POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

# Miglioramento di modelli di turbolenza mediante inversione del campo

**Relatori**
Prof. Francesco Larocca
Dott. Andrea Ferrero

**Candidato**
Davide Balbo

Luglio 2019

# Sommario

Lo scopo della presente tesi è lo studio di un possibile metodo di miglioramento dei modelli di turbolenza RANS comunemente usati in campo fluidodinamico. La turbolenza è sicuramente un fenomeno fisico molto complesso e difficile da studiare ed analizzare correttamente, in ambito CFD infatti si ricorre solitamente a dei modelli che prevedono molte semplificazioni ed approssimazioni per poterla modellizzare e raggiungere quindi dei risultati attendibili. Teoricamente, utilizzando i metodi di simulazione DNS o LES si potrebbero raggiungere risultati molto più accurati, evitando gli errori dovuti alle approssimazioni dei modelli RANS, ma i casi in cui è possibile utilizzare quei metodi sono pochi, dato che sono applicabili solamente a test case a basso numero di Reynolds e hanno un costo computazionale elevatissimo. Proprio per questo motivo sono stati sviluppati diversi modelli di turbolenza, che risultano meno precisi ma largamente più utilizzabili, di cui uno dei più conosciuti è il modello di Spalart-Allmaras (SA) che verrà utilizzato anche per questo lavoro. Il suddetto modello si basa sulle equazioni RANS a cui viene aggiunta un'equazione di trasporto per la viscosità turbolenta che viene approssimata in funzione di un termine di produzione ed uno di distruzione. Il codice utilizzato per lo svolgimento dei calcoli è scritto in linguaggio Fortran e sfrutta il metodo di discretizzazione spaziale agli elementi finiti discontinui di Galerkin, mentre per la discretizzazione temporale permette di sfruttare lo schema esplicito di Runge-Kutta o quello implicito all'indietro di Eulero a seconda delle necessità. Come test case è stato scelto un semplice condotto curvo bidimensionale con parete superiore inviscida e parete inferiore soggetta, invece, agli effetti viscosi, discretizzato con una mesh strutturata consona alla geometria ed alle dimensioni del tubo. L'idea di base per cercare di migliorare il modello di turbolenza è di studiare la differenza tra i risultati generati con quest'ultimo e quelli di riferimento con la tecnica dell'inversione di campo, che prevede l'introduzione di un fattore correttivo nel termine sorgente del modello RANS. Per questa tesi è stato scelto il coefficiente d'attrito agente sulla parete inferiore del canale come variabile da utilizzare per monitorare la correttezza del modello studiato, di cui si hanno infatti a disposizione i risultati derivanti da una simulazione LES per lo stesso caso di studio, utilizzabili come dati di riferimento. Viene definita una funzione che esprime la differenza tra i dati di riferimento ed i risultati del modello scelto, chiamata funzione goal, di cui viene calcolato il

gradiente con il metodo dell'aggiunto per realizzare l'inversione di campo. Noto il gradiente, volendo minimizzare la funzione goal, è necessario muoversi in direzione opposta a quest'ultimo andando a trovare la distribuzione ottimale del fattore di correzione che minimizza la discrepanza rispetto ai dati di riferimento. Questo processo genererà una perturbazione nella soluzione stabile che si era precedentemente raggiunta, facendo quindi ripartire i calcoli con il metodo di SA modificato dal fattore moltiplicativo e generando quindi nuovi risultati che daranno una distribuzione del coefficiente d'attrito più simile a quella sperimentale, ottenendo quindi il miglioramento cercato del metodo. Ripetendo più volte tutto il procedimento si otterrà quindi una distribuzione quasi identica a quella sperimentale ed un modello di turbolenza nettamente più accurato. Per tentare di generalizzare il nuovo modello, ottenuto col fattore moltiplicativo del termine di produzione, rendendolo applicabile anche ad altri casi, si può pensare di estrarre la distribuzione del fattore moltiplicativo in funzione di tutte le grandezze fluidodinamiche del campo di moto per studiarne le dipendenze. Il set di dati così generato può essere utilizzato per l'addestramento di una rete neurale artificiale con gli algoritmi del machine learning, permettendo quindi di ottenere il fattore moltiplicativo direttamente dalle grandezze fluidodinamiche del campo di moto anche per altri casi di studio. Implementando la rete neurale all'interno del modello di turbolenza si ottiene infine un nuovo modello di turbolenza più accurato di quello originale sul problema per il quale è stato ottimizzato e, potenzialmente, anche per problemi simili.

# Ringraziamenti

Vorrei ringraziare tutti coloro che hanno cotribuito, direttamente ed indirettamente, al compimento del mio percorso di studi ed alla realizzazione di questa tesi, supportandomi ed aiutandomi durante questi 3 anni.

Ringrazio il Professor Larocca per avermi dato la possibilità di svolgere questa interessante ed innovativa tesi, fornendomi un'importante occasione per accrescere le mie conoscenze, e per i preziosi consigli che mi ha suggerito durante il lavoro e la stesura della relazione.

Ringrazio il Dottor Ferrero per la pazienza, la disponibilità ed il tempo che ha dedicato per aiutarmi nello svolgimento del lavoro, dando un contributo fondamentale al raggiungimento dei risultati ottenuti.

Ringrazio il Dottor Singh dell'Università del Michigan (USA) per aver risposto alla mia mail fornendomi i chiarimenti ed i dati richiesti in merito al suo articolo, utilizzato come importante termine di paragone per il mio lavoro.

Ringrazio l'iniziativa HPC@POLITO (http://www.hpc.polito.it) per aver fornito le risorse di calcolo utilizzate per questo lavoro di tesi.

Ringrazio Marika, la mia ragazza, per non aver mai smesso di supportarmi, spronandomi ad andare avanti e ad impegnarmi sempre al massimo, dandomi preziosi consigli e restandomi accanto anche nei momenti più difficili.

Ringrazio i miei genitori e tutta la mia famiglia per avermi permesso di svolgere questo importante percorso di studi, per tutti i sacrifici che ciò a richesto e per l'affetto e la sicurezza trasmessemi costantemente in questi anni.

Ringrazio infine, non per importanza, i miei amici che, ormai da una vita, mi aiutano e consigliano in tutte le scelte più importanti e le decisioni più difficili, senza farmi mai mancare il loro supporto sia nelle vittorie che nei fallimenti.

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Capitolo 1

# Introduzione

## 1.1 Machine learning e reti neurali

Al giorno d'oggi si sente sempre più spesso parlare di Intelligenza Artificiale (Artificial Intelligence AI) nel campo della tecnologia e, collegato a questa tematica, anche di machine learning e reti neurali. L'intelligenza artificiale può essere definita come la capacità di un sistema di risolvere problemi o svolgere attività tipici della mente e delle abilità umane che normalmente quindi non possono riguardare una macchina [1]. La ricerca in questo ambito è partita già negli anni '50, in parallelo con lo sviluppo dei primi calcolatori, ma solo successivamente ha suscitato l'interesse della comunità scientifica grazie allo sviluppo di macchine sempre più potenti e performanti. Attualmente questa tematica ha attirato l'attenzione anche del grande pubblico grazie all'elevato grado di sviluppo raggiunto che ne permette l'applicazione nei più svariati ambiti scentifici e tecnologici, arrivando anche ad essere implementata in oggetti di uso quatidiano come smartphone o PC.

Nell'ambito della ricerca sull'intelligenza artificiale uno degli argomenti cruciali riguarda il Machine Learning (ML) o apprendimento automatico, ovvero la capacità delle macchine di apprendere una funzione autonomamente dall'esperienza maturata, in modo simile a quanto fa un essere umano e senza essere state preventivamente programmate per svolgere quella specifica funzione [2]. La definizione attualmente più accreditata è quella di Tom Michael Mitchell della Carnegie Mellon University:

*"Si dice che un programma apprende dall'esperienza E con riferimento a alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E"*

Quindi, a differenza dei classici algoritmi di programmazione che dicono al sistema cosa fare passo passo, nel machine learning vengono forniti alla macchina una serie di dati su cui fare esperienza ed apprendere il miglior metodo per svolgere la funzione richiesta. Il concetto fondamentale di questo tipo di algoritmi è quindi l'apprendimento della macchina che può essere fatto in diversi modi:

- Apprendimento supervisionato, vengono forniti sia i dati di input che di output in modo che la macchina riconosca la logica di collegamento e la possa riutilizzare per applicazioni con dati simili;

- Apprendimento non supervisionato, vengono forniti solamente i dati di input e sarà la macchina ad individuare eventuali schemi o strutture logiche al loro interno

- Apprendimento per rinforzo, in questo caso la macchina svolge una funzione e tramite un sistema di ricompense e punizioni apprende il modo corretto in cui va fatta, ricevendo una ricompensa in caso di scelte corrette e una punizione in caso di errori, in modo che sia portata ad un'ottimizzazione della metodologia di svolgimento della funzione.

Oltre a questi che sono i principali metodi di apprendimento ne esistono altri che sfruttano logiche ibride alle tre citate, o anche metodi più complessi e specifici in base al caso di studio.

Uno dei campi di applicazione più comune degli algoritmi di machine learning è quello delle Reti Neurali Artificali (Artificial Neural Networks ANN), ovvero quei modelli matematici che si ispirano alla struttura del cervello umano, basata su moltissimi neuroni interconnessi tra loro su diversi livelli che ci permettono di ragionare e svolgere tutte le funzioni riguardanti la vita umana [3]. L'unità di calcolo basilare della rete viene infatti definita neurone artificiale e dalla connessione di molti di questi deriva la struttura della rete stessa, nella quale le informazioni vengono elaborate e processate in parallelo dai neuroni per poi essere scambiate e ricongiunte tramite le connessioni al fine di ottenere un risultato unico. L'architettura di una rete può essere molto complessa, articolandosi su diversi livelli e nodi di collegamento, a seconda ovviamente della complessità e del tipo di problema che si deve affrontare (Figura 1.1). Dato un problema si sceglie quindi l'architettura della rete neurale che viene poi addestrata con uno dei metodi del machine learning ed impara quindi a risolvere problemi simili, su cui viene successivamente utilizzata.

## 1.2 La modellazione della turbolenza

Una delle maggiori sfide della fluidodinamica computazionale (Computational Fluid Dynamics CFD) riguarda lo studio dei moti turbolenti dei fluidi [5]. Un fluido viene considerato in regime di moto turbolento quando, in determinate condizioni, sarà caratterizzato da un moto disordinato, imprevedibile e caotico definito da strutture con scale molto diverse quali vortici, bolle di ricircolo e moti secondari (Figura 1.2).

