ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Ingegneria industriale

Corso di Laurea in
INGEGNERIA AEROSPAZIALE

Sviluppo di un Framework Automatizzato Blender-OpenFOAM per Morphing parametrici e simulazioni CFD

Tesi di Laurea in
ING-IND/06: Laboratorio di Aerodinamica Sperimentale

Relatore
Prof. Guglielmo Minelli

Presentata da
Diego Dante Aurelio Dascola

Sessione Marzo 2026
Anno Accademico 2024/2025

Sommario

Il presente lavoro di tesi propone lo sviluppo e la validazione di un framework automatizzato open-source per l'analisi aerodinamica, volto a superare i limiti intrinseci dell'intervento manuale nella conduzione di intere campagne di simulazione. L'architettura sviluppata integra in un'unica pipeline in linguaggio Python la versatilità della modellazione parametrica di Blender con il solutore CFD OpenFOAM.

L'innovazione centrale della ricerca risiede nell'implementazione di una strategia di Design of Experiments (DOE) basata sull'algoritmo Latin Hypercube Sampling (LHS), ottimizzato mediante criteri di Centered Discrepancy e Maximin. Tale approccio garantisce una copertura space-filling del dominio parametrico superiore ai metodi Monte Carlo e Full Factorial, permettendo di mappare con precisione la risposta aerodinamica con un numero ridotto di osservazioni.

Il workflow gestisce autonomamente l'intero ciclo operativo: dalla generazione statistica dei punti di design alla deformazione geometrica tramite Shape Keys, fino al setup dei casi e la risoluzione numerica.

La robustezza della pipeline è stata validata attraverso lo studio di un profilo alare NACA 4412, soggetto a variazioni parametriche di spessore e deflessione del bordo d'uscita. I risultati hanno confermato l'elevata stabilità numerica del sistema e la sua capacità di eliminare l'errore umano nella gestione massiva dei dati.

Indice

Elenco delle figure

Elenco delle tabelle

Acronimi

API	Application Programming Interface
CFD	Computational Fluid Dynamics
CAD	Computer Aided Design
DOE	Design Of Experiments
LHS	Latin Hypercube Sampling
STL	Stereolithography

Capitolo 1

Introduzione

Negli ultimi decenni, la simulazione CFD ha assunto un ruolo sempre più centrale nelle applicazioni ingegneristiche, in particolare per l'analisi e l'ottimizzazione aerodinamica. Tuttavia, l'intero processo di simulazione (dalla creazione della geometria al post-processing) può risultare particolarmente oneroso in termini di tempo e intervento manuale, soprattutto se ripetuto per numerose iterazioni. Questo ha stimolato, sia in ambito accademico sia industriale, lo sviluppo di catene di automazione in grado di ridurre significativamente il carico di lavoro umano e aumentare la ripetibilità delle analisi.

Tra i primi contributi in letteratura si distingue il lavoro di Farah *et al.* [1], in cui viene sviluppato un POC per una catena di calcolo su OpenFOAM focalizzandosi tuttavia su flussi esterni intorno a geometrie canonizzate. Sebbene il sistema sia in grado di generare autonomamente i file di setup e il dominio computazionale, esso presenta il limite critico di vincolare la generazione geometrica all'interno del framework stesso. Tale approccio esclude l'impiego di file .STL esterni, risultando distante dalle reali necessità dell'ingegneria moderna, dove il disegno CAD rappresenta un punto di partenza imprescindibile.

Spostando l'attenzione su contesti applicativi più specifici, Mangini *et al.* [2] hanno introdotto `T-WorkFlow`, uno strumento per l'analisi del flusso nei radiatori di veicoli da competizione che integra OpenFOAM e FreeCAD. Nonostante l'integrazione CAD–CFD tramite GUI dedicata rappresenti un avanzamento significativo, la natura di `T-WorkFlow` come soluzione personalizzata per le esigenze di Tatuus Racing ne limita la generalizzabilità a problemi aerodinamici di diversa natura.

Parallelamente alla necessità di integrazione CAD, la ricerca si è concentrata sulla gestione programmatica dei casi di simulazione. A tal proposito, `caseFoam` di Scheufler *et al.* [3] ha esteso le potenzialità della libreria PyFOAM per semplificare la creazione e manipolazione dei casi OpenFOAM. Tuttavia, la dipendenza da librerie di terze parti ne ha compromesso la compatibilità con le versioni più recenti del solutore.

Per superare le limitazioni dei workflow sopra-citati e, contemporaneamente, introdurre una libertà di modellazione superiore ai metodi CAD tradizionali, si è esplorato l'impiego di strumenti nati per la computer grafica. A tal proposito, incisivo è stato il lavoro di Ryden [4] il quale, attraverso il progetto `BlenderFoam`, ha validato l'efficacia dell'ecosistema open-source integrando Blender e OpenFOAM tramite *Lattice Deformers* per la modifica geometrica. Pur basandosi sulla flessibilità dei modificatori di Blender, il metodo descritto in BlenderFoam si focalizza primariamente su una strategia di ottimizzazione basata su algoritmi genetici o di ricerca locale, risultando meno rigoroso nella fase preliminare di investigazione dello spazio di design. In particolare, emerge una lacuna nel trattamento statistico dei parametri di input: laddove il progetto analizzato procede verso una ricerca iterativa del profilo ottimale, la necessità di una mappatura sistematica del dominio attraverso un DOE strutturato viene trascurata. Questo approccio, pur funzionale alla ricerca di un ottimo locale, espone il processo al rischio di una scarsa copertura dello spazio di design, rendendo difficile l'identificazione delle sensibilità globali del profilo alle diverse variazioni geometriche.

In questo scenario, il presente lavoro si propone di risolvere le criticità metodologiche precedentemente identificate, sviluppando una pipeline basata interamente su software open-source. Questa scelta mira a garantire la totale trasparenza e riproducibilità del flusso di lavoro, permettendone la libera personalizzazione attraverso script Python che legano la deformazione geometrica in Blender alla potenza di calcolo di OpenFOAM. L'obiettivo centrale è trasformare la ricerca "per tentativi" in un processo scientifico, dove ogni variante della geometria non è scelta casualmente, ma segue un piano matematico preciso. A tal fine, la metodologia proposta introduce l'impiego di algoritmi di Latin Hypercube Sampling (LHS) ottimizzati secondo il criterio della Centered Discrepancy (CD) e della massima distanza euleriana.

A differenza degli approcci iterativi locali, tale strategia di campionamento garantisce una copertura omogenea e *space-filling* del dominio parametrico, permettendo di mappare con precisione l'influenza delle deformazioni geometriche sui coefficienti aerodinamici con un numero ridotto di simulazioni OpenFOAM. Il workflow risultante vuole dunque puntare alla ricerca di un ottimo puntuale, ma fornisce una base dati solida e riproducibile, grazie a un sistema di gestione dei seed algoritmici, propedeutica alla costruzione di futuri modelli di ottimizzazione e alla piena comprensione della fisica del problema.

Capitolo 2

Metodologia: Design of Experiments

La filosofia alla base del Design of Experiments moderno risiede nella capacità di estrarre la massima quantità di informazioni con il minor numero possibile di osservazioni. Mentre nel secolo scorso l'enfasi era posta sulla randomizzazione e sulla replicazione per mitigare l'errore sperimentale, l'avvento dei computer experiment ha introdotto il concetto di esperimento deterministico: una simulazione che, a parità di input, produce sempre lo stesso output. Questa natura deterministica rende le repliche superflue e sposta l'obiettivo verso la *space-filling*, ovvero la capacità di distribuire i punti in modo uniforme per catturare non linearità, interazioni complesse e discontinuità della funzione di risposta.

L'esplorazione dello spazio di design, dunque, è il primo e più fondamentale obiettivo di qualsiasi DOE. Senza una conoscenza a priori del comportamento del sistema, la distribuzione dei punti deve essere tale da non trascurare alcuna regione del dominio, evitando che zone potenzialmente critiche rimangano inesplorate. Questo approccio permette l'identificazione delle sensibilità, ovvero la comprensione di come le variazioni nei parametri di input influenzino l'output. In molti sistemi ingegneristici reali vige il principio della "sparsità degli effetti", secondo cui solo una piccola frazione delle variabili di input domina la risposta del sistema; un DOE ben progettato deve quindi essere in grado di isolare questi fattori dominanti proiettando correttamente il design in dimensioni inferiori.

Infine, la riduzione del numero di simulazioni necessarie è l'imperativo economico che guida la ricerca di nuovi metodi di campionamento. Poiché una singola simulazione fluidodinamica può richiedere ore o giorni di calcolo, l'efficienza nel posizionamento dei punti diventa il fattore determinante per la fattibilità di un progetto di ottimizzazione o di analisi di affidabilità.

2.1 Problematiche del Campionamento Tradizionale

Per comprendere il valore dell'approccio proposto, è necessario analizzare il fallimento delle tecniche di campionamento convenzionali quando applicate a problemi moderni. Il campionamento Full Factorial, basato su una griglia regolare di punti, rappresenta l'approccio più intuitivo ma anche il più inefficiente all'aumentare delle variabili [5].

La cosiddetta *curse of dimensionality* [6] descrive l'esplosione esponenziale del volume dello spazio di ricerca e, di conseguenza, del numero di campioni necessari per mantenere una densità costante. Oltre al costo, i design fattoriali soffrono di un grave problema di "collasso": se una variabile si rivela irrilevante, i punti della griglia si sovrappongono perfettamente quando proiettati sulle altre dimensioni, fornendo informazioni ridondanti e sprecando risorse preziose.

Dall'altro lato dello spettro, il campionamento puramente casuale (*Monte Carlo*) tenta di evitare la regolarità delle griglie, ma introduce problemi di natura stocastica. Poiché i punti sono scelti in modo indipendente, non esiste un meccanismo che impedisca a due o più campioni di trovarsi estremamente vicini tra loro, creando dei "cluster". Allo stesso modo, ampie zone del dominio possono rimanere completamente vuote, portando a una stima imprecisa della funzione di risposta in quelle aree [7].

2.2 Il Campionamento Latin Hypercube (LHS)

Il campionamento Latin Hypercube si pone come una soluzione intermedia superiore, combinando la distribuzione capillare delle griglie con l'efficienza e la flessibilità dei metodi stocastici. Il cuore della teoria risiede nella stratificazione marginale, un concetto che garantisce che ogni variabile di input sia rappresentata in tutto il suo range operativo.

