POLITECNICO DI TORINO

Collegio di Ingegneria Meccanica, Aerospaziale, dell'Autoveicolo e della Produzione

Corso di Laurea Magistrale in Ingegneria Aerospaziale

TESI DI LAUREA DI II LIVELLO

Creazione di un modello CFD su OpenFOAM per la dinamica di sloshing di un serbatoio preconfigurato con deflettori smorzanti

Relatore

Dr. Stefano Primatesta

Correlatore

Ing. Riccardo Enrico

Candidato:

Davide Volpatto

A mia nonna Pasqualina, che mi ha ispirato a non mollare mai, e mio nonno Aldo, di cui porto la memoria al polso per ricordare che l'ambizione non è nulla senza il lavoro

Abstract

Questa tesi valuta l'effetto dello sloshing di un liquido contenuto nel serbatoio di un drone utilizzato per l'irrorazione agricola, sia per trattamenti fitosanitari sia per integrazioni nutrizionali. L'analisi si concentra sul profilo di missione del drone, studiando l'efficacia dei deflettori esistenti e di nuove configurazioni progettate su misura per mitigare i picchi caratteristici del fenomeno, variando le condizioni di riempimento del serbatoio.

Il cuore dell'elaborato risiede nella costruzione del modello numerico tramite software open source OpenFOAM ed alcune estensioni applicative per le fasi di pre e post processing, andando a descrivere dettagliatamente l'architettura del codice con l'intenzione di essere utilizzata come guida per l'esecuzione di un set di simulazioni parametriche per la creazione di una base dati necessari alla costruzione di un algoritmo data-driven sfruttando le potenzialità future dei sistemi ad intelligenza artificiale nell'ingegneria.

La ricerca ha permesso di ottenere informazioni sull'accoppiamento geometrico ideale del serbatoio, valutando l'allineamento del sistema di riferimento locale con quello del drone e la rotazione del sistema di riferimento locale di 30 gradi intorno all'asse verticale. In questo modo, la direzione del moto coincide, nel primo caso, con la normale a una faccia piana del serbatoio esagonale, e nel secondo caso, con la diagonale tra due spigoli, risultando quest'ultima configurazione più efficace per lo smorzamento.

L'introduzione dei deflettori nel sistema, già presenti allo stato dell'arte di inizio elaborato, introduce una possibile asimmetria nella configurazione di volo ideale, che si traduce in una componente dinamica

sul piano perpendicolare alla direzione del moto. Per risolvere questo problema, sono state proposte due modifiche progettuali: un diverso accoppiamento dei deflettori radiali e la seconda la progettazione di deflettori tangenziali opportunamente collocati ad un'altezza pari al 70%, che si è dimostrato essere il livello di riempimento che più eccita la dinamica di sloshing, e, per ragioni simmetriche, al 30%.

Tramite la variazione parametrica del riempimento si è potuto estrarre anche un profilo di variazione delle frequenze caratteristiche tramite il quale è possibile agire con metodi di input shaping a supporto dei sistemi passivi costituiti dai deflettori, che risultano comunque sufficienti alla stabilizzazione del fenomeno.

Ringraziamenti

Desidero esprimere la mia profonda gratitudine ai miei relatori, il dott. Stefano Primatesta e il dott. Riccardo Enrico, per il loro inestimabile supporto, guida e fondamentale disponibilità durante lo sviluppo di questa tesi.

Ringrazio la mia famiglia per avermi permesso di studiare e aver cresciuto me con questa grande ambizione e dedizione al lavoro.

Ringrazio il dottore in scienze matematiche Andrea Damiano e il dottore in ingegneria aerospaziale Marco Giolo, amici di una vita, con cui ho condiviso il percorso e i traguardi fin dalla prima elementare, ai quali devo il merito di essere arrivato fino in fondo.

Ringrazio i miei amici Simone, Gabriele, Daniel, Lorenzo, Ludovico, Alessandro, Federica e Lavinia per l'indispensabile supporto morale.

Ringrazio infine la mia compagna Gaia, il cui amore e sostegno sono la mia stazione di ricarica; non dare limite alle prospettive è una virtù fondamentale per un ingegnere, e tu con la tua luce illumini intorno a me a 360 gradi.

Sommario

1. **Effetto sloshing** **1**

1.1 Introduzione 1

1.2 Stato dell'arte 2

2. **Teoria del modello** **5**

2.1 Modello matematico 5

2.2 Metodo dei volumi finiti 9

3. **OpenFOAM e implementazione del problema** **12**

3.1 Introduzione a OpenFOAM 12

3.2 Preparazione dell'ambiente e architettura software 17

3.3 Struttura di un caso 20

3.4 Scelta del solutore 21

4. **Simulazione** **24**

4.1 Pre-processing: preparazione della Mesh 24

4.2 Meshatore cfMesh e parametrizzazione 26

4.3 Qualità della mesh e affinamento 33

4.3.1 Non ortogonalità 34

4.3.2 Skewness (Asimmetria) 36

4.3.3 Aspect ratio (AR) 37

4.4 Creazione della Mesh 38

4.5 Setup della simulazione 40

4.5.2 Directory `constant` 41

4.5.3 Directory `0` 55

4.5.4 Directory `system` 61

4.6 Analisi di sensibilità 81

5. **Casi di studio e post processing** **87**

5.1 Caso 1: Serbatoio senza paratie nelle configurazioni di volo in direzione della faccia piana e dello spigolo 87

5.2 Caso 2: Serbatoio con paratie nelle due configurazioni di volo e proposta di modifica progettuale configurazione sperimentale 97

5.3 Caso 3: Risposta a gradino 106

5.4 Caso 4: Variazione parametrica del riempimento su serbatoio con deflettori allo stato dell'arte 108

5.5 Caso 5: Progettazione e aggiunta di deflettore tangenziale 115

5.6 Caso 6: Doppio deflettore tangenziale 119

6. Conclusioni 126

6.1 Considerazioni finali e limitazioni del modello di ricerca 126

6.2 Sviluppi futuri 128

7. Riferimenti bibliografici 131

Indice delle figure

Figura 1 Struttura delle directory di un caso di simulazione 20
Figura 2 Processo di soluzione implementato dal solutore 23
Figura 3 Modello CAD completo (in alto) e sezione serbatoio in trasparenza (in basso) con dettaglio delle paratie 25
Figura 4 Mesh alla prima iterazione 31
Figura 5 Mesh iterata con affinamento degli spigoli tramite feature *surfaceFeatureEdges* sulla geometria pre-ruotata 32
Figura 6 Immagine rappresentativa del parametro *non-orthogonality* della mesh, valutata tra celle adiacenti 34
Figura 7 Immagine rappresentativa del significato geometrico di *skeweness* per ciascuna cella 36
Figura 8 Struttura della directory *constant* del path di simulazione 41
Figura 9 Profilo di accelerazioni della missione del drone (in alto) e profilo di velocità (in basso) 45
Figura 10 Processo implementato dal loop PIMPLE 52
Figura 11 Struttura della directory *0* del path di simulazione 55
Figura 12 Struttura della directory *system* del path di simulazione 61
Figura 13 In alto a sinistra caso 4 di infittimento, in alto a destra caso 3, in basso a sinistra caso 2 e in basso e destra caso 1 82
Figura 15 Differenza di ampiezza della risposta al variare dell'infittimento della mesh . 84
Figura 14 Risposta completa al variare dell'infittimento della mesh. è possibile notare lo sfasamento progressivo 84
Figura 16 Errore relativo percentuale sulla pulsazione dell'analisi di sensibilità 85
Figura 17 Errore relativo percentuale sulle ampiezze dell'analisi di sensibilità 85
Figura 18 Modello CAD del sistema di spruzzatura 88
Figura 19 Orientamento delle due configurazioni del dominio di campo fluido nel primo caso di studio. A sinistra la direzione di volo (asse rosso) intercorre tra due spigoli, a destra interseca due facce piane 89
Figura 20 Profilo di accelerazioni importato in input (generato tramite ROS Topic) per la missione completa 89
Figura 21 Profilo di velocità della missione completa 90

Figura 22 Cattura immagini di alcuni istanti della simulazione per la fase acqua nella simulazione con direzione di volo intersecante due spigoli....92
Figura 23 Cattura immagini di alcuni istanti della simulazione per la fase acqua nella simulazione con direzione di volo normale alla faccia piana....92
Figura 24 Profilo di forze generato in post processing per le simulazioni del caso di studio 1....93
Figura 25 Profilo di accelerazioni semplificato (solo direzione x)....94
Figura 26 Profilo di forze generato in post processing per le simulazioni con evidenza sull'approssimazione di componenti yz nulle dell'accelerazione nel profilo in input rispetto al profilo completo per la missione....95
Figura 27 Effetto della dinamica di sloshing scorporato dell'effetto inerziale sui profili di missione completo e semplificato.....96
Figura 28 Modello cad del serbatoio in trasparenza con evidenza sul sistema di accoppiamento per assemblare i deflettori al suo interno....98
Figura 29 deflettori radiali pre-progettati (presenti allo stato dell'arte). A sinistra quello corto, assemblato perpendicolarmente a due facce piane, a destra quello lungo, assemblato tra due spigoli....98
Figura 30 Serbatoi in trasparenza nei duoi orientamenti caratteristici. A sinistra si può osservare l'asimmetria dovuta alla disposizione dei deflettori radiali allo stato dell'arte..99
Figura 31 Serbatoio in trasparenza con configurazione sperimentale composta da due deflettori radiali lunghi montati entrambi tra spigoli opposti. La direzione del moto è coincidente con l'asse di vista del lettore....100
Figura 32 Cattura immagini della simulazione da visualizzatore Paraview. In basso si osserva il dettaglio dell'evoluzione fluida attraverso i fori preconfigurati sui deflettori..102
Figura 33 Plot delle forze di sloshing sulle tre varianti del caso di studio 2....105
Figura 34 Risposta a gradino per FX nel caso di serbatoio con deflettori radiali....106
Figura 35 Risposta a gradino per FX nel caso di serbatoio senza deflettori....107
Figura 36 Forze totali lungo z al variare del riempimento. Si osserva la componente stazionaria dovuta al peso del fluido....110
Figura 37 Profilo di forze completo in direzione del moto (lungo x) durante la missione al variare del riempimento....112
Figura 38 Forza di sloshing in direzione del moto per alcuni valori di riempimento campionati....113
Figura 39 Curva interpolante rappresentativa dell'andamento delle frequenze di sloshing al variare del riempimento....114

Figura 40 Esempio di smorzamento tramite deflettori tangenziali 115
Figura 41 Serbatoio in trasparenza con deflettore tangenziale in compinazione ai già presenti deflettori radiali posto al 70% dell'altezza 116
Figura 42 Dettaglio del deflettore tangenziale. Si osserva la presenza di fori simmetrici con concavità verso il vasso per agevolare l'invaso 117
Figura 43 Risultato dell'effetto di smorzamento per il riempimento target confrontato con il caso con i soli deflettori radiali 118
Figura 44 Serbatoio in trasparenza con combinazione dei due deflettori radiali presenti allo stato di fatto e dei due deflettori tangenziali posti al 30 e al 70% dell'altezza 119
F**igura 45** Forze lungo x dovute alla dinamica di sloshing nel caso di riempimento del 70% target di progetto del primo deflettore tangenziale) e di riempimenti adiacenti (60 e 80%) 120
Figura 46 Forze lungo x dovute alla dinamica di sloshing nel caso di riempimento del 30% (targhet di progetto del secondo deflettore tangenziale) e di riempimenti prossimi (20 e 40%)..... 120
Figura 47 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 20%..... 122
Figura 48 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 30%..... 122
Figura 49 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 80%..... 124
Figura 50 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 70%..... 124

Indice delle tabelle

Tabella 1 Prototipo del file accel.dat contenuto in $SIM_DIR/constant 47
Tabella 2 Prototipo del file g contenuto in $SIM_DIR/constant.....48
Tabella 3 Prototipo del file phaseProperties contenuto in $SIM_DIR/constant 49
Tabella 4 Prototipo del file momentumTransport contenuto in $SIM_DIR/constant...50
Tabella 5 Prototipo del file physicalProperties.air contenuto in $SIM_DIR/constant...53
Tabella 6 Prototipo del file physicalProperties.water contenuto in $SIM_DIR/constant 53
Tabella 7 Prototipo del file transportProperties contenuto in $SIM_DIR/constant.....54
Tabella 8 Prototipo del file epsilon contenuto in $SIM_DIR/0 56
Tabella 9 Prototipo del file k contenuto in $SIM_DIR/0.....57
Tabella 10 Prototipo del file nut contenuto in $SIM_DIR/0 58
Tabella 11 Prototipo del file p_rgh contenuto in $SIM_DIR/0.....59
Tabella 12 Prototipo del file U contenuto in $SIM_DIR/0.....60
Tabella 13 Prototipo del file controlDict contenuto in $SIM_DIR/system.....62
Tabella 14 Prototipo del file decomposeParDict contenuto in $SIM_DIR/system.....66
Tabella 15 Prototipo del file fvOption contenuto in $SIM_DIR/system.....67
Tabella 16 Prototipo del file fvSchemes contenuto in $SIM_DIR/system 72
Tabella 17 Prototipo del file fvSolution contenuto in $SIM_DIR/system.....79
Tabella 18 Prototipo del file setFields contenuto in $SIM_DIR/system.....80
Tabella 19 Parametrizzazione delle diverse mesh per l'analisi di sensibilità 81
Tabella 20 Parametri di qualità delle mesh oggetto di analisi di sensitività.....82
Tabella 21 Andamento dei tempi di simulazione con l'infittimento della mesh 86
Tabella 22 Porotitpo del core del file setFieldsDict interpretato dal dizionario setFields (eseguito prima della simulazione) per riempire virtualmente il serbatoio.....90
Tabella 23 Estratto del file MeshDict con feature per affinamento locale della mesh in corrispondenza dei deflettori.....101
Tabella 24 Campionamento frequenze di sloshing nel caso di studio 4.....114

Capitolo 1

Effetto sloshing

1.1 Introduzione

Lo *sloshing* è un fenomeno caratteristico della dinamica dei fluidi che si manifesta quando lo stato di equilibrio statico di un liquido che riempie parzialmente un contenitore chiuso viene perturbato dalla movimentazione meccanica del contenitore stesso. Il movimento della massa fluida come risposta dinamica genera, sulle pareti del contenitore nella quale è racchiusa, delle forze e momenti idrodinamici che vanno a scaricarsi sulla struttura a cui esso è vincolato.

Questo fenomeno, che per altro tutti possono aver sperimentato nel corso della propria vita nell'intento di spostare un recipiente riempito di acqua, acquisisce un aspetto critico in applicazioni meccaniche e aerospaziali che hanno a che fare con lo stoccaggio di liquidi in serbatoi: l'interazione tra la dinamica fluida e la dinamica caratteristica del veicolo o velivolo può avere un effetto catastrofico, come portare al ribaltamento di un'autocisterna o la perdita di qualità nel controllo di assetto in un velivolo che sia esso ad ala fissa o rotante.

Risulta fondamentale una corretta analisi dello sloshing sia per verificare la compatibilità, entro i limiti di manovra e fattori di carico propri dell'inviluppo del volo, dell'autorità di controllo dei velivoli con le forze e i momenti risultanti da questo effetto fluidodinamico, oltre che accertarsi non vi siano sovrapposizioni armoniche nell'interazione dinamica

1.2 Stato dell'arte

Dai primi anni del 1900, significativi sforzi di ricerca sono stati indirizzati alla comprensione del fenomeno dello sloshing dei liquidi in diversi tipi di strutture. Inizialmente, Westergaard (1933) esaminò lo sloshing dei fluidi su dighe verticali, seguito da Hoskins e Jacobsen (1934), che condussero uno studio analitico e sperimentale su contenitori rigidi rettangolari e cilindrici, simulando il movimento sismico orizzontale. Jacobsen (1949) presentò una soluzione chiusa dell'equazione di Laplace per contenitori cilindrici rigidi, assumendo pareti del serbatoio rigide e considerando la velocità radiale degli elementi liquidi all'interfaccia liquido-struttura come quella delle pareti adiacenti, pari al movimento del terreno.

Tuttavia, questi primi studi presupponevano che le pareti del contenitore fossero rigide e che il serbatoio evolvesse con una dinamica di corpo rigido solidale al vincolo del terreno. Questa approfondita indagine trascurava l'interazione tra il moto del liquido e la risposta dinamica della struttura del serbatoio. Housner propose un approccio basato sull'assunzione che la pressione fosse combinazione degli effetti di pressione del liquido che accelera insieme al serbatoio e quelli del liquido che si muove all'interno del serbatoio, giungendo a delle prime espressioni semplificate. Tale approccio separava il movimento del liquido in due modalità: "impulsiva", che rappresentava il movimento del liquido in sintonia con la parete del contenitore, e "convettiva" o "sloshing", che rappresentava il movimento oscillante verticale del liquido.

L'interazione tra fluido e struttura, spesso ignorata, risultava in una semplificazione eccessiva. La realtà mostrava invece significative deformazioni delle pareti dei serbatoi sotto l'azione di forze esterne,

rendendo il comportamento dinamico del sistema liquido-serbatoio più complesso di quanto previsto con pareti rigide.

Nella seconda metà del XX secolo, l'analisi delle vibrazioni nei problemi di interazione tra fluido e struttura ha suscitato grande interesse, con diverse metodologie proposte per la loro risoluzione. Veletsos e Yang (1974, 1976, 1977) presentarono soluzioni per la pressione dinamica e le forze impulsive, considerando specifici schemi di deformazione delle pareti del serbatoio. Veletsos sviluppò condizioni al contorno modificate del moto del liquido considerando l'effetto di accoppiamento tra il carico idrodinamico e la risposta dinamica della parete del serbatoio. Successivamente, Balendra e Nash (1978, 1980) affrontarono i problemi di interazione tra fluido e struttura sviluppando un elemento a guscio sottile bidimensionale assialsimmetrico, ma tralasciarono lo sloshing del liquido a causa di complessità numeriche nell'accoppiamento delle equazioni del moto.

Ang (1980) estese il lavoro di Balendra e Nash per includere gli effetti dello sloshing del liquido utilizzando una matrice di accoppiamento per gli elementi del guscio e del fluido. Infine, Haroun e Housner (1981a, 1981b, 1982) studiarono lo sloshing del liquido in serbatoi cilindrici flessibili sottoposti a eccitazione della base orizzontale, impiegando metodi degli elementi finiti e degli elementi di contorno.

Negli stessi anni, parallelamente a questi tentativi di dare una descrizione analitica del fenomeno e della interazione elastica con i volumi di contenimento, per altro concentrata su sollecitazioni laterali, la *National Aeronautics and Space Administration* (NASA) ha intrapreso un esperimento denominato *Middeck 0-Gravity Dynamics Experiment* condotto sulla missione *Shuttle STS-48*, attraverso cui ha potuto studiare sperimentalmente il comportamento laterale dello slosh dei liquidi,

confrontando i risultati con esperimenti condotti a terra ed identificando gli effetti della gravità sul comportamento lineare e non lineare del fenomeno. Nel 1966 venne pubblicato dalla NASA il trattato *The Dynamic Behavior of Liquids in Moving Containers, with Applications to Space Vehicle Technology* (H. Norman Abramson), contenente una trattazione analitica dello sloshing laterale, dei suoi effetti non lineari e di smorzamento, interazione con le strutture elastiche ed effetti di stabilità e tecniche di simulazione. Le tecniche di simulazione della fluidodinamica computazionale (CFD) erano già note nei primi anni del 1960 grazie agli studi del fisico Francis Harvey Harlow e furono portate avanti grazie agli sforzi di ricerca dell'ingegnere americano John A. Swanson, che diede nascita del primo software nativo Ansys. Fu però poi durante gli anni '80 che sono comparsi i sistemi CAD/CAM/CAE ed è in questo decennio da cui la loro applicazione è divenuta di massa e hanno iniziato a diffondersi rapidamente. Con l'avvenire del nuovo millennio, grazie alla notevole evoluzione delle capacità computazionali, questa tecnica vide una virtuosa espansione.

La tecnica delle CFD rappresenta tutt'ora il principale strumento utilizzato anche per l'analisi della dinamica di sloshing dei liquidi e la progettazione delle strutture di contenimento ed elementi smorzanti, ma non è in grado di fornire soluzioni prestanti ai disturbi di controllo che il fenomeno genera, a causa degli elevati costi di tempo computazionale che impediscono di utilizzarla direttamente per sistemi di controllo e stabilità aumentata.

Alcune possibili soluzioni, applicabili in campo lineare, riguardano la possibilità di descrivere il fenomeno attraverso teorie a Metodi Meccanici Equivalenti costruite mediante algoritmi data driven che, sulla base di un set di dati di simulazione ne estraggono un modello predittivo.

Capitolo 2

Teoria del modello

2.1 Modello matematico

Il modello matematico che governa il moto di un fluido deriva dai principi della meccanica del continuo, fondandosi su:

1. L'ipotesi del continuo,
2. La conservazione della massa,
3. Il bilancio della quantità di moto.

In accordo al principio di conservazione della massa, è possibile affermare che, dato un volume di controllo $\forall$, la massa al suo interno varia nel tempo solo a causa dei flussi di massa che attraversano la superficie di controllo S. Questa considerazione è espressa in forma integrale dalla seguente equazione:

$$\frac{\partial}{\partial t}\int_{\forall}(\rho d\forall) = -\int_{S}\rho u \cdot n\, dS$$

Poiché il volume di controllo è costante nello spazio, la derivata temporale sul lato sinistro dell'equazione può essere spostata all'interno del segno di integrale. Inoltre, assumendo che il campo di velocità sia differenziabile e applicando il teorema della divergenza, si ottiene la seguente relazione:

$$\int_{\forall}\left[\frac{\partial \rho}{\partial t} + \nabla(\rho u)\right] d\forall = 0$$

Considerando la libertà nella scelta del volume di controllo $\forall$, ciò implica che la funzione integranda debba essere identicamente nulla, ovvero:

$$\frac{\partial \rho}{\partial t} + \nabla(\rho u) = 0$$

che esprime l'equazione di conservazione della massa in forma differenziale.

L'equazione di bilancio della quantità di moto è una diretta conseguenza della seconda legge della dinamica e descrive il concetto per cui, in un volume di controllo prestabilito, la variazione nella quantità di moto è uguale alla somma delle forze esterne che agiscono su quel volume. Pertanto, può essere espressa come segue:

$$\frac{\partial}{\partial t}\int_{\forall} \rho u \, d\forall = -\int_S \rho u (u \cdot n) dS + \int_{\forall} \rho F \, d\forall + \int_S n \cdot T \, dS$$

Nell'equazione sopra, il primo integrale sul lato destro rappresenta il flusso convettivo della quantità di moto, dove F è un campo vettoriale che denota le forze per unità di volume che agiscono sul volume del fluido, mentre T è il tensore degli sforzi che contempla le forze superficiali, sia normali che tangenziali, che agiscono sul confine di $\forall$.

In modo analogo a quanto già fatto per l'equazione di continuità, è possibile derivare in forma differenziale l'equazione di bilancio della quantità di moto, ottenendo:

$$\frac{\partial \rho u}{\partial t} = -\nabla \cdot (\rho u u) + \rho F + \nabla \cdot T$$

dove il tensore degli sforzi, per un fluido Newtoniano, è legato linearmente al tensore della deformazione D secondo la relazione costitutiva:

$$T = -\rho I - \frac{2}{3}\mu(\nabla \cdot u) I + 2\mu D$$

Dove p rappresenta la pressione, I indica il tensore unitario, μ è il coefficiente di viscosità dinamica e $D = \frac{1}{2}(\nabla u + \nabla u^T)$ rappresenta la parte simmetrica del tensore gradiente della velocità. Nel caso di un fluido incomprimibile, sostituendo l'espressione del tensore degli sforzi nella formula si giunge all'equazione di bilancio della quantità di moto:

$$\frac{\partial \rho u}{\partial t} = -\nabla \cdot (\rho u u) - \nabla p + \rho F + \nabla \cdot (2\mu D)$$

Applicando la proprietà algebrica

$$\nabla \cdot (\rho u u) = \rho u \cdot \nabla u + u \nabla \cdot (\rho u)$$

ed utilizzando l'equazione di bilancio della massa, si ottiene:

$$\rho \frac{\partial u}{\partial t} - \nabla p + \rho F + \nabla \cdot (2\mu D)$$

Limitando l'attenzione ai fluidi incomprimibili, è agevole derivare le equazioni di Navier-Stokes:

$$\nabla \cdot u = 0$$

$$\rho \frac{du}{dt} = -\nabla p + \rho F + \mu \nabla^2 u$$

La completa descrizione del comportamento di un fluido richiede, in realtà, anche l'incorporazione dell'equazione di conservazione dell'energia. È infatti il sistema formato dalle equazioni di conservazione della massa, della quantità di moto e dell'energia che descrive in modo completo il cosiddetto "problema fluido-termodinamico", tenendo conto della possibilità di scambi termici tra il campo fluido e l'ambiente esterno. Tuttavia, quando il moto del fluido avviene in condizioni isoterme (o quando, approssimativamente, può essere considerato tale), l'equazione dell'energia perde rilevanza.

Le equazioni di Navier-Stokes, da sole, non sono sufficienti per risolvere completamente il problema in forma chiusa. Per chiudere il sistema di equazioni e ottenere soluzioni praticabili, è necessario includere le condizioni al contorno effetto delle pareti solide, le quali impongono condizioni di no-slip (velocità del fluido nulla) e condizioni dinamiche e cinematiche per la superficie libera del fluido bifase, tramite l'equazione di fase.

L'equazione di fase nel metodo Volume of Fluid (VOF) utilizza la frazione volumetrica α per descrivere la frazione di volume di una cella occupata da un fluido, come l'acqua, in un sistema bifase aria-acqua. L'equazione di fase per è α :

$$\frac{\partial \alpha}{\partial t} + \vec{u} \cdot \nabla \alpha = 0$$

dove α è la frazione volumetrica del fluido (in questo caso acqua) e varia tra 0 e 1.

Le proprietà del fluido, come la densità ρ e la viscosità μ, sono funzioni della frazione volumetrica α:

$$\rho(\alpha) = \alpha \rho_{liquido} + (1 - \alpha)\rho_{gas}$$

$$\mu(\alpha) = \alpha \mu_{liquido} + (1 - \alpha)\mu_{gas}$$

Per chiudere le equazioni Reynolds-Averaged Navier-Stokes è infine necessario definire i modelli di turbolenza.

2.2 Metodo dei volumi finiti

Lo scopo della discretizzazione numerica è quello di sostituire lo spazio in cui la soluzione esatta del problema differenziale è definita con uno spazio vettoriale di dimensione finita, in cui è possibile cercare una soluzione approssimata. La risoluzione numerica delle equazioni di Navier-Stokes si sviluppa principalmente in tre fasi:

- Discretizzazione del dominio computazionale;
- Discretizzazione delle equazioni;
- Soluzione numerica delle equazioni discretizzate

In sintesi, la base di un codice CFD (Computational Fluid Dynamics) consiste nel passaggio da grandezze ed equazioni continue a grandezze ed equazioni discretizzate, utilizzando griglie. In un dominio continuo, ogni variabile fluidodinamica è definita in ogni punto, mentre in un dominio discreto, le variabili fluidodinamiche sono definite solo nei punti della griglia.

Nell'ambito di una soluzione tramite codice CFD, le variabili fluidodinamiche sono risolte direttamente solo nei punti della griglia, con i valori delle variabili in altri punti ottenuti tramite interpolazione dai valori ottenuti sui punti della mesh.

Esistono diversi metodi per approssimare la soluzione di problemi descritti da equazioni differenziali alle derivate parziali; OpenFOAM permette di adottare il metodo dei volumi finiti. Benché competa in alcuni contesti con altre strategie numeriche (come il metodo delle differenze finite o il metodo degli elementi finiti), il metodo dei volumi finiti è considerato uno dei migliori per il calcolo fluidodinamico su geometrie complesse e particolari.

Nel metodo dei volumi finiti, il dominio della soluzione è suddiviso in un numero finito di volumi di controllo, corrispondenti alle celle della griglia. L'obiettivo è ottenere un sistema di equazioni algebriche lineari, con il numero di incognite pari al numero di celle della griglia. Le grandezze fisiche sulle superfici di controllo (le facce dei volumi finiti) sono approssimate usando valori medi. Questo comporta l'uso di schemi numerici per interpolare le proprietà del fluido dai centri delle celle alle facce delle celle. Gli schemi possono variare in accuratezza e complessità, con opzioni comuni come lo schema ad upwind, lo schema centrale differenze e schemi di ordine superiore. Il bilancio dei flussi di massa, quantità di moto ed energia attraverso le facce dei volumi finiti è calcolato sommando i contributi di ciascuna faccia. Questo garantisce che la conservazione delle grandezze fisiche sia rispettata su ogni volume finito. Le equazioni discretizzate risultano in un sistema di equazioni algebriche che sono risolte iterativamente tramite metodi iterativi comuni quali il metodo di Gauss-Seidel, il metodo dei gradienti coniugati e il metodo SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) per il calcolo della pressione e della velocità.

L'approccio tipico richiede di discretizzare il dominio fluido in celle elementari per ottenere una griglia di calcolo (o mesh), sulla quale si applicano metodi di risoluzione iterativi per risolvere le equazioni di Navier-Stokes o di Eulero. Nei vari approcci, la procedura di analisi segue generalmente queste fasi principali:

- Definizione della geometria: si stabilisce il dominio fisico del problema da analizzare.
- Discretizzazione del volume fluido: il volume occupato dal fluido viene suddiviso in un gran numero di celle elementari, generando una griglia di calcolo.

- Definizione del modello fisico e numerico: si scelgono le equazioni del moto, l'equazione dell'energia e le equazioni delle specie, i modelli di turbolenza, e si stabilisce il metodo di discretizzazione delle equazioni e gli algoritmi per la loro risoluzione.
- Definizione delle condizioni al contorno: si specificano le proprietà del fluido nel dominio di calcolo. Per problemi dipendenti dal tempo, si definiscono anche le condizioni iniziali.
- Risoluzione iterativa delle equazioni: le equazioni vengono risolte iterativamente, interrompendo il calcolo una volta raggiunto il grado di accuratezza desiderato.
- Visualizzazione dei risultati: i risultati prodotti vengono visualizzati tramite un post-processore.
- Analisi dei risultati: si effettua un'analisi dettagliata dei risultati ottenuti.
- Integrazione iterativa con l'analisi strutturale: combinando i risultati dell'analisi fluidodinamica con quelli dell'analisi strutturale a elementi finiti, è possibile valutare non solo gli effetti dei flussi attorno ai solidi, ma anche le loro deformazioni, permettendo una valutazione approfondita degli effetti variabili nel tempo delle pressioni su una superficie. Questo aspetto è stato trascurato nel trattato.