Questo regime di moto risulta molto complesso da studiare e prevedere proprio per le sue caratteristiche di irregolarità ed imprevedbilità, infatti l'unico metodo che viene considerato corretto e non approssimato è la simulazione numerica diretta

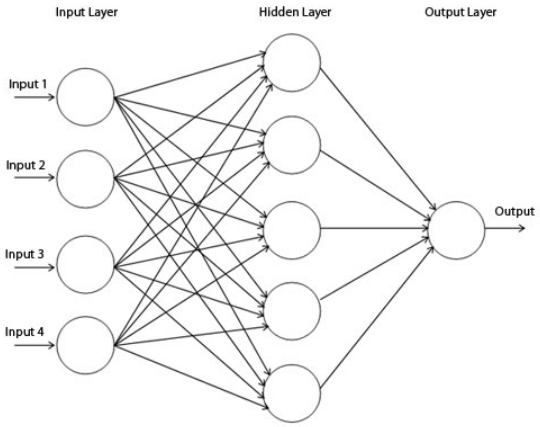


Figura 1.1. Esempio di architettura di una rete neurale artificiale (Fonte:[4])

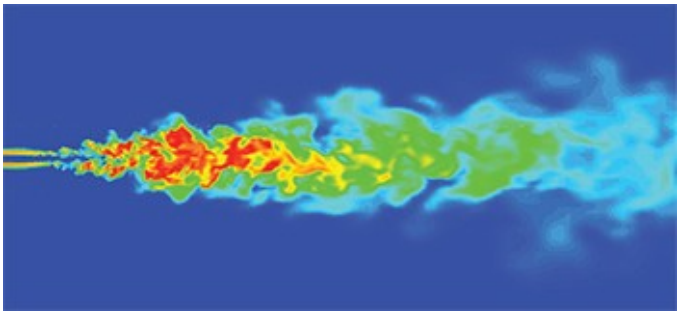

Figura 1.2. Moto turbolento (Fonte:[6])

(Direct Numerical Simulation DNS). Con la DNS le equazioni fluidodinamiche vengono risolte nella loro formulazione tridimensionale non-stazionaria, senza l'utilizzo di modelli empirici e parametrizzazioni. Questo permette di raggiungere risultati molto precisi ma con un costo computazionale elevatissimo che cresce rapidamente con il Reynolds del problema. Sono stati sviluppati dei metodi di modellazione della turbolenza con un costo computazionale più accettabile ma che danno dei risultati approssimati e non esatti, utilizzabili comunque per lo studio dei problemi fluidodinamici.

Uno degli approcci utilizzati per lo studio della turbolenza è quello della Large Eddy Simulation (LES) che si basa sul principio secondo il quale le strutture turbolente di dimensione maggiore, anisotropiche e dipendenti dal problema specifico, sono responsabili della maggior parte degli effetti legati alla turbolenza sul moto del fluido stesso mentre le più piccole, isotropiche e universali, hanno un'incidenza nettamente minore. Seguendo questo ragionamento l'approccio numerico della LES

cattura direttamente tramite una simulazione 3D non stazionaria le scale temporali e spaziali di dimensione maggiore, mentre quelle di dimensioni minori vengono affidate ad un opportuno modello che ne valuta gli effetti. Con questo approccio il costo computazionale scende ed il metodo è utilizzabile anche per problemi di una certa complessità, con livelli di turbolenza di interesse reale. Restano comunque alcuni problemi legati alla difficoltà nella scelta dei paramentri delle simulazioni e nell'interpretazione dei risultati che non sempre rendono la LES più utilizzabile della DNS nelle applicazioni pratiche.

Un'altro tipo di approccio è basato sui modelli di equazioni Reynolds Averaged Navier-Stokes (RANS) che si basano sull'idea per cui, ai fini ingegneristici, l'interesse è sempre relativo unicamente ai valori medi delle grandezze fluidodinamiche. Questo metodo scompone ogni caratteristica del moto turbolento in un temine medio, ottenuto tramite media temporale su un itervallo infinito per flussi stazionari e su un intervallo finito per flussi instazionari, ed un termine di disturbo o fluttuazione. Esistono diversi modelli che si basano su questo approccio:

- Spalart-Allmaras (SA, 1 equazione)
- $k - \epsilon$ (2 equazioni)
- $k - \omega$ (2 equazioni)
- Reynolds-Stress Model (RSM)

Sono tutti caratterizzati da un costo computazionale nettamente inferiore a DNS e LES che li rendono i modelli attualmente più utilizzati per problemi reali, nonostante un livello di approssimazione e quindi di incertezza maggiore.

Considerando che i modelli più utilizzabili per studiare la turbolenza in applicazioni reali sono quelli basati sulle RANS ma allo stesso tempo sono anche i meno affidabili, negli ultimi anni si è pensato di migliorarne la precisione, mantenendone il basso costo computazionale e la vasta applicabilità, lavorando con il machine learning e le reti neurali, descritti nel precedente paragrafo. In particolare l'idea di base è di intervenire dove vi sono delle approssimazioni, dato che si possono ottenere grosse migliorie addestrando delle reti neurali tramite gli algoritmi del machine learning utilizzando i dati sperimentali o quelli ottenuti tramite DNS/LES per ricercare il legame tra le variabili studiate ed il campo di moto turbolento [7]. Si parte sempre con l'addestramento delle reti su casi semplici, di cui si conoscono facilmente i risultati accurati, per poi passare ad applicare il modello con la rete implementata a problemi complessi ed analizzare i miglioramenti ottenuti. La maggior parte degli sforzi viene fatta nell'individuazione del miglior punto di applicazione della rete neurale nell'algoritmo del modello di turbolenza, oltre che ovviamente nel corretto e più ampio possibile addestremento della rete stessa in modo che possa poi essere utilizzata per i casi più diversi. Con questo metodo è possibile ottenere notevoli aumenti della precisione e dall'affidabilità dei vari modelli, avvicinandosi

molto ai dati sperimentali e delle DNS/LES, mantenendo però sempre un basso costo computazionale ed una generalità che ne permette un utilizzo per i casi reali. Nel corso degli anni sono stati utilizzati diversi approcci per fare quanto precedentemente spiegato; tramite le reti neurali Milano e Koumoutsakos hanno approssimato i termini turbolenti di ordine maggiore [8], Hoceva ha modellato alcune variabili della scia turbolenta [9], Duraisamy e colleghi hanno modellato il termine sorgente di SA ed hanno dedotto e ricostruito meglio le relazioni tra le variabili nella modellazione della turbolenza e dei transitori [10, 11] e molti altri ancora hanno ottenuto risultati rilevanti.

## 1.3 L'approccio di inversione del campo

Come già detto precedentemente, esistono diverse e molteplici possibilità di lavorare per migliorare i modelli di turbolenza già esistenti basati sulle RANS e cercare di incrementarne la correttezza e l'affidabilità. Vari tentativi sono già stati fatti in questi anni, ma sicuramente uno degli approcci più interessanti è quello presentato dal professor Duraisamy ed alcuni suoi collaboratori [10, 11], che verrà utilizzato come linea guida per la realizzazione di questa tesi. Uno dei grossi problemi dei modelli di turbolenza esistenti è che spesso contengono al loro interno diversi termini, variabili o costanti, che vengono scelti arbitrariamente dall'esperienza di chi ha progettato il modello e sono basati su un piccolo numero di semplici casi di studio reale. Questo, infatti, fa si che quando il modello viene applicato a casi differenti dai pochi testati vi sia una forte perdita di accuratezza, che rende il modello meno preciso ed utilizzabile. Per quanto si sia lavorato per migliorare la valutazione di questi termini, la loro dipendenza dal caso reale utilizzato per derivarli non può essere eliminata ma sicuramente può essere mitigata per aumentare la correttezza e la generalità dei modelli. Sfruttando la sempre crescente potenza di calcolo a disposizione attualmente è possibile ottenere risultati esatti, tramite simulazioni DNS o LES, per un vasto numero di casi reali differenti, da utilizzare poi per migliorare appunto i modelli di turbolenza RANS esistenti. Per ottenere questi miglioramenti Duraisamy propone di studiare le discrepanze e gli errori presenti nei risultati ottenuti tramite i modelli in confronto con quelli di riferimento, applicando quindi i diversi metodi di calcolo ad ogni test case singolarmente per ottenere complessivamente il maggior numero di informazioni possibili.

In particolare in questo caso verrà analizzato il modello di Spalart-Allmaras in cui la viscosità turbolenta viene calcolata mediante un'equazione di trasporto che contiene un termine di produzione ed uno di distruzione, per cui la formulazione corretta di questi due parametri risulta molto complessa e fonte dei problemi precedentemente citati. Nel metodo di inversione del campo il termine di produzione viene quindi moltiplicato per un fattore di correzione, il cui valore viene calcolato punto per punto tramite un'ottimizzazione effettuata con il metodo del gradiente calolato con il metodo dell'aggiunto, per avvicinare il più possibile la soluzione del

modello di turbolenza ai dati di riferimento. Successivamente viene tracciato l'andamento della suddetta variabile in funzione di una serie di parametri del campo di moto, in modo da poterne ricavare le relazioni presenti. Per ricavare queste relazioni è possibile, avendo a disposizione un sufficiente numero di dati, utilizzare una rete neurale da addestrare con gli algoritmi del machine learning. Un volta che la rete è stata debitamente addestrata potrà essere inclusa all'interno dell'algoritmo del modello di turbolenza ed utilizzata per ottenere le migliorie volute anche per casi di studio differenti da quelli utilizzati per effettuare il lavoro. Ovviamente maggiori e più differenti saranno i test case utilizzati per l'addestramento della rete, maggiore sarà l'affidabilità e la vastità del campo dei possibili utilizzi del modello con la rete implementata.

# Capitolo 2

# Descrizione modello fisico e CFD

## 2.1 Scelta del modello fisico

La prima ed importantissima fase di un qualsiasi lavoro riguardante la fluidodinamica è incentrata sulla scelta del modello fisico da utilizzare, cosa non semplice da fare in quanto questa decisione influenzerà tutto il progetto, dalla fattibilità alla veridicità dei risultati. Per effettuare la scelta nel miglior modo possibile occorre tener conto di svariati fattori, tra cui i casi di studio su cui si dovrà lavorare, la potenza di calcolo a disposizione ed il grado di conoscenza del modello stesso. Per questo progetto, volendo trattare le problematiche relative al moto turbolento dei fluidi, si è scelto di utilizzare il modello fisico descitto dalle RANS [12, 13] nella forma bidimensionale comprimibile.