Il concetto di "Hypercube" estende la logica del Latin Square a d dimensioni. Immaginando un dominio d-dimensionale normalizzato, l'LHS divide l'intervallo di ciascuna variabile in N sotto-intervalli di uguale probabilità. La restrizione fondamentale dell'LHS è che ogni riga e ogni colonna (o meglio, ogni strato marginale lungo ogni asse) deve contenere esattamente un punto campionario, come mostrato in Figura 2.1.

Matematicamente, per generare un campione di N punti in d dimensioni, si procede come segue:

1. Si generano d permutazioni casuali dei primi N numeri interi:
   $\pi_j = (\pi_{j,1}, \pi_{j,2}, \ldots, \pi_{j,N})$ per ogni dimensione $j = 1, \ldots, d$.
2. Si selezionano valori casuali uniformi $U_{i,j} \in [0, 1)$ per ogni punto i e dimensione j.

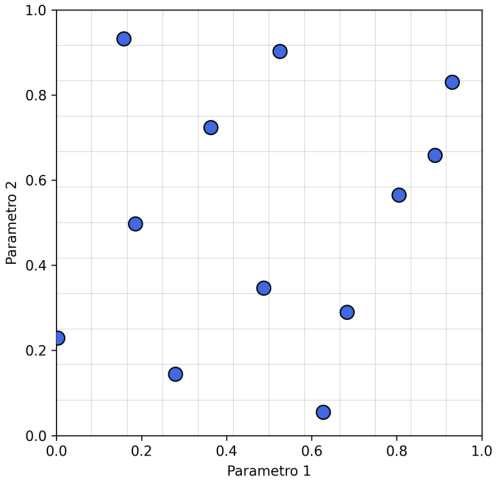


Figura 2.1: Rappresentazione bidimensionale di un campionamento LHS.

3. La coordinata del punto i nella dimensione j è calcolata come:

$$x_{i,j} = \frac{\pi_{j,i} - 1 + U_{i,j}}{N} \tag{2.1}$$

Questa struttura garantisce che la proiezione dei punti su ciascuna coordinata sia uniforme, eliminando il rischio di campionare valori ridondanti per una singola variabile [8].

2.3 Ottimizzazione del Campionamento

Sebbene l'LHS standard garantisca l'uniformità sulle proiezioni unidimensionali, esso non offre alcuna garanzia sulla distribuzione dei punti nello spazio multidimensionale. Un campionamento LHS generato casualmente potrebbe accidentalmente disporre tutti i punti lungo la diagonale principale dell'ipercubo, rispettando le restrizioni di riga e colonna ma lasciando gran parte del volume dello spazio vuoto. Per questo motivo, il valore aggiunto dell'approccio risiede nell'ottimizzazione del campionamento attraverso criteri geometrici e statistici rigorosi.

Il primo pilastro di tale ottimizzazione è il criterio della Centered Discrepancy (CD). La discrepanza è una misura quantitativa della non-uniformità di un insieme di punti che confronta la distribuzione empirica dei campioni con la distribuzione uniforme ideale nel cubo unitario. Il criterio della Centered Discrepancy è considerato uno dei più robusti perché corregge i difetti della discrepanza "star" tradizionale, la quale tende ad essere eccessivamente influenzata dalla scelta dell'origine del sistema di coordinate [9].

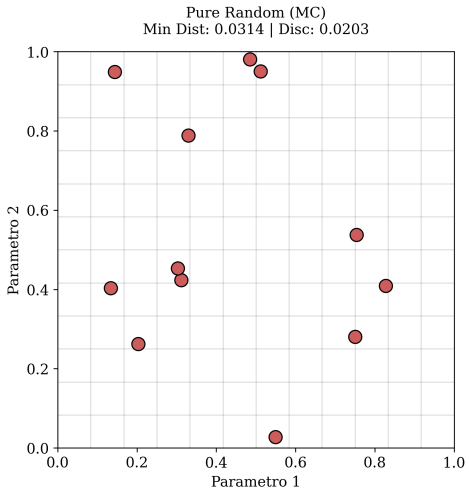
Minimizzare il valore di discrepanza significa ottenere un campionamento che non privilegia alcuna zona del dominio, garantendo una copertura "giusta" e uniforme

che include efficacemente i bordi dello spazio di design.

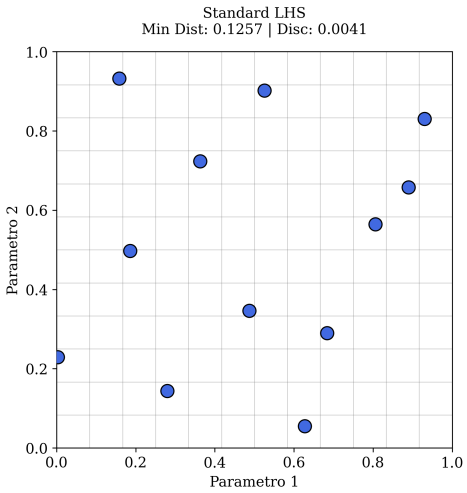
Parallelamente alla distribuzione globale, è fondamentale considerare la separazione locale tra i punti tramite il criterio Maximin (o algoritmo di Lloyd). L'obiettivo è massimizzare la distanza minima tra qualsiasi coppia di punti nel design. L'algoritmo cerca di allontanare i punti il più possibile per evitare la ridondanza di informazioni. Se due campioni sono molto vicini, essi forniranno dati quasi identici sul comportamento del sistema, sprecando una preziosa simulazione. Un design Maximin "spinge" i punti verso i confini e negli spazi vuoti, assicurando che ogni nuova osservazione esplori una regione dello spazio di design che è sostanzialmente diversa da quelle già investigate.

Per identificare la strategia più idonea alla mappatura aerodinamica, è stato condotto un confronto bidimensionale qualitativo tra le diverse tecniche di campionamento elencate. Come mostrato in Figura 2.2, il metodo Monte Carlo (Fig. 2.2a) evidenzia i limiti intrinseci della casualità pura, manifestando fenomeni di *clustering* e lasciando ampie porzioni del dominio del tutto inesplorate. Il passaggio al Latin Hypercube Sampling standard (Fig. 2.2b) introduce un primo livello di regolarità grazie alla stratificazione marginale, riducendo sensibilmente la discrepanza, pur non garantendo una separazione ottimale tra i punti. Tale aspetto viene invece massimizzato dall'algoritmo Lloyd (Fig. 2.2c), il quale, agendo secondo il criterio Maximin, incrementa la distanza minima tra i campioni per evitare ridondanze informative.

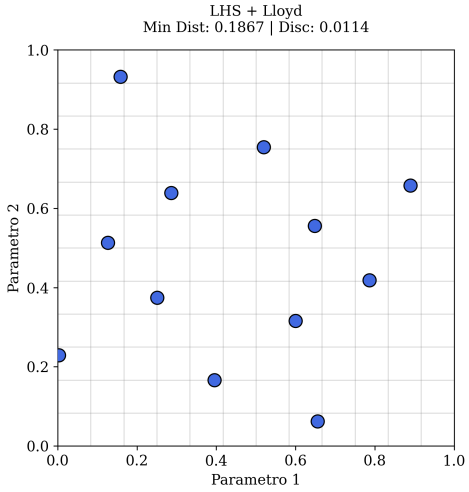
Tuttavia, per le finalità del presente lavoro, la configurazione LHS + Random-CD (Fig. 2.2d) è risultata la più efficiente. Minimizzando esplicitamente la discrepanza, questo approccio garantisce la copertura più omogenea e bilanciata dello spazio di design. La capacità di questo metodo di non privilegiare alcuna zona del dominio assicura che un eventuale modello di ottimizzazione, costruito successivamente sui dati CFD, sia in grado di catturare le non-linearità della risposta aerodinamica con un errore di approssimazione uniformemente distribuito. Si sottolinea che il processo di ottimizzazione per la ricerca del set di punti a minima discrepanza è stato gestito computazionalmente dal modulo `scipy.stats.qmc` della libreria `SciPy`, che integra nativamente il procedimento matematico proposto da Zhou *et al.* [10].



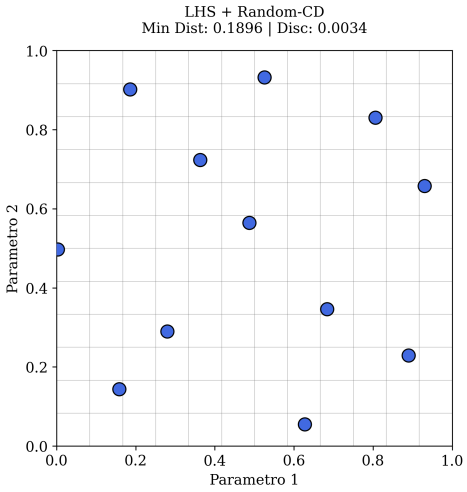
(a) Monte Carlo (MC)



(b) LHS Standard



(c) LHS + Lloyd (Maximin)



(d) LHS + Random-CD

Figura 2.2: Confronto qualitativo tra diverse strategie di campionamento per $N = 12$ campioni e $d = 2$ parametri: (a) campionamento Monte Carlo (MC); (b) Latin Hypercube Sampling (LHS) standard; (c) LHS ottimizzato con criterio Maximin (algoritmo di Lloyd); (d) LHS ottimizzato tramite Centered Discrepancy (random-cd).

Capitolo 3

Risultati: Implementazione di un workflow automatizzato

Lo sviluppo del workflow oggetto della presente trattazione si fonda su un'architettura modulare concepita per automatizzare il ciclo iterativo tra generazione di design, modellazione geometrica e analisi fluidodinamica. La scelta dei software è stata guidata dalla necessità di operare in un ecosistema interamente *open-source*, garantendo una comunicazione efficace tra questi. Come illustrato nello schema in Figura 3.2, il cuore dell'intero sistema è rappresentato dal linguaggio Python (v3.10+). Tale scelta è motivata dalla disponibilità di librerie scientifiche avanzate e dalla capacità del linguaggio di interfacciarsi in modalità headless con i solutori e i motori di modellazione.

Per quanto concerne il comparto software, si è optato per l'impiego di Blender (v4.1) per la modellazione geometrica e di OpenFOAM (v2406) come solutore CFD. È opportuno sottolineare che la selezione di queste specifiche versioni non è casuale, bensì risponde a un'esigenza di continuità metodologica e tecnica con le attività di ricerca intraprese durante il precedente periodo di tirocinio, assicurando la piena compatibilità degli script sviluppati e la stabilità delle API coinvolte. L'integrazione numerica e statistica è invece affidata a un ecosistema di librerie consolidate: `NumPy` per la manipolazione di array multidimensionali, `SciPy` per l'implementazione del campionamento Latin Hypercube, `Pandas` per la gestione strutturata dei database dei punti di design, e infine `Matplotlib` e `Seaborn` per la generazione di output grafici.

