POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

Validazione di un codice ai volumi finiti con metodo Immersed Boundary per flussi compressibili e parziale implementazione della parte aerotermochimica

Relatore:
prof. Domenic D'Ambrosio

Candidato:
Andrea Nalin

Anno Accademico 2018–2019

Sommario

Un codice ai volumi finiti con metodo *immersed boundary*, sviluppato dalla OPTIMAD Engineering srl, viene validato in regime ipersonico riproducendo un esperimento condotto dal centro di ricerca aerospaziale ONERA in cui è presente un'interazione tra urti di tipo Edney IV. I risultati ottenuti combaciano sufficientemente bene con i dati sperimentali e perfettamente con quelli ottenuti da altri codici ai volumi finiti. In seguito viene svolto uno studio preliminare atto a implementare gli effetti aerotermochimici presenti in caso di flusso ipersonico e viene scritto un codice per il calcolo dei termini sorgente delle equazioni di conservazione aggiuntive. L'equazione di conservazione della massa per l'i-esima specie viene integrata nel tempo e i risultati confermano la correttezza dei termini di produzione. Particolare attenzione viene posta alla spiegazione della metodologia *immersed boundary* e viene dato un esempio di implementazione di quest'ultima nel caso unidimensionale del tubo d'urto.

Indice

1 Introduzione

Nell'ambito della fluidodinamica un corpo immerso in un fluido viene usualmente considerato in regime ipersonico se la sua velocità relativa con il fluido è maggiore di Mach 5. Questo regime si differenzia da quello supersonico perché insorgono vari fenomeni che vanno a modificare completamente il comportamento del velivolo e le caratteristiche che esso deve avere. Gli elevati numeri di Mach portano l'urto a essere molto vicino al corpo, con la possibilità di avere interazione tra l'urto stesso e lo strato limite. Quest'ultimo è caratterizzato da valori molto alti di temperatura, che causano l'aumento dello spessore dello strato limite, con conseguente cambiamento della forma del corpo vista dal fluido inviscido esterno. Le temperature molto elevate, inoltre, causano l'eccitazione dei gradi di libertà vibrazionali del gas e la dissociazione delle sue molecole [7].

La distinzione tra regime supersonico e ipersonico è quindi fondamentale; tutti questi fenomeni rendono necessario validare un codice, studiato per l'analisi di flussi comprimibili, in questo particolare regime. Ai fini della validazione sono spesso utilizzati casi test in cui si abbia interazione tra due urti; infatti, questi fenomeni causano picchi locali molto elevati di pressione e temperatura che potrebbero avere effetti distruttivi e addirittura catastrofici se non considerati in fase di progetto [8]. Nel caso poi in cui si raggiungano temperature molto elevate, ovvero nella maggior parte dei casi reali e non sperimentali, risulta necessario utilizzare un codice che tenga conto delle reazioni chimiche presenti nel gas e dei suoi gradi di libertà vibrazionali, onde evitare di ottenere valori di temperatura dopo un urto troppo elevati e irrealistici considerando il suo $\gamma = c_p/c_v$ costante.

Questa tesi è stata svolta in collaborazione con la OPTIMAD Engineering srl e il suo obiettivo è la validazione di un software CFD, sviluppato dalla stessa azienda, in regime ipersonico; contemporaneamente è stato svolto uno studio preliminare atto a implementare la parte aerotermochimica in questo codice.

Il software utilizzato, di nome **ImmerFlow**, permette l'analisi di flussi in regime comprimibile e incomprimibile; la sua principale peculiarità è l'utilizzo di un metodo appartenente alla famiglia degli *immersed boundary*. Questi metodi, che vengono descritti minuziosamente nel Capitolo 2, consentono di rappresentare un corpo immerso in un fluido utilizzando una griglia che non sia conforme al corpo stesso. In questo modo è possibile usare semplici griglie cartesiane o più elaborate griglie *octree* come fatto con **ImmerFlow**.

Data l'importanza e la centralità di questo approccio nella rappresentazione del

corpo all'interno del software utilizzato, nel Capitolo 3 viene dato un esempio di applicazione di questi metodi al caso unidimensionale del tubo d'urto. In particolare, vengono utilizzati due metodi ispirati al *ghost-cell method* (Figura 1.1) per imporre la presenza di una parete che rifletta un urto all'interno del dominio unidimensionale; i risultati vengono poi analizzati e viene studiato l'errore commesso nell'imporre la presenza della parete all'interno di una cella e non in corrispondenza dell'interfaccia tra una cella e l'altra.

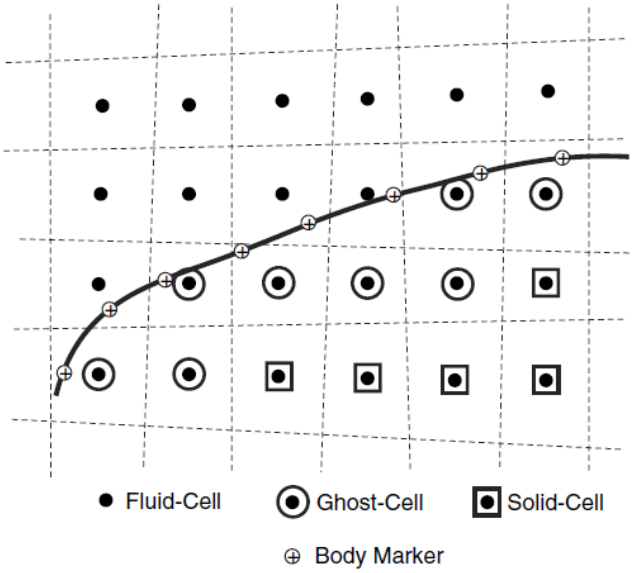


Figura 1.1: Suddivisione del dominio secondo il *ghost-cell method*. Le celle all'esterno del corpo sono considerate facenti parte del fluido e sono risolte normalmente; quelle interne al corpo sono suddivise in *ghost cells*, utilizzate per imporre la condizione al contorno sulla superficie del corpo, e nelle celle appartenenti al solido, la cui soluzione banale è disaccoppiata dal resto del problema. [15]

Il Capitolo 4 è il cuore di questa tesi e in esso viene inizialmente presentato **ImmerFlow**, per poi passare al caso studio utilizzato per la validazione del software in regime ipersonico. Un esperimento condotto al centro di ricerca aerospaziale ONERA, nel wind-tunnel a bassa densità R5Ch [33], viene riprodotto numericamente e i risultati ottenuti sono confrontati con quelli sperimentali e con quelli numerici ottenuti da altri software per l'analisi di flussi ipersonici [9]. Particolare attenzione viene posta sulla regione di interazione tra urti di tipo Edney IV (Figura 1.2).

Nel Capitolo 5, infine, vengono esposte le principali caratteristiche di un flusso ipersonico, concentrandosi sui fenomeni dovuti ai grandi incrementi di temperatura dopo un urto e in prossimità del corpo. L'eccitazione dei gradi di libertà vibrazionali all'interno delle molecole e la loro dissociazione rendono necessario considerare

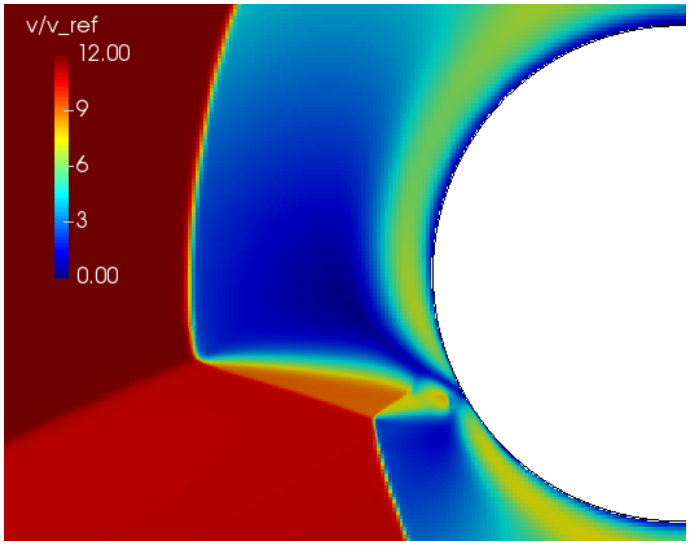


Figura 1.2: Andamento della velocità adimensionalizzata nella regione di interazione tra i due urti di tipo Edney IV.

il fluido in non equilibrio chimico e vibrazionale. I calori specifici c_p e c_v sono, di conseguenza, non più costanti, ma funzioni della temperatura, della pressione e della composizione istantanea del fluido. I termini sorgenti necessari a scrivere le equazioni di conservazione aggiuntive sono calcolati attraverso l'utilizzo di un programma scritto in `C++`, utilizzando in parte dei metodi del software open-source **Cantera**. I termini di produzione delle equazioni di conservazione di massa dell'i-esima specie sono poi validati confrontando i risultati provenienti da un calcolo dell'equilibrio e dall'integrazione temporale delle equazioni stesse.

2 Metodi Immersed Boundary

I metodi *Immersed Boundary* sono tutti quei metodi numerici che permettono di simulare la presenza di un corpo all'interno di un flusso viscoso utilizzando una griglia computazionale che non sia conforme al corpo immerso [27].

Un approccio classico *body conformal* nella discretizzazione di un corpo solido immerso in un fluido consiste nella rappresentazione della superficie del corpo Γ_b attraverso una mesh superficiale e una successiva discretizzazione del dominio esterno, Ω_f in Figura 2.1, attraverso una mesh volumica strutturata o non. Invece, come si può visualizzare nella parte destra della Figura 2.1, un approccio *nonbody conformal* rappresenta il dominio di calcolo attraverso una semplice mesh cartesiana non tenente conto della presenza del corpo, onde poi dover imporre successivamente la presenza del *boundary*, chiamato appunto *immersed boundary*, andando a modificare localmente le equazioni di governo.

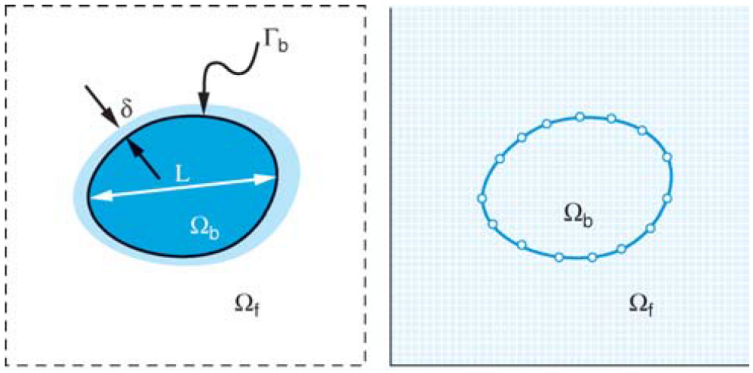


Figura 2.1: Confronto di una suddivisione del dominio di calcolo *body conformal* a sinistra e una *nonbody conformal* tipica di un metodo *immersed boundary* a destra. [27]

I principali vantaggi derivanti dall'utilizzo di un metodo simile risiedono fondamentalmente nella facilità di creazione di una griglia cartesiana, indipendente dalla complessità del corpo rappresentato, e nella facilità di rappresentazione di corpi in movimento. Le maggiori difficoltà invece risiedono nell'imposizione delle condizioni al contorno in prossimità del corpo, e nel fatto che all'aumentare del numero di Reynolds la taglia di una griglia cartesiana debba crescere molto più velocemente di una conforme al corpo.

2.1 Imposizione delle condizioni al contorno

Mittal e Iaccarino nella loro review [27] distinguono due grandi categorie di metodi *immersed boundary*: *continuos forcing approach* e *discrete forcing approach*.

Si consideri un dominio simile a quello rappresentato in Figura 2.1, il comportamento del fluido può essere descritto dal seguente sistema:

$$\begin{cases} N(\boldsymbol{U}) = 0 & \text{su } \Omega_f \\ \boldsymbol{U} = \boldsymbol{U}_\Gamma & \text{su } \Gamma_b \end{cases} \tag{2.1}$$

dove $\boldsymbol{U} = (\boldsymbol{u}, p)$ è il vettore delle incognite formato dalle componenti della velocità e dalla pressione e N è un operatore rappresentante le equazioni di Navier-Stokes. Un metodo *immersed boundary* è in grado di imporre la presenza del corpo attraverso una modifica della prima equazione del sistema 2.1, ovvero delle equazioni di governo; in particolare viene aggiunto un termine sorgente, chiamato anche *forcing function*. Il modo di imporre la presenza di questo termine sorgente aggiuntivo determina la distinzione nei due metodi di cui sopra.

Nel cosiddetto *continuos forcing approach* il termine sorgente $\boldsymbol{f}_b$ viene applicato alle equazioni di governo a livello continuo, ottenendo il sistema di equazioni:

$$N(\boldsymbol{U}) = \boldsymbol{f}_b \tag{2.2}$$

che viene poi discretizzato ottenendo il sistema:

$$[N]\{\boldsymbol{U}\} = \{\boldsymbol{f}_b\} \tag{2.3}$$

Nel *discrete forcing approach*, invece, le equazioni di governo vengono prima discretizzate sulla griglia di calcolo:

$$[N]\{\boldsymbol{U}\} = 0 \tag{2.4}$$

per poi essere modificate successivamente, utilizzando un operatore discreto modificato $[N']$ e un termine associato all'imposizione delle condizioni al contorno sul corpo $\{\boldsymbol{r}\}$, ottenendo:

$$[N']\{\boldsymbol{U}\} = \{\boldsymbol{r}\} \tag{2.5}$$

L'Equazione 2.5 può essere riscritta utilizzando $\{\boldsymbol{f}_b'\} = \{\boldsymbol{r}\} + [N]\{\boldsymbol{U}\} - [N']\{\boldsymbol{U}\}$:

$$[N]\{\boldsymbol{U}\} = \{\boldsymbol{f}_b'\} \tag{2.6}$$

2.1.1 Continuos forcing approach

Come scritto sopra, i metodi che si basano sul *continuos forcing approach* modificano le equazioni di governo attraverso l'uso di un termine sorgente prima che

esse vengano discretizzate. Per questo motivo essi sono indipendenti dal tipo di discretizzazione utilizzato.

Questi sono utilizzati soprattutto per problemi in cui sia necessario rappresentare delle pareti elastiche e furono effettivamente i primi metodi *immersed boundary*, introdotti da Peskin nel 1972. Queste pareti elastiche vengono rappresentate da delle fibre elastiche il cui effetto sul fluido viene modellato come una forza applicata a quest'ultimo e legata alla deformazione delle fibre dalla legge di Hooke. Nel caso in cui si rappresenti un corpo con pareti rigide la filosofia utilizzata è lo stessa, ma la rigidezza delle fibre elastiche viene assunta molto elevata.

Proprio questo è il limite più grande dei metodi *continuos forcing approach*. Infatti la rappresentazione di corpi rigidi attraverso un elevato valore di rigidezza della fibre elastiche appare quantomeno difficoltosa, introducendo problemi dal punto di visto dell'accuratezza numerica e della stabilità, e non permettendo di ottenere una rappresentazione netta e *sharp* dell'*immersed boundary* considerato.

2.1.2 Discrete forcing approach

I metodi che si basano sul *discrete forcing approach* introducono il termine sorgente dopo che le equazioni di governo sono state discretizzate. In questo modo ogni metodo è strettamente dipendente dalla discretizzazione utilizzata, si ha però la possibilità di avere maggiore controllo sull'accuratezza e sulla stabilità della soluzione.

Questi metodo vengono suddivisi a loro volta in *indirect BC imposition* e *direct BC imposition*. I primi non verranno analizzati in questo testo. I secondi, invece, sono in grado di fornire una rappresentazione netta dell'IB (*immersed boundary*) considerato, utilizzando un diverso stencil computazionale in prossimità di questo. I principali metodi che cadono sotto questa categoria sono fondamentalmente due: i *ghost-cell methods*, che verranno trattati in maniera approfondita nella prossima sezione, e i *cut-cell methods*.

I *cut-cell methods* sono basati sull'utilizzo di una formulazione ai volumi finiti che consente di soddisfare le equazioni di conservazione della massa e della quantità di moto in tutto il dominio, proprietà spesso non presente negli altri metodi fin qui citati. Essi consistono nell'andare a tagliare le celle che vengono intersecate dall'IB; queste celle vengono rimodellate scartando i pezzi all'interno del corpo e unendo ad altre celle i pezzi all'esterno di esso, ottenendo delle celle di forma complessa in prossimità dell'IB. Una rappresentazione di questo metodo può essere visualizzata in Figura 2.2.

I flussi necessari all'utilizzo di uno schema ai volumi finiti riguardanti le nuove facce create dopo il taglio delle celle (f_{sw}, f_i e f_e nella Figura 2.2) sono calcolati in base a dei valori delle variabili caratterizzanti il flusso interpolate dal dominio stesso. Ad esempio, il flusso f_{sw} in Figura 2.2 può calcolato utilizzando una variabile del

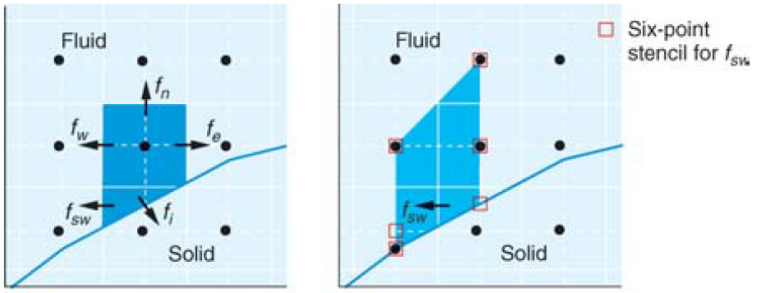


Figura 2.2: Suddivisione di una cella tagliata dall'IB secondo il *cut-cell method* e stencil a 6 punti necessario per il calcolo di uno dei nuovi flussi. [27]

flusso ϕ interpolata a partire dai 6 punti riportati in Figura 2.2 attraverso una funzione polinomiale lineare in x_1 e quadratica in x_2:

$$\phi = C_1 x_1 x_2^2 + C_2 x_2^2 + C_3 x_1 x_2 + C_4 x_1 + C_5 x_2 + C_6 \tag{2.7}$$

Questi metodi sono sicuramente di più difficile applicazione rispetto a quelli basati sul *continuos forcing approach* e presentano anche delle difficoltà nell'implementazione di superfici in movimento. Tuttavia, consentono di ottenere una rappresentazione molto più netta dell'IB, caratteristica decisamente importante ad alti numeri di Reynolds e, inoltre, consentono di disaccoppiare le equazioni da risolvere all'esterno del corpo con quelle all'interno di esso, dotate di una soluzione banale.

Si rimanda alla sezione successiva per una analisi approfondita dei metodi basati sull'utilizzo delle *ghost cells*.

2.2 Metodo Ghost-Cell

I *ghost-cell immersed boundary methods* si basano sull'utilizzo delle cosiddette *ghost cells* per modificare localmente le equazioni di governo in prossimità dell'*immersed boundary* [15]. Esse sono definite come le celle all'interno del solido che hanno almeno una cella a loro prossima appartenente al fluido [27]. Per posizionare nel dominio di calcolo il corpo immerso viene solitamente utilizzata la *level set function* inizialmente utilizzata da Osher [30]. Questa consente di calcolare per ogni punto del dominio di calcolo una funzione, detta *level set function*, così definita:

$$\varphi(x,y) = \begin{cases} |\boldsymbol{x} - \boldsymbol{x}_{IB}| & \boldsymbol{x} \text{ dentro il corpo} \\ -|\boldsymbol{x} - \boldsymbol{x}_{IB}| & \boldsymbol{x} \text{ fuori dal corpo} \end{cases} \tag{2.8}$$

Questa funzione è pari a $\varphi(x,y) = 0$ sulla superficie del corpo, assume valori positivi all'esterno di esso e negativi all'interno. Inoltre, ha l'interessante proprietà di poter

descrivere le normali alla superficie dirette verso l'esterno del corpo attraverso il suo gradiente [17]:

$$\boldsymbol{n}_x = \nabla\varphi(x, y) \tag{2.9}$$

Questi metodi sono solitamente utilizzati nell'ambito delle differenze finite ma, con le opportune modifiche, sono applicabili anche ai volumi finiti.

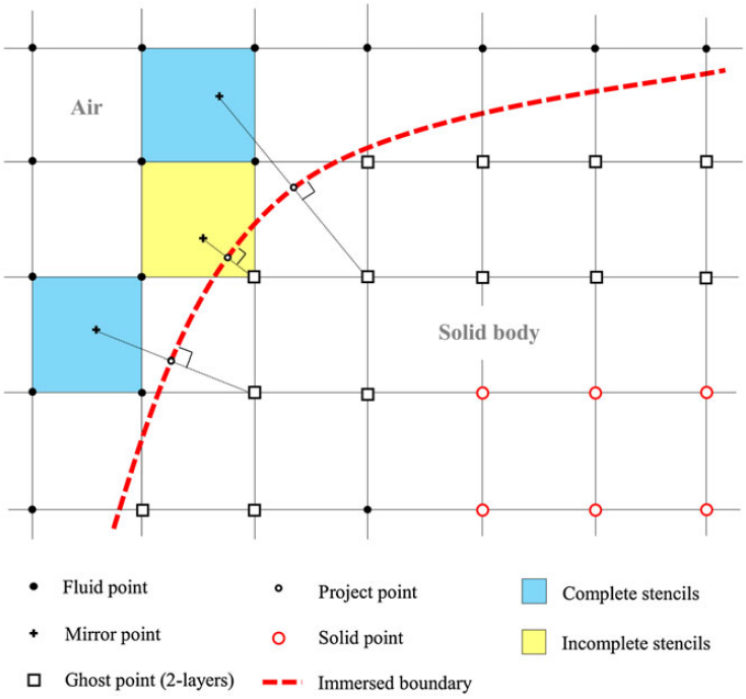


Figura 2.3: Rappresentazione del dominio suddiviso attraverso l'uso della *level set function* e l'approccio *ghost-cell*. [23]

Una volta identificate le *ghost cells* è necessario stabilire quale valore assegnarle in modo che le condizioni al contorno siano soddisfatte in corrispondenza dell'IB. Per fare ciò bisogna identificare un punto nel dominio fluido, detto *image point*, definito come il punto simmetrico al centro della *ghost cell* considerata rispetto alla superficie del corpo immerso. Si faccia riferimento alla Figura 2.3 per una rappresentazione di quanto spiegato finora, i vari *image points* vengono chiamati anche *mirror points*. Il *project point*, equidistante dal centro della *ghost cell* e dall'*image point* e posizionato sull'*immersed boundary*, sarà quello dove la condizione al contorno verrà rispettata.

Il valore delle variabili del flusso nell'*image point*, necessario all'imposizione delle condizioni al contorno, siano esse di Dirichlet o di Neumann, può essere calcolato attraverso un'interpolazione bilineare [15], ottenendo:

$$\phi_{IP} = C_1 y_{1_{IP}} y_{2_{IP}} + C_2 y_{1_{IP}} + C_3 y_{2_{IP}} + C_4 \tag{2.10}$$

dove i 4 coefficienti indicati possono essere ricavati a partire dai valore delle variabili del flusso nei 4 nodi in prossimità dell'*image point* attraverso la matrice di Vandermonde:

$$\begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} = \begin{bmatrix} y_1 y_2|_1 & y_1|_1 & y_2|_1 & 1 \\ y_1 y_2|_2 & y_1|_2 & y_2|_2 & 1 \\ y_1 y_2|_3 & y_1|_3 & y_2|_3 & 1 \\ y_1 y_2|_4 & y_1|_4 & y_2|_4 & 1 \end{bmatrix} \begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} \tag{2.11}$$

Come chiaramente visibile in Figura 2.3, possono verificarsi varie situazioni in cui l'*image point* non sia circondato da 4 nodi appartenenti al fluido, in questo caso lo stencil utilizzato per 2.11 risulta incompleto [23] ed è necessario utilizzare come quarto membro dell'interpolazione il *project point* stesso o altri punti caratteristici dello schema.

Le possibili condizioni al contorno applicabili sono quella di Dirichlet:

$$\phi_{IB} = \Phi \tag{2.12}$$

e quella di Neumann:

$$(\boldsymbol{n} \cdot \nabla\phi)_{IB} = \Psi \tag{2.13}$$

Ricordando che l'obiettivo finale è quello di trovare il valore da attribuire alle *ghost cells* tale da soddisfare le condizioni al contorno sull'IB, è ora possibile utilizzare un'interpolazione lineare a partire dai valori ricavati per l'*image point* e da quelli noti sul *project point* per scrivere [15]:

$$\phi_{GC} = \zeta\phi_{IP} + \Gamma \begin{cases} \zeta = -1 \quad \Gamma = 2\Phi \quad \text{Dirichlet BC} \\ \zeta = 1 \quad \Gamma = \Psi \cdot \Delta l \quad \text{Neumann BC} \end{cases} \tag{2.14}$$

dove Δl è la lunghezza del segmento normale.

Questa equazione può quindi essere risolta insieme alle equazioni di governo discretizzate su tutto il dominio, considerando però che per i punti interni al corpo, ad eccezione delle *ghost cells*, la soluzione banale è $\phi = 0$.

2.2.1 Condizioni al contorno nel caso di flusso comprimibile inviscido

Nel caso di flusso compressibile inviscido, applicando quanto scritto sopra, si ottengono le condizioni da imporre nella *ghost cell* [17, 23, 3]:

$$\rho_{GC} = \rho_{IP} \quad p_{GC} = p_{IP} \quad u^t_{GC} = u^t_{IP} \quad u^n_{GC} = -u^n_{IP} \tag{2.15}$$

Si è quindi esplicitata la cosiddetta condizione di non penetrazione.

Quanto descritto in 2D può essere facilmente esteso al caso 3D.

2.3 Schema al secondo ordine per le equazioni di Eulero in regime comprimibile

Verrà ora descritto nei particolari un lavoro presentato da Gorsse et al. [17], perché questa metodologia è alla base del codice **ImmerFlow**, utilizzato nel proseguimento di questa tesi.

Questo metodo è uno schema ai volumi finiti del secondo ordine e prende ispirazione dal *ghost-cell method* sopra descritto per risolvere un flusso compressibile inviscido. Le equazioni di Eulero, accoppiate all'equazione di stato dei gas perfetti:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{u} = 0 \tag{2.16}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (p\boldsymbol{u} \times \boldsymbol{u} + p\boldsymbol{I}) = 0 \tag{2.17}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E+p)\boldsymbol{u}) = 0 \tag{2.18}$$

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \boldsymbol{u}^2 \qquad p = \rho RT \qquad c = \sqrt{\gamma RT} \tag{2.19}$$

possono essere riscritte nella classica forma integrale conservativa tipica degli schemi ai volumi finiti:

$$\frac{\partial}{\partial t} \int_\Omega \boldsymbol{W} \, \mathrm{d}\Omega + \int_\sigma \boldsymbol{F} \cdot \boldsymbol{n} \, \mathrm{d}\sigma = 0 \tag{2.20}$$

dove $\boldsymbol{W}$ è il vettore delle variabili conservative e $\boldsymbol{F}(\boldsymbol{W})$ è il vettore dei flussi convettivi, funzione delle variabili conservative:

$$\boldsymbol{W} = \left\{ \begin{array}{c} \rho \\ \rho \boldsymbol{u} \\ E \end{array} \right\} \qquad \boldsymbol{F}(\boldsymbol{W}) = \left\{ \begin{array}{c} \rho \boldsymbol{u} \\ \rho \boldsymbol{u}^2 + p \\ (E+p)\boldsymbol{u} \end{array} \right\}$$

Integrando la forma conservativa 2.20 su una griglia cartesiana 2D si ottiene:

$$\frac{\mathrm{d}W_{ij}}{\mathrm{d}t} + \frac{1}{\Delta x}\left(F^x_{i+1/2j} - F^x_{i-1/2j}\right) + \frac{1}{\Delta y}\left(F^y_{j+1/2i} - F^y_{j-1/2i}\right) = 0 \tag{2.21}$$

dove W_{ij} è la media integrale delle variabili conservative su tutta la cella ij e $F^x_{i+1/2j}$ è il flusso medio nella direzione x attraverso l'interfaccia $i + 1/2j$. Questo flusso viene approssimato utilizzando il flusso numerico di Osher, calcolato risolvendo il problema di Riemann con valori iniziali provenienti da una ricostruzione delle variabili conservative di tipo MUSCL. Per maggiori informazioni si rimanda all'articolo stesso [17] e [37].

Per l'integrazione nel tempo viene utilizzato un schema Runge-Kutta di secondo ordine, dove $R(W^n)$ è il RHS dell'Equazione 2.21, funzione delle variabili conservative al time step n:

$$\begin{cases} W^\star = W^n - \Delta t R(W^n) \\ W^{n+1} = W^n - \Delta t/2(R(W^n) + R(W^\star)) \end{cases} \tag{2.22}$$

La descrizione dell'*immersed boundary* si basa sulla *level set function* precedentemente definita nell'Equazione 2.8.

2.3.1 Eulero 1D

Questo metodo è in grado di imporre la presenza dell'*immersed boundary* modificando localmente il problema di Riemann nell'interfaccia a esso più vicina. Considerando un semplice caso $1D$ rappresentato in Figura 2.4, si agisce sull'interfaccia $i+1/2$ imponendo che la velocità della discontinuità di contatto proveniente dalla risoluzione del problema di Riemann, chiamata $u^\star$, sia tale da imporre $u_b = u(x_b) = 0$, ovviamente nel caso di muro fermo.

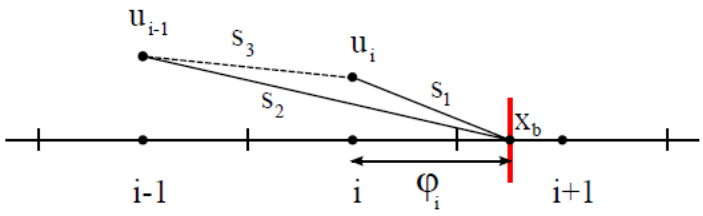


Figura 2.4: Applicazione del metodo IB a un dominio unidimensionale: vengono rappresentate tutte le principali pendenze utilizzate nello schema insieme all'IB stesso, indicato dalla linea rossa. [17]

Si può definire una distanza adimensionale $d = \varphi_i/\Delta x$, dove φ_i corrisponde al valore della *level set function* assunto al centro della cella i; in tal modo la velocità $u^\star$ può essere così scritta:

$$u^\star = u_b + \left(\frac{1}{2} - d\right) s_b \tag{2.23}$$

Questa equazione può essere facilmente interpretata facendo riferimento alla Figura 2.4. Infatti si potrebbe pensare di interpolare linearmente il valore di $u^\star$ in $x(i+1/2)$ a partire dal valore assunto dalla velocità in corrispondenza dell'IB, ovvero $u_b = u(x_b) = 0$, e retrocedendo lungo x di una distanza pari a $(1/2 - d)$ lungo una generica pendenza s_b. Anche questa pendenza dipende dalla posizione di x_b:

$$s_b = d s_1 + (1 - d) s_2 \tag{2.24}$$

e risulta praticamente una media pesata tra le due pendenze s_1 e s_2:

$$s_1 = \frac{u_b - u_1}{d} \qquad s_2 = \frac{u_b - u_{i-1}}{d}$$

Nel caso in cui sia necessario utilizzare un limiter, si può definire $s_b^l = minmod(s_b, s_3)$, dove $s_3 = u_i - u_{i-1}$.

Una volta ricavato il valore di $u^\star$, è possibile modificare il problema di Riemann definendo uno stato destro fittizio $U_+ = (-u_- + 2u^\star, p_-, c_-)$, che accoppiato allo vero stato sinistro $U_- = (u_-, p_-, c_-)$ dà origine a una velocità della discontinuità di contatto pari esattamente a $u^\star$. Questo è facilmente dimostrabile tenendo conto di come, essendo la pressione la stessa ai due estremi del problema, le due onde della prima e della terza famiglia del problema all'interfaccia debbano essere per forza o due espansioni o due urti.

Pur essendo non conservativo all'interfaccia, questo metodo risulta essere accurato al secondo ordine per la pressione, mentre l'accuratezza delle altre variabili varia da 1 a 2.

2.3.2 Eulero 2D

Quanto descritto in una dimensione, può essere facilmente esteso al caso bidimensionale considerando i flussi in ogni direzione, tenendo però conto di come, nel caso di corpo fermo, la condizione di impermeabilità risulti essere $\boldsymbol{u}_A \cdot \boldsymbol{n}_A = 0$, facendo riferimento alla Figura 2.5.

Risulta quindi necessario considerare l'angolo formato tra la normale all'*immersed boundary* e la normale alla faccia della cella su cui si vuole modificare localmente il problema di Riemann. La presenza dell'IB, infatti, influenza pesantemente il flusso numerico nel caso in cui queste due normali siano parallele, ma ha un effetto praticamente nullo nel caso in cui esse siano perpendicolari.

Facendo sempre riferimento alla Figura 2.5, volendo modificare il problema di Riemann in corrispondenza dell'interfaccia verticale in posizione $x(i+1/2, j)$, prossima al punto A, lo stato sinistro del problema sarà $U_- = (u_-, v_w, p_-, c_-)$ e quello destro $U_+ = (-u_- + 2u_w, v_w, p_w, c_w)$; dove le grandezze caratterizzate dal pedice w provengono da una media pesata di due stati, uno simile a quanto descritto per il caso unidimensionale, e uno proveniente da una estrapolazione del fluido. Si sottolinea come il peso di questa media dipenda dall'angolo di cui sopra.

$$\begin{Bmatrix} u_w \\ v_w \\ p_w \\ c_w \end{Bmatrix} = \alpha \begin{Bmatrix} u^\star \\ v^\star \\ p_- \\ c_- \end{Bmatrix} + (1-\alpha) \begin{Bmatrix} u_f \\ v_f \\ p_f \\ c_f \end{Bmatrix} \qquad \alpha = \boldsymbol{n}_A \cdot \boldsymbol{n}_{cell} \tag{2.25}$$

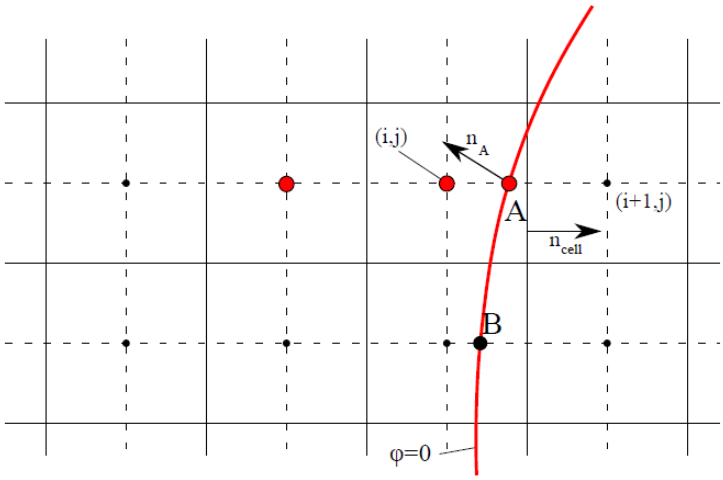


Figura 2.5: Applicazione del metodo IB a un dominio bidimensionale. Nel caso rappresentato il problema di Riemann viene modificato localmente in corrispondenza di $x(i+1/2,j)$ in modo da imporre una condizione al contorno nel punto A. [17]

Per quanto riguarda le grandezze con esponente $\star$, ovvero le componenti della velocità della discontinuità di contatto che si vuole imporre in $x(i+1/2,j)$, si procede in maniera simile a quanto fatto nel caso unidimensionale. Una distanza adimensionalizzata viene descritta:

$$d = \frac{|\varphi_i ij|}{|\varphi_i ij| + |\varphi_{i+1,j}|} \tag{2.26}$$

Questa viene utilizzata per definire il vettore normale nel punto A a partire da quelli nei centri cella $x(i,j)$ e $x(i+1,j)$.

$$\boldsymbol{n}_A = \boldsymbol{n}_{i,j} + d(\boldsymbol{n}_{i+1,j} - \boldsymbol{n}_{i-1,j}) \tag{2.27}$$

Si ricorda come le informazioni riguardanti le normali all'IB provengano dal gradiente della *level set function*.

