriche espresse in funzione dei valori di  $\phi$  nelle celle adiacenti a quella considerata il risultato sarà un sistema di equazioni algebriche, una per ogni cella, che potrà essere risolto con una procedura ricorsiva da cui si otterrà il campo dello scalare  $\phi$  nel dominio computazionale.

### 3.1.3 Condizioni di bordo

In corrispondenza dei bordi del dominio computazionale le condizioni di bordo devo essere impiegate per chiudere il problema. Le due condizioni maggiormente impiegate nel FVM sono:

- (i) Dirichlet BC: Il valore della proprietà  $\phi$  al bordo del dominio è specificato.

$$\phi_b = \phi_{b, noto} \quad (3.18)$$

Questa condizione spesso è utilizzata per descrivere un inlet di proprietà  $\phi$  assumendo il flusso diffusivo trascurabile rispetto quello convettivo [30]. Il flusso al bordo è calcolato conoscendo il valore della proprietà al bordo ( $\phi_b$  nota).

$$\mathbf{J}_b^\phi \cdot \mathbf{S}_b = \mathbf{J}_b^{\phi, C} \cdot \mathbf{S}_b = (\rho \mathbf{U} \phi)_b \cdot \mathbf{S}_b = (\rho_b \mathbf{U}_b \cdot \mathbf{S}_b) \phi_b = \dot{m}_f \phi_{b, noto} \quad (3.19)$$

- (ii) Neumann BC: Il valore del flusso della proprietà al bordo del dominio è specificato.

$$\mathbf{J}_b^\phi \cdot \mathbf{S}_b = \mathbf{J}_b^\phi \cdot \mathbf{n}_b S_b = q_{b, noto} S_b \quad (3.20)$$

dove  $q_{b, noto}$  è una quantità nota, espressa come flusso di  $\phi$  per unità di area.

### 3.1.4 Risoluzione del sistema di equazioni algebriche

Il processo di discretizzazione descritto nelle sezioni 3.1.1 e 3.1.2 si conclude con la costruzione di un sistema lineare di equazioni algebriche della forma:

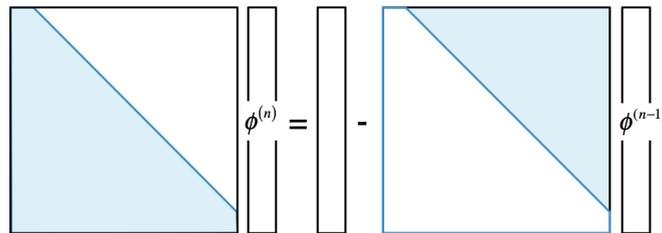
$$\mathbf{A}\phi = \mathbf{b} \quad (3.21)$$

dove  $\mathbf{A}$  è la matrice dei coefficienti ottenuti come risultato dal processo di linearizzazione del flusso. Essi dipendono dallo schema di discretizzazione adottato.  $\phi$  è il vettore delle incognite, contiene il valore del generico scalare  $\phi$  nei centroidi di tutte le celle del dominio computazionale.  $\mathbf{b}$  è il vettore dei termini noti dove sono presenti i termini sorgente, le condizioni di bordo e i termini non lineari del flusso. Esistono molti metodi per risolvere un sistema lineare di equazioni algebriche (Fig.3.22) e questi si suddividono in due grandi classi: i metodi diretti ed i metodi iterativi. Per applicazioni di CFD i metodi iterativi sono preferiti in quanto richiedono un minore costo computazionale ed una minore allocazione di memoria [32].

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N-1} & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N-1} & a_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-11} & a_{N-12} & \cdots & a_{N-1N-1} & a_{N-1N} \\ a_{N1} & a_{N2} & \cdots & a_{NN-1} & a_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix} \quad (3.22)$$

Un metodo iterativo particolarmente adottato in CFD per risolvere Eq.(3.21) è il metodo di Gauss-Seidel, un metodo simile a quello di Jacobi ma più efficace in termini di convergenza e consumo di memoria. Il metodo Gauss-Seidel è implementato secondo la seguente formula iterativa:

$$\phi_i^{(n)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{(n)} - \sum_{j=i+1}^N a_{ij}\phi_j^{(n-1)} \right) \quad i = 1, 2, 3, \dots, N \quad (3.23)$$



**Figura 3.3:** Rappresentazione grafica del metodo di Gauss-Siedel [32].

### Metodi iterativi preconditionati

La velocità di convergenza dei metodi iterativi dipende fortemente da una proprietà della matrice di iterazione  $\mathbf{B}$  detta raggio spettrale  $\varrho$ . Esso indica il massimo autovalore della matrice  $\mathbf{B}$ .

$$\varrho(\mathbf{B}) = \max_{i=1}^N(\lambda_i) \quad (3.24)$$

E' possibile dimostrare [32] che un metodo iterativo è convergente soltanto se  $\varrho < 1$ . Inoltre minore è il raggio spettrale, maggiore è la velocità di convergenza del metodo iterativo alla soluzione.

Talvolta è possibile migliorare le prestazioni del processo iterativo trasformando il sistema di equazioni algebriche (Fig.3.22) in un sistema equivalente con migliori proprietà spettrali. Si definisce matrice di preconditionamento  $\mathbf{P}$  una matrice capace di applicare tale trasformazione:

$$\mathbf{P}^{-1}\mathbf{A}\phi = \mathbf{P}^{-1}\mathbf{b} \quad (3.25)$$

In tal caso  $\varrho(\mathbf{P}^{-1}\mathbf{A}) < \varrho(\mathbf{A})$  ed il sistema convergerà più velocemente. E' possibile dimostrare che la matrice di preconditionamento per il metodo di Gauss-Seidel è:

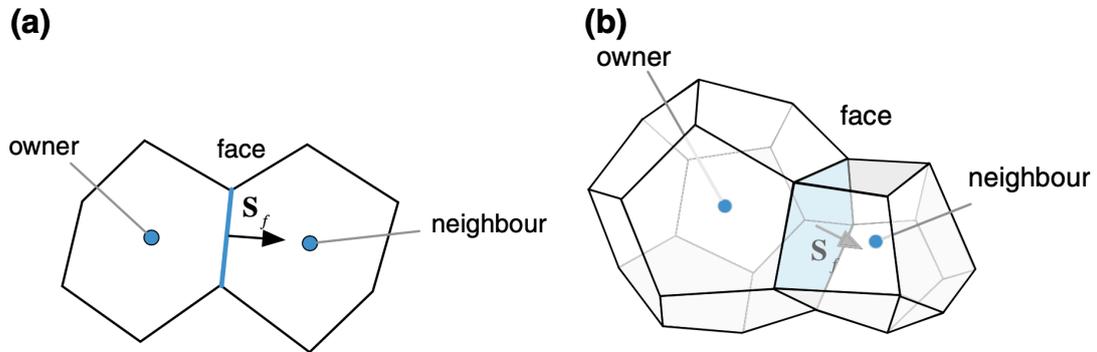
$$\mathbf{P}_{GS} = \mathbf{D} + \mathbf{L} \quad (3.26)$$

dove  $\mathbf{D}$  e  $\mathbf{L}$  sono rispettivamente le matrici diagonale e triangolare inferiore della matrice  $\mathbf{A}$ .  $\mathbf{P}_{GS}$  non è l'unica matrice di preconditionamento possibile, esistono infatti anche altre matrici i cui coefficienti sono definiti in modi più complicati.

## 3.2 The Finite Volume Mesh

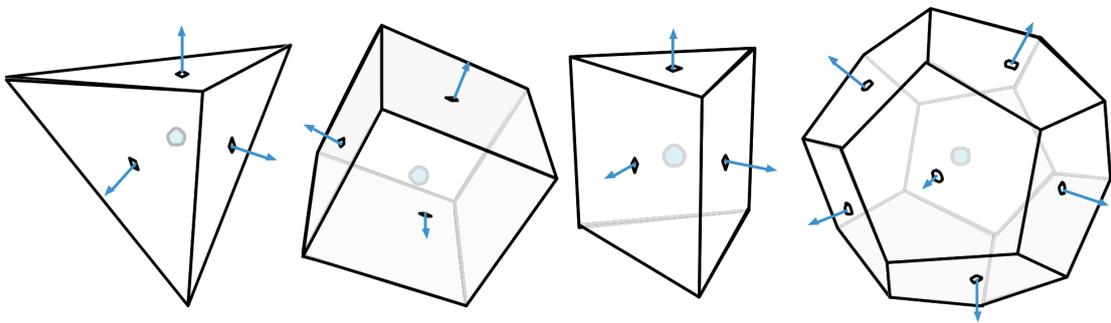
Parallelamente alla discretizzazione delle equazioni di governo il FVM prevede anche la definizione di una *mesh* computazionale. La *mesh* consiste in un insieme di celle e rappresenta il passaggio di discretizzazione di un dominio continuo in una serie finita di elementi (i.e. celle), caratterizzate da un numero finito di facce interconnesse tra di loro o di bordo. Il processo di meshing della geometria non solo decompone il sistema in un numero finito di celle ma ne calcola importanti proprietà e determina le relazioni topologiche tra i vari elementi della mesh. Il risultato è un insieme finito di celle, non sovrapposte, ben caratterizzate topologicamente, comunicanti mediante le loro facce con le celle adiacenti e con il bordo del dominio. Le equazioni discretizzate saranno definite e risolte su ogni elemento della mesh. Ogni cella è caratterizzata dalla presenza di un centroide (punto centrale) nel quale vengono solitamente memorizzate le informazioni circa i campi scalari e vettoriali definiti su quella cella. Il centroide così come il volume della cella e la superficie di

ogni sua faccia vengono calcolati e memorizzati in modo da essere successivamente utilizzati durante il calcolo dei flussi volumici e superficiali Eqs.(3.11 e 3.13).



**Figura 3.4:** Topologia delle celle: owners, neighbors, faces [33].

Le celle della mesh solitamente sono poliedri in un dominio tridimensionale o poligoni in uno spazio bidimensionale, un esempio è raffigurato in Fig.(3.5). La griglia è solitamente classificata come strutturata o non strutturata in base alla possibilità di numerare sequenzialmente, secondo logica (cartesiana o di altro tipo), le celle a partire da un determinato punto fino ad includerle tutte. I *tools* del software OpenFOAM sono in grado di costruire griglie non strutturate, poliedriche.



**Figura 3.5:** Tipi di elementi in mesh tridimensionali [33].

### 3.3 Discretizzazione spaziale

Dopo aver costruito la mesh computazionale ed aver definito le equazioni semi-discretizzate Eq.(3.10) è necessario procedere convertendo tali equazioni in equazioni algebriche per essere infine risolte, sotto forma di sistema Eq.(3.23). Per ogni termine dell'equazione di trasporto è necessario specificare l'approccio utilizzato per discretizzare il laplaciano ed il gradiente nello spazio. Per tale motivo i coefficienti definiti nell'Eq.(3.14) dipendono dal termine che viene discretizzato e dall'approccio utilizzato. Una breve panoramica sulla discretizzazione spaziale di tali termini sarà fornita in questa sezione. Per semplicità sarà considerata la griglia bidimensionale, cartesiana mostrata in Fig.(3.6).

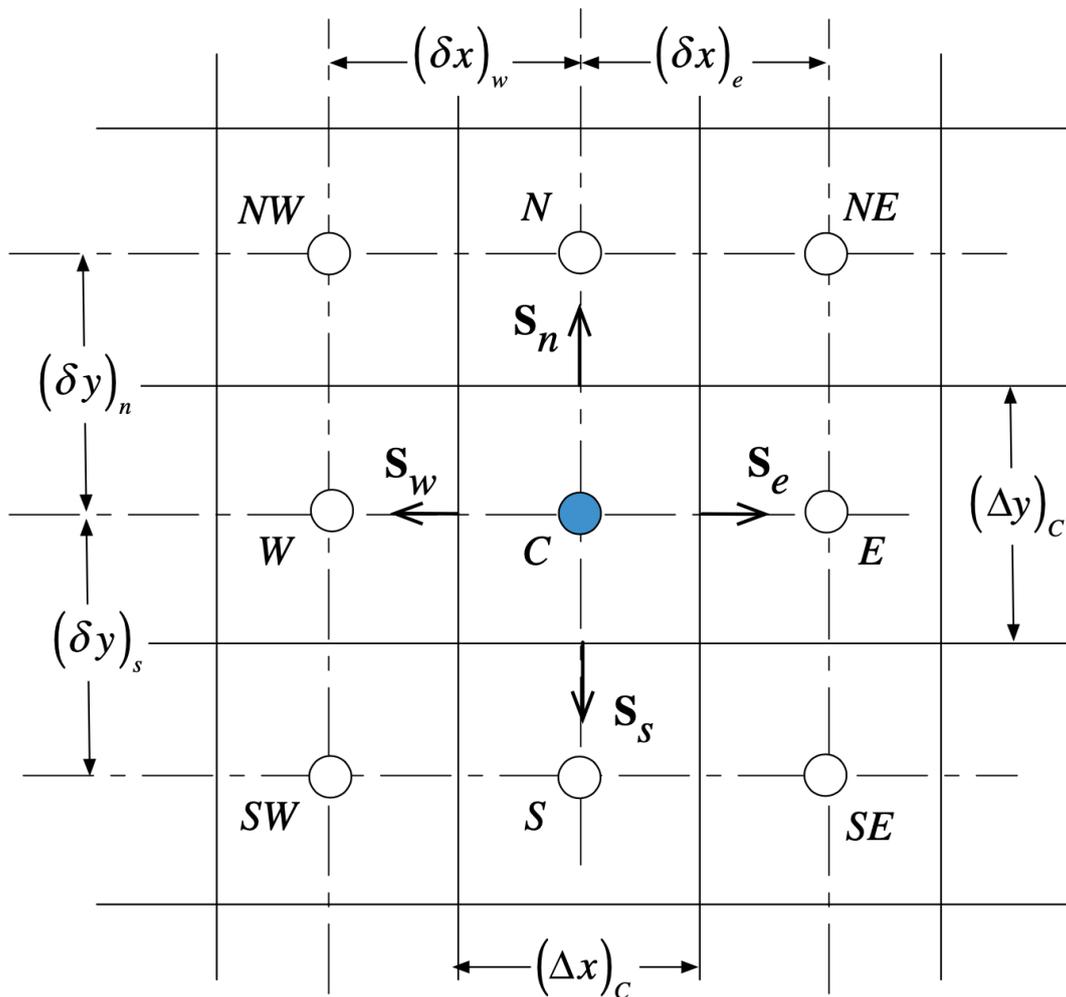


Figura 3.6: Griglia cartesiana uniforme [34].

### 3.3.1 Termine diffusivo

Il termine diffusivo

$$-\nabla \cdot (\Gamma^\phi \nabla \phi) \quad (3.27)$$

contiene il laplaciano della generica proprietà  $\phi$  e, seguendo il primo stadio di discretizzazione descritto nel capitolo 3.1.1, può essere formulato come:

$$\sum_{f \sim nb(C)} (-\Gamma^\phi \nabla \phi)_f \cdot \mathbf{S}_f \quad (3.28)$$

Per la cella C nella Fig.(3.6), l'equazione precedente può essere espansa come:

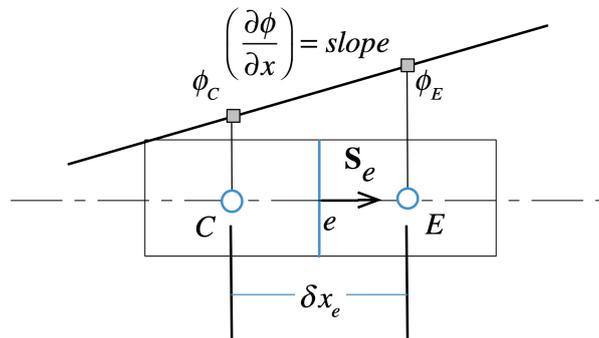
$$(-\Gamma^\phi \nabla \phi)_e \cdot \mathbf{S}_e + (-\Gamma^\phi \nabla \phi)_w \cdot \mathbf{S}_w + (-\Gamma^\phi \nabla \phi)_n \cdot \mathbf{S}_n + (-\Gamma^\phi \nabla \phi)_s \cdot \mathbf{S}_s \quad (3.29)$$

ovvero la somma dei flussi attraverso rispettivamente le facce est, ovest, nord, sud adiacenti alla cella C. I singoli flussi possono essere calcolati espandendo la definizione del gradiente; per esempio nella faccia est:

$$J_e^{\phi,D} = -(\Gamma^\phi \nabla \phi)_e \cdot \mathbf{S}_e = -\Gamma_e^\phi S_e \left( \frac{\partial \phi}{\partial x} \mathbf{i} + \frac{\partial \phi}{\partial y} \mathbf{j} \right) \cdot \mathbf{i} = -\Gamma_e^\phi (\Delta y)_e \left( \frac{\partial \phi}{\partial x} \right)_e \quad (3.30)$$

Per completare il calcolo del flusso dobbiamo imporre un profilo di  $\phi$  tra i valori dei centroidi delle delle adiacenti (in questo caso le celle E, A) in modo da poter calcolare il gradiente. Il profilo utilizzato nel termine diffusivo è lineare Fig.(3.7).

$$\left( \frac{\partial \phi}{\partial x} \right)_e = \frac{\phi_E - \phi_C}{(\delta x)_e} \quad (3.31)$$



**Figura 3.7:** Profilo di interpolazione alla faccia  $e$  [34].

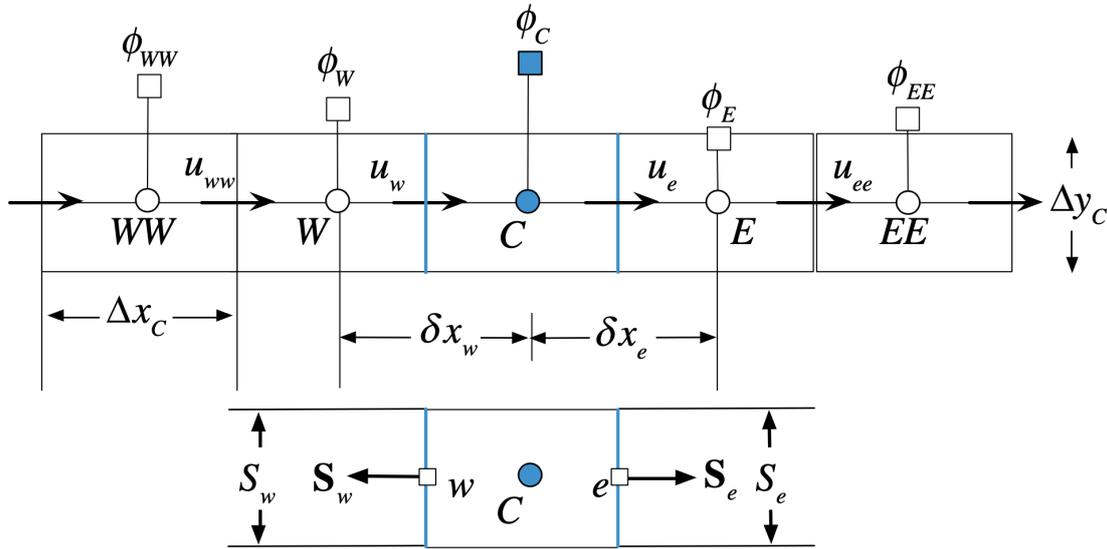
Infine sostituendo Eq.(3.31) in Eq.(3.30) si ottiene la completa discretizzazione del termine diffusivo, che può essere successivamente decomposto nei contributi descritti nella sezione 3.1.2.

$$FluxT_e = -\Gamma_e^\phi(\Delta y)_e \frac{(\phi_E - \phi_C)}{\delta x_e} = \Gamma_e^\phi \frac{(\Delta y)_e}{\delta x_e} (\phi_C - \phi_E) \quad (3.32)$$

dove  $FluxT_e = FluxC_e\phi_C + FluxF_e\phi_E + FluxV_e$ , ovvero un contributo proporzionale a  $\phi_C$ , un contributo proporzionale a  $\phi_E$  ed un contributo non lineare.

### 3.3.2 Termine convettivo

La discretizzazione del termine convettivo è più complessa rispetto alla precedente per via della elevata non linearità che questo può introdurre nell'equazione di trasporto. Il tema sarà trattato facendo riferimento ad un generico problema di advezione-diffusione, monodimensionale, con griglia cartesiana (Fig.3.8).



**Figura 3.8:** Griglia monodimensionale nel problema advezione-diffusione [35].

L'equazione di trasporto che descrive il problema di advezione-diffusione è così definita:

$$\int_{V_C} [\nabla \cdot (\rho \mathbf{U} \phi) - \nabla \cdot (\Gamma^\phi \nabla \phi)] dV = 0 \quad (3.33)$$

dove  $\mathbf{U} = U\mathbf{i}$ .

Applicando il teorema di Gauss ad Eq.(3.33) e sostituendo l'integrale superficiale con la somma dei flussi attraverso le facce si ottiene:

$$\sum_{f \sim nb(C)} \left( \rho U \phi \mathbf{i} - \Gamma \phi \frac{d\phi}{dx} \mathbf{i} \right)_f \cdot \mathbf{S}_f = 0 \quad (3.34)$$

Ed espandendo la somma sulle due uniche facce adiacenti (problema monodimensionale) si ottiene:

$$\left[ (\rho U \Delta y \phi)_e - \left( \Gamma \phi \frac{d\phi}{dx} \Delta y \right)_e \right] - \left[ (\rho U \Delta y \phi)_w - \left( \Gamma \phi \frac{d\phi}{dx} \Delta y \right)_w \right] = 0 \quad (3.35)$$

In questa equazione le proprietà del fluido sono note, il valore della velocità alle facce delle celle è noto, il gradiente di  $\phi$  può essere discretizzato come precedentemente trattato in Eq.(3.31). L'unica decisione da prendere è come discretizzare il valore di  $\phi$  in corrispondenza delle facce  $e, w$  rispetto ai valori presenti nei centroidi delle celle adiacenti  $\phi_E, \phi_C, \phi_W$ . Diverse decisioni introducono diversi schemi di discretizzazione che, in letteratura, sono noti come schemi di advezione.

Molti schemi di discretizzazione sono studiati in letteratura; in questo lavoro verranno utilizzati schemi di primo e secondo ordine pertanto di seguito sono presentati alcuni cenni in merito agli schemi *upwind*.

### Schemi upwind

I processi convettivi sono caratterizzati dalla presenza di una direzione (i.e. moto del fluido) che non è invece presente nei processi di pura diffusione. In tali condizioni un'approccio migliore rispetto all'interpolazione lineare precedentemente utilizzata per trattare il problema di pura diffusione è dato dai metodi *upwind*. Il valore di  $\phi$  sulla faccia è funzione del valore memorizzato nel centroide delle celle che precedono il moto del fluido come mostrato nella Fig.(3.9).

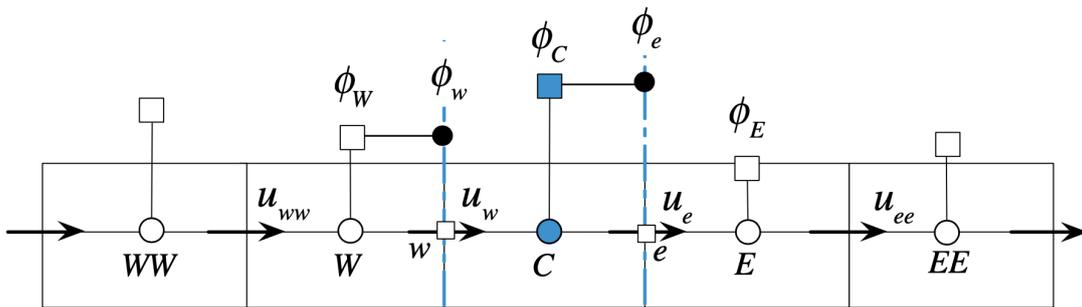


Figura 3.9: Termine convettivo: schema upwind [35].

Di conseguenza è possibile definire il valore della generica proprietà  $\phi$  sulle facce di  $C$  come:

$$\phi_e = \begin{cases} \phi_C & \text{if } \dot{m}_e > 0 \\ \phi_E & \text{if } \dot{m}_e < 0 \end{cases} \quad \text{and} \quad \phi_w = \begin{cases} \phi_C & \text{if } \dot{m}_w > 0 \\ \phi_W & \text{if } \dot{m}_w < 0 \end{cases} \quad (3.36)$$

dove  $\dot{m}_e$  e  $\dot{m}_w$  sono le portate massiche alle facce  $e, w$ . Esse sono calcolate con le seguenti realazioni:

$$\begin{aligned} \dot{m}_e &= (\rho \mathbf{U} \cdot \mathbf{S})_e = (\rho U S)_e = (\rho U \Delta y)_e \\ \dot{m}_w &= -(\rho \mathbf{U} \cdot \mathbf{S})_w = -(\rho U S)_w = -(\rho U \Delta y)_w \end{aligned} \quad (3.37)$$

Pertanto il flusso convettivo alla faccia  $e$  può essere riscritto come:

$$\begin{aligned} \dot{m}_e \phi_e &= \|\dot{m}_e, 0\| \phi_C - \|\dot{m}_e, 0\| \phi_E \\ &= FluxC_e^{Conv} \phi_C + FluxF_e^{Conv} \phi_E + FluxV_e^{Conv} \end{aligned} \quad (3.38)$$

dove l'operatore  $\|a, b\|$  calcola il massimo tra  $a, b$ .