L'intero workflow è organizzato all'interno di una cartella principale denominata `Tesi`, la cui gerarchia è stata progettata per riflettere la separazione funzionale tra i domini di competenza del workflow (Figura 3.1). La struttura si articola in quattro cartelle principali:

- `01_Geometry/`: preposto alla conservazione del file .blend e alla gestione delle

varianti geometriche generate;

- `02_Simulation/`: ambiente dedicato ai casi CFD, basato su una logica di *templating* che garantisce l'omogeneità delle condizioni di calcolo;
- `03_Data`: archivio dei dati in ingresso e in uscita, dove risiede il file di controllo dei punti di design;
- `scripts/`: contiene tutti gli scripts Python utili ad eseguire la pipeline.

Completano il workflow una serie di script shell (`run.sh`, `clear_case.sh`) necessari per l'esecuzione della pipeline e la manutenzione dell'ambiente di calcolo.

Un requisito primario nello sviluppo è stata la portabilità del sistema. Per evitare la necessità di riconfigurare manualmente i percorsi assoluti a ogni cambio di postazione di lavoro, è stato implementato un modulo di configurazione dinamica denominato `config.py`.

Sfruttando le capacità della libreria standard di Python `os`, il workflow è in grado di auto-identificare la propria posizione nel file system definendo una cartella base denominata `$BASE_DIR`. A partire da questo riferimento, tutti i collegamenti alle sottocartelle e ai file necessari vengono generati in modo relativo. Grazie a questa architettura, l'utente è sollevato dall'onere di inserire manualmente i link alle risorse; l'unico parametro esterno richiesto rimane il percorso dell'eseguibile di Blender, variabile in base all'installazione locale del software.

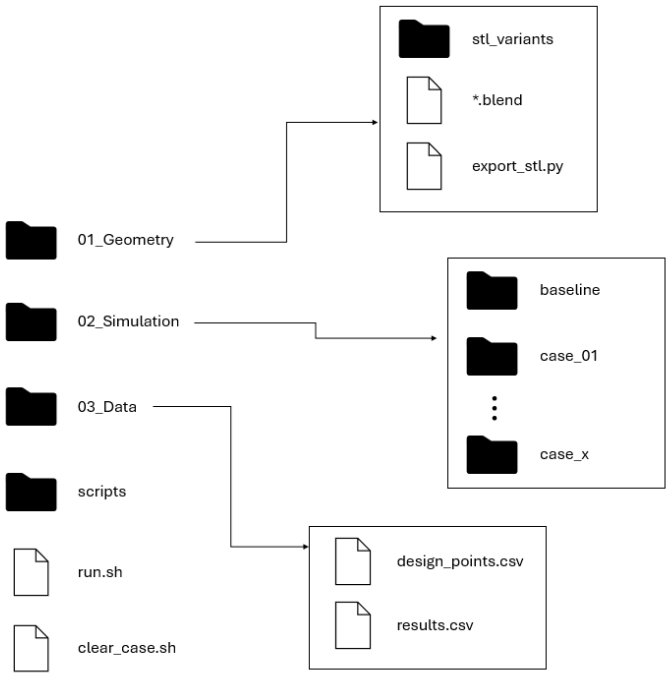


Figura 3.1: Organizzazione gerarchica del repository di progettoi.

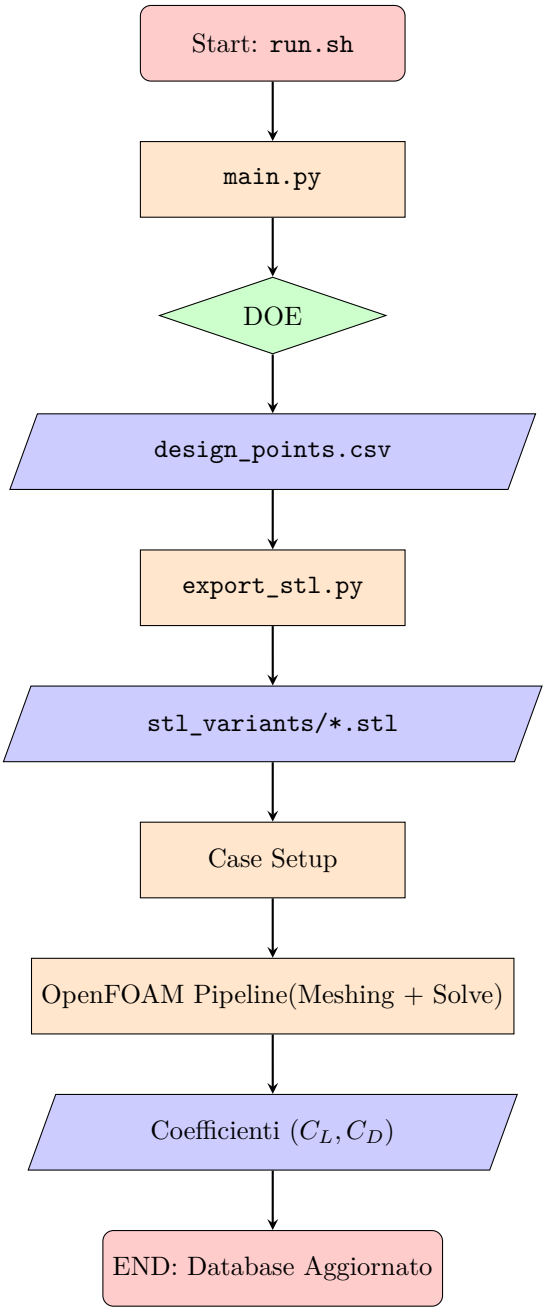


Figura 3.2: Diagramma logico delle fasi operative: dalla generazione statistica del DOE alla risoluzione fluidodinamica e raccolta dati.

3.1 Generazione del Piano Sperimentale (DOE)

Il coordinamento statistico dell'intero workflow è affidato alla logica di generazione del Design of Experiments (DOE), implementata all'interno del modulo `main.py`. Questa fase definisce la distribuzione dei punti di campionamento all'interno dello spazio di design, garantendo che l'esplorazione delle variabili geometriche sia efficiente e priva di ridondanze informative. Per massimizzare la versatilità dello strumento, il sistema integra una funzione denominata `handle_design_points` (schematizzata in Figura 3.3), che offre all'utente la possibilità di scegliere tra due modalità operative in base alle finalità dell'indagine.

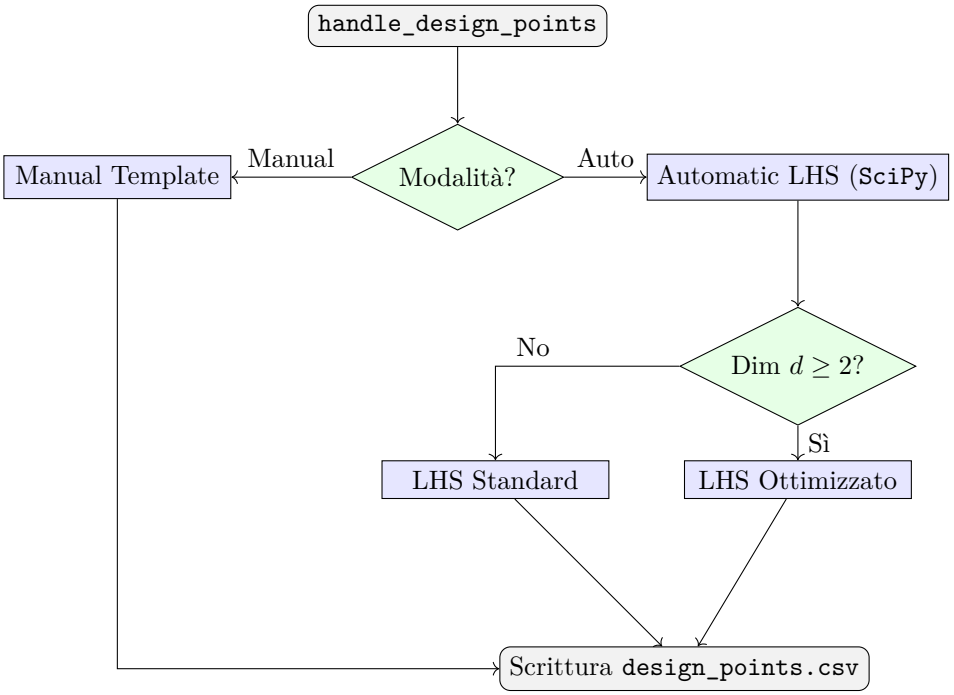


Figura 3.3: Logica di selezione della strategia di campionamento.

La modalità Automatic LHS rappresenta il pilastro del campionamento ottimizzato. In questo scenario, il workflow utilizza il sottomodulo `qmc.LatinHypercube` della libreria `SciPy`. La logica algoritmica è programmata per adattarsi automaticamente alla dimensionalità del problema, identificata dal numero di Shape Keys attive nel file di modellazione. Se il numero di parametri d è superiore o uguale a due, il sistema attiva i criteri di ottimizzazione geometrica (come la minimizzazione della discrepanza o il criterio Maximin) discussi nel primo capitolo. Qualora invece si operi su una singola variabile geometrica, il workflow commuta automaticamente su un campionamento LHS standard; tale precauzione tecnica è necessaria per prevenire instabilità numeriche del modulo `qmc` che potrebbero verificarsi nel tentativo di ottimizzare uno spazio monodimensionale.

Un elemento cardine della validazione scientifica del processo è la gestione del seed di riproducibilità. Attraverso il parametro `rng` (Random Number Generator), l'utente può inserire un codice numerico univoco che caratterizza la specifica matrice di permutazione. L'inclusione di questo valore nella prima riga del file di output `design_points.csv` assicura che l'intero esperimento sia perfettamente riproducibile, permettendo di rigenerare l'esatta sequenza di varianti geometriche in qualsiasi momento.

In alternativa, la modalità Manual Template risponde all'esigenza di effettuare test mirati o analisi di sensibilità locali. In questa configurazione, l'utente ha la facoltà di gestire direttamente la scrittura del file CSV, definendo manualmente le coordinate dei punti di design. Il workflow, in questo caso, agisce esclusivamente come motore di esecuzione, leggendo i valori forniti e procedendo alla generazione delle varianti senza applicare logiche di campionamento stocastico.

Il file risultante, `design_points.csv` (si veda l'esempio in Figura 3.4), viene strutturato in modo da fungere da ponte informativo tra Python e Blender: la seconda riga del file riporta i nomi esatti delle chiavi di deformazione letti dinamicamente dal file .blend, seguiti, nelle righe successive, dai valori numerici normalizzati. Questa struttura garantisce una sincronizzazione perfetta tra la fase di progettazione statistica e la trasformazione geometrica, eliminando ogni rischio di disallineamento tra i parametri del DOE e le proprietà fisiche del modello.