Si scrivono quindi le 3 equazioni fondamentali che governano il moto dei fluidi:

- Equazione di conservazione della massa

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{2.1}$$

- Equazione di conservazione della quantità di moto

$$\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} = -\frac{1}{\rho} \nabla p \tag{2.2}$$

- Equazione di conservazione dell'energia

$$\frac{\partial E}{\partial t} + \vec{v} \cdot \nabla E = -\frac{1}{\rho} \nabla \cdot (\rho \vec{v}) \tag{2.3}$$

Le variabili utilizzate indicano: $\rho$ la densità, $p$ la pressione, $\vec{v}$ la velocità, $E$ l'energia totale. Per completare il set di equazioni serve ancora introdurre l'equazione di stato dei gas perfetti

$$p = \rho R T \tag{2.4}$$

Introducendo l'equazione di stato all'interno della definizione di energia totale si ottiene:

$$E = c_v T + \frac{(\vec{v})^2}{2} = c_v \frac{p}{\rho R} + \frac{(\vec{v})^2}{2} \tag{2.5}$$

in cui $T$ è la temperatura, $c_v$ il calore specifico a volume costante ed $R$ la costante universale dei gas.

Queste equazioni, dette anche di Eulero, non tengono però conto degli effetti diffusivi dovuti alla viscosità e degli scambi termici, vanno infatti modificate e scritte nella forma di Navier-Stokes

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho v_i) = 0 \tag{2.6}$$

$$\frac{\partial}{\partial t}(\rho v_i) + \frac{\partial}{\partial x_j}(\rho v_i v_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.7}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho v_j H) = \frac{\partial}{\partial x_j}(v_i \tau_{ij}) + \frac{\partial}{\partial x_j}\left(k \frac{\partial T}{\partial x_j}\right) \tag{2.8}$$

dove $v_{i,j}$ e $x_{i,j}$ indicano una generica componente del vettore della velocità e delle coordinate rispettivamente, mentre $\tau_{ij}$ è il tensore degli sforzi viscosi, assumendo l'ipotesi di Stoke per la viscosità di volume

$$\tau_{ij} = 2\mu\left(S_{ij} - \frac{1}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right) \quad \text{con} \quad S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.9}$$

dove $S_{ij}$ è il tensore della velocità di deformazione.

Per trasformare ulteriormente le equazioni ed ottenere il modello RANS le variabili di flusso vengono scomposte in un termine medio ed un termine fluttuante. Con questo approccio le equazioni di governo vengono poi risolte per i valori medi, in quanto i risultati che ne derivano risultano i più significanti e utili per le applicazioni ingegneristiche. I termini medi possono essere calcolati con il metodo di Reynolds, che prevede una media temporale (o d'insieme), o con il metodo di Favre, che tiene conto della densità variabile del flusso comprimibile, a seconda della grandezza in esame. Quando viene utilizzato il metodo di Reynolds la notazione prevede che i termini medi abbiano un trattino sovrapposto e i termini fluttuanti siano caratterizzati da un apice, mentre se viene utilizzato il metodo di Favre i termini medi presentano una tilde sovrapposta e i termini fluttuanti due apici. Si scrive quindi, per esempio

$$v_i = \tilde{v}_i + v_i'' \qquad \rho = \bar{\rho} + \rho' \tag{2.10}$$

Prendendo, per esempio, in esame la velocità, la media di Reynolds si ottiene con

$$\bar{v}_i = \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} v_i dt \tag{2.11}$$

mentre quella di Favre con

$$\tilde{v}_i = \frac{1}{\bar{\rho}} \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} \rho v_i dt \tag{2.12}$$

Si ottiene quindi il sistema di equazioni RANS per flussi compressibili, utilizzato come modello base del presente lavoro

$$\begin{cases} \dfrac{\partial \bar{\rho}}{\partial t} + \dfrac{\partial}{\partial x_i}(\bar{\rho}\tilde{v}_i) = 0 \\ \dfrac{\partial}{\partial t}(\bar{\rho}\tilde{v}_i) + \dfrac{\partial}{\partial x_j}(\bar{\rho}\tilde{v}_i\tilde{v}_j) = -\dfrac{\partial \bar{p}}{\partial x_i} + \dfrac{\partial}{\partial x_j}\hat{\tau}_{ij} \\ \dfrac{\partial \tilde{E}}{\partial t} + \dfrac{\partial}{\partial x_j}(\tilde{v}_j\tilde{H}) = \dfrac{\partial}{\partial x_j}(\tilde{v}_i\hat{\tau}_{ij} - q_j) \end{cases} \tag{2.13}$$

Tramite l'analogia di Reynolds e l'introduzione del numero di Prandtl turbolento viene, inoltre, calcolato il flusso di calore

$$q_i = -\frac{\gamma}{\gamma - 1}\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial T}{\partial x_i} \tag{2.14}$$

mentre il tensore degli sforzi viscosi totale, laminare e turbolento, vale

$$\hat{\tau}_{ij} = (2\mu + 2\mu_t)\left(S_{ij} - \frac{1}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right) \tag{2.15}$$

## 2.2 Scelta del modello di turbolenza

Come già spiegato precedentemente, di modelli per il calcolo della turbolenza ne esistono svariati in letteratura, ognuno con particolari caratteristiche e punti di forza ma comunque tutti soggetti a forti approssimazioni. Uno dei modelli più conosciuti ed utilizzati è sicuramente quello di Spalart-Allmaras [14], sfruttato anche per questo lavoro, che è basato ovviamente sulle RANS ed è caratterizzato da un buon livello di affidabilità anche per flussi altamente turbolenti e, soprattutto, da un relativa semplicità di implementazione [12]. Il modello prevede di calcolare gli sforzi di Reynolds utilizzando l'assunzione di Boussinesq per la viscosità cinematica turbolenta [15], che è data da

$$\nu_t = \tilde{\nu} f_{v1} \tag{2.16}$$

dove $\nu$ è la viscosità cinematica e $f_{v1}$ è la funzione di smorzamento

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{vi}^3} \qquad \chi = \frac{\tilde{\nu}}{\nu} \tag{2.17}$$

La variabile su cui lavora SA è $\tilde{\nu}$ che obbedisce all'equazione di trasporto

$$\frac{D\tilde{\nu}}{Dt} = P - D + \frac{1}{\sigma}[\nabla \cdot ((\nu + \tilde{\nu})\nabla\tilde{\nu}) + c_{b2}(\nabla\tilde{\nu})^2] \tag{2.18}$$

La formulazione precedente vale sia nel campo incomprimibile che in quello comprimibile, ma per il campo comprimibile si può ricavare una forma conservativa equivalente che verrà generata combinando la precedente con l'equazione di conservazione della massa. L'equazione di trasporto diventerà quindi

$$\frac{\partial}{\partial t}(\rho\tilde{\nu}) + \frac{\partial}{\partial x_j}(\rho v_j\tilde{\nu}) = \rho(P - D) + \frac{1}{\sigma}\frac{\partial}{\partial x_j}\left[\rho(\nu + \tilde{\nu})\frac{\partial\tilde{\nu}}{\partial x_j}\right] + \frac{c_{b2}}{\sigma}\rho\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k} - \frac{1}{\sigma}\frac{\partial\rho}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k} \tag{2.19}$$

con il termine di produzione $P$ e quello di distruzione $D$ pari a

$$P = c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} \qquad D = \left(c_{w1}f_w - \frac{c_{b1}}{k^2}f_{t2}\right)\left(\frac{\tilde{\nu}}{d}^2\right) \tag{2.20}$$

e la vorticità modificata pari a

$$\tilde{S} = \begin{cases} S + \bar{S} & \text{se } \bar{S} \geq -c_{v2}S \\ S + \dfrac{S(c_{v2}^2 S + c_{v3}\bar{S})}{(c_{v3} - 2c_{v2})S - S} & \text{se } \bar{S} < -c_{v2}S \end{cases} \tag{2.21}$$

$$\bar{S} = \frac{\tilde{\nu}}{k^2 d^2}f_{v2} \qquad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.22}$$

La funzione $f_w$ vale invece

$$f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{\frac{1}{6}} \quad g = r + c_{w2}(r^6 - r) \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}k^2d^2}, r_{\lim}\right) \tag{2.23}$$

ed i termini $f_t$

$$f_{t1} = c_{t1}g_t e^{-c_{t2}\frac{\omega_t}{\Delta u^2}(d^2 + g_t^2 d_t^2)} \qquad f_{t2} = c_{t3}e^{-c_{t4}\chi^2} \tag{2.24}$$

La viscosità dinamica turbolenta è definita come

$$\mu_t = \rho\tilde{\nu}f_{v1} \tag{2.25}$$

Le costanti utilizzate nel modello valgono: $c_{b1} = 0.1355$, $\sigma = 2/3$, $c_{b2} = 0.622$, $\kappa = 0.41$, $c_{w1} = c_{b1}/\kappa^2 = 0.806$, $c_{w2} = 0.3$, $c_{w3} = 2$, $c_{v1} = 7.1$, $c_{t1} = 1$, $c_{t2} = 2$, $c_{t3} = 1.3$, $c_{t4} = 0.5$, $r_{\lim} = 10$, $c_{v2} = 0.7$, $c_{v3} = 0.9$.

## 2.3 Discretizzazione spaziale

Dopo aver scelto il modello fisico da utilizzare e, conseguentemente, anche il modello specifico per la turbolenza ed averne ricavato il set di equazioni da risolvere, è necessario operare un'altra scelta fondamentale, ovvero il metodo di discretizzazione spaziale che permette di applicare il suddetto set di equazioni al dominio fisico del caso in esame. In ambito CFD i metodi più comunemente utilizzati sono tre: metodo alle differenze finite, metodo ai volumi finiti e metodo agli elementi finiti. Tutti presentano vantaggi e svantaggi caratteristici per cui la scelta sul quale utilizzare va fatta in base al campo di applicazione ed ai risultati che si vogliono ottenere. Per la realizzazione di questa tesi si è utilizzato un metodo di discretizzazione ibrido tra i volumi finiti e gli elementi finiti che prende spunto dai vantaggi di entrambi, il metodo di Galerkin discontinuo (Discontinuous Galerkin DG) [12]. Dai volumi finiti prende la definizione di numerosi flussi numerici che permettono di tenere conto della propagazione delle onde, mentre dagli elementi finiti prende l'elevato numero di gradi di libertà (degrees of freedom DOFs) presenti in ogni cella della mesh, che permettono di avere tutte le informazioni richieste all'interno, ed anche il metodo di mappatura degli elementi della mesh.