Applicando uno schema di discretizzazione lineare al termine diffusivo ed uno schema upwind del primo ordine al termine convettivo (Eqs. 3.32 e 3.38) nel problema di advezione-diffusione (Eq.3.33) otteniamo la seguente equazione algebrica:

$$\begin{aligned} & (FluxC_e^{Conv} + FluxC_e^{Diff} + FluxC_w^{Conv} + FluxC_w^{Diff}) \phi_C + \\ & (FluxF_e^{Conv} + FluxF_e^{Diff}) \phi_E + \\ & (FluxF_w^{Conv} + FluxF_w^{Diff}) \phi_W = 0 \end{aligned} \quad (3.39)$$

che può essere rimaneggiata ed espressa nella seguente forma canonica:

$$a_C \phi_C + a_E \phi_E + a_W \phi_W = 0 \quad (3.40)$$

E' possibile dimostrare che  $a_C = -(a_W + a_E)$  e questo garantisce allo schema upwind del primo ordine l'importate proprietà della *limitatezza*.

### Altri schemi di discretizzazione

Molti altri schemi di discretizzazione sono disponibili, essi si caratterizzano in base a proprietà computazionali come la limitatezza e la stabilità ma anche in base all'accuratezza della soluzione. Metodi del secondo ordine come il "central differencing scheme" forniscono una migliore accuratezza alla soluzione, abbattendo

l'errore di iterazione in iterazione ad una maggiore velocità rispetto ai metodi del primo ordine. Tuttavia alcuni metodi di alto ordine come il CDS sono affetti da problemi di stabilità e limitatezza; in altre parole non garantiscono una convergenza uniforme alla soluzione e possono introdurre errori numerici durante il processo risolutivo quando il fenomeno convettivo è molto intenso (i.e. elevati valori del  $Pe_{cella}$ ). Altri popolari schemi di ordine superiore sono il "second order upwind" ed il metodo QUICK, che fornisce una accuratezza del terzo ordine [35]. La scelta dello schema da adottare è parte della strategia risolutiva del problema di CFD e talvolta è possibile susseguire l'uso di più schemi di discretizzazione spaziale al fine di migliorare le performance in termini di velocità di convergenza, stabilità dell'algoritmo e costo computazionale.

$$Pe_{cella} = \frac{\rho U \delta_x}{\Gamma \phi} \quad (3.41)$$

Misura il rapporto tra la velocità del trasporto convettivo e la velocità del trasporto diffusivo. Spesso gli schemi di alto ordine sono stabili e limitati solo per determinati range di  $Pe_{cella}$ , per esempio nel caso del CDS la stabilità è garantita da  $Pe_{cella} < 2$ .

### 3.4 Discretizzazione temporale

La discretizzazione temporale, utilizzata dai solver transitori, può essere descritta procedendo in modo simile a quanto fatto nella discretizzazione del termine convettivo. E' possibile riscrivere la generica equazione di trasporto (Eq.3.1) evidenziando un primo operatore temporale ed un secondo operatore spaziale, il quale contiene il termine diffusivo, convettivo e sorgente:

$$\frac{\partial(\rho\phi)}{\partial t} + \mathcal{L}(\phi) = 0 \quad (3.42)$$

L'equazione può essere integrata sulla cella C ottenendo:

$$\begin{aligned} \int_{V_C} \frac{\partial(\rho\phi)}{\partial t} dV + \int_{V_C} \mathcal{L}(\phi) dV &= 0 \\ \frac{\partial(\rho\phi)}{\partial t} V_C + \mathcal{L}(\phi_C^t) &= 0 \end{aligned} \quad (3.43)$$

L'operatore di discretizzazione spaziale può essere scritto in forma algebrica come:

$$\mathcal{L}(\phi_C^t) = a_C \phi_C^t + \sum_{F \sim NB(C)} a_F \phi_F^t - b_C \quad (3.44)$$

dove  $\phi_C^t$  e  $\phi_F^t$  sono i coefficienti di discretizzazione spaziale al tempo  $t$ . Integrando Eq.(3.43) nel tempo

$$\int_{t-\Delta t/2}^{t+\Delta t/2} \frac{\partial(\rho\phi_C)}{\partial t} V_C dt + \int_{t-\Delta t/2}^{t+\Delta t/2} \mathcal{L}(\phi_C) dt = 0 \quad (3.45)$$

si ottiene la forma semi-discreta dell'equazione di trasporto transitoria:

$$\frac{(\rho\phi_C)^{t+\Delta t/2} - (\rho\phi_C)^{t-\Delta t/2}}{\Delta t} V_C + \mathcal{L}(\phi_C^t) = 0 \quad (3.46)$$

L'equazione è completamente discretizzata scegliendo un profilo di interpolazione per esprimere i valori della proprietà  $\phi$  in  $(t - \Delta t/2)$  e  $(t + \Delta t/2)$  in funzione dei valori al tempo  $(t)$  e  $(t - \Delta t)$ . Esiste una grande varietà di metodi di interpolazione trattati in letteratura ma nel presente lavoro è stato adottato il "metodo di Eulero implicito". E' un metodo del primo ordine, incondizionatamente stabile. Tale metodo prevede

$$(\rho\phi_C)^{(t+\Delta t/2)} = (\rho\phi_C)^t \quad \text{and} \quad (\rho\phi_C)^{(t-\Delta t/2)} = (\rho\phi_C)^{t-\Delta t} \quad (3.47)$$

che applicato ad Eq.(3.45) restituisce la forma completamente discretizzata dell'equazione di trasporto transitoria.

$$\frac{(\rho\phi_C)^t - (\rho\phi_C)^{t-\Delta t}}{\Delta t} V_C + \mathcal{L}(\phi_C^t) = 0 \quad (3.48)$$

## 3.5 OpenFoam tools

**Openfoam** [36] è un potente strumento open-source, disponibile sotto libera licenza, basato su codice **C++** che permette di costruire griglie computazionali e risolvere le equazioni di trasporto mediante solver già implementati nel software. **Openfoam** mette a disposizione anche diversi *tools* utili nel preprocessing e nel postprocessing della CFD. In questo lavoro ne sono stati impiegati alcuni e questo capitolo ne renderà una breve descrizione. Infine saranno discussi i solver adottati per la risoluzione numerica delle equazioni di trasporto.

### 3.5.1 Impostazione foamCases

**Openfoam** è un framework basato su cartelle organizzate, ogni simulazione (i.e. *running case*) è composta da una *foamCase*, Fig.(3.10), con una serie di sottocartelle strutturata nel seguente modo:

- **0**: Rappresenta le condizioni iniziali della simulazione. Contiene una serie di files nei quali sono descritte le condizioni iniziali e di bordo per i vari campi da risolvere, nel caso più semplice il campo di flusso.
- **constant**: Contiene al suo interno i files che definiscono le proprietà del fluido ed il tipo di simulazione (laminar, RANS, LES). Al suo interno, durante il processo di meshing della geometria (sezione 3.2), viene generata la cartella `polyMesh` nella quale sono contenuti tutti i file di topologia della mesh tra cui `boundary`, `faces`, `neighbour`, `owner`, `points`.
- **system**: Contiene i files che definiscono il tipo di solver impiegato, gli schemi di discretizzazione adottati, eventuali coefficienti di sottorilassamento e tutte le specifiche relative alle iterazioni del solver. I files essenziali sono `controlDict`, `fvScheme`, `fvSolution`. Sono presenti anche i files di definizione della mesh computazionale `blockMeshDict` e `snappyHexMeshDict` nei quali vengono definite le regioni su cui costruire la mesh, il livello di discretizzazione desiderato e le soglie per le variabili di controllo della qualità della griglia.
- **time\_directories**: Durante il processo iterativo di risoluzione delle equazioni di trasporto sarà possibile salvare la soluzione per diversi valori temporali (simulazioni transitorie) o soltanto raggiunta la convergenza (simulazioni stazionarie).

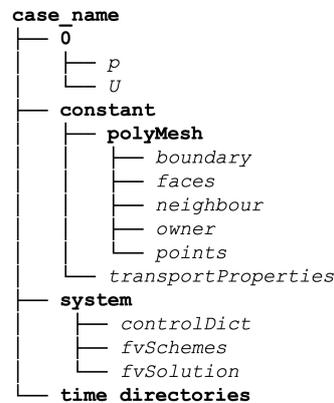


Figura 3.10: Struttura di una generica foamCase.

### 3.5.2 Meshing tools

Gli strumenti utilizzati nell'ambito della definizione della mesh computazionale sono tre, di cui uno dedicato al controllo della qualità della griglia.

- **blockMesh**: E' il tool responsabile della definizione della mesh cubica di background, richiede la definizione delle coordinate dei vertici della mesh tridimensionale e le facce. Il parametro chiave è  $BM_{cells}$  che rappresenta il numero di celle per lato della bounding box cubica.
- **snappyHexMesh**: Questo tool è responsabile del posizionamento della schiuma (i.e. file `.stl`) al centro della mesh di background generata precedentemente. In seguito l'algoritmo ricorsivo di **snappyHexMesh** evidenzia i bordi (i.e. features) della geometria e suddivide le celle attraversate dai bordi in un processo che genera da ogni cella otto celle equivolumiche. La procedura continua fino al grado di discretizzazione desiderato. Infine l'utente seleziona un punto appartenente alla regione in cui si vuole creare la griglia computazionale. Il risultato sarà il completo meshing della fase di interesse. Il file di configurazione del tool, **snappyHexMeshDict**, permette di impostare anche molti parametri che gestiscono il grado di raffinamento dei bordi della geometria (Fig.4.2b), le opzioni di *snap* della geometria e gli eventuali layers superficiali. La fase di snapping è condotta a carico di un secondo algoritmo ricorsivo che cerca di smussare i bordi della geometria permettendo di definire profili realistici per la superficie delle schiume solide.
- **checkMesh**: Il tool permette di analizzare la mesh e stilare un report nel quale figurano dei parametri che ne valutano la qualità come la *skewness*, l'ortogonalità delle celle ed altri. Se la mesh supera determinati limiti viene giudicata inadatta in quanto i risultati possono essere affetti da errore o le sue caratteristiche sono tali da far fallire il solver.

### 3.5.3 Solvers

**Openfoam** mette a disposizione degli utenti dei solver già strutturati e configurati per determinate applicazioni. In questo lavoro le simulazioni del campo di moto sono state eseguite con il solver **simpleFoam** mentre quelle di trasporto del calore sono state condotte con il solver **scalarTransportFoam**.

#### **simpleFoam solver**

**simpleFoam** è un solver stazionario, per fluidi incomprimibili basato sull'algoritmo SIMPLE (Semi-Implicit Method for Pressure Linked Equations). Sono implementate tre equazioni principali, la prima descrive il trasporto della quantità di moto

Eq.(2.4), la seconda è l'equazione di continuità Eq.(2.1) che però in caso di fluido incomprimibile non fornisce una relazione esplicita velocità-pressione. Applicando la divergenza all'equazione di Navier-Stokes e sostituendo l'equazione di continuità si ottiene la terza equazione che lega il campo di moto al campo di pressione: l'equazione di Poisson (3.49).

$$\frac{\partial}{\partial x_i} \left( \frac{\partial p}{\partial x_i} \right) = - \frac{\partial}{\partial x_i} \left( \frac{\partial(\rho U_i U_j)}{\partial x_j} \right) \quad (3.49)$$

Inizialmente è risolto il campo di moto utilizzando Eq.(2.4) essendo noto il campo di pressione dall'iterazione precedente  $p^n$ . Così facendo si ottiene un campo di velocità  $\mathbf{U}^*$  che non soddisfa l'equazione di continuità. A questo punto è possibile utilizzare l'equazione di Poisson per calcolare un nuovo campo di pressione  $p^{n+1}$ . L'algoritmo si ripete finché non si riscontra convergenza tra  $p^n$  e  $p^{n+1}$ .

### scalarTransportFoam solver

`scalarTransportFoam` è un solver transitorio che risolve l'equazione di trasporto Eq.(3.1) per un generico scalare non reattivo, noto un campo di velocità imposto dall'utente. Nel presente lavoro `scalarTransportFoam` è stato impiegato per studiare il trasporto dello scalare  $T$  (i.e. temperatura) Eq.(2.6), in presenza del campo di moto ottenuto come soluzione del solver `simpleFoam` sulla medesima geometria.

### 3.5.4 Postprocessing

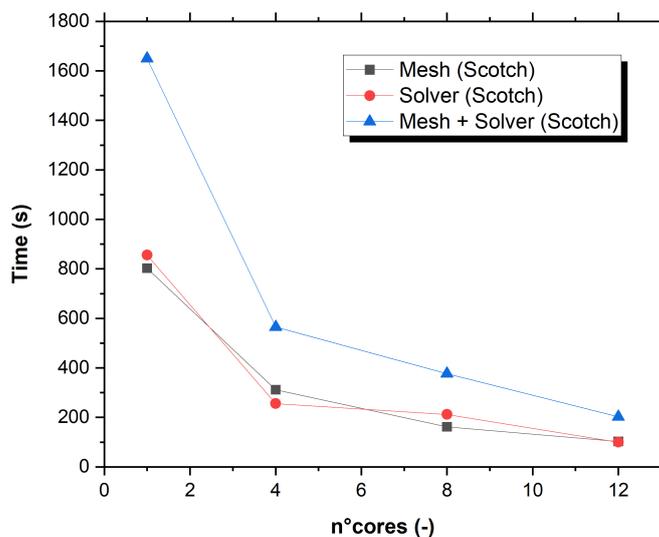
Molti strumenti sono disponibili in `Openfoam` per eseguire operazioni di postprocessing sui risultati ottenuti dalle simulazioni; gli strumenti utilizzati, ai fini di questo lavoro, sono i seguenti:

- `flowRatePatch`: Questo tool permette il calcolo della portata volumica che attraversa una patch. E' stato impiegato per valutare la portata  $\dot{q}$  che fluisce nella geometria in presenza di un gradiente di pressione noto.
- `patchAverage`: Il tool permette il calcolo del valore medio di un campo su di una patch. E' stato impiegato per valutare la temperatura media  $T_{out}$  in corrispondenza dell'outlet.
- `volFieldValue`: E' un tool che permette di calcolare il valore medio di un campo su di un volume. E' stato impiegato per calcolare il valore medio di velocità del fluido  $U_{avg}$ .

Durante il postprocessing sono estratte anche altre informazioni dai risultati grezzi della simulazione tra cui il volume di fluido  $V_{fluid}$  dal report di `checkMesh` e la superficie della schiuma  $S_{foam}$ . Estratte queste informazioni dalla simulazione e note le condizioni al contorno e iniziali come  $L_{REV}$ ,  $\Delta\mathcal{P}$ ,  $T_{in}$  e le proprietà del fluido è possibile calcolare tutte le quantità discusse nel Capitolo 2.

### 3.6 Cenni HPC

Lo studio della fluidodinamica attraverso simulazioni numeriche permette il risparmio di ingenti risorse connesse all'attività sperimentale e per tale ragione il suo impiego è in crescita in moltissimi campi ingegneristici, tra cui lo studio delle microstrutture e le loro interazioni con un flusso fluido. L'accuratezza di tali simulazioni numeriche è determinata, oltre che dall'impostazione degli algoritmi numerici, anche dall'intensità di discretizzazione del dominio in quanto in presenza di simulazioni stabili l'errore numerico tende a zero per un numero di celle tendente ad infinito (i.e. dominio continuo). La capacità di calcolo necessaria per gestire una tale discretizzazione del dominio, in questo lavoro, è stata fornita da HPC@Polito [37]. Tutte le simulazioni condotte nell'ambito di questo lavoro si sono svolte sul cluster di ateneo del Politecnico di Torino: `legion@polito.it`



**Figura 3.11:** Tempo computazionale di meshing e solver per una generica geometria parallelizzata su più *cores*.

Un secondo vantaggio dei sistemi HPC consiste nella possibilità di parallelizzare il lavoro su più processori, in modo tale da svolgere più attività in parallelo ed abbattere significativamente i tempi computazionali. Una semplice analisi preliminare

ha mostrato come un sistema parallelizzato su *12-cores* possa gestire il meshing di una generica geometria OCF e la risoluzione della corrispondente simulazione di flusso in un tempo fino a dieci volte inferiore rispetto alle medesime tasks gestite da un sistema *single-core* Fig.(3.11). In questo lavoro tutte le simulazioni sono state condotte imponendo una parallelizzazione su 12-cores per le fasi di meshing (`snappyHexMesh`) e di risoluzione delle equazioni di governo (`simpleFoam` e/o `scalarTransportFoam`).



## Parte III

# Impostazione Workflow e Dettagli Numerici



---

La terza parte di questo lavoro sarà dedicata alla descrizione qualitativa e quantitativa degli studi condotti sulle geometrie di OCF. Il capitolo 4 introduce l'impostazione delle simulazioni di flusso e lo studio del REV, ovvero il volume elementare rappresentativo (sezioni 4.1 - 4.2). Successivamente, nella sezione 4.3, un'analisi preliminare vedrà lo studio di due mini dataset di trenta geometrie, ottenuti variando randomicamente tre e cinque dei parametri geometrici (Tab.2.1) in ingresso all'algoritmo basato su Plugim.

Completato lo studio preliminare delle geometrie, il target successivo è stato quello di sviluppare un algoritmo (i.e. `autoWorkflow`) capace di costruire autonomamente un dataset di dimensione  $\mathcal{N}$ , dall'immissione degli input all'ottenimento delle  $\mathcal{N}$  foamCases, pronte per essere risolte. Tale strumento ha reso possibile la generazione di un grande numero di geometrie e foamCases in completa autonomia rispetto all'utente. `autoWorkflow` ha permesso, nell'ambito di questo lavoro, di sviluppare rapidamente le più di 400 foamCases contenenti altrettante geometrie generate dal modello geometrico (sezione 2.1), risolverle sul cluster HPC ed effettuare il *postprocessing* dei risultati. L'algoritmo ed il suo funzionamento sono discussi nella sezione 4.4.1. La trascrizione completa del codice è riportata nell'allegato B.

Reso ora possibile uno studio esteso, nella sezione 4.4, è stato impostato un dataset di 300 geometrie di OCF allo scopo di raccogliere i risultati derivanti dall'analisi CFD ed allenare una rete neurale MLP, ottenendo così un modello surrogato capace di predire la permeabilità (sezione 2.3), a partire dai parametri del modello geometrico (Tab.2.1).

Nella sezione 4.5 è dettagliata la MLPL (Machine Learning Pipeline), ovvero l'insieme delle fasi di pre e post processing del dataset; necessarie per una buona riuscita nella costruzione del modello di regressione.

Conclusi gli studi sul campo di moto nelle MLPL si è desiderato introdurre una seconda fisica nel sistema, lo scambio termico. Un semplice modello di scambio termico è definito nel capitolo 5 per un mini-dataset di geometrie allo scopo di indagare la migliore microstruttura, capace di massimizzare scambio termico (Eq.2.14) e minimizzare le perdite di carico (Eq.2.12).

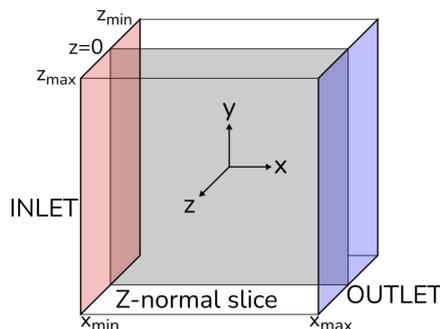


# Capitolo 4

## Simulazioni di flusso

### 4.1 Setup delle simulazioni

Le simulazioni numeriche sono state effettuate utilizzando il codice CFD open source `OpenFOAM 8`. Il dominio computazionale utilizzato in tutte le simulazioni ha forma cubica di lato  $L_{REV}$ , dove tale grandezza identifica la dimensione lineare del volume rappresentativo. Lo studio del REV è affrontato nella sezione 4.2.



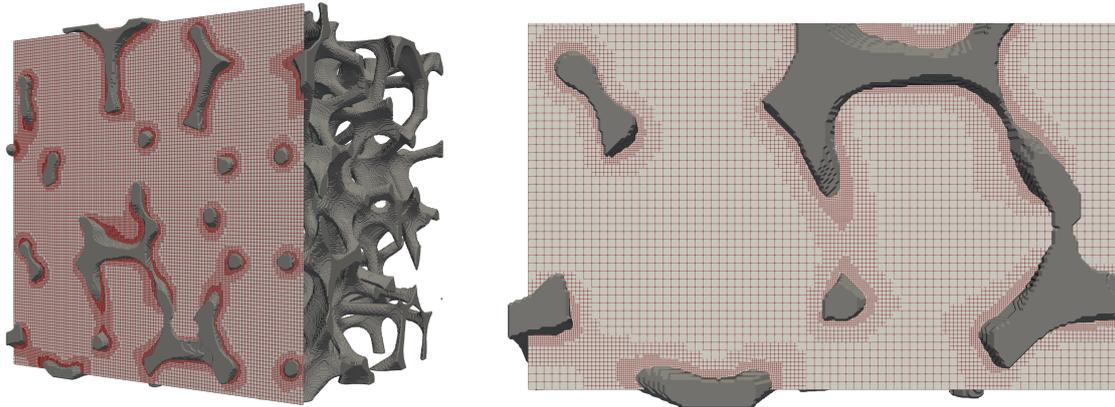
**Figura 4.1:** Bounding box simulata in `OpenFOAM`. Sono evidenziate le patch di inlet ed outlet del fluido [8].

Il flusso di fluido è inizializzato imponendo un gradiente di pressione  $\Delta\mathcal{P} = \frac{\Delta P}{\rho}$  lungo l'asse  $x$ , tra le patch di inlet ed outlet. Sulla superficie della schiuma è applicata la condizione di `zeroGradient` per la pressione mentre le restanti patch presentano condizioni `symmetry`. Il numero di Reynolds adottato per questo particolare setup è definito in Eq.(2.9). Le condizioni di `noSlip` per la velocità  $U$  è applicata alla superficie della schiuma, le patch di inlet ed outlet sono caratterizzate dalla condizione `pressureInletOutletVelocity`, utile della risoluzione di problemi con flussi *pressure-driven*. Alle rimanenti patch sono assegnate le condizioni

*symmetry*, che implicano nessun movimento di fluido attraverso di esse. Il fluido incomprimibile considerato è acqua, caratterizzato da una densità di  $997 \text{ kg m}^{-3}$  ed una viscosità cinematica di  $0.89 \times 10^{-6} \text{ m}^2 \text{ s}^{-1}$ . Le simulazioni di flusso sono risolte utilizzando il solver `simpleFoam`, allo stato stazionario, in condizioni di moto laminare (Eq.2.8).

#### 4.1.1 Grid Independency

Ogni applicazione CFD richiede inizialmente uno studio di *grid independency*, ovvero uno studio di come variano i risultati incrementando la discretizzazione spaziale. L'obiettivo è identificare il livello di discretizzazione della mesh sufficiente a garantire una soluzione indipendente dalla griglia. La scelta di una mesh più fitta non comporta miglioramenti in termini di accuratezza dei risultati ed aumenta il costo delle simulazioni in termini di tempo e risorse computazionali. La scelta di una griglia troppo lasca comporta invece un minore livello di accuratezza della soluzione. Sono stati investigati due parametri di costruzione della mesh ovvero il numero di blockMesh cells per pore ed il numero di livelli di raffinamento alla parete ( $R_{Layer}$ ).



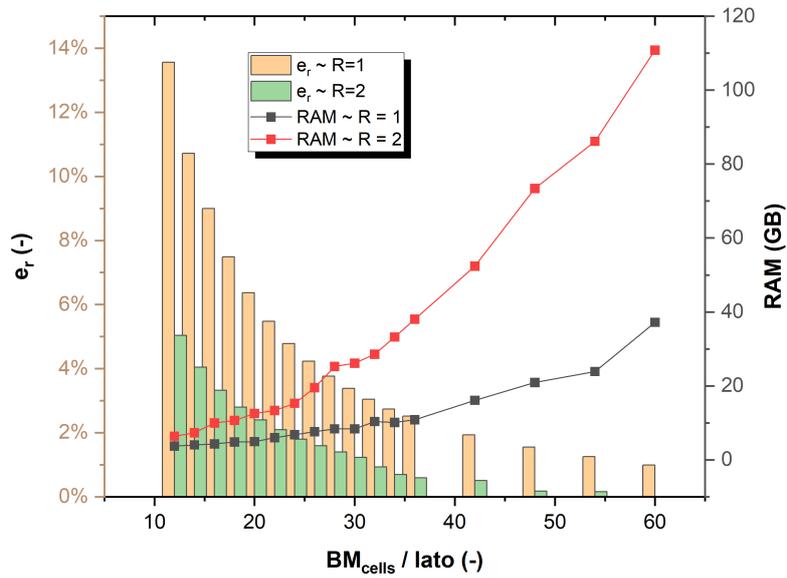
(a) Mesh di una patch (front) del dominio computazionale.

(b) Dettaglio della mesh computazionale.