Capitolo 3

OpenFOAM e implementazione del problema

3.1 Introduzione a OpenFOAM

Tra i molteplici codici di risoluzione numerica per l'analisi fluidodinamica spicca, per interesse accademico, libertà di licenza ed espandibilità, il toolbox OpenFOAM; sviluppato inizialmente come Software commerciale e rilasciato con il 70% del codice sorgente, venne poi rimpiazzato dalla distribuzione completamente open source a partire dal dicembre del 2004, permettendo così una notevole crescita della base utenti. Il software è sviluppato dalla società commerciale OpenCFD Ltd, la quale lo distribuisce senza alcun costo sotto licenza GPL (General Public License). Questa licenza è ampiamente diffusa nel campo del software libero e assicura la libertà di utilizzo, copia, modifica e distribuzione.

OpenFOAM, acronimo di Open Field Operation and Manipulation, rappresenta un set di strumenti utili per la soluzione delle equazioni di campo. Questo software non è limitato alla fluidodinamica, ma trova applicazione anche nella dinamica dei solidi, nell'elettromagnetismo e nel settore finanziario. Presenta una struttura modulare, scritta in linguaggio C++, organizzata gerarchicamente e completamente accessibile, garantendo quindi un alto livello di espandibilità. Una delle caratteristiche più interessanti del programma è la sua capacità di

eseguire calcoli in parallelo. In pratica, l'utente non deve gestire direttamente la suddivisione del lavoro tra più processori; è il programma stesso a gestire l'interfaccia tra l'algoritmo e il processo di parallelizzazione.

La modellazione di fenomeni fisici e matematici richiede spesso la rappresentazione di equazioni complesse in un linguaggio di programmazione. In questo contesto, l'utilizzo di un linguaggio di dominio specifico (DSL) può semplificare notevolmente il processo di sviluppo e la comprensione del codice. OpenFOAM implementa un DSL denominato "Equation Mimicking", progettato per consentire la rappresentazione efficiente di equazioni matematiche all'interno dell'ambiente di programmazione C++. La sintassi del DSL è stata progettata per essere intuitiva e diretta, consentendo agli sviluppatori di esprimere equazioni complesse in modo chiaro e conciso. Ad esempio, l'implementazione dell'equazione della quantità di moto può essere realizzata utilizzando il DSL in modo efficiente e leggibile come segue:

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \Phi \mathrm{U} - \mu \nabla^2 U = -\nabla p$$

che nel codice si scrive come:

```
fvm: :ddt(rho, U)
+ fvm: :div(phi, U)
- fvm: :laplacian(mu, U)
==
- fvc: :grad(p)
```

Le librerie di OpenFOAM forniscono gli strumenti necessari per creare solvers e utilities, che consentono la risoluzione e la visualizzazione di problemi termofluidodinamici. I solvers di OpenFOAM sono progettati per risolvere equazioni di bilancio termofluidodinamiche, mentre le utilities gestiscono il pre e post-processing dei dati. Le utilities includono meshatori come blockMesh e cfMesh, nonché programmi di

visualizzazione come paraView. L'esecuzione di queste applicazioni è semplice, poiché possono essere lanciate direttamente da un terminale Linux, o, come nel caso di studio, da una Windows Subsystem for Linux (WSL).

OpenFOAM utilizza file di testo denominati dizionari per specificare le impostazioni delle applicazioni attraverso l'uso di parole chiave (keywords). Questi dizionari delineano il comportamento delle utilities e consentono agli utenti di personalizzare le loro simulazioni in base alle specifiche esigenze. I dizionari di riferimento propri del codice sorgente di ciascuna simulazione si configurano come file di testo, riscrivibili e personalizzabili con qualsiasi editor come Notepad++; si riporta, a titolo di esempio, un FoamFile contentente le informazioni relative alla costante di accelerazione gravitazionale

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);
```

ciascuna keywords viene seguita da un dataEntry nel seguente formato:

`<keyword> <dataEntry>;`

La maggior parte dei file di dati, come ad esempio controlDict, sono essi stessi dei dizionari in quanto contengono una serie di voci di parole chiave. Ogni dizionario può contenere uno o più sotto-dizionari, generalmente

indicati da un nome di dizionario e le rispettive keywords racchiuse tra parentesi graffe come segue.

```
<dictionaryName>
{
        ... keyword entries ...
}
```

I sotto-dizionari possono essere inseriti all'interno di altri, come dimostrato nell'esempio successivo. Nell'estrazione da un file di dizionario fvSolution, si trovano due dizionari: solvers e PIMPLE. Il dizionario solvers include sotto-dizionari dedicati a diverse equazioni matriciali relative a differenti variabili di soluzione, come ad esempio p, U e k

```
solvers
{
    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       10e-06;
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "(U|k|epsilon).*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       10e-06;
        relTol          0;
        minIter         1;
    }
}

PIMPLE
{
      momentumPredictor   no;
      nOuterCorrectors    1;
      nCorrectors         4;
      nNonOrthogonalCorrectors 0;
      pRefCell    0;
      pRefValue    0;

}
```

Tutti i file dati letti e scritti da OpenFOAM iniziano con un dizionario chiamato FoamFILE che contiene un insieme standard di voci di parole chiave, elencate di seguito:

- `version:` versione del formato di I/O, facoltativa, predefinita a 2.0
- `format:` formato dei dati, ascii o binario
- `class:` classe relativa ai dati, sia un dizionario che un campo, ad esempio `uniformDimensionedVectorField;`
- `object:` nome del file, ad esempio controlDict (obbligatorio, ma non utilizzato)
- `location:` percorso del file (opzionale)

3.2 Preparazione dell'ambiente e architettura software

Per semplificare l'uso di OpenFOAM, è consigliabile eseguire il sourcing dello script

`//etc/bashrc`

Nella shell di comando Linux. Questa si trova in tutti i sistemi nativi linux o come applicazione eseguibile Windows nel caso in cui si faccia uso, come in questo caso, di una WSL. WSL è una funzionalità di Windows che consente di eseguire un ambiente Linux direttamente su un computer Windows, eliminando la necessità di utilizzare una macchina virtuale separata o avviare un doppio sistema operativo. WSL è stato progettato per offrire agli sviluppatori un'esperienza semplice e produttiva, consentendo loro di utilizzare sia Windows che Linux contemporaneamente, facilitando lo sviluppo di applicazioni cross-platform e l'esecuzione di strumenti specifici di Linux su un sistema Windows.

Attraverso lo script riportato si configura una serie di alias (comandi abbreviati) e variabili d'ambiente contenenti principalmente percorsi di cartelle, per semplificare l'uso quotidiano del software. Inoltre, lo script aggiunge il percorso di tutte le applicazioni eseguibili di OpenFOAM alla variabile d'ambiente $PATH, la quale contiene i percorsi degli eseguibili accessibili dalla riga di comando.

Le variabili d'ambiente si dividono tra cartelle di progetto e cartelle utente. Vengono riportate le principali variabili di ambiente ed Alias associati

Cartella	Percorso	Alias
`$FOAM_INST_DIR`	`/opt/OpenFOAM`	
`WM_PROJECT_DIR`	`/opt/openfoam11`	`foam`
`$FOAM_TUTORIALS`	`/opt/openfoam11/tutorials`	`tut`
`$FOAM_APP`	`/opt/openfoam11/applications`	`app`
`$FOAM_SOLVERS`	`/opt/openfoam11/applications/solvers`	`sol`
`$FOAM_UTILITIES`	`/opt/openfoam11/applications/utilities`	`util`
`$FOAM_SRC`	`/opt/openfoam11/src`	`src`

Tramite il comando `mkdir -p $FOAM_RUN` dalla shell vengono settate le cartelle utente:

`M_PROJECT_USER_DIR`	`OME/OpenFOAM/username-11`	
`OAM_RUN`	`OME/OpenFOAM/username-11/run`	`n`

Si osserva ora la struttura della cartella principale di OpenFOAM

`$WM_PROJECT_DIR`

Subdirectories	Contenuto
`applications`	codice sorgente di solvers e utilities
`bin`	shell scripts di uso generale
`doc`	documentazione
`etc`	file di setup dell'ambiente e altro
`platforms`	file binari di applicazioni e librerie
`src`	codice sorgente delle librerie
`tutorials`	casi di esempio per la maggior parte delle applicazioni
`wmake`	impostazioni per la compilazione

La subdirectory tutorials contiene dei codici di simulazione di esempio che possono essere modificati ed adattati per personalizzare le simulazioni. È possibile lanciare una simulazione di tutorial entrando nella directory *tutorials* attraverso l'alias `tut`, copiare la cartella della simulazione che si vuole lanciare con il comando

```
cp ./SimulazioneCheSiVuoleEseguire $FOAM_RUN
```

La stessa operazione è possibile eseguirla senza i comandi del terminale ma tramite la finestra di navigazione del browser delle cartelle. In questo caso occorre percorrere tutta la path corretta per arrivare alla directory tutorials, che nel caso di WSL sarà una cartella di rete con un percorso tipo $\backslash\backslash wsl.localhost\backslash Ubuntu\backslash opt\backslash openfoam11\backslash tutorials\backslash$

Selezionare la simulazione che si intende eseeguire, copiarla tramite shortcut o con finestra apribile da tasto destro del mouse, quindi spostarsi nella cartella utente di foam run, che presenterà un percorso simile al seguente

$\backslash\backslash wsl.localhost\backslash Ubuntu\backslash home\backslash utente\backslash OpenFOAM\backslash volpatto\text{-}11\backslash run$

ed incollarla.

Con questa operazione si evita di modificare direttamente i file del tutorial e di scrivere i risultati nella medesima cartella, creando una directory di simulazione personalizzata (che si può rinominare a piacere).

Una volta creata la path di simulazione, tramite la shell si entra nel percorso tramite comando

```
cd $FOAM_RUN/NomeSimulazione
```

Una volta fatto ciò è possibile lanciare il comando del solutore o dell'utility qualora sia previsto (viene tipicamente specificata la sequenza di utility da lanciare in un file README oppure viene fornito direttamente un bash file chiamato Allrun da eseguire).

3.3 Struttura di un caso

I passaggi necessari per condurre un'analisi CFD con OpenFOAM includono:

- Costruzione della griglia.
- Selezione del solutore appropriato.
- Configurazione delle condizioni iniziali e dei contorni necessari.
- Impostazione dei dizionari.
- Avvio del calcolo.
- Monitoraggio dei residui della simulazione.
- Post-processing dei dati.

A differenza di altre applicazioni, come numerosi software commerciali, in OpenFOAM la griglia, le impostazioni e i risultati sono suddivisi tra vari file. Il caso è memorizzato in una cartella con una struttura fissa `case_directory` ha una struttura fissa che comprende le sottocartelle:

- `0`: Contiene le definizioni delle condizioni iniziali e ai contorni.

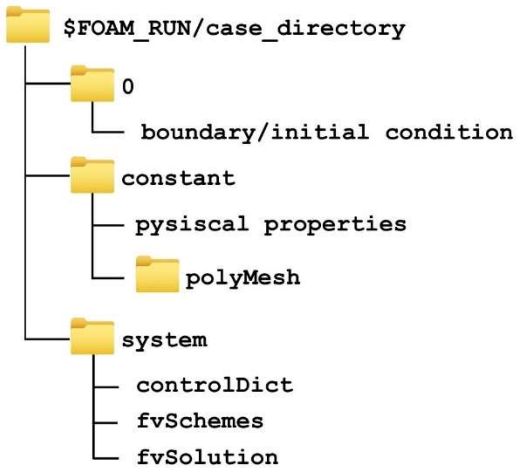


Figura 1 Struttura delle directory di un caso di simulazione

- `constant`: Contiene la scelta dei modelli e le definizioni delle proprietà dei materiali, inclusi i file della mesh ("polyMesh").
- `system`: Contiene le impostazioni del solutore, come gli schemi di discretizzazione e le tolleranze di convergenza.

Nel file *controlDict* vengono definiti tutti i parametri temporali, quali l'inizio e la fine dell'intervallo di integrazione, il passo temporale e il numero massimo di Courant. Inoltre, all'interno di questo file, nella sezione *function object*, sono specificate tutte le grandezze aggiuntive che si desidera calcolare insieme alle variabili principali. Nel file *fvSchemes* viene indicato lo schema numerico utilizzato per la discretizzazione. Infine, nel file *fvSolution*, vengono specificate le metodologie per la risoluzione del sistema di equazioni discretizzate e le tolleranze di iterazione da applicare.

La sottocartella `polyMesh` contenuta in `constant` è il risultato dell'operazione di Meshing operata in fase di preprocessing utilizzando una delle utilities compatibili con OpenFOAM.

3.4 Scelta del solutore

Questo studio utilizza il modulo InterFoam in OpenFOAM, che è un risolutore NS a due fasi per fluidi immiscibili incompressibili e isoterma con opzionale movimento della griglia e cambiamenti nella topologia della griglia, compresa la rimappatura adattiva.

La scelta non è stata univoca ma si è testato parallelamente anche il solutore foamRun con l'esecuzione dell'utility movingMesh, ma le simulazioni risultanti richiedevano maggiori tempi computazionali, rendendo questo modello più adatto ad una simulazione qualitativa ma

meno performante nell'ottica di eseguire diverse simulazioni variando i parametri di configurazione come livello di rimpimento, posizione e forma delle paratie anti sciabordio e profilo di missione.

L'approccio di cattura dell'interfaccia utilizzato nel modello interFoam è il volume di fluido (VOF) basato sulla frazione di fase. Le equazioni di continuità, equazioni di momento e equazioni di fase sono, rispettivamente, le seguenti.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0$$

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u u) - \nabla \cdot \tau = Ck\nabla\alpha - gh\nabla\rho - \nabla p_{rgh}$$

$$\frac{\mathcal{D}\alpha}{\mathcal{D}t} = \frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha u) = 0$$

dove ρ è la densità, U è il vettore della velocità del fluido, τ è lo sforzo tangenziale, C è il coefficiente di tensione superficiale che è impostato a 0 nell'indagine attuale, k è la curvatura dell'interfaccia, α è la frazione di volume, g è l'accelerazione di gravità, h è il vettore posizione del centro della griglia misurato dall'origine delle coordinate, p_{rgh} è la pressione dinamica. La densità del fluido ρ e il coefficiente di viscosità μ sono rispettivamente calcolati dalle densità (ρ_1, ρ_2) e dalla viscosità (μ_1, μ_2) di due fluidi utilizzando la frazione di volume α.

$$\rho = \alpha\rho_1 + (1 - \alpha) \times \rho_2$$

$$\mu = \alpha\mu_1 + (1 - \alpha) \times \mu_2$$

OpenFOAM utilizza il Metodo dei Volumi Finiti per discretizzare le sue equazioni fondamentali. Per gestire i termini derivati rispetto al tempo, ad esempio $\partial U/\partial t$ e $\partial \alpha/\partial t$, viene utilizzato uno schema di discretizzazione implicito di primo ordine di tipo Euler. Lo schema di discretizzazione lineare di Gauss è selezionato per l'approssimazione del gradiente, ad

esempio $\nabla \cdot U$. Viene considerato il metodo lineare corretto di Gauss per gli schemi laplaciani come ad esempio ∇p_{rgh}, $\nabla \rho$. Per quanto riguarda i termini di divergenza come $\nabla \cdot (UU)$ e $\nabla \cdot (U\alpha)$, viene utilizzato lo schema di van Leer. L'algoritmo PIMPLE, che è una combinazione di PISO (Pressure Implicit with Splitting of Operator) e SIMPLE (Semi-Implicit Method for Pressure-Linked Equations), viene utilizzato per il disaccoppiamento velocità-pressione. Il processo di soluzione di InterFoam è illustrato per completezza:

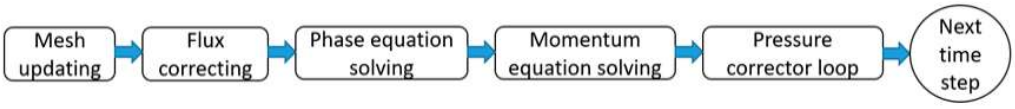


Figura 2 Processo di soluzione implementato dal solutore

Capitolo 4

Simulazione

4.1 Pre-processing: preparazione della Mesh

Partendo da un modello CAD del serbatoio questo è stato esportato in formato STL, *STereo Lithography interface format*, un tipo di file, sia in formato binario che ASCII, sviluppato per i software di stereolitografia CAD, ampiamente impiegato nel processo di prototipazione rapida tramite software. Un file con estensione .stl rappresenta un solido il cui contorno superficiale è stato discretizzato in triangoli. Ogni triangolo è descritto attraverso le coordinate X, Y e Z dei suoi tre vertici, insieme a un vettore che specifica l'orientamento della normale alla superficie.

Viene mostrato di seguito un modello CAD completo di tutte le geometrie del sistema di spraying, viti di accoppiamento, ed il corrispettivo modello scorporato di tutti gli elementi non utili al fine della simulazione.

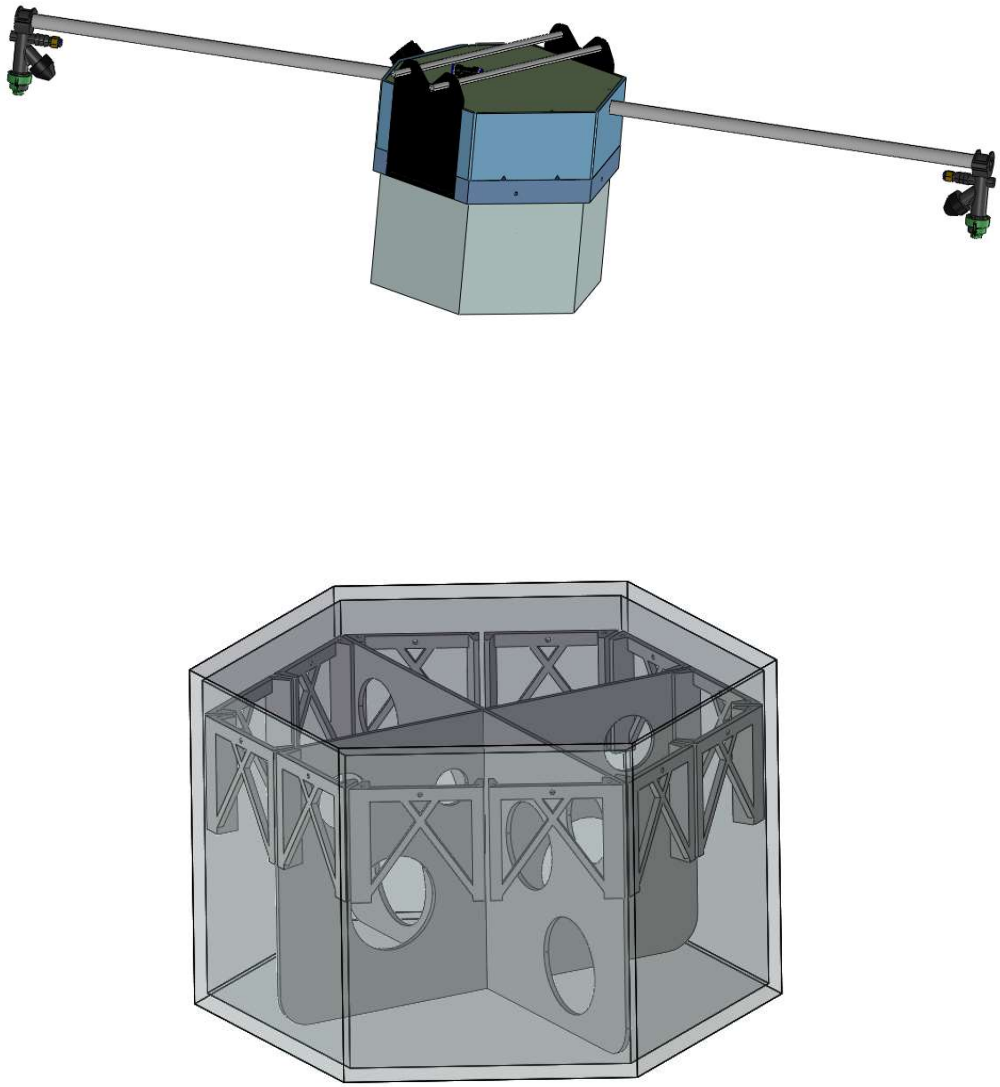


Figura 3 Modello CAD completo (in alto) e sezione serbatoio in trasparenza (in basso) con dettaglio delle paratie

4.2 Meshatore cfMesh e parametrizzazione

Il file prodotto viene quindi utilizzato come risorsa nella compilazione di cfMesh, una libreria open Source ottimizzata per OpenFoam che non viene rilasciata con la suite. Le patch sono definite sulla geometria iniziale e poi applicate alla mesh volumetrica. I sottoinsiemi (punti, facce) sono impiegati nella creazione della mesh, ma non sono direttamente trasferiti alla mesh volumetrica.

Si può osservare come la geometria si compone di tre elementi principali:

- il serbatoio di contenimento
- le due paratie montate a croce
- le guide di ancoraggio per le paratie

Una prima analisi prevede di modellare il fenomeno dello sloshing el solo serbatoio senza guide di ancoraggio e paratie. Successivamente vengono introdotti gli altri elementi che prevedono un infittimento della mesh, operato localmente, e alcune considerazioni aggiuntive come verrà mostrato in seguito.

Per la creazione della griglia 3D è stato usato il dizionario *cartesianMesh* di cfMesh, eseguibile tramite omonimo comando sulla shell dedicata del software di meshing open source. La griglia generata è a celle esaedriche. Definita la geometria e il tipo di griglia che si vuole creare, si passa alla compilazione del meshDict, nel quale si specificano le diverse risoluzioni che si vogliono avere all'interno del dominio. Il dizionario definisce alcune keywords per la parametrizzazione della griglia.

surfaceFile definisce il file di importazione della geometria; questo può essere direttamente in formato *stl* esportato dal CAD oppure in formato *fms* più consono per cfMesh

maxCellSize rappresenta la dimensione predefinita della cella utilizzata per il lavoro di meshing. È la dimensione massima della cella generata nel dominio

boundaryCellSize viene utilizzata per il raffinamento delle celle al bordo. Si tratta di un'opzione globale e la dimensione della cella richiesta viene applicata ovunque sul bordo. *boundaryCellSizeRefinementThickness* specifica la distanza dal bordo alla quale la *boundaryCellSize* viene ancora applicata.

minCellSize è un'opzione globale che attiva il raffinamento automatico del modello di mesh. Questa opzione esegue il raffinamento nelle regioni in cui le celle sono più grandi della dimensione caratteristica stimata. Il valore scalare fornito con questa impostazione specifica la dimensione della cella più piccola consentita da questa procedura. Questa opzione è utile per simulazioni rapide perché può generare mesh in geometrie complesse con un basso sforzo dell'utente. Tuttavia, se è richiesta una alta qualità della mesh, fornisce suggerimenti su dove è necessario un certo raffinamento della mesh.

localRefinement consente di definire regioni di raffinamento locale. Si tratta di un dizionario di dizionari e ciascun dizionario all'interno del dizionario principale *localRefinement* è denominato da un sottoinsieme di patch o facce nella geometria che viene utilizzato per il raffinamento. La dimensione della cella richiesta per un'entità è controllata dal termine *cellSize* e da un valore scalare, oppure specificando il termine *additionalRefinementLevels* e il numero desiderato di raffinamenti

rispetto alla dimensione massima della cella. È possibile specificare patch tramite espressioni regolari.

surfaceMeshRefinement consente di utilizzare reti di superficie come zone di raffinamento nella mesh. Viene specificata come un dizionario di dizionari in cui ogni zona di raffinamento è un sotto-dizionario all'interno del dizionario *surfaceMeshRefinement*. La rete di superficie utilizzata per la zona di raffinamento è fornita con il termine *surfaceFile*, e la dimensione della cella richiesta per un'entità è controllata dal termine *cellSize* e da un valore scalare, oppure specificando il termine *additionalRefinementLevels* e il numero desiderato di raffinamenti rispetto alla dimensione massima della cella

I *boundaryLayers* in cfMesh sono creati estrudendo le facce di confine della mesh volumetrica verso l'interno e non possono essere generati in anticipo rispetto al processo di meshing. Il loro spessore è controllato dalla dimensione della cella specificata sul bordo e il mesher tende a produrre strati con uno spessore simile a quello della cella. Gli strati possono estendersi su più patch se queste condividono spigoli concavi o angoli con valenza maggiore di tre. Tutte le impostazioni relative agli strati di bordo sono contenute in un dizionario denominato *boundaryLayers*, Le opzioni fornite dal dizionario includono:

nLayers: specifica il numero di strati generati nella mesh. Se non specificato, il flusso di lavoro del meshing utilizza il numero predefinito di strati, che può essere uno o zero.

thicknessRatio: rappresenta il rapporto tra lo spessore di due strati successivi. Il valore deve essere maggiore di 1, con un valore predefinito di 1.

maxFirstLayerThickness: garantisce che lo spessore del primo strato di bordo non superi mai il valore specificato.

L'impostazione *patchBoundaryLayers* è un dizionario utilizzato per definire le proprietà locali degli strati di bordo per singole patch. È possibile specificare *nLayers, thicknessRatio* e *maxFirstLayerThickness* per ogni patch all'interno di un dizionario che porta lo stesso nome della patch. Di default, il numero di strati generati su una patch è governato dal numero globale di strati o dal massimo numero di strati specificato su qualsiasi patch che forma un layer continuo insieme alla patch in questione. L'opzione *allowDiscontinuity* garantisce che il numero di strati necessari per una patch non si estenda ad altre patch nello stesso layer. Le patch possono essere specificate utilizzando espressioni regolari.

Una prima fase di analisi è stata operata mediante uno studio parametrico per ottenere una soluzione di ottimo in grado di mantenere contenuto il limite di nodi nella mesh senza comprometterne la qualità, iterando per diversi valori di massima e minima dimensione della cella. Viene riportato di seguito il prototipo del file *meshDict*

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | cfMesh: A library for mesh generation           |
|  \\    /   O peration     |                                                 |
|   \\  /    A nd           | Author: Franjo Juretic                          |
|    \\/     M anipulation  | E-mail: franjo.juretic@c-fields.com             |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version   2.0;
    format    ascii;
    class     dictionary;
    location  "system";
    object    meshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

surfaceFile "Tankm.fms";

maxCellSize 0.008;

boundaryCellSize 0.004;
boundaryCellSizeRefinementThickness 0.035;

minCellSize 0.005;
```

in questa fase la geometria importata per la definizione della griglia, il file "Tankm.fms", contiene tutti gli elementi.

La geometria *.fms* è stata creata tramite utility *surfaceFeatureEdges* che viene utilizzata per generare spigoli caratteristici nella geometria. Nel caso in cui l'output sia un file fms, gli spigoli generati vengono memorizzati come spigoli caratteristici. In caso contrario, genera patch delimitate dagli spigoli caratteristici selezionati. I punti superficiali in cui si incontrano tre o più spigoli caratteristici vengono trattati come angoli. Questa utility può essere compilata fornendo come opzione aggiuntiva l'angolo caratteristico in gradi.

Si osservano di seguito l'output che si ha senza utilizzo del dizionario *surfaceFeatureEdges,* vale a dire importando come *surfaceFile* direttamente il file .stl della geometria CAD

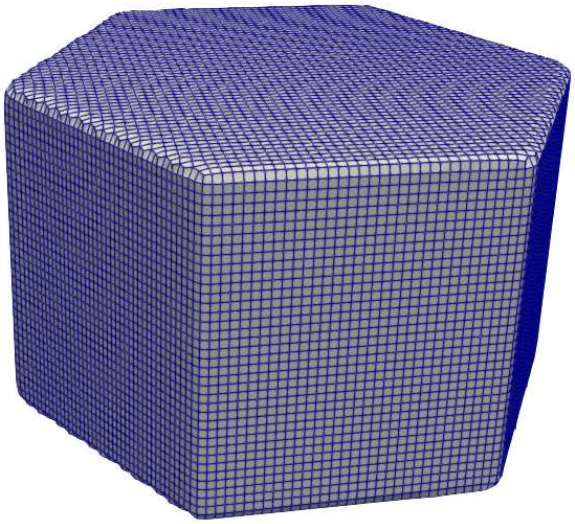


Figura 4 Mesh alla prima iterazione

Nella successiva iterazione per l'affinamento della mesh invece è stato fatto uso del dizionario *surfaceFeatureEdges* con parametrizzazione dell'angolo in ingresso pari a 15°, scelto in modo tale da risultare sufficientemente piccolo e al contempo sottomultiplo di 90° (spigoli verticali) e 120° (spigoli nel piano xy).

In virtù della logica di individuazione degli spigoli operata da *surfaceFeatureEdges,* si ruota inizialmente la parte solida utilizzando una funzione di rotazione del solido su Solidworks in modo tale che l'asse di estrusione della base esagonale coincida con x, quindi viene generata la mesh tramite utility *surfaceFeatureEdges* con angolo caratteristico di input 15° e successivamente si ruota la Mesh generata tramite l'utility *rotateMesh* per riorientare correttamente la geometria. I risultati di

questa operazione, per quanto macchinosa, sono ben apprezzabili; si osserva immediatamente come l'elaborazione degli spigoli sia ottimale.