Ipotizzando di prendere una generica equazione di conservazione in 2D

$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = Q \qquad (x, y) \in \Omega \subset R^2, t \in R_0^+ \tag{2.26}$$

dove $u$ è la variabile che si conserva, $F$ e $G$ sono le componenti dei flussi e $Q$ è un generico termine di sorgente. Il dominio $\Omega$ viene quindi suddiviso in molti elementi più piccoli e non sovrapposti $\Omega_e$, all'interno dei quali la soluzione viene approssimata tramite lo spazio funzionale $V_h$ definito come

$$V_h = \{v \in L^2(\Omega) : v|_{\Omega_e} \in P^p(\Omega_e) \, \forall \Omega_e \in T_h\} \tag{2.27}$$

dove $P^p$ è lo spazio dei polinomi di ordine fino a $p$ definito all'interno del singolo elemento, tenendo conto che $p$ può variare tra i vari elementi non essendoci requisiti di continuità sugli stessi. La soluzione numerica viene approssimata tramite combinazione lineare delle $N_e$ funzioni di base di $V_h$

$$u_h(x, t) = \tilde{u} \cdot \Phi = \sum_{i=1}^{N_e} \tilde{u}_i \Phi_i \tag{2.28}$$

Le funzioni di base $\Phi$ possono essere scelte in diversi modi per il metodo di Galerkin discontinuo e questa scelta può influire pesantemente sulla robustezza della discretizzazione. In questo caso si sono utilizzate delle basi di tipo modale, che hanno il vantaggio di essere gerarchiche ed ortonormali, ottenute tramite la procedura di Gram-Schmidt modificata secondo l'approccio di Bassi e colleghi [16]. Sostituendo

$u_h$ nell'equazione di governo si ottengono i residui

$$R_h = \frac{\partial u_h}{\partial t} + \frac{\partial F_h}{\partial x} + \frac{\partial G_h}{\partial y} - Q \tag{2.29}$$

che vanno poi proiettati nello spazio $V_h$ e posti uguali a zero

$$\int_{\Omega_e} R_h \nu dxdy = \int_{\Omega_e} \frac{\partial u_h}{\partial t}\nu dxdy + \int_{\Omega_e} \frac{\partial F_h}{\partial x}\nu dxdy + \int_{\Omega_e} \frac{\partial G_h}{\partial y}\nu dxdy - \int_{\Omega_e} Q\nu dxdy = 0 \tag{2.30}$$

Integrando quindi per parti si ottiene

$$\int_{\Omega_e} \frac{\partial u_h}{\partial t}\nu dxdy + \int_{\partial\Omega_e} (\hat{F}_h n_x + \hat{G}_h n_y)\nu ds - \int_{\Omega_e} \left(\frac{\partial \nu}{\partial x}F_h + \frac{\partial \nu}{\partial y}G_h\right)dxdy - \int_{\Omega_e} Q\nu dxdy = 0 \tag{2.31}$$

dove $n_x$ ed $n_y$ sono le componeni cartesiane del vettore normale mentre $s$ è la curva di contorno dell'elemento. La precedente equazione deve valere $\forall \nu \in V_h$, ovvero per tutte le $N_e$ funzioni che formano la base dell'elemento, per cui si impone $\nu = \Phi_i$ per $1 \leq i, j \leq N_e$

$$\int_{\Omega_e} \sum_{i=1}^{N_e} \frac{\partial \tilde{u}_i}{\partial t}\Phi_i\Phi_j dxdy + \int_{\partial\Omega_e} (\hat{F}_h n_x + \hat{G}_h n_y)\Phi_j ds + \\ - \int_{\Omega_e} \left(\frac{\partial \Phi_j}{\partial x}F_h + \frac{\partial \Phi_j}{\partial y}G_h\right)dxdy - \int_{\Omega_e} Q\Phi_j dxdy = 0 \tag{2.32}$$

ottenendo l'evoluzione degli $N_e$ coefficienti delle soluzioni delle rispettive equazioni all'interno dell'elemento $\Omega_e$. Il sistema può essere riscritto in forma compatta introducendo la matrice di massa [M]

$$[M]_{ij} = \int_{\Omega_e} \Phi_i\Phi_j dxdy \tag{2.33}$$

$$[M]\frac{\partial \tilde{u}}{\partial t} = -\int_{\partial\Omega_e} (\hat{F}_h n_x + \hat{G}_h n_y)\Phi ds + \int_{\Omega_e} \left(\frac{\partial \Phi}{\partial x}F_h + \frac{\partial \Phi}{\partial y}G_h\right)dxdy + \int_{\Omega_e} Q\Phi dxdy \tag{2.34}$$

Gli integrali sono approssimati tramite le formule di quadratura di Gauss definite sull'elemento trasformato di riferimento ed il numero di punti di quadratura è scelto per integrare esattamente i polinomi di ordine $2p$ su quest'ultimo. Per questo motivo è necessario mappare gli elementi della mesh di calcolo e trasformarli negli elementi di riferimento, per quanto riguarda gli elementi quadrilateri si è utilizzato il metodo di mappatura Serendipity (Figura 2.1), la cui trasformazione risulta

$$x = \sum_{i=1}^{n_{nod}} N_i(\xi, \eta)x_i \tag{2.35}$$

$$y = \sum_{i=1}^{n_{nod}} N_i(\xi, \eta) y_i \tag{2.36}$$

dove $n_{nod}$ è il numero di nodi che definisce la geometria dell'elemento e $N_i$ sono le funzioni di forma. La mappatura è stata implementata per elementi fino a 17 nodi:

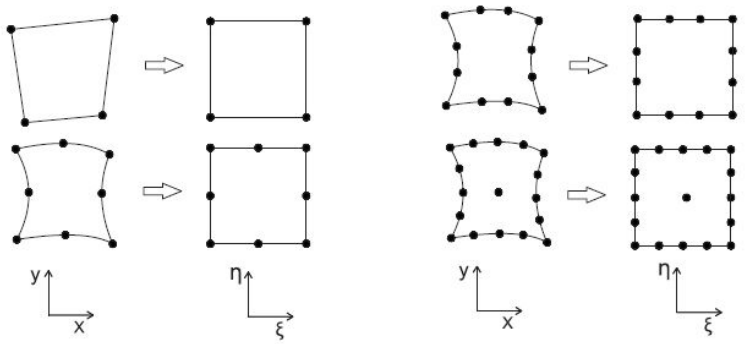


Figura 2.1. Esempio di mappatura Serendipity per elementi quadrilateri (Fonte:[12])

- Elementi lineari ($n_{nod} = 4$)

$$N_i = \frac{1}{4}(1 + \xi_i\xi)(1 + \eta_i\eta) \qquad \xi_i = \pm 1 \quad \eta_i = \pm 1 \tag{2.37}$$

- Elementi quadratici ($n_{nod} = 8$)

$$\begin{cases} N_i = \frac{1}{4}(1 + \xi_i\xi)(1 + \eta_i\eta)(\xi_i\xi + \eta_i\eta - 1) & \xi_i = \pm 1 \quad \eta_i = \pm 1 \\ N_i = \frac{1}{2}(1 - \xi^2)(1 + \eta_i\eta) & \xi_i = 0 \quad \eta_i = \pm 1 \\ N_i = \frac{1}{2}(1 - \eta^2)(1 + \xi_i\xi) & \xi_i = \pm 1 \quad \eta_i = 0 \end{cases} \tag{2.38}$$

- Elementi cubici ($n_{nod} = 12$)

$$\begin{cases} N_i = \frac{1}{32}(1 + \xi_i\xi)(1 + \eta_i\eta)[9(\xi^2 + \eta^2) - 10] & \xi_i = \pm 1 \quad \eta_i = \pm 1 \\ N_i = \frac{9}{32}(1 - \xi^2)(1 + 9\eta_i\eta) & \xi_i = \pm\frac{1}{3} \quad \eta_i = \pm 1 \\ N_i = \frac{9}{32}(1 - \eta^2)(1 + \xi_i\xi)(1 + 9\eta_i\eta) & \xi_i = \pm 1 \quad \eta_i = \pm\frac{1}{3} \end{cases} \tag{2.39}$$

- Elementi quartici ($n_{nod} = 17$)

$$\begin{cases} N_i = \frac{1}{12}(1+\xi_i\xi)(1+\eta_i\eta)[4(\eta^2-1)\xi\xi_i + 4(\eta^2-1)\eta\eta_i + 3\eta\xi\eta_i\xi_i] & \\ & \xi_i = \pm 1 \quad \eta_i = \pm 1 \\ N_i = 2(1+\xi_i\xi)(\eta^2-1)(\eta^2 - \frac{\xi_i\xi}{4}) & \xi_i = \pm 1 \quad \eta_i = 0 \\ N_i = 2(1+\eta_i\eta)(\xi^2-1)(\xi^2 - \frac{\eta_i\eta}{4}) & \xi_i = 0 \quad \eta_i = \pm 1 \\ N_i = \frac{4}{3}(1-\eta^2)(1+\xi_i\xi)(\eta^2+\eta_i\eta) & \xi_i = \pm 1 \quad \eta_i = \pm\frac{1}{2} \\ N_i = \frac{4}{3}(1-\xi^2)(1+\eta_i\eta)(\xi^2+\xi_i\xi) & \xi_i = \pm\frac{1}{2} \quad \eta_i = \pm 1 \\ N_i = (1-\xi^2)(1-\eta^2) & \xi_i = 0 \quad \eta_i = 0 \end{cases} \tag{2.40}$$