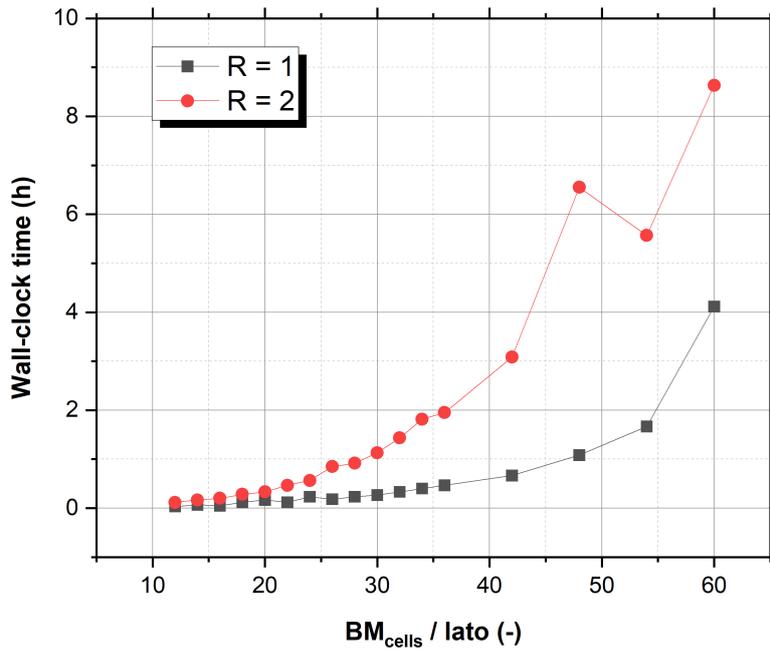
**Figura 4.2:** Porzione di mesh computazionale con  $R_{Layer} = 2$  e  $BM_{cells}/pore = 20$ , costruita su una generica geometria di open cell foam.

Tutte le seguenti simulazioni sono eseguite sulla stessa geometria. Tutte le combinazioni di  $BM_{cells}/pore$  ed  $R_{Layer}$  sono indagate.

$$\begin{aligned} BM_{cells}/pore &= [10, 12, 14, \dots, 32, 34, 36, 42, 48, 54, 60] \\ R_{Layer} &= [1, 2] \end{aligned} \quad (4.1)$$



(a) Grafico di ottimo tra impronta computazionale ed errore relativo sulla permeabilità, calcolato rispetto la simulazione più accurata.



(b) Costo computazionale al variare della risoluzione della griglia.

**Figura 4.3:** Risultati della grid independence analysis.

I risultati evidenziano il classico profilo discendente dell'errore all'aumentare dell'infittimento della griglia computazionale (mesh).

La permeabilità  $\kappa$  decresce rapidamente fino a raggiungere un plateau, lo stesso trend è riportato anche nel grafico dell'errore relativo calcolato rispetto alla simulazione più accurata, con  $R_{Layer} = 2$  e  $BM_{cells}/pore = 60$  (Fig.4.3a). Si osserva che a partire da  $20 - 24 BM_{cells}/pore$  il costo computazionale cresce rapidamente senza significative variazioni di accuratezza dei risultati, pertanto si è deciso di utilizzare per tutte le simulazioni del presente lavoro il seguente setup della mesh:

$$\begin{aligned} BM_{cells}/pore &= 20 \\ R_{Layer} &= 2 \end{aligned} \tag{4.2}$$

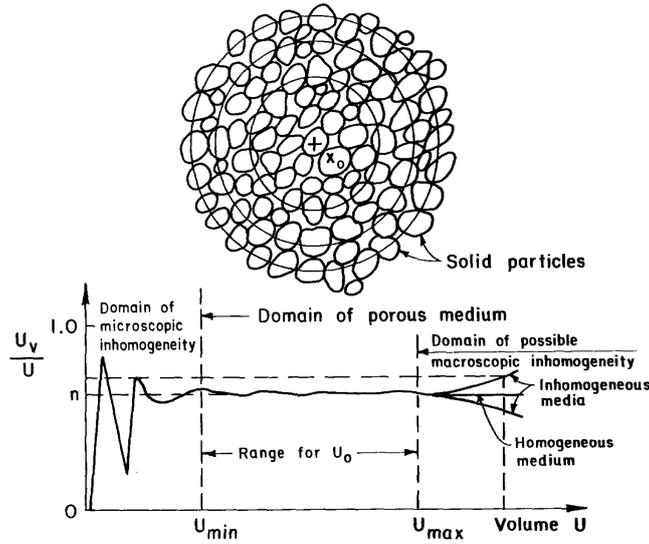
Il wall-clock time (Fig.4.3b) è il tempo impiegato dall'hardware per risolvere completamente il case-study ovvero entrambe le fasi di meshing e di risoluzione delle equazioni mediante il solver `simpleFoam`. Il core-walltime è il tempo effettivo di calcolo speso da tutti i cores impiegati ed essendo le simulazioni parallelizzate su 12-core è più di dieci volte superiore rispetto al wall-clock time.

## 4.2 Analisi del REV

Lo studio della microstruttura dei mezzi porosi è condotto eseguendo simulazioni di fluidodinamica computazionale per determinare numericamente l'evoluzione del campo di moto all'interno di una piccola porzione di schiuma. Minore è la porzione che costituisce il dominio e minore saranno le risorse computazionali allocate durante il processo di risoluzione delle equazioni discretizzate Eqs.(3.12 e 3.43). Tuttavia il dominio computazionale deve essere sufficientemente grande per essere rappresentativo dei macrodescrittori della particolare geometria analizzata come porosità Eq.(1.1), superficie specifica Eq.(2.11), permeabilità Eq.(2.12).

Jacob Bear [38] definisce il REV come la porzione di volume di un mezzo poroso tale per cui i parametri che rappresentano la distribuzione di vuoto e solido nel dominio siano statisticamente significativi. Una definizione simile è utilizzata anche da S. Whitaker in "The method of volume averaging" [39].

In presenza di altri descrittori, come la permeabilità, dovrebbe essere definito un REV per ogni proprietà ed infine scelto il più grande in modo tale da risultare un volume per cui tutte le proprietà e tutti i macrodescrittori della geometria sono statisticamente significativi. Solitamente, nel campo dei mezzi porosi viene identificato un REV basato sulla porosità  $\epsilon$ ; per tale ragione, nel presente lavoro, il REV sarà definito rispetto alla porosità della geometria seguendo i risultati ottenuti da Agostini et al. [8], ovvero considerando come volume rappresentativo una bounding box contenente almeno 4 *foam cells* per lato. La bontà di tale approccio è stata consolidata da simulazioni fluidodinamiche preliminari, dettagliate nella sezione 4.2.2.

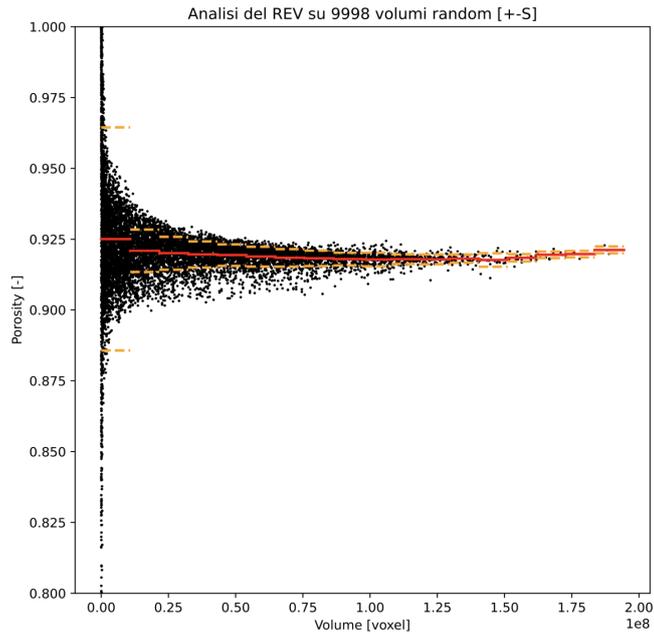


**Figura 4.4:** *Representative elementary volume* in un mezzo poroso [38].

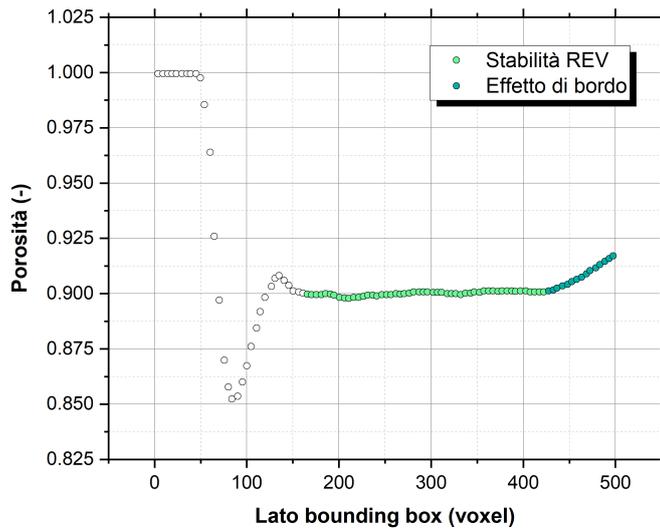
Lo studio del REV sulle singole geometrie è stato condotto utilizzando due approcci differenti:

- (i) **Porespy:** E' una libreria di Python [40] che implementa diverse funzionalità utili nello studio dei mezzi porosi. Lo script utilizzato per analizzare il REV utilizzando la libreria **Porespy** è riportato integralmente nell'allegato A.1. L'input dello script è la *stack* ovvero l'insieme di tutte le *slices* in formato binario (.tif). Essa viene interpretata in Python come un tensore tridimensionale mediante la libreria **numpy** che si occupa della gestione di array multidimensionali. La funzione utilizzata è **metrics.representative\_elementary\_volume** che richiede in input il tensore ed un valore intero  $N_{porespy}$ . La funzione estrae  $N_{porespy}$  volte un volume randomico in dimensione e posizione all'interno della bounding box e ne calcola la porosità. Scegliendo tale valore sufficientemente grande è possibile studiare la distribuzione di  $\epsilon$  in funzione del volume ed identificare la presenza di un *plateau* (i.e. dimensione del REV) come mostrato in Fig.(4.5).
- (ii) **Volumi concentrici:** In questo caso è utilizzato uno script, riportato integralmente nell'allegato A.2, che partendo dal centro della bounding box cubica di lato  $W$ , seleziona volumi concentrici sempre più grandi e ad ogni iterazione calcola il valore di porosità. Anche questo secondo approccio permette di studiare la distribuzione della porosità in funzione di porzioni sempre maggiori di volume come mostrato in Fig.(4.6).

**Figura 4.5:** Identificazione del REV con la libreria python *porespy*. L'asse delle ascisse è discretizzato in venti classi di volume, i tratti rossi e gialli identificano rispettivamente il valor medio e la deviazione standard della porosità per ogni classe.



**Figura 4.6:** Identificazione del REV con lo script dei volumi concentrici. Aumentando la dimensione della box il profilo di porosità si stabilizza. Un'effetto di bordo è presente ai confini della bounding box.



Entrambi gli approcci adottati restituiscono un profilo di porosità, in funzione della dimensione del volume considerato, in accordo con quello descritto da J.Bear in Fig.(4.4). Entrambi i metodi rivelano una sorta di effetto di bordo in prossimità dei confini della bounding box. In questa zona esiste un trend di ascesa della porosità che si discosta dal valore stazionario di  $\epsilon$  raggiunto in corrispondenza del REV. La sezione successiva tratterà l'effetto di bordo e come questo fattore sia stato considerato nel presente lavoro.

### 4.2.1 Effetto di bordo

Durante lo studio del REV è stato identificato un'effetto di bordo presente in tutte le geometrie generate mediante l'algoritmo PlugIm. L'effetto di bordo può essere osservato in Fig.(4.5) e Fig.(4.6). L'effetto di bordo si origina durante l'impaccamento delle sfere ovvero al primo step dell'algoritmo. Una analisi approfondita condotta su centinaia di geometrie evidenzia come la magnitudine  $\mathcal{M}_{border}$  di questo effetto possa essere identificata nella seguente espressione:

$$\begin{aligned} 1.5D_{sphere} < \mathcal{M}_{border} < 2.0D_{sphere} \\ D_{sphere} = 2R \end{aligned} \quad (4.3)$$

dove  $R$  è il raggio delle sfere impaccate. Dal punto di vista operativo si è pertanto deciso di tagliare la stack (file `.tif`) di una dimensione  $\geq 2.0D_{sphere}$  a partire dai bordi. In altre parole la lunghezza  $L_{REV}$  della bounding box che definisce il dominio computazionale nelle simulazioni CFD sarà:

$$L^* < L_{REV} < (W - 2.0D_{sphere}) \quad (4.4)$$

dove  $L^*$  identifica la lunghezza minima del lato della bounding box affinché 4 *foam cells* siano comprese.

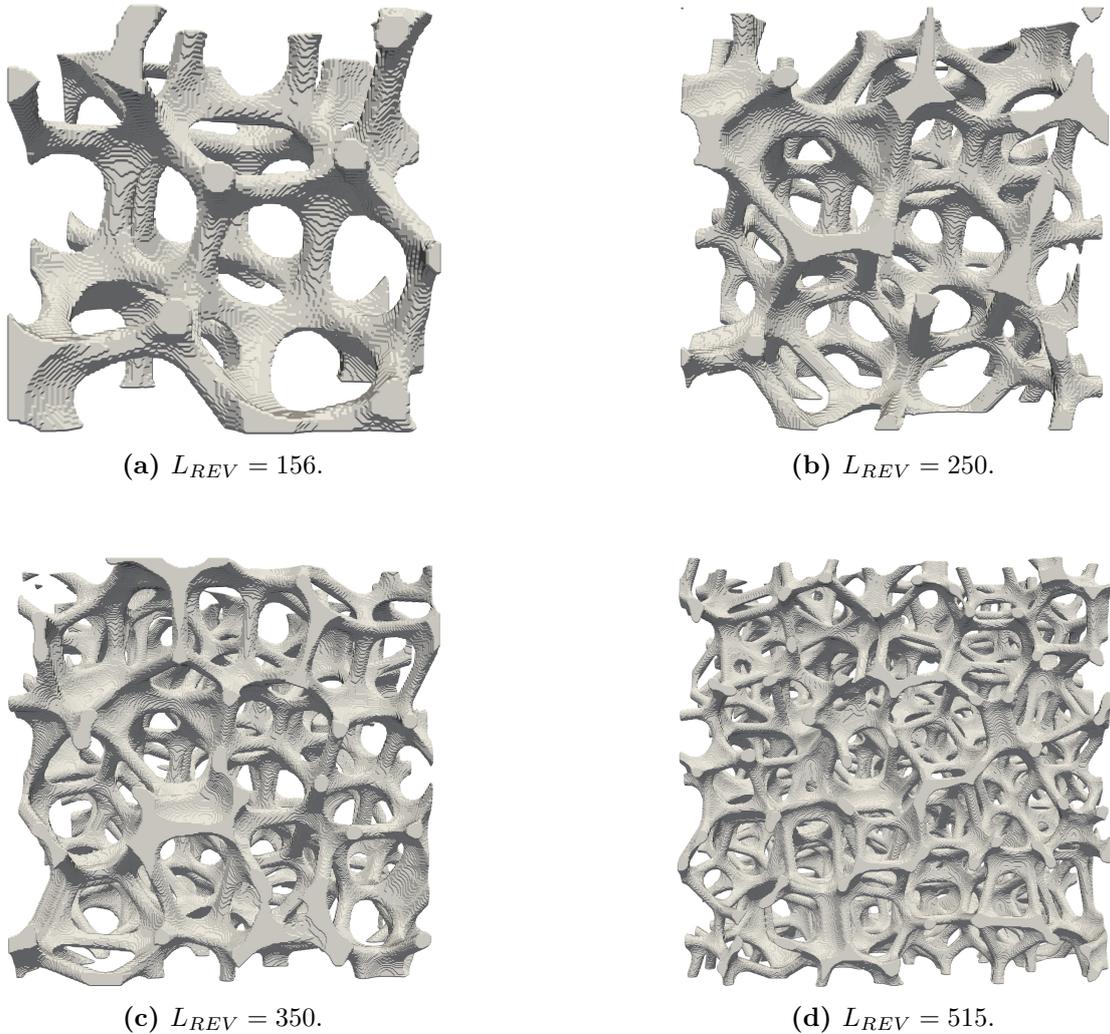


**Figura 4.7:** Immagini PlugIm di tre slices in una geometria di OCF: una iniziale, una intermedia ed una finale. La prima e l'ultima presentano evidenti effetti di bordo.

### 4.2.2 Analisi CFD del REV

L'analisi del REV è stata condotta anche mediante simulazioni CFD al fine di confermare il setup offerto da Agostini et al. [8] riguardo alla scelta del volume rappresentativo da simulare, anche per le geometrie trattate in questo lavoro.

Il setup CFD è dettagliato nella sezione 4.1. L'obiettivo di questo studio è identificare un'opportuna dimensione lineare del REV cubico in termini del numero di *foam cells* contenuti in esso. Sono state svolte simulazioni su volumetti crescenti della stessa schiuma contenenti 2, 3, 4 e 5 *foam cells* per lato.



**Figura 4.8:** Volumi progressivi simulati nell'analisi CFD del REV.

Seguendo la strategia definita nella sezione 4.12, le simulazioni sono state condotte utilizzando  $BM_{cells}/pore = 20$  ed  $R_{layer} = 2$ , in condizioni stazionarie di moto laminare (Eq.2.8). Le simulazioni sono risolte sul cluster HPC sfruttando i tools di parallelizzazione offerti da OpenFOAM e, come precedentemente accennato nella sezione 3.6, si è scelto di lavorare con 12-cores. Considerando il rapido

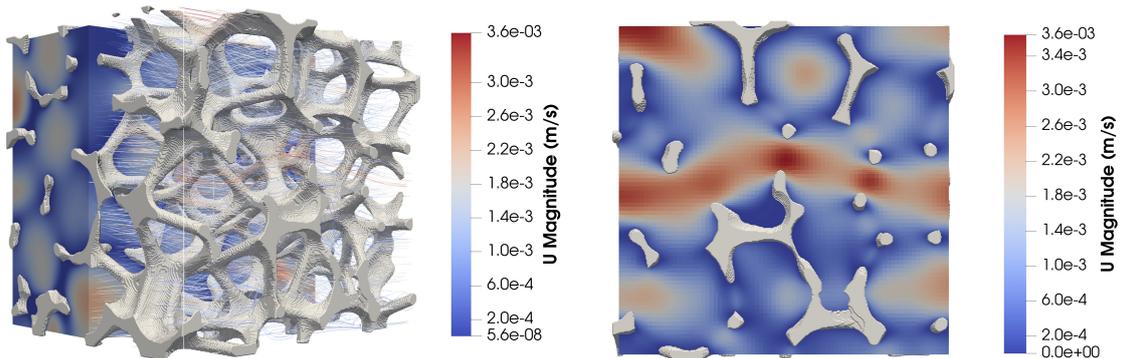
incremento del costo computazionale (Tab.4.1) e la trascurabile variazione delle caratteristiche geometriche (Tab.4.2), 4 foam cells per lato identifica un’opportuno bilanciamento tra l’accuratezza nella scelta della dimensione del REV ed il costo computazionale richiesto dalle simulazioni CFD. Tale valore sarà utilizzato come volume rappresentativo per tutte le simulazioni del presente lavoro.

**Tabella 4.1:** Dettagli numerici su geometrie e mesh.

$L_{REV}$	$Foam\ cells$	$BM_{cells}$	$R_{Layer}$	$Mesh\ cells$
156	2	50	2	1.0 MLN
250	3	60	2	2.5 MLN
350	4	80	2	6.0 MLN
515	5	100	2	13.0 MLN

**Tabella 4.2:** Risultati delle simulazioni su porzioni crescenti di schiuma.

$\epsilon$	$e_{r,\epsilon}$	$\kappa$	$e_{r,\kappa}$	$S_v$	$Re$
0.8883	1.0036 %	$4.4830 \times 10^{-8}$	13.620 %	$1.4183 \times 10^3$	$5.2469 \times 10^{-2}$
0.8951	0.2402 %	$4.8801 \times 10^{-8}$	5.9674 %	$1.3252 \times 10^3$	$3.8145 \times 10^{-2}$
0.8952	0.2364 %	$4.9550 \times 10^{-8}$	4.5246 %	$1.3084 \times 10^3$	$2.8844 \times 10^{-2}$
0.8973	0.0000 %	$5.1891 \times 10^{-8}$	0.0000 %	$1.2669 \times 10^3$	$2.0965 \times 10^{-2}$



(a) Campo di moto: linee di flusso.

(b) Campo di moto: profilo di velocità.

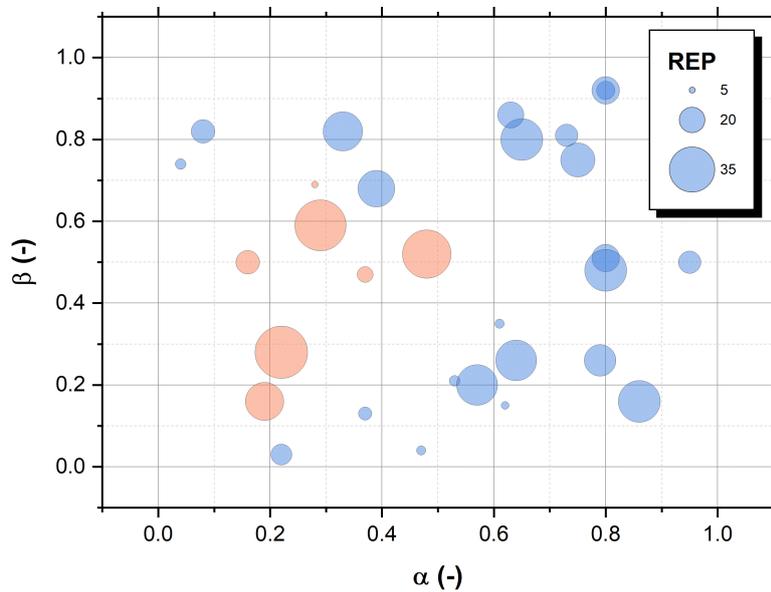
**Figura 4.9:** Campo di moto, geometria caratterizzata da  $L_{REV} = 350$ .

## 4.3 Studio preliminare

Conclusa l'analisi del REV, uno studio preliminare sulle OCF è stato condotto. L'obiettivo è l'analisi dell'influenza dei tre parametri geometrici  $\alpha$ ,  $\beta$  e  $REP$  sulla microstruttura delle OCF, in particolare sul campo di moto interno alle geometrie. Successivamente lo stesso esperimento è stato riproposto valutando l'influenza di cinque parametri geometrici:  $\alpha$ ,  $\beta$ ,  $REP$ ,  $R_{strut}$ ,  $tv$ .

### 4.3.1 Geometrie a 3|5 DOF

Sfruttando il campionamento random, trenta terne dei tre parametri sono state generate ed utilizzate per costruire altrettante geometrie. Si farà riferimento a tale insieme di geometrie come mini-dataset a tre gradi di libertà (i.e. DS-3DOF).



**Figura 4.10:** Dataset casuale di trenta geometrie con valori variabili di  $\alpha$ ,  $\beta$ ,  $REP$ ; quest'ultimo rappresentato dalla dimensione dei cerchi. Le geometrie arancioni presentano maggiore instabilità del REV per via di un impaccamento delle sfere meno compatto, dovuto a bassi valori di  $\alpha$ ,  $\beta$ .

Le trenta geometrie così ottenute sono studiate, in termini di REV, utilizzando le metodologie descritte nella sezione 4.2. Conoscendo il numero di sfere effettivamente impaccate  $N_{sph}^* \leq N_{sph}$ , queste corrispondono al numero di pori della microstruttura pertanto è possibile calcolare in numero medio di pori per lato,  $NFC_{edge}$

$$NFC_{edge} = \sqrt[3]{N_{sph}^*} \quad (4.5)$$

ed il numero medio di pori per lato nella geometria estratta (REV).

$$NFC_{edge|REV} = NFC_{edge} \cdot \left( \frac{L_{REV}}{W} \right) \quad (4.6)$$

In accordo con i risultati della sezione 4.2.2, è stato scelto  $L_{REV} = 360 \text{ voxel}$ . Tutte le geometrie modellate sono caratterizzate da  $NFC_{edge|REV} \geq 4$ . I range dei parametri geometrici sono riportati di seguito.

$$\begin{aligned} \alpha &\in (0; 1) \\ \beta &\in (0; 1) \\ REP &\in (5; 40) \end{aligned} \quad (4.7)$$

Determinata la dimensione del volume rappresentativo si procede estraendo dalla geometria originale, di lato  $W$ , il dominio computazionale di lato  $L_{REV}$ . Il processo è condotto a livello matriciale sul file `.tif` e soltanto in seguito è convertito in file `.stl`. Inoltre è stato scelto di estrarre dal file originale una geometria più grande del dominio computazionale di 4 voxel, allo scopo di facilitare successivamente il processo di meshing. Il dominio computazionale ne rimane inalterato (lato pari ad  $L_{REV}$ ). Infine il file `.stl` è ridimensionato alla grandezza specifica della schiuma in funzione dei PPI desiderati o ad una grandezza prefissata della bounding box. Tale workflow è gestito con uno script python: `cutAndSTL.py`. Il file `.stl` è ora pronto per la costruzione della mesh computazionale.

$$f_{scale} = \left( \frac{25.4}{PPI} \right) \left( \frac{NFC_{edge}}{W} \right) \times 10^{-3} \quad (4.8)$$

dove  $f_{scale}$  è lo scale factor utilizzato per scalare la geometria da voxel a metri.