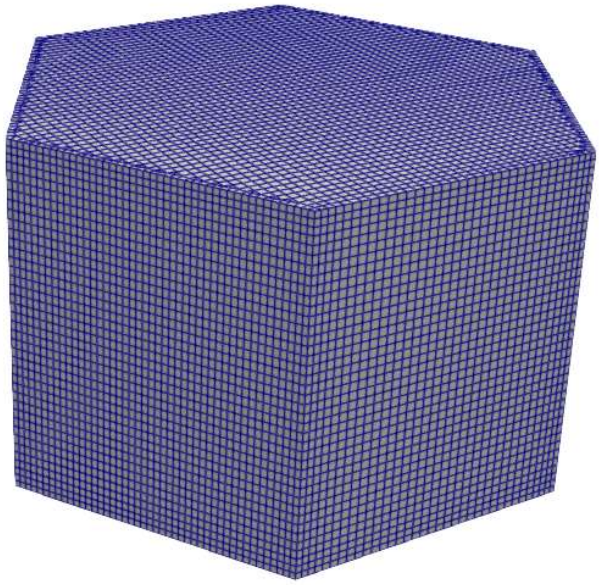


Figura 5 Mesh iterata con affinamento degli spigoli tramite feature *surfaceFeatureEdges* sulla geometria pre-ruotata

Anche per la successiva generazione della mesh, come volume di campo fluido interno, relativa al serbatoio con deflettori anti-sciabordio, che verrà mostrata nelle sezioni successive, fa uso della feature *surfaceFeatureEdges.* Il dettaglio della parametrizzazione di ciascuna mesh corrispondente alle diverse configurazioni studiate viene riportato nel capitolo 5 dell'elaborato. In questa sezione si descrive l'approccio utilizzato in tutti i casi, i metodi per valutare la qualità e le principali caratteristiche del meshatore.

4.3 Qualità della mesh e affinamento

La qualità della mesh è stata monitorata tramite dizionario *checkMesh* di cui viene riportato il log di output

```
Mesh stats
    points:           124887
    internal points:  109129
    faces:            360022
    internal faces:   344142
    cells:            117588
    faces per cell:   5.9884
    boundary patches: 1
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     116472
    prisms:        248
    wedges:        0
    pyramids:      620
    tet wedges:    0
    tetrahedra:    248
    polyhedra:     0

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
    Patch               Faces    Points   Surface topology
    emptyRot            15880    15758    ok (closed singly connected)

Checking geometry...
       Overall domain bounding box (-0.14192 -0.163945 0.0200364) (0.144612
       0.163949 0.212536)
    Mesh (non-empty, non-wedge) directions (1 1 1)
    Mesh (non-empty) directions (1 1 1)
    Boundary openness (1.9221e-015 -1.44857e-015 -2.56126e-016) OK.
    Max cell openness = 2.9162e-016 OK.
    Max aspect ratio = 5.66313 OK.
       Minimum face area = 1.83941e-006. Maximum face area = 3.16394e-005.
       Face area magnitudes OK.
       Min volume = 1.61211e-009. Max volume = 1.47701e-007.  Total volume =
       0.0134461.  Cell volumes OK.
    Mesh non-orthogonality Max: 37.0857 average: 2.0525
    Non-orthogonality check OK.
    Face pyramids OK.
    Average skewness 0.014908
    Max skewness = 0.550659 OK.
    Coupled point location match (average 0) OK.

Mesh OK.
```

Dal log file si osservano alcuni importanti indicatori relativi alle celle che compongono la mesh:

- Non-ortogonalità
- Skewness (Asimmetria)
- Aspect ratio (AR)

4.3.1 Non ortogonalità

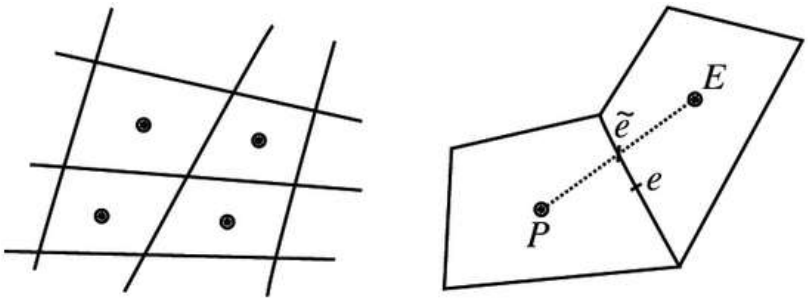


Figura 6 Immagine rappresentativa del parametro *non-orthogonality* della mesh, valutata tra celle adiacenti

L'assenza di ortogonalità in una mesh, che comprende elementi di varie dimensioni (1D, 2D o 3D), si riflette nella configurazione delle sue geometrie, che possono variare in prossimità della regolarità. Formalmente, l'ortogonalità della mesh è definita come la differenza angolare tra il segmento che collega i centri di due celle e la normale alla faccia condivisa. La non-ortogonalità di una cella misura quindi quanto l'angolo tra due celle adiacenti si discosti dal valore ideale (ad esempio, 90° per facce quadrangolari o 60° per facce triangolari). Angoli lontani dall'ottimale aumentano notevolmente la complessità del calcolo dei gradienti delle grandezze e compromettono la continuità della soluzione. L'obiettivo è quello di minimizzare questo parametro. È importante notare che la non-ortogonalità della mesh è definita come il massimo

valore di non-ortogonalità presente all'interno della stessa; in altre parole, se si considera che due celle adiacenti hanno una non-ortogonalità pari a 0 quando formano un angolo perfettamente ideale, allora la non-ortogonalità della mesh è pari alla massima non-ortogonalità tra tutte le coppie di celle adiacenti.

- Per non-ortogonalità > 85: è praticamente impossibile ottenere una soluzione accettabile; si dovrà procedere alla revisione della mesh.
- Per $70 <$ non-ortogonalità < 85: valore di non-ortogonalità accettabile che richiede però alcuni accorgimenti da parte del progettista; sarà per esempio richiesto l'utilizzo di correttori all'interno dei metodi di risoluzione numerici. La soluzione ottenuta potrebbe essere attendibile, ma andrà verificata con attenzione.
- Per $50 <$ non-ortogonalità < 70: valore di non-ortogonalità discreto, viene tipicamente considerato come un buon compromesso per ottenere soluzioni relativamente affidabili senza incrementare eccessivamente il costo computazionale.
- Per $25 <$ non-ortogonalità < 50: valore di non-ortogonalità buono, permette di utilizzare schemi di calcolo meno robusti e più accurati rispetto a mesh meno regolari; con valori di non-ortogonalità inferiori a 40 si possono sfruttare schemi di calcolo del secondo ordine, piuttosto che del primo, migliorando considerevolmente l'affidabilità della soluzione.
- Per non-ortogonalità < 25: valore di non-ortogonalità ottimale grazie al quale è possibile utilizzare in tranquillità schemi risolutivi del secondo ordine, ottenendo soluzioni molto precise ed attendibili. Il raggiungimento di un valore così ridotto di non-

ortogonalità rischia, in alcuni casi, di pesare gravemente sul tempo, e dunque sul costo, computazionale.

In questo caso la costruzione della mesh permette di usare schemi del secondo ordine, che verranno definiti nell'apposito dizionario *fvSolution* della soluzione

4.3.2 Skewness (Asimmetria)

Questo indice misura quanto la geometria effettiva di una cella si discosti dalla sua forma ideale (ad esempio, per una cella esaedrica, la forma ideale sarebbe un cubo). Più alta è la skewness, maggiore sarà la deviazione dalla forma ideale. Nella simulazione di fluidodinamica computazionale (CFD), l'asimmetria delle celle rappresenta una sfida significativa, poiché le equazioni numeriche si basano sull'assunzione di celle che approssimano la geometria ideale il più possibile.

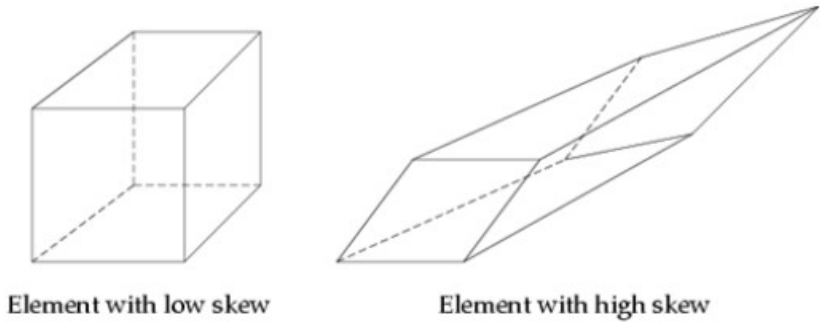


Figura 7 Immagine rappresentativa del significato geometrico di *skeweness* per ciascuna cella

La skewness di una cella viene calcolata mediante il metodo della deviazione angolare normalizzata e può essere definita come:

$$\text{skewness} = \max\left[\frac{\theta_{max}-\theta_e}{180-\ _e};\frac{\theta_e-\theta_{min}}{\theta_e}\right]$$

dove θ_{max} si riferisce all'angolo maggiore della cella in oggetto, θ_{min} a quello minore, θ_e è l'angolo della relativa geometria ideale di riferimento.

La scala di riferimento per monitorare la qualità della mesh attraverso questo indice è la seguente:

- Compreso tra 0.9 e 1.0 – Pessima
- Compreso tra 0.75 e 0.9 – Scarsa
- Compreso tra 0.5 e 0.75 – Accettabile
- Compreso tra 0.25 e 0.5 – Buona
- Compreso tra 0 e 0.25 – Eccellente

È auspicabile che tutte le celle nella mesh soddisfino almeno uno standard accettabile, con una skewness inferiore a 0.5. Tuttavia, è ammissibile utilizzare mesh che includano un numero limitato di celle con una qualità inferiore, specialmente se queste si trovano in aree meno rilevanti dal punto di vista fisico. Una mesh tridimensionale può essere considerata di alta qualità quando tutte le celle hanno una skewness inferiore a 0.4 e la maggior parte di esse raggiunge uno standard eccellente, con un valore inferiore a 0.25.

Per il caso in esame si osserva una skewness media eccellente, con una massima accettabile. La differenza tra i due parametri suggerisce che il numero di celle con skewness eccellente sia notevolmente superiore.

4.3.3 Aspect ratio (AR)

Rappresenta il rapporto tra la dimensione massima e la dimensione minima di una cella. A differenza della non-ortogonalità e della skewness, tuttavia, non esistono valori di riferimento definiti per stabilire un AR ottimale nella creazione della mesh per una simulazione fluidodinamica.

Sebbene valori ridotti di AR favoriscano la stabilità della soluzione, prevenendo disomogeneità nel flusso del fluido, è possibile utilizzare mesh con AR molto elevati (anche oltre 1000) quando la dimensione della mesh è allineata con la direzione del flusso. In generale, un Aspect Ratio inferiore a 20 non dovrebbe causare problemi di convergenza della soluzione, indipendentemente dalle caratteristiche del flusso di fluido, e nel caso riportato viene ampiamente rispettata questa condizione.

4.4 Creazione della Mesh

L'esecuzione del meshatore è stata pilotata tramite shell scripting contenuto in un file run.sh scritto per eseguire in cascata le varie utility di cfMesh. Viene riportata la catena di comandi necessari all'esecuzione

```
#!/bin/bash

cd ~/foamMesh/casePath/case
surfaceFeatureEdges Tank.stl Tank.fms -angle 15
cartesianMesh
rotateMesh "(1 0 0)" "(0 0 1)"
transformPoints -translate "(0.15117 -0.17145 0)"
#transformPoints  -rollPitchYaw '(0 0 30)'
checkMesh
```

riassumendo brevemente le operazioni in sequenza

- la stringa *#!/bin/bash* prepara l'environment (in questo caso la Windows Subsystem for Linux) ad unsare Bash come command interpreter.
- La riga di codice `cd ~/foamMesh/casePath/case` cambia la directory a quella del caso
- La riga di codice `surfaceFeatureEdges Tank.stl Tank.fms -angle 15` esegue l'utility surfaceFeatureEdges

prendendo in input il file Tank.stl e generando in output la geometria Tank.fms parametrizzata con l'angolo indicato come attributo. Il file Tank.stl deve già essere presente nel path indicato alla riga precedente.

- `cartesianMesh` è il comando vero e proprio di esecuzione del meshatore
- `rotateMesh "(1 0 0)" "(0 0 1)"` è il comando che esegue l'utility rotateMesh per ruotare la mesh in modo tale che la terna di assi "(1 0 0)" vada a coincidere con la terna "(0 0 1)". Questa operazione, come detto, serve perché l'utility surfaceFeatureEdges ha una direzione preferenziale che coincide con l'asse x, mentre la geometria finale si desidera abbia il suo sviluppo verticale lungo z.
- la riga di codice `transformPoints -translate "(0.15117 -0.17145 0)"` sposta i punti della mesh con la parametrizzazione indicata. Questo serve a postare l'origine degli assi in un punto pre scelto (coincidente con il centro geometrico del serbatoio).
- La riga `transformPoints -rollPitchYaw '(0 0 30)'`, quando priva dell'asterisco davanti, esegue l'utility trasformPoints ruotando la mesh con angolo di imbardata 30 gradi. Questa è stata utilizzata per valutare diverse condizioni di orientamento di volo, in particolare perpendicolare allo spigolo del serbatoio e perpendicolare alla faccia piana del serbatoio.
- `checkMesh` esegue il comando omonimo per l'utility che restituisce in output a terminale i parametri per la valutazione della qualità della mesh.

L'output della mesh è possibile monitorarlo generando tramite comando `touch output.foam` il foamFile ed aprendolo da visualizzatore Paraview. I file contenenti la mesh sono salvati nella subdirectory *polyMesh* contenuta in *constant* (e generata dal meshatore) del path del caso.

4.5 Setup della simulazione

La fase di setup della simulazione è cruciale per garantire la corretta esecuzione e l'affidabilità dei risultati ottenuti attraverso la simulazione CFD (Computational Fluid Dynamics). Durante questa fase, vengono definiti tutti i parametri e le condizioni necessarie per risolvere il problema fluidodinamico in esame. Innanzitutto, si seleziona il modello di turbolenza più appropriato in base alle caratteristiche del flusso e agli obiettivi della simulazione. Un secondo aspetto fondamentale di questa fase include la definizione accurata delle condizioni al contorno, che devono riflettere fedelmente le condizioni reali del problema, e la scelta dei parametri numerici, come la discretizzazione spaziale e temporale, che influenzano la stabilità e la convergenza della simulazione..

Come già evidenziato della sezione *Struttura di un caso,* OpenFOAM nella configurazione di setup della simulazione richiede la presenza di una directory, all'interno del Path della simulazione, chiamata *constant.*

4.5.2 Directory `constant`

In questa directoy è stata importata la cartella *polyMesh* generata in output dal meshatore, contenuta a sua volta nella cartella *constant* del caso di meshing. Oltre ad essa si trovano diversi file contenenti le costanti fisiche del caso:

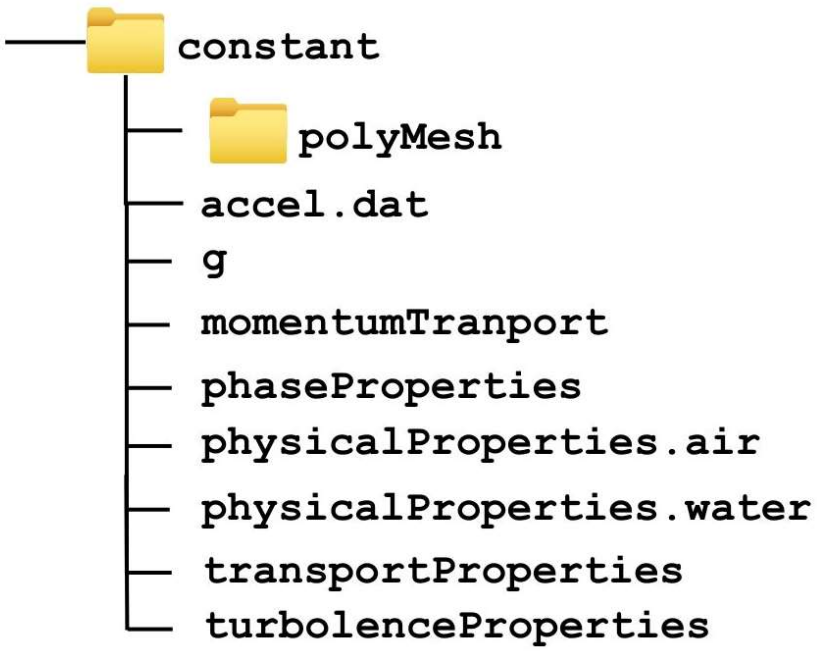


Figura 8 Struttura della directory *constant* del path di simulazione

File `accel.dat`

Questo file contiene il profilo di accelerazioni in input della simulazione, che si traduce in un movimento inerziale della massa fluida. I dati importati sono quelli della piattaforma IMU del drone in risposta ad una simulazione di volo caratterizzata dal seguente profilo di missione:

- partenza da condizione di hovering
- accelerazione fino al raggiungimento di una velocità di volo di 2 m/s lungo l'asse x
- Mantenimento della velocità per 20 s sul filare
- Decelerazione fino alla condizione di hovering

Per l'estrazione dei dati di usa il topic ROS VehicleLocalPosition (uORB message) di cui si riporta il codice sorgente

```
# Fused local position in NED.
   # The coordinate system origin is the vehicle position at the time when
   the EKF2-module was started.

   uint64 timestamp                # time since system start (microseconds)
   uint64 timestamp_sample                 # the timestamp of the raw data
   (microseconds)

   bool xy_valid                   # true if x and y are valid
   bool z_valid                    # true if z is valid
   bool v_xy_valid                       # true if vx and vy are valid
   bool v_z_valid                        # true if vz is valid

   # Position in local NED frame
   float32 x                       # North position in NED earth-fixed
   frame, (metres)
   float32 y                       # East position in NED earth-fixed
   frame, (metres)
   float32 z                       # Down position (negative altitude) in
   NED earth-fixed frame, (metres)

   # Position reset delta
   float32[2] delta_xy             # Amount of lateral shift of position
   estimate in latest reset (in x and y) [m]
   uint8 xy_reset_counter                  # Index of latest lateral position
   estimate reset
```

```
float32 delta_z                         # Amount of vertical shift of
position estimate in latest reset [m]
uint8 z_reset_counter                   # Index of latest vertical
position estimate reset

# Velocity in NED frame
float32 vx                      # North velocity in NED earth-fixed
frame, (metres/sec)
float32 vy                      # East velocity in NED earth-fixed
frame, (metres/sec)
float32 vz                      # Down velocity in NED earth-fixed
frame, (metres/sec)
float32 z_deriv                         # Down position time derivative in
NED earth-fixed frame, (metres/sec)

# Velocity reset delta
float32[2] delta_vxy                    # Amount of lateral shift of
velocity estimate in latest reset (in x and y) [m/s]
uint8 vxy_reset_counter                 # Index of latest vertical
velocity estimate reset
float32 delta_vz                # Amount of vertical shift of velocity
estimate in latest reset [m/s]
uint8 vz_reset_counter                  # Index of latest vertical
velocity estimate reset

# Acceleration in NED frame
float32 ax        # North velocity derivative in NED earth-fixed frame,
(metres/sec^2)
float32 ay        # East velocity derivative in NED earth-fixed frame,
(metres/sec^2)
float32 az        # Down velocity derivative in NED earth-fixed frame,
(metres/sec^2)

float32 heading                         # Euler yaw angle transforming the
tangent plane relative to NED earth-fixed frame, -PI..+PI,  (radians)
float32 heading_var
float32 unaided_heading                  # Same as heading but generated
by integrating corrected gyro data only
float32 delta_heading                   # Heading delta caused by latest
heading reset [rad]
uint8 heading_reset_counter             # Index of latest heading reset
bool heading_good_for_control

float32 tilt_var

# Position of reference point (local NED frame origin) in global (GPS /
WGS84) frame
bool xy_global                          # true if position (x, y) has a
valid global reference (ref_lat, ref_lon)
bool z_global                   # true if z has a valid global reference
(ref_alt)
uint64 ref_timestamp                    # Time when reference position was
set since system start, (microseconds)
float64 ref_lat                         # Reference point latitude,
(degrees)
float64 ref_lon                         # Reference point longitude,
(degrees)
float32 ref_alt                         # Reference altitude AMSL,
(metres)

# Distance to surface
float32 dist_bottom             # Distance from from bottom surface to
ground, (metres)
```

```
bool dist_bottom_valid                  # true if distance to bottom
surface is valid
uint8 dist_bottom_sensor_bitfield       # bitfield indicating what type of
sensor is used to estimate dist_bottom
uint8 DIST_BOTTOM_SENSOR_NONE = 0
uint8 DIST_BOTTOM_SENSOR_RANGE = 1      # (1 << 0) a range sensor is used
to estimate dist_bottom field
uint8 DIST_BOTTOM_SENSOR_FLOW = 2       # (1 << 1) a flow sensor is used
to estimate dist_bottom field (mostly fixed-wing use case)

float32 eph                         # Standard deviation of horizontal
position error, (metres)
float32 epv                         # Standard deviation of vertical
position error, (metres)
float32 evh                         # Standard deviation of horizontal
velocity error, (metres/sec)
float32 evv                         # Standard deviation of vertical
velocity error, (metres/sec)

bool dead_reckoning                      # True if this position is
estimated through dead-reckoning

# estimator specified vehicle limits
float32 vxy_max                         # maximum horizontal speed - set
to 0 when limiting not required (meters/sec)
float32 vz_max                          # maximum vertical speed - set to
0 when limiting not required (meters/sec)
float32 hagl_min                    # minimum height above ground level -
set to 0 when limiting not required (meters)
float32 hagl_max                    # maximum height above ground level -
set to 0 when limiting not required (meters)

# TOPICS vehicle_local_position vehicle_local_position_groundtruth
external_ins_local_position
# TOPICS estimator_local_position
```

Dal file csv in output al ROS topic si è quindi estratto i valori di interesse (velocità e accelerazioni) e si crea il file di profilo di accelerazioni *accel.dat* da importare nella directory del path di simulazione

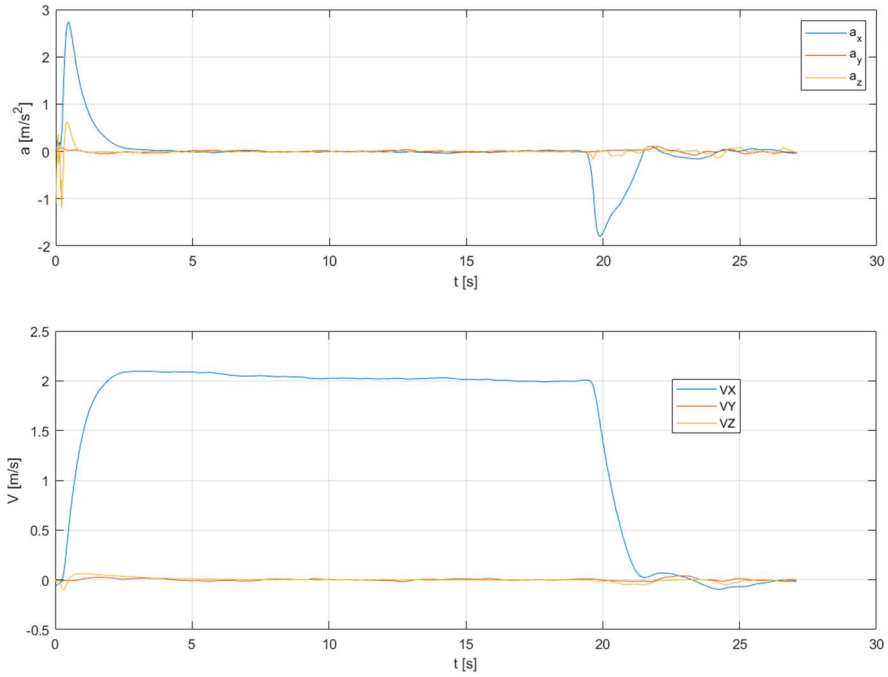


Figura 9 Profilo di accelerazioni della missione del drone (in alto) e profilo di velocità (in basso)

Si riporta lo script matlab utilizzato per la lettura del log file prodotto dal ros topic e scrittura del file accel.dat

```
clc
clear
T = readtable('test1.csv');
%% velocità
j=1;
for i=1:8318
    if isnan(T.x_fmu_out_vehicle_local_position_vx(i))
    else
        if (T.x__time(i)-T.x__time(1)>16.3916)
            t(j)=T.x__time(i)-T.x__time(1)-16.3916;
            vx(j)=T.x_fmu_out_vehicle_local_position_vx(i);
            vy(j)=T.x_fmu_out_vehicle_local_position_vy(i);
            vz(j)=T.x_fmu_out_vehicle_local_position_vz(i);
            j=j+1;
        end
    end
end
figure()
plot(t,vx);
hold on
```

```
plot(t,vy);
hold on
plot(t,vz);
grid on
legend("VX","VY","VZ");
ylabel('V [m/s]');
xlabel('t [s]');
%%
j=1;
fid = fopen('accel.dat','wt');
fprintf(fid,'(\n');
for i=1:8319
    if isnan(T.x_fmu_out_vehicle_local_position_ax(i))
    else
        if (T.x__time(i)-T.x__time(1)>16.3916)
            fprintf(fid,'\t(%.3f\t((%.4f %.4lf %.4lf)\t(0 0 0)\t(0 0
0)))\n',T.x__time(i)-T.x__time(1)-
16.3916,T.x_fmu_out_vehicle_local_position_ax(i),T.x_fmu_out_vehicle_local_
position_ay(i),T.x_fmu_out_vehicle_local_position_az(i));
            t(j)=T.x__time(i)-T.x__time(1)-16.3916;
            ax(j)=T.x_fmu_out_vehicle_local_position_ax(i);
            ay(j)=T.x_fmu_out_vehicle_local_position_ay(i);
            az(j)=T.x_fmu_out_vehicle_local_position_az(i);
            j=j+1;
        end
    end
end
fprintf(fid,');');
figure()
plot(t, ax);
hold on
plot(t,ay);
hold on
plot(t,az);
legend("a_x","a_y","a_z");
ylabel('a [m/s^2]');
xlabel('t [s]');
grid on
```

il file di accelerazioni letto da OpenFOAM ha quindi il formato che segue di esempio:

```
(
       //time       ax         ay    az   omega        omegadt
       (0.000 ((-0.0065 0.0627 1.1154)   (0 0 0)      (0 0 0)))
       (0.008 ((0.0360 0.0604 0.9191)    (0 0 0)      (0 0 0)))
       (0.020 ((0.0940 0.0574 0.5953)    (0 0 0)      (0 0 0)))
       (0.030 ((0.1507 0.0552 0.3967)    (0 0 0)      (0 0 0)))
       (0.041 ((0.1953 0.0532 0.2070)    (0 0 0)      (0 0 0)))
       (0.049 ((0.2239 0.0515 0.0266)    (0 0 0)      (0 0 0)))
       (0.060 ((0.2402 0.0494 -0.2192)   (0 0 0)      (0 0 0)))
       (0.071 ((0.2328 0.0491 -0.2774)   (0 0 0)      (0 0 0)))
       * * *
       * * *
       (27.080      ((-0.0395 -0.0182 0.0379)  (0 0 0)      (0 0 0)))
       (27.090      ((-0.0395 -0.0177 0.0416)  (0 0 0)      (0 0 0)))
);
```

Tabella 1 Prototipo del file accel.dat contenuto in $SIM_DIR/constant

Come si può osservare, sebbene OpenFOAM prevede la possibilità di importare il profilo di velocità e accelerazioni angolari, che per altro dal topic ROS vengono estratte, per agevolare la comprensione del fenomeno si è scelto di trascurarle, in quanto il contributo per piccole rotazioni quali delle del profilo di missione non sarebbe stato rilevante. Occorre tenere presente che per la corretta valutazione di questi effetti sarebbe necessario anche conoscere precisamente la coordinata del centro di gravità del drone completo relativa al centro di massa del serbatoio, che cambia al variare del livello di riempimento.

File `g`

Questo dizionario contiene le informazioni relative alla costante di accelerazione gravitazionale.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);


// ************************************************************************* //
```

Tabella 2 Prototipo del file g contenuto in $SIM_DIR/constant

File `phaseProperties`

Questo dizionario specifica le fasi come in tabella. Vengono listate due fasi, acqua e aria. Le equazioni per la frazione di fase vengono risolte per le fasi nell'elenco, ad eccezione dell'ultima fase elencata, cioè l'aria in questo caso. Poiché ci sono solo due fasi, in questo caso viene risolta solo un'equazione per la frazione di fase, per l'equazione della frazione di fase dell'acqua, specificata nel file alpha.water nella directory 0.

Il file phaseProperties contiene anche un'voce per la tensione superficiale tra le due fasi, specificata dalla keywords sigma [N/m]

Questo parametro rappresenta una costante fisica ben descritta in letteratura

Viene mostrato il prototipo del file cosìcome appare nell'editor

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      phaseProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases           ( water air );

sigma            0.072;


// ************************************************************************* //
```

Tabella 3 Prototipo del file phaseProperties contenuto in $SIM_DIR/constant

File `momentumTransport`

Questo FOAM File contiene i parametri del modello di turbolenza adottato nel trasporto della quantità di moto.

Tramite keywords simulationType viene impostato il modello di turbolenza RAS, parametrizzato nell'omonimo sub-dictionary

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      momentumTransport;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType      RAS;

density             variable;

RAS
{
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Tabella 4 Prototipo del file momentumTransport contenuto in $SIM_DIR/constant

Il modello k (k-epsilon) è stato il pioniere tra i modelli di turbolenza adottati diffusamente per una vasta gamma di flussi nella CFD. Appartiene a una famiglia di modelli a due equazioni che risolvono due equazioni di trasporto, di solito una per l'energia cinetica turbolenta (k) e un'altra per una variabile aggiuntiva, spesso associata al tasso di dissipazione. Questi modelli sono diventati lo standard dell'industria per la CFD.

Le equazioni di trasporto per k ed ε risultano essere

$$\frac{\partial k}{\partial t} + \nabla \cdot (uk) - \nabla \cdot (D_k \nabla k) = G - \frac{2}{3} k(\nabla \cdot u) - \varepsilon$$

$$\frac{\partial \varepsilon}{\partial t} + \nabla \cdot (u\varepsilon) - \nabla \cdot (D_\varepsilon \nabla \varepsilon) = \frac{c_1 G\varepsilon}{k} - \frac{2}{3} c_1 \varepsilon(\nabla \cdot u) - \frac{c_2 \varepsilon^2}{k}$$

$$D_k = \nu + \frac{\nu_t}{\sigma_k} \qquad D_\varepsilon = \nu + \frac{\nu_t}{\sigma_\varepsilon}$$

Con i coefficienti standard, stampati ad inizio simulazione in virtù del settaggio `printCoeffs on;`

```
RAS
{
    RASModel        kEpsilon;
    turbulence      on;
    printCoeffs     on;
    Cmu             0.09;
    C1              1.44;
    C2              1.92;
    C3              0;
    sigmak          1;
    sigmaEps        1.3;
}
```

Il modello usa la formulazione dell'energia cinetica turbolenta

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot \rho uu = 2\nabla \cdot \rho \nu_{eff} devD - \nabla p + \rho b$$

Con la viscosità cinematica effettiva è calcolata come $\nu_{eff} = \nu + \nu_t$

La viscosità effettiva rappresenta la diffusione del momento dalle movimentazioni molecolari e turbolente combinate. Le proprietà dovute al moto molecolare sono spesso descritte come laminari, ad esempio l'equazione della viscosità laminare. Mediando l'equazione di momento e introducendo nel modello la viscosità turbolenta ν_t si crea un'ulteriore incognita. Sono necessari modelli aggiuntivi per chiudere il sistema di equazioni.