## 2.4 Discretizzazione temporale

Per completare lo schema di risoluzione delle equazioni di governo del modello fisico utilizzato è necessario scegliere ancora il metodo con cui effettuare la discretizzazione temporale di queste ultime. In ambito CFD i principali metodi utilizzati sono quello implicito e quello esplicito, con schemi di tipo centrato, all'avanti od all'indietro; tutti i metodi possono assumere diversi valori dell'ordine di accuratezza ma devono comunque sottostare ai criteri di consistenza, stabilità e convergenza che assicurano il raggiungimento di una soluzione corretta. Anche in questo caso chiaramente ogni tipologia di metodo ha vantaggi e svantaggi e la scelta deve ricadere sul più consono al tipo di applicazione voluta, per cui in questo caso sono stati utilizzati il metodo esplicito di Runge-Kutta ed il metodo implicito all'indietro di Eulero [12]. Dato che, come già detto, ogni metodo ha specifici punti di forza, è risultato molto utile poter scegliere quale dei due utilizzare a seconda della fase di risoluzione del problema, permettendo infatti il raggiungimento della soluzione in maniera corretta ed in tempistiche accettabili. Riprendendo il set di equazioni espresso in forma generale è possibile scrivere

$$[M]\frac{d\tilde{u}}{dt} = -R(\tilde{u}) \tag{2.41}$$

dove $[M]$ è la matrice di massa, $\tilde{u}$ il vettore dei gradi di libertà e $R(\tilde{u})$ il vettore dei residui. Per semplificare la notazione si considera un problema di riferimento descritto da un sistema di equazioni differenziali

$$\frac{du}{dt} = L(u) \tag{2.42}$$

Per quanto riguarda il metodo esplicito in particolare, per il primo ordine di accuratezza si è utilizzato lo schema di Eulero all'avanti

$$u^{n+1} = u^n + \Delta t L(u^n) \tag{2.43}$$

per il secondo ordine il metodo Total Variation Diminishing di Runge-Kutta (TVD-RK) a due stadi

$$\begin{cases} u^{(1)} = u^n + \Delta t L(u^n) \\ u^{n+1} = \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \frac{1}{2}\Delta t L(u^{(1)}) \end{cases} \tag{2.44}$$

per il terzo ordine sempre il metodo TVD-RK ma a tre stadi

$$\begin{cases} u^{(1)} = u^n + \Delta t L(u^n) \\ u^{(2)} = \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}) \\ u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}) \end{cases} \tag{2.45}$$

mentre per il quarto ordine il metodo Strong Stability Preservation di Runge-Kutta (SSP-RK) a cinque stadi

$$\begin{cases} u^{(1)} = u^n + 0.918\Delta t L(u^n) \\ u^{(2)} = 0.444u^n + 0.556u^{(1)} + 0.368\Delta t L(u^{(1)}) \\ u^{(3)} = 0.620u^n + 0.380u^{(2)} + 0.252\Delta t L(u^{(2)}) \\ u^{(4)} = 0.178u^n + 0.822u^{(3)} + 0.545\Delta t L(u^{(3)}) \\ u^{n+1} = 0.007u^n + 0.517u^{(2)} + 0.128u^{(3)} + 0.348u^{(4)} + \\ \qquad\qquad + 0.085\Delta t L(u^{(3)}) + 0.226\Delta t L(u^{(4)}) \end{cases} \tag{2.46}$$

Per quanto riguarda il metodo implicito invece si è utilizzato lo schema all'indietro di Eulero, ottenibile linearizzando l'equazione (2.41)

$$[M]\frac{\tilde{u}^{n+1} - \tilde{u}^n}{\Delta t} = -[R(\tilde{u}^n) + [J](\tilde{u}^{n+1} - \tilde{u}^n)] \tag{2.47}$$

dove [J] è la matrice Jacobiana, ovvero una matrice sparsa che contiene le derivate dei residui rispetto ai gradi di libertà

$$J_{ij} = \frac{\partial R_i}{\partial \tilde{u}_j} \tag{2.48}$$

Riscrivendo quindi l'equazione (2.47)

$$\left(\frac{1}{\Delta t}[M] + [J]\right)(\tilde{u}^{n+1} - \tilde{u}^n) = -R(\tilde{u}^n) \tag{2.49}$$

$$[A](\tilde{u}^{n+1} - \tilde{u}^n) = -R(\tilde{u}^n) \tag{2.50}$$

si ottiene un sistema lineare da risolvere ad ogni step temporale per aggiornare il vettore dei gradi di libertà. Per il calcolo della matrice Jacobiana il metodo più efficiente risulta essere quello di differenziare analiticamente le equazioni discretizzate. Siccome però il codice utilizzato per le simulazioni è in continuo sviluppo questo approccio richiederebbe di rigenerare il sorgente per il calcolo dello jacobiano analitico dopo ogni modifica. Nel presente lavoro si è pertanto utilizzato un approccio alternativo basato sull'approssimazione numerica della matrice tramite le differenze finite, ovvero applicando una perturbazione $\epsilon$ ad ogni grado di libertà della soluzione numerica e valutando i residui prima e dopo l'applicazione. Il livello di perturbazione deve però essere scelto sufficientemente alto per dare un buon valore di approssimazione e non troppo piccolo per evitare errori, in accordo con la seguente regola

$$\epsilon = \max(\epsilon_{min}, \epsilon_{rel} u_i) \tag{2.51}$$

con valori pari a circa $\epsilon_{min} = 10^{-10}$ ed $\epsilon_{rel} = 10^{-8}$

## 2.5 Il codice

Riassumendo le scelte precedentemente descritte, per effettuare i calcoli è stato utilizzato un modello RANS con turbolenza studiata tramite il modello di Spalart-Allmaras discretizzato nello spazio tramite il metodo agli elemeti finiti discontinui di Galerkin e nel tempo con schema esplicito di Runge-Kutta o schema implicito all'indietro di Eulero, a seconda della necessità specifica della fase del calcolo in cui si stava lavorando. Le simulazioni numeriche sono state svolte tramite un codice [12] scritto in linguaggio Fortran che permette di sfruttare la parallelizzazione dei calcoli grazie al modello di programmazione MPI con cui è stato realizzato. Il codice usa la libreria Petsc per la parallelizzazione e per la risoluzione dei sistemi lineari. Sono state svolte innumerevoli prove per la messa a punto del codice che hanno permesso di giungere a dei risultati certi e precisi in maniera efficiente e con tempistiche di calcolo accettabili. Le risorse di calcolo sono state fornite dal cluster "Hactar" del Politecnico di Torino (hpc@polito, progetto di Academic Computing del Dipartimento di Automatica e Informatica, http://www.hpc.polito.it).

# Capitolo 3

# Metodo dell'aggiunto

Il metodo dell'aggiunto è un procedimento molto utilizzato per l'ottimizzazione del processo di design e caratterizzazione di diversi aspetti in vari ambiti ingegneristici. In ambito fluidodinamico il primo ad utilizzare e studiare il metodo fu Pironneau [17], mentre il primo ad applicarlo all'ambito CFD aeronautico fu Jameson [18] che ne sviluppò l'applicazione ai flussi potenziali, alle equazioni di Eulero ed alle equazioni di Navier-Stokes. Il punto chiave del metodo, nella forma che trova più utilizzo in ambito CFD, è l'ottimizzazione intesa come massimizzazione o minimizzazione di un certo parametro che dipende da un grande numero di variabili fluidodinamiche. La forza del metodo risiede, infatti, nel fatto che grazie ad esso è possibile ridurre drasticamente il costo computazionale dell'ottimizzazione precedentemente citata, soprattutto quando il numero di variabili da cui dipende il parametro che si vuole studiare è elevato. Il parametro da analizzare viene solitamente definito tramite una funzione detta goal, che è quella da ottimizzare, di cui si monitorizza il gradiente per poter capire la direzione in cui muoversi a seconda che si voglia massimizzare o minimizzare la stessa. Il risparmio nel costo computazionale dell'operazione lo si ottiene risolvendo un sistema di equazioni aggiunto, da cui il nome del metodo, che risulta molto più semplice del sistema diretto ma, come già detto, solo nel caso di dipendenza della funzione goal da molte variabili fluidodinamiche del campo in esame [19]. Ipotizzando di voler lavorare in modo diretto, per esempio sarebbe possibile utilizzare il metodo delle differenze finite con perturbazione, ovvero applicare una perturbazione ad ogni punto della griglia e successivamente risolvere col metodo delle differenze finite, dovendo quindi però risolvere il problema un numero elevatissimo di volte, pari al numero dei punti della griglia di calcolo, ed aumentando enormemente le tempistiche di calcolo per effettuare anche solo un passo del metodo. Con il metodo dell'aggiunto, invece, il problema viene risolto una volta soltanto per tutto il dominio di calcolo con un costo computazionale consistente, a causa della complessità del sistema lineare, ma comunque accettabile, grazie all'introduzione del sistema aggiunto, permettendo infatti di effettuare molti passi ed ottenere un'ottimizzazione notevole. Questo metodo può essere utilizzato

sia in forma continua che in forma discreta, a seconda della fase del calcolo in cui viene applicato, ovvero prima o dopo la discretizzazione delle equazioni di governo, ma per questo lavoro si utilizzerà la formulazione discreta

## 3.1 L'aggiunto nell'approccio di inversione del campo

Come già accennato nell'introduzione, lo scopo del presente lavoro è la valutazione delle possibilità di miglioramento del modello di turbolenza di Spalart-Allmaras con il metodo del gradiente e l'elaborazione dei risultati, per tentare di generalizzarli e renderli applicabili a diversi casi, tramite una rete neurale con gli algoritmi del machine learning, seguendo l'approccio di Duraisamy [11]. Prendendo in esame il modello di turbolenza di SA, è necessario lavorare quindi sulla definizione della viscosità turbolenta in funzione di un termine di produzione e di uno di distruzione. In particolare, in questo caso si è scelto di intervenire sul termine di produzione moltiplicandolo per un parametro $\beta$ che ne influenzerà l'effetto sulla viscosità turbolenta e quindi su tutto il modello.

$$\frac{\partial}{\partial t}(\rho\tilde{\nu})+\frac{\partial}{\partial x_j}(\rho v_j\tilde{\nu}) = \rho(\beta P-D)+\frac{1}{\sigma}\frac{\partial}{\partial x_j}\left[\rho(\nu+\tilde{\nu})\frac{\partial\tilde{\nu}}{\partial x_j}\right]+\frac{cb_2}{\sigma}\rho\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}-\frac{1}{\sigma}\frac{\partial\rho}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k} \tag{3.1}$$

Scegliendo poi una specifica variabile fluidodinamica come parametro di controllo della correttezza del modello, si è definita una funzione goal $g$ che dà un valore numerico alla suddetta correttezza, comparando i dati di riferimento e quelli calcolati con SA della variabile fluidodinamica scelta come controllo, di cui si parlerà più nel dettaglio nel prossimo capitolo. Monitorando infatti il gradiente della funzione goal rispetto al parametro $\beta$ ($\frac{\partial g}{\partial \beta}$) è stato possibile ricavare punto per punto il valore di quest'ultimo, ma, visto l'elevato numero di variabili coinvolte, il gradiente è stato calcolato con il metodo dell'aggiunto per ridurre il costo computazione che altrimenti sarebbe molto più elevato.