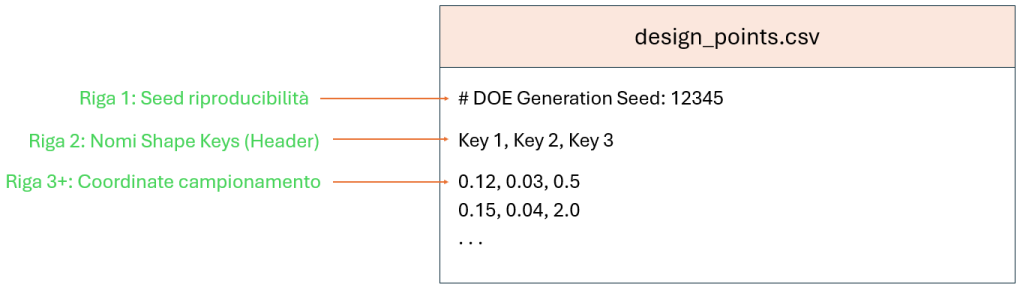
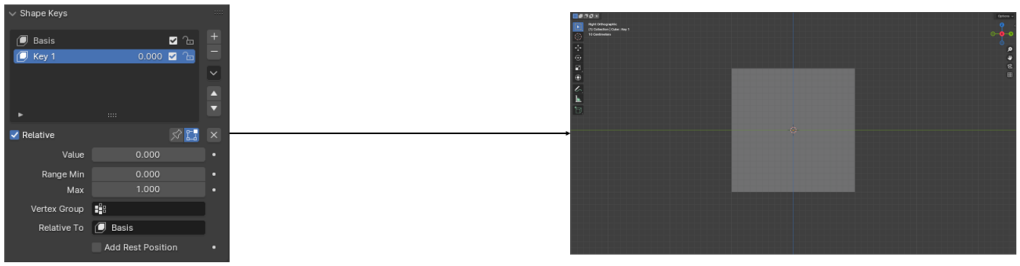


Figura 3.4: Struttura interna del file di controllo `design_points.csv`. La riga di intestazione permette la sincronizzazione dinamica con le Shape Keys di Blender.

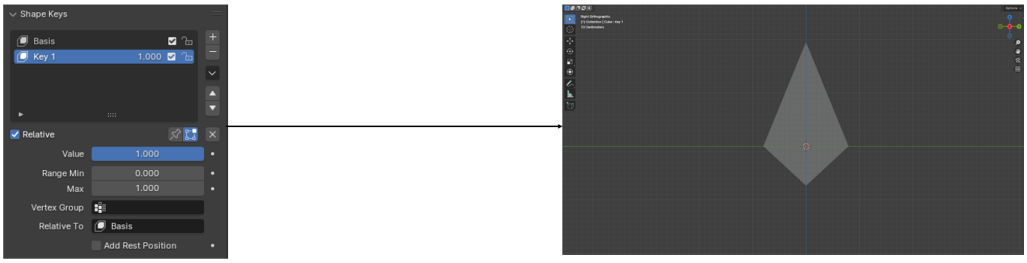
3.2 Modellazione parametrica e Automazione Geometrica in Blender

La traduzione dei parametri numerici definiti nel piano sperimentale in varianti geometriche tridimensionali è gestita all'interno della cartella `01_Geometry/`. Il fulcro operativo è rappresentato dal file master di Blender, il quale ospita la geometria di riferimento consolidata in un'unica mesh poligonale. L'unificazione della

mesh costituisce un requisito funzionale imprescindibile per garantire l'integrità topologica durante le fasi di deformazione; tale accorgimento previene la formazione di discontinuità superficiali o di geometrie non-manifold che comprometterebbero irreversibilmente gli algoritmi di meshing in OpenFOAM. La variabilità geometrica è ottenuta mediante l'implementazione delle Shape Keys, che definiscono scostamenti lineari dei vertici rispetto alla configurazione di partenza, denominata *Basis*. Ogni coordinata definita nel piano sperimentale (DOE) viene mappata direttamente su una di queste chiavi di deformazione, permettendo di variare le caratteristiche del profilo in modo fluido e controllato, come evidenziato dal confronto tra lo stato nominale e quello deformato in Figura 3.5



(a) Configurazione *Basis* con valore d'influenza nullo (0.0); la mesh mantiene la geometria indeformata di riferimento.



(b) Applicazione della *Shape Key 1* con valore d'influenza massimo (1.0); si osserva la deformazione lineare dei vertici verso il target impostato.

Figura 3.5: Pannello delle *Shape Keys* e manipolazione della mesh.

L'automazione di questo passaggio è affidata allo script `export_stl.py`, il quale viene eseguito in background per ottimizzare l'allocazione della memoria e accelerare le operazioni di input/output, evitando il sovraccarico computazionale legato all'interfaccia grafica. Lo script esegue un'interrogazione sistematica delle API di Blender per identificare dinamicamente le Shape Keys associate all'oggetto mesh. Escludendo la configurazione *Basis*, il codice indicizza le chiavi presenti e le sincronizza con i nomi estratti dalla seconda riga del file `design_points.csv`. Questa architettura conferisce al sistema un'elevata flessibilità: l'utente può aggiungere, rimuovere o rinominare i parametri geometrici direttamente nell'ambiente di modellazione senza dover intervenire sul codice sorgente dello script. Per ciascun set di parametri

registrato nel database, lo script sovrascrive i valori di influenza delle Shape Keys e forza l'aggiornamento dei vertici della mesh. La geometria risultante viene esportata in formato STL nella sottocartella `stl_variants/` seguendo una convenzione di nomenclatura serializzata (case_x.stl), dove l'indice x corrisponde alla riga specifica della matrice del DOE. Questo protocollo garantisce la corrispondenza univoca tra i parametri di input e la geometria prodotta, annullando la possibilità di errore umano nella generazione massiva delle varianti e preparando il terreno per la successiva fase di simulazione fluidodinamica.

3.3 Configurazione della Simulazione CFD

La gestione della fase di calcolo fluidodinamico è centralizzata all'interno della cartella `02_Simulation/`. Il pilastro strutturale di questa sezione è la directory denominata `baseline/`, concepita come un caso studio "template" che racchiude l'intera gerarchia di file necessaria per un'analisi in OpenFOAM. La funzione della baseline è quella di fungere da archivio delle impostazioni fisiche e numeriche comuni: ogni parametro relativo ai modelli di turbolenza, alle proprietà del fluido, alle condizioni al contorno e agli schemi di discretizzazione viene definito in questa sede e successivamente ereditato da tutte le varianti generate dal workflow.

L'impostazione della cartella rispecchia la struttura canonica di un caso OpenFOAM:

- `0/`: contenente i file per l'inizializzazione dei campi (pressione, velocità, parametri di turbolenza);
- `constant/`: preposto alla definizione delle proprietà fisiche e della sottocartella `triSurface`;
- `system/`: ospitante i dizionari di controllo e i parametri per il meshing e la risoluzione numerica.

Una delle principali criticità nell'automazione di analisi massive risiede nella necessità di aggiornare i riferimenti ai file geometrici all'interno dei dizionari di pre-processing (quali snappyHexMeshDict o surfaceFeatureExtractDict) ogni volta che si analizza una nuova variante. Per ovviare a questa limitazione e ridurre drasticamente il rischio di errori di sintassi, il workflow implementa una strategia di "astrazione" dei nomi. All'interno della cartella `triSurface/` di ogni caso istanziato, il file geometrico specifico (case_x.stl) viene importato e rinominato invariabilmente come `model.stl`.

Tale architettura, rappresentata schematicamente in Figura 3.6, permette di mantenere statici i riferimenti all'interno dei file di configurazione: i dizionari punteranno costantemente al file `model.stl`, mentre la variabilità geometrica è

garantita dal fatto che ogni file, pur condividendo lo stesso nome, risiede in una sottodirectory isolata dedicata al rispettivo caso studio. Questa architettura non solo semplifica la gestione logistica dei file, ma rende il workflow estraneo alla complessità del modello, delegando all'utente esclusivamente il compito di configurare correttamente la fisica della baseline e lasciando al sistema l'onere dell'integrazione e dell'adattamento automatico dei casi.

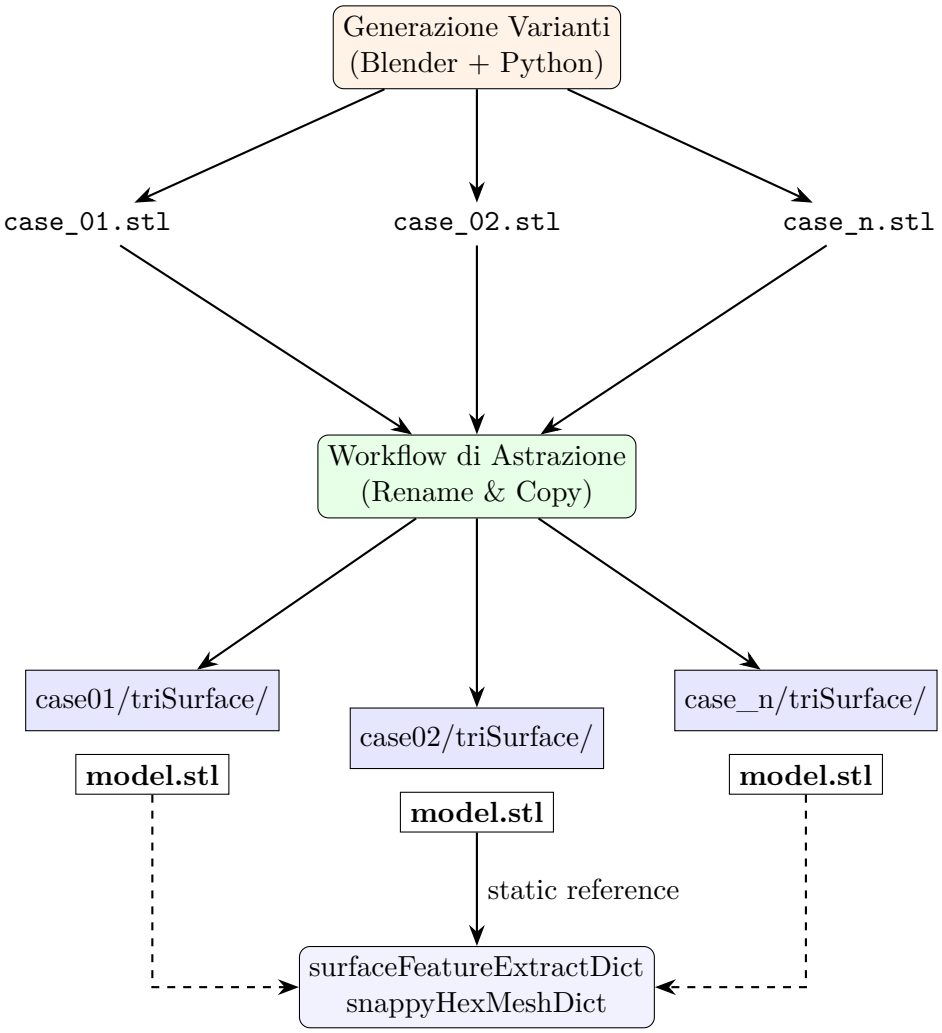


Figura 3.6: Schema del meccanismo di astrazione geometrica: le varianti STL vengono normalizzate nel nome `model.stl` per mantenere invariati i dizionari di setup.