Considerando $\nu_t = u_m l_m$ il modello si compone di due elementi che rappresentano velocità del flusso e lunghezza di scala di riferimento.

La scala di $u_m$corrisponde alle fluttuazioni turbolente, quindi è ragionevole assumere che sia proporzionale a $k^{\frac{1}{2}}$.

La figura qui sotto mostra gli aggiornamenti all'algoritmo transitorio, a partire dal passaggio correttore del momento.

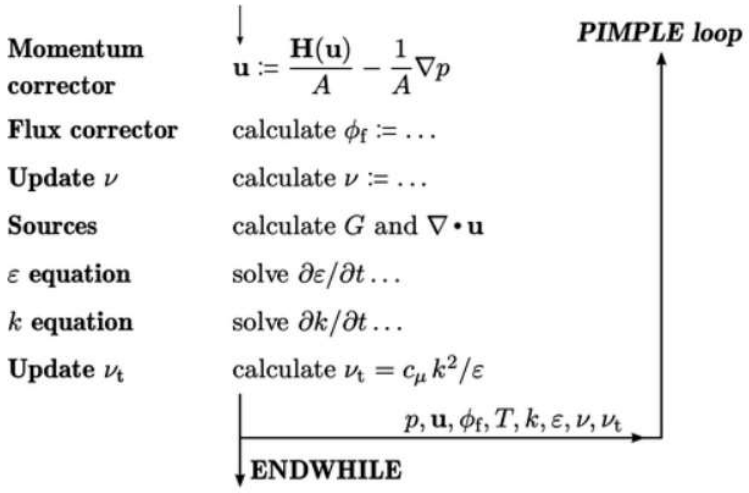


Figura 10 Processo implementato dal loop PIMPLE

File **physicalProperties.air**

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      physicalProperties.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

viscosityModel  Newtonian;

nu              1.48e-05;

rho             1;


// ************************************************************************* //
```

Tabella 5 Prototipo del file physicalProperties.air contenuto in $SIM_DIR/constant

File **physicalProperties.water**

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      physicalProperties.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

viscosityModel  Newtonian;

nu              1e-06;

rho             1000;


// ************************************************************************* //
```

Tabella 6 Prototipo del file physicalProperties.water contenuto in $SIM_DIR/constant

Questi dizionari, analoghi nella forma, contengono i parametri fisici delle due fasi, come densità e viscosità dinamica. Il modello di viscoità viene impostato newtoniano.

File `transportProperties`

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases           (water air);

water
{
    transportModel  Newtonian;
    nu              1e-06;
    rho             1000;
}

air
{
    transportModel  Newtonian;
    nu              1.48e-05;
    rho             1;
}

sigma            0.072;

// ************************************************************************* //
```

Tabella 7 Prototipo del file transportProperties contenuto in $SIM_DIR/constant

4.5.3 Directory `0`

In questa seconda directory principale del progetto si definiscono le condizioni iniziali e al contorno, ciascuna atraverso uno specifico dizionario, dei parametri fisici.

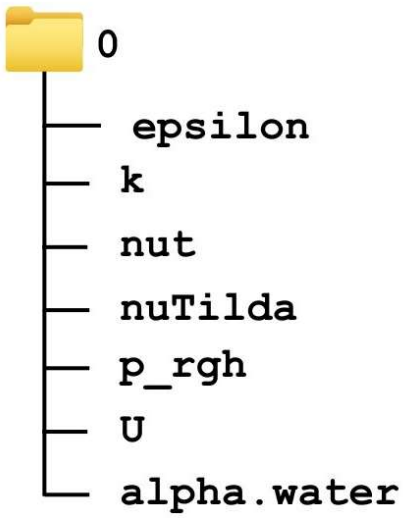


Figura 11 Struttura della directory *0* del path di simulazione

Di seguito viene riportato per ciascun file una descrizione e un prospetto del contenuto

File **epsilon**

Contiene le condizioni al contorno per il parametro epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2306                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 0.01;

boundaryField
{
    emptyRot
    {
        type            epsilonWallFunction;
        value           uniform 0.5;
    }

}
```

Tabella 8 Prototipo del file epsilon contenuto in $SIM_DIR/0

La condizione di quiete iniziale permette di settare un valore interno per la dissipazione cinetica minore. Per le condizioni a parete viene impostato inizialmente un valore pari a 0.1, preso come riferimento per una configurazione cilindrica di dimensinoi rapportabili. In seguito, tramite studio delle caratteristiche geometriche e analisi di sensibilità si è fissato questo valore pari a 0.5.

File k

Come per il caso precedente, la condizione interna all'istante iniziale di quiete risulta con parametro ridotto. Per le condizioni a parete, partendo da una geometria cilindrica e adattando il caso, tenuto opportunamente conto del rapporto reciproco con $\boldsymbol{\epsilon}$ dovuto alla viscosità turbolenta $\boldsymbol{\nu_t}$, si è adottato un valore pari a 1.35.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2306                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.01;

boundaryField
{
    emptyRot
    {
        type            kqRWallFunction;
        value           uniform 1.35;
    }

}
```

Tabella 9 Prototipo del file k contenuto in $SIM_DIR/0

File `nut`

questo parametro è legato ai due precedenti ed un coefficiente di viscosità $C_\nu \approx 0.09$

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2306                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 2e-3;

boundaryField
{
    emptyRot
    {
        type            nutkWallFunction;
        value           uniform 0.2;
    }

}
```

Tabella 10 Prototipo del file nut contenuto in $SIM_DIR/0

File `p_rgh`

Rappresenta la pressione relativa alla pressione idrostatica, che è particolarmente utile nei calcoli in cui la componente idrostatica della pressione è significativa, come nei flussi multifase. Per il caso di flusso statico iniziali si impostano le condizioni al contorno seguenti.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2306                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    emptyRot
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
        value           uniform 0;
    }

}
```

Tabella 11 Prototipo del file p_rgh contenuto in $SIM_DIR/0

File U

Rappresenta la velocità del flusso, pari a zero nelle condizioni inziali. A parete viene impostata una condizione no slip.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2306                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    emptyRot
    {
        type            noSlip;
    }

}
```

Tabella 12 Prototipo del file U contenuto in $SIM_DIR/0

4.5.4 Directory `system`

In questa cartella sono contenuti i file che definiscono i modellli numerici, schemi di approssimazione, metodi di disretizzazione oltre ad alcuni dizionari per la divisione del dominio di calcolo in parallelo e per l'assegnazione iniziale di parametri di campo del dominio fluido

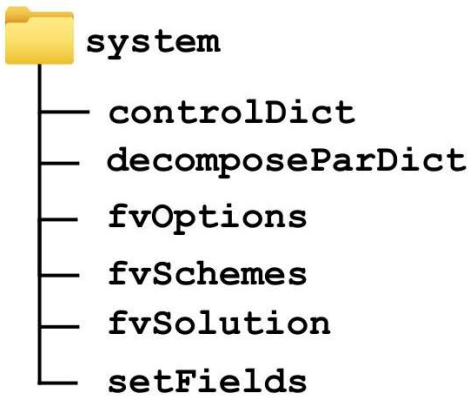


Figura 12 Struttura della directory *system* del path di simulazione

File `controlDict`

I solutori di OpenFOAM avviano tutte le esecuzioni stabilendo un database. Questo database controlla le operazioni di input/output (I/O) e, poiché tipicamente durante l'esecuzione è richiesto l'output dei dati a intervalli temporali specifici, il tempo è una componente essenziale del database. Il dizionario controlDict imposta i parametri di input fondamentali per la creazione del database. Le voci chiave in controlDict sono elencate nelle sezioni seguenti. Solo le voci obbligatorie per il controllo del tempo e l'intervallo di scrittura (writeInterval) sono necessarie, mentre il database utilizza valori predefiniti per le voci opzionali che vengono omesse

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         28;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  on;

maxCo           1;

maxAlphaCo      1;

maxDeltaT       1;

functions
{
forces
{
type forces;
functionObjectLibs ("libforces.so"); //Lib to load
patches (emptyRot);
rhoName rhoInf;
rhoInf 998.2; //Reference density for fluid
CofR (0 0 00); //Origin for moment calculations
outputControl outputTime;
}
}
// ************************************************************************* //
```

Tabella 13 Prototipo del file controlDict contenuto in $SIM_DIR/system

La keywords `application` imposta il solutore desiderato. Seguono i controlli relativi all'istante iniziale da cui parte la simulazione e quello finale, impostato pari a 28s.

`writeControl` controlla il tempismo della scrittura dell'output su file.

`timeStep:` Scrive i dati ogni passo temporale di writeInterval.

`runTime:` Scrive i dati ogni writeInterval secondi di tempo simulato.

`adjustableRunTime:` Scrive i dati ogni writeInterval secondi di tempo simulato, regolando i passi temporali per coincidere con writeInterval se necessario. Viene utilizzato nei casi con adeguamento automatico del passo temporale.

`writeInterval:` Scala utilizzata in congiunzione con writeControl descritto sopra.

`purgeWrite:` Intero che rappresenta un limite sul numero di directory temporali che vengono memorizzate sovrascrivendo le directory temporali su base ciclica. Ad esempio, se le simulazioni iniziano a $t = 5s$ e $\Delta t = 1s$, quindi con purgeWrite 2 i dati vengono prima scritti in due directory, 6 e 7, quindi quando 8 viene scritto, 6 viene eliminato, e così via, in modo che esistano solo 2 nuove directory dei risultati in qualsiasi momento. Il valore 0 disabilita questa feature

`writeFormat:` Specifica il formato dei file di dati.

- ascii (predefinito): formato ASCII, scritto con writePrecision cifre significative.
- binary: formato binario.

`writePrecision:` Intro utilizzato in congiunzione con writeFormat descritto sopra, 6 per impostazione predefinita.

`writeCompression:` Interruttore per specificare se i file sono compressi con gzip quando scritti: on/off (sì/no, vero/falso).

`timeFormat:` Scelta del formato del nome delle directory temporali. L'attributo General specifica il formato scientifico se l'esponente è inferiore a -4 o maggiore o uguale a quello specificato da timePrecision.

`timePrecision:` Intero utilizzato in congiunzione con timeFormat

`adjustableTimeStep:` wsitch condizionale, quando in on permette alla simulazione di valutare il delta temporale in accordo con il numero di Courant, operando in parallelo a `maxCo`

`maxCo:` Massimo numero di Courant. Il numero di Courant è un parametro adimensionale utilizzato nella simulazione numerica dei fenomeni di trasporto ed è definito come il rapporto tra la velocità del flusso locale e la velocità di propagazione delle informazioni nel dominio. In termini matematici, è espresso come:

$$Co = u \cdot \frac{\Delta t}{\Delta x}$$

Dove:

- u è la velocità del flusso locale,
- Δt è il passo temporale,
- Δx è la dimensione caratteristica del reticolo o del dominio.

Il massimo numero di Courant viene scelto come compromesso tra stabilità numerica e affidabilità della simulazione. Nel caso in esame viene scelto confrontando in letteratura con fenomeni simili.

Il dizionario functions definisce la funzione come sottodizionario forces che calcola forze e momenti distribuiti lungo i tre assi, restituendo in

output un file *forces.dat* nel path si simulazione *postProcessing/forces/0* con il seguent formato dati:

```
# Time        forces(pressure viscous)  moments(pressure viscous)
```

Ad esempio:

```
0.1               ((-8.178407e-01 -2.068378e-02 -6.995510e+01) (-
                  3.797156e-03 3.320313e-04 1.366488e-03)) ((8.191373e-
                  01 -1.498436e+00 -8.782219e-03) (-2.930234e-05
                  2.531800e-04 -6.728189e-05))
```

La prima parentesi riporta le forze di pressione per x, y e z, la seconda le forze viscose ed allo stesso modo per i momenti, calcolati rispetto al centro di massa specificato come attributo nel sotto-dizionario.

Per lo studio di questa tesi viene considerato di interesse il solo apporto delle forze; per altro il calcolo del momento richiederebbe la conoscenza del centro di massa del sistema completo drone-serbatoio, che varia in funzion del riempimento del serbatoio, a sua volta decrescente durante la missione di irrorazione.

File `decomposeParDict`

Questo dizionario divide geometria e campi in sottodomini per l'esecuzione in parallelo della simulazione, sfruttando architetture multicore o cluster multiprocessore.

Per il caso in esame la simulazione viene eseguita su una macchina così assemblata:

- Intel i5-7200U dual core
- 16 Gb RAM DDR4

Per tale configurazione si è osservato che i vantaggi, in termini di clocktime di simulazione, forniti dall'esecuzione in parallelo, vengono

assorbiti dalla differenza prestazionale tra single core e multicore del processore, che dispone in architettura di soli due core fisici. Si è ritenuto pertanto controproducente l'utilizzo dell'utility *decomposePar* pilotata dall'omonimo dizionario (che necessiterebbe poi in fase di post processing dell'esecuzione dell'utility *recontructPar* portando ad un maggiore dispendio di tempo.

Si riporta tuttavia il prototipo del file dati del dizionario sul quale sono state eseguiti questi test, con la possibilità, tramite opportuna modifica delle assegnazioni ale varie keyword di dizionario, di replicare la simulazione su un'architettura di tipo cluster-multiprocessore.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 2;

method          scotch;

// ************************************************************************* //
```

Tabella 14 Prototipo del file decomposeParDict contenuto in $SIM_DIR/system

Il numero di sottodomini è funzione del numero di processori sul quale si ripartisce la soluzione. Il metodo scotch di decomposizione non richiede input geometrico dall'utente e cerca di minimizzare il numero di confini del processore.

File `fvOptions`

Questo dizionario permette di aggiungere codice sorgente eseguito in runtime senza necessità di riscrivere i file sorgente originali della modellazione fluidodinamica. In questo caso è stato utilizzato per importare la sorgente di accelerazione dal path indicato dall'attributo `file` specificando attraverso le keywords `type accelerations` la formattazione del file importato (accel.dat)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

accelerations
{
     type             sixDoFAccelerationSource;
    selectionMode   all;
    fields           (U);
    accelerations   table;

    file "$FOAM_CASE/constant/accel.dat";
}


// ************************************************************************* //
```

Tabella 15 Prototipo del file fvOption contenuto in $SIM_DIR/system

File `fvSchemes`

Il dizionario fvSchemes definisce gli schemi numerici per i termini, come le derivate delle equazioni, che compaiono nelle applicazioni, i metodi di interpolazione tra punti e i modelli di risoluzione numerica per la discretizzazione temporale.

Gli schemi temporali definiscono come una proprietà viene integrata in funzione del tempo, cioè

$$\frac{\partial}{\partial t}(\phi)$$

In base alla scelta dello schema, sono richiesti i valori del campo ai passaggi temporali precedenti, rappresentati nei seguenti come ϕ_o e ϕ_{oo} per i livelli di tempo precedenti.

Per il modello in esame viene scelto il metodo di Eulero del primo ordine che implementa l'equazione

$$\frac{\partial}{\partial t}(\phi) = \frac{\phi - \phi^0}{\Delta t}$$

Il sotto-dizionario `gradSchemes` contiene i termini di gradiente di cella.

l gradiente di un campo scalare ϕ in un punto può essere approssimato utilizzando la formula delle differenze finite:

$$\nabla\phi \approx \frac{(\phi_E - \phi_W)}{\Delta x}$$

dove:

ϕ_E è il valore del campo alla faccia orientata verso est,

ϕ_W è il valore del campo alla faccia orientata verso ovest,

Δx è la distanza tra le facce.

All'interno del sotto-dizionario `gradSchemes`, vengono specificati gli schemi numerici utilizzati per approssimare il gradiente del campo. Lo schema `Gauss linear` adoperato per la modellizzazione del caso in esame utilizza un'interpolazione lineare tra i valori del campo sulle facce adiacenti per calcolare il gradiente.

Il sotto-dizionario `divSchemes` definisce gli schemi numerici utilizzati per la discretizzazione dei termini di divergenza nelle equazioni governanti. Questi schemi numerici sono responsabili della valutazione della divergenza dei flussi attraverso le interfacce delle celle della mesh. All'interno del sotto-dizionario divSchemes, è possibile specificare diversi schemi numerici per gestire la divergenza dei flussi in base alle esigenze specifiche della simulazione.

La divergenza di un campo vettoriale F in un punto può essere approssimata utilizzando la formula:

$$div(F) \approx \left(\frac{1}{V_P}\right) \sum_i^{N_f} (F \cdot n_i \cdot A_i)$$

dove:

- V_P è il volume del control volume centrato in PP,
- N_f è il numero di facce che delimitano il control volume,
- n_i è il vettore normale alla ii-esima faccia,
- A_i è l'area della ii-esima faccia.

Alcuni degli schemi numerici comuni disponibili in `divSchemes` vengono riportati di seguito:

- Schemi lineari: Questi schemi utilizzano una combinazione lineare dei valori dei flussi alle interfacce delle celle per calcolare la divergenza. Sono spesso utilizzati per la loro semplicità e stabilità. La funzione di distribuzione gaussiana assegna pesi

maggiori ai valori del flusso che si trovano più vicini al centro del control volume e pesi minori a quelli che sono più lontani. Questo assicura che i valori del flusso più vicini al centro del control volume abbiano una maggiore influenza sull'approssimazione della divergenza rispetto a quelli che si trovano più lontani.

- Schemi upwind: Questi schemi assegnano un peso maggiore ai flussi che provengono dalla direzione dominante del flusso. Sono particolarmente utili per la gestione di flussi con discontinuità o elevati gradienti.

Il sotto-dizionario `laplacianSchemes` definisce gli schemi numerici utilizzati per la discretizzazione del termine laplaciano nelle equazioni governanti. Il termine laplaciano è comunemente utilizzato per modellare fenomeni di diffusione o dispersione in diverse applicazioni fisiche e ingegneristiche. La formula generale per uno schema upwind del Laplaciano può essere espressa come:

$$\nabla^2\phi \approx \frac{2\phi_P - \phi_E - \phi_W}{\Delta x^2}$$

dove ϕ_P è il valore del campo scalare al centro della cella, ϕ_E e ϕ_W sono i valori del campo scalare nelle celle adiacenti a est e ovest rispettivamente, e Δx è la lunghezza del lato della cella.

Il sotto-dizionario `interpolationSchemes` definisce gli schemi numerici utilizzati per l'interpolazione dei valori da un insieme di punti a un altro. Questo è particolarmente importante quando si calcolano i valori dei campi su facce delle celle o punti all'interno delle celle basati sui valori nei punti della mesh. Il metodo lineare implementa lo schema del tipo

$$\phi = (1 - \alpha)\Phi_1 + \alpha\phi_2$$

Dove ϕ_1 e ϕ_2 sono i valori nei due punti di origine e α è un parametro che varia tra 0 e 1, rappresentante la distanza relativa tra i due punti.

All'interno di `snGradSchemes`, vengono definiti gli schemi numerici per approssimare il gradiente normale del campo scalare su facce non-ortogonali. Questi schemi sono essenziali per garantire una corretta discretizzazione delle equazioni nei casi in cui la geometria non consente un allineamento perfetto tra i vettori normali alle facce e i gradienti dei campi scalari.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)   Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss upwind;
    div(phirb,alpha) Gauss linear;
    div(rhoPhi,k)       Gauss upwind;
    div(rhoPhi,epsilon) Gauss upwind;
    div(phi,epsilon) Gauss linear;
    div(phi,k) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear orthogonal;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
 default         orthogonal
;
}


// ************************************************************************* //
```

Tabella 16 Prototipo del file fvSchemes contenuto in $SIM_DIR/system

File `fvSolution`

Il file *fvSolution* è un dizionario che definisce i parametri relativi alla soluzione numerica dei sistemi di equazioni differenziali alle derivate parziali (PDE) risolti durante la simulazione. Questo file contiene le impostazioni relative ai metodi numerici utilizzati per approssimare le equazioni del modello matematico, nonché le impostazioni per i solutori iterativi e i controlli di convergenza. La sezione `solvers` definisce i solutori numerici utilizzati per risolvere le equazioni del modello. Può includere solutori per equazioni di trasporto, equazioni di equilibrio chimico, equazioni di turbolenza, ecc.

Per `alpha.water` vengono settate le keywords:

- `nAlphaCorr`: Specifica il numero di correzioni aggiuntive da eseguire per la correzione dell'alpha. È un parametro importante nelle simulazioni multiphase per garantire la stabilità e la convergenza delle soluzioni.
- `nAlphaSubCycles:` Specifica il numero di sotto-cicli da eseguire per la correzione dell'alpha. È utilizzato quando è necessario eseguire più iterazioni per stabilizzare la soluzione alpha.
- `cAlpha:` Coefficiente numerico utilizzato per la correzione dell'alpha. Il suo significato specifico può variare a seconda dell'implementazione e delle specifiche del problema.
- `MULESCorr:` Una keyword booleana che specifica se attivare o meno la correzione MULES (Multidimensional Universal Limiter with Explicit Solution). Quando attivata (`yes`), la correzione MULES viene applicata durante la simulazione.
- `nLimiterIter:` Specifica il numero massimo di iterazioni per il correttore del limite durante la correzione MULES. Questo

parametro controlla la precisione e la stabilità della correzione del limite.

- `solver:` Specifica il tipo di solutore numerico da utilizzare per risolvere l'equazione associata a `alpha.water`. Nel caso specifico, viene utilizzato `smoothSolver`, un solutore che applica un metodo di smoothing per migliorare la convergenza.
- `smoother:` Specifica il tipo di metodo di smoothing da utilizzare durante la risoluzione numerica. In questo caso, viene utilizzato `symGaussSeidel`, un metodo di Gauss-Seidel simmetrico.
- `tolerance:` Specifica la tolleranza per la convergenza del solutore. È il valore massimo di errore accettabile per la soluzione.
- `relTol:` Specifica una tolleranza relativa per la convergenza del solutore. In questo caso, è impostato a `0`, indicando che viene utilizzata solo la tolleranza assoluta definita dalla keyword `tolerance`.

Per `pcorr` vengono settate le keywords:

- `solver:` Specifica il tipo di solutore numerico da utilizzare per risolvere l'equazione associata a `pcorr.*`. In questo caso, è impostato su `PCG`, che indica il metodo del gradiente coniugato precondizionato.
- `preconditioner:` Definisce il precondizionatore da utilizzare con il solutore PCG. È un sottodizionario che contiene ulteriori keywords:
  - `preconditioner:` Specifica il tipo di precondizionatore da utilizzare. In questo caso, è impostato su `GAMG`, che

indica un precondizionatore algebrico multigriglia geometrico.

- `tolerance:` Specifica la tolleranza per la convergenza del precondizionatore. È il valore massimo di errore accettabile per la soluzione.
- `relTol:` Specifica una tolleranza relativa per la convergenza del precondizionatore.
- `smoother:` Specifica il tipo di metodo di smoothing da utilizzare durante la risoluzione numerica. In questo caso, è impostato su `DICGaussSeidel`, che indica un metodo di Gauss-Seidel con aggiustamento delle differenze centrali.
- `cacheAgglomeration:` Una keyword booleana che specifica se attivare o meno l'agglomerazione della cache. Quando attivata (`no`), l'agglomerazione della cache non viene utilizzata durante la simulazione.

- `tolerance:` Specifica la tolleranza per la convergenza del solutore PCG. È il valore massimo di errore accettabile per la soluzione.
- `relTol:` Specifica una tolleranza relativa per la convergenza del solutore PCG.
- `maxIter:` Specifica il numero massimo di iterazioni consentite per il solutore PCG prima di interrompere la simulazione. In questo caso, è impostato su `100`.

I solutori per `p_rgh, p_rghFinal, U, k` ed `epsilon` sfruttano le stesse kewords per cui viene rimandata alla descrizione sopraelencata. La scelta della parametrizzazione è rimandata a ciascuna grandezza di riferimento tenendo conto della dimensione fisica dell'ordine di grandezza

e prendendo spunto dalla letteratura dei modelli simili presenti nella sezione tutorial di openFOAM.

Il metodo `PIMPLE` (Pressure Implicit with Splitting of Operators) è un algoritmo numerico utilizzato in OpenFOAM per risolvere equazioni di Navier-Stokes per flussi di fluidi incomprimibili o quasi incomprimibili. È una variante del metodo SIMPLE (Semi-Implicit Method for Pressure-Linked Equations), che è ampiamente utilizzato nei codici di dinamica dei fluidi computazionale (CFD).Questo metodo utilizza un ciclo iterativo composto da cicli esterni e cicli interni. Durante i cicli esterni, il campo di pressione viene aggiornato in base alle correzioni delle velocità e viene applicata una correzione della pressione per garantire la soddisfazione dell'equilibrio di massa. Durante i cicli interni, vengono applicate correzioni alle velocità per garantire la soddisfazione dell'equilibrio di quantità di moto. Il rispettivo sottodizionario viene parametrizzato attraverso:

- `momentumPredictor`: Specifica se utilizzare il predittore di momento durante il ciclo PIMPLE. Quando impostato su `no`, indica che il predittore di momento non verrà utilizzato. Il predittore di momento è un passaggio aggiuntivo nel ciclo PIMPLE che può migliorare la stabilità numerica nelle simulazioni transitorie.
- `nOuterCorrectors`: Specifica il numero di cicli esterni da eseguire nel ciclo PIMPLE. Il ciclo PIMPLE è composto da cicli esterni e cicli interni, e questo parametro controlla quanti cicli esterni verranno eseguiti prima che il ciclo converga alla soluzione desiderata. In questo caso, è impostato su `1`.
- `nCorrectors`: Specifica il numero di cicli interni da eseguire nel ciclo PIMPLE. Il ciclo interno è responsabile della correzione

delle soluzioni in ciascun passaggio del ciclo esterno. In questo caso, è impostato su `4`.

- `nNonOrthogonalCorrectors`: Specifica il numero di correzioni non ortogonali da eseguire durante il ciclo. Le correzioni non ortogonali vengono utilizzate per gestire i problemi di non ortogonalità nella mesh. Quando impostato su `0`, indica che non verranno eseguite correzioni non ortogonali durante il ciclo.
- `pRefCell`: Specifica la cella di riferimento per la pressione. Questo valore viene utilizzato per calcolare il termine di correzione della pressione durante il ciclo.
- `pRefValue`: Specifica il valore di riferimento per la pressione. Questo valore viene utilizzato insieme alla cella di riferimento per calcolare il termine di correzione della pressione durante il ciclo.

I fattori di rilassamento sono coefficienti utilizzati nei solutori numerici per controllare la velocità di convergenza della soluzione. Durante ciascuna iterazione, i solutori cercano di avvicinarsi alla soluzione finale riducendo gradualmente l'errore residuo.

Il sottodizionario `relaxationFactors` descrive nella sezione `equations` che per tutte le equazioni (`".*"` è una regola di espressione regolare che corrisponde a tutti i nomi delle equazioni presenti nel solver, dove l'asterisco (`*`) indica che corrisponde a qualsiasi stringa di caratteri di qualsiasi lunghezza) viene settato il fattore di rilassamento 1, che significa che non viene applicato alcun attenuamento del residuo.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      2;
        nAlphaSubCycles 1;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter    4;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       10e-6;
        relTol          0;
    }

    "pcorr.*"
    {
        solver          PCG;
        preconditioner
        {
            preconditioner  GAMG;
            tolerance       1e-06;
            relTol          0;
            smoother        DICGaussSeidel;
            cacheAgglomeration no;
        }

        tolerance       1e-06;
        relTol          0;
        maxIter         100;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       10e-07;
        relTol          0.05;
    }
```

```
    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "(U|k|epsilon).*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       10e-07;
        relTol          0;
        minIter         1;
    }
}

PIMPLE
{
    momentumPredictor   no;
    nOuterCcorrectors    1;
    nCorrectors          4;
    nNonOrthogonalCorrectors 0;
       pRefCell    0;
       pRefValue    0;

}

relaxationFactors
{
    equations
    {
        ".*"                1;
    }
}


// ************************************************************************* //
```

Tabella 17 Prototipo del file fvSolution contenuto in $SIM_DIR/system

File `setFields`

Attraverso il dizionario `setFields` è possibile settare i valori dei campi all'interno della mesh. Questa utilità è serve per inizializzare i campi delle variabili in modo specifico prima di avviare una simulazione. In questo caso viene impostato il campo scalare `alpha.water` a 0 (aria) di default in tutta la mesh e poi viene creata tramite la kewords `regions` una zona a forma di parallelepipedo confinata dai vertici opposti (-0.18 -0.19 0) (0.18 0.19 0.125) che interseca la mesh in una serie di celle per le quali viene definito il campo scalare della frazione di fase pari ad 1 (acqua). In questo modo si riempie il serbatoio.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha.water 0
);

regions
(
    boxToCell
    {
        box (-0.18 -0.19 0) (0.18 0.19 0.125);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);

// ************************************************************************* //
```

Tabella 18 Prototipo del file setFields contenuto in $SIM_DIR/system

4.6 Analisi di sensibilità

Oltre ai parametri di qualità che definiscono globalmente la bontà dell'operazione di meshing, un'ulteriore operazione necessaria a validare le simulazioni risulta essere quella dell'analisi di sensibilità, che coinsiste nel prendere in esame diverse soluzioni del meshatore, con densità di cella differente, e pargonarne i risultati di post processing della simulazione CFD imponendo tutti i parametri della simulazione uguali per le diverse configurazioni.