Come precedentemente discusso, la mesh viene costruita a partire da una griglia cubica cartesiana di background, ottenuta con il tool `blockMesh` e caratterizzata dal numero di celle per lato. Su tale griglia viene discretizzata la geometria di OCF ottenuta dal modello geometrico mediante il tool `snappyHexMesh`. L'impostazione dei parametri dei tools di meshing è dettagliata nella sezione 4.1.1.

Definite le impostazioni di meshing per le trenta foamCases ed impostati i file di simulazione discussi nella sezione 3.5 le cartelle sono caricate sul sistema HPC e sono risolte sfruttando il calcolo parallelo su 12-cores. Al termine del processo, i risultati delle simulazioni sono stati estratti mediante gli strumenti di postprocessing descritti nella sezione 3.5.4.

Riassumendo, le operazioni eseguite sono:

- (i) Generazione di trenta terne random  $(\alpha, \beta, REP)$  e geometrie (sezione 2.1)
- (ii) Analisi del REV e scelta di  $L_{REV}$  (sezione 4.2)
- (iii) Estrazione del dominio computazionale e ridimensionamento (`cutAndSTL.py`)
- (iv) Costruzione delle trenta *foamCases*
- (v) Risoluzione CFD sul cluster HPC
- (vi) Postprocessing dei risultati grezzi

Successivamente un secondo esperimento è stato condotto utilizzando nuovamente un dataset da trenta geometrie (i.e. DS-5DOF) con cinque parametri geometrici variabili:  $(\alpha, \beta, REP, R_{strut}, tv)$ . La stessa metodologia è stata adottata. Sono riportati i range di variabilità dei parametri geometrici per questo esperimento.

$$\begin{aligned}
 \alpha &\in (0; 1) \\
 \beta &\in (0; 1) \\
 REP &\in (5; 40) \\
 R_{strut} &\in (3; 6) \\
 tv &\in (12; 24)
 \end{aligned} \tag{4.9}$$

In entrambi gli esperimenti si è scelto di fissare la dimensione lineare della bounding box ad un valore costante pari ad  $L = 11.88$  mm, conseguendo valori verosimili per la dimensione media dei pori delle schiume [8] (Tab.4.3). Infine sono riportati i valori fissi dei rimanenti parametri geometrici di generazione.

$$\begin{aligned}
 R &= 25 \\
 W &= 760 \\
 R_{node} &= 2 \cdot R_{strut} \\
 N_{sph} & \text{ a saturazione}
 \end{aligned} \tag{4.10}$$

**Tabella 4.3:** Dimensione media delle celle negli esperimenti DS-3DOS e DS-5DOS.

	$d_C$ mean	$d_C$ STD
DS-3DOS	2.33 mm	0.33 mm
DS-5DOS	2.29 mm	0.28 mm

## 4.4 Dataset esteso

Conclusi gli esperimenti sui mini-dataset lo step successivo è stato quello di costruire un dataset esteso, 300 geometrie, in modo da studiare il campo di moto su una vasta gamma di schiume, associare la permeabilità ai parametri geometrici ed allenare un modello di machine learning al fine di predire il valore di permeabilità noti i parametri geometrici in ingresso, evitando così il dispendioso passaggio attraverso l'uso della CFD. Per procedere in questa direzione è stato concretizzato il desiderio di sviluppare un algoritmo di automazione (i.e. `autoWorkflow`) necessario per ottimizzare tempi e risorse permettendo di completare in autonomia, senza l'intervento dell'utente, tutte le operazioni necessarie a partire dalla generazione delle schiume fino al caricamento e la risoluzione delle foamCases sul sistema HPC.

### 4.4.1 Algoritmo di automazione

L'algoritmo è stato scritto interamente in linguaggio Python ed è riportato per intero in appendice B. I parametri richiesti in input all'utente sono schematizzati in Tab.(4.4).

**Tabella 4.4:** Parametri in ingresso all'algoritmo di automazione.

Descrizione	Simbolo	Dimensione
Cardinalità del dataset	$\mathcal{N}$	(#)
Dimensione della geometria simulata	$PPI$	(#/inch)
Pressione cinematica all'inlet	$\mathcal{P}_{in}$	( $m^2s^{-2}$ )
Dimensione della bounding box originale	$W$	( <i>voxel</i> )
Dimensione del REV	$L_{REV}$	( <i>voxel</i> )
Celle di blockMesh per <i>foam pore</i>	$BM_{cells}$	(#)
Dimensione del buffer	<i>buffer</i>	( <i>voxel</i> )

E' di seguito esposto, per punti, il funzionamento dell'algoritmo, i suoi input ed i suoi output ed il grado di personalizzazione che può essere definito dall'utente.

- (i) Scelta casuale dei parametri geometrici per  $\mathcal{N}$  geometrie.  
Ogni geometria è generata da un file `template.xml` che contiene tutti i parametri in ingresso per il modello geometrico, cinque di questi sono variabili in range definiti dall'utente. Il primo step consiste nel generare randomicamente  $\mathcal{N}$  array

$$\mathbf{par}_i = \left[ \alpha_i \quad \beta_i \quad REP_i \quad R_{strut,i} \quad tv_i \right] \quad i = 1, 2, 3, \dots, \mathcal{N} \quad (4.11)$$

contenenti i valori da impostare nei corrispettivi files `template.xml`.

(ii) Generazione di  $\mathcal{N}$  templates.

Generati  $\mathcal{N}$  vettori le informazioni sui parametri variabili in essi contenuti sono trascritte, insieme alle informazioni invarianti, nei file `template.xml` che costituiscono l'input per il modello geometrico discusso alla sezione 2.1.

(iii) Prima di procedere con l'esecuzione del modello geometrico è generata una tabella Fig.(4.11) di riepilogo contenente per ogni geometria il valore dei parametri generatori  $\mathbf{par}_i$ .

SAMPLE	ALPHA	BETA	REP	R_strut	R_node	tv	W
1	0.78	0.53	21	5	6	14	500
2	0.94	1.00	24	3	4	19	500
3	0.96	0.56	15	4	5	11	500
4	0.64	0.64	25	7	8	18	500
5	0.69	0.81	22	3	4	10	500
6	0.65	0.66	13	5	6	19	500
7	0.73	0.89	22	6	7	17	500
8	0.98	0.95	12	3	4	10	500
9	0.61	0.95	19	4	5	13	500
10	0.73	0.93	12	6	7	13	500
11	0.63	0.84	23	6	7	19	500
12	0.86	0.93	15	3	4	16	500
13	0.91	0.62	20	5	6	16	500
14	0.79	0.64	22	4	5	18	500
15	0.90	0.70	16	6	7	16	500
16	0.79	0.84	23	8	9	18	500
17	0.92	0.75	21	8	9	13	500
18	0.91	0.59	24	3	4	17	500
19	0.73	0.64	24	8	9	11	500
20	0.65	1.00	16	3	4	11	500
21	0.65	0.98	12	8	9	15	500
22	0.98	0.94	13	5	6	15	500
23	0.76	0.58	24	4	5	20	500

Figura 4.11: Summary table del dataset.

(iv) Esecuzione iterativa del modello geometrico.

Il modello basato su PlugIm viene eseguito  $\mathcal{N}$  volte, generando altrettanti file `.stl`, uno per ogni geometria.

(v) Generazione di  $\mathcal{N}$  `foamCases`.

Si definiscono tutte le configurazioni di `Openfoam` in una `foamCases` template che verrà successivamente replicata e modificata opportunamente per adattarla ad ogni singola geometria.

(vi) Calcolo adattivo di  $BM_{cells}$ .

Il parametro determina il numero di celle desiderato per ogni lato della mesh cubica di background. Tale valore influenza il numero finale di celle della

mesh e pertanto è uno dei parametri influenti nell’analisi di *grid independence*. A seguito dello studio condotto nella sezione 4.1.1, è stato individuato come siano sufficienti 20  $BM_{cells}$  per *foam pore* per raggiungere l’indipendenza dei risultati dalla griglia. Essendo le geometrie costituite da un minimo di 4 *foam cells* per lato segue Eq.(4.12).

$$BM_{cells} \geq 80 \quad (4.12)$$

La scelta del numero di celle è cruciale in termini di costo computazionale pertanto, analizzando i dati di tempo e memoria utilizzata dalle simulazioni discusse nella sezione 4.3 è stata definita una funzione empirica allo scopo di predire il carico computazionale delle simulazioni di OCF in funzione di  $BM_{cells}$ .

$$RAM_{estimate} = 0.92697 e^{(0.03317 \times BM_{cells})} \quad (4.13)$$

Tale funzione è stata implementata nell’algoritmo in modo da richiedere dinamicamente le risorse computazionali necessarie, senza dover allocare più memoria RAM di quella effettivamente utile.

- (vii) Trasformazione e scaling dei file `.stl`.  
Ogni file `.stl` generato al 4° step deve essere centrato nel sistema di riferimento e scalato alla dimensione reale (Eq.4.8). Queste operazioni di traslazione e scaling della geometria sono automatizzate mediante la libreria Python `paraview.simple` e le sue funzioni `TranslateStl` e `ScaleStl`. Conclusa l’operazione le  $\mathcal{N}$  *foamCases* sono completamente configurate.
- (viii) Generazione del file *sbatch*, necessario per l’impostazione nel sistema di gestione delle code HPC *Slurm*, per lanciare le  $\mathcal{N}$  simulazioni sul cluster.
- (ix) Generazione del file *sbatch* per lanciare lo script di postprocessing dei risultati delle  $\mathcal{N}$  simulazioni.

## 4.4.2 Struttura del dataset

Al termine degli step di generazione, la struttura del dataset può essere così riassunta:

- $\mathcal{N}$  cartelle denominate *foam<sub>i</sub>*, con  $i = 1, 2, 3, \dots, \mathcal{N}$ . In queste cartelle è conservato il file `.tif` della geometria, un file `.png` con l’analisi del REV (Fig.4.6) e diversi file `.txt` utili ai fini dell’algoritmo tra cui `scaling.txt` nel quale è riportato il fattore di scaling della geometria `.stl`

- $\mathcal{N}$  cartelle denominate *foamCase<sub>i</sub>*, con  $i = 1, 2, 3, \dots, \mathcal{N}$ . Sono le cartelle di configurazione delle  $\mathcal{N}$  simulazioni di flusso condotte con Openfoam. La struttura di queste cartelle è già stata discussa nella sezione 3.5.
- $\mathcal{N}$  files denominati *template\_i.xml*, con  $i = 1, 2, 3, \dots, \mathcal{N}$ . Sono i file di configurazione del modello geometrico e contengono tutti i parametri utilizzati per generare ognuna delle  $\mathcal{N}$  geometrie.
- Gli script python *launcher.py* e *autoGeoGenArgPostTort.py* che eseguono l'algoritmo di automazione.
- Lo script python *input.py* che definisce tutti i valori delle variabili modificabili per l'algoritmo di automazione.
- Lo script python *findRes.py* che definisce tutte le operazioni di postprocessing da condurre sulle  $\mathcal{N}$  geometrie risolte. Restituisce come output un file di testo con i risultati (Fig.4.12).

TIME	CELLS	FLUID_VOLUME	INPORE_UX	SURFACE_AREA	SPEC_SURFACE	HYDR_DIAMETER	REYNOLDS	DARCY_VELOCITY	POROSITY	PERMEABILITY	STATUS
163	2.0e+07	6.3246e-07	2.8874e-06	1.4259e-03	1781.5	1.7742e-03	5.7561e-03	2.2816e-06	0.7902	1.8854e-08	OK
163	1.8e+07	5.3880e-07	3.1207e-06	1.2550e-03	1751.0	1.7172e-03	6.0213e-03	2.3458e-06	0.7517	1.8684e-08	OK
161	2.2e+07	7.2647e-07	3.4897e-06	1.5141e-03	1724.4	1.9193e-03	7.5255e-03	2.8873e-06	0.8274	2.4607e-08	OK
221	1.2e+07	5.7615e-07	9.2221e-06	6.7713e-04	1106.0	3.4835e-03	3.5267e-02	8.6785e-06	0.9411	6.5386e-08	OK
157	2.4e+07	7.7456e-07	3.3844e-06	1.6490e-03	1789.4	1.8789e-03	7.1450e-03	2.7175e-06	0.8029	2.3897e-08	OK
159	1.3e+07	3.9632e-07	3.6269e-06	9.0027e-04	1762.8	1.7609e-03	7.1760e-03	2.8146e-06	0.7760	2.0023e-08	OK
159	1.5e+07	4.6421e-07	3.3330e-06	1.0544e-03	1743.5	1.7610e-03	6.5950e-03	2.5583e-06	0.7676	1.9255e-08	OK
170	1.3e+07	4.3488e-07	4.6178e-06	8.7958e-04	1654.6	1.9777e-03	1.0261e-02	3.7776e-06	0.8181	2.7236e-08	OK
206	1.2e+07	5.3570e-07	6.2580e-06	7.2527e-04	1242.2	2.9545e-03	2.7414e-02	7.5767e-06	0.9175	5.6361e-08	OK
184	1.3e+07	4.5495e-07	5.3883e-06	8.4493e-04	1571.7	2.1538e-03	1.3039e-02	4.5600e-06	0.8463	3.2999e-08	OK
164	1.9e+07	6.0200e-07	3.1874e-06	1.3476e-03	1749.0	1.7868e-03	6.3992e-03	2.4902e-06	0.7813	2.8319e-08	OK
176	1.4e+07	4.6228e-07	4.4380e-06	9.3898e-04	1650.4	1.9693e-03	9.8198e-03	3.6060e-06	0.8125	2.6593e-08	OK
173	1.2e+07	4.0444e-07	4.6820e-06	8.1443e-04	1628.1	1.9864e-03	1.0450e-02	3.7853e-06	0.8085	2.6743e-08	OK
178	1.5e+07	5.0373e-07	4.4965e-06	9.9539e-04	1621.8	2.0243e-03	1.0227e-02	3.6905e-06	0.8208	2.7913e-08	OK
168	1.8e+07	5.1034e-07	1.8559e-06	1.3707e-03	1896.5	1.4939e-03	3.1056e-03	1.3185e-06	0.7861	1.9467e-08	OK
236	1.1e+07	5.3940e-07	1.0216e-05	5.8615e-04	1030.3	3.6809e-03	4.2252e-02	9.6855e-06	0.9481	7.1428e-08	OK
157	1.9e+07	5.5216e-07	2.5910e-06	1.3628e-03	1810.8	1.6207e-03	4.7181e-03	1.9009e-06	0.7337	1.5389e-08	OK
166	2.1e+07	6.6459e-07	3.4212e-06	1.4390e-03	1742.6	1.8474e-03	7.1013e-03	2.7534e-06	0.8048	2.2990e-08	OK
197	1.7e+07	6.7958e-07	6.1227e-06	1.0571e-03	1396.3	2.5714e-03	1.7690e-02	5.4959e-06	0.8976	4.4581e-08	OK
151	1.7e+07	5.0990e-07	2.9387e-06	1.1857e-03	1808.7	1.7205e-03	5.6808e-03	2.2862e-06	0.7780	1.7675e-08	OK
216	1.4e+07	6.2689e-07	8.6841e-06	7.3350e-04	1101.4	3.4186e-03	3.3357e-02	8.1741e-06	0.9413	6.3532e-08	OK
148	1.8e+07	5.2905e-07	2.5249e-06	1.2893e-03	1860.8	1.6413e-03	4.6563e-03	1.9279e-06	0.7636	1.5183e-08	OK

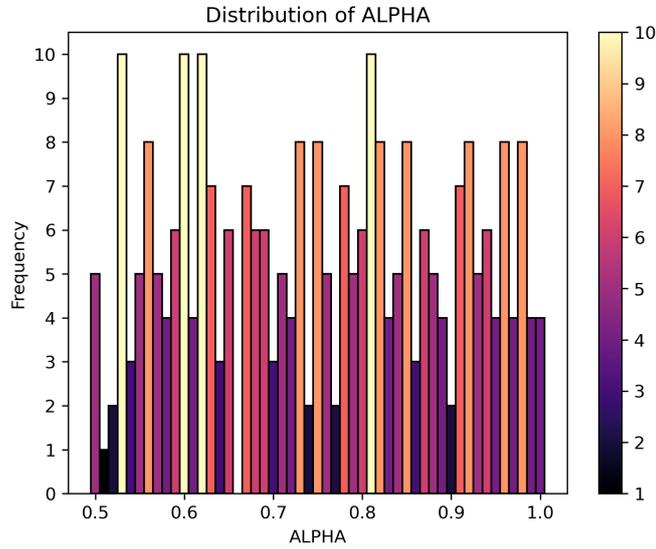
Figura 4.12: File dei risultati delle simulazioni di flusso.

### 4.4.3 Caratterizzazione delle geometrie

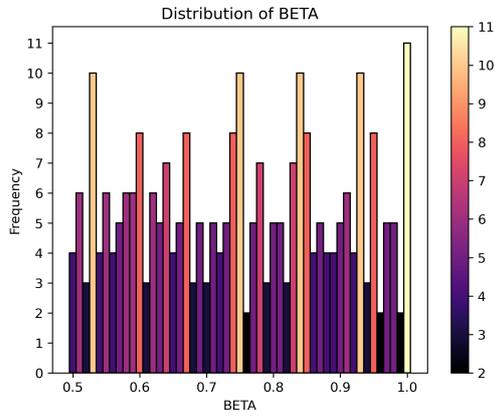
Le geometrie utilizzate per costruire il dataset esteso sono state generate a partire dai seguenti range di valori dei parametri geometrici.

$$\begin{aligned}
 \alpha &\in (0.5; 1) \\
 \beta &\in (0.5; 1) \\
 REP &\in (12; 25) \\
 R_{strut} &\in (3; 8) \\
 tv &\in (10; 20) \\
 R &= 25 \\
 W &= 760 \\
 R_{node} &= R_{strut} + 1 \\
 N_{sph} & \text{ a saturazione}
 \end{aligned} \tag{4.14}$$

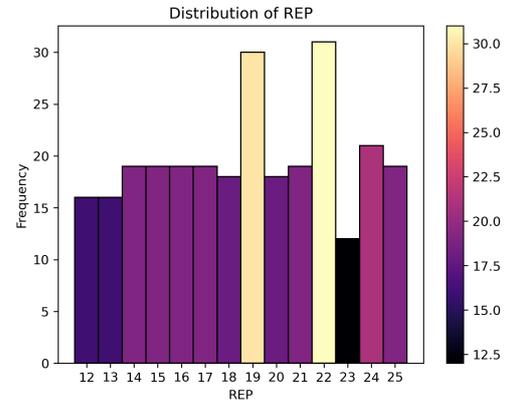
Tutti i modelli del dataset esteso riproducono geometrie di OCF da 16 PPI. L'impostazione CFD è la stessa dettagliata alla sezione 4.1, le simulazioni sono stazionarie, in condizioni laminari (Eq.2.8). La dimensione lineare del REV è scelta in accordo ai risultati della sezione 4.2.2 mentre i parametri di gestione della mesh seguono le indicazioni ottenute dall'analisi di indipendenza della griglia (sezione 4.1.1). Il gradiente di pressione lungo la direzione del flusso è  $\Delta P = 1 \times 10^{-3}$  Pa. Di seguito è riportata la distribuzione delle cinque features in ingresso al modello data-driven.



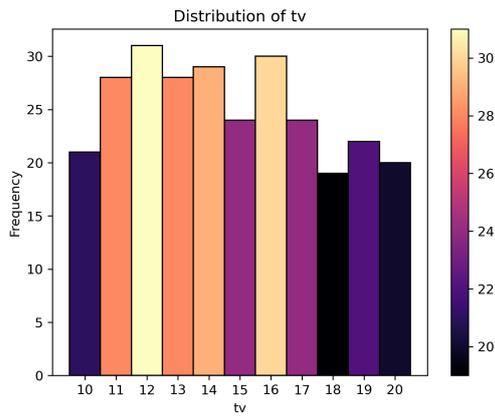
**Figura 4.13:** Distribuzione della feature  $\alpha$  nel dataset esteso.



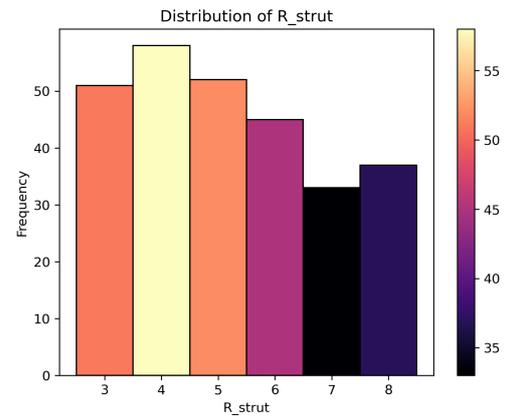
(a) Distribuzione della feature  $\beta$  nel dataset esteso.



(b) Distribuzione della feature  $REP$  nel dataset esteso.



(c) Distribuzione della feature  $tv$  nel dataset esteso.



(d) Distribuzione della feature  $R_{strut}$  nel dataset esteso.

**Figura 4.14:** Distribuzione delle features caratterizzanti le geometrie appartenenti al dataset esteso. Tutte i possibili valori delle features sono campionati; questo aspetto assumerà particolare importanza nel contesto del modello data-driven.

## 4.5 Metodologia Machine Learning

I risultati delle simulazioni sono raccolti in un file con formattazione CSV, Fig.(4.12), un formato leggero e molto versatile per memorizzare set di informazioni strutturate. Ogni riga (data object) nel file corrisponde ad una simulazione di flusso con una geometria casuale. Le colonne rappresentano le features (i.e. attributi) dei data-object, tra queste sono riportate informazioni sulla geometria come l'area superficiale, la superficie specifica, il diametro idraulico (Eq.2.10) e la porosità. La velocità media nei pori è ottenuta come risultato dalla simulazione CFD e permette di calcolare la velocità superficiale, il numero di Reynolds (Eq.2.9) e la permeabilità.

Come precedentemente accennato la costruzione di un modello data-driven richiede una serie di passaggi di elaborazione dei dati grezzi per raggiungere infine lo step di allenamento del modello. Tale processo prende comunemente il nome di Machine Learning Pipeline e successivamente ne saranno dettagliati i passaggi sia per il classificatore SVM sia per il modello di regressione MLP.

### 4.5.1 Classificatore SVM: pipeline

Il modello geometrico utilizzato per generare le schiume OCF non segue alcuna fisica pertanto le geometrie ottenute possono rappresentare schiume non verosimili. Tali geometrie agiscono come *outliers* nei confronti del modello di regressione e non hanno nessun significato dal punto di vista del presente studio. Queste argomentazioni supportano la volontà di costruire un modello di classificazione capace di etichettare istantaneamente le nuove geometrie, analizzando il set di parametri generatori (Tab.2.1), come schiume fisiche o non fisiche. Le schiume non fisiche saranno scartate dal modello di regressione e nessuna risorsa computazionale sarà consumata per eseguire simulazioni su di esse. Di seguito saranno dettagliati i principali step della MLPL per il classificatore SVM:

- (i) Caricamento dei dati. Il dataset utilizzato per allenare e validare il modello di classificazione è costituito dalle prime 200 geometrie OCF appartenenti al dataset esteso, la cui cardinalità è pari a 300. Tali geometrie sono state classificate manualmente come fisiche o non fisiche osservandole una ad una mediante il software *Paraview* [41] ed assegnando rispettivamente una class label pari ad 1 oppure 0. La distribuzione di class labels assegnata alle geometrie è descritta in Fig.(4.16).
- (ii) Separazione dei vettori delle features (Eq.4.11) dalle corrispondenti class labels. Al termine di questo step sarà disponibile un dataframe  $\mathcal{F}$  contenente tutte le features (Tab.4.5) ed un dataframe  $\mathcal{T}$  contenente tutte le class labels (Tab.4.6).

**Tabella 4.5:** Dataframe delle features  $\mathcal{F}$ , per il dataset di train-test del modello di classificazione SVM.

	$\alpha$	$\beta$	$REP$	$R_{strut}$	$tv$
<b>1</b>	$\alpha_1$	$\beta_1$	$REP_1$	$R_{strut,1}$	$tv_1$
<b>2</b>	$\alpha_2$	$\beta_2$	$REP_2$	$R_{strut,2}$	$tv_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<b>200</b>	$\alpha_{200}$	$\beta_{200}$	$REP_{200}$	$R_{strut,200}$	$tv_{200}$

**Tabella 4.6:** Dataframe delle class labels  $\mathcal{T}$ , per il dataset di train-test del modello di classificazione SVM.

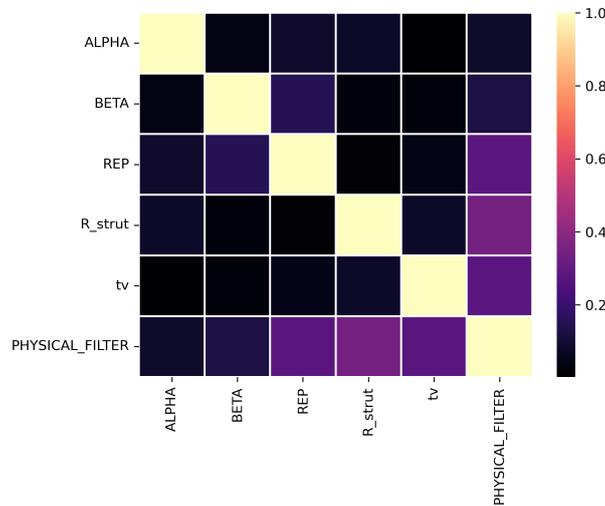
Class label	
<b>1</b>	$l_1$
<b>2</b>	$l_2$
$\vdots$	$\vdots$
<b>200</b>	$l_{200}$

- (iii) Analisi della matrice di correlazione (metodo di Pearson). Nelle applicazioni di supervised learning è importante verificare che le features selezionate siano statisticamente indipendenti le une dalle altre, inoltre è anche possibile sfruttare la matrice di correlazione per osservare quali features sono più correlate alla class label. Per condurre questa analisi spesso sono utilizzati la matrice di correlazione ed il metodo di Pearson. La matrice riporta i valori del coefficiente di correlazione lineare calcolato per coppie di colonne con il metodo di Pearson.