3.4 Esecuzione del workflow

Una delle funzionalità determinanti per l'efficienza del sistema è la gestione dinamica del calcolo parallelo, rappresentata nel diagramma di flusso in Figura 3.7. Attraverso

la definizione del numero di core nelle righe iniziali del file run.sh, lo script provvede a modificare automaticamente il dizionario decomposeParDict nella cartella `system/` del caso *baseline*, adattandolo alle specifiche hardware della workstation in uso. L'attivazione della modalità parallela avviene mediante l'utilizzo della flag -p in fase di lancio (es. `./run.sh -p`); in assenza di tale opzione, il sistema procede all'esecuzione seriale, offrendo una flessibilità operativa fondamentale sia in fase di debug numerico che di produzione massiva dei dati. È importante precisare che l'architettura del framework è concepita per una gestione sequenziale dei casi studio con scomposizione parallela di questi ultimi. Ciò significa che le simulazioni vengono eseguite in successione (una dopo l'altra), ma ogni singola istanza sfrutta la decomposizione del dominio per distribuire il carico computazionale su più core fisici.

Lo script estrae dinamicamente il nome del solutore ($SOLVER) direttamente dal controlDict della *baseline*, garantendo che il workflow sia estraneo alla tipologia di analisi fluidodinamica impostata. La sequenza operativa prevede:

- Generazione del file `case_name.foam` per l'identificazione rapida del caso in ambiente di post-processing (ParaView);
- Esecuzione di `blockMesh` per la definizione del dominio di calcolo;
- Estrazione delle feature geometriche tramite `surfaceFeatureExtract`;
- Generazione della mesh di calcolo con `snappyHexMesh -overwrite`.

Al termine della fase di meshing, il sistema esegue una verifica di integrità e restituisce in console un riscontro immediato sul successo dell'operazione. Nel caso di esecuzione parallela, lo script gestisce l'intera procedura di scomposizione del dominio (decomposePar), il lancio del solutore tramite il comando `mpirun -np $N_CORES $SOLVER -parallel` e la successiva ricomposizione dei campi di moto con `reconstructPar`.

Per ottimizzare l'occupazione delle risorse hardware e gestire correttamente lo spazio disco durante analisi estese, lo script provvede infine alla rimozione automatica delle sottodirectory `processor*`. Ogni comando genera log dettagliati all'interno della rispettiva cartella del caso, assicurando che l'intero passaggio, dalla geometria alla soluzione convergente, sia un processo continuo, monitorabile e caratterizzato da un elevato grado di robustezza numerica.

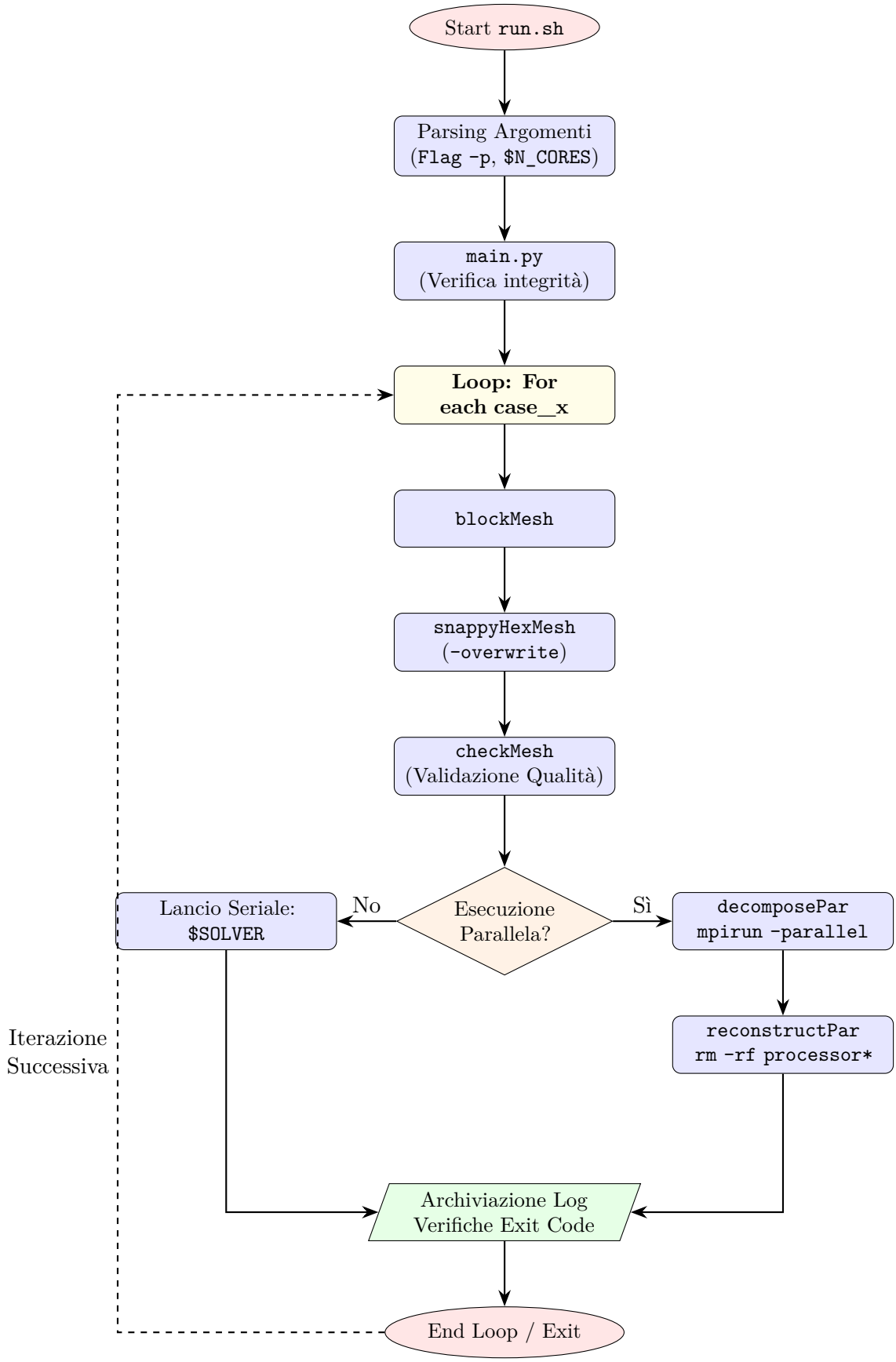


Figura 3.7: Diagramma di flusso ottimizzato della logica di `run.sh`: gestione dell'ereditarietà dei casi e flessibilità del calcolo parallelo.

3.5 Post-processing

L'ultima fase del workflow riguarda lo sviluppo di un algoritmo dedicato alla raccolta e alla sintesi dei risultati numerici. L'obiettivo è trasformare la mole di dati grezzi (raw data) prodotti dalle simulazioni indipendenti in un dataset strutturato pronto per l'analisi statistica e la validazione aerodinamica.

La logica di estrazione, illustrata nel diagramma di flusso di Figura 3.8, segue un algoritmo di parsing selettivo:

1. Scansione iterativa: Lo script identifica le directory `case_x` analizzando i file `.dat` generati dal function object di estrazione dei coefficienti di OpenFOAM.
2. Estrazione del valore a regime: Attraverso la libreria `pandas`, il sistema accede esclusivamente all'ultimo record temporale di ogni file. Questa scelta è motivata dalla natura stazionaria del calcolo: solo il valore a convergenza rappresenta la risposta fisica del profilo alla specifica configurazione geometrica.
3. Sincronizzazione degli Input: Ogni valore di C_L e C_D estratto viene accoppiato all'ID del caso corrispondente.

Il cuore tecnico di questa sezione è la generazione automatica del file `results.csv` nella directory `03_Data/`. La struttura del database consolidato è organizzata in forma tabellare, dove ogni riga rappresenta un esperimento computazionale univoco. Le prime colonne riportano le variabili indipendenti del design geometrico (Input), seguite dai coefficienti aerodinamici calcolati dal solutore (Output). Un esempio della struttura del file è riportato nella Tabella 3.1.

Tabella 3.1: Esempio di struttura del file `results.csv` generato automaticamente.

Case ID	**Shape Key 1**	**Shape Key 2**	C_L	C_D
case_01	0.5301	0.9659	1.1245	0.0215
case_02	0.9232	0.7680	1.4582	0.0284
case_03	0.4903	0.4970	0.9854	0.0192
case_04	0.8355	0.2156	1.2103	0.0247
...	...	...	...	...
case_x	0.2665	0.8732	0.7841	0.0168

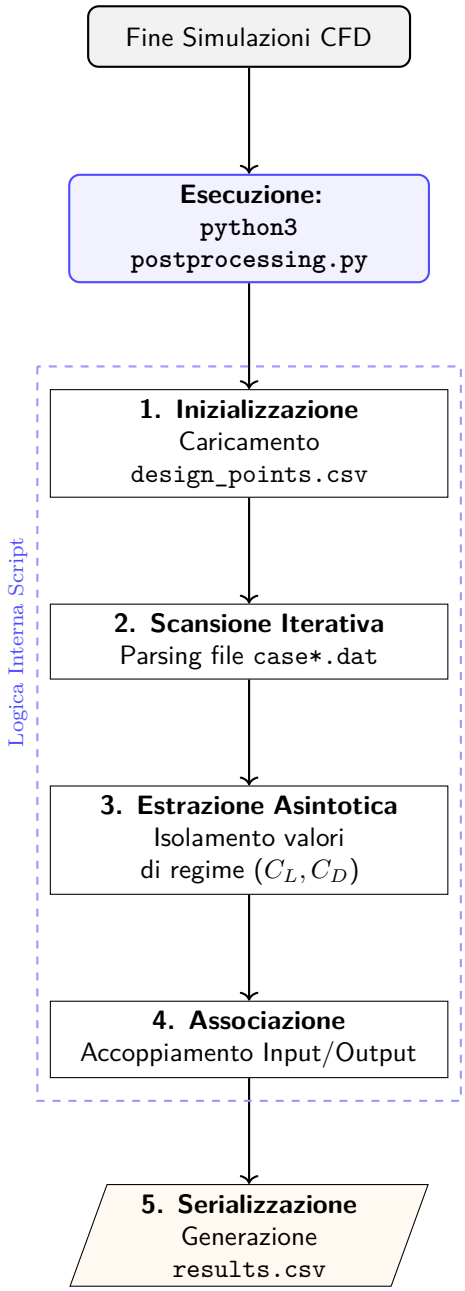


Figura 3.8: Architettura logica del processo di consolidamento dati tramite lo script `postprocessing.py`.