La velocità $\boldsymbol{u}^\star$ può quindi essere espressa come:

$$\boldsymbol{u}^\star = \left\{ \begin{array}{c} u_n^\star n_x + u_\tau^\star \tau_x \\ u_n^\star n_y + u_\tau^\star \tau_y \end{array} \right\} \quad \begin{cases} \boldsymbol{u}^\star \cdot \boldsymbol{n}_A = u_n^\star = \boldsymbol{u}_A \cdot \boldsymbol{n}_A + (1/2 - d) s_A^n \\ \boldsymbol{u}^\star \cdot \boldsymbol{\tau}_A = u_\tau^\star = \boldsymbol{u}_- \cdot \boldsymbol{\tau}_A \end{cases} \tag{2.28}$$

dove $\boldsymbol{n}_A$ e $\boldsymbol{\tau}_A$ sono i vettori normali e tangenziali nel punto A appartenente all'IB. La pendenza s_A^n è così definita:

$$s_A^n = \boldsymbol{u}_A \cdot \boldsymbol{n}_A - \boldsymbol{u}_i \cdot \boldsymbol{n}_A + \frac{1-d}{1+d}(\boldsymbol{u}_A \cdot \boldsymbol{n}_A - \boldsymbol{u}_{i-1} \cdot \boldsymbol{n}_A) \tag{2.29}$$

Le grandezze con pedice f sono invece ricavate utilizzando una semplice interpolazione lineare a partire dalla cella più vicina dal punto di vista *upwind*.

Per maggiori informazioni si rimanda all'articolo stesso [17].

3 Applicazione IB a un tubo d'urto 1D

In questo capitolo viene portato un esempio di applicazione della metodologia *immersed boundary* a un caso unidimensionale; in particolare, viene considerato un codice per la simulazioni di un tubo d'urto con le condizioni iniziali tipiche del problema di Sod. Questo problema è infatti comunemente utilizzato per valutare il comportamento e l'accuratezza di uno schema numerico. Il linguaggio utilizzato per scrivere questo programma è il `C++`.

Le equazioni di Eulero, già menzionate nel Capitolo 2, sono discretizzate utilizzando il metodo dei volumi finiti a partire dalla loro formulazione integrale. Lo schema viene inizialmente costruito al primo ordine nello spazio e nel tempo, per poi essere successivamente portato al secondo ordine. L'attenzione si sposta poi sulle condizioni al contorno, che vengono modificate per poter riflettere un urto incidente. Infine, la riflessione dell'urto viene spostata dai bordi del dominio al suo interno, attraverso due diversi approcci *immersed boundary*.

3.1 Discretizzazione al primo ordine delle Equazioni di Eulero in forma integrale

3.1.1 Approccio ai volumi finiti

Volendo descrivere campi di moto fluido in cui siano presenti delle discontinuità, è necessario partire dalle equazioni conservative (*fixed control volumes*) in forma integrale; esse infatti sono considerate "più fondamentali" [5] di quelle in forma differenziale, in quanto non presentano singolarità nel caso in cui siano presenti delle onde d'urto. Per chiarezza vengono ora richiamate le equazioni di Eulero 1D in questa forma:

$$\frac{\partial}{\partial t}\int_{\Omega} \boldsymbol{W}\,\mathrm{d}\Omega + \int_{\sigma} \boldsymbol{F}\cdot\boldsymbol{n}\,\mathrm{d}\sigma = 0 \tag{3.1}$$

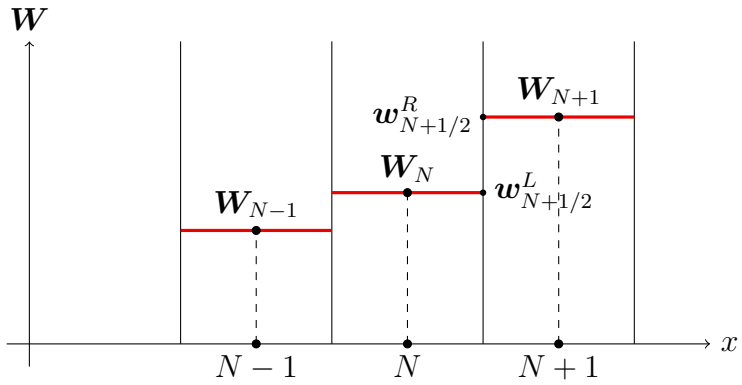


Grafico 3.1: Distribuzione costante a tratti delle variabili conservative.

dove $\boldsymbol{W}$ è il vettore delle variabili conservative e $\boldsymbol{F}$ è il vettore dei flussi convettivi:

$$\boldsymbol{W} = \begin{Bmatrix} \rho \\ \rho u \\ E \end{Bmatrix} \qquad \boldsymbol{F}(\boldsymbol{W}) = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{Bmatrix} \tag{3.2}$$

La derivazione di queste equazioni può essere ritrovata in qualunque libro di fluidodinamica.

Su una griglia unidimensionale formata da N celle si può scrivere:

$$\frac{\partial}{\partial t} \int_{x_{N-1/2}}^{x_{N+1/2}} \boldsymbol{W} \, \mathrm{d}x + \boldsymbol{F}(x_{N+1/2}) - \boldsymbol{F}(x_{N-1/2}) = 0 \tag{3.3}$$

La quantità che viene effettivamente calcolata dal programma scritto è discreta e coincide con la media integrale di cella della variabile conservativa $\boldsymbol{W}$:

$$\boldsymbol{W}_N = \frac{1}{\Delta x} \int_{x_{N-1/2}}^{x_{N+1/2}} \boldsymbol{W} \, \mathrm{d}x \tag{3.4}$$

Si può quindi scrivere:

$$\frac{\partial \boldsymbol{W}_N}{\partial t} + \frac{1}{\Delta x} \left[\boldsymbol{F}(x_{N+1/2}) - \boldsymbol{F}(x_{N-1/2}) \right] = 0 \tag{3.5}$$

Conoscendo solo il valore della media di cella per ogni N, la distribuzione delle variabili conservative che si ha nel dominio risulta essere costante a tratti, come si può vedere in Figura 3.1.

Integrando nel tempo:

$$\int_{t^K}^{t^{K+1}} \frac{\partial \boldsymbol{W}_N}{\partial t} + \frac{1}{\Delta x} \left[\int_{t^K}^{t^{K+1}} \boldsymbol{F}(x_{N+1/2}) - \int_{t^K}^{t^{K+1}} \boldsymbol{F}(x_{N-1/2}) \right] = 0$$

$$\boldsymbol{W}_N^{K+1} - \boldsymbol{W}_N^K + \frac{1}{\Delta x}\left[\int_{t^K}^{t^{K+1}} \boldsymbol{F}(x_{N+1/2}) - \int_{t^K}^{t^{K+1}} \boldsymbol{F}(x_{N-1/2})\right] = 0$$

dove K è il passo temporale. Definendo ora $\boldsymbol{F}_{N+1/2}^K$ come la media integrale del flusso all'interfaccia $N+1/2$ durante il tempo Δt tra l'istante K e l'istante $K+1$:

$$\boldsymbol{F}_{N+1/2}^K = \frac{1}{\Delta t}\int_{t^K}^{t^{K+1}} \boldsymbol{F}(x_{N+1/2})\,\mathrm{d}t \tag{3.6}$$

si arriva alla formulazione finale:

$$\boldsymbol{W}_N^{K+1} = \boldsymbol{W}_N^K - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}(x_{N+1/2}) - \boldsymbol{F}(x_{N-1/2})\right) \tag{3.7}$$

Si sottolinea come i flussi siano funzioni delle variabili conservative $\boldsymbol{F}(\boldsymbol{W})$, per cui si può scrivere, a patto che la condizione CFL venga rispettata (i.e. non si esca dal dominio di dipendenza [37]):

$$\boldsymbol{F}(x_{N+1/2}) = \mathcal{F}(\boldsymbol{W}_N^K, \boldsymbol{W}_{N+1}^K) \tag{3.8}$$

dove $\mathcal{F}$ è un generico flusso numerico.

Data la natura iperbolica delle equazioni di Eulero in campo compressibile, le informazioni propagano lungo le cosiddette linee caratteristiche. In corrispondenza dell'interfaccia $i+1/2$ nel Grafico 3.1 è presenta una discontinuità; infatti a sinistra dell'interfaccia si ha il valore $w_{i+1/2}^L$ di una generica variabile conservativa, mentre a destra di essa si ha $w_{i+1/2}^R$. Questa discontinuità dà origine a un problema ai valori iniziali, detto problema di Riemann, che divide lo spazio in quattro zone diverse, come visibile nel Grafico 3.2.

Queste sono separate da tre onde caratteristiche: un'onda della prima famiglia, che viaggia con velocità $u-a$ dove $a=\sqrt{\gamma RT}$ è la velocità del suono, un'onda della seconda famiglia che viaggia con velocità u e un'onda della terza famiglia che viaggia con velocità $u+a$. Le onde della prima e della terza famiglia possono essere urti o espansioni, quella della seconda famiglia è sempre una superficie di contatto. Si sottolinea come in ognuna delle quattro zone dello spazio a, b, c e d l'andamento delle variabili conservative sia costante.

La scelta del flusso numerico $\mathcal{F}$ in base alla propagazione di queste linee caratteristiche, quindi in base al comportamento fisico del problema, è alla base della filosofia *upwind*. Infatti, come si può chiaramente vedere nel Grafico 3.2, la linea verticale nera che segnala la posizione dell'interfaccia al variare del tempo giace sempre nella stessa zona del piano (la zona c in questo caso), indicando come il valore di $\boldsymbol{W}_{N+1/2} = \boldsymbol{W}^\vee(\boldsymbol{W}_N^K, \boldsymbol{W}_{N+1}^K)$ sia costante nel tempo e, in questo caso, corrisponda al set di variabili conservative presente nella zona c dello spazio. Si può quindi calco-

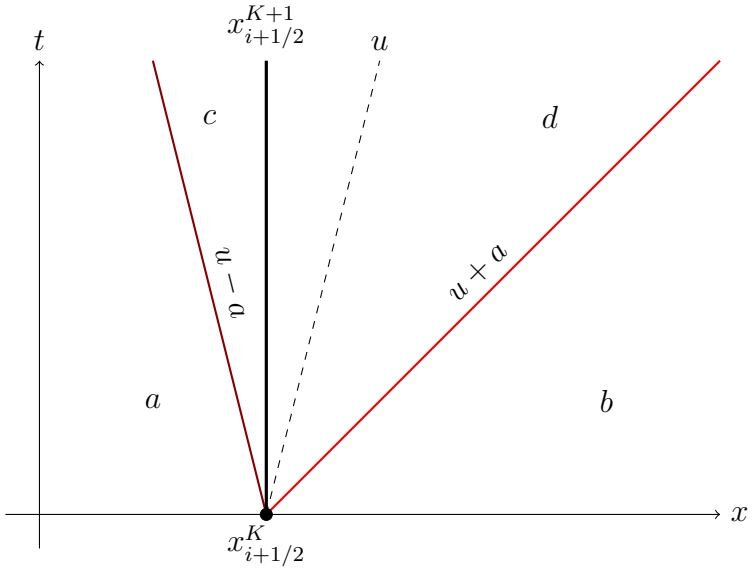


Grafico 3.2: Divisione del piano delle fasi nel problema di Riemann per Eulero.

lare il flusso numerico utilizzando le variabili conservative derivanti dalla risoluzione del problema di Riemann, indicate dall'apice $\vee$:

$$\boldsymbol{F}(x_{N+1/2}) = \frac{1}{\Delta t}\int_{t^K}^{t^{K+1}} \boldsymbol{F}(\boldsymbol{W}^{\vee}(x_{N+1/2}))\,\mathrm{d}t = \boldsymbol{F}(\boldsymbol{W}^{\vee}(\boldsymbol{W}_N^K, \boldsymbol{W}_{N+1}^K)) \tag{3.9}$$

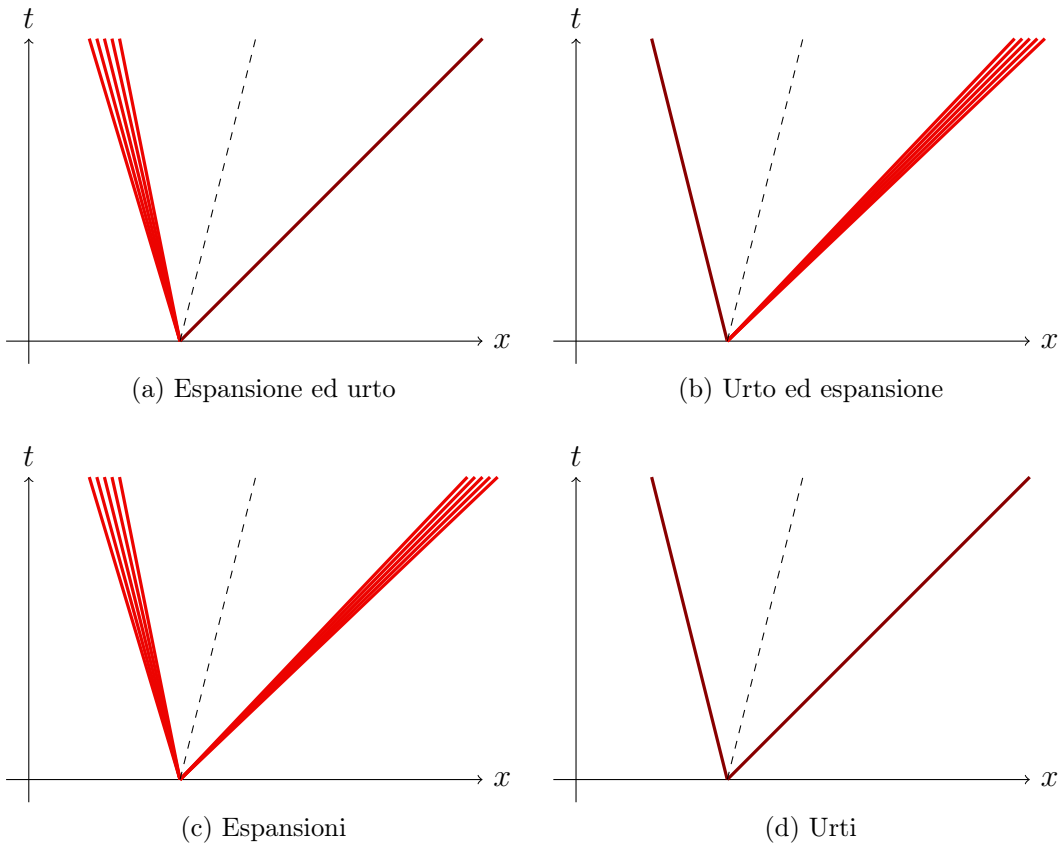
Una volta calcolati i flussi in questo modo è possibile, a partire dalla soluzione al passo temporale K, evolvere la soluzione al passo $K+1$ utilizzando la formula 3.7 da cui si è partiti.

L'approccio fin qui descritto ricade nell'ampia famiglia dei metodi detti "alla Godunov". Per maggiori informazioni su questi metodi o più in generale sulle varie strategie di risoluzione del problema di Riemann, approssimate e non, si rimanda ai numerosi testi sull'argomento citati in bibliografia; nella parte successiva, invece, viene solamente descritto il metodo per la risoluzione esatta di questo problema.

3.1.2 Soluzione esatta del problema di Riemann per le equazioni di Eulero

Riferendosi al Grafico 3.2, delle quattro zone in cui è diviso il piano delle fasi le zone a e b sono quelle che si riferiscono alle condizioni iniziali della discontinuità. Le variabili conservative in queste due zone sono note e costituiscono l'input che permette di risolvere completamente il problema di Riemann, determinando la natura delle tre onde. Si sottolinea come a cavallo della superficie di contatto la pressione e la

componente normale della velocità (l'unica componente nel caso unidimensionale) siano le stesse. Questo riduce le incognite da 6 a 4, ovvero: $p_c = p_d$, $u_c = u_d$, E_c e E_d. Le quattro equazioni che accoppiate permettono di risolvere il problema dipendono dalla natura delle onde della prima e della terza famiglia.



(a) Espansione ed urto

(b) Urto ed espansione

(c) Espansioni

(d) Urti

Grafico 3.3: Alcune casistiche per le onde caratteristiche in Eulero.

Si consideri ad esempio il caso di espansione a sinistra e urto a destra; gli altri casi saranno facilmente deducibili da questo. In questo caso per legare la zona a alla zona c si possono utilizzare gli invarianti di Riemann [37]; infatti, attraverso un'espansione della prima famiglia, gli invarianti $\mathrm{d}R_2$ e $\mathrm{d}R_3$ si conservano:

$$\begin{cases} \frac{2}{\gamma-1}\,\mathrm{d}a + \mathrm{d}u - \frac{a}{\gamma(\gamma-1)}\,\mathrm{d}S = 0 \\ \mathrm{d}S = 0 \end{cases} \Rightarrow u = u_a + \frac{2}{\gamma-1} a_a \left[1 - \left(\frac{p}{p_a}\right)^{\frac{\gamma-1}{2\gamma}}\right] \tag{3.10}$$

dove $u = u_c = u_d$ e $p = p_c = p_d$.

Per legare la zona b alla zona d si utilizzano le condizioni di salto di Rankine-Hugoniot [6] valide attraverso l'urto della terza famiglia.

$$\begin{cases} \rho_b u_b = \rho u \\ p_b + \rho_b u_b^2 = p + \rho u^2 \\ (E_b + p_b)u_b = (E + p)u \end{cases} \Rightarrow u = u_b + (p - p_b)\left[\frac{\frac{2}{(\gamma+1)\rho_b}}{p + \frac{\gamma-1}{\gamma+1}p_b}\right] \tag{3.11}$$

Uguagliando queste due espressioni è possibile ottenere una equazione non lineare che, una volta risolta, fornisce il valore di $p = p_c = p_d$. A partire da questo valore è possibile ricavare facilmente tutte le altre incognite utilizzando la relazione isentropica per una espansione e le condizioni di salto per un urto.

La natura delle due onde lineari della prima e della terza famiglia non è pero nota a priori, pur dipendendo solamente dagli stati in a e b; è necessario scegliere quale equazione non lineare risolvere per ottenere il valore di p e questa equazione dipende dalla configurazione delle onde, come si può vedere nel grafico 3.3. Solitamente una prima scelta di due possibili configurazioni viene fatta in base ai valori di pressione e velocità in a e b, per poi risolvere entrambe le equazioni lineari rimaste e vedere quale dei due set di risultati ottenuti soddisfi le ipotesi di partenza [10]. Questo controllo finale può essere fatto calcolando le pendenza delle linee caratteristiche delle tre famiglie utilizzando gli appropriati valori di u ed a.

Una volta ottenuti i valori delle variabili conservative nelle zone c e d è possibile calcolare le pendenze della varie onde e capire in che zona dello spazio giace la linea verticale $x_{N+1/2}/t = 0$.

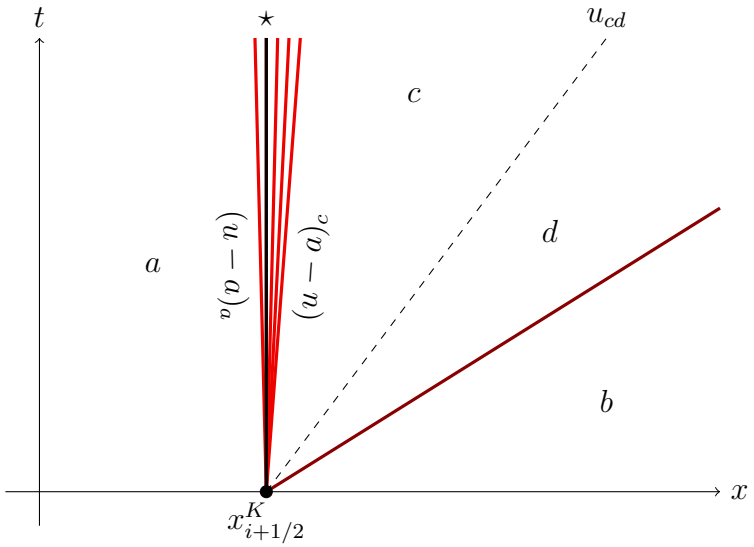


Grafico 3.4: Espansione transonica.

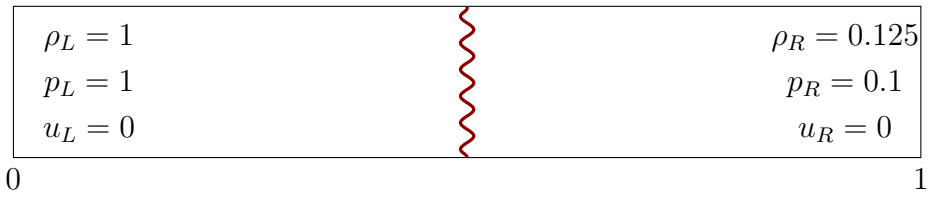


Figura 3.1: Tubo d'urto nelle condizioni iniziali.

Particolare attenzione deve essere posta nel caso in cui questa linea cada in corrispondenza di un fascio di espansione; questo caso viene solitamente chiamato "espansione transonica". Come visibile nel Grafico 3.4, in corrispondenza della condizione sonica la pendenza della linea nel piano delle fasi è $u^\star - a^\star = 0$. A partire dall'Equazione 3.10 si può quindi ottenere:

$$u^\star = \frac{\gamma - 1}{\gamma + 1}\left(\frac{2}{\gamma - 1}a_a + u_a\right) \tag{3.12}$$

3.1.3 Struttura del codice

Un programma scritto in C++ che simuli un tubo d'urto unidimensionale è stato sviluppato per poter poi implementare vari *immersed boundary methods*. Si sottolinea come in questo codice sia stata utilizzata una routine per risolvere il problema di Riemann esatto fornita da OPTIMAD Engineering srl, insieme alle classi in essa utilizzate. Un tubo d'urto è uno strumento realmente utilizzato in vari ambiti di ricerca e consiste fondamentalmente in un dominio unidimensionale diviso in due zone da un diaframma removibile, indicate in Figura 3.1 come zona L e zona R. Si immagini di eliminare il diaframma presente a $x = 0.5$ (dominio $0 < x < 1$) al tempo $t = 0$; essendo presente una discontinuità tra le variabili del flusso si genera un problema ai valori iniziali, o problema di Riemann, in questa posizione. L'iniziale discontinuità evolve generando onde non lineari che propagano nel dominio. Date le condizioni iniziali del problema di Sod [36], le onde generate sono un'espansione che si muove verso sinistra e una superficie di contatto e un'onda d'urto che si muovono verso destra, come visibile in Figura 3.2.

Gli input del codice risultano essere il CFL $= u_{max}\Delta t/\Delta x$, il numero di celle n in cui il dominio viene suddiviso e il tempo finale t_f a cui si vuole avere la soluzione.

Il primo passo consiste nell'inizializzazione del dominio formato da $N + 2$ celle utilizzando le condizioni iniziali di Sod:

```
//fluid initialization
Fluid in[n+2], interf[n+2];
for (int i=0; i<n+2; i++) {
  in[i].setSpecificHeatRatio(gam);
```

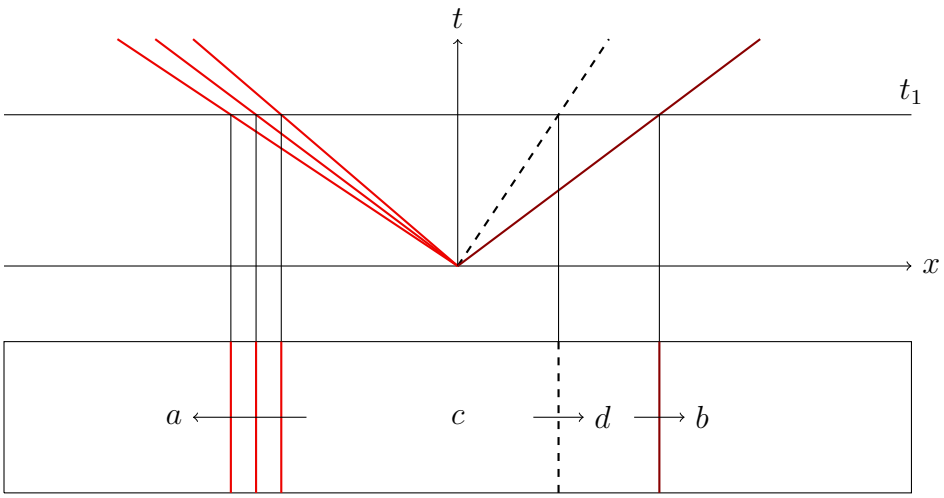


Figura 3.2: Tubo d'urto per $t_1 > 0$.

```
  if (i<n/2+1) in[i].setPrimitive(ul,pl,al);
  else in[i].setPrimitive(ur,pr,ar);
}
```

Il dominio viene suddiviso in $n+2$ celle in modo tale da avere una cella 0 e una cella $n+1$ non facenti parte del dominio in cui si vuole calcolare la soluzione, queste celle saranno utili successivamente per imporre le condizioni al contorno.

Per la gestione delle grandezze legate al fluido è stata utilizzata una particolare classe denominata `Fluid` dotata di alcuni metodi pubblici e privati per la gestione delle sue variabili. In particolare il metodo `setSpecificHeatRatio` permette di settare il $\gamma = c_p/c_v$ per ogni i-esimo oggetto appartenente alla classe `Fluid`, mentre il metodo `setPrimitive` consente di assegnare a ogni oggetto un certo valore di velocità, pressione e velocità del suono. Queste variabili primitive sono sufficienti, insieme al γ, per calcolare il set delle variabili conservative 3.2 e tutte le altre grandezze di interesse. Tutto questo è fatto all'interno della classe stessa.

Per procedere all'evoluzione della soluzione iniziale al passo successivo tramite l'Equazione 3.7 è necessario innanzitutto calcolare il Δt invertendo la definizione del CFL a ogni time step:

$$\Delta t = \frac{\mathrm{CFL} \cdot \Delta x}{u_{max}}$$

dove la massima velocità di propagazione del segnale u_{max} viene così calcolata:

```
for (int j=0; j<n+2; j++) {
  maxs=std::max(in[j].speedOfSound()+
   std::abs(in[j].velocity()),maxs);
}
```

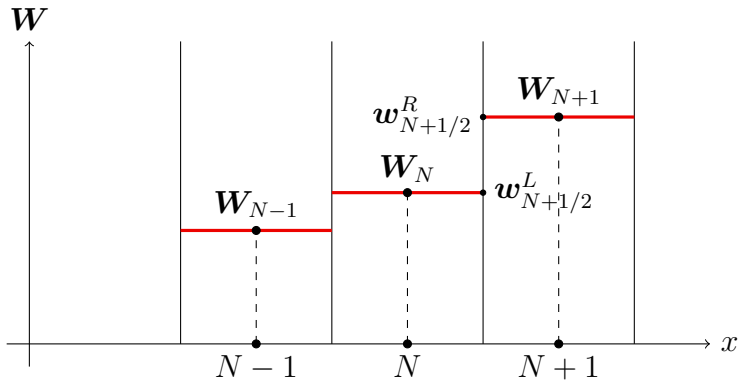


Grafico 3.5: Distribuzione costante a tratti delle variabili conservative.

Anche qui vengono utilizzati dei metodi della classe `Fluid`, ad esempio `in[j].speedOfSound()` consente di avere il valore della velocità del suono a nella j-esima cella del vettore `in`. Nel prosieguo si eviterà di spiegare singolarmente ogni metodo, data la chiarezza della sua funzione in base al suo nome.

Data la ricostruzione costante a tratti delle variabili conservative $\boldsymbol{W}$ (Equazione 3.2) che si utilizza nello schema al primo ordine (Figura 3.5), i valori di input del problema di Riemann all'interfaccia $N + 1/2$ risultano essere:

$$\begin{cases} \boldsymbol{w}^R_{N+1/2} = \boldsymbol{W}_{N+1} & \text{regione } b \text{ del problema di Riemann} \\ \boldsymbol{w}^L_{N+1/2} = \boldsymbol{W}_N & \text{regione } a \text{ del problema di Riemann} \end{cases}$$

I valori stessi delle $\boldsymbol{W}$ nelle celle possono essere dati in input alla funzione che consente di risolvere il problema di Riemann in maniera esatta, ovvero come mostrato nella sezione precedente.

```
h=dt/dx;
// calcolo flussi celle interne
for (int k=1; k<n; k++) {
    riemann::godunov( in[k], in[k+1], 1, 0, interf[k], flux);
    moltiplicaVecCost(flux,h);
    sommaVec(fluxdiff[k],flux);
    diffVec(fluxdiff[k+1],flux);
}
```

La routine `riemann::godunov`, utilizzando come input le variabili conservative nelle regioni a e b, consente di calcolare i flussi conservativi $\boldsymbol{F}(\boldsymbol{W})$ (Equazione 3.2)

semplicemente come:

$$\boldsymbol{F}(\boldsymbol{W}^{\vee}) = \left\{ \begin{array}{c} \rho^{\vee}u^{\vee} \\ \rho^{\vee}u^{\vee}u^{\vee} + p^{\vee} \\ (E^{\vee} + p^{\vee})u^{\vee} \end{array} \right\}$$

dove $\boldsymbol{W}^{\vee}$ è il vettore delle variabili conservative caratteristico della regione dove giace l'interfaccia $N+1/2$ nel piano delle fasi. I flussi vengono poi sommati e sottratti (in base ovviamente all'indice della cella) a una vettore `fluxdiff`, attraverso una funzione di questo tipo:

```
void sommaVec(std::array<double,3> &a, const
 std::array<double,3> &b) {
    a[0]=a[0]+b[0];
    a[1]=a[1]+b[1];
    a[2]=a[2]+b[2];
}
```

$$\text{fluxdiff} = \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}(x_{N+1/2}) - \boldsymbol{F}(x_{N-1/2})\right)$$

Questo procedimento è sicuramente valido per le celle interne del dominio, ma per le celle di estremità non è possibile calcolare in questo modo i flussi nelle interfacce $1-1/2$ e $n+1/2$. Dato che al momento non si è interessati ai fenomeni di bordo, nè si vuole fare in modo che le perturbazioni escano indisturbate dal dominio (*non-reflecting boundary condition*), si è imposto che nelle celle 0 e $n+1$ le variabili conservative abbiano lo stesso valore che assumono, rispettivamente, nelle celle 1 e n. In tal modo è stato imposto gradiente nullo ai bordi del dominio, condizione perfettamente accettabile finché le perturbazioni non raggiungono i bordi. I flussi ai bordi del dominio possono essere quindi calcolati con la stessa metodologia di cui sopra.

```
//bc sinistra
u0=in[1].velocity();
p0=in[1].pressure();
a0=in[1].speedOfSound();
in[0].setPrimitive(u0,p0,a0);
riemann::godunov( in[0], in[1], 1, 0, interf[0], flux);
moltiplicaVecCost(flux,h);
diffVec(fluxdiff[1],flux);
```

L'ultimo passo è l'evoluzione della soluzione, che attraverso l'utilizzo del vettore `fluxdiff` risulta essere semplicemente:

$$\boldsymbol{W}_N^{K+1} = \boldsymbol{W}_N^K - \text{fluxdiff}$$

Questo procedimento viene ripetuto in un ciclo temporale fino al raggiungimento di $t = t_f$, ovviamente registrando su un file i valori di $\boldsymbol{W}$ per ogni cella e per ogni time step, prima che essi vengano aggiornati.

In parallelo al calcolo della soluzione con il metodo alla Godunov di cui sopra, viene anche calcolata la soluzione esatta per ogni time step, in modo da poter effettuare un confronto tra le due. La soluzione esatta del problema di Sod risulta essere la soluzione di un singolo problema di Riemann, quello che si genera inizialmente a $t = 0$ e $x = 0.5$ (Grafico 3.2). La soluzione delle zone c e d e della zona interna all'espansione è immediata in quanto basta risolvere l'equazione non lineare ricavata precedentemente per ottenere il valore di $p = p_c = p_d$, da cui è poi banale ricavare le altre incognite.

$$u_a + \frac{2}{\gamma - 1} a_a \left[1 - \left(\frac{p}{p_a} \right)^{\frac{\gamma-1}{2\gamma}} \right] = u_b + (p - p_b) \left[\frac{\frac{2}{(\gamma+1)\rho_b}}{p + \frac{\gamma-1}{\gamma+1} p_b} \right]$$

3.1.4 Risultati

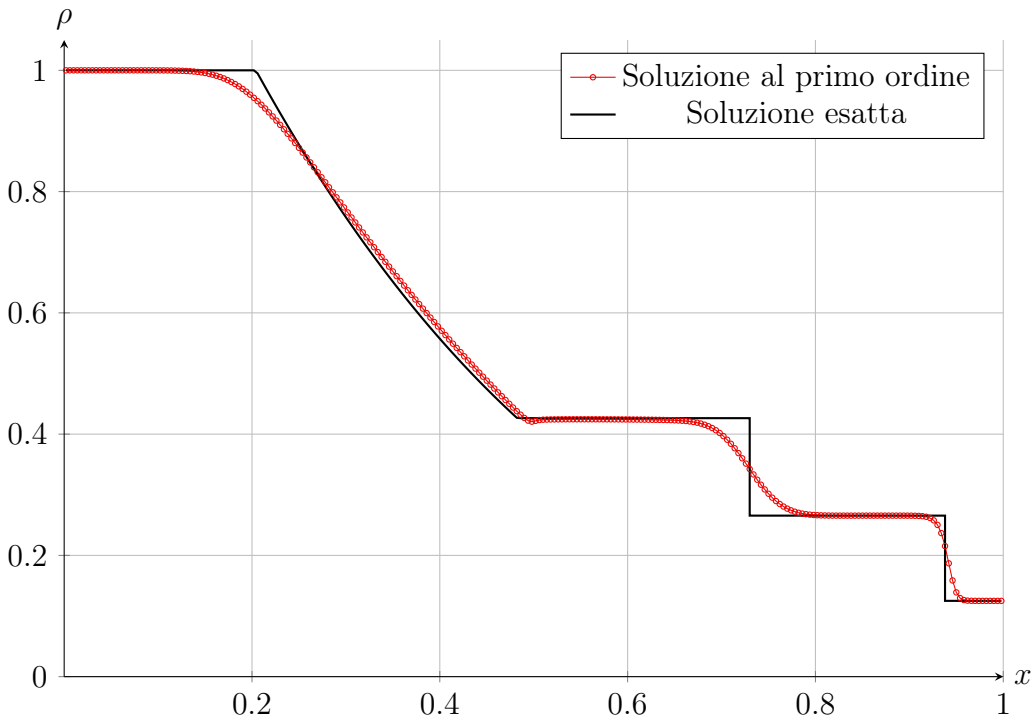


Grafico 3.6: Grafico della densità. $t = 0.25$, $N = 250$, CFL $= 0.4$.

I risultati al primo ordine possono essere visualizzati nei Grafici 3.6 e 3.7. Gli andamenti sono quelli classici di un tubo d'urto, su cui molto è stato scritto in letteratura, e confermano la correttezza del codice scritto. Risulta interessante notare

come l'espansione, l'onda più a sinistra, e l'urto, l'onda più a destra, siano ben visibili in tutti e due i grafici presentati; mentre, la discontinuità di contatto sia visibile solamente nel grafico relativo alla densità. Infatti, come scritto precedentemente, a cavallo di una superficie di contatto la componente normale della velocità e la pressione rimangono costanti. Visualizzando l'evoluzione nel tempo della soluzione, i cui Grafici non sono riportati qui per motivi di spazio, è possibile vedere come la superficie di contatto sia quella che diffonda di più rispetto alle altre due; essa infatti è mal rappresentata dalla soluzione numerica e con l'avanzare del tempo viene diffusa sempre di più. Questo è dovuto al fatto che le linee caratteristiche della seconda famiglia "fasciano" la superficie di contatto [10], a differenza degli urti, che vengono raggiunti sul lato di bassa pressione da tutte le linee caratteristiche.