Per calcolare il gradiente si scrive quindi

$$\frac{dg}{d\beta} = \frac{\partial g}{\partial \beta} + \Psi\frac{\partial R}{\partial \beta} \tag{3.2}$$

ed il sistema aggiunto risulta

$$\left[\frac{\partial R}{\partial U}\right]^T\Psi = -\left[\frac{\partial g}{\partial U}\right]^T \tag{3.3}$$

dove $\Psi$ è la variabile aggiunta, risultato del sistema aggiunto, mentre $U$ indica il set di varibili fluidodinamiche del campo di moto ed $R$ i residui del set di equazioni di governo. Tutti i termini delle due precedenti equazioni risultano essere ovviamente

matrici o vettori la cui dimensione caratteristica è $N$ o, proporzionalmente al numero di equazioni di governo che sono 5, $5N$. Il valore di $N$ è pari al numero di celle della mesh di calcolo moltiplicato per il numero di gradi di liberà per cella che, dato che si è scelto di utlizzare il metodo di discretizzazione spaziale agli elementi finiti discontinui di Galerkin del secondo ordine che prevede tre gradi di libertà modali rispetto a cui esprimere tutte le grandezze fluidodinamiche discretizzate in ogni cella, è pari a 3. Per cui risulta:

- $\frac{\partial g}{\partial \beta}$ un vettore riga di dimensione $1 * N$ che esprime la derivata della funzione goal rispetto al parametro $\beta$ ma, dato che la funzione goal non dipende direttamente da $\beta$, ha valore nullo;
- $\Psi$ un vettore colonna di dimensione $5N * 1$ che, come già spiegato, indica la variabile aggiunta risultante dalla risoluzione del sistema lieare dell'aggiunto (3.3);
- $\frac{\partial R}{\partial \beta}$ una matrice di dimensione $5N * N$ che rappresenta la derivata dei residui delle equazioni di governo rispetto al parametro $\beta$, ma l'unica equazione in cui compare $\beta$ risulta essere quella aggiuntiva di SA per cui l'espressione della derivata si riduce a

$$\frac{\partial R}{\partial \tilde{\beta}_i} = \frac{\partial R}{\partial \beta}\frac{\partial \beta}{\partial \tilde{\beta}_i} = \left(\sum_{i=1}^{n_{quad}} P(x_i)\Phi_j(x_i)\mathrm{det}_J W_{Gauss}\right)\Phi_i \tag{3.4}$$

  dove $\mathrm{det}_J$ è il determinante della matrice Jacobiana e $W_{Gauss}$ sono i pesi di Gauss, in riferimento al fatto che, discretizzando con gli elementi finiti discontinui di Galerkin, $\beta$ come tutte le variabili è espresso in funzione dei gradi di libertà modali di ogni cella tramite le funzioni di forma;
- $\frac{\partial R}{\partial U}$ una matrice quadrata di dimensione $5N * 5N$ che indica la derivata dei residui delle equazioni di governo rispetto ai gradi di libertà e coincide esattamente con la matrice Jacobiana richiesta per l'integrazione temporale con l'implicito;
- $\frac{\partial g}{\partial U}$ un vettore riga di dimensione $1 * 5N$ che indica la derivata della funzione goal rispetto ai gradi di libertà ed è stata ricavata tramite lo strumento di differenziazione automatica Tapenade, di cui si parlerà dettagliatamente nel paragrafo successivo, vista l'elevata complessità del ricavarla analiticamente.

## 3.2 La differenziazione automatica

Spesso quando si devono svolgere calcoli molto complessi che prevedono funzioni molto complesse ed articolate si incorre nella problematica del dover derivare le suddette funzioni, operazione che con i metodi analitici classici risulta molto lunga

e difficile da realizzare correttamente. Inoltre, bisogna anche tenere conto del fatto che il tutto viene complicato da quanto risulta articolato il codice che realizza la funzione da integrare e dalla successiva scrittura del codice che realizza la derivata. Per ovviare a questo problema sono state sviluppate le tecniche di differenziazione automatica che permettono, dato un codice sorgente che realizza la funzione da derivare, di ottenere immediatamente il codice che ne relizza la derivata, riducendo quindi di molto la complessità dell'operazione. Queste tecniche non lavorano sulla funzione, come si farebbe col metodo analitico, ma lavorano direttamente sul codice fornito come sorgente con diverse metodologie che permettono di effettuare la differenziazione in maniera analitica ed efficiente. Per questo lavoro si è scelto di utilizzare il tool di differenziazione automatica realizzato dall'INRIA (Institut national de recherche en informatique et en automatique) chiamato Tapenade [20], molto efficace e semplice da utilizzare essendo liberamente reperibile in rete. Questo tool supporta codici scritti in linguaggio C o Fortran, è specializzato nella sola derivazione primaria e permette di utilizzare diversi metodi di differenziazione tra cui quello tangente da noi scelto.

Prendendo come esempio una semplicissima funzione in due variabili

$$c = 2a + b^3 \tag{3.5}$$

è sufficiente scrivere un codice in Fortran che la implementi per poter utilizzare Tapenade per derivarla. Il codice risulta quindi

```
SUBROUTINE TEST(a, b, c)
IMPLICIT NONE
REAL*8 :: a, b, c
c = 2.d0*a + b**3
END SUBROUTINE TEST
```

che dopo l'elaborazione di Tapenade diventa

```
SUBROUTINE TEST_D(a, ad, b, bd, c, cd)
IMPLICIT NONE
REAL*8 :: a, b, c
REAL*8 :: ad, bd, cd
cd = 2.d0*ad + 3*b**2*bd
c = 2.d0*a + b**3
END SUBROUTINE TEST_D
```

Osservando il nuovo codice si deduce che la derivata calcolata da tapenade è espressa dalla variabile *cd* e che i termini *ad* e *bd* servono a permettere all'utente di scegliere quale derivata parziale utilizzare, infatti se si volesse scegliere la derivata di $c$ rispetto ad $a$ sarebbe sufficiente porre *ad* uguale a 1 e *bd* uguale a 0. Derivando

analiticamente è facilmente verificabile la correttezza del risultato fornito dal tool, si ottiene infatti

$$\frac{\partial c}{\partial a} = 2 \qquad \frac{\partial c}{\partial b} = 3b^2 \tag{3.6}$$

La stessa logica di funzionamento vale ovviamente per funzioni più complesse con un numero elevato di variabili implemetate in codici molto più articolati, per le quali questo strumento è stato pensato.

## 3.3 L'ottimizzazione

Avendo descritto e ricavato il metodo per ottenere tutte le componenti necessarie al calcolo del gradiente della funzione goal, è possibile ora effettuare l'ottimizzazione vera e propria. Come già più volte spiegato, l'obbiettivo di questa ottimizzazione è la minimizzazione della differenza tra i risultati ottenibili con il modello di turbolenza e quelli di riferimento, quindi la minimizzazione della funzione goal che esprime la suddetta differenza. Volendo lavorare sul termine di produzione di SA si riprende l'equazione (2.19) in cui compare e si inserisce la variabile $\beta$ con cui verrà modificato (come visibile nell'equazione (3.1)), il cui valore è inizialmente 1 e viene poi aggiornato ad ogni step temporale col gradiente della funzione goal. Volendo ottenere una minimizzazzione della funzione goal ci si muoverà in direzione opposta al gradiente di quest'ultima, che indica infatti la direzione in cui la funzione cresce, per cui risulterà

$$\beta_{k+1} = \beta_k - \delta \frac{dg}{d\beta} \tag{3.7}$$

dove $k$ indica il generico passo dell'algoritmo di ottimizzazione, mentre $\delta$ indica l'entità dello spostamento da imporre alla fuzione che va scelto arbitrariamente per tentativi, fino ad ottenere un effetto accettabile. Ricapitolando, quindi, la logica di funzionameto della simulazione con il metodo dell'aggiunto implementato: inizialmente partiranno i calcoli per la risoluzione del sistema di equazioni RANS e di SA che proseguiranno fino al raggiungimento della convergenza ad una soluzione stabile, di cui verrà quindi calcolato il gradiente della funzione goal, con il metodo dell'aggiunto e la risoluzione dei relativi sistemi, che verrà utilizzato per aggiornare il campo di $\beta$ che modificherà a sua volta il valore del termine di produzione di SA. La soluzione stazionaria che si era ottenuta subirà quindi una perturbazione dovuta al differente termine di produzione e, quindi, non sarà più a convergenza, per cui ripartiranno i calcoli per la risoluzione delle equazioni RANS e di SA, facendo ripartire la sequenza da capo ed ottenendo ogni volta che si giungerà a convergenza una nuova soluzione che dovrebbe avvicinarsi sempre più a quella di riferimento (sperimentale o LES/DNS).

# Capitolo 4

# Il test case

L'ultima fase di preparazione di tutto il necessario per lo svolgimento dei calcoli consiste nella scelta del caso di studio reale a cui applicare il modello descritto dalle scelte spiegate nei precedenti capitoli. Normalmente si inizia scegliendo test case molto semplici, con geometrie poco complesse e condizioni semplici da simulare, per poter verificare il corretto funzionamento del modello di calcolo generato e, successivamente, si passa a casi più complessi per testarne le reali capacità e l'effettiva applicabilità. L'ideale è verificare il funzionamento e la correttezza dei risultati scegliendo test case che riproducono modelli realmente sperimentati, di cui si conoscono quindi tutti i dati ed i risultati precisi. Un'altra possibilità, invece, è quella di scegliere casi di studio su cui è possibile applicare modelli di calcolo già testati e considerabili come fonte di risultati corretti per confrontarne il comportamento con il modello che si sta mettendo a punto. Le scelte da effettuare per definire il test case sono diverse, si parte prima di tutto dalla geometria che può essere bidimensionale o tridimensionale, poi, in base alle dimesioni ed alla complessità di quest'ultima, si deve scegliere il tipo di mesh di calcolo da applicare ed il numero di celle che la compongono, per finire è necessario definire le condizioni del flusso in ingresso, in uscita ed al contorno. Dato che il modello studiato per questa tesi è nella fase iniziale di sviluppo si è scelto un test case semplice composto da un condotto curvo bidimensionale di cui si conoscono i risultati di una simulazione LES, utilizzabili come dati reali di confronto, su cui è gia stato applicato un'approccio simile a quello presentato in questo lavoro da Duraisamy e Singh [10].