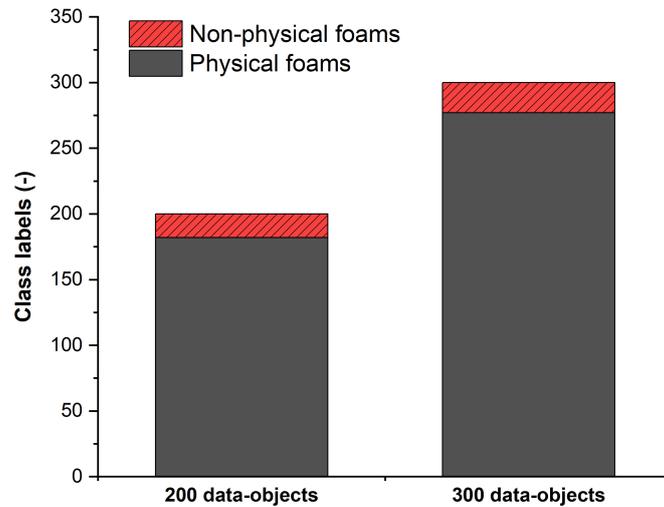
$$\begin{aligned}
 r_{xy} &= \frac{cov(X, Y)}{\sigma_X \sigma_Y} \\
 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}
 \end{aligned} \tag{4.15}$$

Il coefficiente  $r_{xy}$  può assumere valori compresi nel range (0,1) dove 0 indica una completa scorrelazione mentre 1 indica una relazione lineare perfetta. Analizzando la matrice ottenuta è possibile osservare che le features non sono

linearmente correlate e che le più significative nella determinazione della class label sono  $R_{strut}$ ,  $REP$  e  $tv$ , come mostrato in Fig.(4.15).



**Figura 4.15:** Matrice di correlazione (metodo di Pearson) applicata alla features ed alla class label.



**Figura 4.16:** Distribuzione delle class labels per il dataset di train/test del modello SVM e per il dataset esteso. Le geometrie etichettate come non fisiche saranno rimosse dal pool di training del modello di regressione.

- (iv) Scaling delle features. Uno step cruciale da svolgere prima di procedere alla fase di training del modello è effettuare lo scaling delle features, ovvero scalare

i valori delle features in modo che la loro influenza sulla class label non sia condizionata dal loro valore assoluto. In questo modo features con valori numerici differenti di vari ordini di grandezza potranno contribuire equamente al training del modello. In questo lavoro due operazioni di scaling sono state impiegate ovvero la normalizzazione MinMax

$$\mathbf{x}' = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \quad (4.16)$$

e la standardizzazione zScore.

$$\mathbf{x}' = \frac{\mathbf{x} - \bar{\mathbf{x}}}{s_x} \quad (4.17)$$

dove  $s_x$  è la deviazione standard campionaria ed  $x$  è il generico vettore di una feature.

$$s_x = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{\mathbf{x}})^2} \quad (4.18)$$

- (v) Training del modello: approccio LOO (Leave One Out). Considerando il modesto numero di data-objects rispetto alla canonica pratica delle applicazioni machine learning, basate su decine di migliaia di istanze, è stato deciso di adottare l'approccio LOO come alternativa al classico partizionamento 80% train - 20% test. La metodologia LOO prevede di re-iterare la fase di training tante volte quante sono le istanze del dataset; ad ogni iterazione il training set è composto soltanto dall' $i$ -esima istanza mentre il training set è composto da tutte le altre. Ad ogni iterazione sono valutate le metriche di riferimento per la classificazione ed al termine della procedura si effettua una media su tutte le iterazioni. Le metriche utilizzate in questo lavoro sono la precision  $prc$ , il recall  $rec$  ed  $F_1$ .

$$prc = \frac{TP}{TP + FP} \quad (4.19)$$

$$rec = \frac{TP}{TP + FN} \quad (4.20)$$

$$F_1 = \frac{2 \text{ rec } prc}{rec + prc} \quad (4.21)$$

dove TP e FN sono i valori true positive e false negative mentre FP e TN sono i valori false positive e true negative. Positive e negative sono esempi di due class label in un problema di classificazione binaria.

## 4.5.2 Regressore FFNN: pipeline

Come introdotto in precedenza, l'uso di FFNN permetterà la costruzione di un modello data-driven per la predizione della permeabilità di una particolare schiuma OCF generata dall'algoritmo Plugim, noti i valori dei parametri generatori (Tab.2.1). I principali step per questa applicazione machine learning sono i seguenti.

- (i) Caricamento dei dati. Il dataset utilizzato per allenare e validare il modello di regressione della permeabilità è costituito dal sopracitato dataset esteso, di 300 geometrie.
- (ii) Applicazione filtri. Due filtri sono stati impiegati per rimuovere dal dataset due particolari classi di data-object. La prima classe comprende le geometrie per le quali risulta fallita la simulazione CFD o la costruzione della mesh computazionale. La solida robustezza dell'algoritmo di automazione e dell'impostazione fluidodinamica hanno permesso di riscontrare una sola geometria fallita sulle trecento generate nel dataset esteso. La seconda classe comprende le schiume etichettate come 'non fisiche', ovvero quelle geometrie nelle quali la scelta dei parametri geometrici ha determinato la formazioni di micro-strutture altamente inverosimili (e.g. celle sferiche, assenza di *windows*, etc. . . ); le prime duecento sono etichettate manualmente mentre le ultime cento sono state etichettate dal classificatore SVM (sezione 4.5.1). Terminato questo step è disponibile il dataset pulito, pronto per gli step successivi di training e test.
- (iii) Separazione dei vettori delle features dalle corrispondenti class labels. Al termine di questo step sarà disponibile un dataframe  $\mathcal{F}^*$  contenente tutte le features (Tab.4.7) ed un dataframe  $\mathcal{T}^*$  contenente tutte le class labels (Tab.4.8).
- (iv) Scaling delle features, condotto in completa analogia con il medesimo passaggio della MLPL del modello di classificazione SVM.
- (v) Training del modello: approccio Leave One Out. La procedura di training e validazione del modello di regressione è stata performata utilizzando l'approccio LOO, precedentemente discusso per il classificatore SVM.

**Tabella 4.7:** Dataframe delle features  $\mathcal{F}^*$ , per il dataset di train-test del modello di regressione MLP.

	$\alpha$	$\beta$	$REP$	$R_{strut}$	$tv$
<b>1</b>	$\alpha_1$	$\beta_1$	$REP_1$	$R_{strut,1}$	$tv_1$
<b>2</b>	$\alpha_2$	$\beta_2$	$REP_2$	$R_{strut,2}$	$tv_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<b>276</b>	$\alpha_{276}$	$\beta_{276}$	$REP_{276}$	$R_{strut,276}$	$tv_{276}$

**Tabella 4.8:** Dataframe dei numerical targets  $\mathcal{T}^*$ , per il dataset di train-test del modello di regressione MLP.

Numerical target	
<b>1</b>	$\kappa_1$
<b>2</b>	$\kappa_2$
$\vdots$	$\vdots$
<b>276</b>	$\kappa_{276}$

## Capitolo 5

# Seconda fisica: trasporto di calore

Completato il lavoro sulle simulazioni di flusso è stata indagata una seconda fisica all'interno delle schiume solide: il trasporto di calore. Il capitolo sarà interamente dedicato alla descrizione del sistema e dell'impostazione adottata per le simulazioni di trasporto di calore.

### 5.1 Setup delle simulazioni di scambio termico

Lo studio dei fenomeni di trasporto di calore è stato svolto sfruttando il solver `scalarTransportFoam`, discusso nella sezione 3.5. Inizialmente viene condotta una simulazione di flusso per ottenere il campo di velocità e di pressione, successivamente il campo di moto è impiegato per calcolare la distribuzione di temperatura sul dominio computazionale. L'impostazione ed il setup della simulazione di flusso è descritto nella sezione 4.1.

Il campo di temperatura è inizializzato fissando una temperatura costante sulla superficie della schiuma,  $T_{foam} = 353\text{ K}$ . Tale condizione di bordo descrive, in modo approssimativo, un'applicazione nella quale una schiuma solida metallica è utilizzata come *heat sink* rispetto ad un device operante ad un livello termico superiore. Nei materiali metallici il fenomeno di conduzione è molto efficace pertanto è verosimile immaginare un continuo afflusso di calore alla schiuma che mantiene, in questo modello approssimato, una temperatura superficiale costante. In tutte le simulazioni condotte il fluido considerato è l'acqua, caratterizzata da una temperatura di inlet pari a  $T_{in} = 293\text{ K}$ . Nota la temperatura media del fluido all'outlet, come risultato della simulazione, è possibile calcolare le quantità rilevanti ai fini del problema di scambio termico come la portata di calore asportata, il

coefficiente di scambio termico ed il numero di Nusselt (Eq.2.17). Tali equazioni sono state dettagliate nella sezione 2.2.2.

## 5.2 Simulazioni di interesse

Lo studio dello scambio termico affrontato in questo lavoro è affetto da due principali limitazioni che saranno riportate prima di procedere con la metodologia utilizzata per impostare tre differenti esperimenti sullo scambio termico. Inizialmente è stato condotto un primo confronto tra due geometrie di OCF al variare del  $Re$ . Successivamente è stato impostato un secondo esperimento di analisi del comportamento di nove differenti schiume OCF a  $Re$  costante. Infine è stata eseguita una seconda battuta del precedente esperimento ma a portata costante.

Le principali limitazioni di cui è affetto lo studio sullo scambio termico sono:

- Moto laminare. Le applicazioni ingegneristiche di sottrazione del calore, *heat sink*, necessitano nella realtà di sviluppare elevati coefficienti di scambio termico, e questo può essere raggiunto soltanto in condizioni di moto turbolento.
- Temperatura superficiale costante. L'imposizione della temperatura superficiale costante rappresenta un'ipotesi forte. Un modello più dettagliato dovrebbe tenere in considerazione anche i fenomeni di scambio termico all'interno della fase solida in modo da rappresentare verosimilmente l'inerzia termica delle strutture caratteristiche delle OCF come nodi e struts.

### 5.2.1 Esperimenti numerici

Un primo esperimento è stato condotto su due differenti geometrie di OCF al fine di studiare l'evoluzione delle perdite di carico e del coefficiente di scambio termico per differenti valori di  $Re$  in condizioni laminari e di transizione. Le geometrie scelte sono caratterizzate dalle seguenti proprietà:

**Tabella 5.1:** Caratterizzazione delle due geometrie impiegate per il primo esperimento numerico.

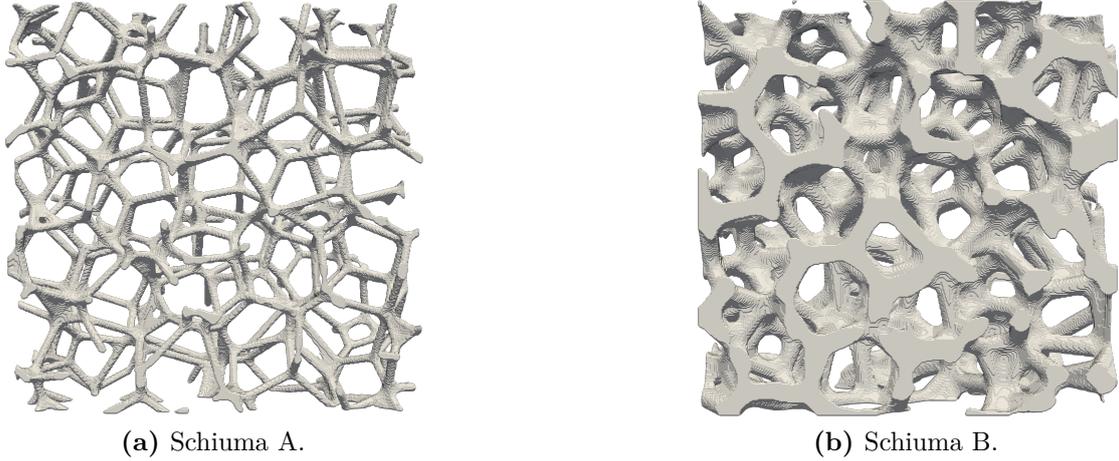
Schiuma	$L_{bb}$ (m)	$\epsilon$ (-)	$S_{foam}$ (m <sup>2</sup> )	$S_v$ (m <sup>2</sup> m <sub>REV</sub> <sup>-3</sup> )	$D_h$ (m)
A	$8.336 \times 10^{-3}$	0.957	$5.531 \times 10^{-4}$	955	$4.010 \times 10^{-3}$
B	$8.477 \times 10^{-3}$	0.779	$1.045 \times 10^{-3}$	1715	$1.817 \times 10^{-3}$

Per ogni geometria sono state svolte cinque simulazioni di flusso, variando il gradiente di pressione, e cinque corrispondenti simulazioni di trasporto del calore.

Le cinque condizioni di bordo per le simulazioni di flusso sono le seguenti:

$$\Delta\mathcal{P} = [1 \ 5 \ 10 \ 50 \ 100] \times 10^{-5} \ (m^2 \ s^{-2}) \quad (5.1)$$

mentre la condizione di bordo per la simulazione di trasporto del calore è descritta nella sezione 5.1.

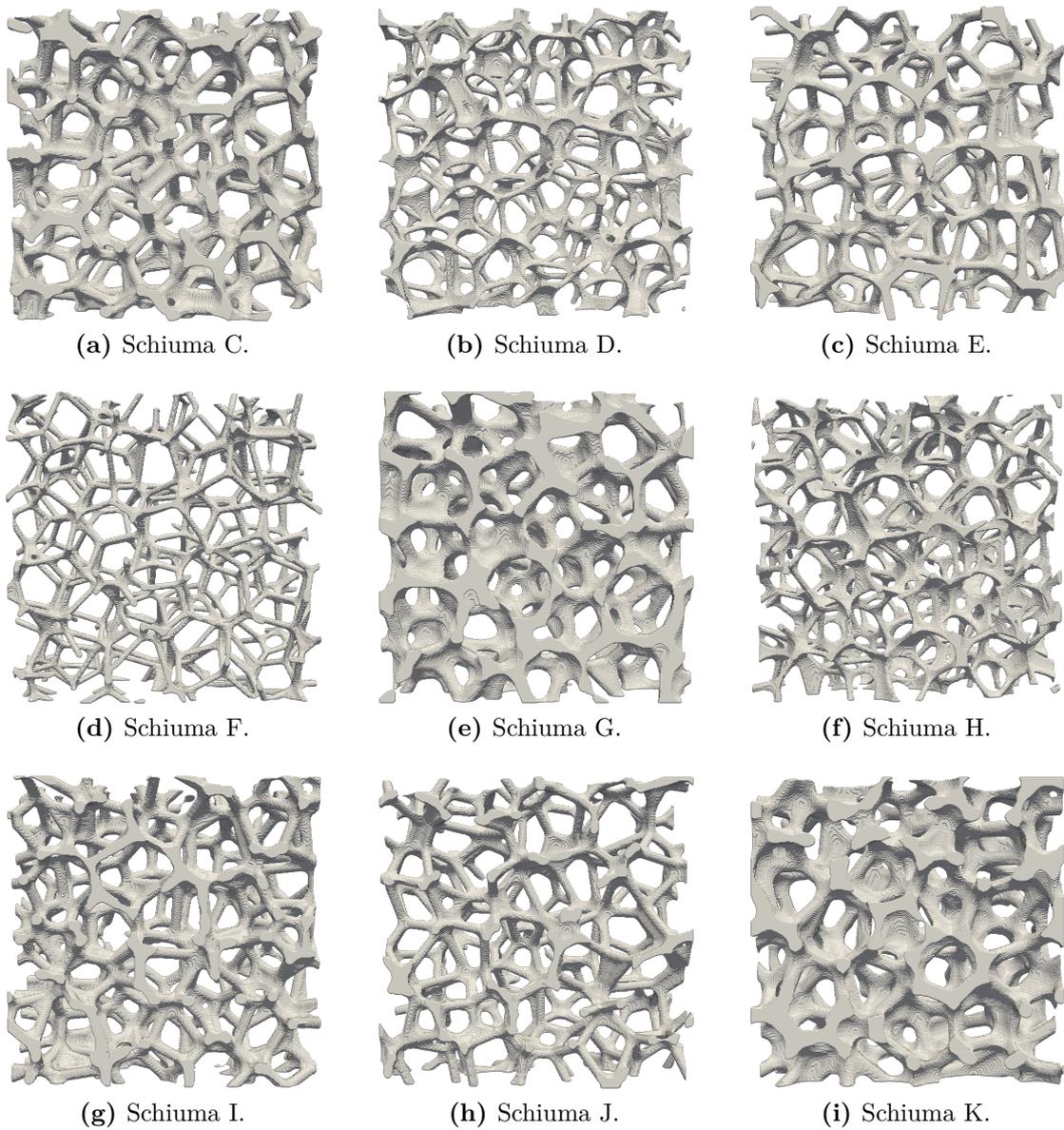


**Figura 5.1:** Sezione y-normal delle due geometrie utilizzate nel primo esperimento numerico.

**Tabella 5.2:** Caratterizzazione delle nove geometrie impiegate per il secondo ed il terzo esperimento numerico.

Schiuma	$L_{bb} \ (m)$	$\epsilon \ (-)$	$S_{foam} \ (m^2)$	$S_v \ (m^2 \ m_{REV}^{-3})$	$D_h \ (m)$
C	$8.912 \times 10^{-3}$	0.845	$1.093 \times 10^{-3}$	1545	$2.189 \times 10^{-3}$
D	$8.470 \times 10^{-3}$	0.943	$6.613 \times 10^{-4}$	1088	$3.466 \times 10^{-3}$
E	$8.078 \times 10^{-3}$	0.902	$6.877 \times 10^{-4}$	1305	$2.765 \times 10^{-3}$
F	$8.337 \times 10^{-3}$	0.957	$5.519 \times 10^{-4}$	953	$4.019 \times 10^{-3}$
G	$8.855 \times 10^{-3}$	0.788	$1.201 \times 10^{-3}$	1729	$1.822 \times 10^{-3}$
H	$9.042 \times 10^{-3}$	0.933	$8.529 \times 10^{-4}$	1154	$3.234 \times 10^{-3}$
I	$8.887 \times 10^{-3}$	0.849	$1.073 \times 10^{-3}$	1529	$2.223 \times 10^{-3}$
J	$8.093 \times 10^{-3}$	0.907	$6.701 \times 10^{-4}$	1264	$2.869 \times 10^{-3}$
K	$7.954 \times 10^{-3}$	0.786	$8.603 \times 10^{-4}$	1710	$1.839 \times 10^{-3}$

In secondo esperimento numerico sono state selezionate nove geometrie di OCF con l'obiettivo di esaminare perdite di carico e scambio termico in differenti microstrutture, a parità del numero di Reynolds. Il terzo esperimento numerico ripropone l'utilizzo delle stesse geometrie ma conducendo simulazioni a parità di portata fluida in ingresso al dominio.



**Figura 5.2:** Sezione y-normal delle nove geometrie utilizzate nel secondo e nel terzo esperimento numerico.

**Parte IV**

**Risultati e Conclusioni**



## Capitolo 6

# Sistema complesso

Una delle principali difficoltà riscontrate nello studio fluidodinamico di geometrie di OCF è la complessità della microstruttura, la vastità di parametri e descrittori disponibili per caratterizzare le geometrie. Un confronto può essere d'aiuto per focalizzare il concetto: immaginiamo un sistema *semplice* come un fluido che scorre in un tubo a sezione circolare. In questo sistema è possibile scegliere facilmente una dimensione caratteristica, il diametro della tubazione, come principale descrittore. La scelta di una dimensione caratteristica permette di definire, in modo relativamente diretto, un numero adimensionale che permetta di misurare l'importanza relativa dei fenomeni di trasporto advettivo e diffusivo di quantità di moto: il numero di  $Re$ . Seguendo i binari così definiti sarà diretto lo studio dei regimi di flusso laminare e turbolento, la transizione tra i due e tutte le conseguenze che ne conseguono sui fenomeni di trasporto di altre quantità come calore e materia.

Ritorniamo ora alle geometrie di schiume a celle aperte; in questo sistema si deve trattare un mezzo poroso, semi-randomico, che nel più semplice dei casi può essere caratterizzato facendo riferimento alle tre strutture chiave dettagliate nella sezione 1.1: le struts, le celle e le windows. Questa immagine, estremamente semplificata, determina già delle difficoltà nel scegliere quale dovrebbe essere la dimensione caratteristica da adottare come riferimento. Agostini et al. [8] propone diverse grandezze tra cui la dimensione media delle celle, la dimensione media delle windows, il diametro idraulico. In questo lavoro, come in precedenza argomentato, è stato deciso di scegliere come dimensione caratteristica il diametro idraulico

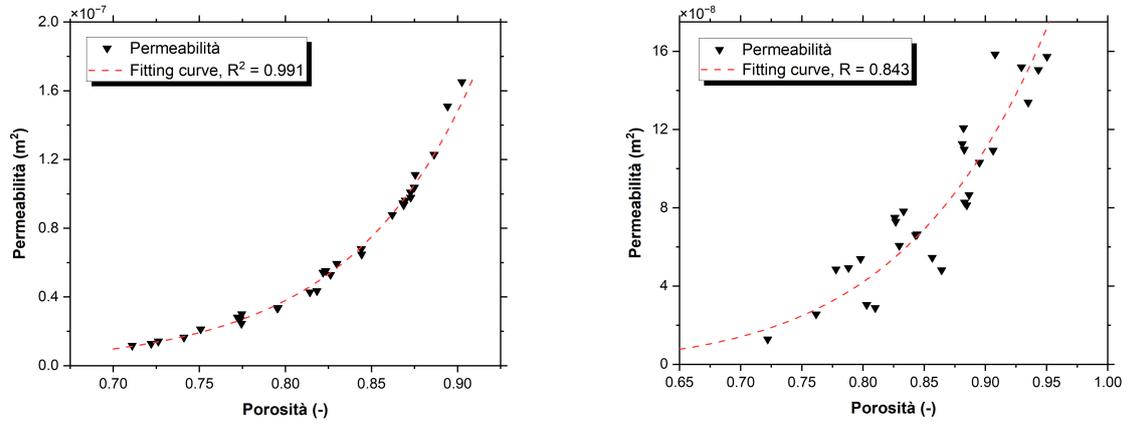
$$D_h = \frac{4V_{wet}}{S_{wet}} = \frac{4V_{fluid}}{S_{foam}} = \frac{4\epsilon}{S_v} \quad (6.1)$$

in modo tale da definire il  $Re$  utilizzando Eq.(2.9). Per questo setup è stato identificato il range di transizione turbolenta del flusso in corrispondenza di  $Re = 150 \sim 200$ , condizione per la quale l'elevata instabilità richiede l'utilizzo di modelli turbolenti.

Ulteriori difficoltà sorgono nel momento in cui si cerca di applicare un'approccio classico, come una legge di potenza, per derivare relazioni tra le grandezze significative per queste particolari geometrie. Tali avversità sono principalmente ricondotte all'elevato numero di descrittori significativi delle geometrie (problema di elevata dimensionalità) e sono responsabili dell'appellativo "*sistema complesso*".

## 6.1 Fallimento dell'approccio classico

L'indagine sugli effetti del numero e della tipologia di parametri geometrici utilizzati nella costruzione di geometrie OCF ha rapidamente fatto emergere l'intrinseca difficoltà nell'identificare relazioni classiche, di tipo *power law*, capaci di esprimere relazioni funzionali tra variabili rilevanti per il sistema indagato.



(a) Relazione *power law* tra porosità e permeabilità nel dataset DS-3DOF, contenente geometrie generate a partire da tre parametri geometrici.

(b) Relazione *power law* tra porosità e permeabilità nel dataset DS-5DOF, contenente geometrie generate a partire da cinque parametri geometrici.

**Figura 6.1:** Deterioramento delle classiche relazioni *power law* tra variabili chiave, all'aumentare della dimensionalità del problema.

La figura 6.1 mostra due relazioni *power law* utilizzate per il fitting dei dati di permeabilità e porosità, ottenuti da due dataset di geometrie OCF: nel primo caso i parametri di generazione variabili sono tre ( $\alpha$ ,  $\beta$ ,  $REP$ ) mentre nel secondo caso i parametri sono cinque ( $\alpha$ ,  $\beta$ ,  $REP$ ,  $R_{strut}$ ,  $tv$ ). Le equazioni delle curve di fitting dei dati sono:

$$\begin{aligned} \kappa_3 &= a \times \epsilon^b & R &= 0.991 \\ \kappa_5 &= c \times \epsilon^d & R &= 0.843 \end{aligned} \quad (6.2)$$

dove i pedici indicano il numero di parametri geometrici variabili.