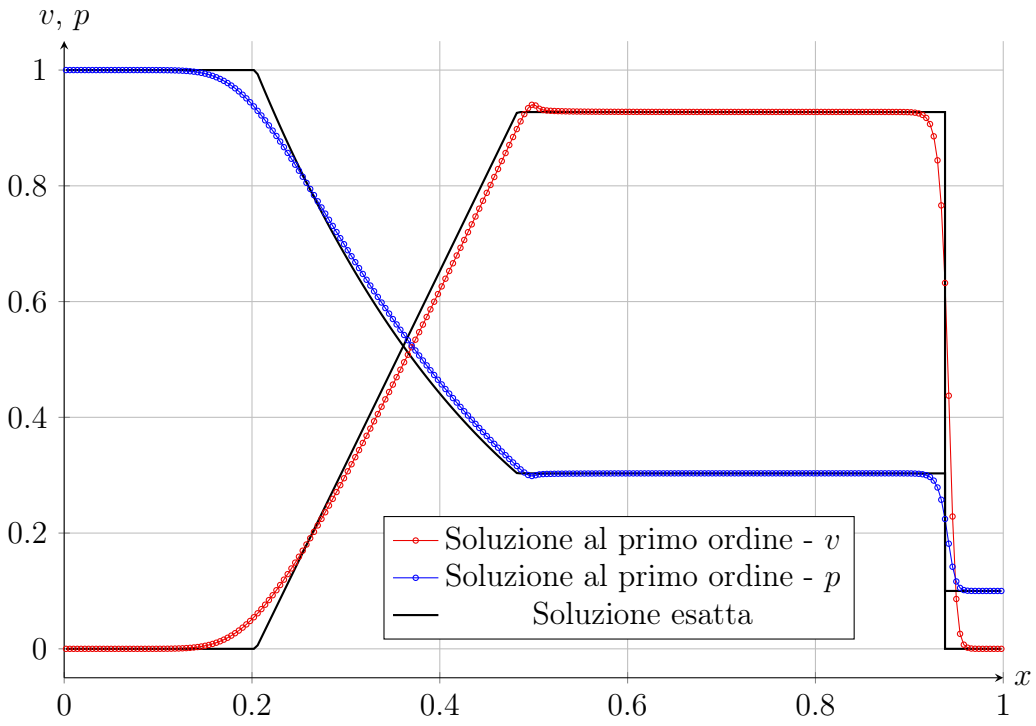


Grafico 3.7: Grafico della velocità e della pressione. $t = 0.25$, $N = 250$, CFL $= 0.4$.

Nel Grafico 3.8 si può notare come a diversi CFL la discontinuità di contatto venga catturata in modo diverso dallo schema al primo ordine. Contrariamente a quanto ci si potrebbe aspettare, a CFL più alti la discontinuità viene catturata meglio rispetto a quanto avvenga a CFL più bassi. Questo perché negli schemi al primo ordine di questo tipo la diffusività numerica è proporzionale al termine $(1 - a^2\Delta t^2/\Delta x^2)$ [22].

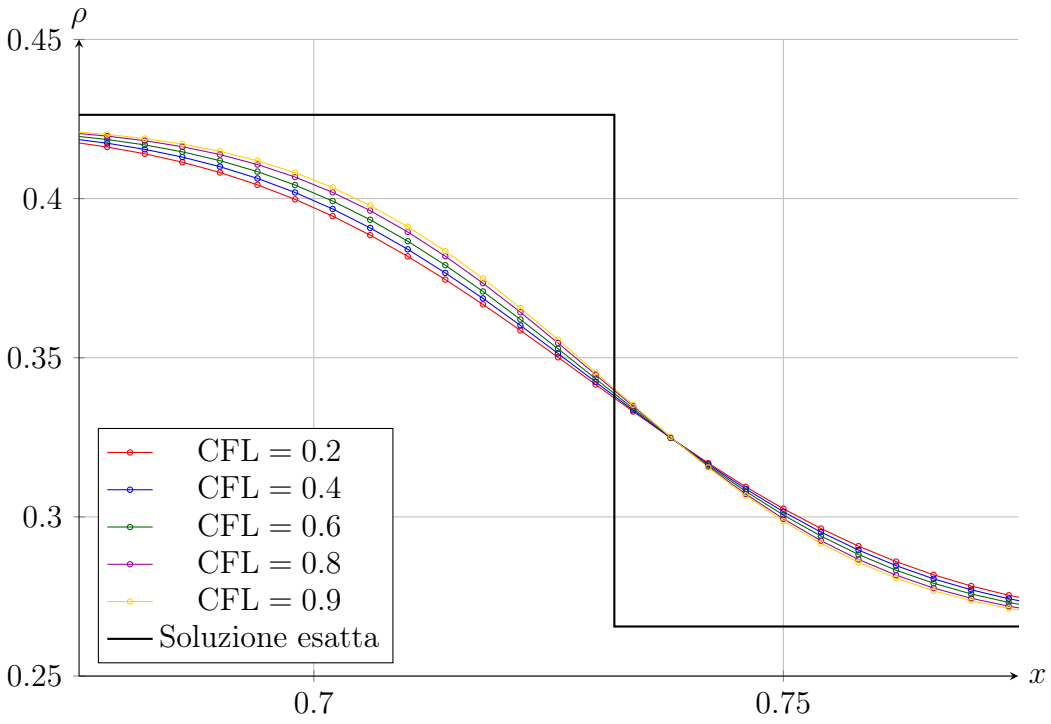


Grafico 3.8: Confronto a diversi CFL della discontinuità di contatto. $t = 0.25$, $N = 250$.

3.1.4.1 Modifica delle condizioni al contorno per la riflessione dell'urto

Si vogliono ora modificare le condizioni al contorno sul lato destro del dominio per fare in modo che l'urto venga riflesso dalla parete. Per poterlo fare è innanzitutto necessario riflettere sul significato fisico della discontinuità di contatto. Si immagini di avere un tubo d'urto come in Figura 3.1; a sinistra del diaframma è presente uno scalare passivo in grado di colorare il fluido in rosso, a destra un altro scalare passivo che colori il fluido in blu. Per $t > 0$ la linea di demarcazione tra fluido rosso e blu coincide esattamente con la posizione della discontinuità di contatto, che separa "fisicamente" i due fluidi destro e sinistro. Si immagini ora di essere posizionati in corrispondenza dell'ultima interfaccia del dominio, $n + 1/2$; se si vuole imporre che il fluido si fermi in corrispondenza di quella interfaccia, allora la discontinuità di contatto in quel punto dovrà avere velocità nulla (Grafico 3.9). Questa può essere imposta in due modi.

Il flusso delle variabili conservative all'interfaccia può essere calcolato manualmente, senza risolvere il problema di Riemann a partire dagli stati a e b (lo stato b giace fuori dal dominio), tenendo conto che si vuole imporre velocità della discontinuità

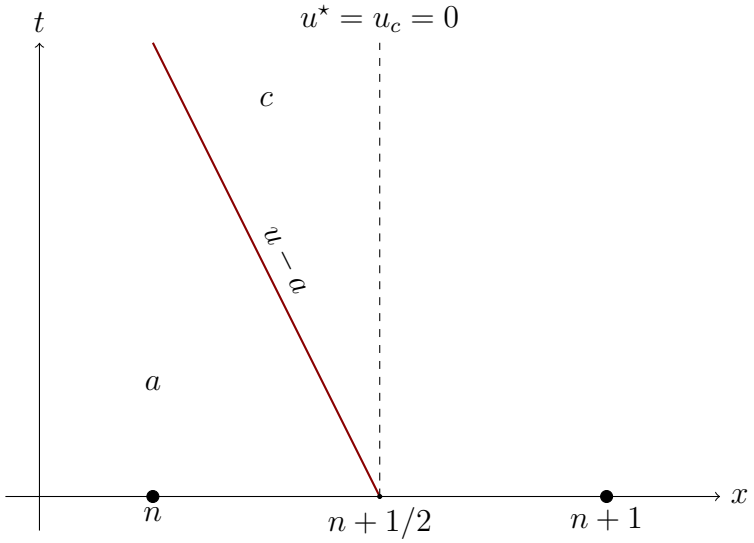


Grafico 3.9: Imposizione della presenza di una parete in corrispondenza dell'interfaccia $n + 1/2$.

di contatto nulla $u^\star = u_c = 0$.

$$\boldsymbol{F}_{n+1/2} = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ (E+p)u \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ p_c \\ 0 \end{array} \right\} \tag{3.13}$$

Come si poteva immaginare, attraverso una parete non si ha flusso di materia o energia, ma solo flusso di quantità di moto, in forma ovviamente di pressione applicata. Si deve quindi ricavare il valore della pressione p_c; questo è ignoto in quanto non si conosce la natura dell'onda non lineare che viaggia con velocità $u - a$. Si può però supporre per semplicità che essa sia una espansione, in quanto il risultato finale non sarebbe molto diverso in caso di urto (che è simile a una compressione isoentropica). La pressione si può quindi ricavare con l'Equazione già vista in precedenza 3.10, imponendo $u_c = 0$:

$$u_c = u_a + \frac{2}{\gamma - 1} a_a \left[1 - \left(\frac{p_c}{p_a} \right)^{\frac{\gamma-1}{2\gamma}} \right] \Rightarrow p_c = \left(\frac{u_a + 2a_a/(\gamma-1)}{2a_a/(\gamma-1)} \right)^{\frac{2\gamma}{\gamma-1}} p_a$$

In termini di codice:

```
//impongo flusso ultima faccia - hard bc destra
flux[0]=0;
flux[1]=pow((in[n].velocity()+2/(gam-1)*in[n].speedOfSound())
  /(2/(gam-1)*in[n].speedOfSound())*pow(in[n].pressure(),
```

```
    (gam-1)/(2*gam)),2*gam/(gam-1));
flux[2]=0;
moltiplicaVecCost(flux,h);
sommaVec(fluxdiff[n],flux);
```

Il secondo modo di imporre $u^\star = u_c = 0$ utilizza la cella $n+1$ che, opportunamente modificata, cambia il problema di Riemann all'interfaccia $n+1/2$ in modo da ottenere quanto voluto. Come già visto con l'Equazione 2.15, è sufficiente imporre che la cella $n+1$ abbia lo stesso valore delle variabili primitive p e a, e il valore opposto di u rispetto alla cella n.

$$\boldsymbol{P}_n = \begin{Bmatrix} u_n \\ p_n \\ a_n \end{Bmatrix} \qquad \boldsymbol{P}_{n+1} = \begin{Bmatrix} -u_n \\ p_n \\ a_n \end{Bmatrix} \tag{3.14}$$

dove $\boldsymbol{P}$ è il vettore delle variabili primitive. Dato che la pressione nelle regioni a e b del problema di Riemann è la stessa, le due onde non lineari della prima e della terza famiglia sono dello stesso tipo. Attraverso l'utilizzo delle equazioni 3.10 e 3.11 è facile dimostrare che, con questi dati di input, la risoluzione del problema di Riemann dia $u_c = u_d = 0$, come si voleva ottenere. In termini di codice:

```
//hard bc destra
un1= -in[n].velocity();
pn1=in[n].pressure();
an1=in[n].speedOfSound();
in[n+1].setPrimitive(un1,pn1,an1);
riemann::godunov( in[n], in[n+1], 1, 0, interf[n], flux);
moltiplicaVecCost(flux,h);
sommaVec(fluxdiff[n],flux);
```

I risultati ottenuti con questi due metodi coincidono perfettamente. Nel Grafico 3.10 è possibile visualizzare gli effetti della riflessione dell'urto sulla velocità. Tralasciando le differenza visibili nella parte sinistra del Grafico, dovute all'espansione che ha raggiunto i bordi del dominio dove è stata imposta la condizione al contorno di gradiente nullo, è possibile vedere come l'urto, prima di essere riflesso (colore rosso), si stia muovendo verso destra. Viceversa, dopo la riflessione avvenuta a $t \approx 0.28$, l'urto si muove in direzione opposta in maniera simmetrica. Si sottolinea come la discontinuità di contatto non sia ancora stata riflessa; infatti, non è possibile vedere nessuna interazione con l'urto, che è perfettamente simmetrico.

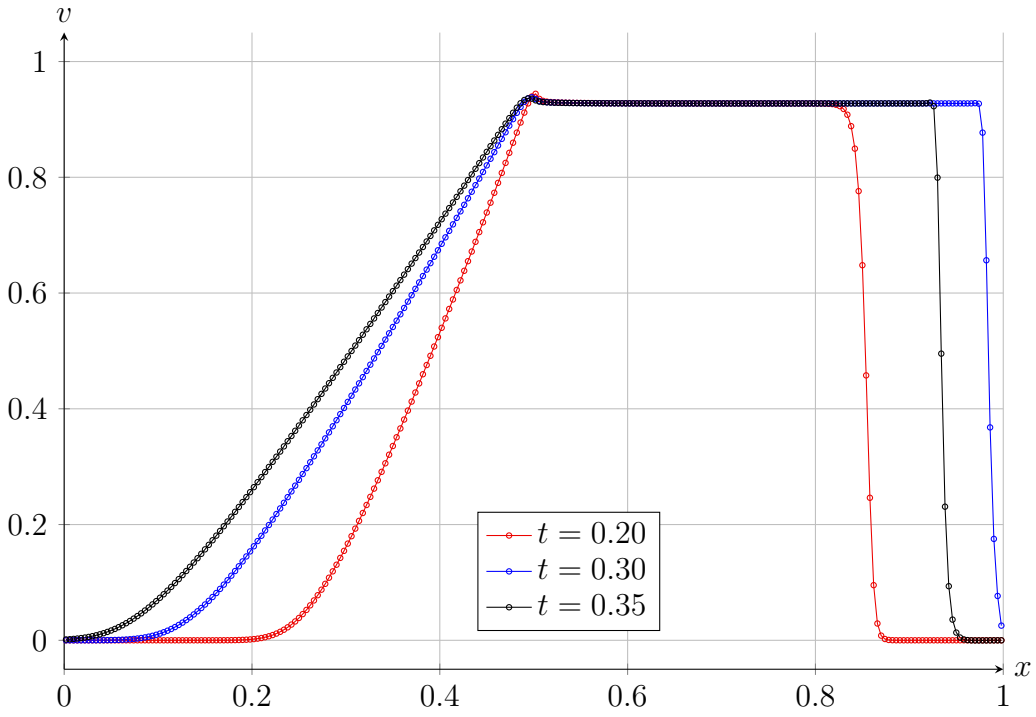


Grafico 3.10: Grafico della velocità prima e dopo la riflessione. $N = 250$, CFL $= 0.4$. L'urto viene riflesso a $t \approx 0.28$

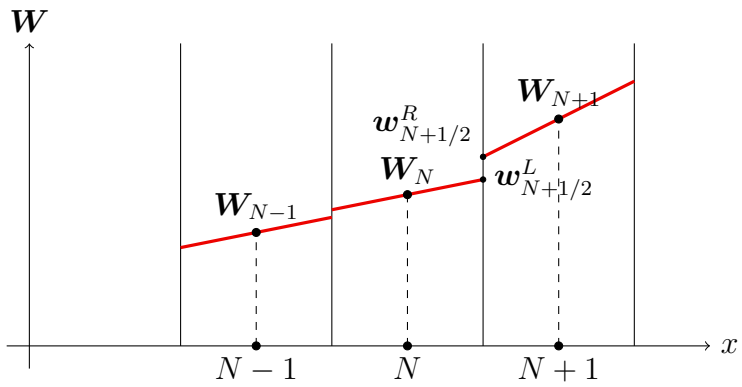


Grafico 3.11: Distribuzione lineare delle variabili conservative.

3.2 Discretizzazione al secondo ordine delle Equazioni di Eulero in forma integrale

Si vuole ora passare da uno schema al primo ordine a uno del secondo ordine nello spazio e nel tempo. Il secondo ordine nello spazio può essere raggiunto passando da una distribuzione costante a tratti delle variabili conservative a una distribuzione lineare (Grafico 3.11). Ovviamente anche in questo caso il valore numerico di cui si tiene conto è la media integrale sulla cella N:

$$\boldsymbol{W}_N = \frac{1}{\Delta x} \int_{x_{N-1/2}}^{x_{N+1/2}} \boldsymbol{W} \, \mathrm{d}x$$

L'andamento lineare consente di avere dei valori ricostruiti ai bordi, diversi dalla media di cella, e dipendenti dalla pendenza σ_N di ogni cella:

$$\begin{cases} \boldsymbol{w}^R_{N+1/2} = \boldsymbol{W}_{N+1} - \boldsymbol{\sigma}_{N+1} \cdot \Delta x/2 & \text{regione } b \text{ del problema di Riemann} \\ \boldsymbol{w}^L_{N+1/2} = \boldsymbol{W}_N + \boldsymbol{\sigma}_N \cdot \Delta x/2 & \text{regione } a \text{ del problema di Riemann} \end{cases}$$

Nel Grafico 3.12 si può vedere come per ogni cella sia possibile definire due pendenze:

$$\boldsymbol{\sigma}^L_N = \frac{\boldsymbol{W}_N - \boldsymbol{W}_{N-1}}{\Delta x} \qquad \boldsymbol{\sigma}^R_N = \frac{\boldsymbol{W}_{N+1} - \boldsymbol{W}_N}{\Delta x} \tag{3.15}$$

La scelta tra queste due possibili pendenze per ogni cella viene fatta attraverso un *limiter*, in questo caso il *minmod*. L'operatore *minmod* consente di scegliere la pendenza minore in valore assoluto, se le due pendenze hanno stesso segno, oppure una pendenza nulla, se le due pendenze hanno segno diverso.

In termini di codice la ricostruzione è stata effettuata sulle variabili primitive, che sono poi state utilizzate per calcolare le variabili conservative all'interfaccia, usate

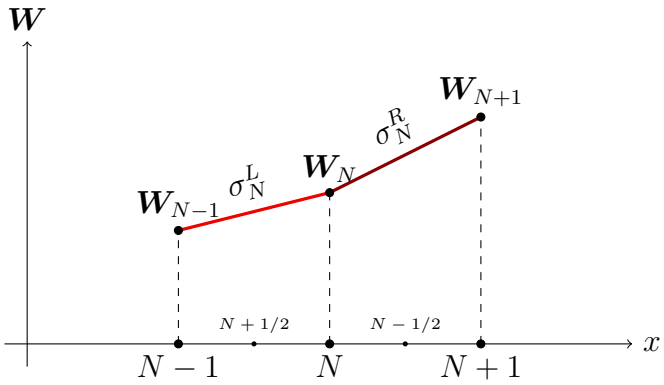


Grafico 3.12: Pendenza destra e sinistra della cella N.

a loro volta come input per il solutore di Riemann e quindi per il calcolo dei flussi, in modo simile a quanto mostrato nella sezione precedente.

```
//calcolo pendenze e ricostruisco valori interfaccia
for (int k=1; k<n+1; k++) {
    if (k==1) {
        calcP(penf[k],in[k],in[k+1],dx);
    }
    if ((k!=1)&&(k!=n)) {
        calcP(penl[k],in[k-1],in[k],dx);
        calcP(penr[k],in[k],in[k+1],dx);
        minmod(penf[k],penl[k],penr[k]);
    }
    if (k==n) {
        calcP(penf[k],in[k-1],in[k],dx);
    }
    reconL(priml[k],in[k],penf[k],dx);
    reconR(primr[k],in[k],penf[k],dx);
    inL[k].setPrimitive(priml[k][0],priml[k][1],priml[k][2]);
    inR[k].setPrimitive(primr[k][0],primr[k][1],primr[k][2]);
}
```

Come si può notare dal listato sopra riportato, ai bordi è stata utilizzata l'unica pendenza disponibile, senza ricorrere all'operatore *minmod*. Le funzioni utilizzate per il calcolo delle pendenze, il *minmod* e la ricostruzione sono:

```
//funzione per il calcolo delle pendenza
void calcP(std::array<double,3> &p,const Fluid &a,
 const Fluid &b,const double dx) {
p[0]=(b.velocity()-a.velocity())/dx;
```

```
p[1]=(b.pressure()-a.pressure())/dx;
p[2]=(b.speedOfSound()-a.speedOfSound())/dx;
}

void minmod(std::array<double,3> &pf, const std::array<double,3> &a,
 const std::array<double,3> &b) {

for (int i=0; i<3; i++) {
    if (a[i]*b[i]<=0) {pf[i]=0;}
    else if (std::abs(a[i])<std::abs(b[i])) {pf[i]=a[i];}
    else if (std::abs(b[i])<std::abs(a[i])) {pf[i]=b[i];}
    else {pf[i]=a[i];}
}
}

// funzione ricostruzione interfaccia sinistra
void reconL(std::array<double,3> &wl,const Fluid &a,
 const std::array<double,3> &pf, const double dx) {
priml[0]=a.velocity()-pf[0]*dx/2;
priml[1]=a.pressure()-pf[1]*dx/2;
priml[2]=a.speedOfSound()-pf[2]*dx/2;
}
```

Per ottenere uno schema al secondo ordine nel tempo non è più possibile utilizzare l'Equazione 3.7; si sceglie invece di utilizzare il metodo di Runge-Kutta 2:

$$\begin{cases} \boldsymbol{W}_N^{\star} = \boldsymbol{W}_N^K - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^N(x_{N+1/2}) - \boldsymbol{F}^N(x_{N-1/2})\right) \\ \boldsymbol{W}_N^{K+1} = \frac{1}{2}\boldsymbol{W}_N^K + \frac{1}{2}\left[\boldsymbol{W}_N^{\star} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^{\star}(x_{N+1/2}) - \boldsymbol{F}^{\star}(x_{N-1/2})\right)\right] \end{cases} \tag{3.16}$$

Questo metodo viene implementato all'interno del ciclo temporale in questo modo:

```
if (rk==1) { //evoluzione soluzione rk step 2
    for (int z=1; z<n+1; z++) {
        star[z][fid_density]=in[z].density();
        star[z][fid_momentum]=in[z].momentum();
        star[z][fid_energy]=in[z].energy();
        in[z].differenza(fluxdiff[z]);
    }
}
if (rk==2) { //evoluzione soluzione rk step 2
    for (int z=1; z<n+1; z++) {
        in[z].differenza(fluxdiff[z]);
        in[z].somma(star[z]);
```

```
        in[z].moltiplicazioneCost(0.5);
    }
}
```

Nel Grafico 3.13 è possibile visualizzare un confronto tra la soluzione al primo e al secondo ordine. Ovviamente quella al secondo ordine è in grado di catturare in maniera più precisa tutte e tre le onde non lineari.

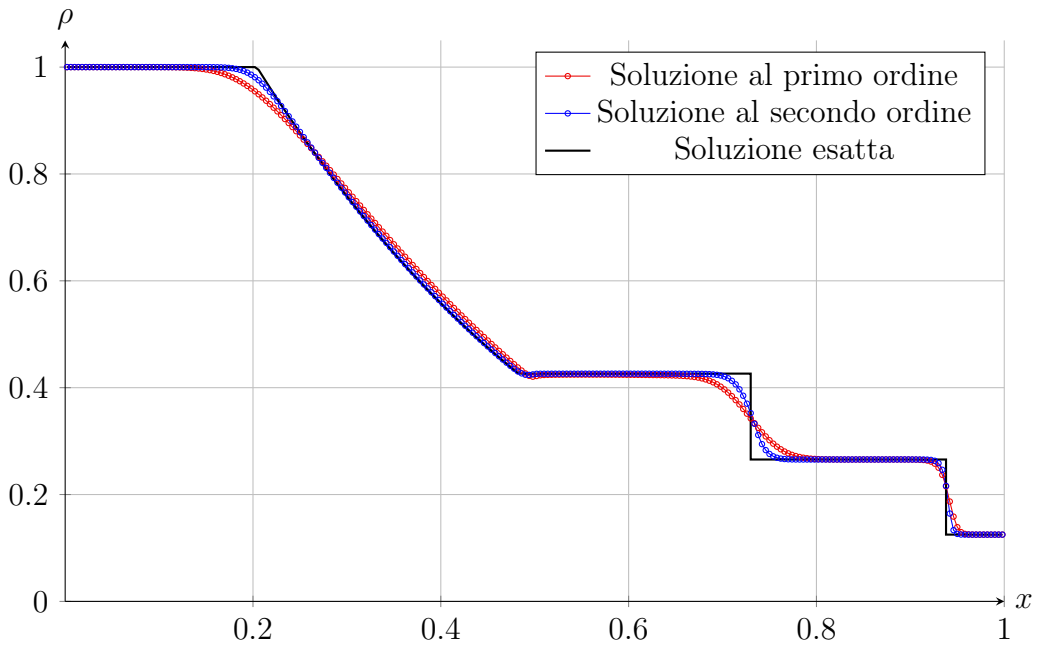


Grafico 3.13: Confronto tra soluzione al primo e al secondo ordine. $t = 0.25$, $N = 250$, CFL $= 0.4$.

3.2.1 Imposizione dell'IB

Si vuole ora introdurre nel codice la possibilità di avere una riflessione dell'urto non solo a $x_w = 1$, ovvero in corrispondenza del bordo destro del dominio, ma per qualsiasi posizione $0.5 < x_w \leq 1$. Data l'eventualità che il boundary imposto non coincida con un'interfaccia tra una cella e l'altra, è necessario utilizzare dei metodi *immersed boundary*.

3.2.1.1 Modifica della velocità della discontinuità di contatto

Il primo metodo utilizzato è quello presentato da Gorsse et al. [17], di cui viene data una spiegazione dettagliata alla fine del Capitolo 2. Il primo passo per l'applicazione del metodo è la localizzazione del boundary imposto, la cui posizione assoluta è un

input del codice e viene indicata con `w`, in termini di celle coinvolte e distanza adimensionale d. Ipotizzando che il boundary cada tra il centro cella N e il centro cella $N+1$, la cella $N+1$ diventa la *ghost cell* nel quale sarà necessario correggere le variabili conservative in modo da modificare il problema di Riemann sull'interfaccia $N+1/2$. Si riporta nuovamente l'immagine tratta da [17] per chiarezza (Figura 3.3).

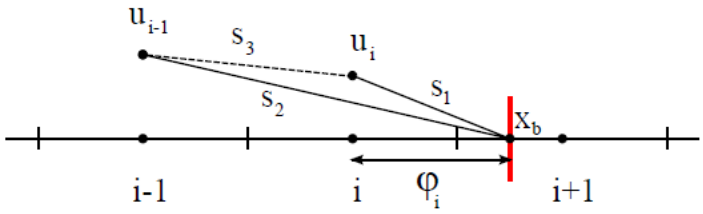


Figura 3.3: *Tagging* delle celle. Data la presenza di x_w tra il centro cella i e quello $i+1$, la cella $i+1$ diventa la *ghost cell* e viene utilizzata per imporre $u^\star$ come velocità della discontinuità di contatto in $i+1/2$. [17]

In termini di codice:

```
//tagging
double phi=1000000;
int tphi; //indice ultima cella fluido (ghost cell-1)
for (int t=1; t<n+1; t++) {
    if (std::abs(cellcenter[t]-w)<phi) tphi=t;
    phi=std::min(std::abs(cellcenter[t]-w),phi);
}
if (w<=cellcenter[tphi]) {
    phi=dx-phi;
    tphi=tphi-1;
}
d=phi/dx; //distanza adimensionata
```

Dopo il *tagging* è possibile procedere con la modifica del problema di Riemann all'interfaccia $N+1/2$:

```
sbi=ub-in[tphi].velocity()+(1.0-d)/(1.0+d)*(ub-
  in[tphi-1].velocity());
s3=in[tphi].velocity()-in[tphi-1].velocity();
minmod(sb,sbi,s3);
ucont=ub+(0.5-d)*sb;

ugh= -inR[tphi].velocity()+2.0*ucont;
pgh=inR[tphi].pressure();
```

```
agh=inR[tphi].speedOfSound();
in[tphi+1].setPrimitive(ugh,pgh,agh);
riemann::godunov( inR[tphi], in[tphi+1], 1, 0,
 interf[tphi], flux);
```

In questo modo si impone la presenza di una parete, in grado di riflettere l'urto, nella posizione x_w. Si sottolinea come a questo punto non sia più necessario risolvere tutte le celle il cui centro sia $x_N > x_w$; esse, infatti, non faranno più parte del fluido e potranno essere disaccoppiate dalla risoluzione del resto del dominio. Tutti i cicli spaziali utilizzati nel codice possono fermarsi alla cella con indice `tphi`. Questo è proprio uno dei vantaggi dei metodi IB di tipo *discrete forcing approach*. Nel caso in cui il boundary imposto in x_w coincida con un'interfaccia tra una cella e l'altra, si avrebbe $d = 0.5$:

$$\begin{cases} u^\star = u_b + \left(\frac{1}{2} - d\right) s_b \\ d = 0.5 \quad u_b = 0 \end{cases} \quad \Rightarrow u^\star = 0$$

Quindi gli stati destro e sinistro del problema di Riemann sarebbero $U_+ = ((-u_- + 2u^\star) = -u_-, p_-, a_-)$ e $U_- = (u_-, p_-, a_-)$, che coincidono perfettamente con quanto fatto nella sezione precedente per imporre la riflessione dell'urto al bordo del dominio.

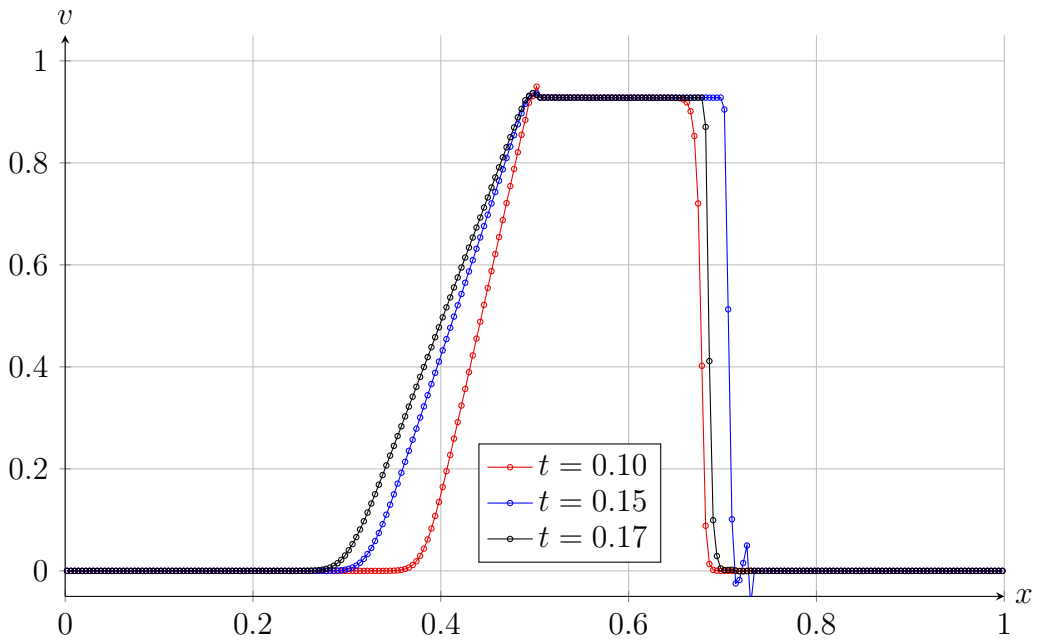


Grafico 3.14: Grafico della velocità prima e dopo la riflessione IB. $N = 250$, CFL $= 0.4$, $w = 0.727$. L'urto viene riflesso a $t \approx 0.14$.

Nel Grafico 3.14 è possibile visualizzare i risultati di una simulazione con un urto riflesso a $x_w = 0.727$. Si vede come l'urto all'istante $t = 0.1$ (in rosso) si muova

verso destra e venga successivamente riflesso a $t \approx 0.14$. Dopo qualche istante in cui la soluzione è oscillante (in blu), questa si stabilizza e continua a muoversi verso sinistra (in nero, $t = 0.17$).

3.2.1.2 Metodo Ghost-Cell

Il secondo metodo utilizzato è stato scritto ispirandosi ai *ghost-cell methods* descritti nel Capitolo 2. Una volta definita la posizione del boundary x_w, il *tagging* viene effettuato in questo modo:

$$x_{N-1/2} < x_w \leq x_{N+1/2} \Rightarrow N+1 \text{ è la ghost cell}$$

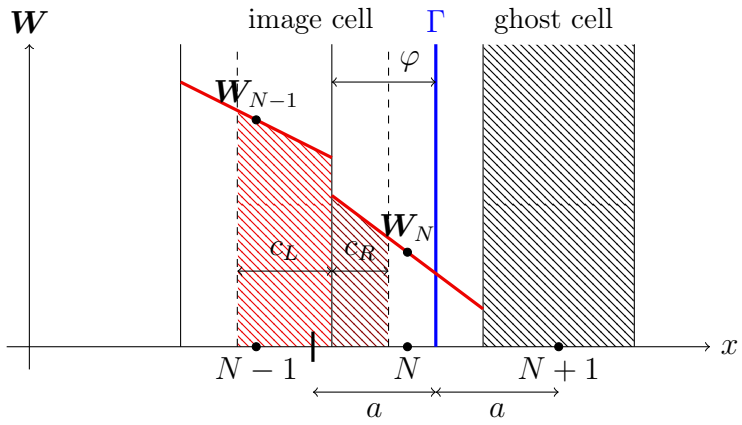


Grafico 3.15: Visualizzazione dell'*image cell*, ovvero la *ghost cell* specchiata rispetto alla posizione dell'IB. Sono evidenziate anche le aree utilizzate per il calcolo del valore medio integrale delle variabili conservative nell'*image cell*.

Come è possibile vedere nel Grafico 3.15, la *ghost cell* scelta viene specchiata all'interno del dominio identificando l'*image cell*. I valori delle variabili dell'*image cell* vengono calcolati come l'integrale della distribuzione all'interno delle celle interessate, ad esempio le celle N e $N-1$ nel Grafico 3.15. Identificando con A_1 l'area sottesa dalla distribuzione in $N-1$ (in rosso chiaro), e A_2 quella sottesa dalla distribuzione in N (in rosso scuro), il valore delle variabili dell'*image cell* è semplicemente:

$$\boldsymbol{W}_i = \frac{\boldsymbol{A}_1 + \boldsymbol{A}_2}{\Delta x} \tag{3.17}$$

Una volta trovati i valori delle variabili all'interno dell'*image cell*, i valori delle variabili conservative da imporre nella *ghost cell* sono, come già spiegato nell'equazione 2.15:

$$a_{GC} = a_{IC} \qquad p_{GC} = p_{IC} \qquad u_{GC} = -u_{IC}$$

Non è però sufficiente trovare solamente i valori delle variabili all'interno della *ghost cell*, che si ricordano essere medie di cella, è necessario trovare anche i valori delle pendenze da utilizzare per ricostruire i valori della *ghost cell* all'interfaccia $N+1/2$, che saranno utilizzati come input del solutore di Riemann. Facendo riferimento ai Grafici 3.15 e 3.16, le pendenze all'interno dell'*image cell* vengono calcolate come media pesata tra le pendenze delle due celle coinvolte nella media integrale:

$$\boldsymbol{\sigma}_i = \frac{c_L \boldsymbol{\sigma}_{N-1} + c_R \boldsymbol{\sigma}_N}{\Delta x} \tag{3.18}$$

Le pendenze della *ghost cell* vengono poi ottenute da quelle appena calcolate in questo modo:

$$\sigma^a_{GC} = -\sigma^a_{IC} \qquad \sigma^p_{GC} = -\sigma^p_{IC} \qquad \sigma^u_{GC} = \sigma^u_{IC}$$

Con questa scelta, nel caso in cui il boundary x_w cada in corrispondenza di un'interfaccia tra due celle, si recupera la configurazione esatta di *reflective BC* imposta nella sezione precedente (Equazione 3.14), in cui viene definito un problema di Riemann all'interfaccia i cui dati di input sono $U_+ = (-u_-, p_-, a_-)$ e $U_- = (u_-, p_-, a_-)$. Questo ragionamento è chiaramente visibile nel Grafico 3.16.

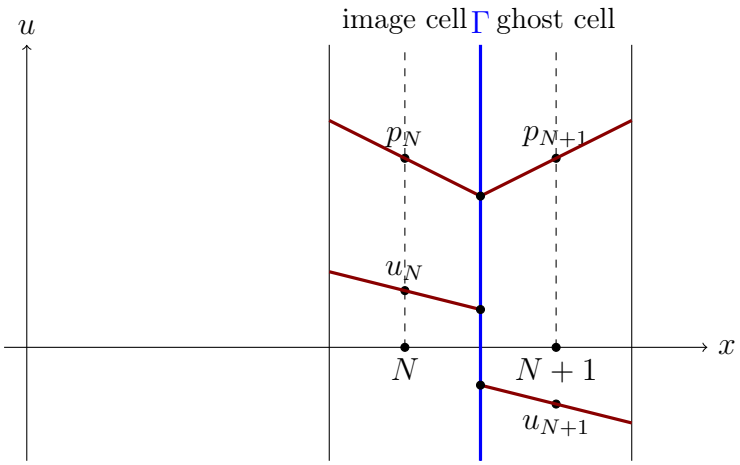


Grafico 3.16: Pendenze utilizzate per la *ghost cell* rispetto a quelle utilizzate per l'*image cell*.

Per motivi di lunghezza, il codice con il quale è stato implementato questo metodo viene riportato in Appendice A.

I risultati sono estremamente simili a quanto mostrato in precedenza nel Grafico 3.14, perciò si passa direttamente al confronto tra questi due metodi.