Capitolo 4

Validazione del workflow

Il processo di validazione del workflow ha inizio con la selezione di una geometria di riferimento che permetta di testare la robustezza dell'intera pipeline, dalla modellazione parametrica alla risoluzione fluidodinamica. Si è optato per il profilo alare NACA 4412, una geometria ampiamente documentata in ambito aerodinamico.

L'origine della geometria all'interno dell'ambiente Blender è avvenuta tramite l'importazione di coordinate puntuali, estratte dal database digitale Airfoil Tools [11], successivamente consolidate in una mesh poligonale mediante un processo di estrusione. Un aspetto cruciale di questa fase ha riguardato la gestione della risoluzione superficiale: per ovviare alla spigolosità nativa derivante dalla discretizzazione dei vertici in Blender, è stato implementato il modificatore Subdivision Surface, come si evince dalla Figura 4.1. Tale accorgimento tecnico ha permesso di regolarizzare la superficie, garantendo una continuità geometrica (*smoothness*) indispensabile per prevenire instabilità numeriche o errori topologici durante le successive fasi di meshing in OpenFOAM.

Al fine di validare l'automazione della variabilità geometrica, il profilo è stato dotato di due Shape Keys indipendenti, che costituiscono i parametri dello spazio di design:

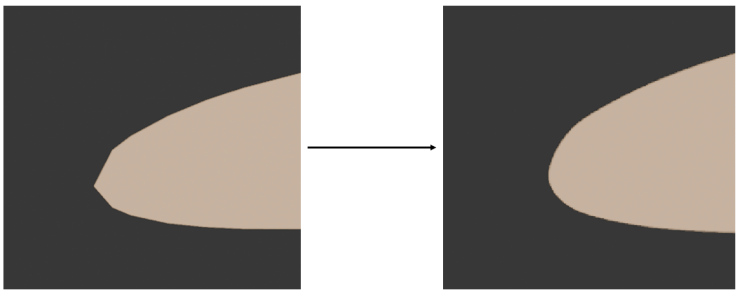


Figura 4.1: Dettaglio del bordo d'attacco: confronto tra il profilo originale con spigoli vivi (sinistra) e il profilo smussato (destra).

- Spessore relativo: Una deformazione scalare applicata lungo l'asse normale alla corda, finalizzata a testare la capacità del sistema di gestire variazioni volumetriche della mesh.
- Deflessione del Trailing Edge: Una deformazione locale configurata per simulare l'estensione di un flap, sollecitando la robustezza del workflow nel gestire cambiamenti significativi della curvatura al bordo d'uscita.

Le due modalità di deformazione sono illustrate in Figura 4.2, evidenziando come il sistema sia in grado di alterare sia l'ingombro volumetrico che la curvatura locale del profilo senza compromettere la coerenza della mesh.

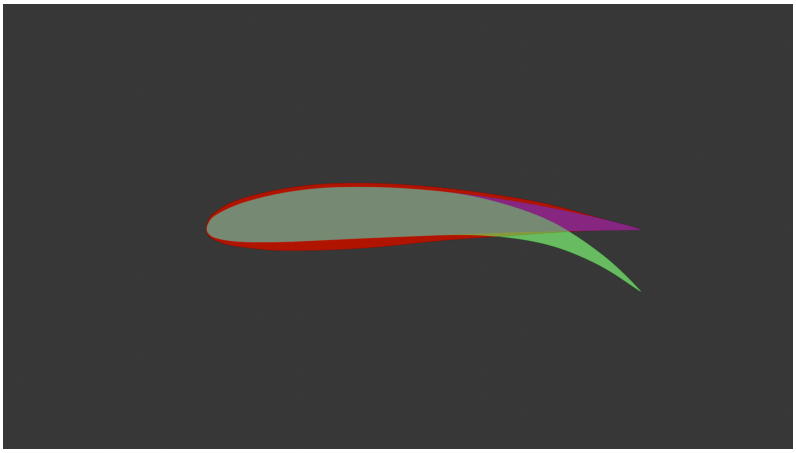


Figura 4.2: Visualizzazione delle deformazioni geometriche indotte dalle Shape Keys: profilo indeformato (viola), deflessione massima del bordo d'uscita (verde) e variazione massima dello spessore (rosso).

Definita la geometria master, la validazione del workflow si è concentrata sulla parte statistica. L'obiettivo di questa fase è dimostrare la capacità del sistema di esplorare lo spazio di design in modo autonomo, garantendo la generazione di un set di dati coerente e riproducibile.

Per il test case in esame, si è optato per una modalità di campionamento automatico basata sull'algoritmo Latin Hypercube Sampling (LHS). La scelta è ricaduta su un numero di 10 punti di design, un campione ritenuto sufficiente per verificare la stabilità delle iterazioni senza eccedere nel carico computazionale. Il workflow, interrogando le API di Blender, ha identificato le due Shape Keys attive (spessore e deflessione del flap) e ha generato una matrice di campionamento ottimizzata per coprire uniformemente il dominio bidimensionale delle variabili. Come evidenziato dal log di sistema (Figura 4.3), la corretta inizializzazione del modulo statistico ha permesso la creazione del file `design_points.csv`, che funge da registro centrale per l'intera campagna di simulazione.

Per quantificare oggettivamente la qualità della copertura del dominio, sono stati calcolati gli indici di dispersione del set di dati. Il campionamento, illustrato in Figura 4.4, ha restituito una distanza minima euclidea tra i punti pari a $\delta_{min} = 0.1712$, valore che attesta l'efficacia del criterio nell'evitare ridondanze geometriche. Inoltre, l'analisi della discrepanza ha fornito un valore di $CD = 0.0045$, confermando un'elevata uniformità distributiva.

L'efficacia di questa architettura è stata confermata dall'esecuzione dello script `export_stl.py`. Operando in modalità *headless*, lo script fornisce un feedback in tempo reale (si veda il log in Figura 4.5) circa lo stato dell'esportazione, indicando per ogni iterazione i valori geometrici correnti e il successo della scrittura del file STL. La perfetta corrispondenza tra i valori definiti nel DOE e le geometrie esportate nella cartella `01_Geometry/stl_variants/` attesta l'eliminazione del rischio di errore umano nella gestione massiva dei dati, validando la logica di integrazione tra il modulo statistico Python e l'ambiente di modellazione.

```
[FILE FOUND] 'design_points.csv' already exists.
   > Use existing [U] or Overwrite/Create new [O]? O

--- Design of Experiments (DOE) Setup ---
Detected Keys: Flap, Thickness
How many cases do you want to simulate? 10
Mode: [A]utomatic LHS or [M]anual Template? A

[METHODOLOGY] Using Seeded LHS for scientific reproducibility.
Insert Seed value (Label): 1

[OK] LHS generated with 10 points (Seed: 1).
```

Figura 4.3: Print in console della procedura di inizializzazione del Design of Experiments. Si evidenziano il parsing automatico delle variabili geometriche e l'impostazione del seed algoritmico per garantire la riproducibilità della sequenza di campionamento LHS.

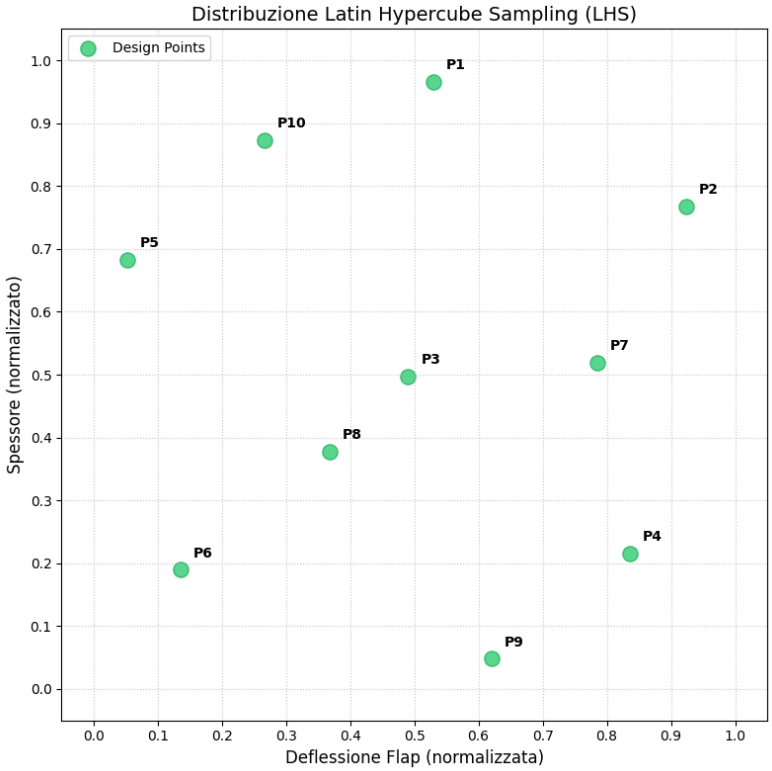


Figura 4.4: Distribuzione dei 10 punti di design nello spazio normalizzato [0, 1] generata tramite algoritmo Latin Hypercube Sampling (LHS) ottimizzato.

Figura 4.5: Validazione a schermo del completamento dei task di pre-processing per le 10 varianti geometriche previste dal piano sperimentale.

4.1 Setup della baseline e Strategia di Calcolo

Il cuore operativo della validazione risiede nella configurazione della cartella `baseline`, ovvero la struttura modello che il workflow clona e adatta per ogni singola istanza

di calcolo. In questa fase, le scelte numeriche e fisiche sono state orientate esclusivamente alla massimizzazione della robustezza del solutore e al contenimento dei tempi di calcolo, garantendo la percorribilità del ciclo iterativo su 10 casi consecutivi senza necessità di monitoraggio costante, sacrificando tuttavia la veridicità fisica del problema.