Si sono quindi valutate 4 diverse configurazioni parametrizzate come segue:

	Caso 1	Caso 2	Caso 3	Caso 4
`maxCellSize`	`0.004`	`0.005`	`0.006`	`0.007`
`minCellSize`	`0.004`	`0.005`	`0.006`	`0.007`
`boundaryCellSize`	`0.004`	`0.005`	`0.006`	`0.007`

Tabella 19 Parametrizzazione delle diverse mesh per l'analisi di sensibilità

In tutti i casi le celle risultano uniformi; per mesh più dense il costo computazionale non sarebbe più sostenibile. È stato inoltre operato uno studio con densità non uniforme, più concentrata verso le pareti e minore verso il centro, dove localmente le velocità del flusso risultano inferiori. Immaginando infatti il fenomeno di sloshing come un sistema meccanico, di osserva la dinamica del pelo libero dell'acqua come un sistema che ruota attorno ad un perno con un fattore di richiamo e di smorzamento dovuto alla viscosità del fluido. Localmente la velocità tangenziale, essendo funzione della distanza dal centro di rotazione, risulta essere minore, per cui si è fatta l'ipotesi che a parità di numero di Courant l'infittimento locale nella regione centrale potesse essere minore. Il risparmio computazionale risultato non è risultato interessante a fronte di un forte degrado della qualità della mesh, per cui si è proseguito con le

4 configurazioni per l'analisi di sensibilità. Si riportano i parametri di qualità in output per ogni caso.

	Caso 1	Caso 2	Caso 3	Caso 4
internal points	214151	109129	63707	39775
non-orthog Max	55.50	37.09	54.83	34.95
non-orthog Average	1.98	2.05	3.14	2.87
Average skewness	0.012	0.015	0.019	0.021
Max skewness	0.629	0.55	0.82	0.59

Tabella 20 Parametri di qualità delle mesh oggetto di analisi di sensitività

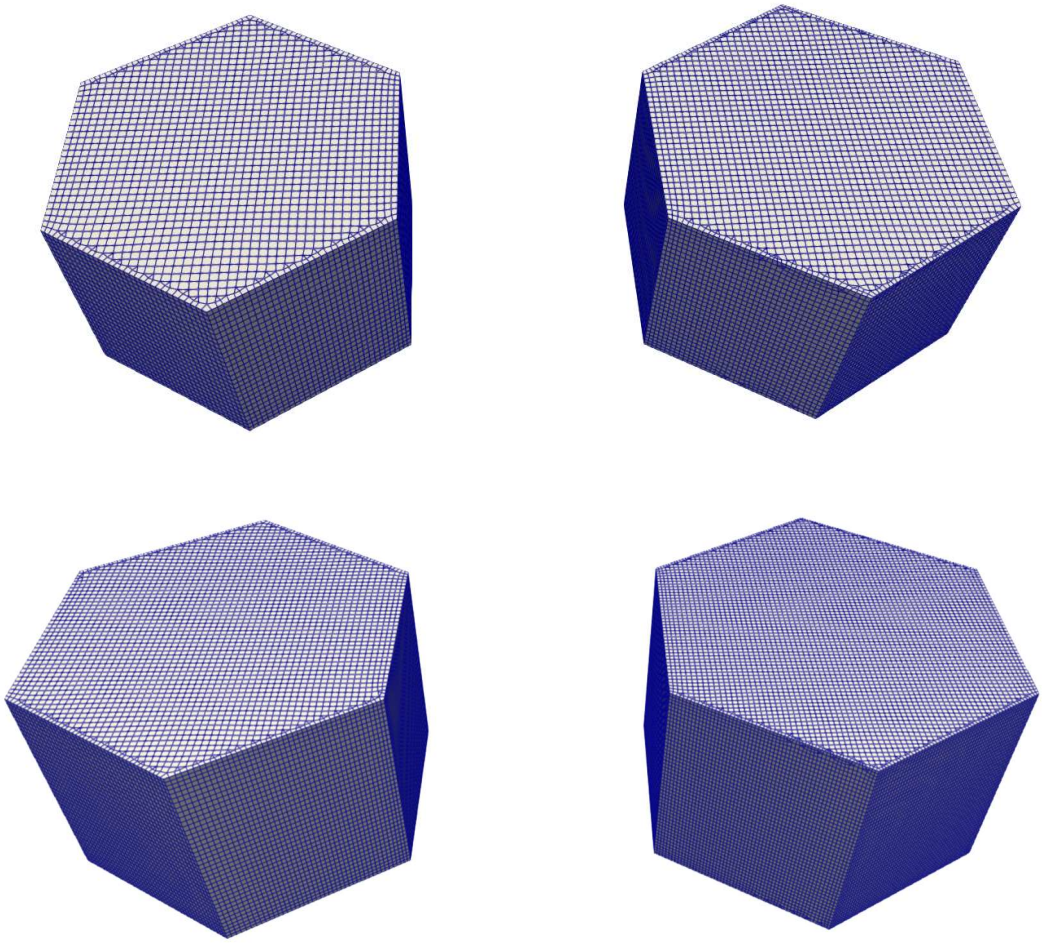


Figura 13 In alto a sinistra caso 4 di infittimento, in alto a destra caso 3, in basso a sinistra caso 2 e in basso e destra caso 1

La simulazione è stata poi eseguita con i seguenti parametri di controllo dal rispettivo dizionario:

```
application     interFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         10;
adjustTimeStep  on;
maxCo           1;
maxAlphaCo      1;
maxDeltaT       1;
```

Il profilo di accelerazione che eccita la dinamica di sloshing per le quattro simulazioni è ad impulso di ampiezza 3 m/s^2, dopo 0.05s, descritto nel file *accel.dat* nella directory *constant* del path di ciascuna simulazione.

Si osservano le forze in output, valutate come somma dei contributi di forze viscose e forze di pressione, lungo i 3 assi, sovrapponendo i 4 casi di studio; nei primi istanti la soluzione risulta quasi perfettamente sovrapponibile. Nel transitorio invece si osserva un discostamento in termini di ampiezza e frequenza di oscillazione

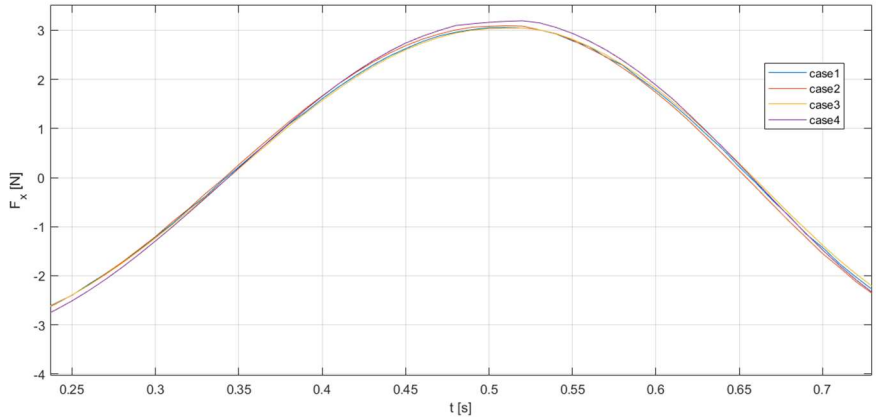


Figura 15 Differenza di ampiezza della risposta al variare dell'infittimento della mesh

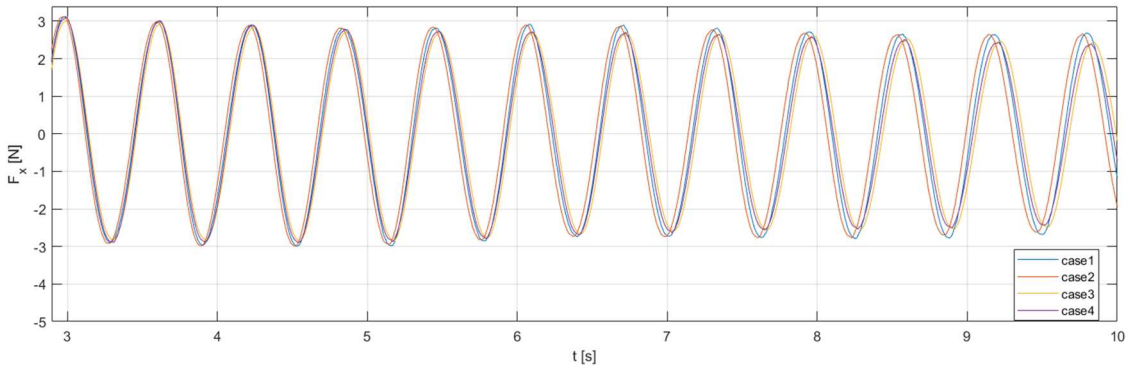


Figura 14 Risposta completa al variare dell'infittimento della mesh. è possibile notare lo sfasamento progressivo

È possibile valutare l'errore in pulsazione ed ampiezza per i diversi casi di infittimento rispetto alla soluzione con massimo infittimento. Valutando infatti l'errore percentuale relativo sull'intervallo temporale per ciascun massimo e minimo locale delle distribuzioni rispetto al caso 1 (massimo infittimento) si ottiene infatti

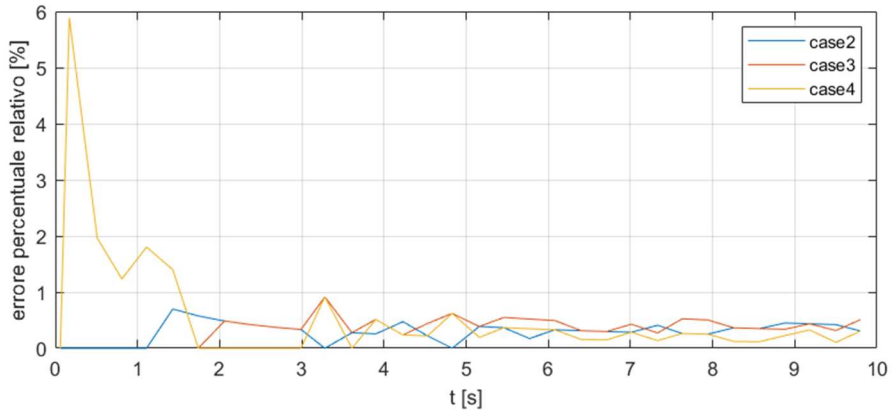


Figura 16 Errore relativo percentuale sulla pulsazione dell'analisi di sensibilità

Mentre per le ampiezze la distribuzione di errore percentuale nel tempo (sempre valutata sui massimi e minimi locali della funzione armonica di fitting) risultano essere, rispetto al caso di massimo infittimento:

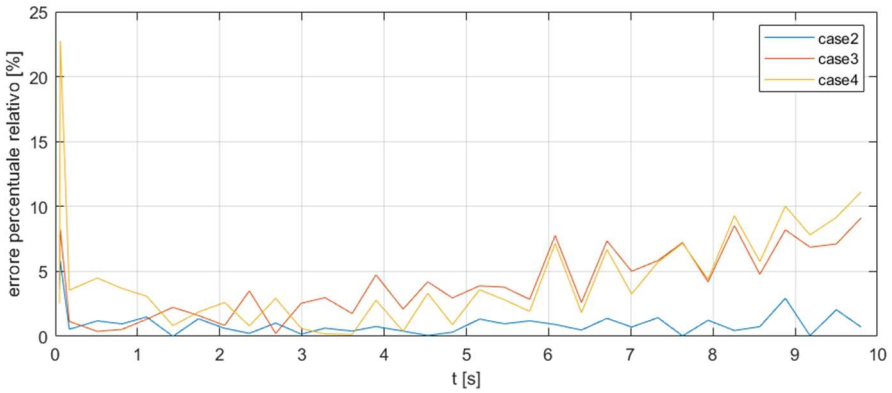


Figura 17 Errore relativo percentuale sulle ampiezze dell'analisi di sensibilità

Il caso 4, a minore infittimento, risulta essere più sincronizzato con il caso1 ma con picchi di ampiezza modulati da un errore percentuale maggiore, che supera il 10%. Il caso 2 invece risulta essere ideale, con un errore in pulsazione estremamente contenuto (0.3% sul transitorio) ed un errore relativo percentule sull'ampiezza sempre inferiore al 3%.

Monitorando il clocktime delle simulazioni tramite comando

```
interFoam > log.case1
```

che stampa in un file di log interno al path di simulazione per ogni δt di discretizzazione alcuni parametri caratteristici, tra i quali il clocktime, si osserva come l'infittimento della mesh vada a tradursi in tempi computazionali

Caso 1	Caso 2	Caso 3	Caso 4
15002 s	5731 s	2848 s	1493 s

Tabella 21 Andamento dei tempi di simulazione con l'infittimento della mesh

Il caso 2 risulta, al netto delle considerazioni fatte sugli errori percentuali con il caso 1, il miglior compromesso.

Capitolo 5

Casi di studio e post processing

5.1 Caso 1: Serbatoio senza paratie nelle configurazioni di volo in direzione della faccia piana e dello spigolo

La missione tipo del drone, di cui si dispone del ROS Topic di output della simulazione, prevede che da una condizione di hovering iniziale l'esarotore percorra la direzione del filare rettilineo raggiungendo e mantenendo una velocità costante pari a 2 m/s fino al termine dell'irrorazione, in cui decelera fino ad una nuova condizione di hovering. Allo stato dell'arte, per ragioni di interferenza nell'assemblaggio, il sistema di spruzzatura forma un angolo di 30° con l'asse x body del drone, mentre risulta perpendicolare ad una faccia piana del serbatoio, così come mostrato in figura:

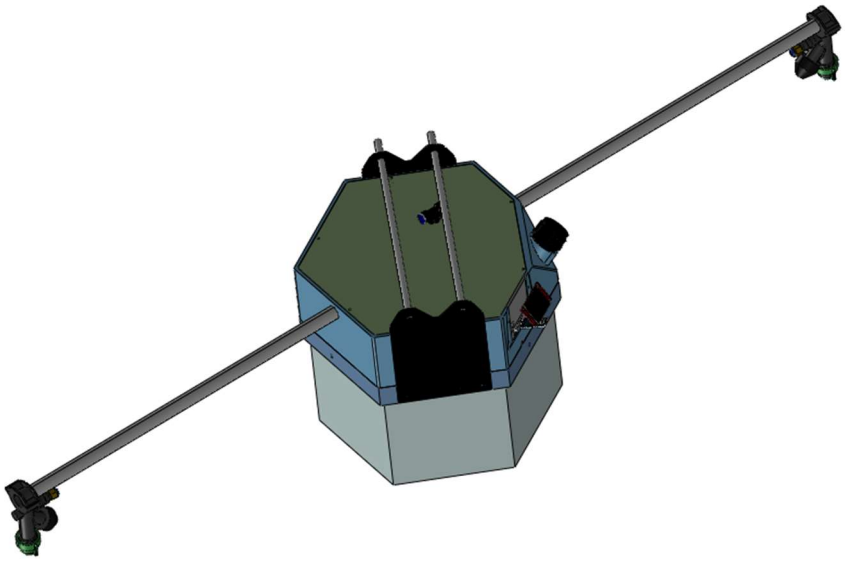


Figura 18 Modello CAD del sistema di spruzzatura

Questo disallineamento tra gli assi rende necessario sperimentare due diverse configurazioni di volo:

Con volo in direzione X_{DRONE}, ruotata di 30° rispetto ala direzione di riferimento della simulazione X, passante per due spigoli del serbatoio

Con volo diretto lungo X_{TANK} del serbatoio, coincidente con l'asse che percorre il sistema di irroratura, perpendicolare alla faccia piana del serbatoio

Volendo valutare unicamente gli effetti dinamici dello sloshing nelle due diverse configurazioni risulta meno oneroso mantenere un unico profilo di accelerazioni per la simulazione CFD e generando due mesh per il dominio di campo fluido ruotate opportunamente tramite feature di openFoam

```
transformPoints  -rollPitchYaw '(0 0 30)'
```

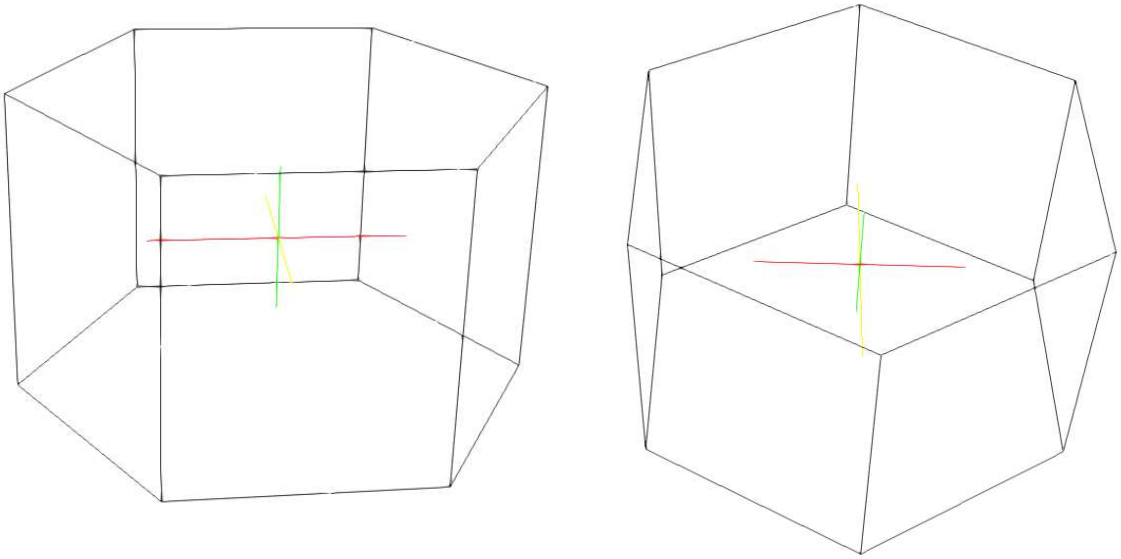


Figura 19 Orientamento delle due configurazioni del dominio di campo fluido nel primo caso di studio. A sinistra la direzione di volo (asse rosso) intercorre tra due spigoli, a destra interseca due facce piane.

I parametri per la generazione mesh del dominio di campo fluido sono stati scelti in seguito all'analisi di sensibilità descritta nel capitolo precedente, importando in input il file cad stl del serbatoio privo di paratie anti sciabordio.

il profilo di accelerazioni importato per il profilo di missione descritto è il seguente:

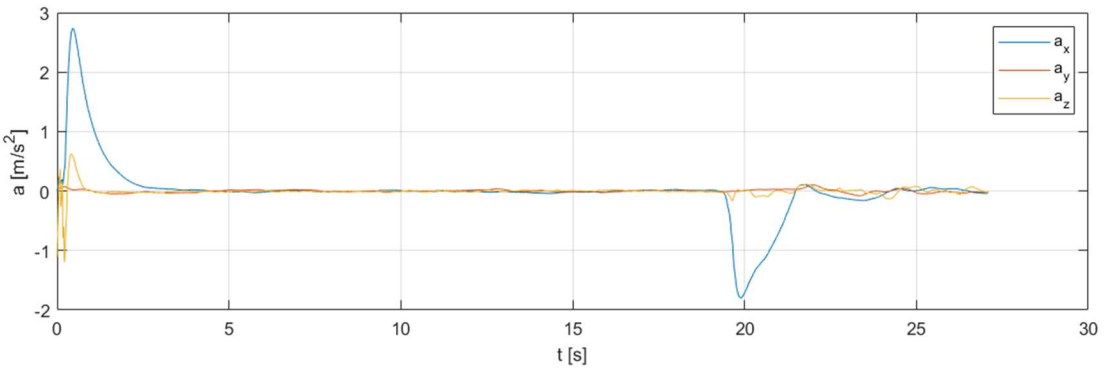


Figura 20 Profilo di accelerazioni importato in input (generato tramite ROS Topic) per la missione completa

Che descrive il seguente campo di velocità

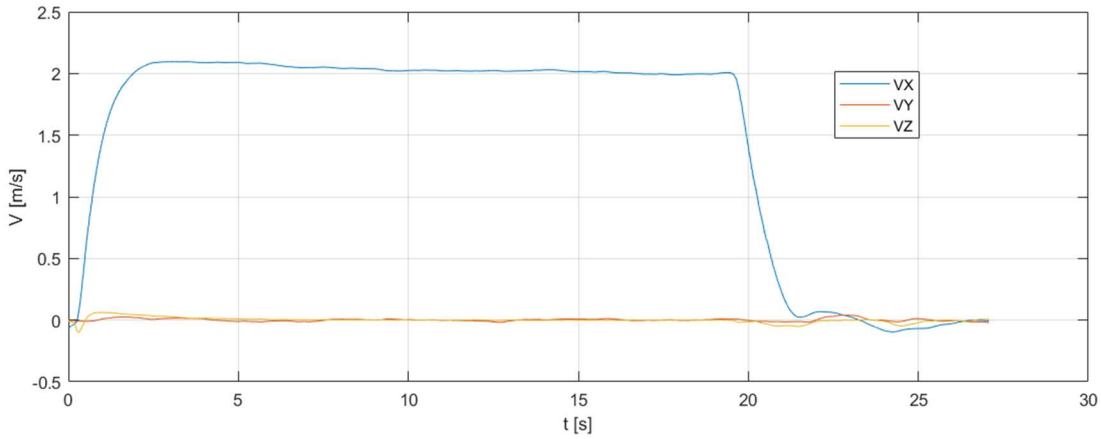


Figura 21 Profilo di velocità della missione completa

Si osservano un'eccitazione non solo nella direzione X ma anche in direzione Z e Y come conseguenza del sistema di stabilizzazione che interviene a seguito del rollio necessario alla movimentazione del drone dalla sua condizione stabile di hovering.

Il livello di riempimento del serbatoio viene impostato al 50% agendo sul file *setFieldsDict.*

```
regions
(
    boxToCell
    {
        box (-0.18 -0.19 0) (0.18 0.19 0.125);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);
```

Tabella 22 Porotitpo del core del file setFieldsDict interpretato dal dizionario setFields (eseguito prima della simulazione) per riempire virtualmente il serbatoio

Il risultato delle simulazioni viene salvato in un file chiamato *forces.dat* generato da openFOAM nel path `$simulation_path/postProcessing/forces/0` come desritto nella sezione dedicata di questa tesi. Per estrarre i dati dal formato dati predefinito in scrittura in output da OpenFOAM, non volendo andare a modificare il codice sorgente, si deciso di compilare un eseguibile programmato in C che, preso in input il file forces.dat dalla directory del progetto dell'eseguibile stesso (in cui il file *forces.dat* va incollato manualmente o tramite comandi da shell pre-programmati in un bash file) e restituisce in output un file con i vettori tempo, Fx, Fy ed Fz già in una sintassi compatibile con Matlab.

Tramite un'estensione applicativa chiamata Paraview è possibile osservare la visualizzazione dell'andamento delle variabili di stato e di campo nel dominio di calcolo attraverso scale grafiche ad ogni istante in cui è stata catturata la simulazione, scelta tramite parametro `writeInterval` del file `controlDict.` Questo parametro, impostato pari a 0.05, infatti ha l'effetto di generare, nel path di simulazione, una cartella con i valori di campo di tutte le variabili monitorate, ogni 0.05 secondi, denominandola progressivamente con listante di timestamp (0.05, 0.1, 0.15 ...). Questi sono gli istanti temporali osservabili tramite visualizzatore Paraview. Nelle immagini seguenti è stato catturato l'andamento della fase acqua (alpha.water) in alcuni istanti temporali.

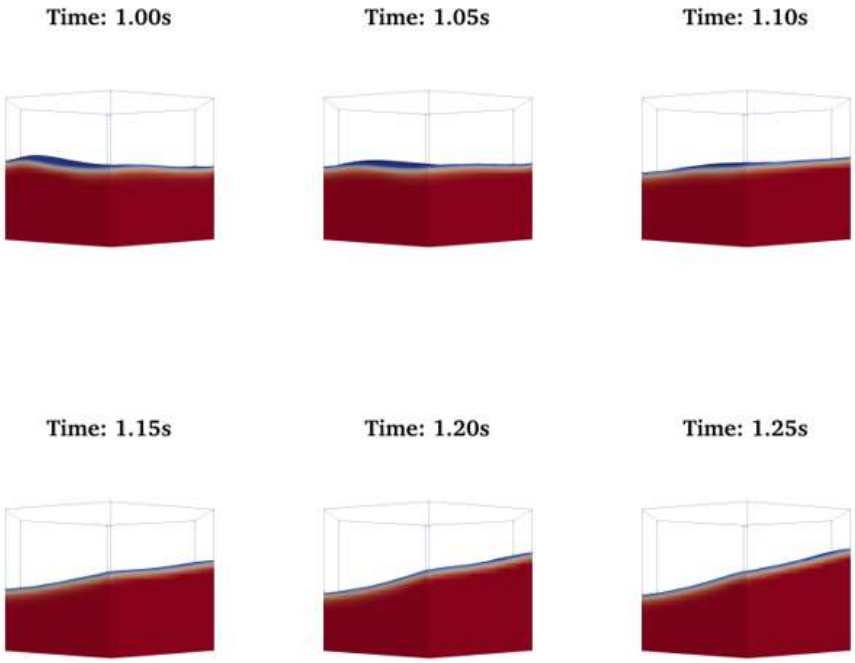


Figura 23 Cattura immagini di alcuni istanti della simulazione per la fase acqua nella simulazione con direzione di volo normale alla faccia piana

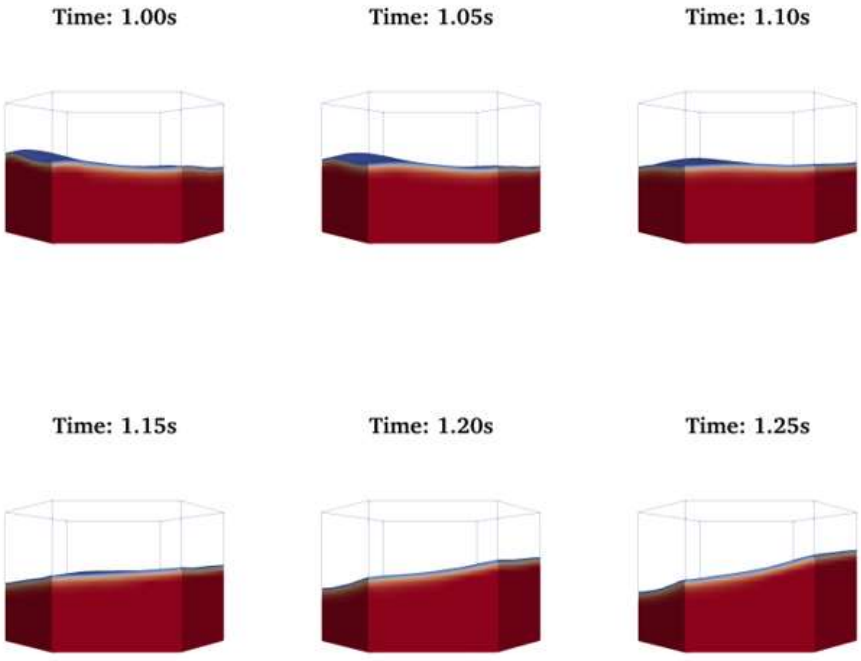


Figura 22 Cattura immagini di alcuni istanti della simulazione per la fase acqua nella simulazione con direzione di volo intersecante due spigoli

I risultati del post processing sono riportati di seguito:

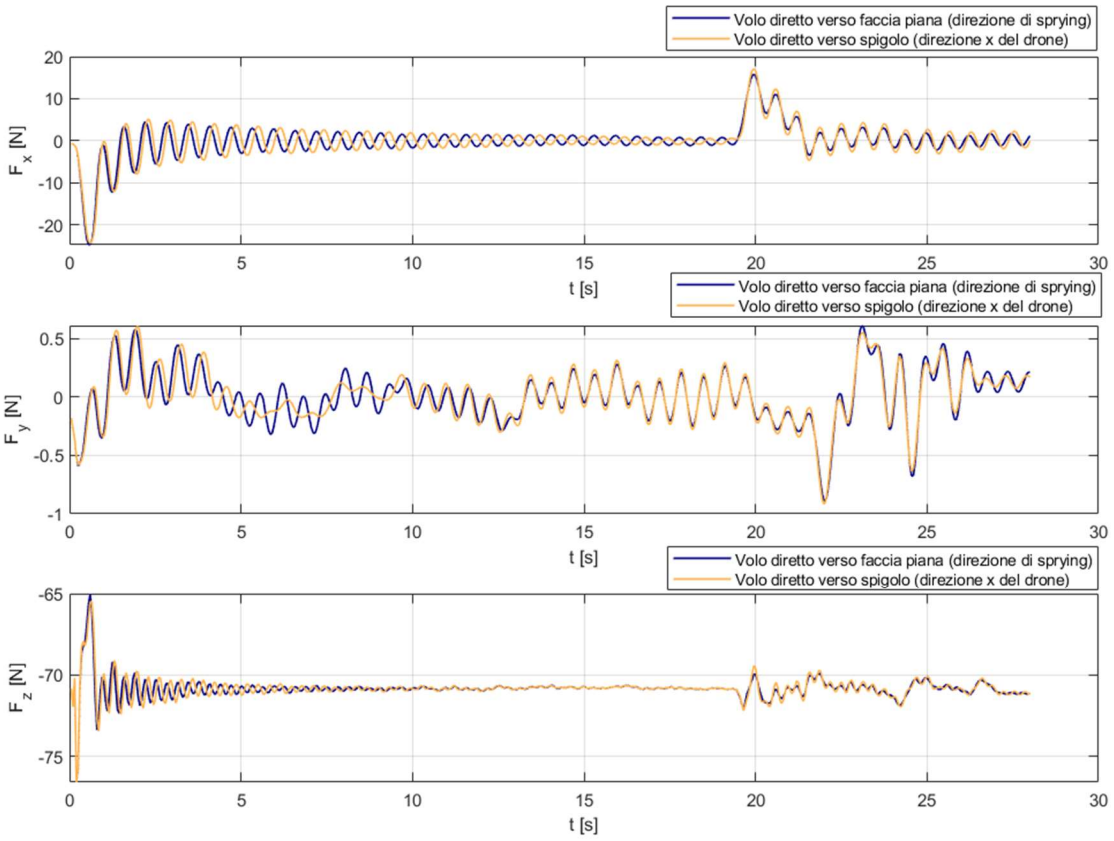


Figura 24 Profilo di forze generato in post processing per le simulazioni del caso di studio 1

Sull'asse x si osserva una differenza piccola in termini di ampiezza dell'oscillazione, dovuta alla regolarità della geometria, calcolata come integrale delle pressioni locali. Queste ultime risulteranno diverse nei nodi, dovute all'effetto di interferenza tra la diversa geometria delle onde riflesse, ma globalmente, come effetto della pressione sulle pareti, la differenza tra le configurazioni è minima. La forza di picco della prima onda risulta, come atteso, maggiore nel caso di volo in direzione della faccia piana, mentre nel transitorio l'effetto smorzante di questa configurazione determina un'ampiezza inferiore. In termini di frequenza

la differenza è più marcata, dovuto soprattutto alla lunghezza caratteristica del moto fluido, maggiore sulla semidiagonale rispetto all'apotema, da cui una frequenza maggiore nel caso di volo diretto tra facce piane.