3.2.1.3 Confronto tra i due metodi

Per ognuno dei due metodi sopra esposti è stato effettuato il calcolo dell'errore derivante dall'utilizzo di un IB. Infatti, come mostrato precedentemente, nel caso in

cui il boundary imposto coincida con un'interfaccia tra due celle, il metodo risulta esatto e si recupera la condizione al contorno classica (Equazione 3.14); viceversa, nel caso in cui il boundary sia all'interno di una cella si avrà un certo errore. Come soluzione di riferimento non è stata utilizzata la soluzione esatta legata all'urto riflesso, bensì una soluzione raffinata 10 volte rispetto a quella di partenza.

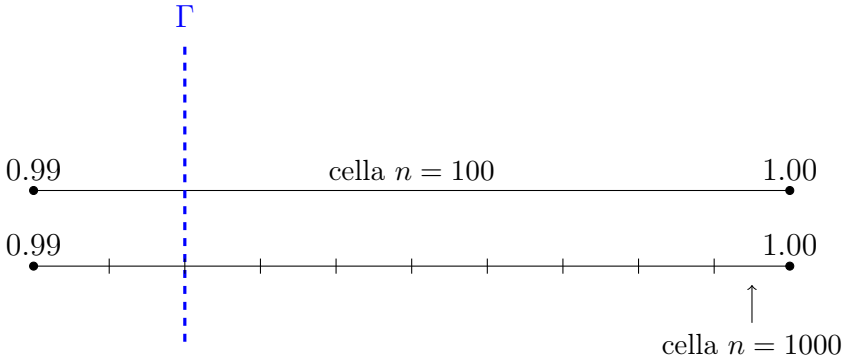


Grafico 3.17: Posizione del boundary sulla griglia meno raffinata con $n = 100$ e su quella di riferimento con $n = 1000$. Il boundary cade sempre all'interno dell'ultima cella sulla griglia meno raffinata, mentre giace sulle varie interfacce sulla griglia di riferimento.

Come si può vedere chiaramente nel Grafico 3.17, una griglia con $n = 1000$ punti è stata usata come riferimento, in questo modo è stato possibile far variare il boundary tra $0.99 \leq x_w \leq 1.00$ a step di $\Delta x_w = 0.001$. La soluzione su una griglia con $n = 100$ punti, dove x_w è sempre posizionato all'interno dell'ultima cella, è stata comparata con quella di riferimento, calcolando lo scarto quadratico medio su ogni cella. Ovviamente dovendo comparare soluzioni con diverso numero di celle, è stato necessario mediare i risultati ottenuti sulla griglia di riferimento a gruppi di 10 celle (Grafico 3.18). Da questa media, in prossimità dell'IB, è stato necessario escludere i valori delle celle appartenenti al solido, non facenti parte della soluzione.

Volendo paragonare gli errori per diverse posizioni di x_w, è anche necessario tenere conto di un altro fattore. Nei codici utilizzati fino a questo momento infatti sono state utilizzate le condizioni iniziali di Sod, con la discontinuità applicata a $x = 0.5$. In questo modo però l'urto percorre distanze diverse in base alla posizione di x_w. Per poter comparare correttamente casi con x_w diverse è necessario cambiare le condizioni iniziali in modo da imporre che l'urto percorra sempre la stessa distanza prima di essere riflesso:

```
//origine urto
double wwrel=10000000;
int twwrel;
double ww=w-0.5;
```

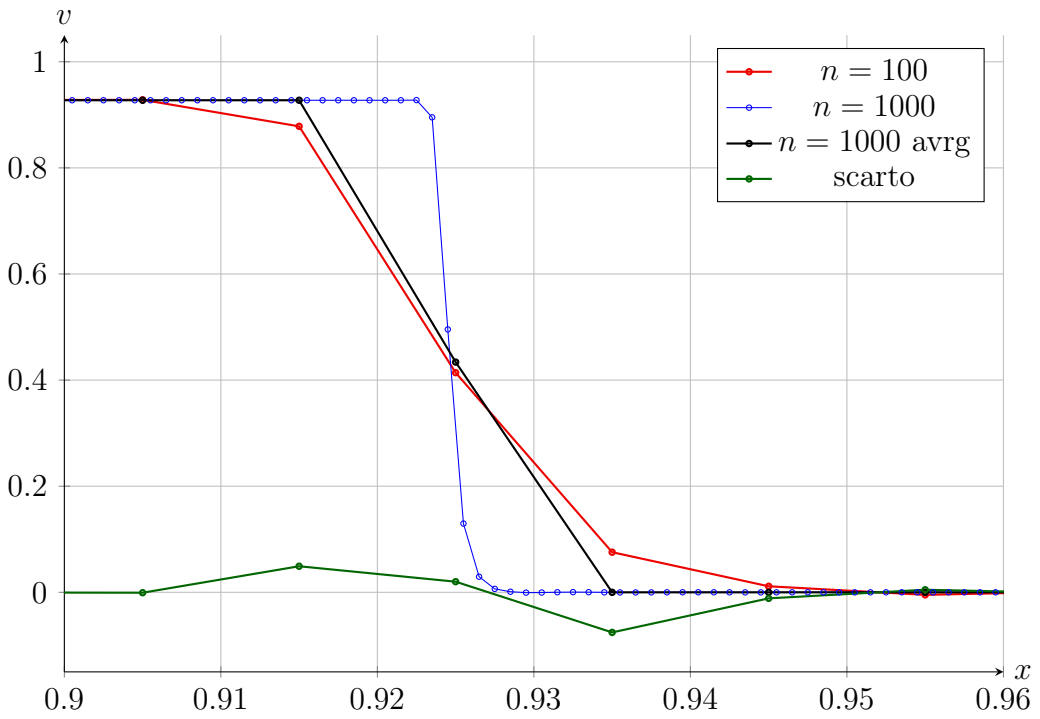


Grafico 3.18: Confronto tra la soluzione meno raffinata con $n = 100$, quella di riferimento con $n = 1000$ e la sua media ogni 10 celle. Viene anche rappresentato lo scarto tra le due soluzione du 100 punti. CFL $= 0.5$, $w = 0.99$, $t_f = 0.35$.

```
for (int t=1; t<n+1; t++) {
    if (std::abs(cellcenter[t]-dx/2.0-ww)<wwrel) twwrel=t;
    wwrel=std::min(std::abs(cellcenter[t]-dx/2.0-ww),wwrel);
}
if (ww<=(cellcenter[twwrel]-dx/2.0)) {
    wwrel=dx-wwrel;
    twwrel=twwrel-1;
}

//fluid initialization
Fluid in[n+2], interf[n+2], inL[n+2], inR[n+2];
double uww,pww,rhoww,aww;
for (int i=0; i<n+2; i++) {
    in[i].setSpecificHeatRatio(gam);
    interf[i].setSpecificHeatRatio(gam);
    inL[i].setSpecificHeatRatio(gam);
```

```
    inR[i].setSpecificHeatRatio(gam);
    if (i<twwrel) in[i].setPrimitive(ul,pl,al);
    else if (i==twwrel) {
        uww=(wwrel*ul+(dx-wwrel)*ur)/dx;
        pww=(wwrel*pl+(dx-wwrel)*pr)/dx;
        rhoww=(wwrel*rhol+(dx-wwrel)*rhor)/dx;
        aww=std::sqrt(gam*pww/rhoww);
        in[i].setPrimitive(uww,pww,aww);
    }
    else in[i].setPrimitive(ur,pr,ar);
}
```

Essendo la posizione della discontinuità iniziale all'interno di una cella e non coincidente con un'interfaccia, è opportuno fare una media integrale per ottenere la condizione iniziale della cella interessata, in maniera del tutto analoga a quanto fatto con l'Equazione 3.14.

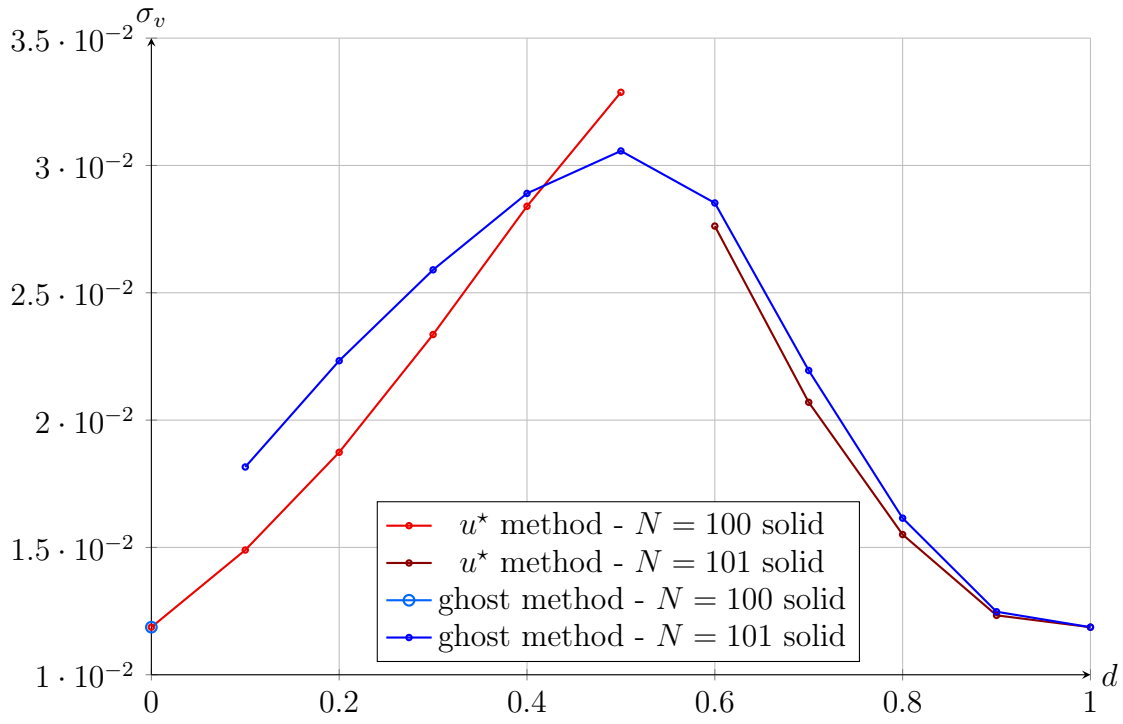


Grafico 3.19: Confronto tra metodi utilizzando lo scarto quadratico medio sulla velocità al variare della posizione del boundary $0.99 \leq x_w \leq 1 \Rightarrow 0 \leq d \leq 1$.

Si possono quindi calcolare gli scarti quadratici medi tra la soluzione meno raffina-

ta e quella di riferimento, al variare della posizione dell'IB. Gli errori sulla velocità, sulla pressione e sulla densità sono visibili rispettivamente nei Grafici 3.19, 3.20 e 3.21.

Si può innanzitutto notare come gli errori per $x_w = 0.99$ e $x_w = 1.00$ siano quelli minori (e uguali tra loro); infatti, quando il boundary è in queste due posizioni, esso coincide con l'interfaccia tra due celle nella griglia meno raffinata. In questo caso la condizione al contorno imposta col metodo IB è esatta e l'errore presente deriva solamente dalla discretizzazione più accurata della griglia di riferimento. In questi punti le soluzioni fornite dai due schemi coincidono perfettamente, infatti nel caso in cui $d = \varphi_N/\Delta x = 0$ essi decadono nella stessa condizione al contorno, come mostrato nelle sezioni precedenti.

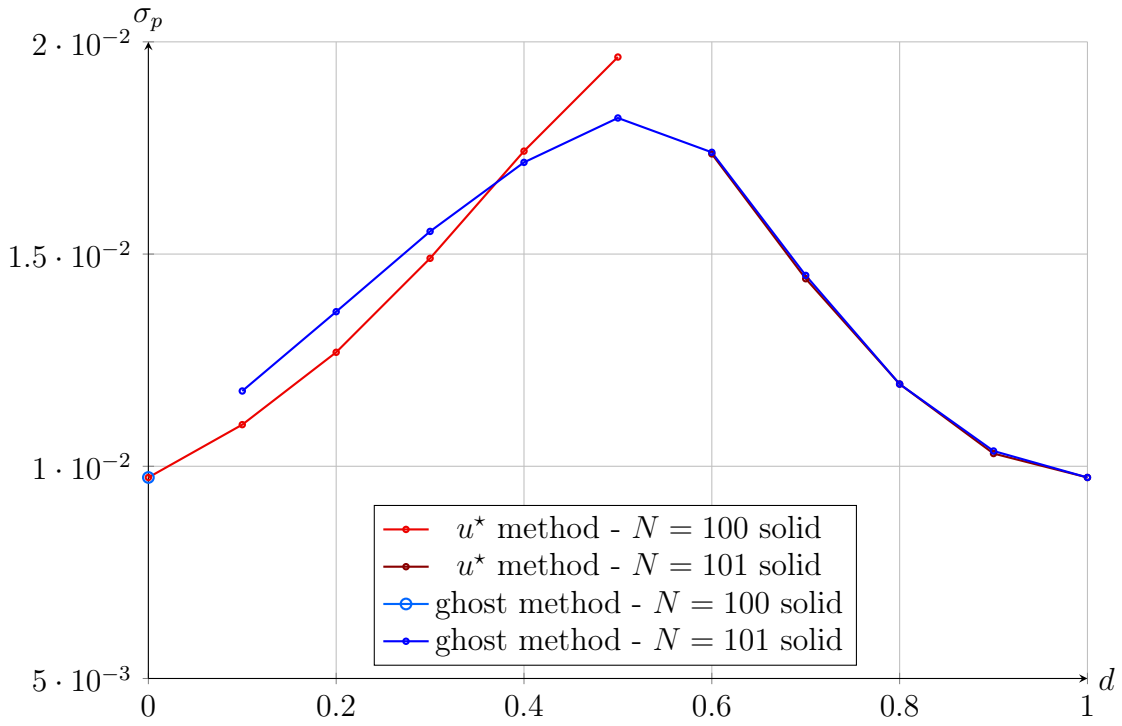


Grafico 3.20: Confronto tra metodi utilizzando lo scarto quadratico medio sulla pressione al variare della posizione del boundary $0.99 \leq x_w \leq 1 \Rightarrow 0 \leq d \leq 1$.

In tutti i grafici i risultati per ogni metodo sono stati divisi in due linee separate, per rappresentare la differenza di *tagging* in base alla posizione dell'IB. Per il primo schema, basato sulla modifica locale del problema di Riemann, la *ghost cell* appartenente al solido è inizialmente la $N = 100$ per $x_w \leq 0.995$, per poi diventare la $N = 101$ per $x_w > 0.995$. Questo cambio di *tagging* è visibile nella rappresentazione

dell'errore e deve essere attentamente considerato durante l'utilizzo di metodi simili. Per lo schema basato sull'*image cell* invece, l'unica posizione per cui $N = 100$ è la *ghost cell* è $x_w = 0.99$; in questo caso il salto nell'errore dovuto al *tagging* è ancora più visibile.

Ovviamente all'allontanarsi dell'IB dalle interfacce di cella l'errore cresce sempre di più, anche se sembra raggiungere livelli fin troppo elevati rispetto alla bibliografia di riferimento [17], in particolare per quanto riguarda la velocità. Inoltre l'andamento dell'errore all'avvicinarsi alle interfacce di cella dovrebbe essere simmetrico e di tipo parabolico, essendo presente un minimo a tangente orizzontale in questi punti. Nel grafico 3.19 dell'errore sulla velocità questo non è vero per $d \to 0$. Le differenze tra i due metodi sono comunque molto ridotte in ogni grafico.

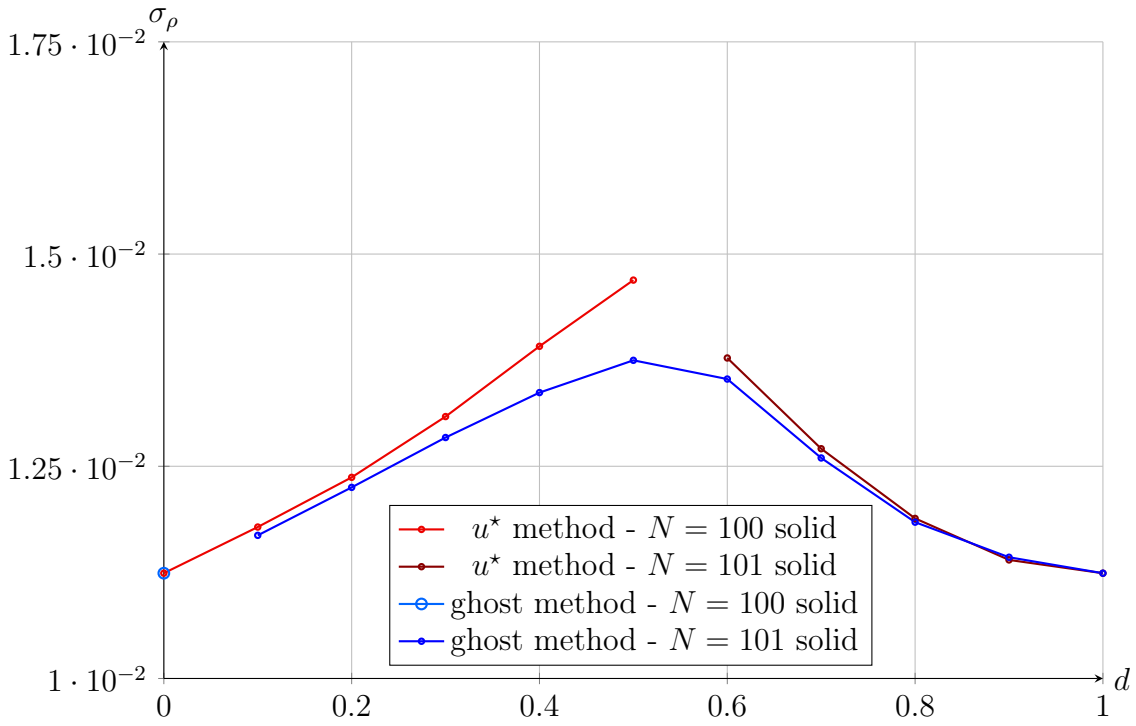


Grafico 3.21: Confronto tra metodi utilizzando lo scarto quadratico medio sulla densità al variare della posizione del boundary $0.99 \leq x_w \leq 1 \Rightarrow 0 \leq d \leq 1$.

Le differenze presenti rispetto ai risultati ottenuti in bibliografia potrebbero essere causate da vari motivi. Innanzitutto, la differenza nella soluzione di riferimento; per questi test è stata utilizzata una soluzione molto raffinata e non la soluzione esatta, ottenibile analiticamente in caso di riflessione di un urto. In secondo luogo, la discretizzazione dell'IB per $0 \leq d \leq 1$; che in questo caso è stata fatta solamen-

te con 10 punti, probabilmente non riesce a rappresentare abbastanza fedelmente l'andamento dell'errore. I risultati ottenuti sono comunque più che sufficienti ad evidenziare le caratteristiche di base di un *immersed boundary method*, di cui è stato dato un esempio applicativo al semplice caso del tubo d'urto unidimensionale.

4 Validazione del codice IB ImmerFlow in regime comprimibile

In questo Capitolo viene analizzato un codice di calcolo sviluppato dalla Optimad Engineering srl di nome **ImmerFlow**. Questo codice permette l'analisi di flussi in regime comprimibile e incomprimibile e utilizza metodi *immersed boundary* per la rappresentazione di corpi immersi in griglie cartesiane.

Dopo una prima analisi del codice e della sua struttura, viene presentato un caso test per la validazione del codice in regime ipersonico in cui è presente un'interazione shock-shock di tipo Edney IV [13].

4.1 Struttura del codice e funzionamento

ImmerFlow è un codice ai volumi finiti la cui principale caratteristica è l'utilizzo di una griglia *octree* accoppiata a metodi *immersed boundary*. Il codice studiato utilizza al suo interno una libreria C++ open source di nome **bitpit**; questa è costituita da diversi moduli che spaziano da nuovi operatori e containers, a metodi per la risoluzione di sistemi lineari, alla gestione dell'*Input Output* (lettura e scrittura di DGF, STL e VTK), a moduli per la gestione e discretizzazione di geometrie, incluso l'utilizzo della funzione *level set* già introdotta nel Capitolo 2.

4.1.1 Generazione della griglia - PArallel Balanced Linear Octree

La generazione della griglia viene fatta da un modulo di **bitpit** denominato **PABLO**, ovvero *PArallel Balanced Linear Octree*. In generale con la parola *octree* si indica una generica struttura di dati in cui un *parent* viene suddiviso in 8 *children* per ogni livello; questa struttura viene utilizzata nel caso in cui si abbia uno spazio 3D, mentre viene sostituita dalla *quadtree* in uno spazio 2D (Figura 4.1). Attraverso l'utilizzo di questa struttura di dati è possibile discretizzare un generico dominio Ω nella maniera più adeguata al caso studiato, raffinando la griglia dove necessario.

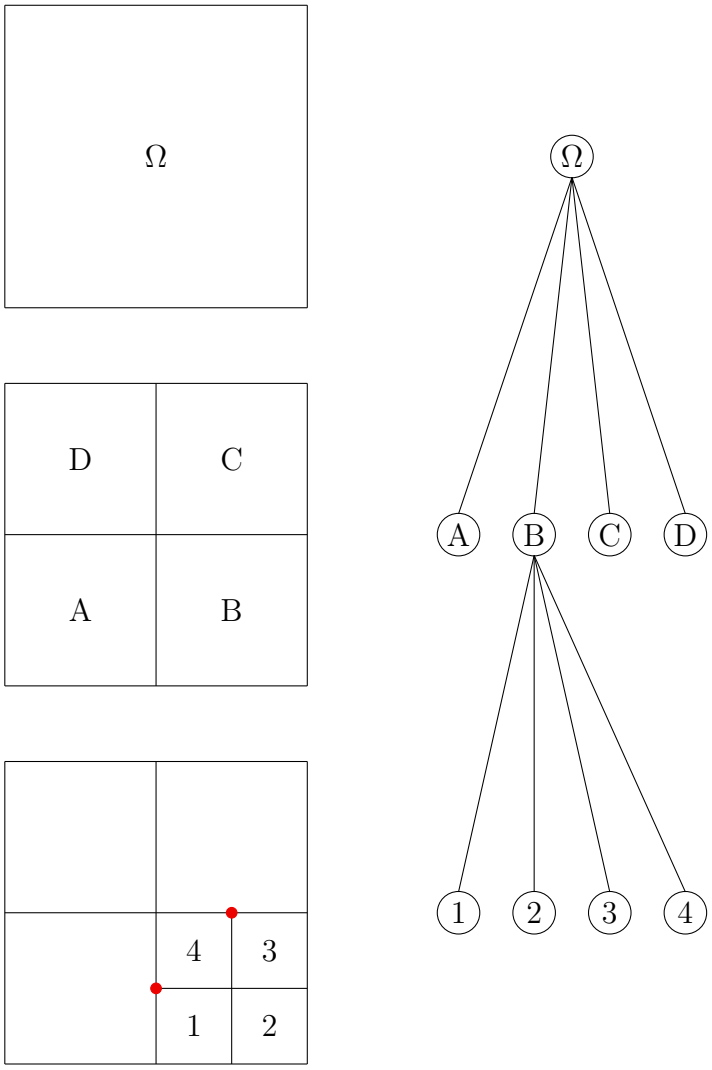


Figura 4.1: Discretizzazione del dominio *quadtree*. Ogni ottante può essere diviso a sua volta in 4 ottanti *children* ad ogni livello (per la *quadtree*). La suddivisione deve essere completa, ogni *parent* deve avere esattamente 4 *children*.

spazio	Vol	Surf
3D	3D	2D
2D	2D	1D

La mesh superficiale (che può essere 2D o 1D in base allo spazio di riferimento) viene gestita attraverso il modulo **SurfUnstructured**, mentre la mesh di volume viene gestita dai moduli **VolCartesian** o **VolOctree**, rispettivamente nel caso di mesh cartesiana uniforme o *octree/quadtree*.

La struttura dati di tipo *tree* viene abbandonata in favore di una rappresentazione lineare in cui ogni *leaf* è un membro di un vettore unidimensionale; questo processo viene fatto attraverso l'uso della *space fitting curve* di Morton.

In **PABLO** un generico dominio Ω può essere raffinato al massimo per 20 livelli e si può imporre il massimo salto di livello accettabile all'interno del dominio (i due punti rossi in Figura 4.1). Solitamente viene imposto un salto di un solo livello ($2:1$ *balancing between octants*), in modo da avere schemi robusti. Questo vuol dire che, nel caso in cui si voglia raffinare ancora di un livello l'ottante 4, anche gli ottanti A e C dovranno essere raffinati di un livello, in modo tale da non avere un salto troppo elevato. Tutte le operazioni descritte fin'ora possono essere gestite in parallelo attraverso l'utilizzo del protocollo MPI; nel momento in cui sia necessario utilizzare più thread **PABLO** è in grado di bilanciare il carico computazionale in base alla griglia considerata.

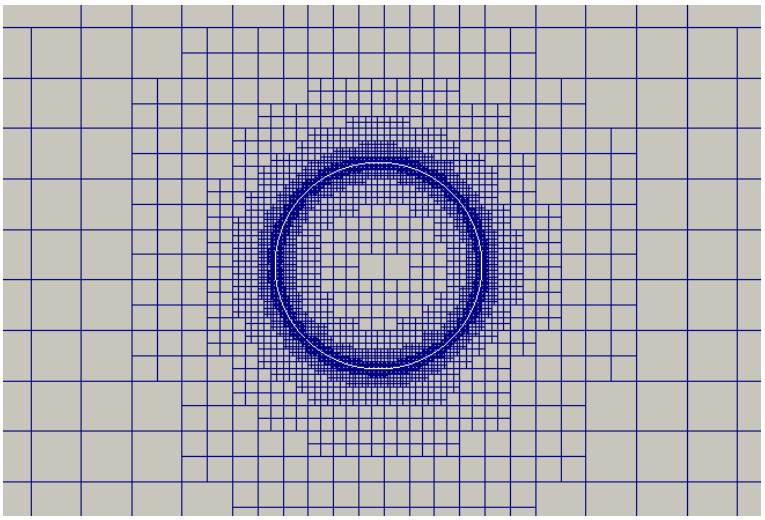


Figura 4.2: Esempio di mesh 2D di tipo *quadtree*. Il vincolo imposto è il raggiungimento di una taglia minima dell'ottante in prossimità del perimetro del cerchio bianco.

In Figura 4.2 è possibile vedere un esempio in cui **bitpit** è stato utilizzato per discretizzare un dominio (che in partenza deve essere quadrato) 2D in modo tale da raggiungere una certa dimensione *target* degli ottanti in prossimità di una superficie circolare; si ricorda come in uno spazio 2D le superfici, secondo il linguaggio di **PABLO**, siano delle geometria 1D definite da file DGF, ovvero un elenco di punti con la corrispondente matrice di connessione. In Figura 4.3 è possibile vedere un particolare della Figura precedente in cui sono ben visibili i differenti livelli a cui gli ottanti appartengono e il modo in cui sono distribuiti.

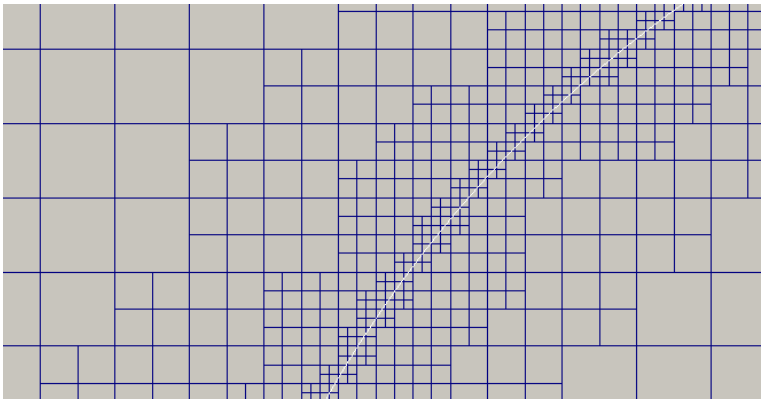


Figura 4.3: Dettaglio di una mesh 2D di tipo *quadtree*. Si possono vedere 4 diversi livelli di mesh; la lunghezza del lato dell'ottante del livello più piccolo è almeno della stessa taglia imposta sul perimetro del cerchio.

4.1.2 Approccio numerico

Verrà ora esposto l'approccio numerico utilizzato da **ImmerFlow** nel caso di analisi di un flusso compressibile viscoso. Le equazioni di Navier-Stokes:

$$\frac{\partial}{\partial t}\int_V \rho \,\mathrm{d}V + \int_S \rho \boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{4.1}$$

$$\frac{\partial}{\partial t}\int_V \rho \boldsymbol{v}\,\mathrm{d}V + \int_S \rho \boldsymbol{v}\boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S + \int_S p\overline{\boldsymbol{I}}\cdot\boldsymbol{n}\,\mathrm{d}S - \int_S \overline{\boldsymbol{\tau}}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{4.2}$$

$$\frac{\partial}{\partial t}\int_V E\,\mathrm{d}V + \int_S (E+p)\boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S - \int_S (\overline{\boldsymbol{\tau}}\cdot\boldsymbol{v})\cdot\boldsymbol{n}\,\mathrm{d}S + \int_S \dot{\boldsymbol{q}}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{4.3}$$

dove $\overline{\boldsymbol{I}}$ è la matrice identità, $\overline{\boldsymbol{\tau}}$ il tensore degli sforzi e $\dot{\boldsymbol{q}}$ il flusso di calore:

$$\tau_{ij} = \mu\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial u_j}{\partial x_j}\right] \tag{4.4}$$

$$\dot{\boldsymbol{q}} = -k\nabla T \tag{4.5}$$

possono essere scritte nella forma integrale conservativa:

$$\frac{\partial}{\partial t}\int_V \boldsymbol{W}\,\mathrm{d}V + \int_S \boldsymbol{F}_C \cdot \boldsymbol{n}\,\mathrm{d}S + \int_S \boldsymbol{F}_V \cdot \boldsymbol{n}\,\mathrm{d}S = 0 \tag{4.6}$$

dove $\boldsymbol{W}$ è il vettore delle variabili conservative, $\boldsymbol{F}_C$ il vettore dei flussi convettivi e $\boldsymbol{F}_V$ il vettore dei flussi viscosi, contenente i termini derivanti dal tensore degli sforzi e dal flusso di calore.

Si suppone che il gas considerato sia un gas perfetto, in particolare aria, per cui risulta valida la legge di stato:

$$\frac{p}{\rho} = \frac{\mathcal{R}}{\mathcal{M}}T \tag{4.7}$$

La viscosità dinamica viene calcolata attraverso la legge di Sutherland e la conducibilità termica deriva da quest'ultima attraverso la definizione del numero di Prandtl:

$$Pr = \mu\frac{c_p}{k} = 0.72 \tag{4.8}$$

Integrando la forma conservativa su una griglia cartesiana 2D si ottiene:

$$\frac{\mathrm{d}\boldsymbol{W}_{ij}}{\mathrm{d}t}\Delta V_{ij} + \sum_{j=1}^{4} \boldsymbol{F}_{C_j} \cdot \boldsymbol{n}_j \Delta S_j + \sum_{j=1}^{4} \boldsymbol{F}_{V_j} \cdot \boldsymbol{n}_j \Delta S_j = 0 \tag{4.9}$$

dove $\boldsymbol{W}_{ij}$ è la media integrale delle variabile conservative sulla cella considerata e i flussi vengono calcolati sulle 4 facce j che circondano la cella i. Uno schema Runge-Kutta di secondo ordine (metodo di Heun) viene utilizzato per l'integrazione nel tempo:

$$\begin{cases} \boldsymbol{W}^\star = \boldsymbol{W}^n - \Delta t \boldsymbol{R}(\boldsymbol{W}^n) \\ \boldsymbol{W}^{n+1} = \boldsymbol{W}^n - \Delta t/2(\boldsymbol{R}(\boldsymbol{W}^n) + \boldsymbol{R}(\boldsymbol{W}^\star)) \end{cases} \tag{4.10}$$

dove $\boldsymbol{R}(\boldsymbol{W}^n)$ è il RHS dell'Equazione 4.9, funzione delle variabili conservative al time step n.

Le variabili primitive vengono ricostruite all'interfaccia secondo uno schema MUSCL [37] e questi valori vengono utilizzati come input del problema di Riemann per il calcolo dei flussi convettivi. Il solutore di Riemann utilizzato può essere esatto (Capitolo 3) oppure basato sul *flux vector splitting* di Van Leer [37].

Per il calcolo dei flussi viscosi è necessario conoscere i gradienti di velocità e temperatura sulle facce su cui si vogliono calcolare i flussi. I gradienti al centro di ogni cella derivano dal processo di minimizzazione dello scarto quadratico tra la ricostruzione delle variabili in una cella e il loro valore medio; ci si riferisce spesso a questo

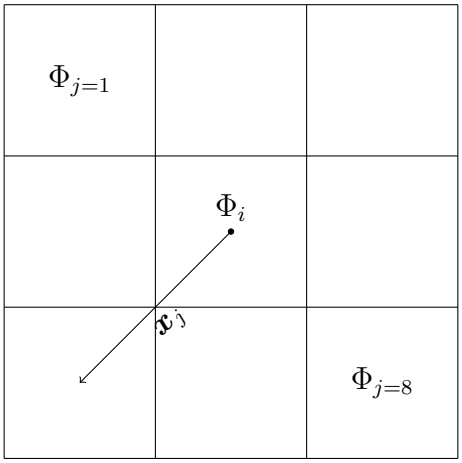


Figura 4.4: Per calcolare il gradiente $\nabla\Phi_i$ è necessario prendere in considerazione anche le celle a essa vicine.

metodo come *gradiente ai minimi quadrati*. Si immagini di voler scrivere l'andamento di una generica variabile Φ all'interno della cella i (Figura 4.4), imponendo un andamento costante a tratti del primo ordine:

$$\Phi^R(\boldsymbol{x}) = \Phi(0)$$

dove $\Phi(0)$ corrisponde alla media integrale della variabile considerata sulla cella i, ed è un valore conosciuto, e $\boldsymbol{x} = \{x, y\}^T$ con l'origine posta nel centro di i. Imponendo invece un andamento lineare del secondo ordine della variabile Φ si ha:

$$\Phi^R(\boldsymbol{x}) = \Phi(0) + \nabla\Phi(0) \cdot \boldsymbol{x}$$

dove $\nabla\Phi = \{\partial\Phi/\partial x, \partial\Phi/\partial y\}^T$. Per trovare il valore incognito del gradiente nel centro della cella i è necessario utilizzare due condizioni. La prima condizione consiste nell'imporre che la media integrale sulla cella i della distribuzione $\Phi^R(\boldsymbol{x})$ corrisponda esattamente al valore medio $\Phi(0)$:

$$\frac{1}{\Omega_i}\int_{\Omega_i} \Phi^R(\boldsymbol{x}) = \Phi(0) \tag{4.11}$$

La seconda condizione invece richiede di calcolare lo scarto tra la media integrale della distribuzione $\Phi^R(\boldsymbol{x})$ sulle varie celle j vicine (Figura 4.4) e il loro valore di media di cella $\Phi(\boldsymbol{x}_j)$. Una volta calcolati questi scarti è possibile calcolare un funzionale $\mathcal{J}$, somma dei quadrati degli scarti, che deve essere minimizzato:

$$\epsilon_j = \frac{1}{\Omega_j}\int_{\Omega_j} \Phi^R(\boldsymbol{x}) - \Phi(\boldsymbol{x}_j) \qquad \mathcal{J} = \frac{1}{2}\sum_{j=1}^{8} \epsilon_j^2 \tag{4.12}$$

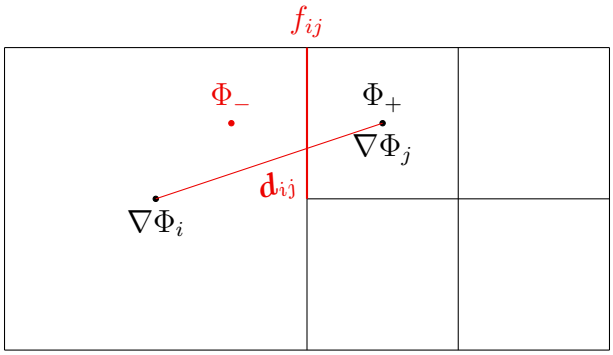


Figura 4.5: Schema necessario al calcolo del gradiente $\nabla\Phi_{f_{ij}}$ in corrispondenza dell'interfaccia tra due celle.