L'inizializzazione del dominio, gestita nel modulo `0/`, ha previsto l'impostazione di uno schema *inlet/outlet*, illustrato schematicamente in Figura 4.6, con una velocità del fluido uniforme $\mathbf{U} = (1, 0, 0)$, trattando le superfici del profilo NACA 4412 e le restanti delimitazioni del dominio computazionale come pareti fisiche (*walls*). La semplificazione del modello fisico è proseguita nel modulo `constant/`, dove si è adottato un modello di trasporto Newtoniano con una viscosità cinematica impostata a $\nu = 0.01$ m^2/s.

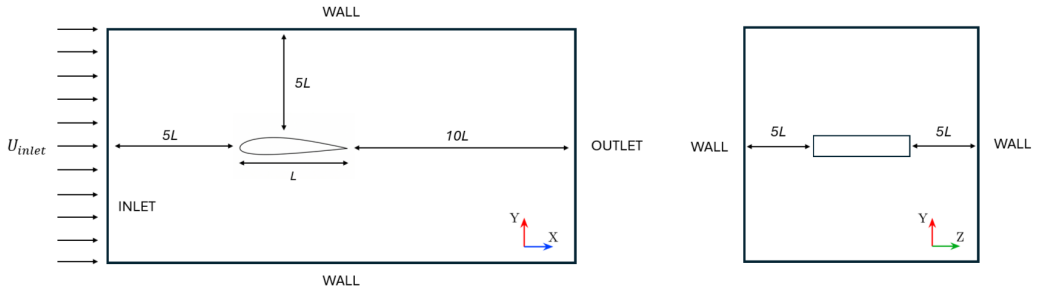


Figura 4.6: Rappresentazione schematica del dominio di calcolo 3D e delle relative condizioni al contorno impostate nella cartella `baseline`.

Per isolare il comportamento del workflow da possibili instabilità numeriche legate alla modellazione della turbolenza, è stata selezionata una simulazione di tipo laminare. Questa scelta metodologica è fondamentale in fase di validazione software: eliminando le variabili aleatorie connesse alla risoluzione dei modelli più complicati, ogni eventuale errore o interruzione della pipeline può essere ricondotto univocamente a difetti nell'architettura del codice o nella generazione della mesh, piuttosto che a divergenze fisiche del solutore.

La strategia di discretizzazione spaziale è stata definita nella configurazione di *baseline*, i cui parametri di setup all'interno della cartella `system/` costituiscono il riferimento per l'intero workflow automatizzato. Nello specifico, si è proceduto inizialmente alla creazione di una *background mesh* tramite l'utility `blockMesh`, definendo un dominio di calcolo coerente con i parametri della geometria analizzata. Successivamente, la definizione del profilo alare è stata finalizzata mediante l'impiego di `snappyHexMesh`, che ha permesso di adattare la griglia alla superficie dello STL esportato da Blender. A titolo illustrativo, in Figura 4.7 viene riportata la mesh ottenuta per uno dei casi esaminati, evidenziando l'accuratezza dello *snapping* sulla superficie del profilo. Per garantire una corretta risoluzione del campo di moto

in prossimità della parete, è stato implementato un raffinamento locale tramite la funzione `addLayers` di `snappyHexMesh`. Il dettaglio del bordo d'attacco, illustrato in Figura 4.8, mostra la generazione dei tre livelli di celle prismatiche (layers) estrusi dalla superficie.

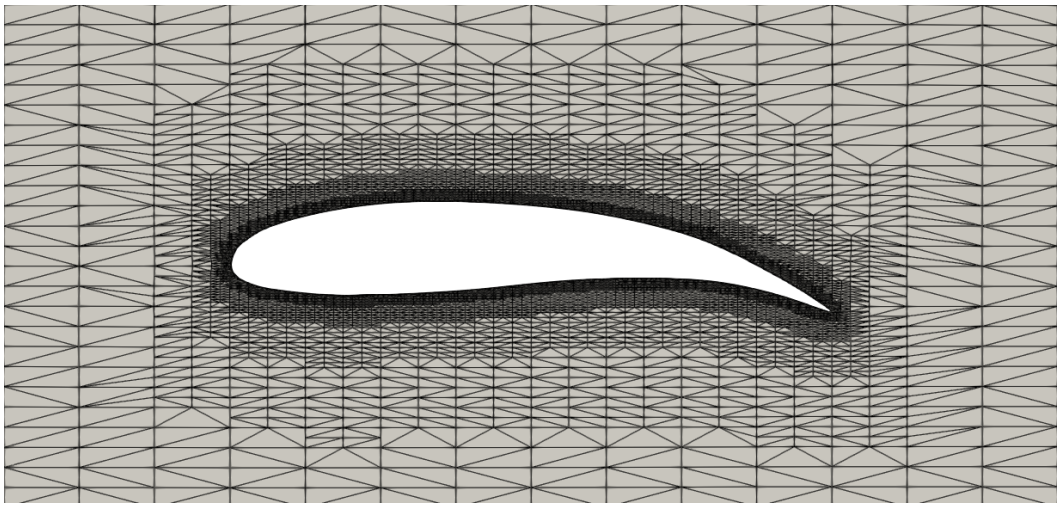


Figura 4.7: Sezione planare della mesh attorno al profilo.

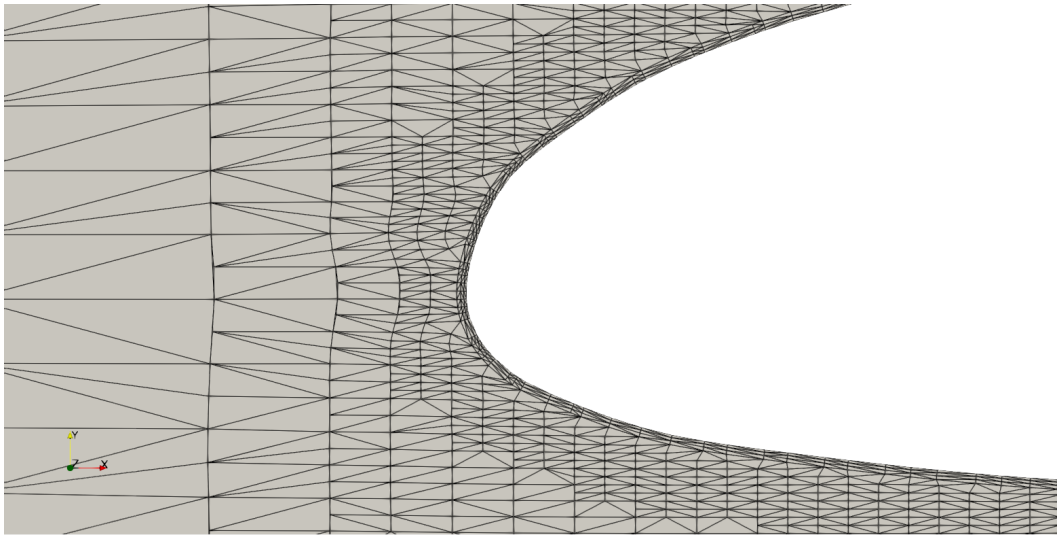


Figura 4.8: Dettaglio della mesh al bordo d'attacco. Si osservino i 3 layers derivanti dalla funzione `addLayers` di `snappyHexMesh`.

La simulazione è stata condotta per un intervallo temporale da 0 a 50 secondi, con un passo temporale $\Delta t = 0.1$ s, utilizzando il solutore `simpleFoam`. Il successo del processo, confermato dall'assenza di errori topologici nel report finale, attesta che la configurazione della *baseline* è sufficientemente solida da supportare l'automazione massiva dei casi studio.

4.2 Estrazione dei Risultati e Analisi della Convergenza del Workflow

L'ultima fase del processo di validazione riguarda la capacità del sistema di gestire l'intero ciclo di vita del dato: dalla generazione alla raccolta, fino alla sintesi in formati pronti per l'analisi statistica. Al termine del calcolo di ciascuno dei dieci casi studio, il workflow attiva le routine di post-processing automatizzate in linguaggio Python.

Il risultato di questa aggregazione è sintetizzato nella Figura 4.9, che illustra la sensibilità del coefficiente di portanza rispetto ai parametri geometrici di input. Come si evince dal grafico, il sistema cattura con precisione il trend fisico atteso, mostrando una chiara dipendenza del C_L dalla deflessione della Shape Key relativa al flap. Tale mappa di sensibilità conferma l'efficacia della trasmissione del dato tra l'ambiente di deformazione geometrica e quello di analisi numerica. Per fornire un quadro esaustivo del comportamento aerodinamico del profilo, l'analisi è stata estesa anche al coefficiente di resistenza C_D, il cui andamento è rappresentato in Figura 4.10.

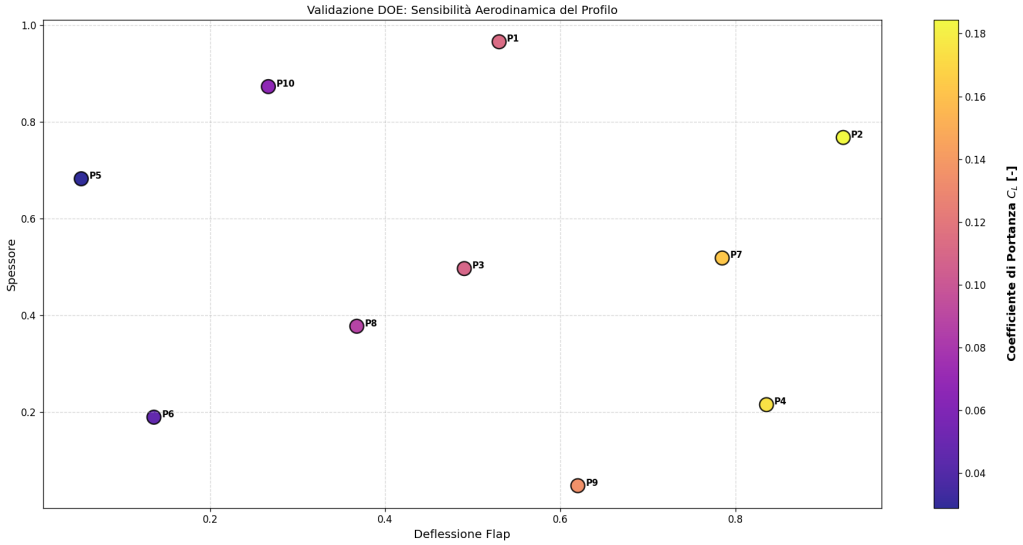


Figura 4.9: Analisi di sensibilità del coefficiente di portanza (C_L). Il grafico a dispersione mostra la correlazione tra la deflessione del flap (asse X) e lo spessore del profilo (asse Y). La risposta aerodinamica è evidenziata dalla scala cromatica, che indica il valore del C_L per ogni punto di design.