Sull'asse y si ha una forza risultante dalla dinamica due ordini di grandezza minore: le accelerazioni importate in questa direzione hanno un contributo del sistema di stabilizzazione e un contributo dovuto al rumore di fondo. La validità dei dati di accelerazione importati rimane legata alla precisione e sensibilità dei sensori, alla qualità di progettazione e assemblaggio geometrico degli stessi e all'effetto dei filtri di pulizia del rumore di fondo come il filtro di Kalman. Per meglio comprendere questi effetti sulla dinamica rispetto a quelli indotti dal moto lungo x, e l'eventuale possibilità di trascurarli, risulta utile confrontare il caso di serbatoio orientato con faccia piana in direzione del moto, che rappresenta lo stato dell'arte, importando il profilo di accelerazioni di missione riportato in precedenza, con un caso di medesima geometria e orientamento ma con il solo effetto delle accelerazioni lungo x, direzione del moto, come in immagine:

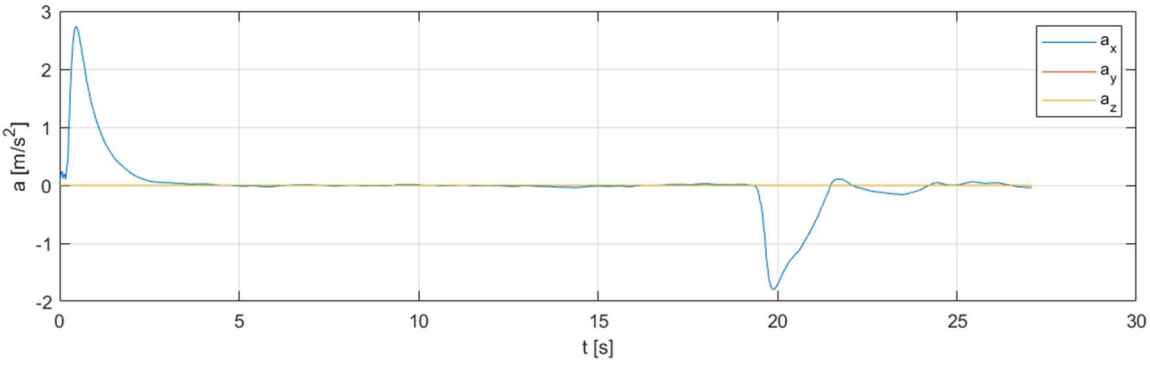


Figura 25 Profilo di accelerazioni semplificato (solo direzione x)

L'output del processo di post processing risulta quindi essere il seguente

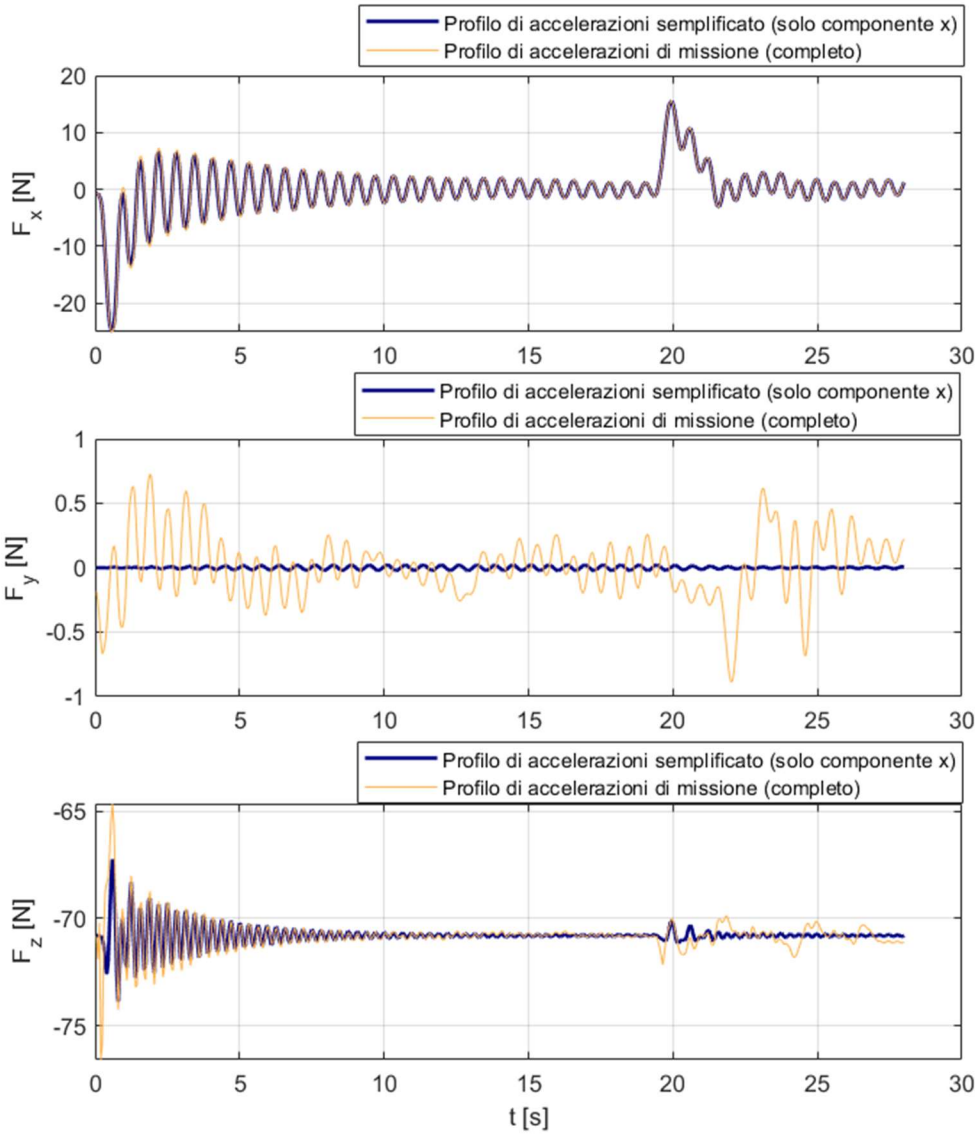


Figura 26 Profilo di forze generato in post processing per le simulazioni con evidenza sull'approssimazione di componenti yz nulle dell'accelerazione nel profilo in input rispetto al profilo completo per la missione

Si può osservare come l'effetto della forza di sloshing lungo x rimanga circa invariato; questo risulta essere l'output più interessante al fine di studiare il condizionamento del fenomeno sul sistema, con picchi di 24 N in valore assoluto.

Lungo y, avendo scorporato l'effetto delle accelerazioni lungo questo asse, la simulazione produce in output un picco di accelerazione di 0.01 N, rispetto agli 0.8 del profilo di accelerazioni completo.

Lungo z le correzioni di assetto risultano più marcate e l'effetto della semplificazione sul profilo di accelerazioni risulta più gravoso. In entrambi i casi l'evoluzione dinamica è quella di un moto armonico smorzato, eccitato in prevalenza dal fenomeno dello sloshing, intorno ad un asintoto di stabilità dettato dal peso del fluido stesso contenuto.

Volendo scorporare gli effetti di sloshing dagli effetti dinamici inerziali di moto di corpo rigido, si ottiene:

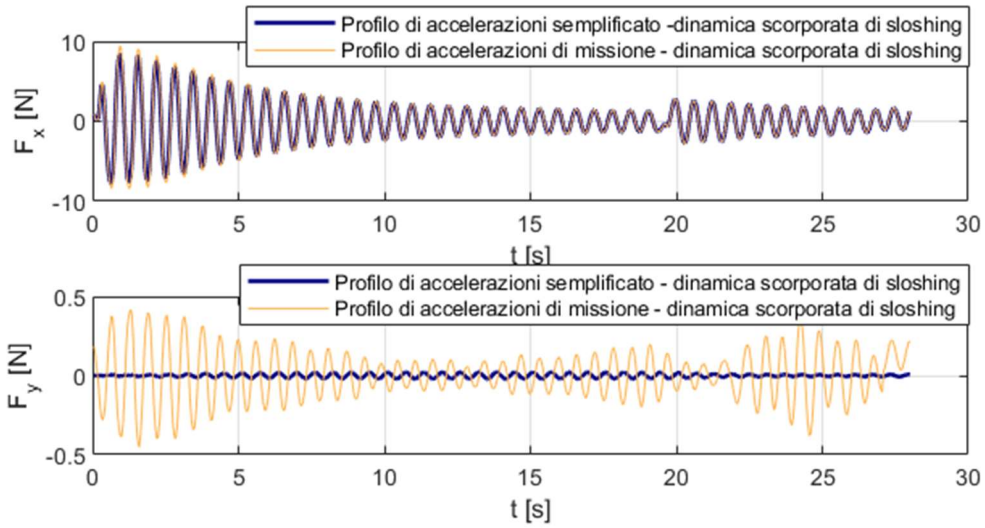
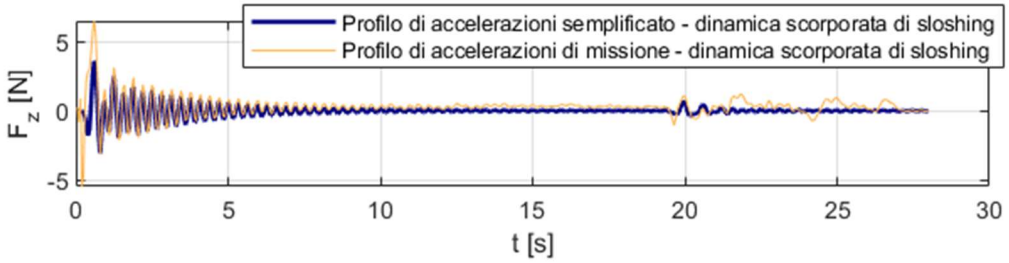


Figura 27 Effetto della dinamica di sloshing scorporato dell'effetto inerziale sui profili di missione completo e semplificato.



Tramite questa operazione è possibile apprezzare il ritardo della risposta dinamica di sloshing sull'input e il reale andamento del fenomeno. Si osserva che durante la fase di accelerazione l'effetto di sloshing assume valori prossimi al 38% dell'effetto inerziale, ma nel transitorio ad accelerazione cessata lo smorzamento è scarso, con tempi di dimezzamento superiori ai 5 secondi.

5.2 Caso 2: Serbatoio con paratie nelle due configurazioni di volo e proposta di modifica progettuale configurazione sperimentale

Per mitigare l'effetto dello sloshing sono state progettate delle paratie anti sciabordio ed un reticolo di assemblaggio che è stato fissato nella superficie interna del serbatoio. Il reticolo permette la possibilità di inserire più paratie in diverse configurazioni concentriche, opportunamente dotate di asole complementari per l'incastro nella zona centrale. Le paratie, potendo essere assemblate da faccia piana a faccia piana o nella direzione spigolo-spigolo presentano due possibili configurazioni geometriche:

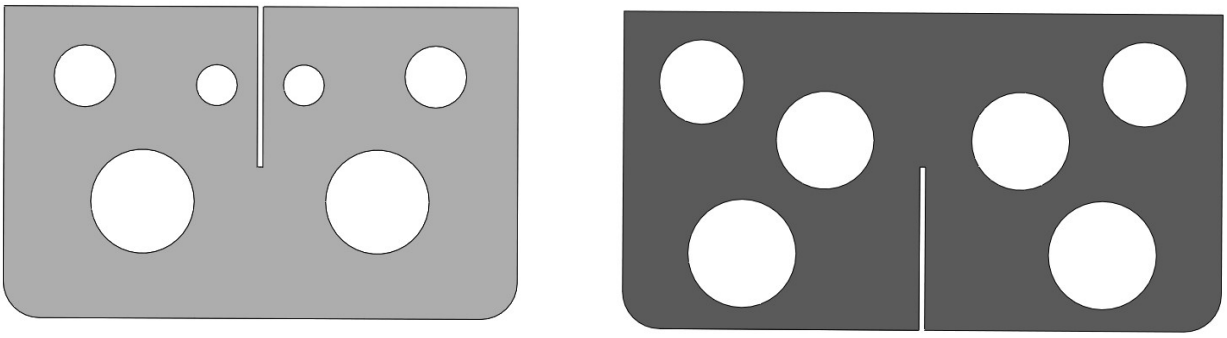


Figura 28 deflettori radiali pre-progettati (presenti allo stato dell'arte). A sinistra quello corto, assemblato perpendicolarmente a due facce piane, a destra quello lungo, assemblato tra due spigoli

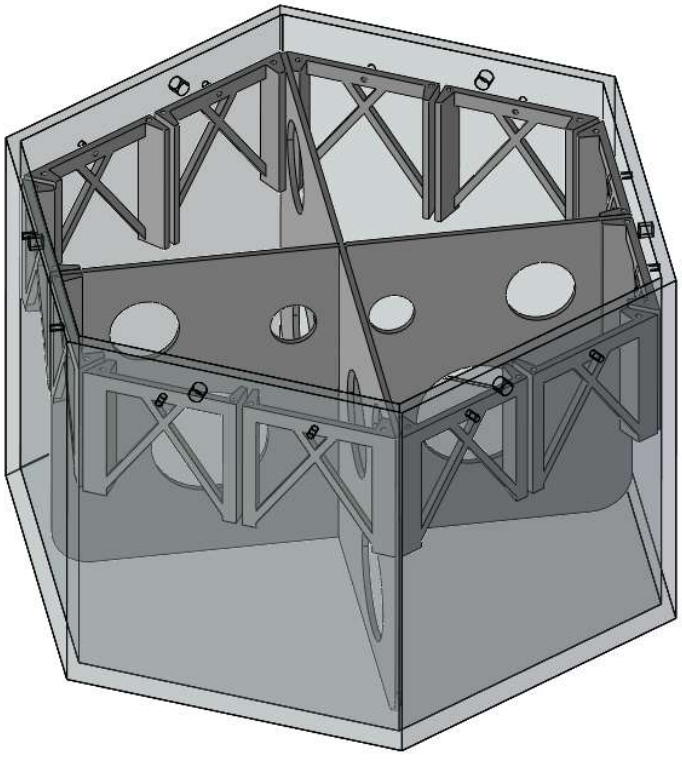


Figura 29 Modello cad del serbatoio in trasparenza con evidenza sul sistema di accoppiamento per assemblare i deflettori al suo interno

Per semplicità nella creazione della mesh si è deciso di trascurare nel modello la presenza della griglia di assemblaggio; l'effetto dinamico introdotto da questo elemento si prefigura più come un restringimento locale del volume di fluido, pari allo spessore della griglia di 3 mm, che in un vero e proprio contributo modale alla dinamica. La mesh per il dominio di campo fluido è stata quindi generata importando in input un stl file con presenti le paratie già assemblate tramite assieme CAD in solidworks.

Una prima analisi è stata fatta in analogia con il serbatoio vuoto, agendo sulla rotazione dell'intero dominio fluido.

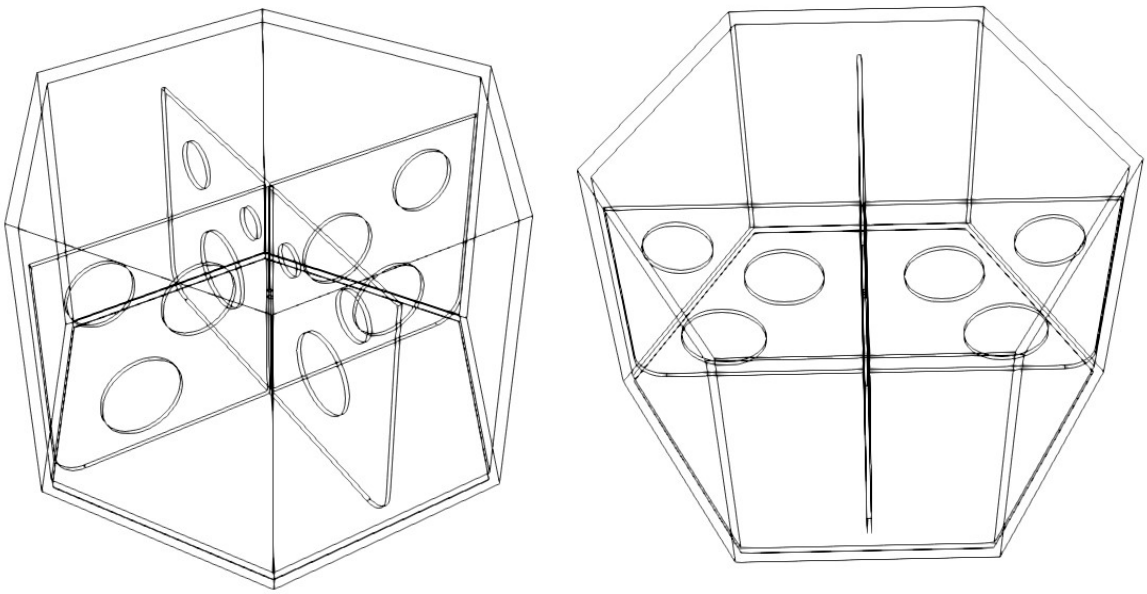


Figura 30 Serbatoi in trasparenza nei duoi orientamenti caratteristici. A sinistra si può osservare l'asimmetria dovuta alla disposizione dei deflettori radiali allo stato dell'arte

L'assemblaggio delle paratie è stato fatto con una paratia lunga e una corta disposte perpendicolarmente.

Si osserva che nel caso di volo diretto da spigolo a spigolo, nell'immagine a sinistra, questa scelta di assemblaggio comporta una asimmetria, con l'asse x che forma un angolo di 30° con la normale al piano della paratia corta ed uno di 60° con la normale al piano della paratia lunga. Per questo motivo è stata generata una geometria orientata spigolo-spigolo con assemblaggio di due paratie lunge a 120°, come in immagine seguente

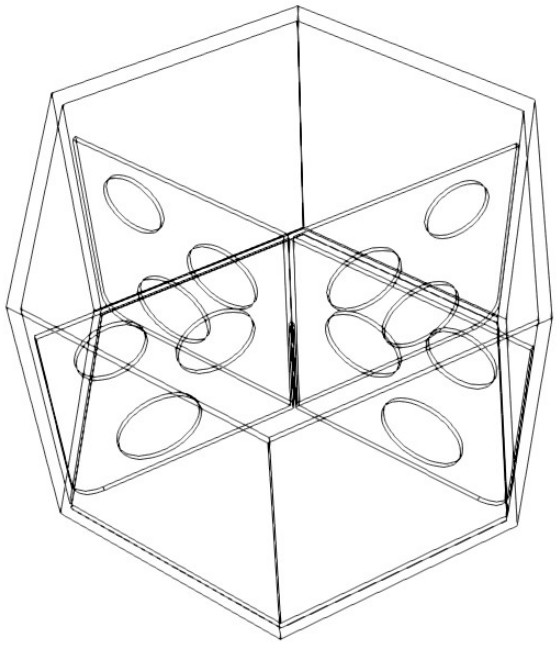


Figura 31 Serbatoio in trasparenza con configurazione sperimentale composta da due deflettori radiali lunghi montati entrambi tra spigoli opposti. La direzione del moto è coincidente con l'asse di vista del lettore

Per la generazione della mesh è stato operato un refinement locale tramite feature nel dizionario *meshDict*

```
surfaceMeshRefinement
{
      mesh_baffle
      {
            surfaceFile       "bafflesm.stl";
            additionalRefinementLevels    2;
      }
}
```

Tabella 23 Estratto del file MeshDict con feature per affinamento locale della mesh in corrispondenza dei deflettori

Per la simulazione il parametro di riempimento è del 50% con il profilo di accelerazioni in input, in seguito alle considerazioni del paragrafo precedente, preso approssimato con la solo componente x.

Si riportano alcuni momenti catturati dalla simulazione diretta verso lo spigolo:

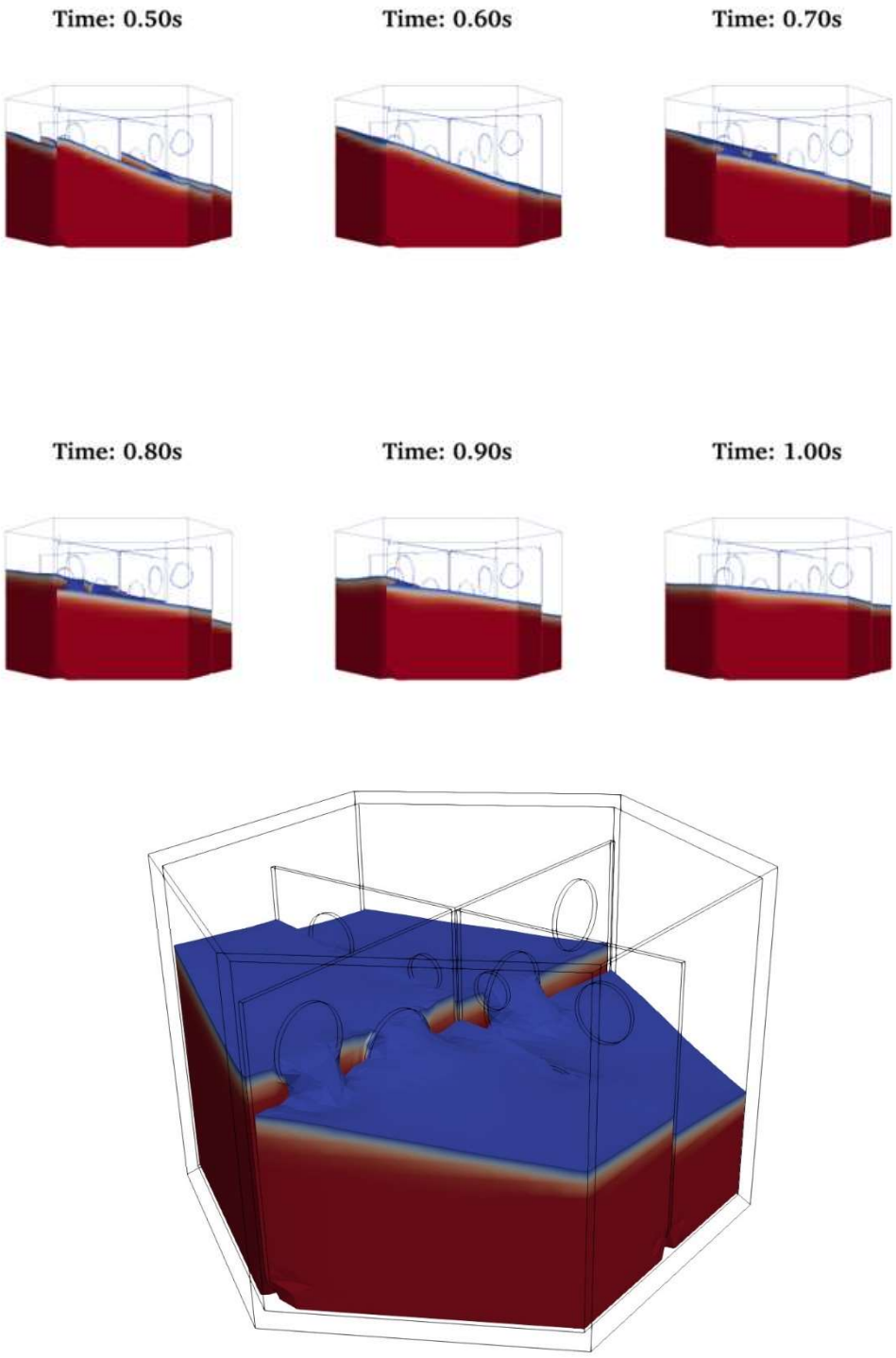


Figura 32 Cattura immagini della simulazione da visualizzatore Paraview. In basso si osserva il dettaglio dell'evoluzione fluida attraverso i fori preconfigurati sui deflettori

In post processing risulta utile, per valutare l'effetto della dinamica di sloshing, scorporarla degli effetti di moto di corpo rigido. L'accelerazione impartita sulla massa determina, come osservato nel caso 1 di studio, un effetto di forza dovuta al moto, intorno al quale avviene l'oscillazione di sloshing. Questo effetto risulta già ben visibile nel caso dei serbatoi vuoti in direzione verticale, dove la componente di peso dovuta alla massa determina la risposta stazionaria. In figura si osserva, per il profilo di missione completo, l'effetto della dinamica di sciabordio e dello smorzamento caratterizzato dai deflettori radiali nelle tre configurazioni. Il moto di sciabordio è preponderante nella direzione del moto, lungo x con un effetto più marcato nel volo diretto verso la faccia piana determina un picco di overshoot del 7% maggiore rispetto al volo diretto tra due spigoli. Questa discrepanza è dovuta in parte allo smorzamento caratteristico della configurazione geometrica, che nel caso di volo tra due spigoli determina una lunghezza caratteristica del moto maggiore, già osservata nel caso di serbatoi vuoti, in parte dovuto alla genesi di una componente trasversale al moto in direzione y di forza di sloshing; La asimmetria della configurazione con paratie a 90° in direzione dello spigolo motivata in precedenza genera una componente di sloshing in questa direzione. L'effetto di questa componente è, in termini di ampiezza massima, pari al 18,6% (0.71N contro i 3.82N della forza di sloshing lungo x) rispetto a quella della direzione del moto. Occorre tenere conto però che in questa direzione non vi sono alte fonti di accelerazione diverse da quella di sloshing, differentemente dalla direzione del moto che presenta anche il moto di corpo rigido. Sommando questo effetto il picco massimo si ha per 22.29 N.

La configurazione sperimentale, con due paratie lunghe in simmetria rispetto alla diagonale, determina un overshoot minore del 22.5% rispetto al caso di volo diretto in direzione della faccia piana, ma mantenendo contenuto l'effetto in direzione trasversale, non possedendo l'asimmetria della configurazione diretta tra due spigoli allo stato dell'arte. In termini di smorzamento si può calcolare tramite il metodo del decremento logaritmico. Questo metodo è particolarmente utile quando il sistema è sottosmorzato. Il decremento logaritmico si basa sulla misura degli estremi successivi (massimi e minimi) dell'oscillazione. In pratica, si prendono due picchi successivi dello stesso segno (due massimi o due minimi) e si calcola il logaritmo naturale del rapporto tra l'ampiezza del primo picco e quella del secondo. La formula del decremento logaritmico δ è data da

$$\delta = \ln\left(\frac{x(t_1)}{x(t_2)}\right)$$

Dove $x(t_1), x(t_2)$ sono le ampiezze dei picchi successivi. Il coefficiente di smorzamento ζ può essere derivato da δ utilizzando la relazione

$$\zeta = \frac{\delta}{\sqrt{4\pi^2+\delta^2}}$$

I valori per il coefficiente di smorzamento calcolato con questo metodo sono $\zeta_1 = 0.038$ $\zeta_2 = 0.040$ e $\zeta_3 = 0.037$ nel caso sperimentale, che risulta quindi leggermente meno smorzato sul periodo ma con performance migliori sul primo picco.