In particolare:

$$\begin{aligned}
 a &= 4.49602 \times 10^{-7} \\
 b &= 10.95998 \\
 c &= 2.61267 \times 10^{-7} \\
 d &= 8.18588
 \end{aligned} \tag{6.3}$$

Validata la capacità del modello geometrico di ricostruire fedelmente la microstruttura di schiume solide reali [8], un’interessante obiettivo è stato quello di ricercare una relazione tra i parametri geometrici di input dell’algoritmo ed un valore target caratterizzante la geometria, ovvero la permeabilità  $\kappa$ . Inizialmente è stato impiegato un modello di regressione lineare multivariabile utilizzando come regressori tre e cinque parametri, rispettivamente per i dataset DS-3DOF e DS-5DOF.

$$\begin{aligned}
 R_3^2 &= 0.974, \\
 R_5^2 &= 0.921
 \end{aligned} \tag{6.4}$$

dove i pedici indicano il numero di parametri geometrici variabili; in particolare:

$$\begin{aligned}
 \kappa_3 &= - (3.07638 \times 10^{-8}) \cdot \alpha - (1.84200 \times 10^{-8}) \cdot \beta \\
 &\quad + (3.90320 \times 10^{-9}) \cdot REP + 5.97200 \times 10^{-9}, \\
 \kappa_5 &= - (2.49113 \times 10^{-8}) \cdot \alpha - (6.36696 \times 10^{-9}) \cdot \beta \\
 &\quad + (4.16255 \times 10^{-9}) \cdot REP - (2.23745 \times 10^{-8}) \cdot R_{strut} \\
 &\quad - (3.72457 \times 10^{-9}) \cdot tv + 1.84407 \times 10^{-7}.
 \end{aligned} \tag{6.5}$$

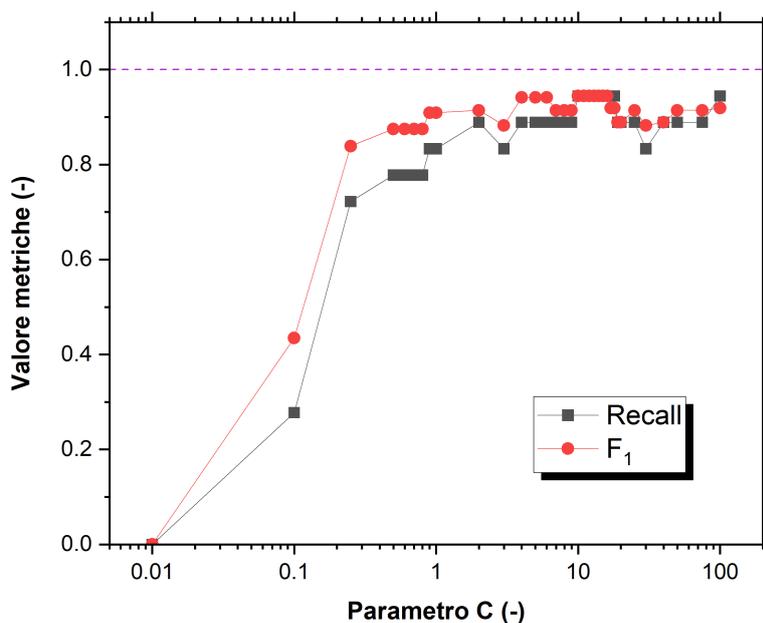
Considerando Eqs.(6.2 e 6.4), risulta evidente come, all’aumentare dei parametri geometrici utilizzati per diversificare le microstrutture di OCF, l’approccio classico basato sul fitting di dati sperimentali, ottenuti da prove sul campo o simulazioni numeriche, sia sempre più debole. All’aumentare della dimensionalità del sistema è difficile interpretare le relazioni funzionali con semplici leggi di potenza o definire modelli di regressione lineare estremamente accurati. Per tale motivo, l’introduzione di un’approccio basato su tecniche di machine learning è stato sperimentato in questo lavoro. Le potenzialità di strumenti come le reti neurali possono fornire un valido punto di svolta nello studio e nella modellazione di sistemi complessi come le geometrie di schiume a celle aperte.

L’esigenza di costruzione di un dataset esteso, necessario per applicare un approccio di tipo data-driven al problema, è stata soddisfatta dall’elaborazione del workflow automatico di generazione, simulazione e raccolta dei risultati, ampiamente discusso e dettagliato nella sezione 4.4.1 del manoscritto e nell’appendice B.

Conclusa la raccolta dei risultati delle simulazioni due tecniche di machine learning sono state applicate al dataset disponibile: un modello data-driven di classificazione per scartare le geometrie non fisiche (sezione 4.5.1) e, successivamente, un modello di regressione di tipo FFNN, discusso nella sezione 4.5.

## 6.2 Risultati dell'approccio machine learning

I modelli data-driven appartenenti alla classe *supervised learning* necessitano di una fase di training. Tale processo è regolato da parametri di modello detti hyper-parameters. Il principale parametro del modello di regressione SVM è  $C$ , il suo effetto è stato ampiamente discusso nella sezione 4.5.1. Sono stati condotti molteplici esperimenti per identificare un'opportuno valore dell'*hyper-parameter*  $C$  ed i risultati della grid search sono riportati nella Fig.(6.2).



**Figura 6.2:** Sono stati svolti esperimenti per diversi valori del parametro  $C$ . Ogni esperimento è caratterizzato da un valore di recall ed un valore di F1-Score. Le due metriche sono massimizzate in corrispondenza del range  $C = 10, 11, \dots, 16$ .

Il modello SVM è stato pertanto allenato impostando  $C = 13$  ed un kernel lineare. La *confusion matrix* del modello allenato è rappresentata in Eq.(6.6).

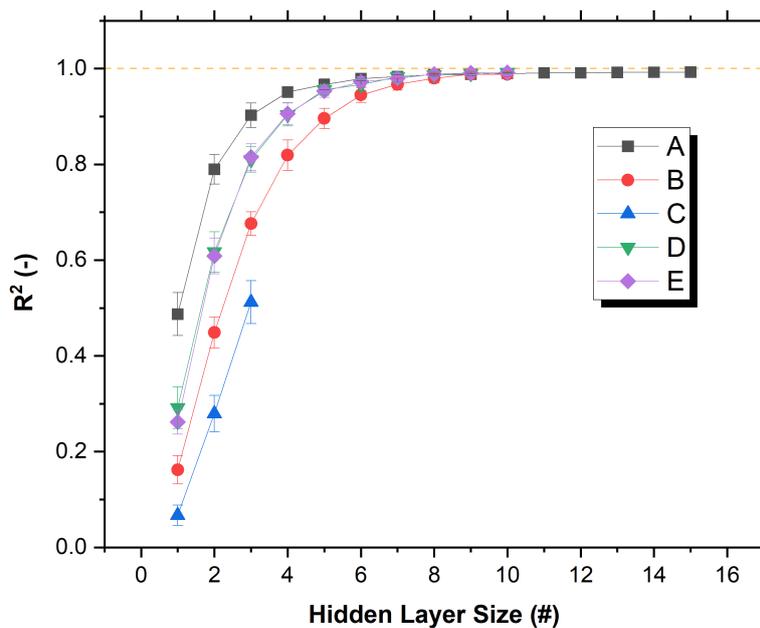
$$\begin{bmatrix} 17 & 1 \\ 1 & 181 \end{bmatrix} \quad (6.6)$$

Le righe rappresentano rispettivamente i tags di schiuma "Reale Non Fisica" e "Reale Fisica" mentre le colonne sono associate rispettivamente ai tags di schiuma "Predetta Non Fisica" e "Predetta Fisica". Riassumendo, su 18 geometrie non fisiche 17 sono state classificate correttamente mentre su 181 geometrie fisiche 180 sono state correttamente etichettate.

Uno studio analogo è stato condotto sul principale hyper-parameter del modello di regressione MLP, ovvero il parametro che determina l'architettura della rete. Differenti configurazioni sono state proposte ed elencate in Tab.(6.1); per ogni configurazione sono stati eseguiti venti processi di training e la Fig.(6.3) riporta il valore medio della metrica di riferimento ( $R^2$ ) e la sua deviazione standard ( $\sigma$ ).

**Tabella 6.1:** Architetture FFNN indagate.

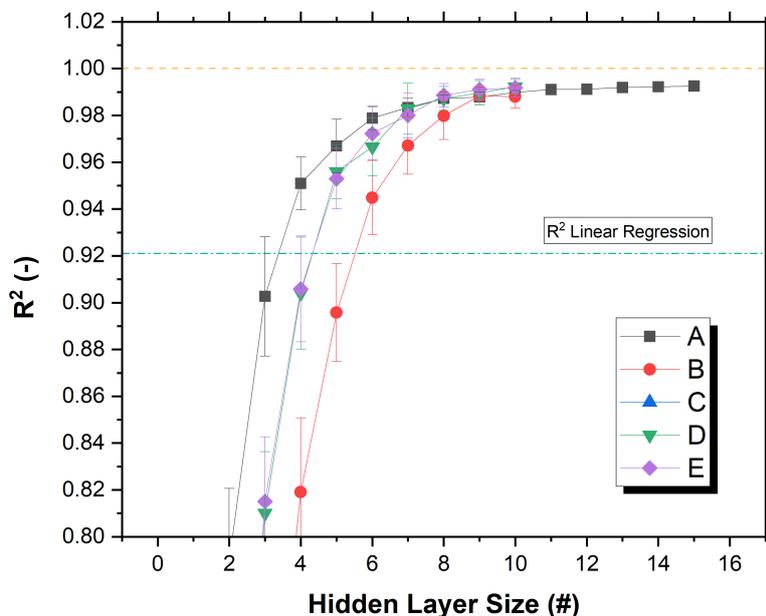
Descrizione	Schema	Tag
Singolo layer	( <i>hls</i> )	A
Due layers omogenei	( <i>hls, hls</i> )	B
Tre layers omogenei	( <i>hls, hls, hls</i> )	C
Due layers non omogenei	( <i>hls, 2hls</i> )	D
Due layers non omogenei	( <i>2hls, hls</i> )	E



**Figura 6.3:** Sono stati svolti esperimenti per architetture della rete neurale. Ogni esperimento è caratterizzato da un valore medio di  $R^2$  e dalla sua deviazione standard  $\sigma$ . Maggiore è il valore di  $R^2$  e migliore è l'accuratezza del modello.

Considerando le architetture *A*, *B*, *C* si osserva che all'aumentare del numero di hidden layers della rete, le prestazioni del modello si riducono a parità di neuroni

per livello. Le due architetture non omogenee ( $D$ ,  $E$ ) presentano performance superiori rispetto alla configurazione omogenea ( $B$ ), tuttavia l'architettura più semplice, a singolo livello ( $A$ ), risulta la migliore sia in termini di performance, sia in termini di costo computazionale.



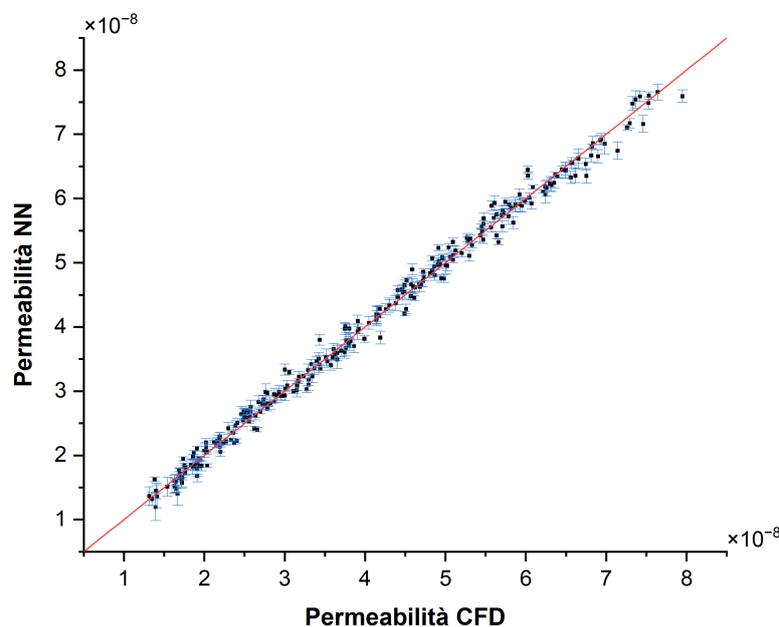
**Figura 6.4:** Focus della Fig.(6.3). E' possibile osservare come un minimo numero di neuroni sono sufficienti per superare le prestazioni del modello di regressione lineare multivariabile, Eq.(6.4).

Valori estremamente alti nel numero complessivo di neuroni possono determinare *overfitting* dei dati iniziali, causando quindi una perdita di generalità del modello data-driven. Pertanto, osservando i risultati in Fig.(6.3), il modello FFNN è stato allenato utilizzando l'architettura  $A$  ed il parametro  $hls = 12$ . E' stata utilizzata un'*activation function* di tipo RELU. Il modello così allenato è caratterizzato dalle seguenti metriche.

$$\begin{aligned} R^2 &= 0.993 \\ (MSE)^{0.5} &= 1.48455 \times 10^{-9} \end{aligned} \quad (6.7)$$

Il training del modello, seguendo l'approccio Leave One Out, è stato condotto venti volte, pertanto ogni valore stimato di permeabilità è stato predetto venti volte da un modello allenato con tutte le altre istanze del dataset. La Fig.(6.5) mostra il *parity plot* ottenuto dai risultati dell'algoritmo di regressione. L'errore medio del modello di regressione è del 3.05%. L'approccio Leave One Out richiede un

maggior numero di risorse computazionali ma è consigliabile in presenza di dataset a ridotta cardinalità.

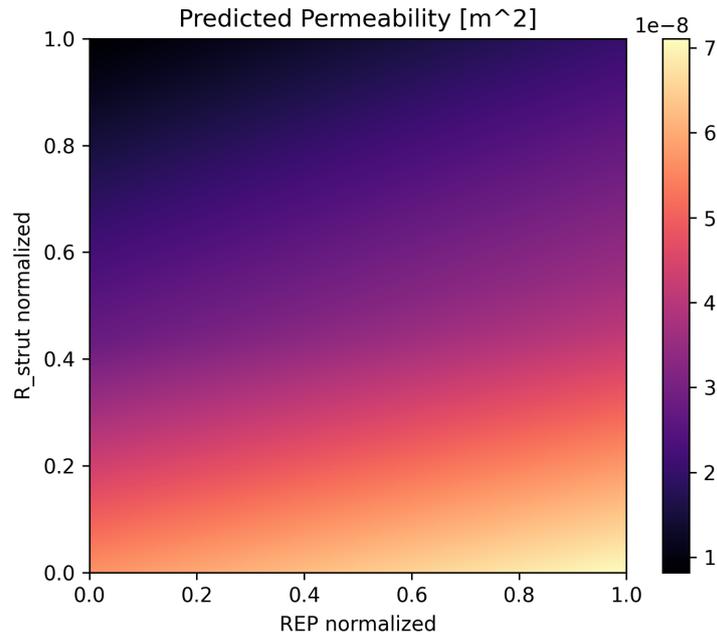


**Figura 6.5:** *Parity plot* della permeabilità per le geometrie OCF risolte nel dataset esteso. Confronto tra valori CFD e valori predetti dal modello data-driven, approccio LOO. L'errore medio è del 3.05%.

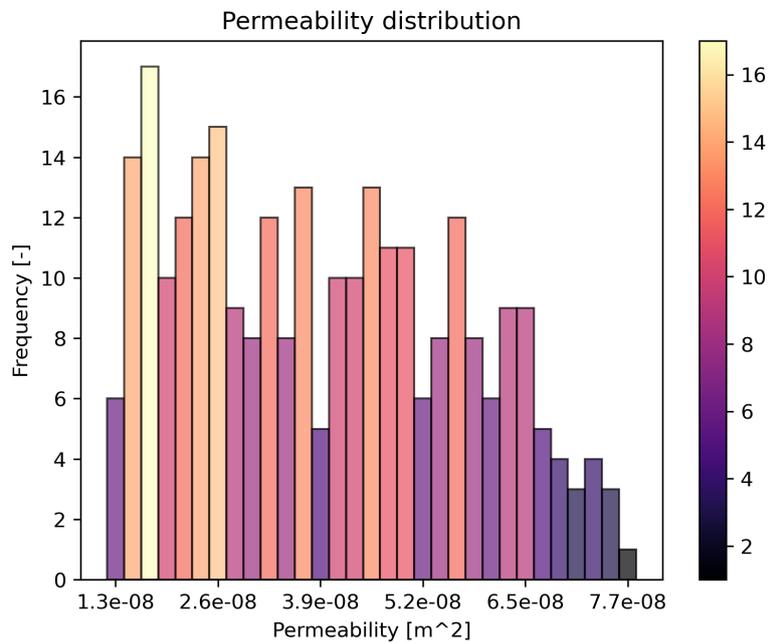
Completato il training del modello di regressione data-driven è possibile mappare completamente la permeabilità all'interno del range dei parametri geometrici descritti in Eq.(4.14). Sviluppato il modello il suo impiego è istantaneo e non richiede ulteriori risorse computazionali. Una slice bidimensionale della mappa di permeabilità per le geometrie descritte dai parametri geometrici descritti in Eq.(4.14) è rappresentata in Fig.(6.6), dove

$$\begin{aligned}\alpha_{norm} &= 0.75 \\ \beta_{norm} &= 0.75 \\ tv_{norm} &= 0.50\end{aligned}\tag{6.8}$$

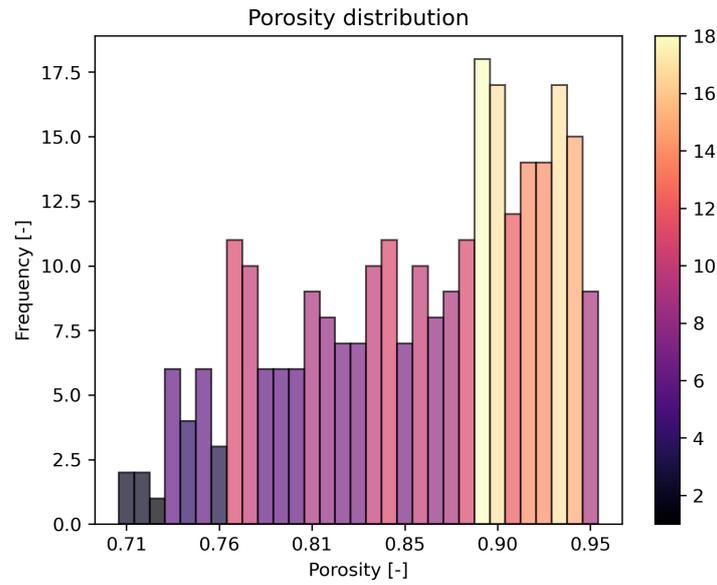
I risultati delle simulazioni eseguite sulle istanze del dataset esteso, in termini di permeabilità, sono riportati in Fig.(6.7) mentre caratterizzazione delle medesime schiume dal punto di vista geometrico è rappresentata dalle Fig.(6.8 e 6.9).



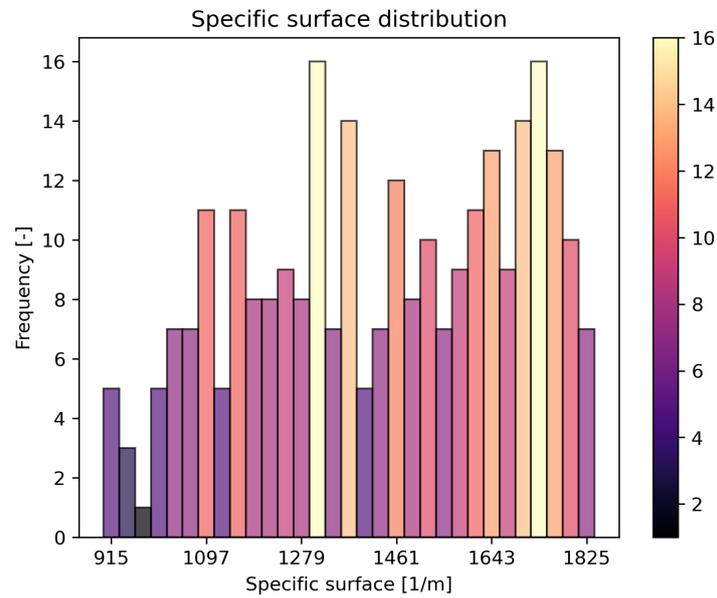
**Figura 6.6:** Slice bidimensionale della mappa di permeabilità ottenuta dal modello data-driven di regressione. I valori dei parametri sono normalizzati con *MinMaxScaler* nell'intervallo (0,1).



**Figura 6.7:** Distribuzione della permeabilità  $\kappa$ , calcolata risolvendo le simulazioni fluidodinamiche sulle istanze del dataset esteso.



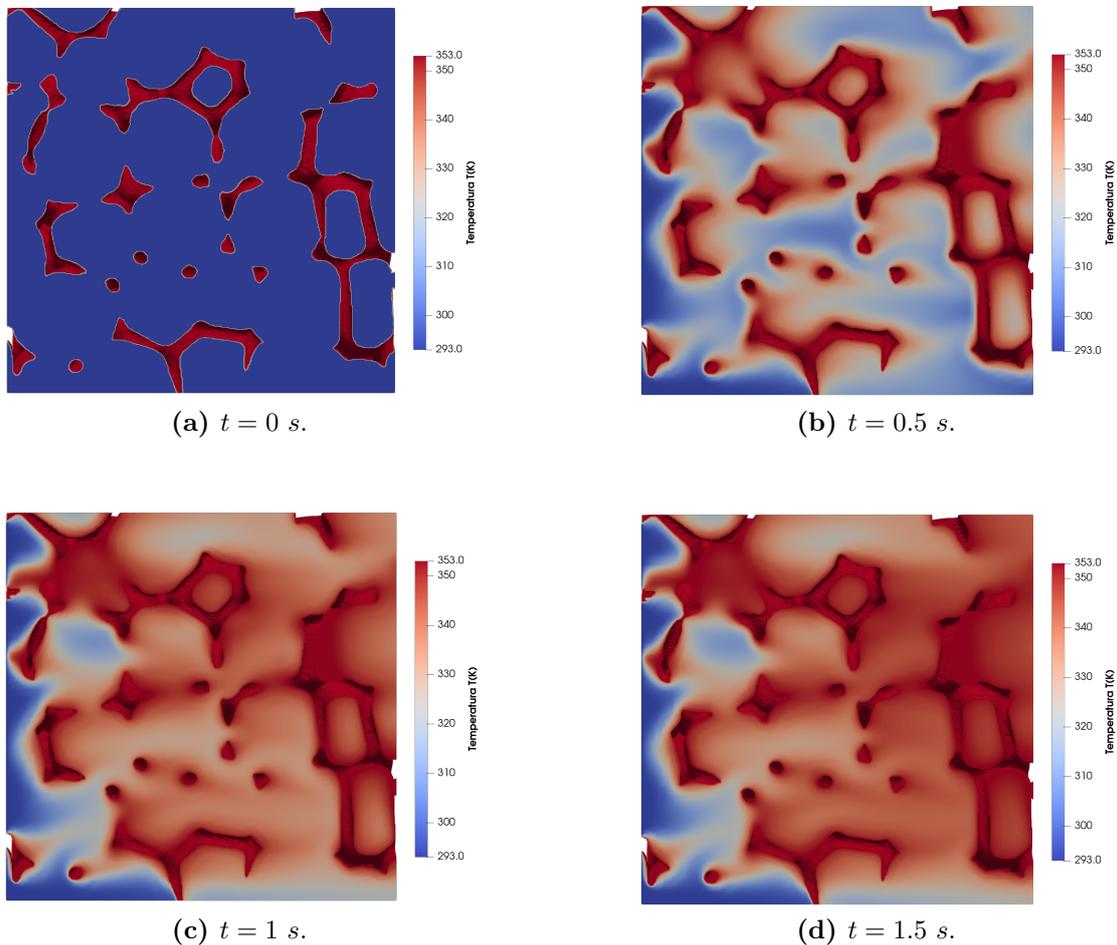
**Figura 6.8:** Distribuzione della porosità  $\epsilon$  delle istanze appartenenti al dataset esteso. Il valore di  $\epsilon$  è stato calcolato utilizzando le informazioni della mesh computazionale.



**Figura 6.9:** Distribuzione della superficie specifica  $S_v$  delle istanze appartenenti al dataset esteso. Il valore di  $S_v$  è stato calcolato utilizzando le informazioni della mesh computazionale.

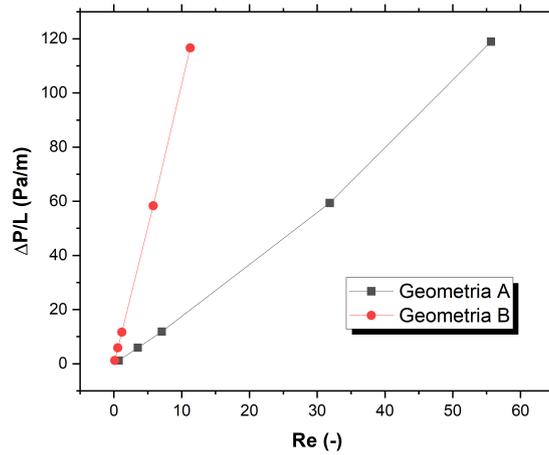
### 6.3 Sistema non isoterma

Concluso il lavoro sulla permeabilità sono stati condotti alcuni esperimenti con l'obiettivo di analizzare il comportamento delle schiume solide in presenza di una seconda fisica: lo scambio termico. Il semplice modello adottato nelle simulazioni prevede una temperatura costante sulla superficie della matrice solida. Tale condizione di bordo simula, seppur in modo approssimato, un sistema nel quale la geometria OCF, in materiale metallico, agisce come *heat sink* nei confronti di un generico *device* che necessita una continua rimozione di calore, operata per mezzo del fluido che scorre nella matrice solida. Il flusso fluido in contatto con la superficie solida della schiuma asporta calore, generando un profilo termico transitorio che raggiunge infine una condizione stazionaria come mostrato in Fig.(6.10).