Attraverso questo problema di minimizzazione è quindi possibile ricavare i valori dei gradienti per ogni centro cella. Si devono però ancora trovare i gradienti in corrispondenza della facce, necessari al calcolo dei flussi viscosi.

Si prenda in considerazione una situazione come quella in Figura 4.5, dove è necessario calcolare il gradiente sulla faccia f_{ij} (in rosso), a cavallo di un salto di livello della mesh. Il gradiente in corrispondenza della faccia ij viene così definito:

$$\nabla\Phi|_{f_{ij}} = \boldsymbol{m}(\nabla\Phi_i, \nabla\Phi_j) - (\boldsymbol{m}(\nabla\Phi_i, \nabla\Phi_j) \cdot \boldsymbol{n}_{ij})\, \boldsymbol{n}_{ij} + \frac{\Phi_+ - \Phi_-}{\boldsymbol{x}_+ - \boldsymbol{x}_-}\boldsymbol{n}_{ij} \tag{4.13}$$

dove con $\nabla\Phi_j = \nabla\Phi(\boldsymbol{x}_j)$, $\boldsymbol{d}_{ij}$ è il vettore che unisce i centri cella i e j e $\boldsymbol{m}(\nabla\Phi_i, \nabla\Phi_j)$ è la media convessa tra i due gradienti al centro cella. Di questa media viene tenuta solamente la parte perpendicolare a $\boldsymbol{d}_{ij}$, a cui viene aggiunto il contributo di una differenza finita attraverso l'utilizzo della variabile Φ_-, ricostruita nella posizione specchiata del centro cella j rispetto alla faccia f_{ij}:

$$\Phi_- = \Phi(\boldsymbol{x}_i) + \nabla\Phi(\boldsymbol{x}_i)(\boldsymbol{x}_- - \boldsymbol{x}_i) \tag{4.14}$$

Una volta calcolati i gradienti di velocità e temperatura in corrispondenza delle facce è semplice calcolare i flussi viscosi necessari per l'utilizzo dell'equazione 4.9. A partire dalle condizioni iniziali e da quelle al contorno è possibile evolvere la soluzione dal time step n al time step $n+1$; all'aumentare del numero di iterazioni, a patto che si stia analizzando un problema stazionario, ci si avvicinerà sempre di più alla soluzione finale stazionaria e i residui, costituiti da tutti i termini delle equazioni di Navier-Stokes tranne le derivate nel tempo (quindi la somma dei flussi), si avvicineranno sempre di più allo zero, fino a poter considerare la soluzione arrivata a convergenza.

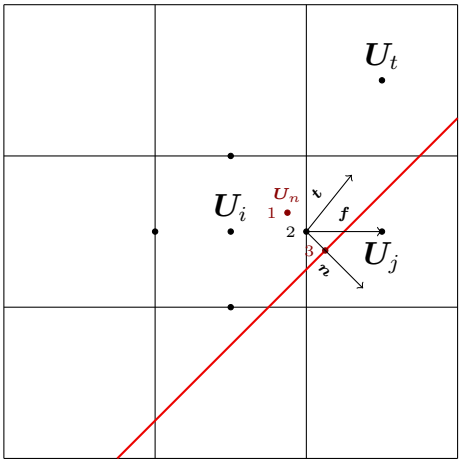


Figura 4.6: Rappresentazione di un *immersed boundary* e dei vettori necessari al calcolo dei flussi sull'interfaccia verticale tra le celle i e j.

4.1.3 Gestione dell'immersed boundary

La presenza degli *immersed boundaries* viene gestita da **ImmerFlow** in maniera simile a quanto spiegato alla fine del Capitolo 2 [17], ma con alcune differenze. Nella seguente trattazione si fa riferimento alla Figura 4.6, dove i vettori $\boldsymbol{n}$ e $\boldsymbol{t}$ sono rispettivamente il vettore normale e tangente all'IB, passante per il centro della faccia ij; il vettore $\boldsymbol{f}$, invece, è il vettore normale alla faccia ij. Una delle differenza maggiori rispetto a quanto elaborato da Gorsse [17] consiste nella procedura di *tagging*; infatti, in questo caso, una cella viene considerata fluida solo se tutti e 5 i suoi punti di collocazione (il centro cella e i punti centrali di ogni faccia) sono all'interno del fluido. Secondo questa procedura di *tagging*, in Figura 4.6 la cella i appartiene al fluido mentre la cella j appartiene al solido.

Si immagini di voler calcolare, inizialmente al primo ordine, il vettore dei flussi convettivi in corrispondenza dell'interfaccia verticale tra la cella i e la cella j:

$$\boldsymbol{F}_{ij}^{1} = \alpha \boldsymbol{F}_n + \beta \boldsymbol{F}_t \tag{4.15}$$

dove $\alpha = \boldsymbol{n} \cdot \boldsymbol{f}$ e $\beta = \boldsymbol{t} \cdot \boldsymbol{f}$. Viene quindi fatta una media pesata tra una componente normale del flusso, che terrà conto della condizione al contorno da imporre, e una componente tangenziale, semplicemente estrapolata dal flusso, in maniera del tutto simile a quanto fatto da Gorsse [17]. La componente normale del flusso $\boldsymbol{F}_n$ viene calcolata, al primo ordine, come il flusso derivante dalla classica imposizione della parete nel semi problema di Riemann (Equazione 3.13), partendo dal valore al centro cella:

$$\boldsymbol{F}_n = \boldsymbol{F}_{BC}(\boldsymbol{n}, \boldsymbol{U}_i) \tag{4.16}$$

La componente tangenziale del flusso viene calcolata attraverso la soluzione del problema di Riemann utilizzando come input i valori al centro cella $\boldsymbol{U}_i$ e $\boldsymbol{U}_t$, dove

la cella t è la cella con il centro più vicino alla retta tangente all'IB identificata dal vettore $\boldsymbol{t}$:

$$\boldsymbol{F}_t = \boldsymbol{F}^{\vee}(\boldsymbol{t}, \boldsymbol{U}_i, \boldsymbol{U}_t) \tag{4.17}$$

Questa procedura consente di imporre la presenza dell'IB solamente al primo ordine. Se si volesse passare al secondo ordine sarebbe necessario introdurre un termine correttivo; questo non è niente altro che la componente normale del flusso, calcolata questa volta utilizzando un valore ricostruito $\boldsymbol{U}_n$ nel punto corrispondente all'*image point* dell'intersezione tra l'IB e il vettore $\boldsymbol{n}$ rispetto al centro della faccia ij (si veda la Figura 4.6):

$$\boldsymbol{F}_n^{\star} = \boldsymbol{F}^{\vee}(\boldsymbol{n}, \boldsymbol{U}_n, \boldsymbol{U}_n) \tag{4.18}$$

Il flusso convettivo attraverso la faccia ij al secondo ordine sarà quindi:

$$\boldsymbol{F}_{ij}^2 = \boldsymbol{F}_{ij}^1 + \frac{\boldsymbol{F}_n^{\star} - \boldsymbol{F}_{ij}^1}{d_{13}} \cdot d_{23} \tag{4.19}$$

dove d_{23} è la distanza tra boundary (punto 3) e centro dell'interfaccia (punto 2) e d_{13} è la distanza tra boundary e l'image point (punto 1).

4.2 ONERA shock-shock interaction

In questa sezione è stato riprodotto tramite **ImmerFlow** un esperimento condotto al centro di ricerca aerospaziale ONERA, nel wind-tunnel a bassa densità R5Ch [33]. L'obiettivo di questo esperimento è lo studio dell'interazione tra urti (*shock-shock interaction*), con particolare enfasi sulle interazioni di tipo Edney IV [13]. Questi fenomeni sono infatti cruciali in regime ipersonico, in quanto vanno a indurre picchi locali di pressione e temperatura e sono in grado di cambiare sostanzialmente le performance di un velivolo [28]; la validazione del codice in presenza di queste interazioni è quindi fondamentale. I dati prodotti da questo esperimento sono stati largamente utilizzati per la validazione di codici atti a riprodurre flussi di questo tipo.

4.2.1 Interazione di tipo Edney IV

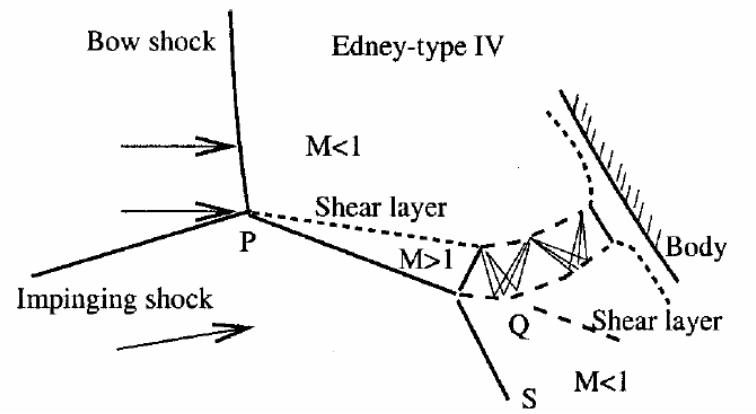


Figura 4.7: Interazione tra urti di tipo Edney IV. [9]

Quando un urto obliquo incidente (*impinging shock*) incontra un urto staccato (*bow shock*) si possono presentare vari tipi di interferenza tra i due urti. Questi diversi *pattern* sono stati studiati, catalogati e divisi da Edney [13] in 6 diverse famiglie, in base alla posizione relativa tra i due urti e alla loro intensità. Di questi vari tipi di interferenza, la più critica è sicuramente quella di tipo Edney IV (Figura 4.7) [9]. Nel caso in cui l'urto incidente incontri il *bow-shock* all'interno del *sonic circle*, ovvero quella zona dello spazio dove l'urto è forte e si ha $M < 1$ dopo di esso, si forma uno *shear layer* che separa una zona subsonica da una supersonica. Nel caso in cui l'angolo formato tra lo *shear layer* e la tangente alla superficie del corpo sia troppo elevato, si forma un getto supersonico che, dirigendosi verso la superficie del corpo, subisce un urto retto finale causando elevati valori di pressione e flusso di calore sulla superficie del corpo [9].

Il primo caso in cui i disastrosi effetti dei carichi termici indotti da questo tipo di interazione furono ben chiari fu durante i test del velivolo sperimentale X-15-2 a un numero di Mach pari a circa 6.7 [8]. L'urto obliquo generato dall'ala del velivolo interagì con l'urto staccato generato dal pilone di supporto del motore, causando danni al pilone stesso e l'incenerimento del *coating* protettivo [2].

4.2.2 Setup sperimentale

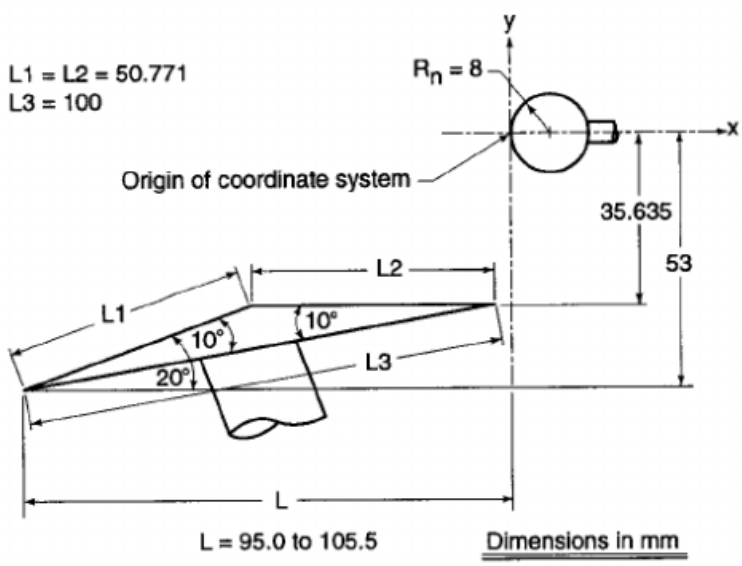


Figura 4.8: Setup sperimentale utilizzato dall'ONERA. [28]

Si può vedere in Figura 4.8 il setup sperimentale utilizzato dall'ONERA per condurre l'esperimento. Un cuneo, la cui sezione è un triangolo isoscele di base 100 mm e angoli alla base di 10° e con profondità (in direzione z, uscente dal foglio) pari a 100 mm, è utilizzato come generatore dell'urto obliquo. Un cilindro di raggio 0.008 mm e profondità 100 mm è invece utilizzato per generare il *bow shock*. Variando la posizione e l'orientamento del cuneo relativamente al cilindro è possibile ottenere diverse configurazioni d'urto; in questo caso il *leading edge* del cuneo è stato posizionato a 102 mm lungo x e 53 mm lungo y dall'origine degli assi e il cuneo stesso è stato inclinato in modo da formare un angolo di 20° con la corrente a monte. Il cilindro è stato equipaggiato con prese di pressione e termocoppie, in modo tale da misurare la distribuzione di pressione e il flusso di calore sulla superficie. Si sottolinea come l'elevata profondità di entrambi i solidi utilizzati permetta di considerare il flusso bidimensionale.

Le condizioni in cui è stato svolto l'esperimento possono essere trovate nella Tabella 4.1.

Grandezza	Valore
T^0	1050 K
p^0	250000 Pa
T_∞	52.5 K
p_∞	5.9 Pa
M_∞	9.95
Re_∞	$1.66 \cdot 10^{-5}$ m^{-1}

Tabella 4.1: Condizioni sperimentali per l'esperimento condotto all'ONERA. [33]

Il basso valore di temperatura totale impedisce di raggiungere localmente valori di temperatura in cui sia necessario considerare le reazioni chimiche che potrebbero avvenire all'interno della miscela aria, composta al 21% da O_2 e al 79% da N_2.

4.2.3 Setup su ImmerFlow

Per la generazione della griglia è stato utilizzato il grigliatore **PABLO** introdotto nella sezione precedente. Come si può vedere in Figura 4.9 il dominio di calcolo è un quadrato di lato 0.25 m la cui origine è $x_Q = (-0.14, 0.13)$, rispetto all'origine degli assi rappresentata in Figura 4.8; la dimensione massima di tutti gli ottanti presenti in questo dominio è fissata a 50 mm. Sono state create varie geometrie (in formato DGF) per rappresentare i corpi immersi (in rosso il cuneo e il cilindro) e delle regioni in cui verrà imposto un certo livello di raffinamento degli griglia *quadtree* (in nero). Le due regioni dall'estensione maggiore sono il semicerchio di raggio 20 mm e la zona che connette obliquamente il *leading edge* del cuneo con il semicerchio (visibili chiaramente nelle Figure 4.10a e 4.12c); in queste due zone la taglia massima degli ottanti è pari a 0.2 mm. In questo modo è possibile catturare in maniera abbastanza precisa l'intero urto obliquo prima del contatto con il *bow shock* del cilindro. Data la configurazione di questo esperimento, infatti, un leggero spostamento dell'urto obliquo incidente produrrebbe un grande cambiamento nel *flow field* in prossimità del cilindro; è necessario quindi calcolare con un'adeguata accuratezza la posizione di questo urto. Per lo stesso identico motivo è stata creata una regione con taglia massima degli ottanti ancora più piccola, 0.1 mm, per rappresentare il *boundary layer* del cuneo, visibile in Figura 4.11. In campo ipersonico sia lo strato limite termico sia quello di quantità di moto hanno spessori considerevoli e la loro presenza agisce sull'urto obliquo che nasce al *leading edge* del cuneo; se non fossero ben rappresentati l'urto potrebbe assumere una posizione più bassa di quella reale.

L'ultima regione di raffinamento ha forma trapezoidale e ha un lato coincidente con la superficie del cilindro (Figura 4.10); qui la taglia massima degli ottanti è pari

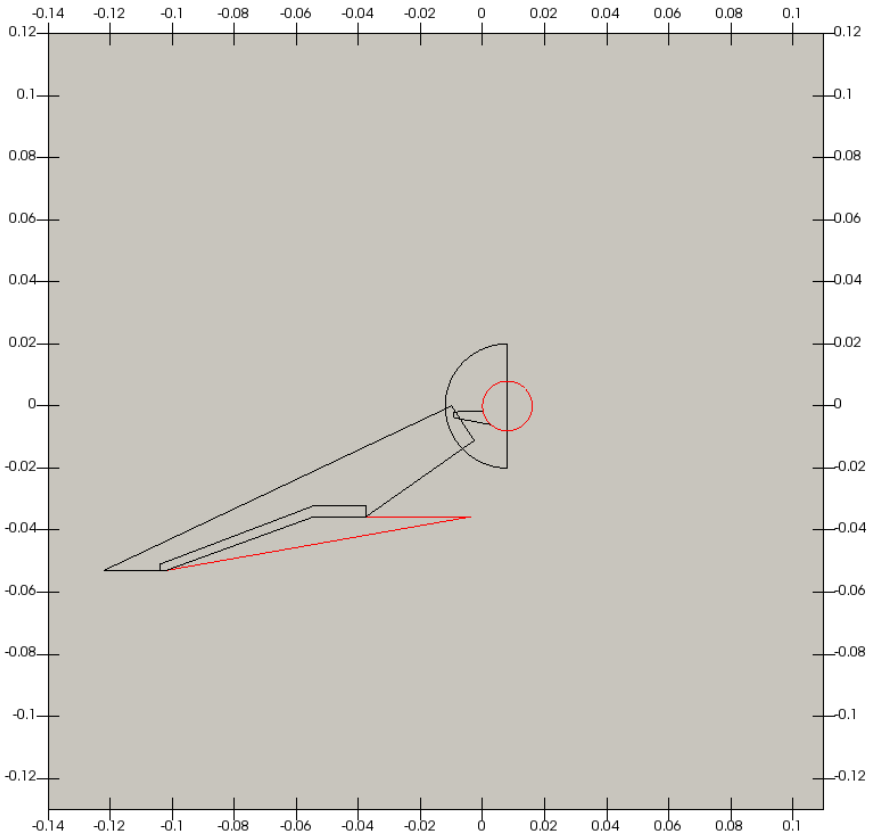
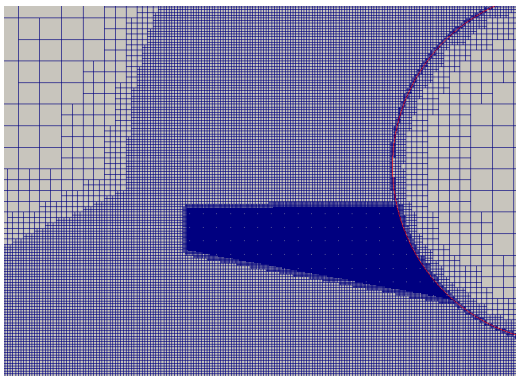


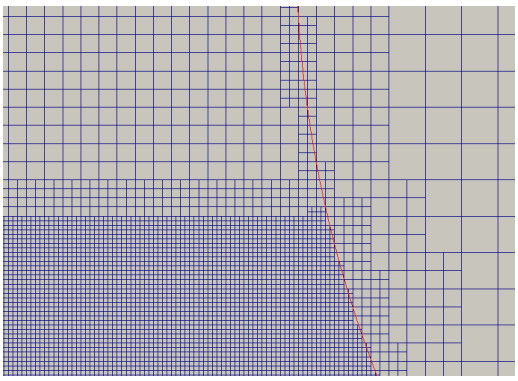
Figura 4.9: Il dominio di calcolo è colorato in grigio. Le linee rappresentanti le superfici degli *immersed bodies* sono rappresentate in rosso. In nero sono rappresentate le linee che definiscono le *regions* utilizzate per il raffinamento della griglia.

a 0.05 mm. Questa regione è stata creata per avere la massima precisione nella zona in cui i due urti interagiscono e dove nasce il getto supersonico. Si sottolinea come all'interno dei corpi si sia imposta la taglia degli ottanti più grande possibile; infatti, le celle all'interno del corpo non influiscono sul carico computazionale, essendo la loro soluzione banale, e averne un grande numero porterebbe solo uno sbilanciamento del carico tra i vari processi MPI. Il processo di bilanciamento del carico fatto da **PABLO**, infatti, non permette di dare un peso alle varie celle in base al loro costo computazionale.

La griglia finale, col desiderato livello di precisione imposto, non è stata utilizzata fin da subito nella simulazioni, ma si è preferito procedere per step nel raffinamento della griglia. Nota la dimensione finale della griglia desiderata, sono state ricavate

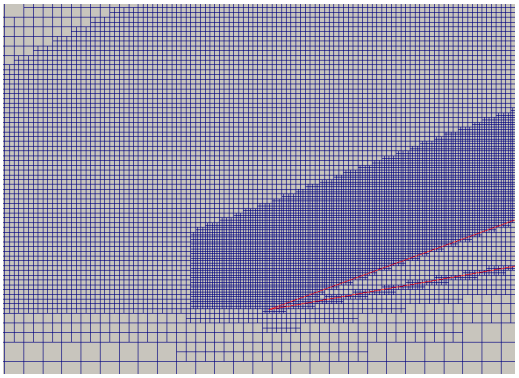


(a) le regioni più esterne contengono l'urto obliquo e l'urto staccato mentre la regione più raffinata interna che tocca il cilindro contiene la zona di interazione tra gli urti

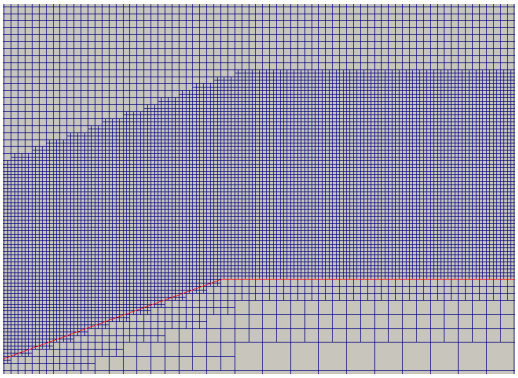


(b) limite superiore della zona di raffinamento contenete l'interazione tra urti a contatto col cilindro

Figura 4.10: Particolari della griglia finale in prossimità del cilindro.

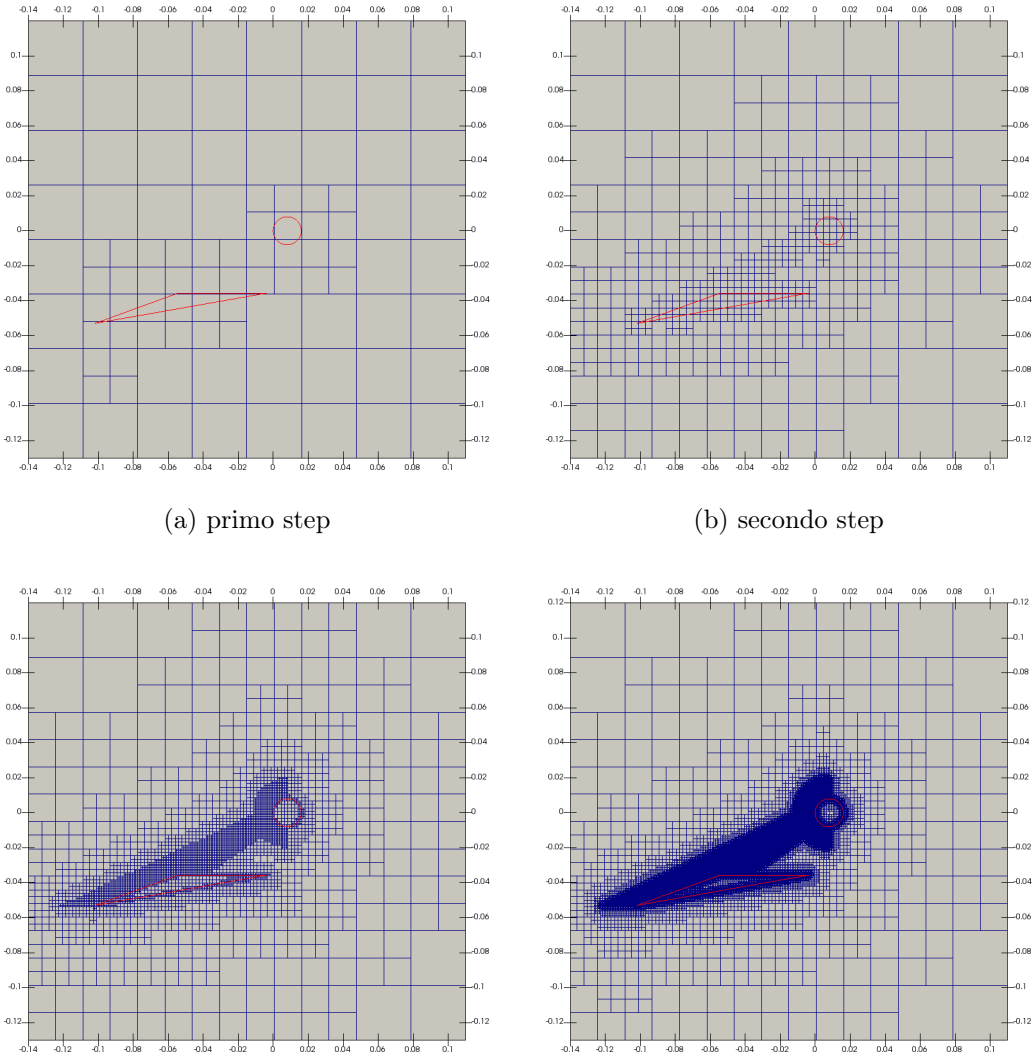


(a) *leading edge* del cuneo con una regione di raffinamento per cattura il *boundary layer*



(b) vertice del cuneo con regione di raffinamento per il *boundary layer*

Figura 4.11: Particolari della griglia finale in prossimità del cuneo.



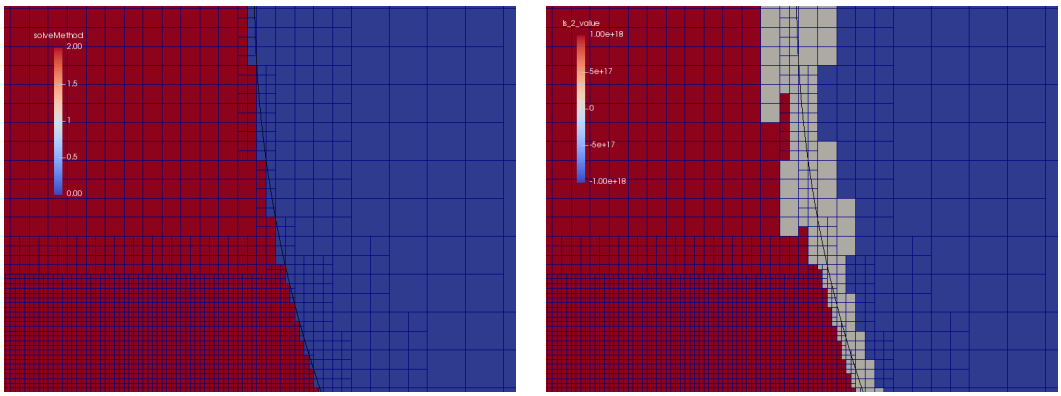
(a) primo step

(b) secondo step

(c) terzo step

(d) quarto step

Figura 4.12: 4 step di raffinamento utilizzati per raggiungere la griglia finale imposta.



(a) suddivisione tra celle fluide, in rosso, e celle solide, in blu.

(b) oltre alle celle fluide e solide sono evidenziate in grigio le celle utilizzate nello stencil dell'*immersed boundary method*

Figura 4.13: Particolari della griglia finale in prossimità del cilindro dal punto di vista IB.

4 griglie intermedie (Figura 4.12). Per ognuna di queste griglie è stata fatta una simulazione utilizzando le condizioni al contorno scelte; una volta soddisfatto un certo *trigger* (basato sul numero di iterazioni per step di raffinamento oppure sul valore dei residui), la griglia è stata raffinata e una nuova simulazione è stata avviata utilizzando come condizioni iniziali i risultati provenienti dallo step precedente.

In Figura 4.13 è possibile visualizzare la differenza tra le celle che sono considerate parte del fluido, in rosso, e quelle che fanno parte del solido, in blu. Queste ultime hanno una soluzione banale e non saranno visualizzate nelle successive immagini riguardanti i risultati della simulazione.

Ai bordi del dominio sono state imposte delle condizioni di *freestream* con i valori presenti in Tabella 4.2. In corrispondenza della superficie del cilindro e del cuneo, invece, è stata utilizzata una condizione *no-slip-isothermal-wall*, imponendo un valore di temperatura pari a $T_w = 300\ K$. Questi valori sono stati utilizzati anche come valori di riferimento per l'adimensionalizzazione dei risultati.

Grandezza	Valore
M_∞	9.95
V_∞	1450 m/s
T_∞	52.5 K
p_∞	5.9 Pa
ρ_∞	$3.910 \cdot 10^{-4}$ kg/m^3
μ_∞	$3.405 \cdot 10^{-6}$ $Pa \cdot s$
Re_∞	$1.66 \cdot 10^{-5}$ m^{-1}
T_w	300.00 K

Tabella 4.2: *Boundary conditions* utilizzate su **ImmerFlow**.

4.2.3.1 Modifiche effettuate

Per ottenere i risultati presentati nella prossima sezione è stato necessario effettuare alcune correzioni e modifiche a **ImmerFlow**. La prima modifica effettuata è stata l'implementazione della legge di Sutherland per il calcolo della viscosità e, attraverso il numero di Prandtl, della conducibilità termica. Le elevate temperature raggiunte, infatti, insieme alla stretta interazione tra strato limite e urto, tipica dei flussi ipersonici, rendono fondamentale l'avere un valore della viscosità che sia funzione della temperatura. Inizialmente il valore della viscosità era considerato costante in tutto il dominio; in questo modo però l'urto obliquo incidente, generato dal cuneo, era posizionato in maniera completamente diversa, e errata ovviamente, rispetto a quanto aspettato. Questo a causa dello strato limite presente sulla faccia obliqua del cuneo che, non essendo sufficientemente spesso, non alzava a sufficienza l'urto obliquo.

La seconda modifica fatta riguarda la gestione dell'*immersed boundary*. Il *tagging* utilizzato inizialmente era analogo a quanto utilizzato da Gorsse et al. [17] ma, in questo modo, nel caso in cui il boundary cadesse come mostrato nella parte superiore della Figura 4.14, era necessario estrapolare il valore della temperatura sull'interfaccia tra la cella fluida (identificata in Figura da una F) e la cella solida (S). Dati i valori di input di questa estrapolazione (l'elevato valore di temperatura nello strato limite e la temperatura della parete fissata a 300 K), il valore di temperatura ricavato per l'interfaccia tra solido e fluido risultava essere minore dello zero assoluto, valore ovviamente errato e impossibile da imporre. Cambiando il *tagging* in modo tale da considerare solida la cella in cui cade il boundary (esattamente come spiegato nelle sezioni precedenti, ovvero tenendo conto dei 5 punti di collocazione della cella bidimensionale) è possibile risolvere questo problema. Infatti, il valore di temperatura da imporre sull'interfaccia tra fluido e solido risulta essere sempre ottenuto

tramite un processo di interpolazione, rendendo impossibile ottenere temperature negative (come nella parte inferiore della Figura 4.14).

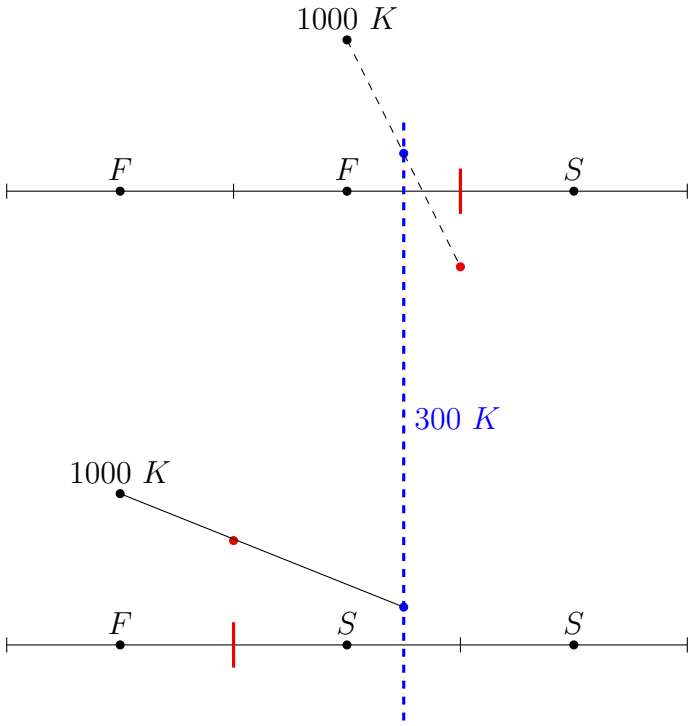


Figura 4.14: Confronto tra i due metodi di *tagging* utilizzati. In alto si può visualizzare il vecchio metodo, che impone l'utilizzo di una estrapolazione (linea tratteggiata) utilizzando come input i dati di centro cella fluida (in nero) e quelli imposti sul boundary (in blu). Il valore ricavato sull'interfaccia può essere negativo (in rosso). In basso si può vedere il nuovo *tagging* utilizzato, questo utilizza solamente una interpolazione (linea continua) e quindi non può dare origine a temperature negative sull'interfaccia tra la cella fluida e quella solida

4.2.4 Risultati

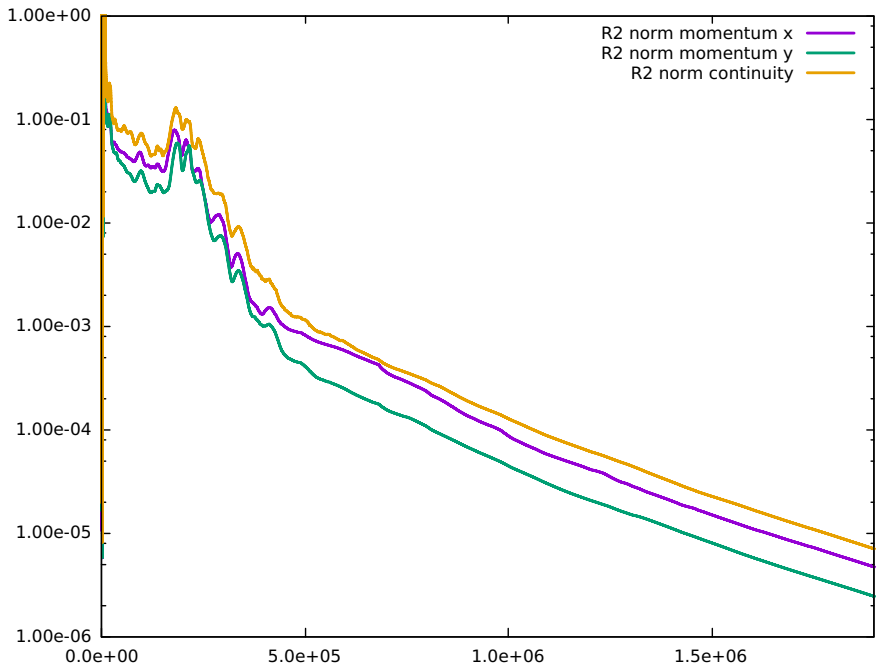


Figura 4.15: Andamento dei residui in norma 2 durante la simulazione.

La simulazione è stata interrotta dopo $1.9 \cdot 10^6$ iterazioni (raggiunte in circa 47 ore su 16 processi in parallelo); la griglia finale utilizzata per la maggior parte delle iterazioni ha circa 235000 celle. Come si può vedere in Figura 4.15, i residui in norma 2 sono diminuiti di 5 ordini di grandezza, quindi si può considerare la simulazione arrivata a convergenza.

Una panoramica dei risultati ottenuti è visibile in Figura 4.16, dove la distribuzione di velocità adimensionalizzata ($V_a = V/\sqrt{RT}$) è rappresentata su tutto il dominio. Si vede chiaramente come l'urto obliquo generato dal cuneo vada a interagire con il *bow shock* generato dal cilindro nella zona in cui questo è forte.

Nella Figura 4.17 si può vedere un dettaglio della regione di interazione tra i due urti. Guardando la velocità si può notare come l'urto incidente venga riflesso in maniera netta (separazione tra regione rossa e arancione) e come sia presente uno *shear layer* in cui la velocità passa da un valore molto basso nella zona superiore (in blu), regione subsonica posta dopo l'urto forte staccato, a una valore più elevato (in arancione), regione supersonica dopo l'urto riflesso debole. Il fatto che queste due zone siano separate da una *slip line* è chiaro anche dalla distribuzione di pressione, che si mantiene costante a cavallo delle due zone. La distribuzione di pressione permette anche di visualizzare la riflessione di Mach presente all'ingresso del getto supersonico e l'urto finale alla fine di esso.