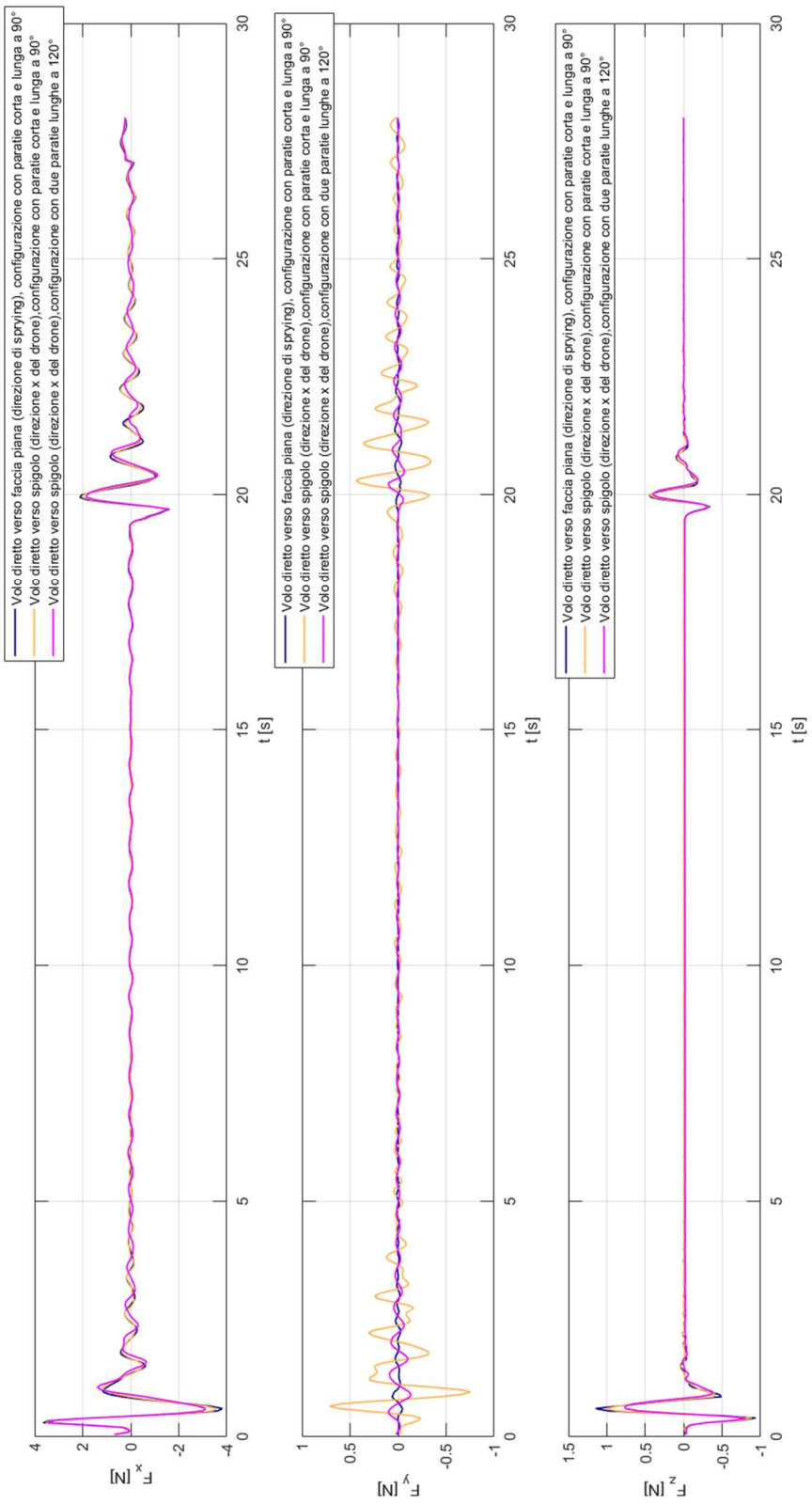


Figura 33 Plot delle forze di sloshing sulle tre varianti del caso di studio 2

5.3 Caso 3: Risposta a gradino

Per paragonare gli effetti smorzanti risulta utile un'analisi della risposta a gradino per la configurazione senza paratie e per l'equivalente configurazione con paratie. Questa permette di evidenziare il reale effetto di smorzamento dei deflettori rispetto al serbatoio sprovvisto. Viene quindi modellato un profilo di accelerazioni a gradino di ampiezza pari ad $a = [2.6 \;\; 0 \; 0]\, m/s^2$, avendo preso come riferimento il massimo valore di accelerazione lineare lungo x assunto durante il profilo di missione. La risposta dinamica del sistema nelle due confiurazioni è la seguente:

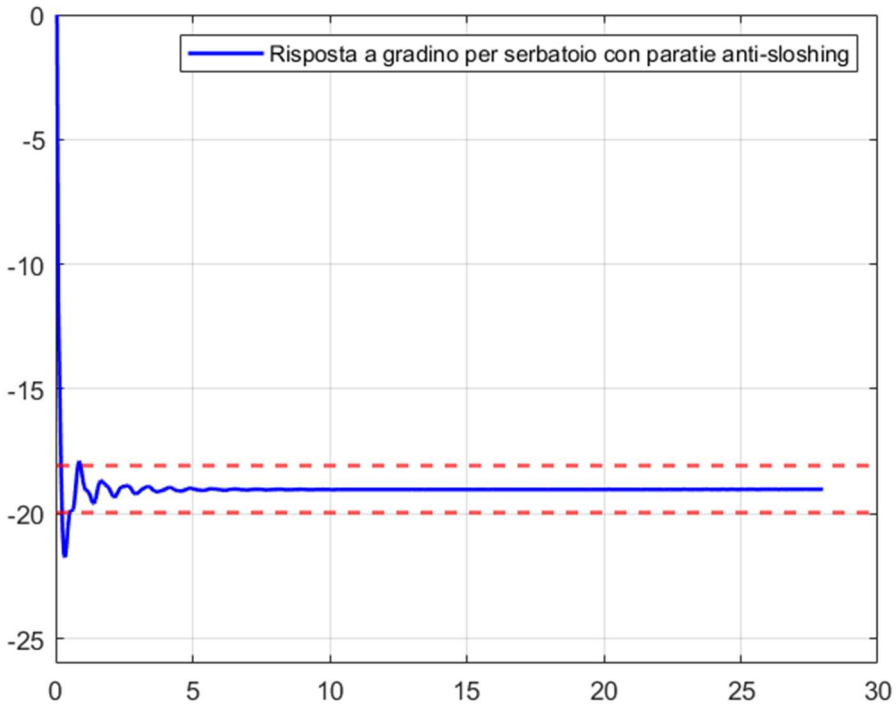


Figura 34 Risposta a gradino per FX nel caso di serbatoio con deflettori radiali

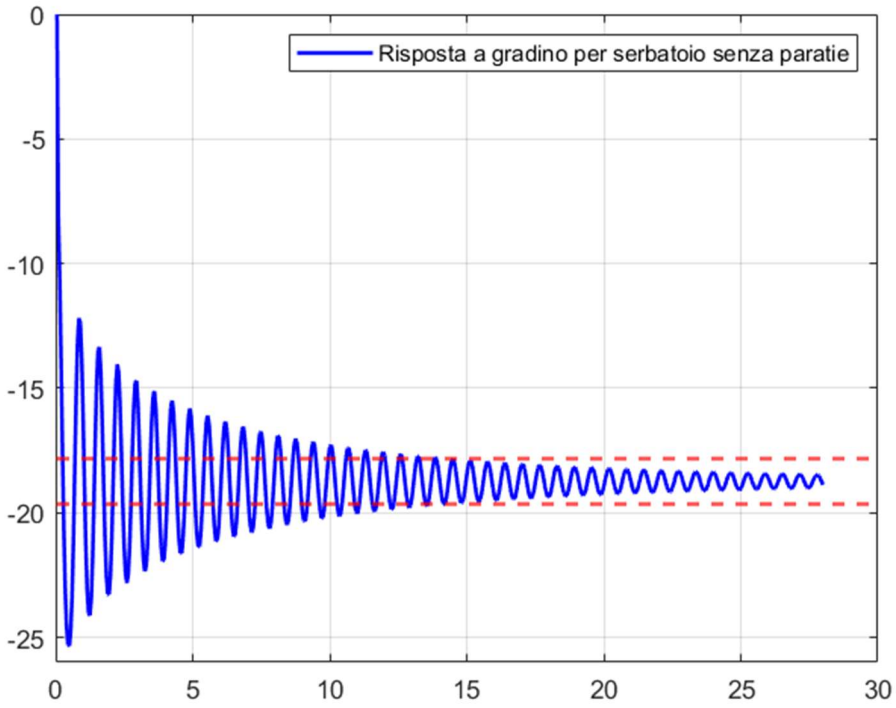


Figura 35 Risposta a gradino per FX nel caso di serbatoio senza deflettori

Si definisce il *settling time* come il tempo t tale per cui la risposta finale rimane confinata in una banda $\epsilon = 5\%$

$$| y(t_{settling}) - y_{final} | \leq \epsilon$$

In figura viene tracciata in rosso la banda ϵ

Il $t_{settling}$ per il serbatoio senza paratie risulta pari a 13.2 s, mentre per il serbatoio con le paratie pari a 0.9 s.

Un'altra caratteristica che si osserva è l'overshoot, definita come la massima sovraelongazione del segnale espressa in percentuale rispetto al valore stazionario. Per il modello senza paratie questo parametro assume il valore di $overshoot\%_{senza\ paratie} = 35.28\ \%$ mentre per il modello con paratie pari a $overshoot\%_{con\ paratie} = 14.35\ \%$

5.4 Caso 4: Variazione parametrica del riempimento su serbatoio con deflettori allo stato dell'arte

In ultima analisi si vuole studiare l'effetto del riempimento del serbatoio nella dinamica di sloshing nel serbatoio con le paratie. Viene scritto un bash file per la generazione automatica delle simulazioni riportato di seguito:

```
#!/bin/bash

mypath='/home/volpatto/OpenFOAM/volpatto-11/run/RiempimentoVar'
backupPath='/home/volpatto/OpenFOAM/volpatto-11
      /run/RiempimentoVar/backup'


# Definisco le percentuali di riempimento per le simulazioni in decimali
riempimento=(0.001 0.1 0.25 0.3 0.5 0.6 0.7 0.75 0.8 0.9 1)
# Ciclo for per ogni elemento del vettore
 for ((i=0; i<${#riempimento[@]}; i++)); do
      cd $mypath
    # # Calcola il valore per il parametro
      percentualeRiempimento=$(awk "BEGIN {printf
      \"%.0f\",${riempimento[i]} * 100}")

    ## Crea il nome della cartella
      chmod u+w "$mypath"
      folder_name="${percentualeRiempimento}%riempimento"

    # # Crea la cartella
      mkdir -p "$folder_name"
      chmod u+w "$folder_name"

      # # Crea le sottocartelle
      mkdir -p "$folder_name/0"
      mkdir -p "$folder_name/constant"
      mkdir -p "$folder_name/system"

      # # Copia gli elementi nelle sotocartelle
      cp -r $backupPath/0 $folder_name
      cp -r $backupPath/system $folder_name
      cp -r $backupPath/constant $folder_name
      cp -r $backupPath/MeshBuffles2Conf/polyMesh $folder_name/constant
      cp -r $backupPath/accelerazioneX/accel.dat $folder_name/constant

      # # Percorso del file setFieldsDict
      file_path="$folder_name/system/setFieldsDict"

      # # Parametro passato allo script
      parameter=$(echo "$(awk 'BEGIN {printf "%.3f", '${riempimento[i]}'
      * 0.2}') + 0.02" | bc)
```

```
        parameter=$(printf "%.3f\n" $parameter)

        # # Contenuto da sostituire
        replacement="        box (-0.18 -0.19 0) (0.18 0.19 $parameter);"

        # # Utilizza sed per sostituire la riga 26 con il nuovo contenuto
        sed -i "26s/.*/$replacement/" "$file_path"

        cd $folder_name
        setFields
        interFoam
        cd $folder_name/postProcessing/forces/0
        touch outforces_dat.txt
        cp forces.dat /mnt/c/acc2/forces.dat
        cd /mnt/c/acc2
        gcc -o logRead2 logRead2.c
        ./logRead2
        cp outforces_dat.txt
        $folder_name/postProcessing/forces/0/outforces_dat.txt

done
```

Questo shell script setta nelle prime due righe gli alias contenenti il path esterno in cui creare le cartelle di simulazione, quindi definisce un vettore contenente i livello di rimpimento (decimali) per il serbatoio e ciclicamente per ogni posizione del vettore genera le directory di simulazione con il nome riferito al livello di rimpimento dell'i-esima posizione del vettore riempimento, copiando i file di simulazione da una cartella backup, e modifica tramite comando *sed* la riga ventiseiesima del file *setFieldsDict* che contiene la parametrizzazione del livello di riempimento, sostituendo l'altezza fino a cui si desidera riempire il serbatoio [m] con quella calcolata in base al livello di riempimento i-esimo del vettore. Successivamente impartisce i comandi per eseguire la simulazione e per il post processing. Osservando l'output di forza sull'asse z, riportato di seguito, è possibile trarre due importanti considerazioni:

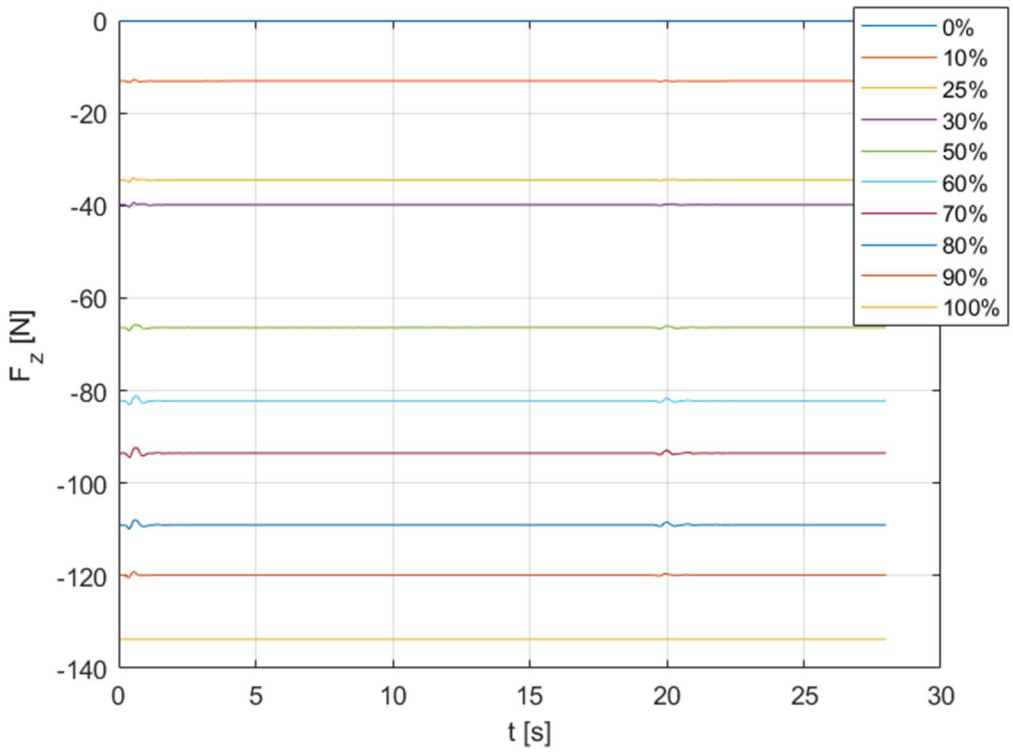


Figura 36 Forze totali lungo z al variare del riempimento. Si osserva la componente stazionaria dovuta al peso del fluido

La prima considerazione, per quanto banale, riguarda l'effetto del riempimento che determina la componente stazionaria, crescente in valore assoluto con il livello di riempimento, a validare la modellizzazione, partendo da 0 nel caso vuoto al valore stazionario in condizione piena coincidente con quello calcolabile tramite volume in output dalla geometria CAD moltiplicato per la densità e accelerazione gravitazionale.

La seconda considerazione riguarda l'effetto sloshante: Si osserva che l'overshoot rispetto al valore stazionario non è crescente con il riempimento ma ha un massimo intorno al 70%. Questo risultato, per altro atteso e già ampiamente descritto in letteratura, è dovuto allo spazio libero attraverso cui l'onda può evolvere: per valori di riempimento successivi lo spazio vuoto occupato dalla seconda fase, in questo caso aria, è sufficientemente piccolo da smorzare l'effetto dinamico dell'onda di sloshing.

Lungo x questo effetto è meno immediato da osservare. Nella pagina seguente si riporta L'estratto completo della forza totale agente in direzione x al variare del riempimento. La dinamica si compone, come nel caso di asse verticale, di un modo di corpo rigido e la combinazione lineare di modi armonici caratteristici della dinamica di sloshing; mentre nel caso di asse verticale il modo di corpo rigido si traduce solo in una componente stazionaria costante, essendo l'accelerazione lungo l'asse z, pari a $g = 9.81\, m/s^2$ costante, in questo caso l'accelerazione segue il profilo di missione, e di conseguenza anche il modo di corpo rigido.

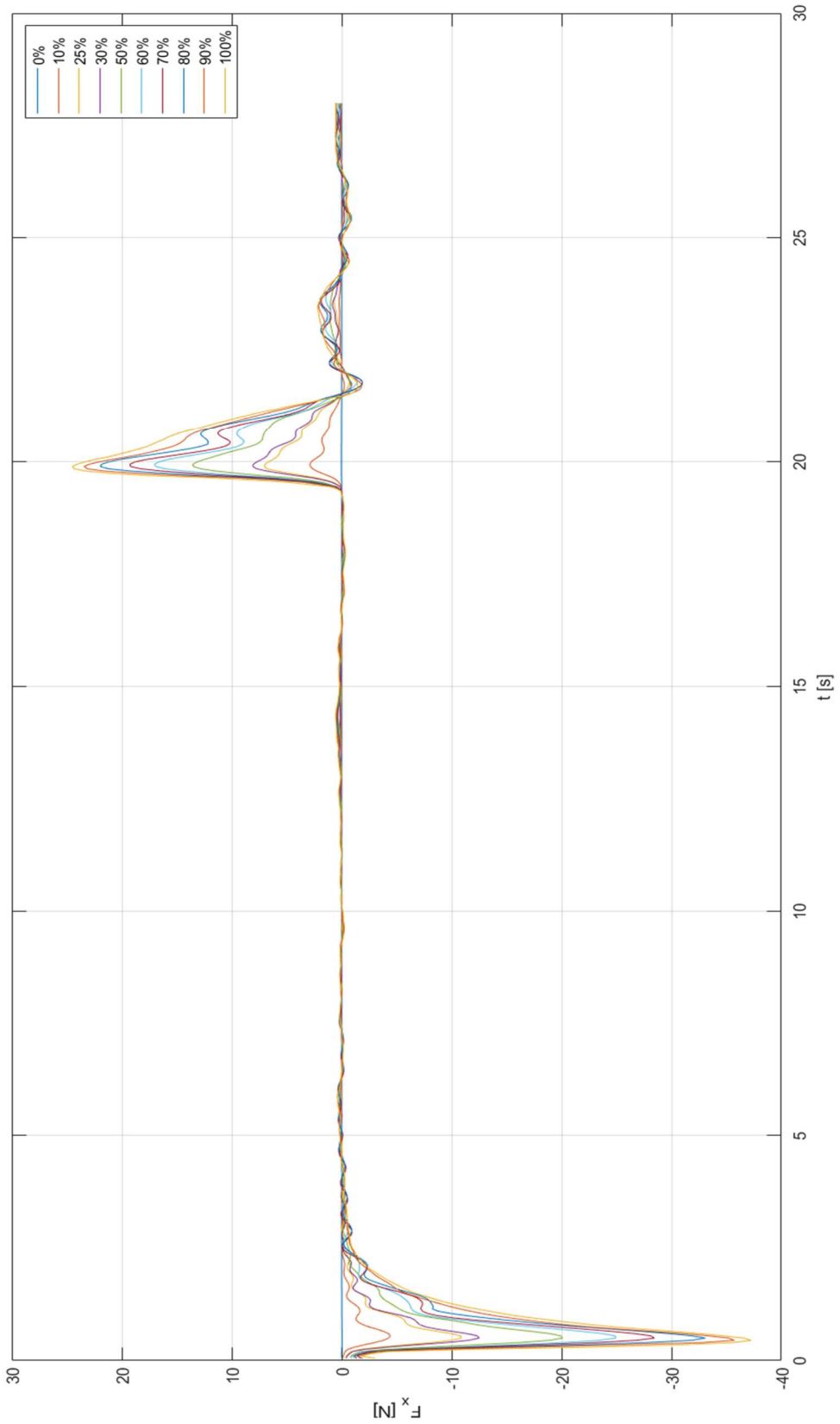


Figura 37 Profilo di forze completo in direzione del moto (lungo x) durante la missione al variare del riempimento

Conoscendo il profilo di accelerazioni in input e la massa sloshante è possibile scorporare la dinamica dall'effetto inerziale e osservare solo l'effetto sloshante. Si osserva, con dettaglio nei primi 10 secondi e per i livelli di rimpimento 0, 10%, 30%, 50%, 50% e 100% l'andamento

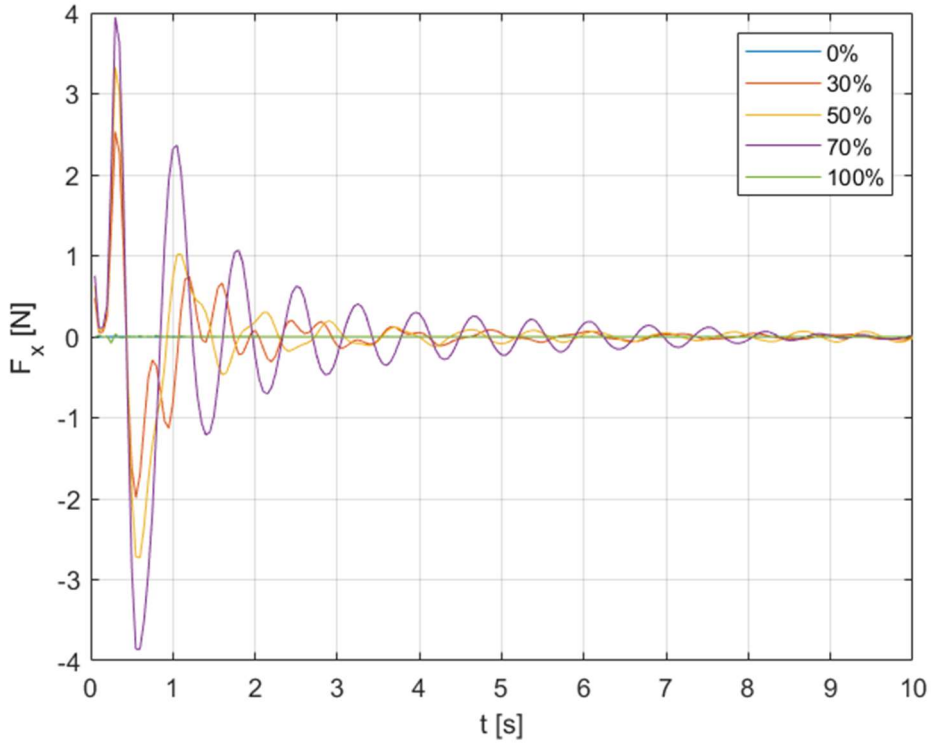


Figura 38 Forza di sloshing in direzione del moto per alcuni valori di riempimento campionati

Anche in questo caso il massimo si ha per un riempimento pari al 70%, mentre per 100% la funzione si appiattisce.

Tramite un tool per l'individuazione dei picchi si determina, come reciproco del periodo, la frequenza di oscillazione al variare del riempimento:

$\omega_{0\%}$	$\omega_{10\%}$	$\omega_{25\%}$	$\omega_{30\%}$	$\omega_{50\%}$
$3.4 \cdot 10^5 \ \frac{rad}{s}$	$9.026 \ \frac{rad}{s}$	$5.925 \ \frac{rad}{s}$	$4.012 \ \frac{rad}{s}$	$2.208 \ \frac{rad}{s}$

$\omega_{60\%}$	$\omega_{70\%}$	$\omega_{80\%}$	$\omega_{90\%}$	$\omega_{100\%}$
$1.7026\ \frac{rad}{s}$	$1.4358\ \frac{rad}{s}$	$1.4766\frac{rad}{s}$	$3.8316\frac{rad}{s}$	$44.23\frac{rad}{s}$

Tabella 24 Campionamento frequenze di sloshing nel caso di studio 4

Risulta immediato comprendere che per valori di riempimento prossimi allo 0%, la cui simulazione è stata fatta solo come termine asintotico per validare i risultati, che come atteso in tale condizione presentano ampiezze di oscillazione nulla, la frequenza di sloshing presenti valori fuori scala, associati alla propagazione del disturbo di pressione dell'aria, unica fase nel recipiente. Per valori crescenti si osserva il seguente andamento, espresso in scala logaritmica:

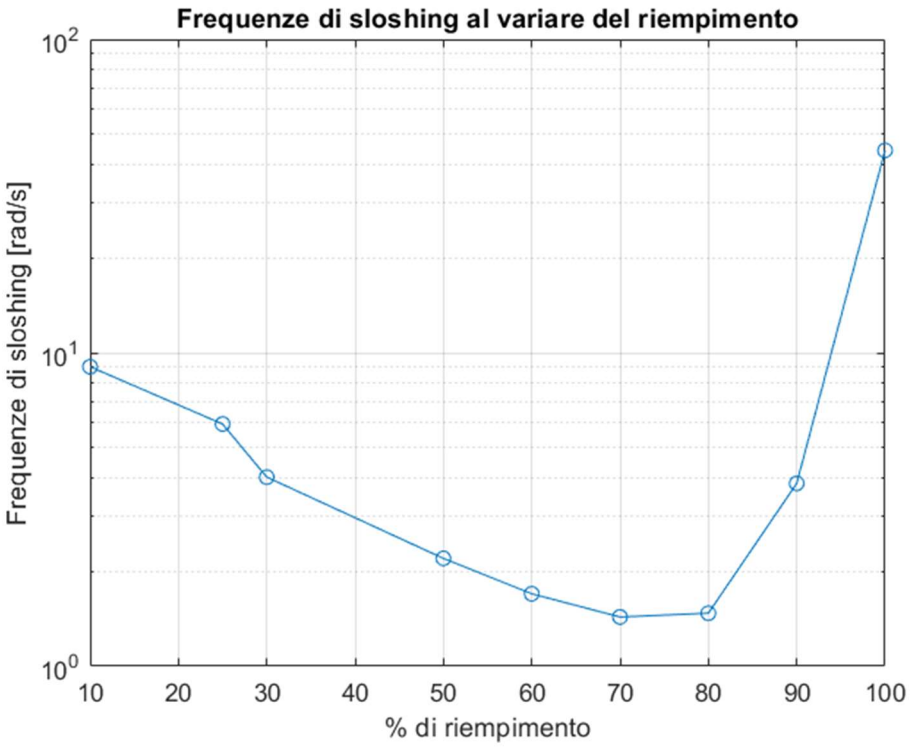


Figura 39 Curva interpolante rappresentativa dell'andamento delle frequenze di sloshing al variare del riempimento

Il minimo si ha per valori di riempimento del 70%, a cui è associato un massimo di ampiezza del fenomeno

5.5 Caso 5: Progettazione e aggiunta di deflettore tangenziale

Diversi studi, tra i quali *Validation of Slosh Model Parameters and Anti-Slosh Baffle Designs of Propellant Tanks by Using Lateral Slosh Testing* pubblicati dal *NASA Marshall Space Flight Center, Huntsville, Design and simulation of anti-sloshing baffles applied to detumbling payload propellant tasks* presentato al *The asian Aerospace and Astronautics Conference* del 2023 hanno passato al vaglio e validato l'efficacia progettuale di deflettori anti sciabordio posti in direzione tangenziale al pelo libero del liquido. In particolare, differentemente dalla progettazione di deflettori radiali, questi risultano particolarmente utili nello smorzamento in ampiezza trasversale della dinamica di sloshing tanto più efficacemente quanto più sono posti in prossimità del pelo libero del liquido, proponendo una legge di progettazione ideale per tutte le condizioni di riempimento in accordo con l'equazione di Miles. In figura viene mostrato un esempio.

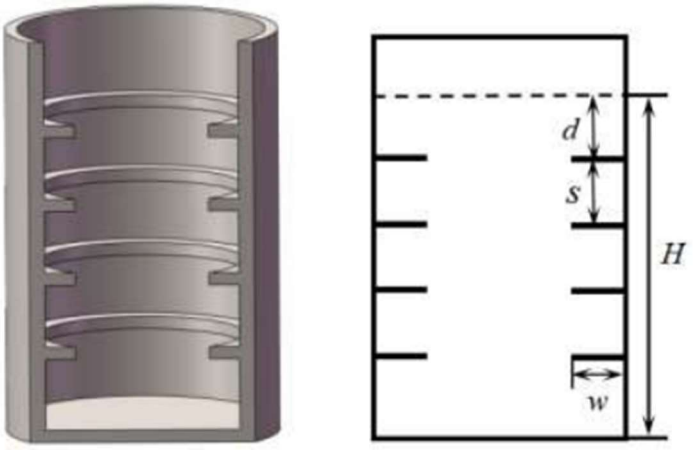


Figura 40 Esempio di smorzamento tramite deflettori tangenziali

Nel caso in esame, avendo già allo stato dell'arte il sistema a paratie radiali, si è presa in esame la possibilità di assemblare una sola ulteriore paratia tangenziale in prossimità del livello di riempimento più critico, che, come osservato, risulta essere pari al 70%. Inoltre, una diposizione circonferenziale come il caso studiato renderebbe complesso l'assemblaggio sul serbatoio già presente. Si è quindi optato per un modello a piastra piana forata tangenziale con una concavità verso il basso dedicata al convogliamento del liquido, per evitare fenomeni di incameramento di porzioni di liquido in fase di riempimento o seguenti ad uno sloshing stesso.

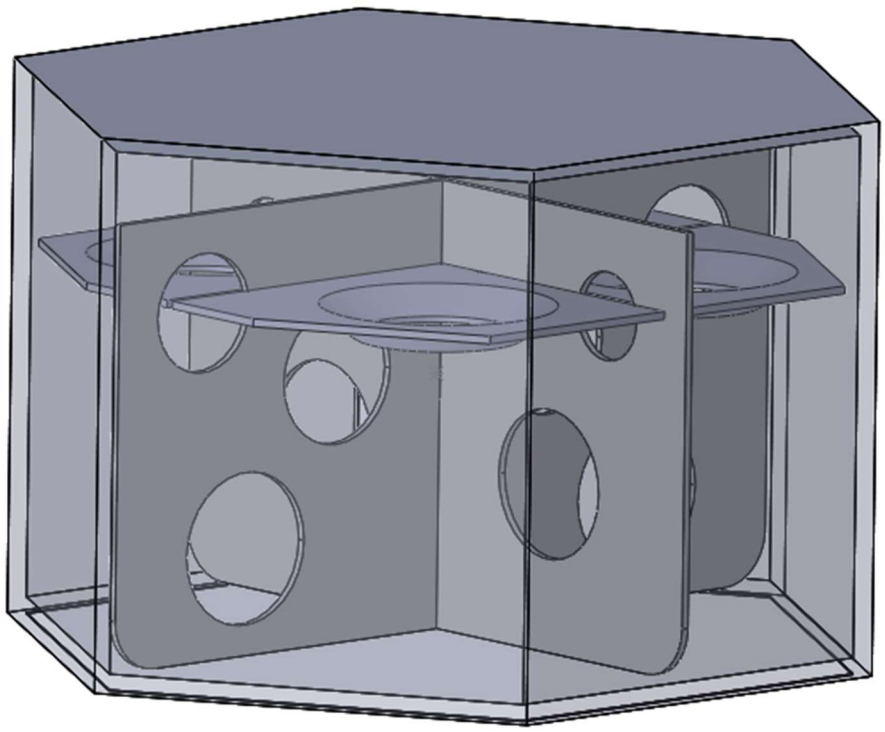


Figura 41 Serbatoio in trasparenza con deflettore tangenziale in compinazione ai già presenti deflettori radiali posto al 70% dell'altezza

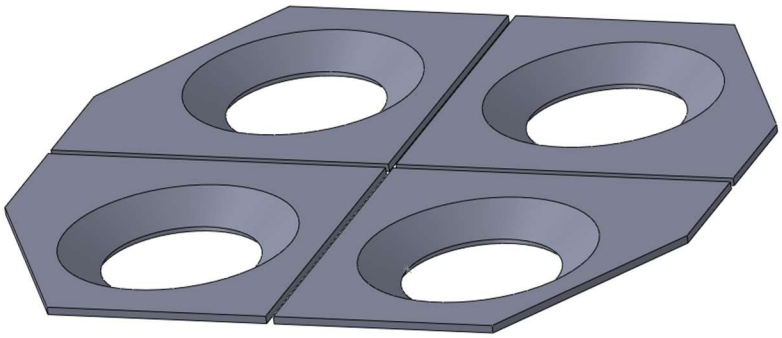


Figura 42 Dettaglio del deflettore tangenziale. Si osserva la presenza di fori simmetrici con concavità verso il vasso per agevolare l'invaso

Il raggio del foro è stato scelto come compromesso tra la massimizzazione del fattore di smorzamento di Miles

$$\zeta = \frac{15\left(\frac{4}{3}\pi\right)^2 aAe{-}4^{\ -4.601d/R}\sqrt{\mu w}}{2\sqrt{\pi}\left(\frac{m_s}{\rho}\right)\Gamma^2}$$

e i vincoli geometrici dettati dalla geometria estremamente non uniforme nelle celle, identificando w come la minima distanza tra la circonferenza e l'esterno del poligono.