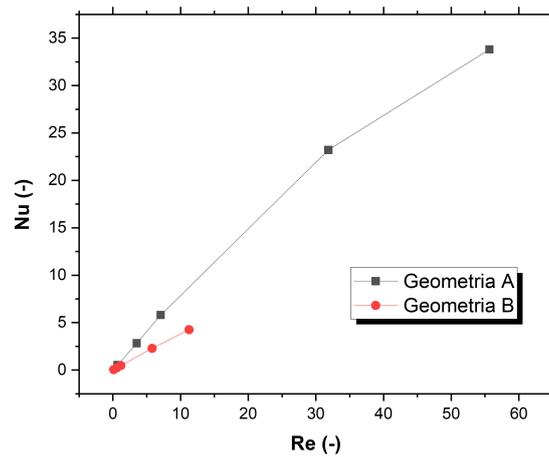
**Figura 6.10:** Evoluzione transitoria del profilo di temperatura nella fase fluida per una generica geometria OCF in una simulazione di scambio termico.

Le simulazioni condotte sulle microstrutture A, B permettono di ricavare due profili per le perdite di carico del flusso all'interno delle geometrie, rappresentati in Fig.(6.11). La differente microstruttura determina diversi profili per le perdite di carico. I profili sono lineari in presenza di validità della legge di Darcy. A parità di  $Re$  la microstruttura con strut più piccole ed elevata porosità garantisce minori perdite di carico.



**Figura 6.11:** Perdite di carico in funzione di  $Re$  per le geometrie A,B.

Le simulazioni di scambio termico restituiscono i profili di  $Nu$  al variare di  $Re$  e sono rappresentati in Fig.(6.12). La condizione al contorno scelta determina una migliore performance di scambio termico nella geometria A.



**Figura 6.12:**  $Nu$  in funzione di  $Re$  per le geometrie A,B.

## 6.4 Conclusioni

In questo lavoro è stato approfondito lo studio del campo di moto in geometrie OCF, costruite digitalmente mediante un algoritmo [8] capace di produrre risultati pienamente comparabili con reali tomografie di schiume a celle aperte. Proseguendo in questa direzione è stato sviluppato un codice Python originale, che implementa l'algoritmo di generazione delle geometrie in un più vasto *workflow end-to-end*, dalla definizione iniziale dei parametri alla risoluzione CFD mediante il software open-source **OpenFOAM**, con annesso postprocessing dei risultati. La flessibilità e l'efficienza messi a disposizione da tale strumento hanno supportato la costruzione e la risoluzione di dataset esplorativi di schiume OCF, evidenziando ancora una volta la complessità della microstruttura di tali geometrie. Tale complessità si riflette in una oggettiva difficoltà nella scelta di dimensioni caratteristiche e nella definizione di relazioni funzionali tra variabili significative per il sistema. In questo lavoro si è cercato di superare alcune di queste difficoltà rivolgendosi ad approcci innovativi per il settore dell'ingegneria chimica ovvero modelli *data-driven*, in primis reti neurali. Il problema di identificare una relazione funzionale tra i parametri del modello geometrico ed una grandezza rappresentativa del campo di moto ( $\kappa$ ) nelle geometrie OCF è stato così affrontato sia in termini classici (regressione lineare multivariabile) sia seguendo un'approccio basato su tecniche di *machine learning*. I risultati riportati nella sezione 6.2 mostrano come una rete di tipo FFNN riesca a costruire un modello di regressione *data-driven* estremamente efficace (Fig.6.5) e più performante rispetto alla classica regressione lineare (Fig.6.4). I modelli data-driven hanno contribuito in modo positivo anche nella fase di esclusione automatica delle geometrie non-fisiche con le ottime performance ottenute dal classificatore SVM e riportate nella *confusion matrix* (Eq.6.6).

I risultati promettenti testimoniati dal presente lavoro sull'uso delle reti neurali per lo studio di sistemi complessi, combinate alle simulazioni CFD, potrebbero essere applicati a sistemi caratterizzati dalla presenza di più fisiche. In presenza di altre fisiche, come il trasporto di massa o di calore in mezzi porosi; l'approccio basato su reti neurali e tecniche di *machine learning* esplorato in questo lavoro potrebbe rivelarsi ancora più prezioso, considerando l'evoluzione verso sistemi sempre più completi e necessariamente più complessi. Un modello di scambio termico è stato impostato nel capitolo 5. Performance termiche e perdite di carico in schiume a diversa microstruttura sono state calcolate mediante simulazioni CFD; ne sono esempio i profili rispettivi rappresentati in Fig.(6.12) e Fig.(6.11). In questo semplice modello le microstrutture ad alta porosità risultano le migliori in termini di performance termiche e perdite di carico. Tuttavia i presenti risultati non tengono in considerazione la differente inerzia termica delle *struts* che possono essere sottili o molto spesse.

Un futuro lavoro di ottimizzazione delle geometrie OCF rivolte allo scambio termico dovrà pertanto rivisitare il processo di meshing del dominio computazionale, tenendo in considerazione l'evoluzione dinamica del trasporto di calore all'interno delle *struts* della geometria. Un secondo vincolo imprescindibile, nel processo di ottimizzazione, sarà dato dalle proprietà meccaniche e strutturali della microstruttura che non sono state indagate in questo lavoro.



# Appendice A

## Python scripts nell'analisi del REV

Sono riportati in questa appendice i codici Python utilizzati per l'analisi del REV, citati nella sezione 4.2.

### A.1 Porespy script

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import porespy as ps
4 from pathlib import Path
5 import os, sys
6 from skimage import io
7 cwd = Path(os.getcwd())
8
9 binImg3d = io.imread('foamTest.tif')
10 binImg3d[binImg3d==0]=1
11 binImg3d[binImg3d==255]=0
12
13 # Porespy REV
14 print("Analisi del REV")
15 profile = ps.metrics.representative_elementary_volume(im=binImg3d,
16     npoints=20000)
17 print(f" ")
18 # Postprocess
19 intervals=10
20 points=intervals+1
21 V = (binImg3d.shape[0])**3
```

```

22 L = len(profile.volume)
23 refVolumes = np.array([0,0.05*V,0.1*V,0.15*V,0.20*V,0.25*V,0.3*V
    ,0.35*V,0.4*V,0.5*V,V], dtype=int)
24
25 N=np.zeros(intervals , dtype=int)
26 sqSum=np.zeros(intervals)
27 Sum=np.zeros(intervals)
28 s2=np.zeros(intervals)
29 stdDEV=np.zeros(intervals)
30 avg=np.zeros(intervals)
31 classIDX=np.full((intervals ,L-1),-1)
32 idx = np.linspace(0,L-1,L)
33
34 for it in range(0,intervals):
35     # Maschera per indici dei volumi nella classe it
36     mask = (profile.volume>refVolumes[it])&(profile.volume<=
refVolumes[it+1]).copy()
37     N[it] = len(idx[mask])
38     # Salvataggio indici nella matrice classIDX
39     classIDX[it ,0:N[it]]=idx[mask]
40     # Somma delle porosità nella classe it
41     Sum[it] = np.sum(profile.porosity[classIDX[it ,0:N[it]]])
42     # Somma dei quadrati delle porosità nella classe it
43     sqSum[it] = np.sum((profile.porosity[classIDX[it ,0:N[it]]])**2)
44     # Varianza Campionaria nella classe it
45     s2[it] = ( (sqSum[it]) -((Sum[it]**2)/N[it]) )/(N[it]-1)
46     # Media Campionaria nella classe it
47     avg[it] = Sum[it]/N[it]
48     # STD nella classe it#
49     stdDEV[it] = s2[it]**0.5
50
51 print(f"Cardinalità delle classi: {N}\n\n")
52 print(f"Somma dei quadrati: {sqSum}\n\n")
53 print(f"Somma dei dati: {Sum}\n\n")
54 print(f"Deviazione standard S delle classi: {stdDEV}\n\n")
55 print(f"Valore atteso delle classi: {avg}\n\n")
56
57 with open(cwd / 'revData20k.txt', 'w') as f:
58     f.write(f'Volumi random: {L}\n\n')
59     f.write(f'Cardinalità delle classi: {N}\n\n')
60     f.write(f'Somma dei quadrati= {sqSum}\n\n')
61     f.write(f'Somma dei dati in ogni classe: {Sum}\n\n')
62     f.write(f'Varianza= {s2}\n\n')
63     f.write(f'Deviazione standard= {stdDEV}\n\n')
64     f.write(f'Valore atteso= {avg}')
65
66 # Grafici
67 fig , ax = plt.subplots(1, 1, figsize=[8, 8])

```

```

68 ax.plot(profile.volume, profile.porosity, '.', markersize=2, color='
    black')
69 for i in range(0, intervals):
70     ax.plot(np.linspace(refVolumes[i], refVolumes[i+1], N[i]), np.ones(N
    [i]) * avg[i], 'r-', linewidth=2)
71     ax.plot(np.linspace(refVolumes[i], refVolumes[i+1], N[i]), np.ones(N
    [i]) * (avg[i] + stdDEV[i]), '—', linewidth=2, color='orange')
72     ax.plot(np.linspace(refVolumes[i], refVolumes[i+1], N[i]), np.ones(N
    [i]) * (avg[i] - stdDEV[i]), '—', linewidth=2, color='orange')
73 ax.set_title(f"Analisi del REV su {L} volumi random [+–S]")
74 ax.set_xlabel("Volume [voxel]")
75 ax.set_ylabel("Porosity [–]");
76 ax.set_ylim([0.75, 0.95])
77 plt.savefig("REV20k_img.pdf")
78 plt.show()

```

## A.2 Concentric volume script

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import porespy as ps
4 from pathlib import Path
5 import os, sys
6 from skimage import io
7
8 binImg3d = io.imread('foamTest.tif')
9 binImg3d[binImg3d==0]=1
10 binImg3d[binImg3d==255]=0
11 im = binImg3d
12 print(im.shape)
13
14 x=np.arange(5, im.shape[0], 5)
15 porosity=np.empty((len(x)))
16 for i, val in enumerate(x):
17     im2= im[int((im.shape[0]-val)/2.):int(im.shape[0]-(im.shape[0]-
    val)/2.), int((im.shape[0]-val)/2.):int(im.shape[0]-(im.shape[0]-
    val)/2.), int((im.shape[0]-val)/2.):int(im.shape[0]-(im.shape[0]-
    val)/2.)]
18     porosity[i]=np.mean(im2)
19     print(np.mean(im2))
20
21 plt.figure()
22 plt.plot(x, porosity, 'ok')
23 plt.show()
24 print(np.mean(im))

```



# Appendice B

## Algoritmo di automazione

In questa appendice sono riportati gli script Python utilizzati per costruire l'algoritmo di automazione `autoWorkflow`. Maggiori dettagli riguardo l'algoritmo sono forniti nella sezione 4.4.1.

### B.1 Input script

Il seguente script (`input.py`) permette all'utente di modificare i parametri utilizzati per costruire le geometrie del dataset. E' possibile variare la dimensione del REV da tagliare a partire dalla geometria completa, la dimensione della geometria completa ed i parametri di meshing come il numero di celle di `blockmesh` per *foam pore*. E' altresì possibile modificare la condizione di bordo per la simulazione di flusso ( $p_{in}$ ), la dimensione del dataset ed i PPI delle geometrie. Tali parametri saranno richiamati da altri script dell'algoritmo durante l'esecuzione.

```
1 # Definition of all MANUALLY CHANGABLE VARIABLES!  
2 # They are re-called in the other scripts from here.  
3  
4 # Mesh and CFD Parameters —  
5  
6 # Linear dimension of REV bounding box used in CFD. It should be  
7   chosen according to REV-analysis plots which are produced as  
8   output of autoWorkflow.  
9 REV = 360           # [voxel]  
10  
11 # Buffer per facilitare il meshing.  
12 buffer = 4         # [voxel]  
13  
14 # Grid independence results.  
15 bMcellsPerEdge = 20           # [bMcells]
```

```

14
15 # Inlet kinematic pressure
16 p_in = 0.000001 # [m^2/s^2]
17
18 # Geometry Parameters —
19
20 # Cardinality of the dataSet
21 DIM = 300 # [-]
22
23 # Linear dimension of original bounding box for spheres packing. I
    have used large dimensions to study the REV and the eventual
    border-effects.
24 wPAR = 500
25
26 # PPI constrain usedb to define the real geometrical dimension.
27 PPI = 16 # [-]

```

## B.2 Building script

Lo script di costruzione delle geometrie (`autoGeoGenArgPostTort.py`) è stato ottenuto come adattamento dello script sviluppato da Enrico Agostini mediante il software `PlugIn` [8]. Oltre al codice `PlugIn` sono state aggiunte altre funzioni necessarie per automatizzare la costruzione, il taglio, lo scaling delle geometrie e la compilazione delle `foamCases`.

Un prima parte di codice analizza il REV mediante l’approccio dei volumi concentrici e salva un’immagine consultabile a posteriori. Successivamente il codice ritaglia il REV a partire dalla geometria originale e scala la dimensione secondo il valore di PPI inserito come input. Infine sono cancellati i *files* intermedi per ottimizzare l’uso del disco.

```

1 from pathlib import Path
2 import xmltodict
3 import shutil as sh
4 import os, sys
5 import time
6 import launchPlugin as plm
7 import tess_foam as bf
8 import argparse
9 import postprocess as pp
10 # —
11 import matplotlib.pyplot as plt
12 import numpy as np
13 import porespy as ps
14 from skimage import io

```

```

15 from skimage.measure import marching_cubes
16 from skimage.measure import mesh_surface_area
17 import tifffile as tif
18 import trimesh
19 # —
20 from paraview.simple import *           # Source of library: conda-forge
21 from input import *                   # Import input variables
22 # —
23 def padding(img, padsize):
24     pad = np.pad(img, padsize, 'constant', constant_values=0)
25     return pad
26
27 def surf(image):
28     verts, faces, __, __ = marching_cubes(image, 0)
29     surface = mesh_surface_area(verts, faces)
30     return surface, verts, faces
31
32 def export_stl(verts, trifaces, filename):
33     mesh = trimesh.Trimesh(vertices=verts, faces=trifaces)
34     mesh_stl = mesh.export(filename)
35
36 def TranslateStl(inputFileName, translatedFileName, translate):
37     # create a new 'STL Reader'
38     myStl = STLReader(registrationName='myStl.stl', FileNames=[
inputFileName])
39     # create a new 'Transform'
40     transform1 = Transform(registrationName='Transform1', Input=myStl
)
41     transform1.Transform = 'Transform'
42     # Properties modified on transform1.Transform
43     transform1.Transform.Translate = translate
44     # save data
45     SaveData(translatedFileName, proxy=transform1, CellDataArrays=['
STLSolidLabeling'], FileType='Binary')
46
47 def ScaleStl(inputFileName, scaledFileName, scale):
48     myStl = STLReader(registrationName='myStl.stl', FileNames=[
inputFileName])
49     transform1 = Transform(registrationName='Transform1', Input=myStl
)
50     transform1.Transform = 'Transform'
51     transform1.Transform.Scale = scale
52     SaveData(scaledFileName, proxy=transform1, CellDataArrays=['
STLSolidLabeling'], FileType='Binary')
53
54 def readxml(xmlfile):
55     with open(xmlfile, 'r') as f:
56         parxml = f.read()
57         parameters = xmldict.parse(parxml)

```

```

58     for elm in parameters['par']:
59         parameters['par'][elm] = dict(parameters['par'][elm])
60     parameters['par'] = dict(parameters['par'])
61     parameters = parameters['par']
62     return parameters
63
64     ### Geometry generation (Code by Enrico Agostini). —
65     parser = argparse.ArgumentParser()
66
67     parser.add_argument('--name', type=str, required=True)
68     args = parser.parse_args()
69
70     t0 = time.time()
71
72     cwd = Path(os.getcwd())
73
74     par = readxml(args.name)
75
76     vx_res = float(par['postProc']['vx_res'])
77     print('\nWARNING: voxel resolution is set to', vx_res, ', you may need
78         to change it!!\n')
79
80     pOut = Path(cwd / par['paths']['pOut'])
81     pIn = Path(cwd / par['paths']['pIn'])
82
83     print('creating folder', pOut)
84
85     try:
86         os.makedirs(pOut, exist_ok = True)
87         print("Directory '%s' created successfully" %pOut.name)
88     except OSError as error:
89         print("Directory '%s' already exist")
90
91     aggSphDict = {
92         'Export' : str(pOut / 'sphPackCoord.txt'),
93         'OUTPREV' : str(pOut / 'sphPackPreview.fda'),
94         'OUT' : str(pOut / 'sphPackOutput.fda'),
95         'NB' : par['aggSph']['NB'],
96         'NBType' : 'Constante',
97         'Repul' : par['aggSph']['Repul'],
98         'R' : par['aggSph']['R'],
99         'RType' : 'Constante',
100        'Beta' : par['aggSph']['Beta'],
101        'Alpha' : par['aggSph']['Alpha'],
102        'val' : 255,
103        'Periodic' : True,
104        'W' : par['aggSph']['W']
105    }

```

```

106 |
107 | nodEdgDict = {
108 |     'FileNameFull': 'ballSticks.sb',
109 |     'FileName': 'ballSticks',
110 |     'IN': str(pOut / 'ballSticks.sb'),
111 |     'OUT': str(pOut / 'ballSticks.fda'),
112 |     'OUTPREV': str(pOut / 'prevBallSticks.fda')
113 | }
114 |
115 | morphDict = {
116 |     'IN': str(pOut / 'ballSticks.fda'),
117 |     'OUT': str(pOut / 'morphOut.fda'),
118 |     'OUTPREV': str(pOut / 'prevMorphOut.fda'),
119 |     'Operation': par['morph']['Operation'],
120 |     'tv': par['morph']['tv']
121 | }
122 |
123 | surfaceDict = {
124 |     'Export': str(pOut / 'surfaceArea3d.txt'),
125 |     'FileNameFull': 'morphOut.fda',
126 |     'FileName': 'morphOut',
127 |     'FileDir': str(pOut),
128 |     'IN': str(pOut / 'morphOut.fda'),
129 |     'OUT': str(pOut / 'morphOutSurf.fda'),
130 |     'OUTPREV': str(pOut / 'morphOutSurfPrev.fda')
131 | }
132 |
133 | tiffDict = {
134 |     'IN': str(pOut / 'morphOut.fda'),
135 |     'OUT': str(pOut / 'foamTest.tif'),
136 | }
137 |
138 | tortDict = {
139 |     'nb_samples': str(50),
140 |     'Weight': 'Dist',
141 |     'Export': str(pOut / 'GBTortuosity.txt'),
142 |     'FileNameFull': 'morphOut.fda',
143 |     'FileName': 'morphOut',
144 |     'FileDir': str(pOut),
145 |     'IN': str(pOut / 'morphOut.fda'),
146 |     'OUT': str(pOut / 'morphOut.fda'),
147 |     'OUTPREV': str(pOut / 'morphOutSurfPrev.fda')
148 | }
149 |
150 | # Launch sphere packing plugin —
151 |
152 | aggSph = plm.run_plugin('RMM_AggSph', aggSphDict, pIn, pOut)
153 | aggSph.copy_exe()
154 | aggSph.edit_xml()

```

```

155 aggSph.run_exe()
156
157 # Launch tessellation —
158
159 bf.bfoam(pOut, par['tess']['R_node'], par['tess']['R_strut'], iters=int(
    par['tess']['iter']), periodic=False)
160
161 # Launch ball&Sticks model creation
162 ballSticks = plm.run_plugin('CreateFDAFromRandModelTXT', nodEdgDict,
    pIn, pOut)
163 ballSticks.copy_exe()
164 ballSticks.edit_xml()
165 ballSticks.run_exe()
166
167 # Launch morphology operations —
168
169 morphOps = plm.run_plugin('BinaryGeodesicMorpho3D', morphDict, pIn,
    pOut)
170 morphOps.copy_exe()
171 morphOps.edit_xml()
172 morphOps.run_exe()
173
174 # Launch surface area calculation
175
176 #surfArea = plm.run_plugin('3DSurfaceArea', surfaceDict, pIn, pOut)
177 #surfArea.copy_exe()
178 #surfArea.edit_xml()
179 #surfArea.run_exe()
180
181 # Launch tiff creation
182
183 fdaToTIFF = plm.run_plugin('FDAToTIFF', tiffDict, pIn, pOut)
184 fdaToTIFF.copy_exe()
185 fdaToTIFF.edit_xml()
186 fdaToTIFF.run_exe()
187
188 # Launch Graph-Based Tortuosity
189
190 #tortGB = plm.run_plugin('GraphBased3D', tortDict, pIn, pOut)
191 #tortGB.copy_exe()
192 #tortGB.edit_xml()
193 #tortGB.run_exe()
194 ### —
195
196 ### REV analysis (Concentric volumes method) —
197 os.chdir(pOut)
198 cwd = Path(os.getcwd())
199
200 imRev = io.imread('foamTest.tif')

```

```

201 imRev[imRev==0]=1
202 imRev[imRev==255]=0
203 NP = int(par[ 'aggSph' ][ 'W' ])
204 x=np.arange(5,NP,5)
205 porosity=np.empty((len(x)))
206 for i,val in enumerate(x):
207     imRev2= imRev[ int((NP-val)/2.):int(NP-(NP-val)/2.) ,int((NP-val)
208         /2.):int(NP-(NP-val)/2.) ,int((NP-val)/2.):int(NP-(NP-val)/2.) ]
209     porosity[i]=np.mean(imRev2)
210 fig, ax = plt.subplots(1, 1, figsize=[6, 5])
211 ax.plot(x,porosity, '.', markersize=2,color='black')
212 ax.set_title(f"Volumi concentrici")
213 ax.set_xlabel("Lato box [voxel]")
214 ax.set_ylabel("Porosity [-]");
215 plt.savefig("IMG_porosityVSvolume.png")
216 ### —
217 ### Cut TIF file to decided REV dimension and generate an STL file
218 binImg3d = io.imread('foamTest.tif')
219
220 L = binImg3d.shape[0]
221 W=REV+2*buffer
222 startCut = int(L/2-W/2)
223 endCut = int(L/2+W/2)
224 im = binImg3d[startCut:endCut, startCut:endCut, startCut:endCut]
225 impad = padding(im,2).astype(np.uint8)
226 tif.imwrite( cwd / 'paddedFoam.tif', impad, dtype=np.uint8)
227 s, v, f = surf(impad)
228 export_stl(v,f, cwd / 'foamSTL.stl')
229 ### —
230
231 ### STL postprocessing with paraView python library —
232 with open(cwd / 'foamCells.txt') as f:
233     cells = f.read()
234 # foam cells per edge (referred to the complete geometry)
235 nfc = int(cells)**(1/3)
236 # foam cells per edge (referred to the REV-cutted geometry)
237 nfcRev = nfc*REV/wPAR
238 inchTOmm = 25.4 #1 [inch] = 25.4 [mm]
239 # 1 voxel = x [metri]
240 scale_factor = ((inchTOmm/PPI) / (REV/nfcRev))*(10**(-3))
241
242 with open(cwd / 'scaling.txt', 'w') as f:
243     f.write(f'{scale_factor}')
244 TranslateStl("foamSTL.stl", "translated.stl", [-(buffer+1), -(buffer+1)
245     , -(buffer+1)])
246 ScaleStl("translated.stl", "foamSTLpp.stl", [scale_factor, scale_factor,
247     scale_factor])
248 ### —

```

```

247
248 ### Delete useless files —
249 os.remove(cwd / 'paddedFoam.tif')
250 os.remove(cwd / 'foamSTL.stl')
251 os.remove(cwd / 'translated.stl')
252 os.remove(cwd / 'ballSticks.fda')
253 os.remove(cwd / 'ballSticks.sb')
254 os.remove(cwd / 'BinaryGeodesicMorpho3D.exe')
255 os.remove(cwd / 'BinaryGeodesicMorpho3DPAR.xml')
256 os.remove(cwd / 'CreateFDAFromRandModelTXT.exe')
257 os.remove(cwd / 'CreateFDAFromRandModelTXTPAR.xml')
258 os.remove(cwd / 'FDAToTIFF.exe')
259 os.remove(cwd / 'FDAToTIFFPAR.xml')
260 os.remove(cwd / 'morphOut.fda')
261 os.remove(cwd / 'prevBallSticks.fda')
262 os.remove(cwd / 'prevMorphOut.fda')
263 os.remove(cwd / 'RMM_AggSPh.exe')
264 os.remove(cwd / 'RMM_AggSPhPAR.xml')
265 os.remove(cwd / 'sphPackOutput.fda')
266
267 t1 = time.time()
268 print('Total runtime is', t1 - t0, 'seconds')
269 ### —