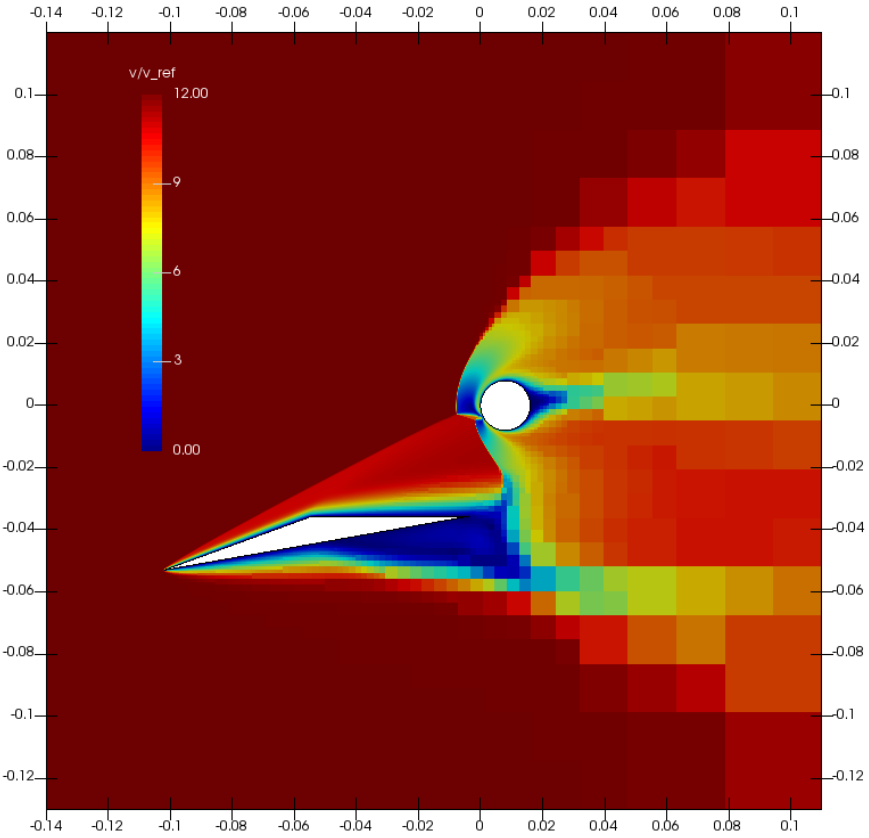
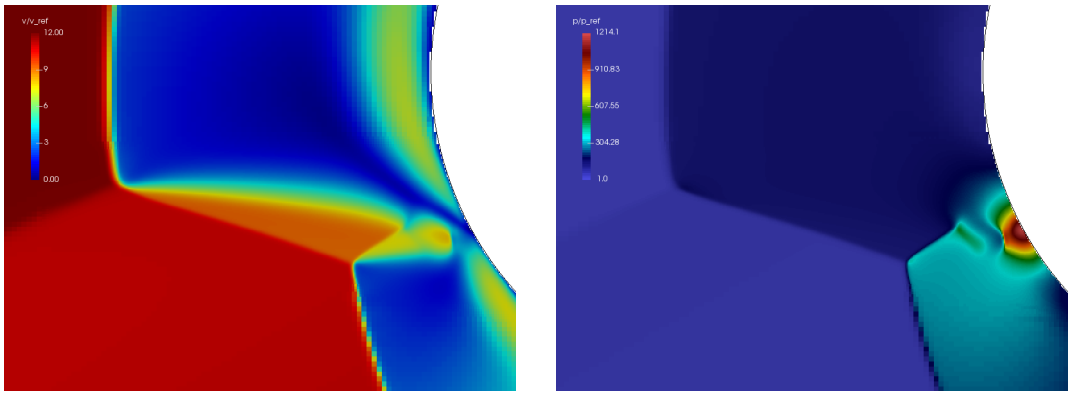


Figura 4.16: Andamento della velocità adimensionalizzata all'interno del dominio.



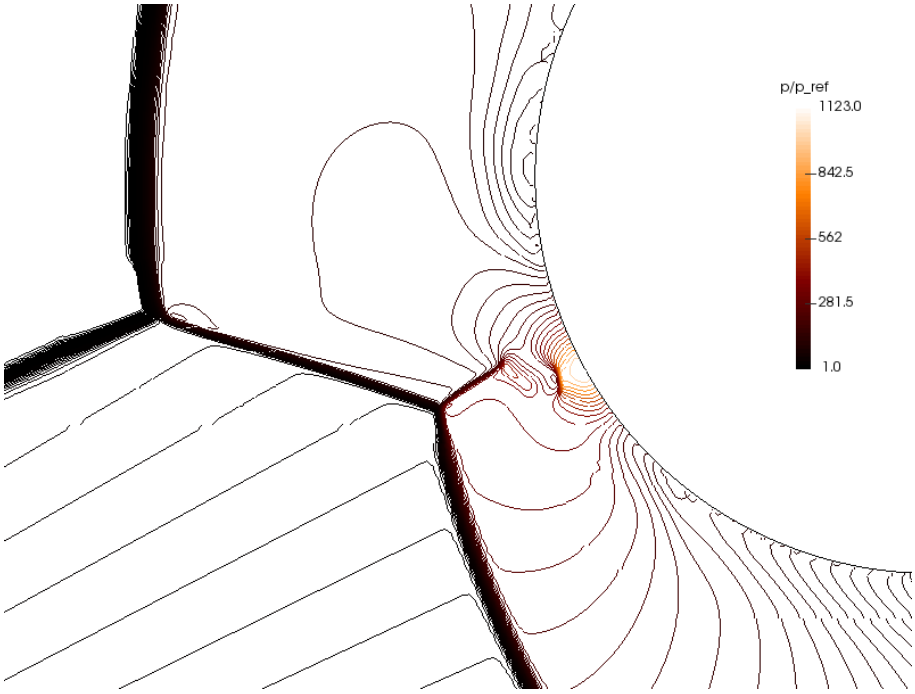
(a) velocità adimensionalizzata (b) pressione adimensionalizzata

Figura 4.17: Dettaglio della regione di interazione tra i due urti.

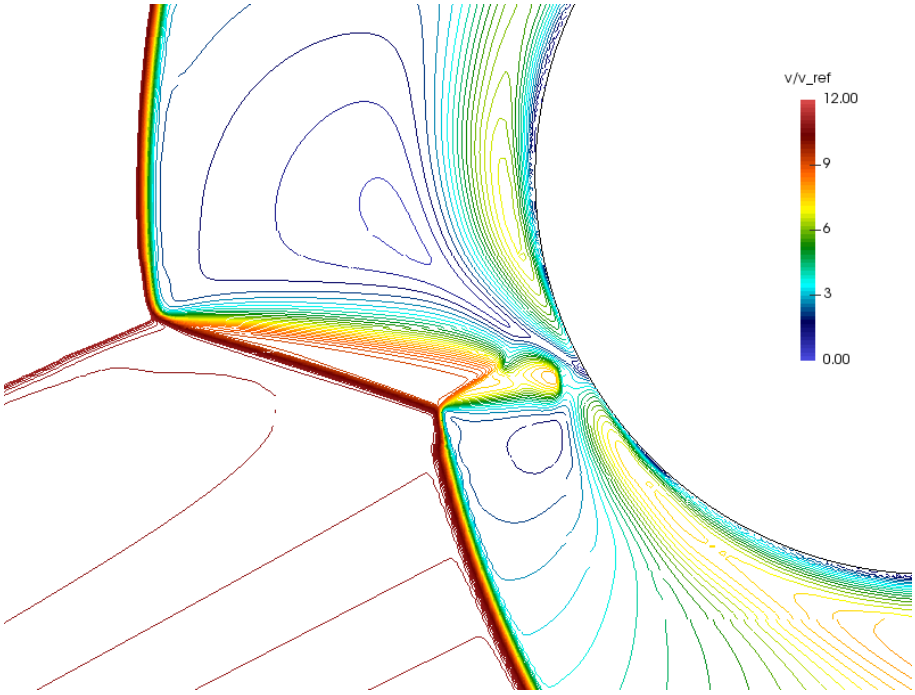
La struttura di questo getto supersonico può essere facilmente visualizzata attraverso l'utilizzo delle isolinee di Figura 4.18. All'ingresso del getto l'urto viene riflesso dalla *slip line* in modo tale da formare un urto di tipo λ. Questo tipo di riflessione porta ad avere due urti obliqui nella parte inferiore uniti ad un urto retto nella parte superiore attraverso un punto triplo. A partire dal punto di contatto tra l'urto retto e la *slip line* si forma il primo fascio di espansione che porta il flusso ad accelerare all'interno del getto. Poco prima della parete del cilindro si forma poi un ultimo urto retto che rallenta il flusso per fare in modo che esso si fermi in corrispondenza del punto di arresto.

In Figura 4.19a può essere visualizzata una schematizzazione della struttura del getto. Questa struttura può essere confermata, oltre che dalle isolinee viste precedentemente, anche dallo studio dell'andamento della pressione su due *streamlines* posizionate all'interno del getto (Figura 4.19b).

Nel Grafico 4.1 si possono vedere i due andamenti della pressione lungo le *streamlines* considerate. La *streamline* superiore passa attraverso l'urto retto posizionato sopra il punto triplo e, infatti, subisce un'unica compressione iniziale; successivamente viene espansa all'interno del getto fino a raggiungere la pressione minima di circa 1800 Pa, per poi subire una forte compressione dovuta all'urto retto presente alla fine del getto. Dopo l'urto questa subisce ancora una compressione isoentropica per poi essere accelerata nuovamente spostandosi sul dorso del cilindro. La *streamline* inferiore (in nero), invece, passa inizialmente attraverso il primo urto obliquo della riflessione di Mach e raggiunge un valore di pressione pari a circa 2000 Pa, solo successivamente (e in posizione anteriore rispetto al punto triplo) passa attraverso il secondo urto obliquo. A questo punto i fasci di espansione presenti ne abbassano la pressione a circa 1800 Pa (lo stesso valore raggiunto dalla *streamline* superiore) per poi passare attraverso l'urto retto finale e successivamente accelerare sul ventre del cilindro.

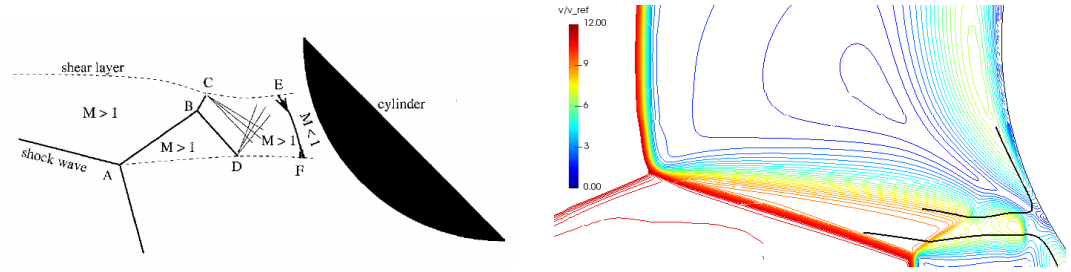


(a) Isolinee del $\log_{10}(p/p_{ref})$ colorate in base al valore di pressione adimensionalizzata



(b) Isolinee del numero di Mach colorate in base al valore di velocità adimensionalizzata

Figura 4.18: Distribuzione di isolinee in corrispondenza della regione di interazione tra i due urti.



(a) struttura del getto supersonico [9]

(b) andamento delle due *streamlines* analizzate nel getto supersonico

Figura 4.19: Particolare del getto supersonico.

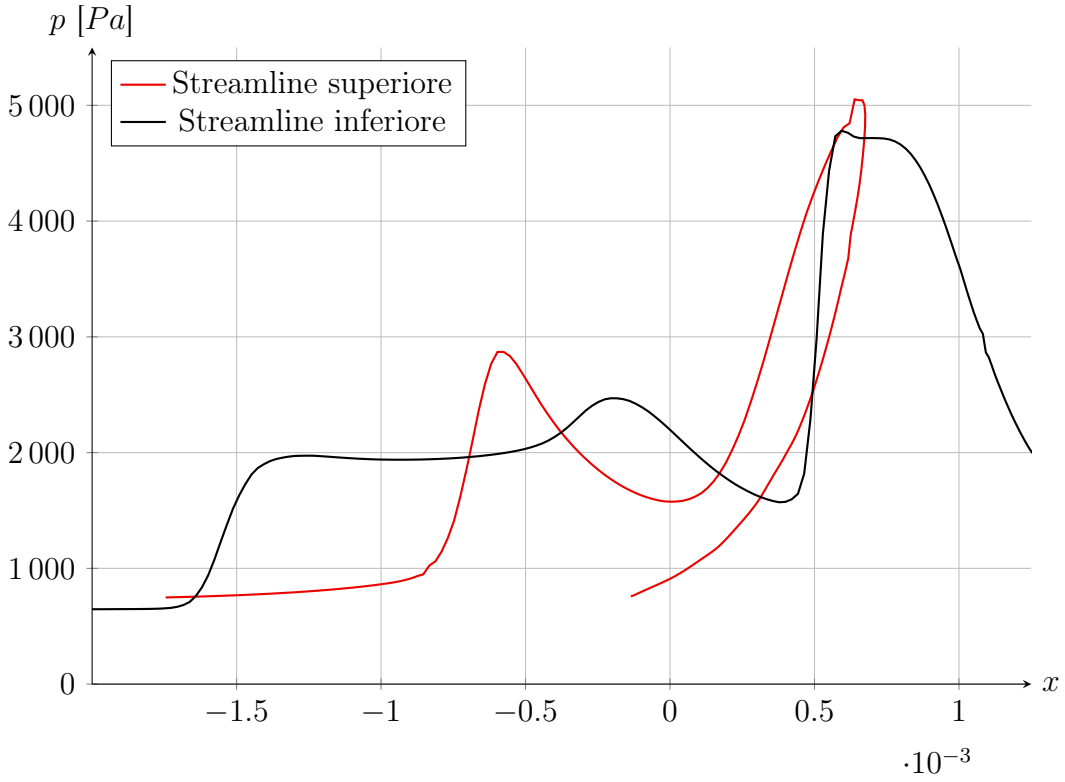


Grafico 4.1: Andamento della pressione lungo le due *streamlines* posizionate all'interno del getto supersonico.

Si vogliono ora confrontare i risultati ottenuti con quelli sperimentali [33]. In regime ipersonico, in particolare nel caso in cui si abbia interazione tra urti di tipo Edney IV, come in questo caso, i dati a cui si è più interessati dal punto di vista progettuale sono i valori di pressione e di flusso termico sulla superficie del corpo di interesse. L'esperimento condotto nel *wind tunnel* R5Ch dell'ONERA è stato fatto utilizzando un cilindro dotato di prese di pressione e termocoppie distribuite lungo tutta la sua superficie, è possibile quindi comparare i risultati sperimentali con quanto calcolato con **ImmerFlow**, come fatto nei Grafici 4.2 e 4.3.

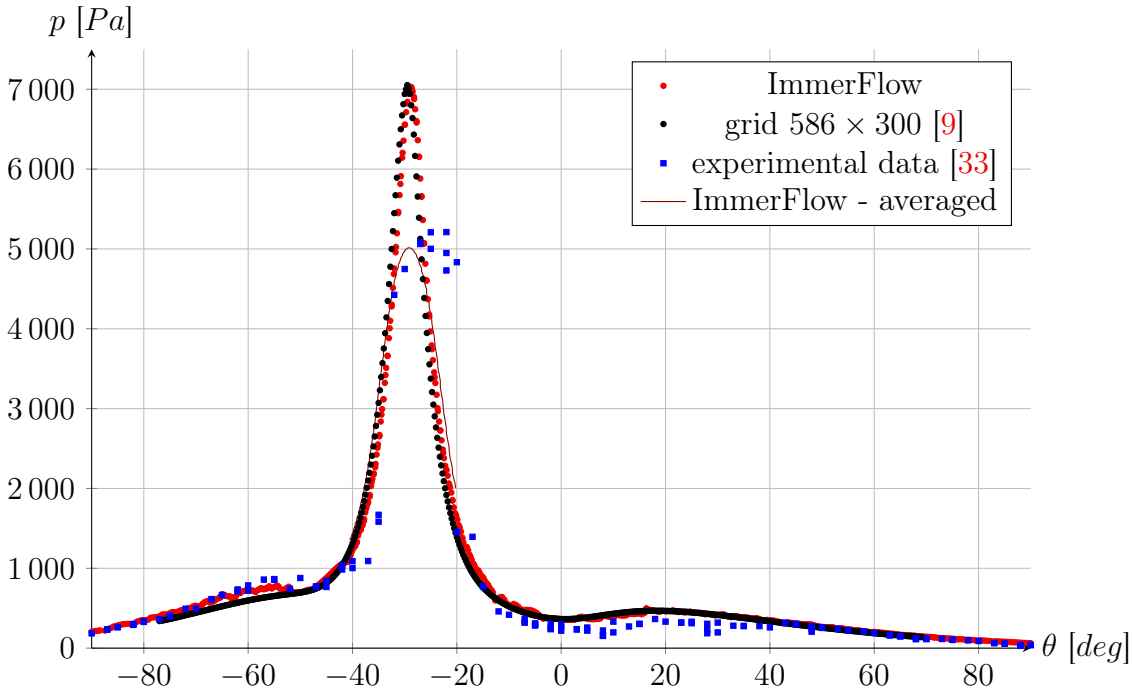


Grafico 4.2: Distribuzione di pressione sulla parete del cilindro.

Nel Grafico 4.2 è possibile vedere la distribuzione di pressione sulla parete del cilindro, in funzione della posizione angolare. Sono stati confrontati i risultati provenienti da **ImmerFlow**, quelli sperimentali [33] e quelli ottenuti da D'Ambrosio [9] utilizzando un codice ai volumi finiti su una griglia *multiblock*. I risultati numerici combaciano sufficientemente bene con quelli sperimentali lontano dal picco di pressione, mentre la posizione di quest'ultimo risulta essere spostata di $-3.72°$ e il modulo è più grande del 34.93%. Come evidenziato da D'Ambrosio, questa differenza potrebbe essere causata dalla scarsa risoluzione spaziale, se comparata ai risultati numerici, dell'apparato sperimentale, dotato di prese di pressione di dimensione finita. In maniera analoga a quanto fatto in [9], i risultati numerici sono stati mediati spazialmente in modo da replicare l'effetto di una presa di pressione di dia-

metro 1.5 mm, i risultati sono visibili nel Grafico 4.2 con una linea rossa continua; i dati così trattati si avvicinano molto a quelli sperimentali per quanto riguarda il modulo. Si sottolinea come i risultati ottenuti con **ImmerFlow** coincidano quasi perfettamente con quelli ottenuti da D'Ambrosio.

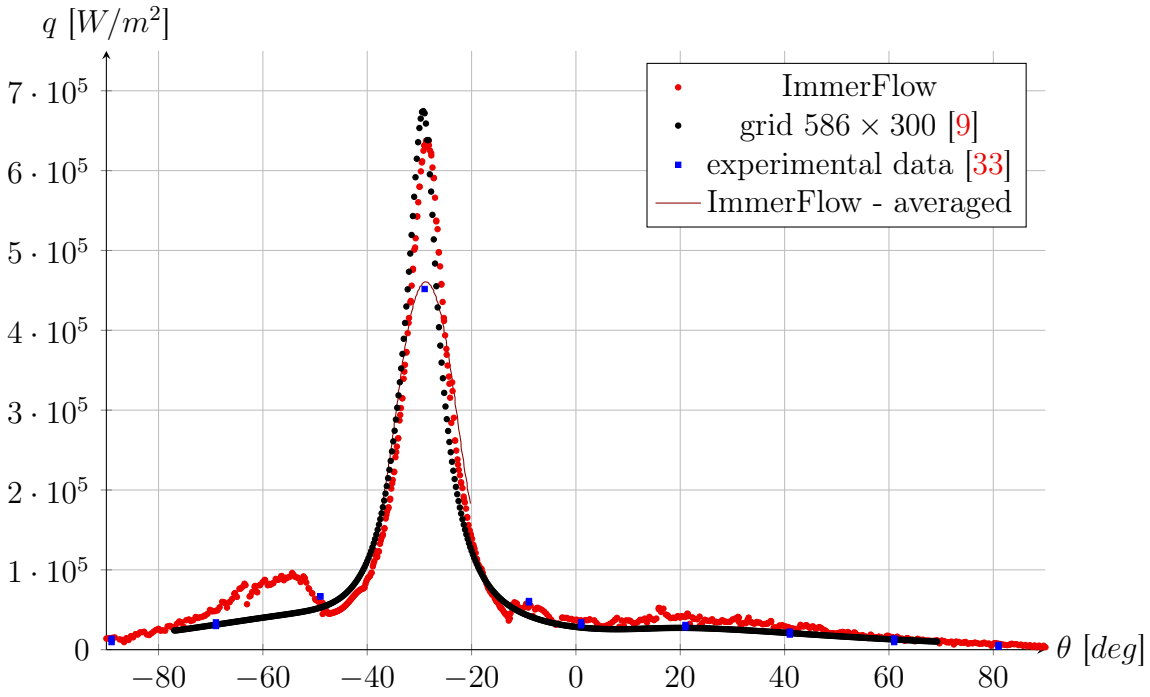


Grafico 4.3: Distribuzione del flusso di calore a parete.

Nel Grafico 4.3 si può visualizzare la distribuzione del picco di flusso di calore a parete, in funzione della posizione angolare. Anche in questo caso la differenza coi dati sperimentali è notevole per quando riguarda il modulo del flusso di calore, si ha infatti una differenza del 41.10%, mentre la posizione angolare è ben rappresentata con un differenza di solo 0.28°; lontano dal picco la corrispondenza è buona, come anche la corrispondenza generale con D'Ambrosio. I dati numerici mediati, invece, rappresentano quasi perfettamente quelli sperimentali. L'unica differenza degna di nota è un picco locale nel flusso di calore a parete per valori di θ prossimi ai $-60°$. Andando a visualizzare l'andamento della temperatura adimensionalizzata in questa zona (Figura 4.21a) si nota come la temperatura inizi a decrescere ma poi, esattamente in corrispondenza del cambio di taglia degli ottanti della griglia utilizzata, si riporti a valori più elevati.

Dato che questo fenomeno potrebbe essere causato dal cambio di taglia della griglia in una zona dagli elevati gradienti, si è deciso di ripetere nuovamente la simulazioni utilizzando una griglia che mantenga la taglia minima degli ottanti (0.05 mm)

non solo nella zona di interazione tra gli urti, ma su tutta la superficie del cilindro (Figura 4.20).

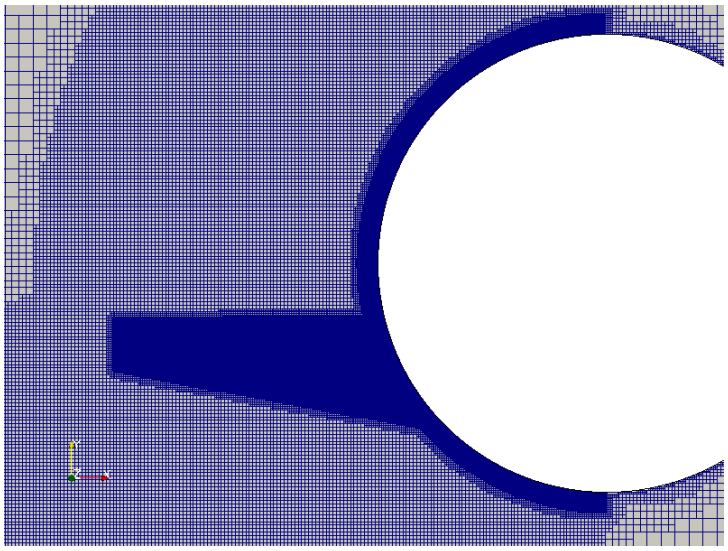


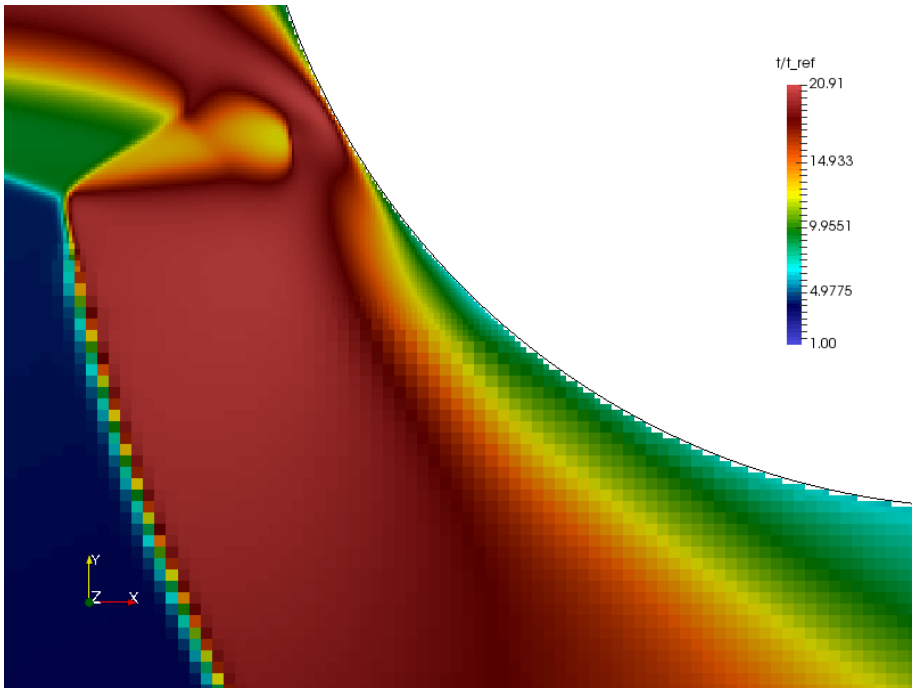
Figura 4.20: Griglia raffinata in prossimità della parete del cilindro.

I risultati ottenuti con la nuova griglia possono essere visualizzati in Figura 4.21b, dove sono comparati con quelli precedenti. Si può subito vedere come il picco locale di temperatura sia scomparso, indicando come quello ottenuto precedentemente fosse solamente dovuto al cambio di taglia della *mesh*.

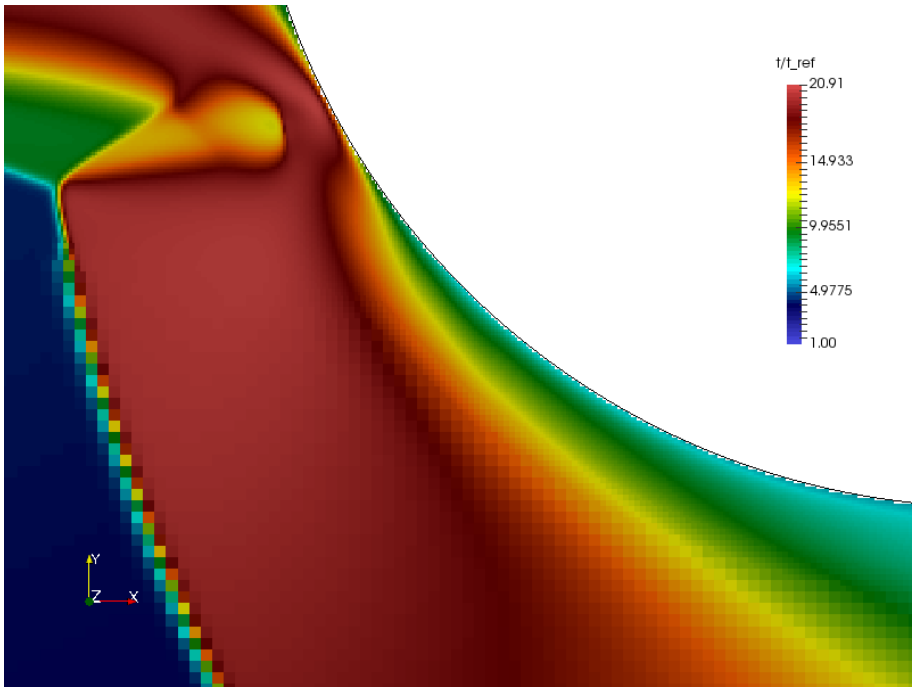
A questo punto vengono nuovamente riportate le distribuzioni di pressione e flusso di calore a parete utilizzando i nuovi risultati ottenuti (Grafico 4.4 e 4.5). La già buona corrispondenza lontano dal picco centrale è migliorata notevolmente e il picco locale nel flusso di calore intorno a $\theta = -60°$ è completamente scomparso. Il picco dovuto all'interazione di tipo Edney IV, invece, si è mosso di 1° rispetto alla soluzione precedente, mentre il valore assoluto del massimo è rimasto praticamente lo stesso.

Come già sottolineato da D'Ambrosio [9], questi risultati dimostrano come un buon livello di precisione della griglia, soprattutto in direzione tangenziale, sia necessario a catturare in maniera precisa la posizione del picco di pressione e flusso di calore a parete. Questo codice sarebbe potuto essere utilizzato su mesh molto più precise e raffinate in questa zona, per cercare di ottenere risultati ancora più attendibili, ma la precisa valutazione di questo fenomeno esula dall'obiettivo di questa tesi, che punta solamente a dimostrare l'usabilità di **ImmerFlow** in queste situazioni.

I risultati fin qui presentati sono quindi più che sufficienti a validare **ImmerFlow** per l'utilizzo in regime ipersonico; i suoi risultati coincidono infatti con quelli ottenuti da altri codici attualmente utilizzati e largamente validati e coincidono parzialmente anche con quelli ottenuti sperimentalmente.



(a) griglia iniziale



(b) griglia raffinata

Figura 4.21: Distribuzione della temperatura adimensionalizzata sulla parete con due diverse griglie.

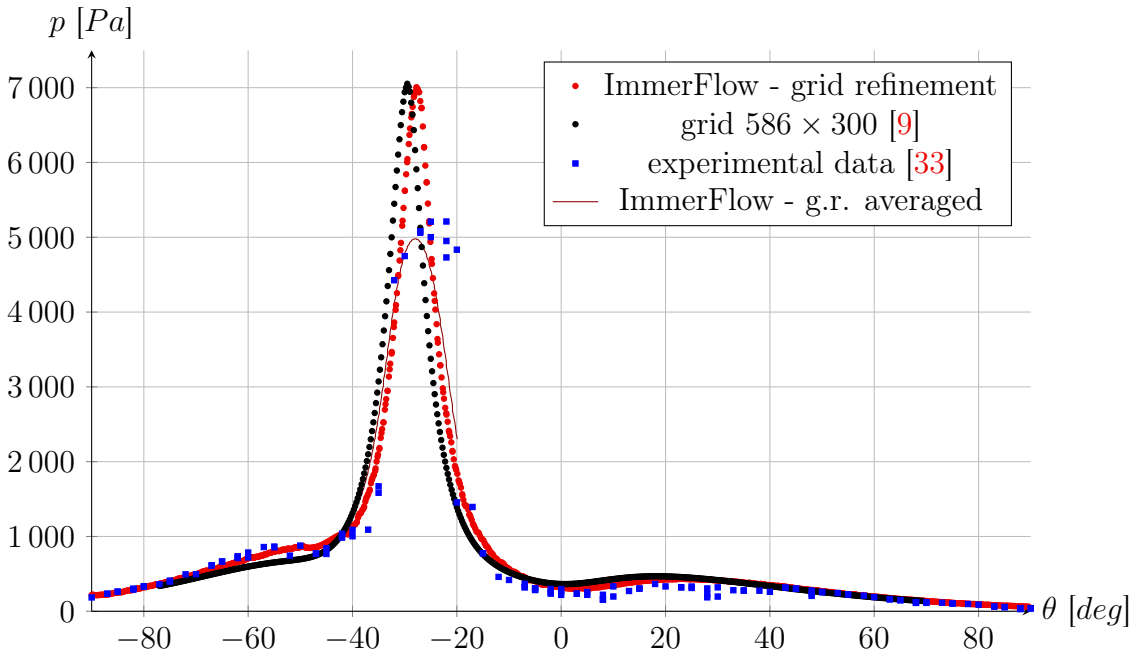


Grafico 4.4: Distribuzione di pressione a parete con la griglia raffinata.

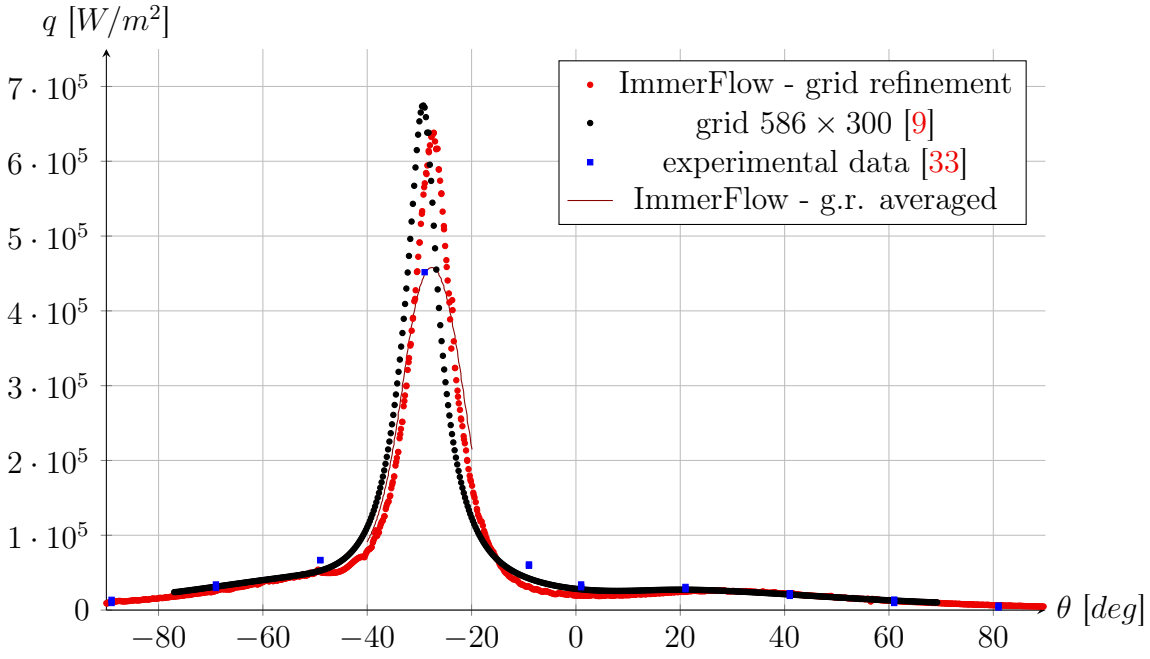


Grafico 4.5: Distribuzione del flusso di calore a parete con la griglia raffinata.

5 Implementazione dei fenomeni aerotermochimici

In questo Capitolo vengono inizialmente introdotte le principali caratteristiche che differenziano un flusso ipersonico da uno supersonico. Successivamente ci si concentra sugli effetti dovuti alle alte temperature e viene data una prima definizione di disequilibrio chimico e vibrazionale. Infine si espone uno studio preliminare atto a implementare le considerazioni appena fatte all'interno di **ImmerFlow**; per fare questo viene utilizzato in parte **Cantera**, una libreria *open-source* di strumenti adatta alla risoluzione di problemi di equilibrio chimico e di termodinamica [1].

5.1 Le caratteristiche di un flusso ipersonico

Viene di seguito riportata la caratterizzazione di un flusso ipersonico presentata da Anderson nel suo libro [7]. Generalmente con flusso ipersonico si intende un flusso con numero di Mach superiore a 5. Ovviamente, come nella maggior parte dei fenomeno fisici, questo non è un limite fisso e inamovibile, ma rappresenta semplicemente una soglia intorno alla quale iniziano a presentarsi dei fenomeni tipici di questo regime; essi vengono di seguito elencati:

- **thin shock layers**: la regione di fluido compresa tra un corpo e un urto da esso generato è chiamata *shock layer*. Dato l'elevato numero di Mach, nel caso di flusso ipersonico è possibile raggiungere elevati valori di densità dietro un urto, quindi si hanno spesso *shock layers* molto sottili. Dato l'estrema vicinanza dell'urto con la parete, si potrebbe avere interazione tra l'urto stesso e il *boundary layer*.