Vengono mostrati di seguito i risultati comparati con la simulazione, di pari riempimento, con i soli deflettori radiali, per la sola dinamica di sloshing scorporata del moto di corpo rigido. L'effetto smorzante si misura come tempo di assestamento che viene dimezzato.

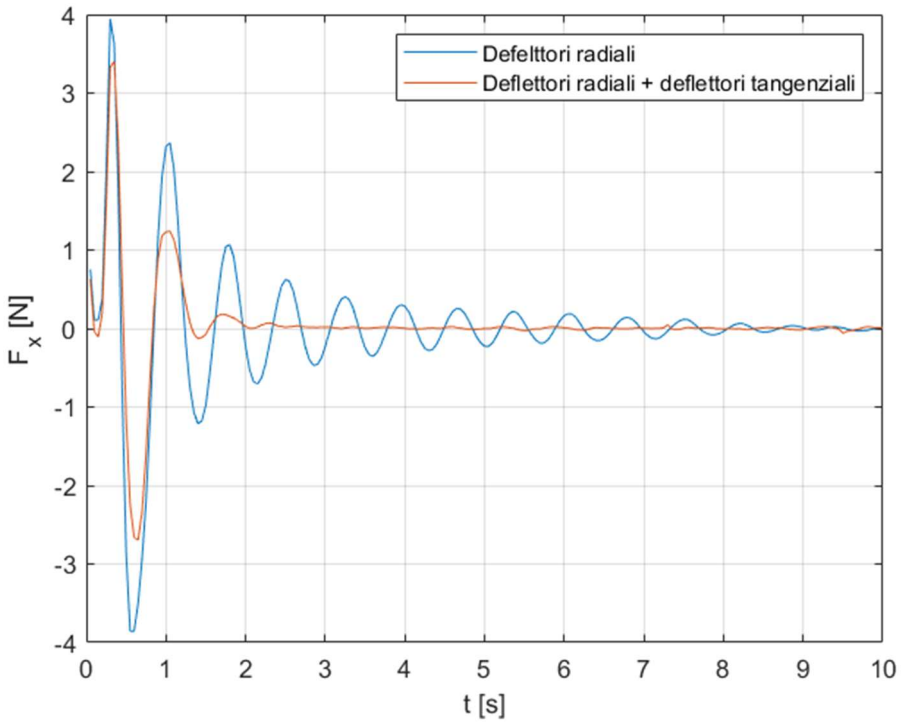


Figura 43 Risultato dell'effetto di smorzamento per il riempimento target confrontato con il caso con i soli deflettori radiali

5.6 Caso 6: Doppio deflettore tangenziale

Non essendo possibile montare deflettori tangenziali per ogni livello di riempimento, pena lo spazio residuo utile al liquido e le difficoltà di aspirazione, appurata la validità del deflettore posto al 70% viene valutata una configurazione con doppio deflettore posto al 30 e al 70% di rimpimento, in modo da risultare simmetrico rispetto al centro serbatoio.

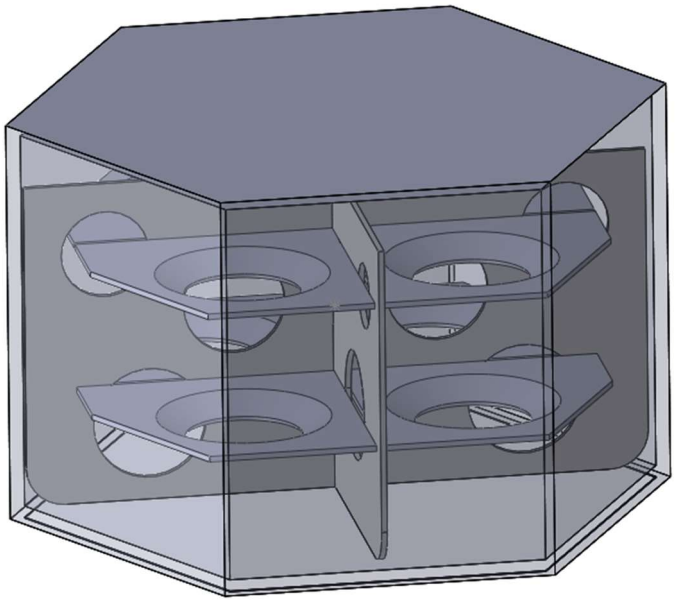


Figura 44 Serbatoio in trasparenza con combinazione dei due deflettori radiali presenti allo stato di fatto e dei due deflettori tangenziali posti al 30 e al 70% dell'altezza

Di seguito l'andamento del fenomeno di sloshing al variare del riempimento per valori nell'intorno dei deflettori tangenziali:

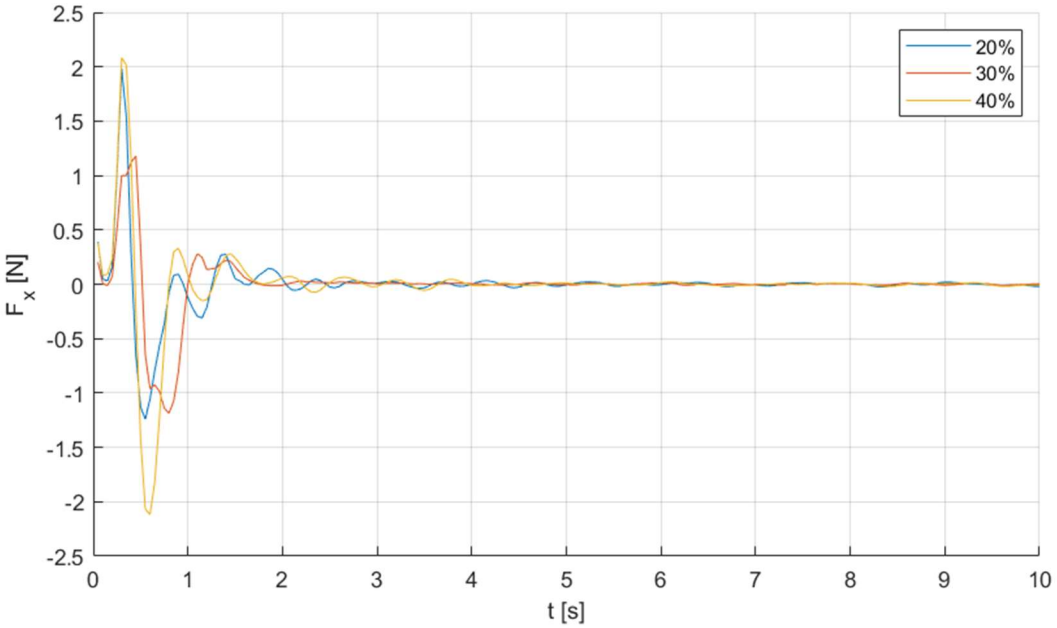


Figura 46 Forze lungo x dovute alla dinamica di sloshing nel caso di riempimento del 30% (targhet di progetto del secondo deflettore tangenziale) e di riempimenti prossimi (20 e 40%)

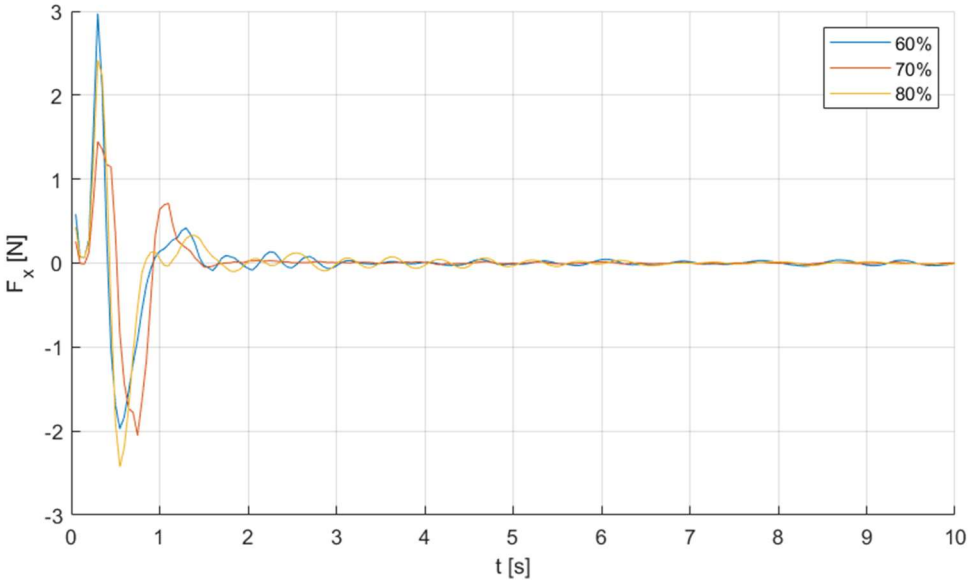


Figura 45 Forze lungo x dovute alla dinamica di sloshing nel caso di riempimento del 70% target di progetto del primo deflettore tangenziale) e di riempimenti adiacenti (60 e 80%)

Si osserva immediatamente come l'aggiunta di un secondo deflettore riduca il picco caratteristico della prima oscillazione per il riempimento del 70%, che raggiunge un massimo in valore assoluto pari a 2.05 N, evidenziando una riduzione del 41% del picco di overshoot rispetto al caso di singolo deflettore tangenziale e del 47% rispetto alla configurazione con soli deflettori radiali. Considerando anche l'effetto inerziale del moto di corpo rigido il picco di overshoot passa da 26.73N a 24.32N, con una riduzione del 9.05%.

L'oscillazione sull'asse verticale in presenza degli smorzatori tangenziali risulta limitata in un volume pieno per i sottodomini che determinano un effetto di sloshing disuniforme tra le 3 camere in cui i deflettori tangenziali dividono il volume, comportando un effetto di cancellazione di onde per altro già parzialmente ipotizzabile osservando la tendenza delle funzioni per i due valori di riempimento nella configurazione con soli deflettori radiali. Questo si nota proprio nei flessi che caratterizzano la configurazione con tutti i deflettori, per i due valori di riempimento target dello smorzamento. Per i valori al contorno l'efficacia è ridotta, ma è possibile apprezzare l'effetto della modifica progettuale:

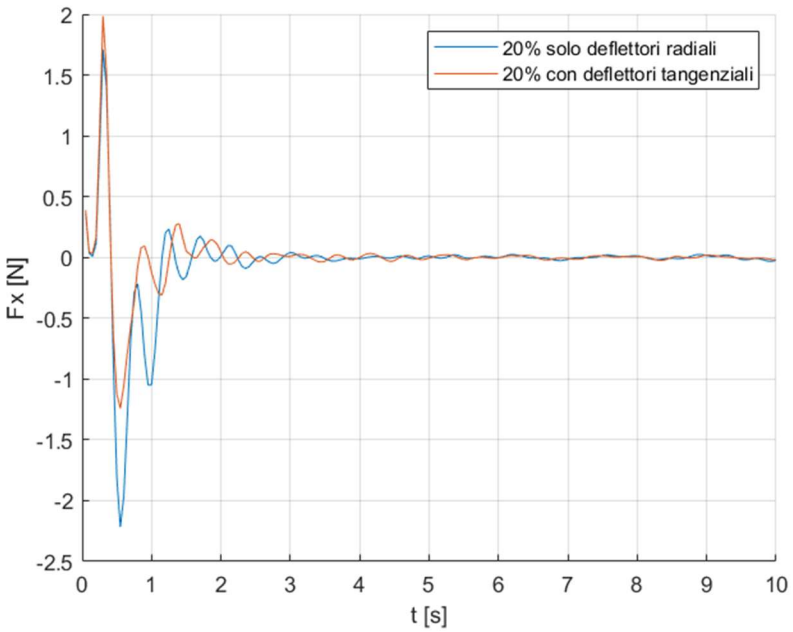


Figura 47 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 20%

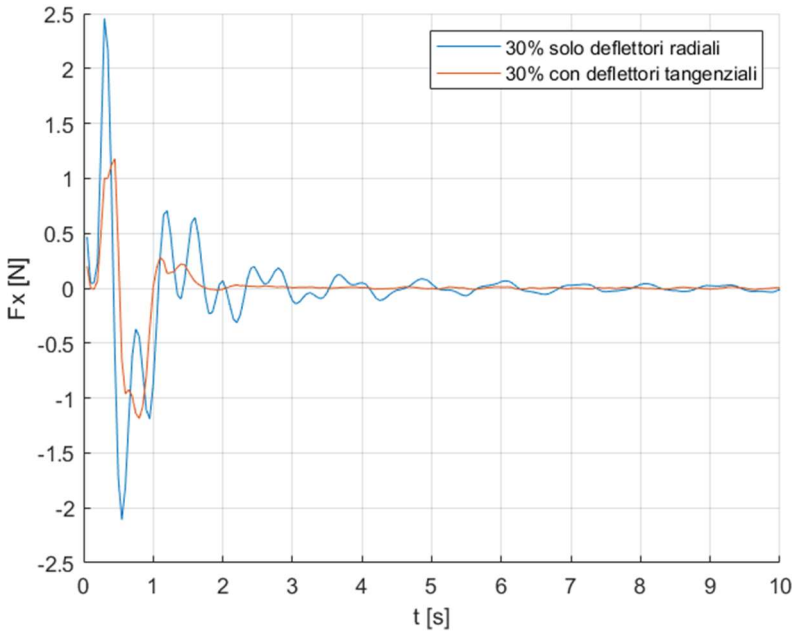


Figura 48 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 30%

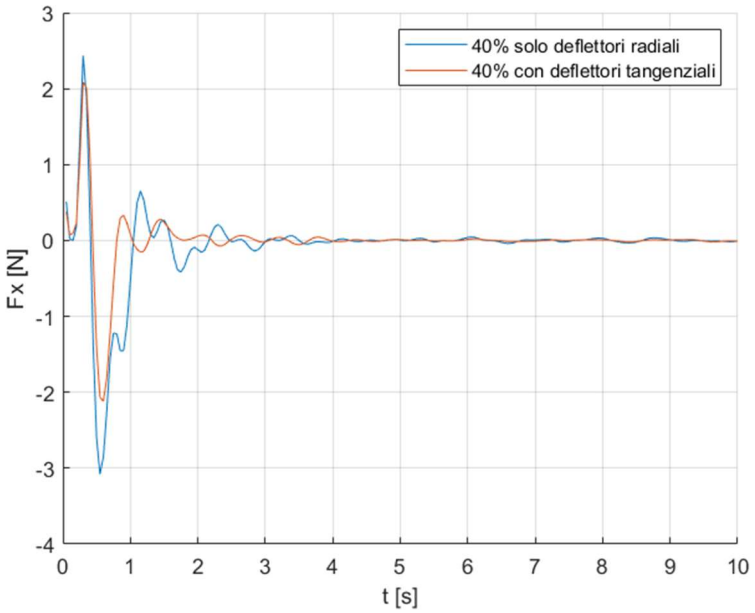


Figura 48 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 40%

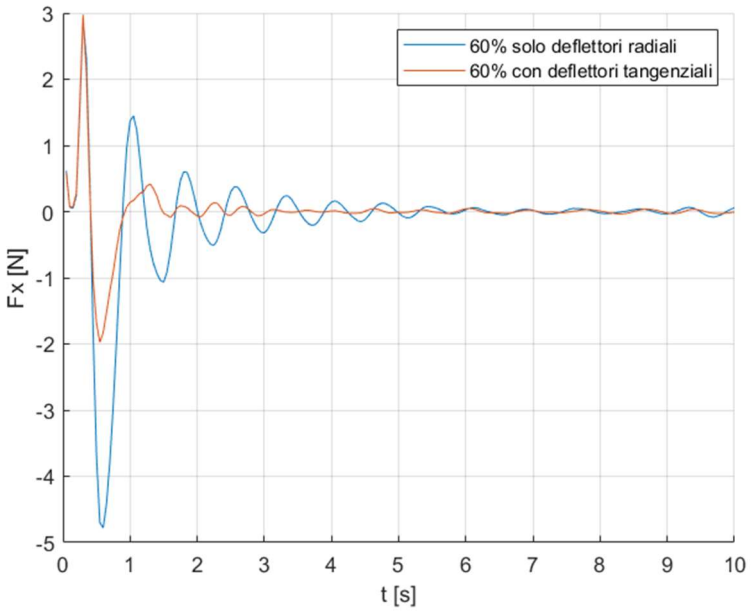


Figura 49 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 60%

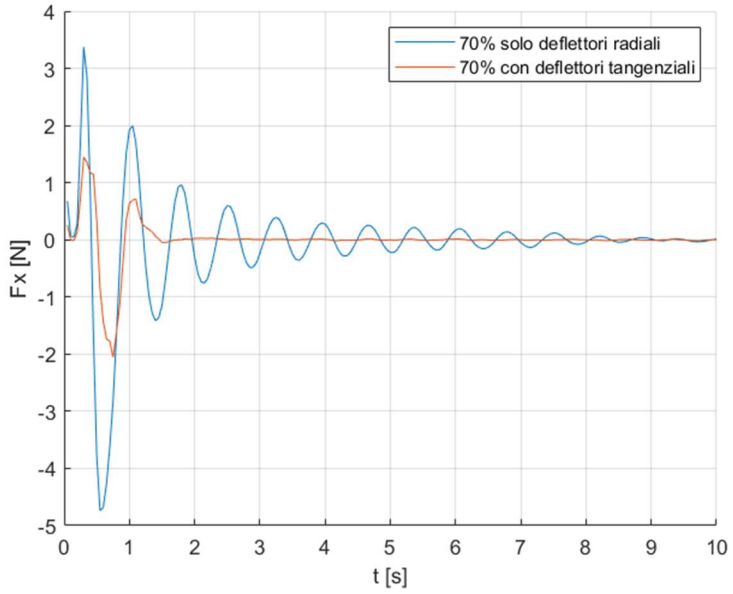


Figura 50 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 70%

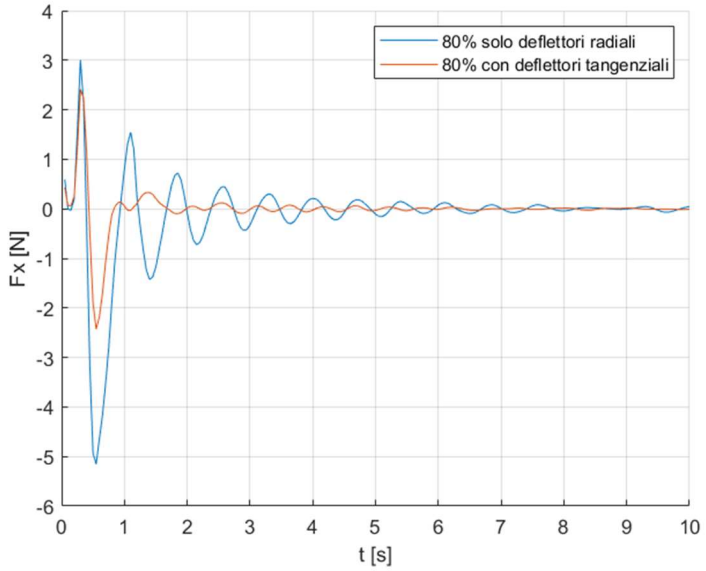


Figura 49 Confronto tra la forza di sloshing in direzione del moto con e senza modifica progettuale proposta per un valore di riempimento pari al 80%

Per condizioni di riempimento immediatamente inferiori a quelle target di progettazione si osserva una leggera amplificazione sulla prima oscillazione di sloshing da 1.71N a 1.98 N, con un fattore di amplificazione 1.15 per un riempimento del 20% e da 2.91N a 2.97 N, con fattore di amplificazione 1.02 per la condizione di riempimento del 60%. In quest'ultimo caso la prima oscillazione rappresenta il picco assoluto del fenomeno, tuttavia l'apporto è estremamente ridotto. Al contrario per tutte le condizioni il fattore di smorzamento risulta sempre migliore con l'aggiunta dei deflettori tangenziali.

Capitolo 6

Conclusioni

6.1 Considerazioni finali e limitazioni del modello di ricerca

Lo studio condotto ha permesso di rispondere in modo articolato alle domande di ricerca iniziali, fornendo una comprensione dettagliata dell'impatto dello sloshing sulla dinamica del serbatoio utilizzato per missioni di agricoltura di precisione.

Impatto dello sloshing sulla dinamica del serbatoio:

È emerso chiaramente che l'effetto sloshing influisce significativamente sulla dinamica del serbatoio non smorzato, con picchi di overshoot fino al 38% rispetto alla dinamica inerziale per il profilo di missione fissato. Le oscillazioni del liquido all'interno del serbatoio generano forze dinamiche che possono compromettere la stabilità e la precisione della missione di spruzzatura.

Effetto dei deflettori e dell'orientamento sullo smorzamento:

L'assemblaggio dei deflettori all'interno del serbatoio ha mostrato un impatto positivo sullo smorzamento delle oscillazioni del liquido. I deflettori agiscono riducendo l'ampiezza delle onde e migliorando la stabilità del sistema. Questo effetto smorzante è stato più evidente nei casi di missioni con direzione di volo intersecante gli spigoli del serbatoio, dove le oscillazioni risultano maggiormente mitigate dalla presenza dei deflettori.

Impatto della variazione di riempimento sullo sloshing: La variazione del livello di riempimento del serbatoio durante la missione ha un impatto significativo sullo sloshing. È stato osservato che livelli di riempimento intermedi (intorno al 70%) tendono a massimizzare l'ampiezza delle oscillazioni, con picchi del 16% rispetto alla componente inerziale, mentre riempimenti molto bassi o molto alti riducono questo fenomeno. L'analisi ha altresì permesso di campionare, per un dato profilo di missione, l'andamento delle frequenze della dinamica di sloshing costruendo una funzione utile ai processi di modellamento della traiettoria tramite input shaping.

Possibilità di modifiche progettuali minori non distruttive:

A seguito delle evidenze circa le condizioni critiche, e sulla base di alcuni risultati di ricerca citati, sono state identificate alcune modifiche progettuali non distruttive che possono migliorare l'efficienza dello smorzamento. In particolare, l'aggiunta di deflettori tangenziali, anche in configurazioni non ottimali, ha mostrato un miglioramento significativo nel controllo delle oscillazioni del liquido in termini di smorzamento. Questa soluzione rappresenta un intervento relativamente semplice e a basso costo che può essere implementato senza necessità di modifiche strutturali importanti.

Occorre tenere conto di alcune limitazioni del modello. Non è stato possibile includere l'effetto dell'oscillazione angolare del serbatoio, data la complessità nel valutare il centro di rotazione per ogni configurazione. Questo aspetto potrebbe essere esplorato in studi futuri per ottenere una visione ancora più completa del fenomeno. Inoltre, la simulazione non ha considerato il fenomeno di spillamento del liquido in pressione e la conseguente variazione del riempimento durante la missione. Al contrario, è stato eseguito un set di simulazioni per diversi valori di riempimento

costanti durante ciascuna simulazione, il che rappresenta una semplificazione della realtà operativa, ma costituisce un'importante base dati. Come ultima semplificazione di cui tenere conto, è bene osservare che il modello di turbolenza scelto (RANS) presenta una validità solo in condizioni di oscillazione in campo lineare, senza distacchi di fluido. Per modellare con più precisione il reale andamento delle fasi sarebbe più indicato un modello Larege Eddy Simulation (LES) o metodi diretti come Direct Numerical Simulation (DNS). Questi ultimi richiedono risorse computazionali, dovute ad un necessario infittimento della mesh per la convergenza, non compatibili con l'analisi parametrica fatta.

6.2 Sviluppi futuri

L'implementazione di modelli predittivi basati sull'intelligenza artificiale (AI) rappresenta un promettente sviluppo futuro per migliorare la gestione e l'ottimizzazione delle missioni di agricoltura di precisione, contribuendo ad una gestione più sostenibile delle risorse agricole. A partire dai dati generati attraverso numerose simulazioni eseguite con il modello OpenFOAM, è possibile costruire una solida base di dati che serve come fondamento per l'addestramento di algoritmi di machine learning (ML). Uno degli obiettivi di questo elaborato è stato proprio quello di descrivere un manuale d'uso del modello e delle parametrizzazioni possibili per creare sistemi di simulazione in automazione. Questi modelli predittivi possono fornire previsioni precise sugli effetti dello sloshing in diverse condizioni operative e configurazioni di serbatoio. ll primo passo cruciale nello sviluppo di un modello predittivo basato su AI consiste nella raccolta sistematica dei dati. Le simulazioni

devono essere progettate per coprire una vasta gamma di scenari operativi, includendo vari livelli di riempimento, configurazioni di deflettori e profili di missione. Ogni simulazione dovrebbe registrare dettagliatamente i parametri dinamici come le forze e momenti di sloshing, le oscillazioni del liquido, e l'impatto sulla stabilità del drone. È essenziale che i dati raccolti siano etichettati correttamente e memorizzati in un formato strutturato per facilitare il successivo preprocessing. Questo processo richiede la pulizia dei dati, l'eliminazione delle anomalie, e la normalizzazione dei valori per assicurare che tutte le variabili siano comparabili. Inoltre, potrebbe essere necessario eseguire tecniche di feature engineering per estrarre caratteristiche rilevanti che possano migliorare le capacità predittive del modello.

Una volta costruita una base dati robusta, il passo successivo è lo sviluppo e l'addestramento del modello predittivo. Per questo scopo possono essere considerati algoritmi di machine learning come le reti neurali artificiali (ANN), le macchine a vettori di supporto (SVM), e i modelli di regressione avanzata. Le reti neurali, in particolare, offrono vantaggi significativi grazie alla loro capacità di apprendere rappresentazioni complesse e non lineari dai dati.

Durante la fase di addestramento, il modello viene alimentato con il dataset suddiviso in set di addestramento e di validazione. La tecnica di cross-validation è consigliata per ottimizzare i parametri del modello e prevenire l'overfitting. È cruciale monitorare le performance del modello attraverso metriche appropriate come l'errore quadratico medio (MSE), per assicurare che il modello stia effettivamente imparando a predire gli effetti dello sloshing con alta accuratezza.

Dopo l'addestramento, il modello deve essere rigorosamente validato utilizzando un set di dati indipendente non utilizzato durante

l'addestramento. Questa fase è essenziale per valutare la generalizzabilità del modello a dati nuovi e non visti. Per questo scopo vengono usate tecniche denominate k-fold cross-validation

Una volta validato, il modello predittivo può essere integrato nel processo decisionale operativo. Ad esempio, può essere utilizzato per fornire previsioni in tempo reale sulle condizioni di sloshing durante una missione di spruzzatura, integrando le già esistenti funzioni di controllo implementate dal sistema di stabilizzazione. Inoltre, il modello può essere utilizzato per ottimizzare la progettazione dei serbatoi e la disposizione dei deflettori, attraverso simulazioni virtuali che esplorano una vasta gamma di configurazioni possibili.

L'implementazione di un modello predittivo basato su AI non è priva di sfide. Una delle principali difficoltà riguarda la qualità e la quantità dei dati disponibili per l'addestramento. Dati insufficienti o di bassa qualità possono compromettere le prestazioni del modello. Per mitigare questo rischio, è fondamentale investire in una raccolta dati accurata e, se necessario, utilizzare tecniche di data augmentation per ampliare il dataset.

Un'altra sfida riguarda la complessità computazionale associata all'addestramento di modelli AI complessi. Questo può richiedere risorse computazionali significative, specialmente per le reti neurali profonde.

Riferimenti bibliografici

[1] H. N. Abramson, W. H. Chu, G. E. Ransleben Jr., Representation of Fuel Sloshing in Cylindrical Tanks by and Equivalent Mechanical Model, ARS J., 1961.

[2] J. D. Anderson, Computational Fluid Dynamics: The Basics with Applications, McGraw-Hill, 1995.

[3] Y. Chen and M.-A. Xue, "Numerical Simulation of Liquid Sloshing with Different Filling Levels Using OpenFOAM and Experimental Validation," Water, vol. 10, no. 12, p. 1752, Nov. 2018.

[4] J. H. Ferziger and M. Peric, "Computational Methods for Fluid Dynamics," Springer, vol. 3, pp. 1-45, 2002.

[5] B. Godderidge, M. Tan, C. Earl, S. Turnock, Grid Resolution for the Simulation of Sloshing Using CFD, School of Engineering Sciences, University of Southampton, Southampton, UK.

[6] Hoskins, E.G., and Jacobsen, L.S., "Earthquake Response of a Rigid Frame Building by the Panel Analogue Method," Bulletin of the Seismological Society of America, vol. 24, no. 4, pp. 321-330, 1934.

[7] Jacobsen, L.S., "Impulsive Hydrodynamics of Fluid Inside a Cylindrical Tank and of Fluid Surrounding a Cylindrical Pier," Bulletin of the Seismological Society of America, vol. 39, no. 3, pp. 189-204, 1949.

[8] S. B. Pope, "Turbulent Flows," Cambridge University Press, vol. 5, no. 2, pp. 100-145, 2000.

[9] H. Rusche, "Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions," in Proc. of the 8th Int. Conf. on CFD, London, 2002, pp. 89-97.

[10] H. K. Versteeg and W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, Pearson Education, 2007.

[11] Westergaard, H.M., "Water Pressures on Dams During Earthquakes," Transactions of the American Society of Civil Engineers, vol. 98, no. 1839, pp. 418-433, 1933.

[12] OpenFOAM Foundation, "OpenFOAM User Guide," Available: www.openfoam.com/documentation/user-guide.

[13] NASA, "Computational Fluid Dynamics," Available: www.grc.nasa.gov.

[14] cfMesh, Dr. Franjo Juretić, M. Eng., Assist. Prof. Managing "User Guide", Available: www.cfmesh.com/cfmesh