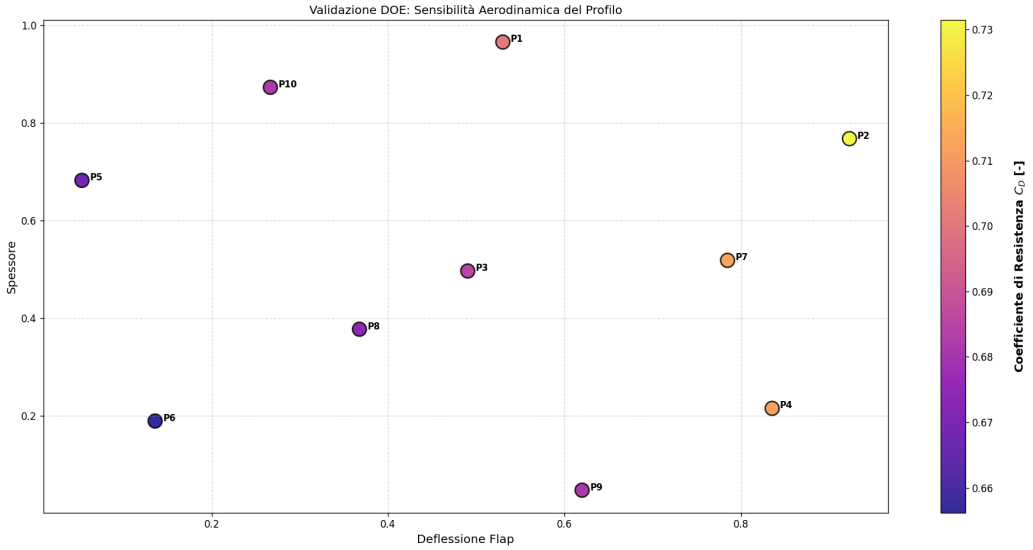


Figura 4.10: Analisi di sensibilità del coefficiente di resistenza (C_D). Il grafico a dispersione mostra la correlazione tra la deflessione del flap (asse X) e lo spessore del profilo (asse Y). La risposta aerodinamica è evidenziata dalla scala cromatica, che indica il valore del C_D per ogni punto di design.

In questa specifica applicazione, l'enfasi non è posta sul valore fisico assoluto dei coefficienti, influenzato dalle semplificazioni computazionali della baseline, bensì sulla precisione e sulla tracciabilità della trasmissione del dato. Il successo dell'integrazione è testimoniato dall'aggiornamento dinamico del database nella directory `03_Data/`, dove a ogni configurazione geometrica campionata dal DOE corrisponde univocamente il relativo output numerico, eliminando ogni possibilità di errore umano nella trascrizione.

A completamento dell'analisi quantitativa, l'automazione del post-processing consente l'estrazione sistematica dei campi di moto, permettendo un riscontro visivo immediato sulla coerenza delle soluzioni numeriche. Nelle Figure 4.11 e 4.12 sono riportati i contour di velocità relativi alle configurazioni di massimo spessore e massima deflessione del flap. Queste visualizzazioni non solo confermano la corretta esecuzione delle operazioni di morphing della mesh, ma evidenziano come le variazioni geometriche imposte dal DOE si traducano in modifiche fisicamente plausibili della struttura del flusso.

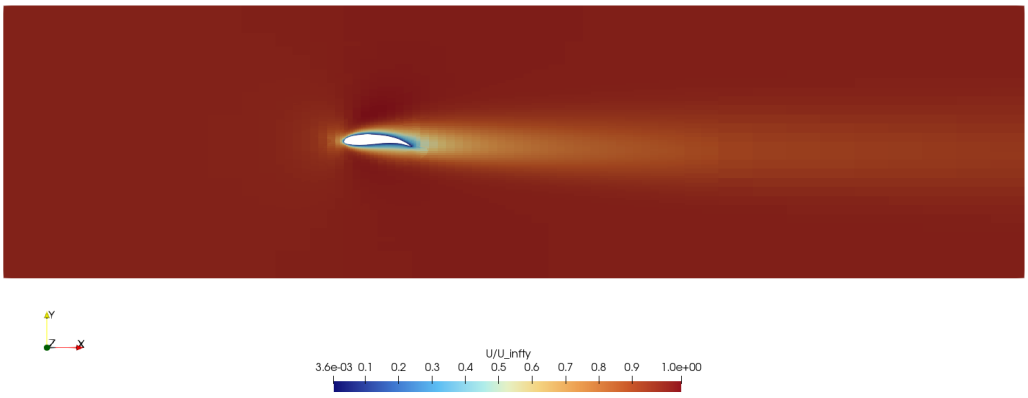


Figura 4.11: Campo di velocità per la configurazione con massimo spessore.

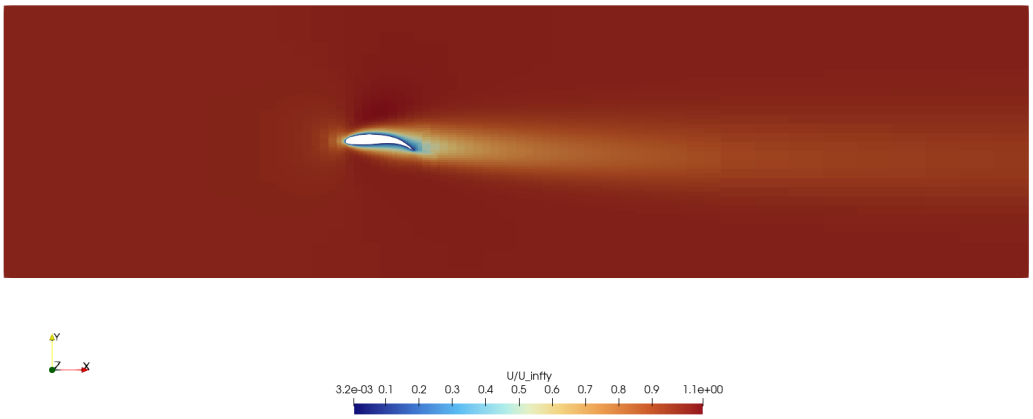


Figura 4.12: Campo di velocità per la configurazione con massima deflessione del flap.

Capitolo 5

Conclusioni

Il presente lavoro ha dimostrato l'efficacia dell'integrazione tra strumenti di computer grafica e solutori fluidodinamici per la creazione di una catena di calcolo automatizzata e open-source. La validazione condotta sul profilo NACA 4412 ha confermato che l'impiego del Latin Hypercube Sampling (LHS) ottimizzato garantisce una mappatura dello spazio di design significativamente più efficiente rispetto ai metodi tradizionali, riducendo la ridondanza informativa.

Il workflow sviluppato ha permesso di eliminare l'intervento manuale nelle fasi critiche di deformazione geometrica e setup dei casi, annullando il rischio di errore umano nella gestione di campagne di simulazione massive e di garantire la piena riproducibilità degli esperimenti numerici attraverso una gestione rigorosa dei seed algoritmici.

Nonostante il successo della validazione, l'attuale framework getta le basi per diverse estensioni metodologiche e applicative:

1. Integrazione di Modelli Surrogati: La base dati strutturata generata dal DOE può essere utilizzata per l'addestramento di reti neurali o Kriging model, permettendo la predizione dei coefficienti aerodinamici in tempo reale senza ricorrere a nuove simulazioni CFD.
2. Ottimizzazione Multiobiettivo: Implementazione di algoritmi genetici o basati su gradiente che utilizzino il framework come motore di valutazione per la ricerca automatica di profili ottimi.
3. Supporto a Geometrie 3D Complesse: Migrazione del workflow verso l'analisi di ali complete o velivoli integrati, testando la robustezza del meccanismo di astrazione geometrica su superfici caratterizzate da topologie multi-oggetto.

Bibliografia

[1] Elias Farah et al., OpenFOAM Automated Tool Chain - PoC. 2021. https://github.com/Eliasfarah0/OpenFOAM-Automated-Tool-Chain-PoC.

[2] Mangini, F.; Vaccalluzzo, M.; Bardoscia, E.; Bortoli, A.; Colombo, A. An Automated Computational Fluid Dynamics Workflow for Simulating the Internal Flow of Race Car Radiators. Appl. Sci. 2024, 14, 9930. https://doi.org/10.3390/app14219930

[3] Henning Scheufler et al., caseFoam: Automated setup and analysis of OpenFOAM Cases, DLR Institute of Aerodynamics and Flow Technology. https://github.com/DLR-RY/caseFoam, ultimo accesso: 24-May-2025.

[4] N. Ryden, "BlenderFoam: Aerodynamic Shape Optimization," 2019. [Online]. Available: `https://nry.me/posts/2019-03-05/blenderfoam-aerodynamic-shape-optimization/`.

[5] Wikipedia contributors, "Factorial experiment," *Wikipedia, The Free Encyclopedia*, 2024. [Online]. Available: `https://en.wikipedia.org/wiki/Factorial_experiment`.

[6] T. Bartz-Beielstein, M. Zaefferer, C. Doerr, *et al.*, "Benchmarking in optimization: best practice and open issues," *Optimization and Engineering*, vol. 24, pp. 145–206, 2023. [Online]. Available: `https://doi.org/10.1007/s11081-022-09731-6`.

[7] D. Foster, "A Gentle Introduction to Monte Carlo Methods," *Medium: Towards Data Science*, 2017. [Online]. Available: `https://medium.com/data-science/a-gentle-introduction-to-monte-carlo-methods-98451674018d`.

[8] Emergent Mind, "Latin Hypercube Sampling (LHS): Techniques and Applications in Statistical Sampling," 2024. [Online]. Available: `https://www.emergentmind.com/topics/latin-hypercube-sampling-lhs`.

[9] SciPy v1.15.1 Manual, “Quasi-Monte Carlo submodule (scipy.stats.qmc) - discrepancy,” 2025. [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.qmc.discrepancy.html`.

[10] Zhou, Yongdao & Fang, Kai-Tai & Ning, Jianhui. (2013). Mixture discrepancy for quasi-random point sets. Journal of Complexity. 29. 283–301. 10.1016/j.jco.2012.11.006.

[11] Airfoil Tools, “NACA 4412 Airfoil Data,“ [Online]. Available: `http://airfoiltools.com/`.