- **entropy layer**: si immagini di analizzare un urto generato da un cono il cui bordo di attacco sia smussato. In questo caso si avrebbe un urto staccato, forte in corrispondenza del'asse di simmetria e con intensità decrescente allontanandosi da tale asse. Dopo un urto forte retto l'aumento di entropia è considerevole, in questo caso, quindi, si avrebbe una distribuzione di entropia nello *shock layer* caratterizzata da forti gradienti; infatti, essa dipende dall'intensità dell'urto, che a causa della forma del corpo è fortemente variabile in base alla distanza dall'asse. Si ha quindi un *entropy layer*, caratterizzato da

forti gradienti, che fluisce lungo tutta la superficie del corpo. La presenza di gradienti di entropia è legata alla vorticità dal teorema di Crocco, e questo causa una forte interazione tra l'*entropy layer* e il *boundary layer*.

- **viscous interaction**: all'interno del *boundary layer* l'energia cinetica del flusso viene in parte dissipata in calore quando gli effetti viscosi rallentano la corrente per soddisfare la condizione di aderenza. In Figura 5.1 è possibile vedere l'andamento della temperatura all'interno di uno strato limite. Nel caso di flusso ipersonico questo aumento di temperatura è talmente elevato da avere importanti effetti su tutto il fluido. L'aumento di temperatura è responsabile di un aumento di viscosità e, essendo la pressione costante in direzione normale alla parete (all'interno del BL), si ha anche una diminuzione di densità. Tutti questi fattori causano un aumento dello spessore dello strato limite, che va a cambiare pesantemente la forma del corpo vista dal flusso inviscido esterno, che, in ultima analisi, va esso stesso a cambiare la forma dello strato limite e la distribuzione di pressione sul corpo. Questo effetto viene chiamato *viscous interaction* e ha importanti effetti sulle forze agenti sul velivolo stesso.

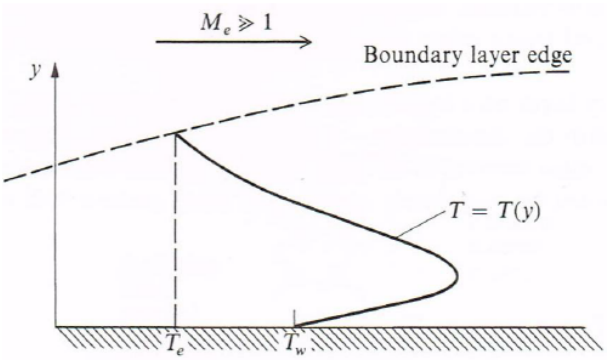


Figura 5.1: Strato limite termico a parete. [7]

- **high temperature flows**: l'enorme incremento di temperatura presente nello strato limite e dopo un urto forte è molto spesso sufficiente a eccitare i gradi di libertà vibrazionali all'interno delle molecole e a causarne la dissociazione; in caso di temperature ancora più elevate sarebbe necessario tenere conto anche della ionizzazione del gas considerato. Si deve quindi tenere conto della presenza di un *chemically reacting boundary layer*. Maggiori dettagli su queste affermazioni sono dati più avanti nel Capitolo. Si sottolinea anche come la presenza di scudi ablativi e in generale la presenza di una parete anche solo parzialmente catalitica renderebbe necessario tenere conto anche di tutte le reazioni chimiche presenti in questi fenomeni. Tutti questi effetti hanno una pesante influenza sulle forze agenti sul velivolo e sul suo comportamento, ma

in particolar modo causano un elevato flusso di calore verso il corpo, detto comunemente *aerodynamic heating*.

- **low density flow**: nel caso in cui il numero di Knudsen $K_n = \lambda/L$, dove λ è il libero cammino medio e L è una dimensione caratteristica del corpo considerato immerso nel fluido, sia superiore a circa 0.2, le equazioni di Navier-Stokes non possono più essere utilizzate (Figura 5.2) in quanto cade l'ipotesi del continuo. Questo è usualmente causato dalla bassa densità assunta dal fluido in condizioni estreme, quali ad esempio elevate altitudini; in queste condizioni il libero cammino medio di una molecola assume valori elevati e paragonabili alle dimensioni del corpo considerato. Spesso alcuni velivoli che si trovano in regime ipersonico sono anche in questo particolare regime detto *low-density flow*, perciò ne vengono qui elencate le caratteristiche tipiche, pur essendo esso non strettamente legato al numero di Mach. Al diminuire della densità del fluido, in particolare a partire da $K_n > 0.03$, iniziano a presentarsi dei fenomeni di *slip*; ovvero, in corrispondenza della parete di un corpo, il fluido non rispetta più la condizione di aderenza, esso ha una velocità finita diversa da quella di parete (*velocity-slip*) e, allo stesso modo, ha una diversa temperatura (*temperature-slip*). Per valori di K_n ancora superiori, le equazioni di Navier-Stokes devono essere abbandonate. Quindi, nel caso in cui il velivolo considerato sia, ad esempio, ad altitudini tali da invalidare l'ipotesi del continuo, tali effetti devono essere considerati, pur essendo essi non strettamente legati al numero di Mach e al regime ipersonico.

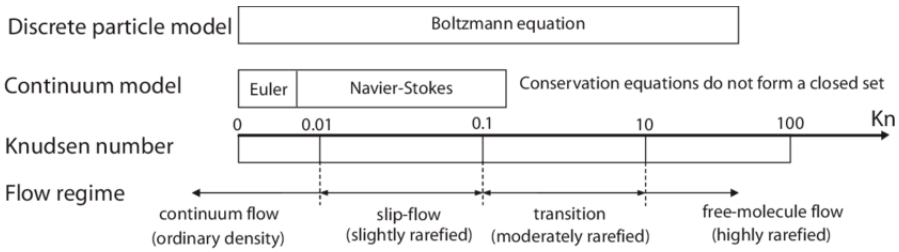


Figura 5.2: Set di equazioni utilizzabili in base al numero di Knudsen. [39]

In Figura 5.3 è possibile visualizzare tutti i fenomeno sopra elencati.
Per maggiori informazioni, si rimanda al testo stesso scritto da Anderson [7].

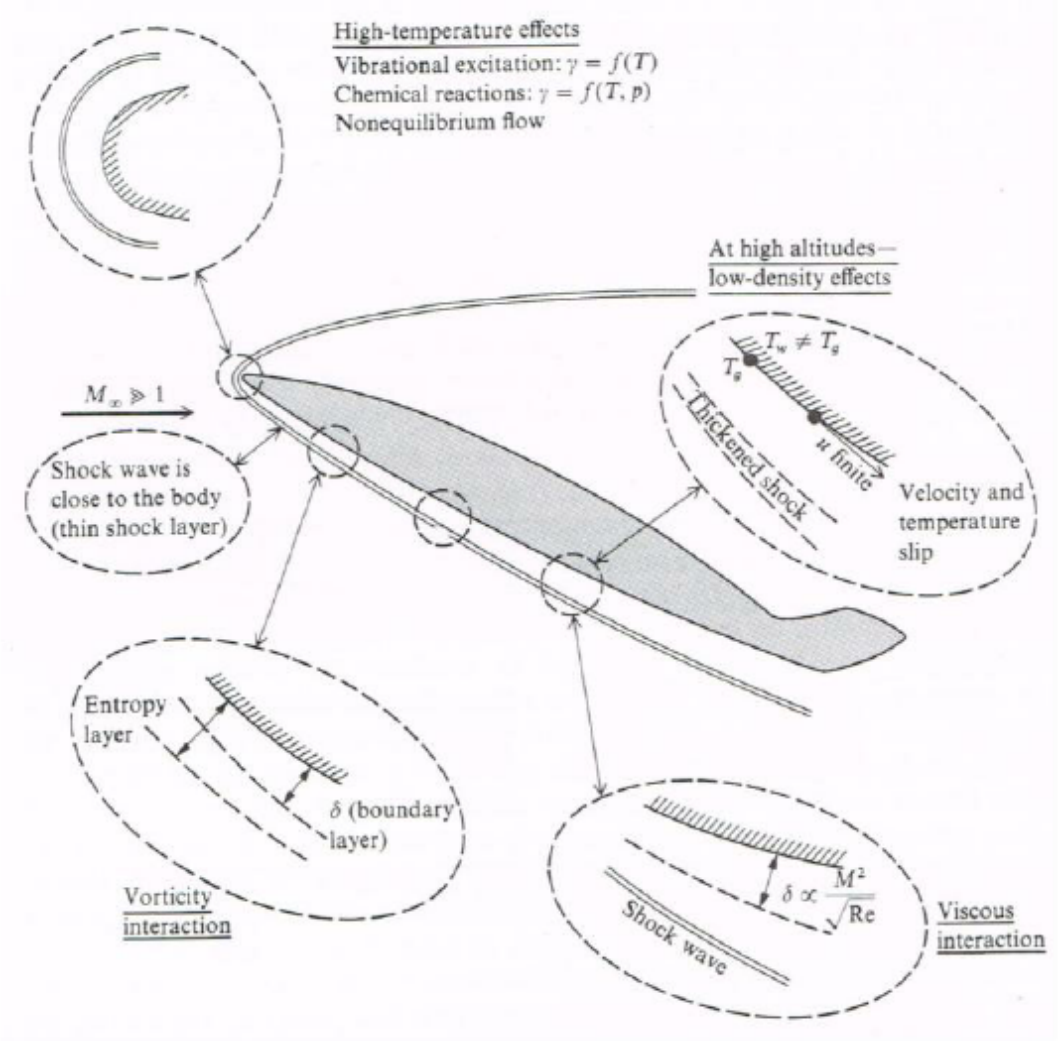


Figura 5.3: Fenomeni tipici del regime ipersonico. [7]

5.2 Effetti dovuti alle alte temperature

Usualmente nello studio della termodinamica e dell'aerodinamica, sia essa in regime comprimibile o incomprimibile, viene utilizzato un gas che rispetti la legge di stato dei gas perfetti $p = \rho RT$, i cui calori specifici c_p e c_v siano costanti, nel caso dell'aria $\gamma = c_p/c_v = 1.4$; questo gas viene chiamato *calorically perfect gas*.

Considerando sempre l'aria, nel caso in cui la temperatura superi 800 K, i gradi di libertà vibrazionali delle molecole presenti diventano eccitati, e il loro contributo non può più essere trascurato. I calori specifici diventano quindi funzioni della temperatura $c_{p,v} = f(T)$, e il gas viene chiamato *thermally perfect gas* [7].

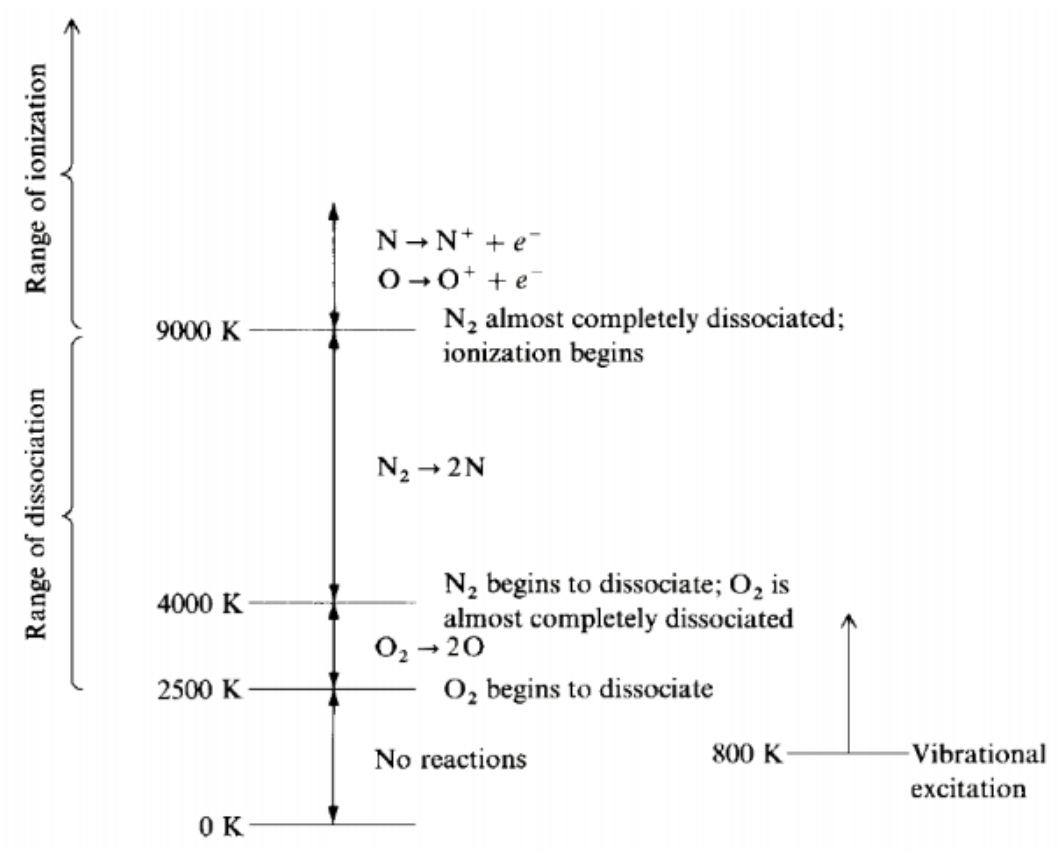


Figura 5.4: Fenomeni presenti nell'aria in funzione della temperatura a 1 atm. [7]

Nel caso in cui l'aria arrivi a temperature sempre più elevate, iniziano a diventare rilevanti le reazioni chimiche che potrebbero avvenire all'interno del gas. In questo caso i calori specifici sono funzione non solo della temperatura, ma anche della pressione $c_{p,v} = f(T, p)$. In Figura 5.4 viene dato un esempio delle reazioni chimiche presenti nell'aria al variare della temperatura, alla pressione fissata di 1 atm. In questo caso è quindi necessario considerare una miscela di gas perfetti reagenti

chimicamente; questa miscela può essere considerata in equilibrio, se al gas viene lasciato il tempo di evolvere liberamente verso una composizione stabile, o fuori equilibrio, se al gas non viene dato tempo sufficiente, in questo caso i calori specifici sono funzioni anche del tempo trascorso, e quindi della composizione istantanea, $c_{p,v} = f(T, p, t)$ [11].

5.2.1 Equilibrio chimico

In questa tesi verrà considerato il modello dell'aria composto da 5 specie: O_2, N_2, NO, O ed N. Ognuna di queste specie si comporta come un *thermally perfect gas*, ma è necessario considerare la miscele nel suo insieme per esprimere le variabile termodinamiche della miscela aria:

$$h, e, c_p, c_v = f(T, y_{O_2}, y_{N_2}, y_{NO}, y_O, y_N)$$

dove h è l'entalpia per unità di massa, e è l'energia interna per unità di massa e y_i è la frazione in massa dell'i-esima specie $y_i = m_i/m$. Si sottolinea come anche per la miscela valga l'equazione di stato dei gas perfetti:

$$\frac{p}{\rho} = \frac{\mathcal{R}}{\mathcal{M}} T$$

però in questo caso il peso molecolare della miscela $\mathcal{M}$ è un'incognita, poiché dipende dalla sua composizione [11]:

$$R = \frac{\mathcal{R}}{\mathcal{M}} = R \sum_{i=1}^{N_s} \frac{y_i}{\mathcal{M}_i} = R \sum_{i=1}^{N_s} q_i \tag{5.1}$$

dove N_s è il numero di specie presenti e q_i è il numero di moli della specie i-esima per unità di massa. Dato che la composizione all'equilibrio è funzione solamente della pressione e della temperatura, si ha che:

$$h, e, c_p, c_v = f(T, p)$$

Per trovare la composizione in equilibrio di una certa miscela è necessario innanzitutto conoscere le reazioni presenti tra le specie considerate. Per le applicazioni di interesse in questa tesi, sono considerate 17 reazioni per le 5 specie presenti (Tabella 5.1).

$O_2 + O \rightleftharpoons 2O + O$
$O_2 + N \rightleftharpoons 2O + N$
$O_2 + NO \rightleftharpoons 2O + NO$
$O_2 + O_2 \rightleftharpoons 2O + O_2$
$O_2 + N_2 \rightleftharpoons 2O + N_2$
$N_2 + O \rightleftharpoons 2N + O$
$N_2 + N \rightleftharpoons 2N + N$
$N_2 + NO \rightleftharpoons 2N + NO$
$N_2 + O_2 \rightleftharpoons 2N + O_2$
$N_2 + N_2 \rightleftharpoons 2N + N_2$
$NO + O \rightleftharpoons N + O + O$
$NO + N \rightleftharpoons N + O + N$
$NO + NO \rightleftharpoons N + O + NO$
$NO + O_2 \rightleftharpoons N + O + O_2$
$NO + N_2 \rightleftharpoons N + O + N_2$
$NO + O \rightleftharpoons N + O_2$
$N_2 + O \rightleftharpoons N + NO$

Tabella 5.1: Lista delle 17 reazioni presenti nel modello di aria considerato a 5 specie. Il primo blocco di cinque reazioni è quello di dissociazione di O_2, il secondo di N_2 e il terzo di NO. In ogni blocco le prime due reazioni utilizzano come terzo corpo un atomo, le altre tre, invece, una molecola. L'ultimo blocco contiene due reazioni di scambio. [11]

Le costanti di equilibrio K_p sono così definite per ogni reazione:

$$K_p(T) = \prod_i^{N_s} p_i^{\nu_i} \tag{5.2}$$

dove p_i è la pressione parziale dell'i-esima specie e ν_i il coefficiente stechiometrico dell'i-esima specie per la reazione considerata. Esse sono funzione della sola temperatura e possono essere ricavate sperimentalmente o dalla termodinamica statistica. Conoscendo le costanti di equilibrio per ogni reazione, utilizzando la definizione di pressione parziale $\sum p_i = p$ e la conservazione del numero di atomi di ogni elemento, è possibile ricavare la composizione all'equilibrio della miscela aria, in funzione delle sole T e p [7].

5.2.2 Equilibrio vibrazionale

L'approssimazione di Born-Oppenheimer consente di esprimere l'energia interna per unità di massa in questo modo:

$$e = e_{TRA} + e_{ROT} + e_{VIBR} + e_{EL} \tag{5.3}$$

La prima componente è l'energia traslazionale, che secondo il teorema dell'equipartizione dell'energia è pari a:

$$e_{TRA} = \frac{3}{2}RT \tag{5.4}$$

poiché i gradi di libertà traslazionali sono 3.

La seconda componente è l'energia rotazionale, i cui gradi di libertà sono 0 per un atomo, 2 per una molecola diatomica o poliatomica lineare e 3 per una molecola poliatomica non lineare.

$$e_{ROT} = \frac{L}{2}RT \qquad L = 0, 2, 3 \tag{5.5}$$

La terza componente è l'energia vibrazionale; immaginando una molecola come dei singoli atomi uniti da molle, essa è legata all'energia potenziale immagazzinata dai legami intra-molecolari e all'energia cinetica dei singoli atomi che vibrano. Per questa componente si abbandona il teorema dell'equipartizione dell'energia e si sfruttano i risultati forniti dalla meccanica quantistica. Il modello dell'oscillatore armonico con numero infinito di livelli energetici, uno dei più semplici, fornisce questa espressione per l'energia vibrazionale legata all'i-esima specie:

$$e_{VIBR_i} = \frac{R_i \Theta_i^v}{e^{\Theta_i^v/T} - 1} \qquad \Theta_i^v = \frac{h\nu_i}{k_b} \tag{5.6}$$

dove h è la costante di Planck, k_b la costante di Boltzmann e Θ_i^v la temperatura vibrazionale caratteristica dell'i-esima specie.

L'ultima componente è l'energia elettronica, la quale è stata trascurata nell'ambito di questa tesi. Si sottolinea nuovamente come, per un singolo atomo, l'energia rotazionale e vibrazionale siano nulle.

Tutte queste frazioni di energia sono quantizzate, perciò è necessario considerare anche la *zero-point energy* per ogni componente; essendo però interessati non al valore assoluto dell'energia interna, ma alle sue variazioni all'interno del dominio di calcolo, è possibile dimostrare che la variazione nella *zero-point energy* in una reazione chimica è uguale alla differenza tra i calori di formazione allo zero assoluto dei prodotti meno quella dei reagenti [11, 7]. In questo modo è possibile definire una pseudo energia interna per la miscela:

$$e = \sum_{i=1}^{N_s} y_i \left(e_{TRA_i} + e_{ROT_i} + e_{VIBR_i} + (\Delta h_f)_i^0 \right) \tag{5.7}$$

che non sarà propriamente la vera energia interna, ma se utilizzata per calcolare le variazioni di energia o entalpia tra due stati diversi, fornirà il valore esatto.

5.2.3 Non equilibrio vibrazionale e chimico

Nel caso in cui non venga dato sufficiente tempo a un gas di evolvere verso una condizione di equilibrio, sia dal punto di vista chimico che dell'energia vibrazionale, è necessario considerare il transitorio a cui esso è sottoposto. I fenomeni considerati sono legati al numero di collisioni tra le molecole e gli atomi presenti nel gas; affinché si raggiunga equilibrio roto-traslazionale sono necessarie $O(10)$ collisioni tra le molecole (numero relativamente basso, perciò questi gradi di libertà sono considerati sempre in equilibrio), per avere invece equilibrio vibrazionale sono necessarie $O(10^4)$ collisioni, e affinché una reazione di dissociazione avvenga un numero ancora superiore [11]; tutte queste collisioni, però, richiedono un certo tempo per avvenire.

L'equazione da considerare per il non equilibrio vibrazionale è una classica equazione di rilassamento, in cui il termine sorgente è proporzionale alla lontananza dalla condizione di equilibrio e inversamente proporzionale a una costante di tempo caratteristica:

$$\frac{\mathrm{d}e_i^v}{\mathrm{d}t} = \frac{e_i^{v\ eq} - e_i^v}{\tau_i} \tag{5.8}$$

Data l'equazione 5.6, è possibile ricavare un valore della temperatura direttamente proporzionale all'energia vibrazionale, chiamata temperatura vibrazionale:

$$T_i^v = \frac{\Theta_i^v}{\ln\left(\frac{R_i\Theta_i^v}{e_i^v} + 1\right)} \tag{5.9}$$

Ovviamente, in caso di equilibrio vibrazionale, la temperatura vibrazionale coinciderebbe con quella generica considerata.

L'equazione 5.8 per il non equilibrio vibrazionale è valida per tutte le specie diatomiche all'interno del gas, ovvero, nel caso dell'aria qui considerata, O_2, N_2 e NO. Per ognuna di queste specie, Park [31] identifica 5 costanti di rilassamento vibrazionale τ_{ij}^v, dove il pedice j indica il secondo partner collisionale della specie considerata (che può essere un'altra molecole della stessa specie, o una delle 4 specie rimanenti). A partire da questi 5 tempi di rilassamento è possibile ricavare un'unica costante di tempo per la molecola i utilizzando questa formula:

$$\tau_i^v = \frac{1}{\mathcal{M}\sum_{j=i}^{N_s}\frac{q_j}{\tau_{ij}^v}} \tag{5.10}$$

Anche nel caso del non equilibrio chimico si può scrivere una equazione simile alla 5.8 per ogni specie, contenente un termine sorgente; per fare questo, però,

è necessario introdurre alcuni concetti legati alle reazioni chimiche, come fatto in [11, 7].

Si immagini di avere una generica reazione di dissociazione dell'ossigeno:

$$O_2 + M \rightleftharpoons 2O + M$$

dove M è un generico terzo corpo della reazione. La variazione di concentrazione $[O]$ (numero di moli di O per unità di volume) dovuta alla reazione in avanti è:

$$\left(\frac{\mathrm{d}[O]}{\mathrm{d}t}\right)_f = 2k_f[O_2][M] \tag{5.11}$$

dove k_f è la costante di velocità in avanti della reazione, funzione della sola temperatura. La stessa reazione può però avvenire anche nella direzione opposta, e la sua velocità di reazione è legata alla *backward reaction rate* k_b. Per cui la velocità di reazione netta è pari a:

$$\frac{\mathrm{d}[O]}{\mathrm{d}t} = 2k_f[O_2][M] - 2k_b[O]^2[M] \tag{5.12}$$

e questa è pari a 0 in condizioni di equilibrio. La formula per una generica specie è:

$$\sum_{i=1}^{N_s} \nu_i' X_i \rightleftharpoons \sum_{i=1}^{N_s} \nu_i'' X_i \Rightarrow \frac{\mathrm{d}[X_i]}{\mathrm{d}t} = (\nu_i'' - \nu_i') \left(k_f \prod_{j=1}^{N_s} [X_j]^{\nu_j'} - k_b \prod_{j=1}^{N_s} [X_j]^{\nu_j''} \right) \tag{5.13}$$

Le due costanti di velocità k_f e k_b sono legate tra loro da una costante di equilibrio basata sulle concentrazioni, legata alla costante di equilibrio vista precedentemente nell'Equazione 5.2:

$$K_c = \frac{k_f}{k_b} = \frac{K_p}{\mathcal{R}T} \tag{5.14}$$

In questa tesi le k_f per ognuna delle 17 reazioni considerate (Tabella 5.1) sono state ricavate utilizzando l'equazione modificata di Arrhenius con i coefficienti indicati da Park [31, 32]:

$$k_f = CT^n e^{-T_r/T_a} \tag{5.15}$$

dove C, n e T_r sono i coefficienti di Arrhenius e T_a è una temperatura di controllo (pari a $\sqrt{TT^v}$ per le 15 reazioni di dissociazione).

In maniera simile a quanto fatto nell'Equazione 5.8, si può quindi scrivere per ogni specie:

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \Omega_i = \mathcal{M}_i \sum_{r=1}^{N_r} (\nu_i'' - \nu_i') \left(k_f \prod_{j=1}^{N_s} [X_j]^{\nu_j'} - k_b \prod_{j=1}^{N_s} [X_j]^{\nu_j''} \right) \tag{5.16}$$

dove N_r è il numero di reazioni totali. Quindi il termine sorgente Ω_i della specie i-esima è legato alla velocità di ogni reazione contenente quella specie.

5.2.4 Equazioni di Eulero per un gas in non equilibrio

Per ottenere delle equazioni in grado di descrivere un fluido inviscido in non equilibrio vibrazionale e chimico è necessario includere quanto esposto precedentemente alle equazioni di Eulero. Alle 3 equazioni classiche:

$$\frac{\partial}{\partial t}\int_V \rho \,\mathrm{d}V + \int_S \rho \boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{5.17}$$

$$\frac{\partial}{\partial t}\int_V \rho \boldsymbol{v} \,\mathrm{d}V + \int_S \rho \boldsymbol{v}\boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{5.18}$$

$$\frac{\partial}{\partial t}\int_V E \,\mathrm{d}V + \int_S (E+p)\boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{5.19}$$

bisogna unire un'equazione di rilassamento per l'energia vibrazionale per ogni specie biatomica:

$$\frac{\partial}{\partial t}\int_V \rho e_i^v \,\mathrm{d}V + \int_S \rho e_i^v \boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S = \int_V \rho \frac{e_i^{v\ eq} - e_i^v}{\tau_i}\,\mathrm{d}V \tag{5.20}$$

e un certo numero di equazioni di conservazione della specie, con il termine sorgente di cui sopra:

$$\frac{\partial}{\partial t}\int_V \rho_i \,\mathrm{d}V + \int_S \rho_i \boldsymbol{v}\cdot\boldsymbol{n}\,\mathrm{d}S = \int_V \Omega_i \,\mathrm{d}V \tag{5.21}$$

In particolare, per un fluido inviscido è sufficiente aggiungere un numero di equazioni pari al numero delle specie (5 nel nostro caso), meno il numero degli elementi coinvolti (2, N e O), meno 1 (si ha già l'equazione di conservazione della massa totale). Questo perché è possibile utilizzare anche due equazioni di conservazione del numero di atomi presenti nella cella fluida, dato che non è presente diffusione.

Il non equilibrio chimico influenza quindi la conservazione della massa, mentre il non equilibrio vibrazionale influenza l'equazione dell'energia; infatti, e_i^v è una componente dell'energia interna per le specie biatomiche:

$$E = \rho e + \frac{1}{2}\rho|\boldsymbol{v}|^2 = \rho \sum_{i=1}^{N_s}\left[y_i\left(\frac{3}{2}R_iT + \frac{L}{2}R_iT + e_i^v + (\Delta h_f)_i^0\right)\right] + \frac{1}{2}\rho|\boldsymbol{v}|^2 \tag{5.22}$$

Si ricorda come sia necessario considerare il fluido in non equilibrio nel caso in cui i tempi caratteristici per raggiungere l'equilibrio chimico o vibrazionale siano comparabili con i tempi caratteristici del fenomeno considerato, il fluido quindi non può essere considerato nè in equilibrio nè congelato [7].

5.3 Parziale implementazione e utilizzo di Cantera

Per implementare quanto esposto nella sezione precedente all'interno di **ImmerFlow** sono necessari vari step. Innanzitutto è necessario creare vari database contenenti dati sugli elementi, ad esempio il peso molecolare, sulle specie utilizzate, ad esempio la geometria e le proprietà termodinamiche, e infine sulle reazioni, ad esempio i coefficienti stechiometrici e i vari coefficienti di velocità. Utilizzando questi dati si devono poi creare delle funzioni e dei metodi in `C++` per il calcolo dei termini sorgenti delle equazioni 5.20 e 5.21; infine l'ultimo step consiste nell'includere questi metodi all'interno dell'ambiente di **ImmerFlow**, avendo cura di risolvere le due nuove equazioni di cui sopra con un metodo implicito, data la natura *stiff* delle equazioni di rilassamento nel caso in cui ci siano basse costanti di tempo. In questa tesi sono stati affrontati i primi passi di questa implementazione.

5.3.1 Input file di Cantera

Per la parte riguardante i database di informazioni necessarie, in particolare per il calcolo dell'equilibrio e delle sue costanti, è stato utilizzato il software open-source **Cantera**. Questo software contiene al suo interno un insieme di tools atti alla risoluzione di problemi di natura chimica e termodinamica [1]. Particolarmente utile risulta essere la gestione dei database da parte di questo programma; infatti, esso permette di definire delle fasi con cui lavorare (può gestire anche i fluidi multifase e la presenza di interfacce tra di esse) e tutti i meccanismi e le reazioni presenti al suo interno. I file di input utilizzati hanno estensione `.cti` e possono essere creati a partire dai file input di **CHEMKIN**, un altro famoso software per la risoluzione di problemi chimici. Oltre ad avere la possibilità di creare manualmente un file di input (come fatto in questa tesi), sono presenti numerosi database precompilati per la gestione di diversi fenomeni, ad esempio la combustione del gas naturale può essere gestita attraverso il meccanismo GRI-Mech 3.0 (325 reazioni) [19]. L'utilizzo di **Cantera** garantisce un'estrema riusabilità del codice, essendo esso sviluppato per gestire problemi di diversa natura, in particolare per quanto riguarda la combustione.

Un file di input è stato creato in modo da rappresentare l'aria a 5 specie di cui sopra:

```
ideal_gas(name = "air5",
                  elements = " O  N ",
                  species = """ O  O2  N  NO  N2 """,
                  reactions = "all",
                  transport = "Mix",
                  initial_state = state(temperature = 300.0,
                  pressure = OneAtm,
                  mass_fractions = 'O2:0.21, N2:0.79')     )
```

Per ogni specie è poi stato utilizzato il fit polinomiale a 9 coefficienti della NASA [20] per ottenere informazioni sulle proprietà termodinamiche. I coefficienti forniti dalla NASA sono utili a calcolare, come polinomi di quarto grado funzioni della temperatura (i coefficienti del polinomio cambiano in base al range di temperatura considerato), il calore specifico a pressione costante, l'entalpia molare e l'entropia molare della singola specie considerata.

```
species(name = "N2",
            atoms = " N:2 ",
            thermo = (
                        NASA9( [   200.00,   1000.00],
                               [    2.210371497E+04, ...
```

Infine vengono elencate le 17 reazioni presenti, con i rispettivi coefficienti di Arrhenius:

```
reaction(   "O + NO <=> N + O2",[8.4E12,0,161300.7568])
reaction(   "O + N2 <=> N + NO",[5.7E12,0.42,357006.798736])
...
```

Dato che, per il calcolo del termine sorgente Ω_i dell'Equazione 5.21, è necessario utilizzare sia la k_f sia la k_b ma, solitamente, in letteratura i dati forniti riguardano solo la k_f [31, 32], è necessario ricorrere all'utilizzo della costante di equilibrio $K_c = k_f/k_b$. I database di **Cantera** sono utilizzati per ricavare la K_c; infatti, utilizzando i polinomi NASA9 per trovare il valore dell'energia libera di Gibbs per ogni specie alla temperatura considerata è possibile ricavare il valore di questa costante. Si rimanda alla documentazione stessa per maggiori informazioni [1].

5.3.2 Calcolo dei termini sorgente delle equazioni aggiuntive

Un programma è stato scritto in `C++` per calcolare tutti i termini necessari all'implementazione del non equilibrio chimico e vibrazionale. Si è dovuta porre particolare attenzione alle diverse unità di misura utilizzate da **Cantera**, in particolare nel ricavare grandezze come la massa molare della miscela a partire da quella delle singole specie e delle frazioni molari.

Inizialmente sono stati creati degli oggetti a partire da varie classi, in particolare la classe `ThermoPhase` è stata utilizzata per creare una fase di `air5`, mentre la classe `Kinetics` è utilizzata per la gestione delle reazioni all'interno di questa fase e servirà successivamente per ricavare i coefficienti stechiometrici di ogni reazione e le costanti di equilibrio.

```
//creazione oggetti Cantera
std::unique_ptr<Cantera::ThermoPhase>
 gas(Cantera::newPhase("air5.cti","air5"));
```

```
std::vector<Cantera::ThermoPhase*> phases{gas.get()};
std::unique_ptr<Cantera::Kinetics>
 kin(newKineticsMgr(gas->xml(), phases));
gas->setState_TPY(T, n_atm*Cantera::OneAtm, compstring);
//costante di equilibrio di ogni reazione
Cantera::vector_fp keqcant(kin->nReactions());
kin->getEquilibriumConstants(keqcant.data());
```

Nel listato appena riportato si vede come sia molto facile e intuitivo l'utilizzo di **Cantera** per creare un vettore la cui dimensione sia pari al numero di reazioni presenti nella fase considerata (17 reazioni presenti nell'aria a 5 specie); in questo vettore vengono poi memorizzate tutte le costanti di equilibrio.