## 4.1 La geometria

Come anticipato precedentemente, il caso di studio scelto per l'applicazione del modello di calcolo in esame è un semplice condotto bidimensionale che presenta una curvatura di 90° circa a metà della sua lunghezza, in cui il flusso d'aria entra da una parte ed esce dall'altra. Più precisamente il tubo è composto da due pareti solide

parallele che collegano l'ingresso e l'uscita (Figura 4.1) imponendo al flusso d'aria di entrare orizzontalmente ed uscire verticalmente. La curvatura genererà quindi un gradiente di pressone nella direzione del flusso ed altre deformazioni dello stesso che permetteranno di riprodurre le giuste condizioni di turbolenza anche a basse velocità. La parete solida superiore viene considerata idealmente inviscida per cui il flusso d'aria vi scorrerà sopra senza attrito, mentre la parete solida inferiore viene considerata adiabatica ma soggetta alla viscosità per cui il flusso in contatto con essa risulterà fermo e la sua velocità crescerà lentamente allontanandosi dalla parete, generando quindi lo strato limite. In riferimento alla figura 4.2: le dimensioni

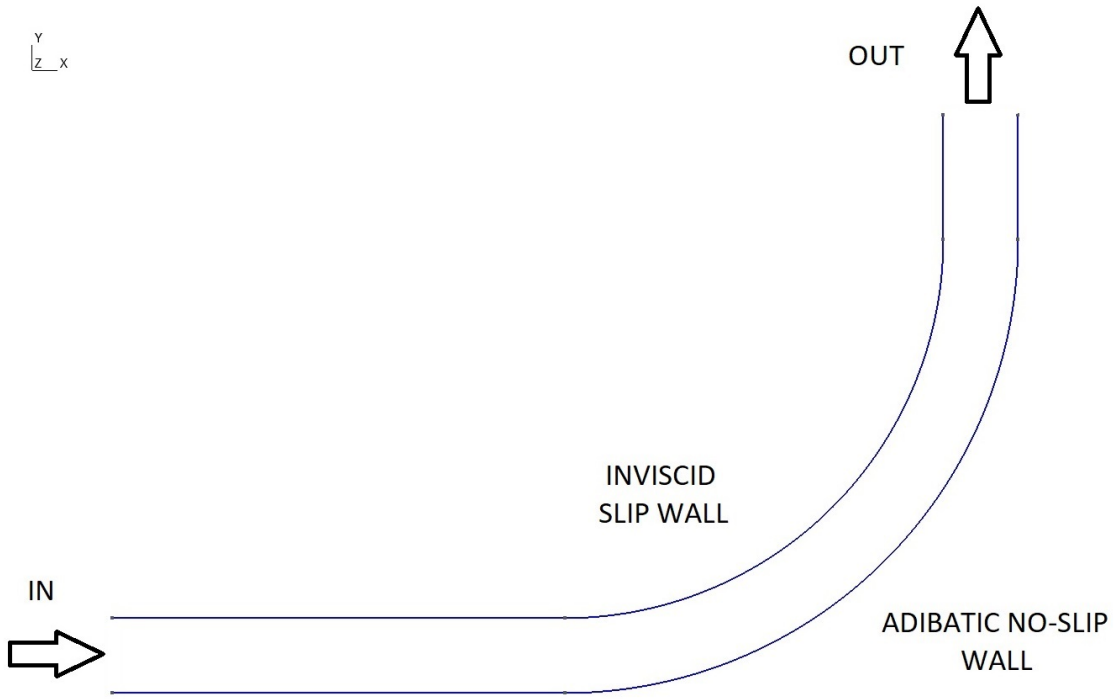


Figura 4.1. Geometria del condotto

dell'ingresso rapresentato dal segmento $\widehat{12}$ e dell'uscita rappresentata dal segmento $\widehat{56}$ sono, ovviamente, le medesime di valore pari a 16.5 cm; le dimesioni del primo tratto orizzontale delle due pareti, rappresentate dai segmenti $\widehat{23}$ e $\widehat{18}$, sono di 1 m; dalla fine del primo tratto rettilineo parte la sezione curva del condotto che compie un angolo di 90° con centro dei raggi di curvatura nel punto 9 e raggio esterno $R$ pari a 1 m e raggio interno $r$ pari a 83.5 cm, che formano i due archi di cerchio che rappresentano le due pareti indicate dai segmenti $\widehat{34}$ e $\widehat{87}$; per finire, dal punto di conclusione del tratto curvo, parte il secondo tratto rettilineo del condotto, le cui pareti sono rappresentate dai segmenti $\widehat{45}$ e $\widehat{76}$, di lunghezza 27.5 cm.

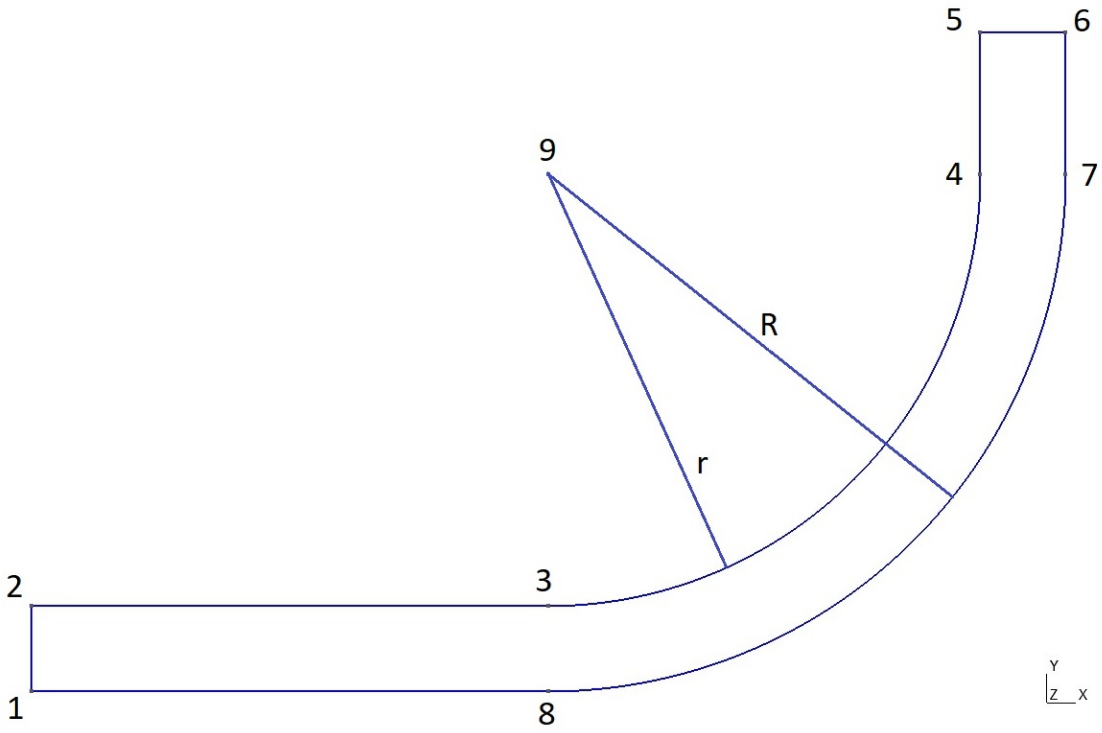


Figura 4.2. Dimensioni del condotto

## 4.2 La mesh

Una volta definita la geometria e le dimensioni del tubo è necessario scegliere come discrettizzarlo fisicamente, quindi come suddividerne lo spazio per mezzo della griglia di calcolo che identifica le celle in cui verranno discretizzate e risolte le equazioni. La scelta di una buona mesh è fondamentale per il raggiungimento del risultato in quanto influenza fortemente la correttezza di quest'ultimo ed il tempo di convergenza, oltre che il tempo e la quantità di risorse necesserie al calcolo. La qualità di una mesh si valuta dal tipo di elementi scelti, dalla loro dimensione, dalla loro densità, dal rapporto delle loro dimensioni e soprattuto dalla capacità della griglia di adattarsi alla geometria del problema. In campo bidimensionale solitamente le celle sono triangoli o quadrilateri, mentre in campo tridimensionale sono tetraedri, piramidi, prismi o poliedri. I due principali tipi di griglia utilizzati in ambito CFD sono quello strutturato, in cui gli elementi sono identificati da un'unica terna di indici e risultano ordinati nello spazio rendendo però questo tipo adatto solo a geometrie semplici, e quello non struttrato, in cui gli elementi sono organizzati in modo arbiratio per cui si adattano molto meglio a geometrie complesse ma richiedono un tempo di calcolo maggiore. Esistono ovviamente anche casi di utilizzo ibrido

dei due tipi con divisone a blocchi dello spazio per migliorare la qualità generale della mesh adattandola meglio ad ogni zona del dominio, in particolare spesso si fa variare molto la densità degli elementi in modo da addensarli nei punti più interessanti dove è richiesta una maggiore discretizzazione per l'ottenimento di risultati migliori. In generale più la mesh è fitta più i risultati saranno precisi ma anche il tempo di calcolo salirà notevolmente, è necessario trovare quindi il giusto compromesso. Dato che, come già detto, la geometria in esame, descritta nel precedente paragrafo, è molto semplice si è scelto di utilizzare una griglia strutturata con elementi rettangolari (Figura 4.3), considerando la bidimensionalità del condotto, che permette di risparmiare tempo e risorse di calcolo mantenendo una qualità ottima. In particolare la mesh utilizzata è formata da 60 celle in direzione perpendicolare al

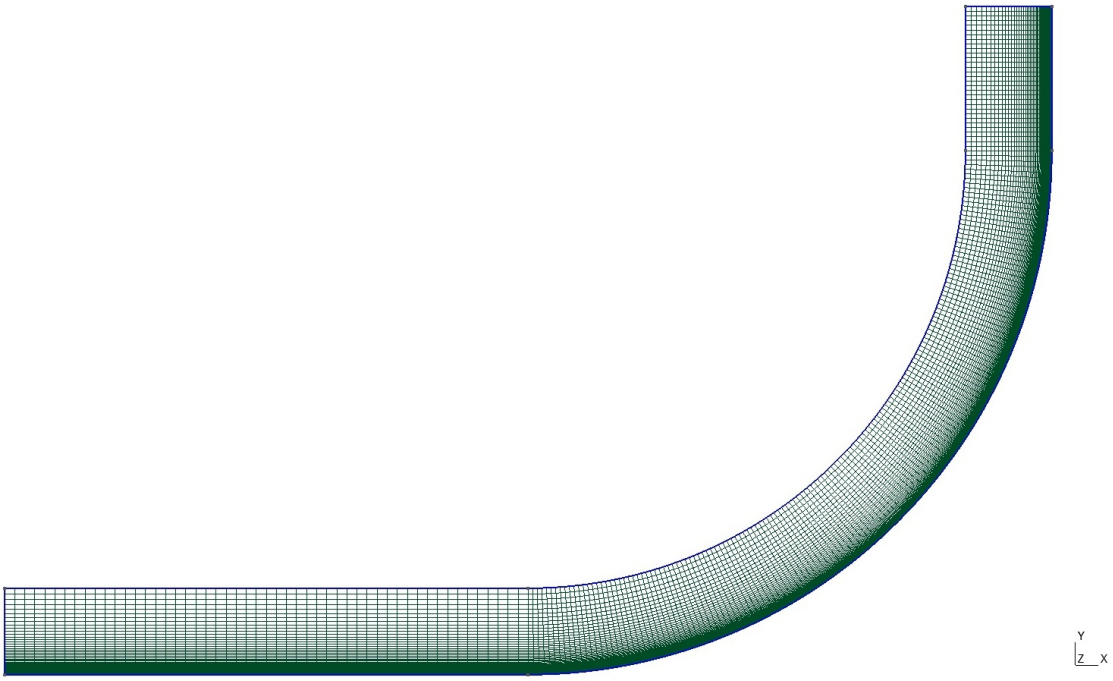


Figura 4.3. Mesh del condotto

flusso e da 236 celle in direzione parallela, queste ultime divise proporzionalmente alla lunghezza sui tre diversi tratti delle pareti slide (53 sul primo tratto rettilineo orizzontale, 151 sul tratto curvo e 32 sul secondo tratto rettilineo verticale). Per migliorare ulteriormente la qualità della griglia lungo la parete inferiore su cui si sviluppa lo strato limite si è, in oltre, imposto un fattore di progressione pari ad 1.07 in direzione perpendicolare al flusso, partendo dalla parete superiore e andando verso quella inferiore dove, infatti, la densità degli elementi aumenta notevolmente permettendo la risoluzione delle equazioni nello strato limite.