```

### B.3 Launcher script

Lo script responsabile dell'esecuzione dell'algoritmo di automazione (`launcher.py`) genera i *templates* per le geometrie di OCF, richiama ricorsivamente lo script di costruzione (`autoGeoGenArgPostTort.py`) e genera dei file output informativi. Inoltre il codice è anche responsabile di duplicare, a partire da un template, e compilare autonomamente le corrispondenti foamCases.

```

1 from pathlib import Path
2 import lxml.etree as ET
3 import numpy as np
4 import os, sys
5 import subprocess
6 import shutil
7 from input import *
8
9 cwd = Path(os.getcwd())
10
11 ### Summary table —
12 text = f' SAMPLE      ALPHA      BETA      REP      R_strut
      R_node      tv      W\n\n'

```

```

13 with open(cwd / 'foamSummaryTable.txt', 'w') as f:
14     f.write('\n SUMMARY TABLE\n\n')
15 with open(cwd / 'foamSummaryTable.txt', 'a') as f:
16     f.write(text)
17
18 # Definition of samples parameters using discrete uniform
19     distribution (random).
20
21 # ALPHA = [0.5,1] # Compromesso REV, variabilità ed eterogeneità
22 alphaRND = np.random.randint(50, high=101, size=DIM, dtype=int)/100
23
24 # BETA = [0.5,1] # Compromesso REV, variabilità ed eterogeneità
25 betaRND = np.random.randint(50, high=101, size=DIM, dtype=int)/100
26
27 # REP = [12,25] # 0.24 < REP/Dsphere < 0.50
28 repRND = np.random.randint(12, high=26, size=DIM, dtype=int)
29
30 # R_strut = [3,8] # 0.06 < R_strut/Dsphere < 0.16
31 rStrutRND = np.random.randint(3, high=9, size=DIM, dtype=int)
32
33 # tv = [10,20] # 0.20 < tv/Dsphere < 0.40
34 tvRND = np.random.randint(10, high=21, size=DIM, dtype=int)
35
36 for el in range(DIM):
37     # Progressive number of foams
38     nPAR = "{:3.0f}".format(el+1)
39
40     # 5 DOF scheme
41     alphaPAR = "{:1.2f}".format(round(alphaRND[el],2))
42     betaPAR = "{:1.2f}".format(round(betaRND[el],2))
43     repPAR = "{:2.0f}".format(repRND[el])
44     rStrutPAR = "{:1.0f}".format(rStrutRND[el])
45     tvPAR = "{:2.0f}".format(tvRND[el])
46
47     # Other parameters
48
49     # R_node = f(R_strut). f may be different. Chosen function:
50     R_node = R_strut + 1
51     rNodePAR = "{:2.0f}".format(1+rStrutRND[el])
52     # name of output XML file and value for pOut.
53     xmlName = f'foamTemplate{el+1}.xml'
54     # folder name (pIN)
55     folderName = f'foam{el+1}'
56
57     with open('template.xml', encoding="utf8") as f:
58         tree = ET.parse(f)
59         root = tree.getroot()
60
61         for elem in root.getiterator():

```

```

60         try:
61             elem.text = elem.text.replace('NTEMPLATE', folderName
62         )
63             elem.text = elem.text.replace('ALPHA_PAR', str(
64         alphaPAR))
65             elem.text = elem.text.replace('BETA_PAR', str(betaPAR
66         ))
67             elem.text = elem.text.replace('REP_PAR', str(repPAR))
68             elem.text = elem.text.replace('RSTRUT_PAR', str(
69         rStrutPAR))
70             elem.text = elem.text.replace('RNODE_PAR', str(
71         rNodePAR))
72             elem.text = elem.text.replace('TV_PAR', str(tvPAR))
73             elem.text = elem.text.replace('W_PAR', str(wPAR))
74         except AttributeError:
75             pass
76
77         tree.write(f'{xmlName}', xml_declaration=True, method='xml',
78         encoding="utf8")
79
80         text = f' {nPAR}           {alphaPAR}           {betaPAR}           {repPAR}
81         {rStrutPAR}           {rNodePAR}           {tvPAR}           {
82         wPAR}'
83         with open(cwd / 'foamSummaryTable.txt', 'a') as f:
84             f.write(text)
85             f.write('\n')
86
87     #### —
88
89     #### Run N-time autoGeoGenArgPostTort.py for each random template —
90     for el in range(DIM):
91         bashCommand = f'python autoGeoGenArgPostTort.py —name
92         foamTemplate{el+1}.xml'
93         os.system(bashCommand)
94
95     #### —
96
97     #### Building and compiling phase for each foamCase —
98     cwd = Path(os.getcwd())
99
100     NFCedge = np.zeros(DIM)
101     NFCedgeREV = np.zeros(DIM)
102     edgeFourNFC = np.zeros(DIM)
103     avgDc = np.zeros(DIM)
104     mx = np.zeros(DIM)
105     ramEstimate = np.zeros(DIM)
106
107     # samplesInfo.txt header
108     text = f' Edge4FC [voxels]           Cell/edge [#]           Cell/REV_edge [#]
109             Avg.Dc [mm]           bM_cells [#]           RAM_estimate [GB]\n\n'
110     with open(cwd / 'foamSamplesInfo.txt', 'w') as f:

```

```

99     f.write('\n CALCULATION OF SOME SAMPLES INFO\n\n')
100 with open(cwd / 'foamSamplesInfo.txt', 'a') as f:
101     f.write(text)
102
103 for el in range(DIM):
104     # Generation of a foamCase for foam1 and soon..
105     src = cwd / 'templateFoamCase'
106     dst = cwd / f'foamCase{el+1}'
107     shutil.copytree(src, dst)
108
109     # Copy-paste of final REV-size STL file, already pp for CFD
110     # purposes.
111     src = cwd / f'foam{el+1}' / 'foamSTLpp.stl'
112     dst = cwd / f'foamCase{el+1}' / 'constant' / 'triSurface' / '
113     foamSTLpp.stl'
114     shutil.copyfile(src, dst)
115     os.remove(cwd / f'foam{el+1}' / 'foamSTLpp.stl')
116
117     # Calculation and setting of Mx parameter in input.txt [bMcells/
118     # dimension]
119     with open(cwd / f'foamCase{el+1}' / 'foamCells.txt') as f:
120         cells = f.read()
121     with open(cwd / f'foamCase{el+1}' / 'scaling.txt') as f:
122         voxel_DIM = f.read()
123     # foam cells per edge (referred to the complete geometry)
124     NFCedge[el] = int(cells)**(1/3)
125     # foam cells per edge (referred to the REV-cutted geometry)
126     NFCedgeREV[el] = NFCedge[el]*REV/wPAR
127     edgeFourNFC[el] = 4*wPAR/NFCedge[el]
128     # average foam cells diameter [mm].
129     avgDc[el] = (float(voxel_DIM)*REV)/NFCedgeREV[el]
130     # Mx parameter for input.txt file in foamCases. Cells/edge in
131     # blockMesh background grid.
132     mx[el] = bMcellsPerEdge*NFCedgeREV[el]
133     with open(cwd / f'foamCase{el+1}' / 'input.txt', 'w') as f:
134         f.write(f'Mx {int(mx[el])};\n')
135         f.write(f'L {int(REV*float(voxel_DIM)*10**6)};\n')
136         f.write(f'p_in {p_in};\n')
137         f.write(f'shmLoc {(REV*float(voxel_DIM))/2};')
138     with open(cwd / f'foamCase{el+1}' / 'scaling.txt', 'w') as f:
139         f.write(f'{voxel_DIM}')
140     # RAM estimate using trend line from interpolated data.
141     ramEstimate[el] = 0.9269732018*(np.exp(0.0331713789*mx[el]))
142     with open(cwd / f'foamCase{el+1}' / 'run.sh', 'r') as f:
143         filedata = f.read()
144         filedata = filedata.replace('LABEL', f'sampl{el+1}')
145         filedata = filedata.replace('FCASE', f'foamCase{el+1}')
146         filedata = filedata.replace('RAM_SIZE', f'{int(ramEstimate[el]
147         )}')

```

```

143     with open(cwd / f'foamCase{el+1}' / 'run.sh', 'w') as f:
144         f.write(filedata)
145
146     NFCedge[el] = "{:2.1f}".format(NFCedge[el])
147     NFCedgeREV[el] = "{:2.1f}".format(NFCedgeREV[el])
148     avgDc[el] = "{:1.2f}".format(avgDc[el]*1000)
149     text = f' {int(edgeFourNFC[el])} {NFCedge[el]}
                {NFCedgeREV[el]} {avgDc[el]}
                {int(mx[el])} {int(ramEstimate[el])}'
150     with open(cwd / 'foamSamplesInfo.txt', 'a') as f:
151         f.write(text)
152         f.write('\n')
153 # —

```

## B.4 Postprocessing script

Lo script di postprocessing (`findRes.py`) è lanciato al termine di tutte le simulazioni e permette di raccogliere le informazioni chiave dai file grezzi generati da `OpenFOAM`. Un file informativo contenente i risultati per tutte le geometrie risolte è generato come output dello script.

```

1 import inspect
2 import os
3 import sys
4 import numpy as np
5 import shutil
6 from input import *
7
8 f = open("/home/abocca/labels.txt", 'w')
9 f.write('TIME, CELLS, FLUID_VOLUME, INPORE_UX, SURFACE_AREA, SPEC_SURFACE,
        HYDR_DIAMETER, REYNOLDS, DARCY_VELOCITY, POROSITY, PERMEABILITY, STATUS
        \n')
10 f = open("/home/abocca/output.txt", 'w')
11 f.write('TIME | CELLS | FLUID_VOLUME | INPORE_UX | SURFACE_AREA |
        SPEC_SURFACE | HYDR_DIAMETER | REYNOLDS | DARCY_VELOCITY |
        POROSITY | PERMEABILITY | STATUS\n')
12
13 p_out = 0 # [(m^2)/(s^2)] Kinematic pressure
14 nu = 0.89*10**(-6) # [(m^2)/s] Kinematic viscosity
15 DA = 9.869233*10**(-13) # [m^2] 1 Darcy
16
17 for el in range(DIM):
18     with open(f"/home/abocca/foamCase{el+1}/scaling.txt", 'r') as f:
19         temp3 = f.read()
20         L = float(temp3)*REV # [m]

```

```

21 vTot = L**3 # [m^3]
22 f = open(f"/home/abocca/foamCase{el+1}/log", 'r')
23 lines = f.readlines()
24 f.close()
25 for l in lines:
26     if 'Mesh OK' in l:
27         f = open(f"/home/abocca/foamCase{el+1}/log.postProcess_Uavg",
28                 'r')
29         lines = f.readlines()
30         f.close()
31         for l in lines:
32             if 'Create mesh for time' in l:
33                 p1 = l.find('=')
34                 solTime = int(l[(p1+2):(-1)])
35             if solTime != 0:
36                 f = open(f"/home/abocca/foamCase{el+1}/postProcessing/
37                         volFieldValue/0/volFieldValue.dat", 'r')
38                 lines = f.readlines()
39                 f.close()
40                 for l in lines:
41                     if 'Cells' in l:
42                         p1 = l.find(':')
43                         cells = int(l[(p1+2):(-1)])
44                     if 'Volume' in l:
45                         p1 = l.find(':')
46                         volume = float(l[(p1+2):(-1)])
47                     if (f'{solTime}' in l):
48                         p1 = l.find('(')
49                         avgUx = float(l[(p1+1):(p1+14)])
50
51                 f = open(f"/home/abocca/foamCase{el+1}/postProcessing/
52                         patchAverage(name=foamSTL,p)/0/surfaceFieldValue.dat", 'r')
53                 lines = f.readlines()
54                 f.close()
55                 for l in lines:
56                     if 'Area' in l:
57                         p1 = l.find(':')
58                         surfaceArea = float(l[(p1+6):(-1)])
59
60                 eps = volume/vTot # [-]
61                 sv = surfaceArea/(vTot) # [m]
62                 dh = (4*eps)/sv # [m]
63                 Re = (avgUx*dh)/nu # [-]
64                 vs = avgUx*eps # [m/s]
65                 k = (vs*nu)/((p_in-p_out)/L) # [m^2]
66                 status = 'OK'
67             elif (solTime == 0) or (solTime == 20000):
68                 solTime = -99
69                 cells = 0

```

```

67         volume = 0
68         avgUx = 0
69         surfaceArea = 0
70         eps = 0
71         sv = 0
72         dh = 0
73         Re = 0
74         vs = 0
75         k = 0
76         status = 'SOLVER FAILED'
77     elif 'Failed' in l:
78         solTime = -99
79         cells = 0
80         volume = 0
81         avgUx = 0
82         surfaceArea = 0
83         eps = 0
84         sv = 0
85         dh = 0
86         Re = 0
87         vs = 0
88         k = 0
89         status = 'MESH FAILED'
90
91     cells = "{:2.1e}".format(cells)
92     volume = "{:1.4e}".format(volume)
93     avgUx = "{:1.4e}".format(avgUx)
94     surfaceArea = "{:1.4e}".format(surfaceArea)
95     sv = "{:4.1f}".format(sv)
96     dh = "{:1.4e}".format(dh)
97     Re = "{:1.4e}".format(Re)
98     vs = "{:1.4e}".format(vs)
99     eps = "{:1.4f}".format(eps)
100    k = "{:1.4e}".format(k)
101    f = open("/home/abocca/labels.txt", 'a')
102    f.write(f'{int(solTime)},{ cells },{ volume },{ avgUx },{ surfaceArea },{ sv
103            },{ dh },{ Re },{ vs },{ eps },{ k },{ status}\n')
104    f = open("/home/abocca/output.txt", 'a')
105    f.write(f'{int(solTime)} , { cells} , { volume} , { avgUx} , {
106            surfaceArea} , { sv} , { dh} , { Re} , { vs} , { eps}
107            , { k} , { status} \n')

```

# Ringraziamenti

Si conclude, dopo quasi un'anno di lavoro, un percorso di tesi che mi ha coinvolto tanto dal punto di vista emotivo, quanto accademico.

Ringrazio il Prof. Daniele Marchisio per avermi trasmesso un profondo interesse riguardo la modellazione dei fenomeni di trasporto grazie alla sua esperienza e la sua professionalità, e per avermi dato la possibilità di intraprendere questo viaggio nel gruppo di ricerca *MuSyChEn* del Politecnico di Torino.

Ringrazio la Dott.ssa Agnese Marcato ed il Prof. Gianluca Boccardo per avermi accolto fin da subito nel gruppo e per avermi fornito tutto il background necessario per intraprendere questa ricerca. Vi ringrazio per la vostra infinita disponibilità e per il clima piacevole nel quale è stato svolto tutto il lavoro. Questa è stata, senza dubbio, una *preziosa* opportunità di crescita personale che porterò sempre con me.

Ringrazio la Prof.ssa Tania Cerquitelli per la disponibilità e l'interesse mostrato verso le tematiche della mia ricerca.

La consegna di questo elaborato rappresenta, a tutti gli effetti, il raggiungimento di un importante traguardo in occasione del quale desidero riservare alcuni preziosi ringraziamenti alle persone che hanno reso importanti questi cinque anni di preparazione universitaria.

Ringrazio, in primis, i miei cari amici Giacomo Sandrone e Simone Chiarla, con voi ho condiviso così tante risate in questi anni e così tanti momenti che spesso vorrei riavvolgere il nastro per ripercorrere quelle giornate spensierate. Molti giorni sono stati migliori in vostra compagnia, vi ringrazio tanto!

Vorrei inoltre estendere questo ringraziamento a tutti i miei cari amici di Alba e di Torino con i quali ho trascorso bellissimi momenti insieme che mi hanno arricchito di tanti *preziosi* ricordi.

Ringrazio la mia famiglia ed in particolare i miei genitori per avermi permesso di dedicarmi completamente agli studi in questi anni e per aver sempre accettato ogni mia decisione, sostenendomi incondizionatamente. Grazie a mia madre, Paola, per esserci sempre; nonostante ogni tanto abbiamo idee differenti apprezzo moltissimo tutto ciò che fai per me, e non potrei mai pensare il contrario. Grazie a mio padre, Renato, per non avermi fatto mancare mai nulla e per i tuoi *preziosi* insegnamenti sul rispetto e sulla gratitudine.

Desidero, infine, esprimere la mia profonda gratitudine a Giorgia, la mia fidanzata straordinaria, a cui dedico questa tesi. Grazie a te ho affrontato ogni sfida con entusiasmo, non ho mai camminato da solo e ho imparato a valorizzare ogni singolo passo del mio percorso. La tua presenza ha reso questo traguardo ancora più significativo. Averti al mio fianco mi dona sicurezza e una gioia indescrivibile. Sono infinitamente grato per tutto l'affetto ed il supporto che mi hai sempre dato.  
*Grazie!*

# Bibliografia

- [1] IEA. *Key World Energy Statistics 2021*. Rapp. tecn. IEA, Paris, 2021. URL: <https://www.iea.org/reports/key-world-energy-statistics-2021> (visitato il 07/10/2023) (cit. a p. 3).
- [2] IEA. *World CO2 emissions from fuel combustion by fuel*. Rapp. tecn. IEA, Paris, 2019. URL: <https://www.iea.org/data-and-statistics/charts/world-co2-emissions-from-fuel-combustion-by-fuel-1971-2019> (visitato il 07/10/2023) (cit. a p. 3).
- [3] IEA. *Greenhouse gas emissions by sector*. Rapp. tecn. IEA, Paris, 2019. URL: <https://www.iea.org/data-and-statistics/charts/greenhouse-gas-emissions-by-sector-2019-2> (visitato il 07/10/2023) (cit. a p. 3).
- [4] Anton Beloconi e Penelope Vounatsou. «Revised EU and WHO air quality thresholds: Where does Europe stand?» In: *Atmospheric Environment* (2023), p. 120110 (cit. a p. 3).
- [5] Michael F Ashby, Tony Evans, Norman A Fleck, JW Hutchinson, HNG Wadley e LJ Gibson. *Metal foams: a design guide*. Elsevier, 2000 (cit. alle pp. 4, 9, 10).
- [6] L-P Lefebvre, John Banhart e David C Dunand. «Porous metals and metallic foams: current status and recent developments». In: *Advanced engineering materials* 10.9 (2008), pp. 775–787 (cit. a p. 4).
- [7] Sooho Kim e Chang-Woo Lee. «A review on manufacturing and application of open-cell metal foam». In: *Procedia Materials Science* 4 (2014), pp. 305–309 (cit. alle pp. 4, 5, 13).
- [8] Daniele Marchisio, Gianluca Boccardo e Enrico Agostini. *Fluid dynamics and mass transfer in porous media: Modelling fluid flow and filtration inside open-cell foams*. eng. 2023 (cit. alle pp. 6, 8, 21, 63, 66, 69, 74, 93, 95, 104, 112).
- [9] Martyn V Twigg e James T Richardson. «Fundamentals and applications of structured ceramic foam catalysts». In: *Industrial & engineering chemistry research* 46.12 (2007), pp. 4166–4177 (cit. alle pp. 7, 14, 15).

- 
- [10] Dipen Kumar Rajak, Manoj Gupta, Dipen Kumar Rajak e Manoj Gupta. «Manufacturing methods of metal foams». In: *An Insight Into Metal Based Foams: Processing, Properties and Applications* (2020), pp. 39–52 (cit. alle pp. 9–12).
- [11] Gunnar Walther, Burghardt Kloeden, Bernd Kieback, R Poss, Y Bienvenu e JD Bartout. «A new PM process for manufacturing of alloyed foams for high temperature applications». In: *World PM* 4 (2010), p. 109 (cit. a p. 9).
- [12] Douglas T Queheillalt, Yasushi Katsumura e Haydn NG Wadley. «Synthesis of stochastic open cell Ni-based foams». In: *Scripta Materialia* 50.3 (2004), pp. 313–317 (cit. a p. 9).
- [13] Brock Partee, Scott J Hollister e Suman Das. «Selective laser sintering process optimization for layered manufacturing of CAPA® 6501 polycaprolactone bone tissue engineering scaffolds». In: (2006) (cit. a p. 10).
- [14] Andrew Kennedy. «Porous metals and metal foams made from powders». In: *Powder Metallurgy* 2 (2012), pp. 31–46 (cit. a p. 10).
- [15] Garrett Ryan, Abhay Pandit e Dimitrios Panagiotis Apatsidis. «Fabrication methods of porous metals for use in orthopaedic applications». In: *Biomaterials* 27.13 (2006), pp. 2651–2670 (cit. a p. 10).
- [16] Kevin Boomsma, Dimos Poulikakos e F Zwick. «Metal foams as compact high performance heat exchangers». In: *Mechanics of materials* 35.12 (2003), pp. 1161–1176 (cit. a p. 13).
- [17] Anna Gancarczyk et al. «Metal foams as novel catalyst support in environmental processes». In: *Catalysts* 9.7 (2019), p. 587 (cit. a p. 13).
- [18] Y Peng e JT Richardson. «Properties of ceramic foam catalyst supports: one-dimensional and two-dimensional heat transfer correlations». In: *Applied Catalysis A: General* 266.2 (2004), pp. 235–244 (cit. a p. 15).
- [19] E Meloni, M Caldera, V Palma, V Pignatelli e V Gerardi. «Soot abatement from biomass boilers by means of open-cell foams filters». In: *Renewable energy* 131 (2019), pp. 745–754 (cit. a p. 15).
- [20] IFPEN. *PlugIm website*. Rapp. tecn. IFPEN, Lione. URL: <https://plugim.fr> (visitato il 07/10/2023) (cit. a p. 21).
- [21] Chris Rycroft. *Voro++: A three-dimensional Voronoi cell library in C++*. Rapp. tecn. Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2009 (cit. a p. 21).
- [22] Giulia Ferri, Severine Humbert, Mathieu Digne, Jean-Marc Schweitzer e Maxime Moreaud. «Simulation of Large Aggregate Particles Systems with a New Morphological Model». In: *Image Analysis & Stereology* 40 (2021), pp. 71–84 (cit. alle pp. 21, 23).

- 
- [23] LorenSen We. «Marching cubes: A high resolution 3d surface construction algorithm». In: *Comput Graph* 21 (1987), pp. 163–169 (cit. a p. 22).
- [24] Zehua Guo, Zhongning Sun, Nan Zhang, Ming Ding e Shuai Shi. «CFD analysis of fluid flow and particle-to-fluid heat transfer in packed bed with radial layered configuration». In: *Chemical Engineering Science* 197 (2019), pp. 357–370 (cit. a p. 27).
- [25] Aurelien Geron Hands-On Machine Learning. *with Scikit-Learn and Tensor-Flow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2017 (cit. a p. 28).
- [26] Agnese Marcato, Gianluca Boccardo e Daniele Marchisio. «A computational workflow to study particle transport and filtration in porous media: Coupling CFD and deep learning». In: *Chemical Engineering Journal* 417 (2021), p. 128936 (cit. a p. 31).
- [27] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. alle pp. 31, 35).
- [28] Agnese Marcato. «Accoppiamento di CFD e Machine Learning: due casi studio nell’ingegneria di processo= Coupling CFD and Machine Learning: two case studies in process engineering». Tesi di dott. Politecnico di Torino, 2019 (cit. alle pp. 32, 33).
- [29] Marwan Darwish e Fadl Moukalled. *The finite volume method in computational fluid dynamics: an advanced introduction with OpenFOAM® and Matlab®*. Springer, 2016 (cit. a p. 37).
- [30] Fadl Moukalled, Luca Mangani, Marwan Darwish, F Moukalled, L Mangani e M Darwish. *The finite volume method*. Springer, 2016, pp. 103–135 (cit. alle pp. 37, 40–42).
- [31] F Moukalled, L Mangani, M Darwish, F Moukalled, L Mangani e M Darwish. «The discretization process». In: *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab* (2016), pp. 85–101 (cit. a p. 38).
- [32] F Moukalled, L Mangani, M Darwish, F Moukalled, L Mangani e M Darwish. «Solving the system of algebraic equations». In: *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab* (2016), pp. 303–364 (cit. alle pp. 43, 44).
- [33] F Moukalled, L Mangani, M Darwish, F Moukalled, L Mangani e M Darwish. «The Finite Volume Mesh». In: *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab* (2016), pp. 137–171 (cit. a p. 45).

- [34] F Moukalled, L Mangani, M Darwish, F Moukalled, L Mangani e M Darwish. «Spatial Discretization: The Diffusion Term». In: *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab* (2016), pp. 211–271 (cit. alle pp. 46, 47).
- [35] F Moukalled, L Mangani, M Darwish, F Moukalled, L Mangani e M Darwish. «Discretization of the Convection Term». In: *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab* (2016), pp. 365–427 (cit. alle pp. 48, 49, 51).
- [36] *OpenFOAM website*. Rapp. tecn. URL: <https://openfoam.org> (visitato il 07/10/2023) (cit. a p. 52).
- [37] *Computational resources provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino*. Rapp. tecn. URL: <http://hpc.polito.it> (visitato il 07/10/2023) (cit. a p. 56).
- [38] Yehuda Bachmat e Jacob Bear. «Macroscopic modelling of transport phenomena in porous media. 1: The continuum approach». In: *Transport in porous media 1* (1986), pp. 213–240 (cit. alle pp. 66, 67).
- [39] Stephen Whitaker. *The method of volume averaging*. Vol. 13. Springer Science & Business Media, 1998 (cit. a p. 66).
- [40] *Quantitative Image Analysis of Porous Materials*. Rapp. tecn. URL: <https://porespy.org> (visitato il 07/10/2023) (cit. a p. 67).
- [41] James Ahrens, Berk Geveci e Charles Law. «ParaView: An End-User Tool for Large Data Visualization». In: *Visualization Handbook*. ISBN 978-0123875822. Elsevier, 2005 (cit. a p. 81).