Inizialmente vengono calcolate le grandezze necessarie a ottenere l'energia vibrazionale fuori equilibrio, a partire dai 5 tempi di rilassamento per ogni specie biatomica:

```
//calcolo dei singolo tempi di rilassamento s-j
for (size_t s = 0; s < gas->nSpecies(); s++) {
 for (size_t j = 0; j < gas->nSpecies(); j++) {
  if (s==sO || s==sN) {
   vibrational_relaxation_time_sj[s][j]=0;
  } else {
   vibrational_relaxation_time_sj[s][j]=(std::exp
    (ab_vib[s][j][0]*(std::pow(T, -1.0/3.0)-
     ab_vib[s][j][1]))-18.421*101325)/p;
  }
 }
}
```

A partire da questi viene poi calcolato il tempo di rilassamento finale per ogni specie biatomica, insieme alla sua energia vibrazionale all'equilibrio (utilizzando le formule della sezione precedente):

```
//calcolo evib in caso di quilibrio
//calcolo tempo di rlassamento finale per ogni specie
for (size_t s = 0; s < gas->nSpecies(); s++) {
 if (s==sO || s==sN) {
  vibrational_energy_species_equilibrium[s]=0;
  vibrational_relaxation_time[s]=0;
  vibrational_temperature[s]=0;
 }
 else {
  vibrational_energy_species_equilibrium[s]=R
   *characteristic_vibrational_temperature[s]*1/
```

```
      (std::exp(characteristic_vibrational_temperature
       [s]/T)-1);
   VBT=0;
   for (size_t j = 0; j < gas->nSpecies(); j++) {
    VBT+=moletomass_species[j]/vibrational_relaxation_
     time_sj[s][j];
   }
   vibrational_relaxation_time[s]=1/moleculare_weight_
    mixture/VBT;
  }
}
```

Questi termini sono quindi sufficienti, insieme all'energia vibrazionale istantanea e_i^v, input proveniente da **ImmerFlow** una volta implementato il programma al suo interno, al calcolo dei termini sorgenti dell'equazione 5.20.

```
//species_i vibrational energy production vector
for (size_t s = 0; s < gas->nSpecies(); s++) {
 if (s==sO || s==sN) {
  evib_production_species[s]=0;
 }
 else {
  evib_production_species[s]=rho*(vibrational_energy_species_
   equilibrium[s]-vibrational_energy_species[s])/vibrational_
    relaxation_time[s];
 }
}
```

Viene anche riportato il codice che consente di calcolare l'energia interna per unità di mole della singola specie e dell'intera miscela; dopo essere stata calcolata in questo modo essa potrà essere inserita all'interno del bilancio dell'energia totale in **ImmerFlow**.

```
//traslational and rotational energy based on species geometry
for (size_t s = 0; s < gas->nSpecies(); s++) {
 std::shared_ptr<Cantera::Species>
  specie = gas->species(gas->speciesName(s));
 const Cantera::GasTransportData* sptran =
  dynamic_cast<Cantera::GasTransportData*>(specie->
   transport.get());
 if (sptran->geometry == "atom") {
  internal_energy_species[s] = 3.0/2.0*R*T+
   formation_entalphy[s];
 }
```

```
 else if (sptran->geometry == "linear") {
  internal_energy_species[s] = 5.0/2.0*R*T+
   formation_entalphy[s]+vibrational_energy_species[s];
 }
 else if (sptran->geometry == "nonlinear") {
  internal_energy_species[s] = 6.0/2.0*R*T+
   formation_entalphy[s]+vibrational_energy_species[s];
 }
}
//molar internal energy of the mixture
for (size_t s = 0; s < gas->nSpecies(); s++) {
 internal_energy+=gas->moleFraction(s)*internal_
  energy_species[s];
}
```

Per il calcolo dei termini sorgente delle equazioni 5.21 **Cantera** viene utilizzato per il calcolo delle costanti di equilibrio, come spiegato sopra, e per la gestione del database delle reazioni e dei suoi coefficienti stechiometrici. Il calcolo delle costanti di velocità in avanti k_f viene fatto manualmente poiché **Cantera** non consente di utilizzare una temperatura di controllo $T_a = \sqrt{TT_v}$; esso si basa, infatti, sulla più classica equazione modificata:

$$k_f = AT^b e^{-E_a/RT} \tag{5.23}$$

dove E_a è l'energia di attivazione. Quindi, i termini sorgente vengono così calcolati:

```
//species_i production vector
for (size_t s = 0; s < gas->nSpecies(); s++) {
 mass_production_species [s]=0;
 for (size_t r = 0; r < kin->nReactions(); r++) {
  REA=1,
  PRO=1;
  nu_reactant=kin->reactantStoichCoeff(s, r);
  nu_product=kin->productStoichCoeff(s, r);
  for (size_t ss = 0; ss < gas->nSpecies(); ss++) {
   if (kin->reactantStoichCoeff(ss, r)) {
    REA*=std::pow(moletovolume_species[ss],
     kin->reactantStoichCoeff(ss, r));
   }
   if (kin->productStoichCoeff(ss, r)) {
    PRO*=std::pow(moletovolume_species[ss],
     kin->productStoichCoeff(ss, r));
   }
```

```
    }
    mass_production_species[s]+=gas->molecularWeight(s)*
     (nu_reactant-nu_product)*(kfwd[r]*REA-kbwd[r]*PRO);
  }
}
```

Data la semplicità di questo termine, formato essenzialmente solo da produttorie, ne viene calcolato anche lo jacobiano, utile per la risoluzione implicita dell'Equazione 5.21:

```
//species_i production vector - jacobian
for (size_t j = 0; j < gas->nSpecies(); j++) {
 for (size_t s = 0; s < gas->nSpecies(); s++) {
  mass_production_species_J [s][j]=0;
  for (size_t r = 0; r < kin->nReactions(); r++) {
   REA=1,
   PRO=1;
   nu_reactant=kin->reactantStoichCoeff(s, r);
   nu_product=kin->productStoichCoeff(s, r);
   for (size_t ss = 0; ss < gas->nSpecies(); ss++) {
    if (kin->reactantStoichCoeff(ss, r) && ss!=j ) {
     REA*=std::pow(moletovolume_species[ss],
      kin->reactantStoichCoeff(ss, r));
    }
    else if (kin->reactantStoichCoeff(ss, r) && ss==j ) {
     REA*=std::pow(moletovolume_species[ss],
      kin->reactantStoichCoeff(ss, r)-1)*kin->
       reactantStoichCoeff(ss, r);
    }
    if (kin->productStoichCoeff(ss, r) && ss!=j) {
     PRO*=std::pow(moletovolume_species[ss],
      kin->productStoichCoeff(ss, r));
    }
    else if (kin->productStoichCoeff(ss, r) && ss==j) {
     PRO*=std::pow(moletovolume_species[ss],
      kin->productStoichCoeff(ss, r)-1)*kin->
       productStoichCoeff(ss, r);
    }
   }
   mass_production_species_J[s][j]+=gas->molecularWeight(s)*
    (nu_reactant-nu_product)*(kfwd[r]*REA-kbwd[r]*PRO);
  }
 }
```

}

In questo modo sono stati calcolati tutti i termini necessari all'utilizzo delle Equazioni 5.20 e 5.21.

5.3.3 Risultati

Per effettuare un controllo dei termini sorgente ricavati nella sezione precedente, in particolar modo quelli riguardanti l'Equazione 5.21, è stato implementato un caso 0D in cui una fase di aria composta al 21% da O_2 e al 79% da N_2 evolve nel tempo fino a raggiungere una composizione stabile a una temperatura elevata (4000 K nel seguente esempio).

Inizialmente le funzionalità di **Cantera** vengono utilizzate per calcolare la composizione in equilibrio del modello aria a 5 specie in funzione della temperature. Il comando `gas->equilibrate("TP")` consente di calcolare la composizione della miscela in equilibrio alla temperatura e alla pressione considerate; il solutore utilizzato per ottenere questo risultato si basa sull'algoritmo *VCS* sviluppato da Smith e Missen [35]. I risultati possono essere visualizzati nel Grafico 5.1 e corrispondono perfettamente con quanto descritto da [7]:

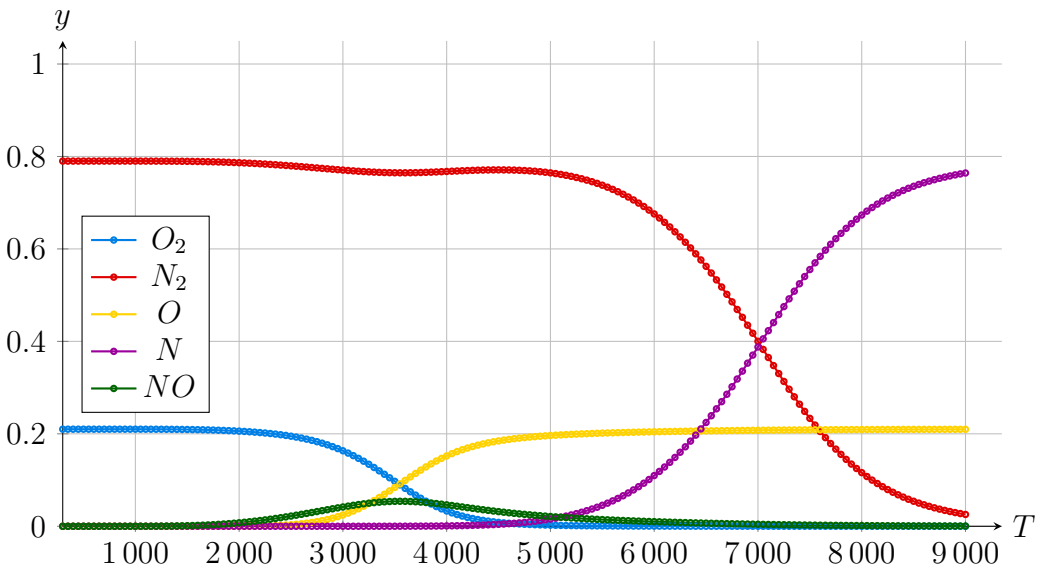


Grafico 5.1: Andamento della composizione di equilibrio al variare della temperatura per l'aria a 5 specie ottenuto tramite **Cantera**. Vengono utilizzate le frazioni in massa $y(X) = \frac{m_X}{m}$.

Una volta ottenuti questi valori di riferimento, è possibile procedere al controllo dei termini sorgente. Si immagini di avere un cilindro con pistone libero (la pressione è quindi costante) piena di aria composta al 21% da O_2 e al 79% da N_2 alla pressione

di 1 atm e a temperatura ambiente; si porti ora istantaneamente la temperatura all'interno del pistone a 4000 K e si mantenga costante (i.e. si continui a fornire energia). La composizione della miscela aria nel pistone cambierà fino a raggiungere, dopo alcune centinaia di nanosecondi, la sua composizione di equilibrio a quella temperatura. A partire dalla composizione iniziale, calcolando i termini sorgenti corrispondenti a quella composizione a 4000 K, si può integrare nel tempo la seguente equazione:

$$\frac{\mathrm{d}[X_i]}{\mathrm{d}t} = \frac{\Omega_i}{\mathcal{M}_i} = P_i([X_i]) = \sum_{r=1}^{N_r}(\nu_i'' - \nu_i')\left(k_f \prod_{j=1}^{N_s}[X_j]^{\nu_j'} - k_b \prod_{j=1}^{N_s}[X_j]^{\nu_j''}\right) \tag{5.24}$$

ottenendo, per un generico time step n (Eulero esplicito):

$$[\boldsymbol{X}]^{n+1} = [\boldsymbol{X}]^n + \Delta t \boldsymbol{P}([\boldsymbol{X}])^n \tag{5.25}$$

dove $[\boldsymbol{X}]^n$ è il vettore delle concentrazioni (numero di moli per unità di volume) per ogni specie al tempo n, e $\boldsymbol{P}([\boldsymbol{X}])^n$ il vettore dei termini di produzione calcolato utilizzando le composizioni al time step n e la temperatura fissata $T = 4000\ K$. Nel Grafico 5.2 si possono visualizzare i risultati ottenuti utilizzando $\Delta t = 0.1 \cdot 10^{-9}\ s$, rappresentati per semplicità utilizzando le frazioni in massa:

Dopo un transitorio della durata di circa $150 \cdot 10^{-9}\ s$, la composizione di equilibrio viene raggiunta. L'equilibrio raggiunto dopo $1 \cdot 10^{-6}\ s$ (Tabella 5.2) tramite questo metodo, ovvero integrando l'equazione 5.24, i cui termini sono ricavati nella sezione precedente, corrisponde perfettamente con quello ricavato utilizzando il solutore di **Cantera**; lo scarto quadratico medio calcolato tenendo conto delle 5 specie è pari a $\sigma = 6.13 \cdot 10^{-8}$.

$y(O_2)$	$y(N_2)$	$y(O)$	$y(N)$	$y(NO)$
$3.31 \cdot 10^{-2}$	$7.68 \cdot 10^{-1}$	$1.52 \cdot 10^{-1}$	$8.18 \cdot 10^{-4}$	$4.64 \cdot 10^{-2}$

Tabella 5.2: Composizione in equilibrio alla temperatura $T = 4000\ K$, raggiunta con metodo esplicito dopo $1 \cdot 10^{-6}\ s$ in 10000 iterazioni. Vengono utilizzate le frazioni in massa $y(X) = \frac{m_X}{m}$

Questa corrispondenza tra risultati ottenuti in maniera così diversa indica la correttezza del codice per il calcolo del termine di produzione Ω_i presentato nella sezione precedente.

Al fine di validare anche lo jacobiano del termine di produzione, è possibile ripetere quanto appena fatto utilizzando un metodo implicito invece del metodo di Eulero esplicito per l'integrazione nel tempo. L'equazione 5.24 può essere così integrata:

$$\frac{[\boldsymbol{X}]^{n+1} - [\boldsymbol{X}]^n}{\Delta t} = \boldsymbol{P}([\boldsymbol{X}])^{n+1} \tag{5.26}$$

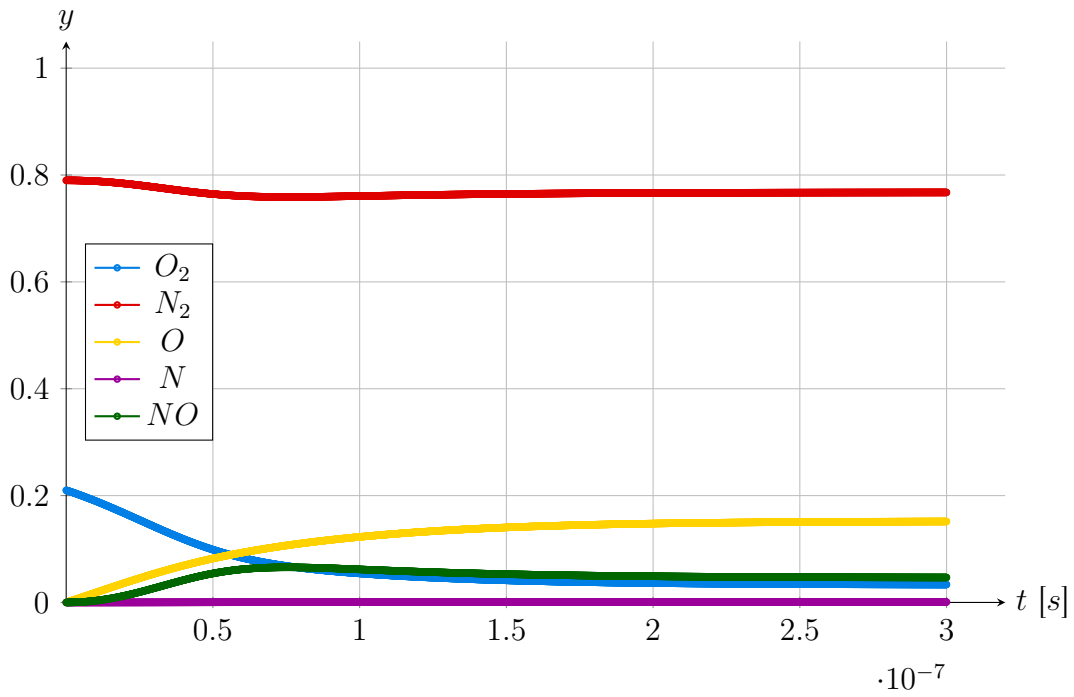


Grafico 5.2: Andamento della composizione in funzione del tempo alla temperatura fissata $T = 4000\ K$ e $p = 1\ atm$. $\Delta t = 0.1 \cdot 10^{-9}\ s$, metodo di Eulero esplicito. Vengono utilizzate le frazioni in massa $y(X) = \frac{m_X}{m}$.

$$\boldsymbol{P}([\boldsymbol{X}])^{n+1} \approx \boldsymbol{P}([\boldsymbol{X}])^{n} + \frac{\partial \boldsymbol{P}([\boldsymbol{X}])^{n}}{\partial [\boldsymbol{X}]^{n}} \left([\boldsymbol{X}]^{n+1} - [\boldsymbol{X}]^{n}\right) = \boldsymbol{P}([\boldsymbol{X}])^{n} + \overline{\boldsymbol{J}}^{n} \Delta[\boldsymbol{X}] \quad (5.27)$$

dove $\overline{\boldsymbol{J}}^{n}$ è la matrice jacobiana del termine di produzione al time step n e

$$\frac{\Delta[\boldsymbol{X}]}{\Delta t} = \boldsymbol{P}([\boldsymbol{X}])^{n} + \overline{\boldsymbol{J}}^{n} \Delta[\boldsymbol{X}] \quad (5.28)$$

$$\left(\frac{1}{\Delta t}\overline{\boldsymbol{I}} - \overline{\boldsymbol{J}}^{n}\right) \Delta[\boldsymbol{X}] = \boldsymbol{P}([\boldsymbol{X}])^{n} \quad (5.29)$$

dove $\overline{\boldsymbol{I}}^{n}$ è la matrice identità. Il sistema lineare 5.29 è stato risolto utilizzando la routine `dgesv` delle librerie LAPACK (viene utilizzata la decomposizione LU).

I risultati ottenuti tramite il metodo implicito sopra descritto possono essere visualizzati nel Grafico 5.3. Utilizzando un metodo implicito non si è limitati dalla stabilità del metodo, perciò si può scegliere un Δt estremamente maggiore di quello precedente. Dopo un tempo pari a $1 \cdot 10^{-6}\ s$, raggiunto con 20 iterazioni, il risultato

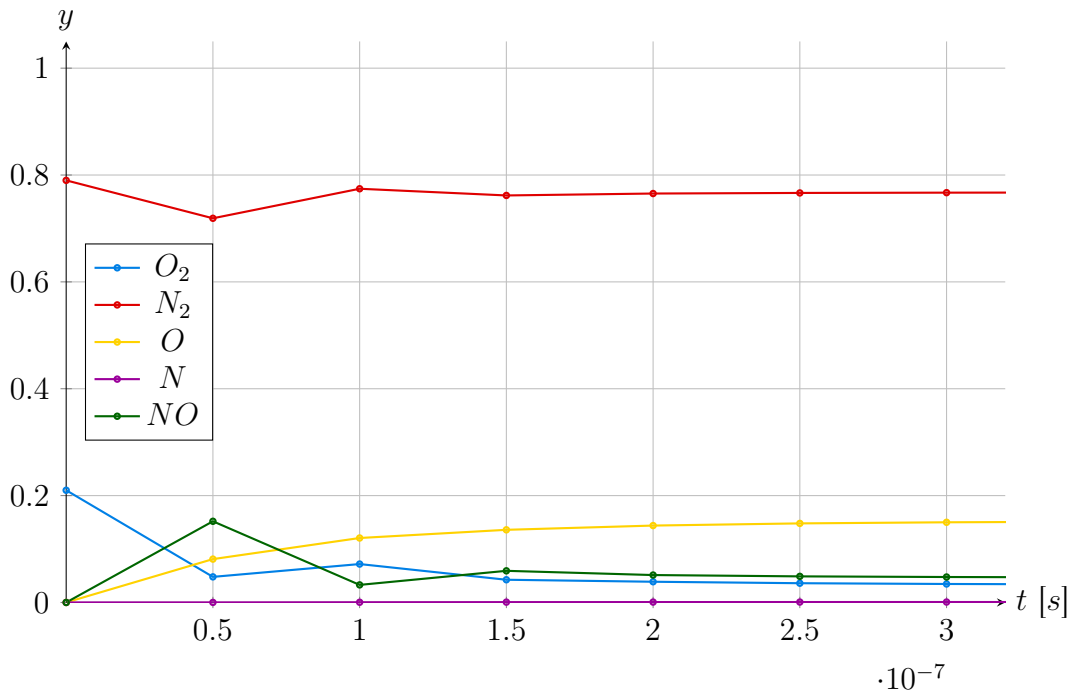


Grafico 5.3: Andamento della composizione in funzione del tempo alla temperatura fissata $T = 4000\ K$ e $p = 1\ atm$. $\Delta t = 0.5 \cdot 10^{-7}\ s$, metodo implicito. Vengono utilizzate le frazioni in massa $y(X) = \frac{m_X}{m}$.

finale è praticamente lo stesso di prima, con uno scarto quadratico medio rispetto all'equilibrio fornito da **Cantera** pari a $\sigma = 7.59 \cdot 10^{-8}$. Anche lo jacobiano del termine di produzione ottenuto nella sezione precedente risulta quindi essere corretto.

Si sottolinea come sia necessario interpretare con cautela i risultati appena ottenuti. Il codice scritto per il calcolo dei termini di produzione è esatto, ma le k_f utilizzate non sono sicuramente corrette; avendo utilizzato sia le k_f che le k_b, legate tra di loro dalla costante di equilibrio K_c, si può solo dedurre come quest'ultima, la costante di equilibrio, sia corretta. Utilizzando una k_f errata insieme alla K_c corretta si sarebbe raggiunto ugualmente l'equilibrio (ovviamente il codice sarebbe dovuto essere comunque corretto), ma il transitorio sarebbe stato errato. Fatta questa osservazione, però, dato che le k_f sono state semplicemente calcolate a partire da delle costanti fornite da Park [31, 32], si assume che anche esse siano corrette.

Non è stato possibile affrontare in questa tesi l'ultimo passo necessario a implementare quanto spiegato finora in **ImmerFlow**. Le equazioni aggiuntive citate nelle sezioni precedenti andrebbero infatti integrate all'interno del programma e risolte in maniera accoppiata alle altre equazioni esposte nel Capitolo 4. Questo passo non dovrebbe essere troppo impegnativo in quanto il software **ImmerFlow** è stata sviluppato dalla Optimad Engineering srl in maniera tale da poter facilmente inserire

un set di equazioni differenziali aggiuntive, quindi questo lavoro verrà sicuramente portato a termine nei mesi successivi.

6 Conclusioni

L'obiettivo principale di questa tesi era l'analisi e la validazione di un codice CFD ai volumi finiti con metodo IB in regime ipersonico. Date infatti le numerose differenze tra questo regime e quello supersonico, è necessario studiare l'attendibilità di un codice in presenza di questi fenomeni.

Nel Capitolo 2 è stato scritto un codice per l'applicazione dei metodi *immersed boundary* al caso unidimensionale del tubo d'urto. Sono stati comparati i risultati ottenuti tramite due metodi, uno basato sul lavoro svolto da Gorsse et al. [17] e l'altro sviluppato ex novo ispirandosi al più classico metodo *ghost-cell*. I risultati ottenuti coi due metodi sono simili e evidenziano le caratteristiche tipiche degli *immersed boundary*; è possibile imporre una condizione al contorno in una posizione interna alla griglia di calcolo, seppur con un piccolo aumento dell'errore. In questo modo è possibile utilizzare griglie non conformi al corpo immerso, con tutti i vantaggi che ne derivano.

I risultati ottenuti nel Capitolo 4 sono sufficienti a validare il codice in questione, **ImmerFlow**, per l'utilizzo in regime ipersonico. I moduli del picco di pressione e del flusso di calore a parete sono differenti da quelli ottenuti sperimentalmente, ma questa differenza può essere spiegata tenendo conto della bassa risoluzione spaziale delle prese di pressione presenti sul modello sperimentale; infatti, mediando spazialmente i valori ottenuti è possibile ottenere un andamento estremamente simile a quello sperimentale. La posizione angolare del picco di pressione risulta essere spostata di circa 4°, mentre la posizione del picco di flusso di calore a parete corrisponde meglio coi dati sperimentali, con una differenza di circa 0.3°. Dai risultati ottenuti si deduce come il raffinamento della griglia utilizzata sia fondamentale nel catturare con precisione il fenomeno presente; la facile gestione della taglia degli elementi della mesh in una griglia *octree* deve essere attentamente ponderata, è infatti necessario prestare molta attenzione al cambio di taglia degli ottanti, esso potrebbe influenzare la soluzione in maniera indesiderata.

Per arrivare a questo risultato è stato necessario operare alcune correzioni all'interno di **ImmerFlow**. La prima è stata l'implementazione della legge di Sutherland per il calcolo della viscosità e, attraverso il numero di Prandtl, della conducibilità termica; senza questa modifica infatti il valore della viscosità costante considerato avrebbe cambiato la posizione dell'urto incidente. La seconda modifica fatta riguarda la gestione dell'*immersed boundary*; è stato necessario cambiare il *tagging* delle celle in modo tale da non effettuare più l'estrapolazione della temperatura sull'inter-

faccia tra due celle, rischiando di raggiungere temperature negative, ma utilizzare solamente l'interpolazione.

Nel Capitolo 5 è stato fatto un primo passo verso l'implementazione degli effetti del non equilibrio chimico e vibrazionale. Il programma open-source **Cantera** è stato utilizzato per la gestione dei database necessari e un codice in `C++` è stato scritto per il calcolo dei termini sorgenti delle equazioni di conservazione aggiuntive. I termini di produzione di massa, incluso il loro jacobiano, sono poi stati validati confrontando la composizione in equilibrio calcolata col solutore interno a **Cantera** e la composizione derivante dall'integrazione nel tempo delle equazioni di conservazione della massa. I risultati ottenuti coincidono perfettamente, indicando la corretta scrittura del codice di cui sopra.

In futuro sarebbe possibile continuare il lavoro di validazione qui svolto utilizzando altri casi studio; molti di questi sono casi assialsimmetrici e questo renderebbe necessario aggiungere questa funzionalità al software utilizzato. Inoltre il lavoro di implementazione della parte aerotermochimica deve essere completato, integrando il codice scritto all'interno di **ImmerFlow**, in modo da poter risolvere in maniera accoppiata le classiche equazioni di Eulero insieme a quelle aggiuntive. Il codice scritto potrebbe poi essere ampliato per il calcolo delle proprietà di trasporto nel caso di non equilibrio chimico e vibrazionale, in modo da poter poi utilizzare le equazioni di Navier-Stokes.

A Metodo Ghost-Cell applicato al tubo d'urto 1D

Di seguito viene riportato il codice utilizzato per il metodo ispirato ai *ghost-cell methods*. Inizialmente si identifica la cella in cui cade il boundary Γ e viene effettuato il *tagging*. Gli estremi dell'*image cell* vengono utilizzati per calcolare le medie di cella delle variabili primitive, che sono poi utilizzate per ricavare i valori della *ghost cell*. Viene infine risolto il problema di Riemann all'interfaccia tra *ghost cell* e fluido.

```
//trovo coordinata centro cella image point
double phi=10000000;
double a,xim,ximl,ximr,h1,h2;
int tphi;
for (int t=1; t<n+1; t++) {
 if (std::abs(cellcenter[t]-dx/2.0-w)<phi) tphi=t;
 phi=std::min(std::abs(cellcenter[t]-dx/2.0-w),phi);
}
if (w<=(cellcenter[tphi]-dx/2.0)) {
 phi=dx-phi;
 tphi=tphi-1;
}
d=phi/dx;

a=(dx-phi)+dx/2.0;
xim=cellcenter[tphi-1]+dx/2.0+phi-a;
ximl=xim-dx/2.0;
ximr=xim+dx/2.0;

//hard immersed bc R
in[0].setPrimitive(inL[1].velocity(),inL[1].pressure(),
 inL[1].speedOfSound());
//std::cout << "d: " << d << std::endl;
if ((d==1.0)||(w==1.0)){
 uimag=in[tphi].velocity();
 pimag=in[tphi].pressure();
 aimag=in[tphi].speedOfSound();
```

```
 pengh[0]=penf[tphi][0];
 pengh[1]=penf[tphi][1];
 pengh[2]=penf[tphi][2];
}
if ((d<1.0)&&(d>0.5)) {
 v1[0]=0; v2[0]=0;
 v1[1]=0; v2[1]=0;
 v1[2]=0; v2[2]=0;
 h1=(cellcenter[tphi-1]+dx/2.0-ximl);
 h2=(-cellcenter[tphi]+dx/2.0+ximr);

 if ((ximl-cellcenter[tphi-1])<0) {
  reconL(ghl1,in[tphi-1],penf[tphi-1],
   (cellcenter[tphi-1]-ximl)*2.0);
  reconL(ghr2,in[tphi],penf[tphi],
   (cellcenter[tphi]-ximr)*2.0);
 }
 else if ((ximl-cellcenter[tphi-1])>0) {
  reconR(ghl1,in[tphi-1],penf[tphi-1],
   (ximl-cellcenter[tphi-1])*2.0);
  reconR(ghr2,in[tphi],penf[tphi],
   (ximr-cellcenter[tphi])*2.0);
 }
 else {
  ghl1[0]=in[tphi-1].velocity();
  ghr2[0]=in[tphi].velocity();
  ghl1[1]=in[tphi-1].pressure();
  ghr2[1]=in[tphi].pressure();
  ghl1[2]=in[tphi-1].speedOfSound();
  ghr2[2]=in[tphi].speedOfSound();
 }
 reconR(ghr1,in[tphi-1],penf[tphi-1],dx);
 reconL(ghl2,in[tphi],penf[tphi],dx);

 sommaVec(v1,ghl1);
 sommaVec(v1,ghr1);
 sommaVec(v2,ghl2);
 sommaVec(v2,ghr2);
 moltiplicaVecCost(v1,h1);
 moltiplicaVecCost(v2,h2);
 moltiplicaVecCost(v1,0.5);
```

```
 moltiplicaVecCost(v2,0.5);

 sommaVec(v1,v2);
 moltiplicaVecCost(v1,1.0/dx);

 uimag=v1[0];
 pimag=v1[1];
 aimag=v1[2];

 moltiplicaVecCost(penf[tphi-1],h1/dx);
 moltiplicaVecCost(penf[tphi],h2/dx);
 sommaVec(penf[tphi],penf[tphi-1]);
 pengh[0]=penf[tphi][0];
 pengh[1]=penf[tphi][1];
 pengh[2]=penf[tphi][2];
}
if (d==0.5){
 uimag=in[tphi-1].velocity();
 pimag=in[tphi-1].pressure();
 aimag=in[tphi-1].speedOfSound();
 pengh[0]=penf[tphi-1][0];
 pengh[1]=penf[tphi-1][1];
 pengh[2]=penf[tphi-1][2];
}
if ((d<0.5)&&(d>0.0)) {
 v1[0]=0; v2[0]=0;
 v1[1]=0; v2[1]=0;
 v1[2]=0; v2[2]=0;
 h1=(cellcenter[tphi-2]+dx/2.0-ximl);
 h2=(-cellcenter[tphi-1]+
  dx/2.0+ximr);

 if ((ximl-cellcenter[tphi-2])<0) {
  reconL(ghl1,in[tphi-2],penf[tphi-2],
   (cellcenter[tphi-2]-ximl)*2.0);
  reconL(ghr2,in[tphi-1],penf[tphi-1],
   (cellcenter[tphi-1]-ximr)*2.0);
 }
 else if ((ximl-cellcenter[tphi-2])>0) {
  reconR(ghl1,in[tphi-2],penf[tphi-2],
   (ximl-cellcenter[tphi-2])*2.0);
```

```
  reconR(ghr2,in[tphi-1],penf[tphi-1],
    (ximr-cellcenter[tphi-1])*2.0);
 }
 else {
  ghl1[0]=in[tphi-2].velocity();
  ghr2[0]=in[tphi-1].velocity();
  ghl1[1]=in[tphi-2].pressure();
  ghr2[1]=in[tphi-1].pressure();
  ghl1[2]=in[tphi-2].speedOfSound();
  ghr2[2]=in[tphi-1].speedOfSound();
 }
 reconR(ghr1,in[tphi-2],penf[tphi-2],dx);
 reconL(ghl2,in[tphi-1],penf[tphi-1],dx);

 sommaVec(v1,ghl1);
 sommaVec(v1,ghr1);
 sommaVec(v2,ghl2);
 sommaVec(v2,ghr2);
 moltiplicaVecCost(v1,h1);
 moltiplicaVecCost(v2,h2);
 moltiplicaVecCost(v1,0.5);
 moltiplicaVecCost(v2,0.5);

 sommaVec(v1,v2);
 moltiplicaVecCost(v1,1.0/dx);

 uimag=v1[0];
 pimag=v1[1];
 aimag=v1[2];

 moltiplicaVecCost(penf[tphi-2],h1/dx);
 moltiplicaVecCost(penf[tphi-1],h2/dx);
 sommaVec(penf[tphi-1],penf[tphi-2]);
 pengh[0]=penf[tphi-1][0];
 pengh[1]=penf[tphi-1][1];
 pengh[2]=penf[tphi-1][2];
}
in[tphi+1].setPrimitive(-uimag,pimag,aimag);
reconLgh(wl[tphi+1],in[tphi+1],pengh,dx);
inL[tphi+1].setPrimitive(wl[tphi+1][0],
 wl[tphi+1][1],wl[tphi+1][2]);
```

```
//bc destra
riemann::godunov( inR[tphi], inL[tphi+1],
 1, 0, interf[tphi], flux);
moltiplicaVecCost(flux,h);
sommaVec(fluxdiff[tphi],flux);
```

Bibliografia

[1] Cantera. https://cantera.org.

[2] Alan Chettle, Erinc Erdem, and Konstantinos Kontis. Edney iv interaction studies at mach 5. 07 2013.

[3] Cheng Chi, Bok Jik Lee, and Hong G. Im. An improved ghost-cell immersed boundary method for compressible flow simulations: An improved ghost-cell immersed boundary method. *International Journal for Numerical Methods in Fluids*, 83, 05 2016.

[4] Janice Christopher, É., and Glass. Numerical simulation of low-density shockwave interactions. 1999.

[5] John D. Anderson. *Computational fluid dynamics : the basics with applications*. 01 1995.

[6] John D. Anderson. *Modern Compressible Flow*. 01 2003.

[7] John D. Anderson. *Hypersonic and High-temperature Gas Dynamics*. AIAA education series. American Institute of Aeronautics and Astronautics, 2006.

[8] Joe D. Watts. Flight experience with shock impingement and interference heating on the x-15-2 research airplane. 11 1968.

[9] Domenic D'Ambrosio. Numerical prediction of laminar shock/shock interactions in hypersonic flow. *Journal of Spacecraft and Rockets - J SPACECRAFT ROCKET*, 40:11, 01 2006.

[10] Domenic D'Ambrosio. *Computational fluid dynamics : slide del corso*. 2018.

[11] Domenic D'Ambrosio. *High-temperature hypersonic flows : slide del corso*. 2018.

[12] Christopher E. Glass. Numerical simulation of low-density shock-wave interactions. 08 1999.

[13] B. Edney. *Anomalous Heat Transfer and Pressure Distributions on Blunt Bodies at Hypersonic Speeds in the Presence of an Impinging Shock*. FFA meddelande. Almqvist & Wiksell, 1968.

[14] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35 – 60, 2000.

[15] R. Ghias, R. Mittal, and H. Dong. A sharp interface immersed boundary method for compressible viscous flows. *Journal of Computational Physics*, 225(1):528 – 553, 2007.

[16] Peter Gnoffo. Cfd validation studies for hypersonic flow prediction. 02 2001.

[17] Yannick Gorsse, Angelo Iollo, Haysam Telib, and Lisl Weynans. A simple second order cartesian scheme for compressible euler flows. *Journal of Computational Physics*, 231(23):7780 – 7794, 2012.

[18] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43, 05 2001.

[19] GRI-Mech3.0. http://combustion.berkeley.edu/gri-mech/version30/text30.html.

[20] Bonnie J. McBride, Michael J. Zehe, and Sanford Gordon. Nasa glenn coefficients for calculating thermodynamic properties of individual species. 10 2002.

[21] Jacobus J. Van Der Vegt. Eno-osher schemes for euler equations. 02 1993.

[22] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics. ETH Zürich. Birkhäuser Basel, 2012.

[23] Cheng Liu and Changhong Hu. An immersed boundary solver for inviscid compressible flows. *International Journal for Numerical Methods in Fluids*, 85(11):619–640, 2017.

[24] M.-C Lo, C.-C Su, JS Wu, and K.-C Tseng. Modelling rarefied hypersonic reactive flows using the direct simulation monte carlo method. *Communications in Computational Physics*, 18:1095–1121, 10 2015.

[25] Sekhar Majumdar, Gianluca Iaccarino, and Paul Durbin. Rans solvers with adaptive structured boundary non-conforming grids. *Annual Research Briefs*, 01 2001.

[26] Marco Marini. Analysis of hypersonic compression ramp laminar flows under sharp leading edge conditions. *Aerospace Science and Technology*, 5(4):257 – 271, 2001.

[27] Rajat Mittal and Gianluca Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, 2005.

[28] J N. Moss, T Pot, Bruno Chanetz, and Michel Lefebvre. Dsmc simulation of shock/shock interactions: Emphasis on type iv interactions. 02 1999.

[29] A N. Semenov, M K. Berezkina, and I V. Krassovskaya. Classification of pseudo-steady shock wave reflection types. *Shock Waves*, 22, 07 2012.

[30] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49, 1988.

[31] Chul Park. Review of chemical-kinetic problems of future nasa missions, i: Earth entries. *Journal of Thermophysics and Heat Transfer*, 7, 10 1993.

[32] Chul Park, Richard Jaffe, and Harry Partridge. Chemical-kinetic parameters of hyperbolic earth entry. *Journal of Thermophysics and Heat Transfer - J THERMOPHYS HEAT TRANSFER*, 15:76–90, 01 2001.

[33] T Pot, Bruno Chanetz, Michel Lefebvre, and P Bouchardy. Fundamental study of shock-shock interference in low density flow: Flowfield measurements by dlcars. *21st Rarefied Gas Dynamics Symposium*, 2:545–552, 01 1998.

[34] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439 – 471, 1988.

[35] William Smith and Ronald W. (Ronald William) Missen. *Chemical reaction equilibrium analysis : theory and algorithms / William R. Smith, Ronald W. Miseen.* 01 1991.

[36] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978.

[37] Eleuterio Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* 03 2009.

[38] Kexin Wu, Guang Zhang, and Heuy Dong Kim. Study on the mach and regular reflections of shock wave. *Journal of Visualization*, pages 1–21, 01 2019.

[39] Huidan Yu. Lattice boltzmann equation simulations of turbulence, mixing, and combustion. 01 2004.