## 4.3 Le condizioni del flusso

Uno degli ultimi passi da svolgere prima di poter iniziare i calcoli è quello di determinare le condizioni iniziali e le condizioni al contorno da imporre al problema. Questa scelta non risulta sempre facile, complicandosi proporzionalemte alla complessità del caso di studio, ma deve essere fatta con grande attenzione in quanto influenza fortemente la correttezza del risultato e le tempistiche necessarie al raggiungimento di quest'ultimo. Esistono diversi metodi e strategie per la scelta dei due tipi di condizioni, a seconda del modello fisico e fluidodinamico che si utilizza, per l'ottenimento di un buon modello matematico che descriva correttamente la fluidodinamica del problema e la sua evoluzione. Le condizioni iniziali non influenzano direttamente la soluzione, in quanto determinano solo l'inizializzazione di alcune variabili del flusso che verranno poi aggiornate con l'avanzare dei calcoli, ma incidono fortemente sulla convergenza della soluzione e sulle relative tempistiche, risultando quindi determinanti soprattuto per problemi molto complessi in cui il raggiungimento della convergenza è molto difficile e le tempestiche di calcolo sono comunque molto elevate. Le condizioni al contorno, invece, influenzano direttamente la soluzione finale e la sua correttezza, in quanto riguardano le condizoni di ingresso e di uscita del flusso, le sue caratteristiche, eventuali periodicità ed interazioni con superfici solide, infatti se mal poste possono portare a gravi errori che invalidano tutto il calcolo.

Per il test case scelto si sono poste le seguenti condizioni iniziali:

- $p_{in}^0 = 1$
- $T_{in}^0 = 1$
- $M_{in} = 0.2$

con la velocità all'interno del condotto inizializzata uniformemente ad un valore pari al mach di ingresso e fatta curvare parallelamente al tubo, con variazione lineare dell'angolo di inclinazione locale nella zona di curvatura da 0 a 90° come visibile in figura 4.4, per limitare la generazione e la riflessione di eventuali onde d'urto o disturbi che complicherebbero notevolmente il campo di moto prolungando le tempistiche di calcolo. Il codice utilizzato per i calcoli, inoltre, lavora con variabili adimensionalizzate per cui la pressione e la temperatura totali di monte sono utlizzate come grandezze di riferimento. I parametri caratteristici della fluidodinamica sono invece stati posti pari a:

- $\gamma = 1.4$
- $Pr = 0.72$
- $Pr_t = 0.9$

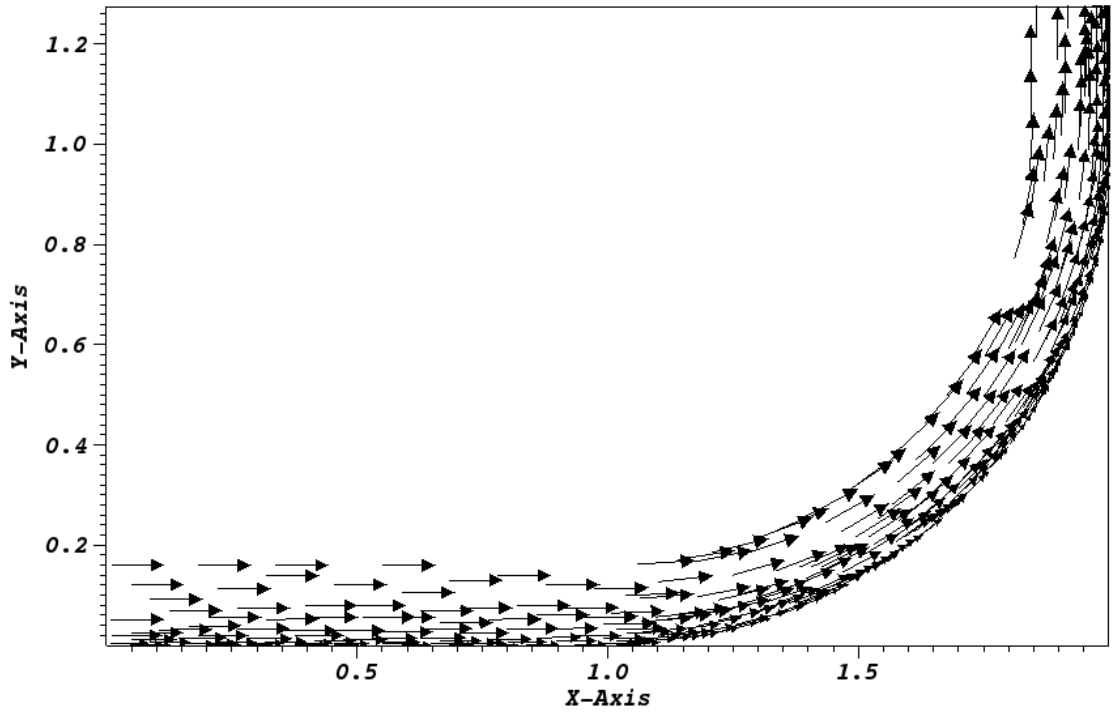


Figura 4.4. Velocità iniziale nel condotto

- $\rho\tilde{\nu} = 3$

dove $\rho\tilde{\nu}$ è la variabile conservativa dell'equazione (2.19) di SA ed indica il livello iniziale di turbolenza. Il numero di Reynolds, invece, è stato imposto andando a cercare il valore che fornisce lo spessore di strato limite riportato da Duraisamy e Singh, pari cioè ad un quarto dell'altezza del canale. Per le condizioni al contorno si è invece posto:

- $p_{out} = f(M_{isentr,out} = 0.2)$
- profilo $\vec{v}$ iniziale
- profilo $\mu_{sa}$ iniziale

dove la prima condizione indica il fatto che nel codice è possibile imporre il valore della pressione statica di uscita tramite il valore del mach isentropico di uscita, mentre le successive due si riferiscono al fatto che è stato imposto un profilo di velocità e viscosita dinamica turbolenta inziale. Il suddetto profilo è stato fornito direttamente da Singh, che insieme a Duraisamy ha sviluppato l'approccio seguito per questa tesi applicandolo allo stesso test case e ad altri [10, 11], ed è stato poi riscalato ed adattato per essere utilizzato con il codice disponibile per questo lavoro.

Il profilo conteneva le varibili $\rho, u, v, p, \mu_{sa}$ riferite alle $y$ dei centri delle facce delle celle della mesh che formano la superficie d'ingresso del canale, da cui sono state ricavate le variabili $p^o, \mu_{sa}$, sempre in funzione delle stesse $y$, utili per implementare nel codice i profili iniziali voluti. Dalla componente della velocità $u$ si è calcolata la pressione totale passando per il mach ed utilizzando la pressione statica fornita, ricordando che si sta lavorando con grandezze adimensionalizzate:

$$M = \frac{u}{a} = \frac{u}{\sqrt{\gamma T}} = \frac{u}{\sqrt{\gamma \frac{T^0}{1+\frac{\gamma-1}{2}M^2}}} \tag{4.1}$$

$$M^2 \gamma \frac{T^0}{1+\frac{\gamma-1}{2}M^2} = u^2 \tag{4.2}$$

$$\frac{\gamma-1}{2}\gamma T^0 M^4 + \left(\gamma T^0 - \frac{\gamma-1}{2}u^2\right)M^2 - u^2 = 0 \tag{4.3}$$

$$M = \sqrt{\frac{-\left(\gamma T^0 - \frac{\gamma-1}{2}u^2\right) + \sqrt{\left(\gamma T^0 - \frac{\gamma-1}{2}u^2\right)^2 + 4\frac{\gamma-1}{2}\gamma T^0 u^2}}{2\frac{\gamma-1}{2}\gamma T^0}} \tag{4.4}$$

$$p^0 = p\left(1 + \frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{4.5}$$

Il risultato della pressione totale è stato poi riscalato in modo da avere valore massimo pari ad 1, conformemente alle adimensionalizzazioni presenti nel codice, mentre per la viscosità dinamica è stato sufficiente utilizzare i dati già noti. Il profilo fornito da Singh è stato ottenuto da una simulazione su una lamina piana priva di gradiente di pressione e tramite l'ausilio del metodo di ricircolo, che suppone vi sia un collegamento virtuale tra ingresso ed uscita che permette di collegare le grandezze in ingresso a quelle in uscita, in particolare per quanto riguarda lo strato limite e la turbolenza. L'imposizione del profilo in ingresso ha permesso, infatti, di assicurare il completo sviluppo dello strato limite e della turbolenza fin dall'inizio del condotto. Inoltre, sempre per quanto riguarda le condizioni al contorno, come già precedentemente detto, sulla parete superiore del canale si è posta la condizione di tangenza, mentre sulla parete inferiore le condizioni di adiabaticità ed assenza di scorrimenti per effetto della viscosità.

## 4.4 La funzione goal

Uno dei punti fondamentali della fase di preparazione preliminare ai calcoli è la scelta della variabile di controllo da utilizzare per verificare la correttezza del moello in esame e gli eventuali miglioramenti che si riescono a raggiungere. Le variabili

fluidodinamiche disponibili sono innumerevoli ma è necessario sceglerne una facilmente rilevabile, di cui si conosce l'andamento reale dai dati di riferimento che si hanno a disposizione e, preferibilmente, che sia influenzata dal fenomeno che si sta studiando, in modo che i risultati varino effettivamente al variare della correttezza del modello utilizzato. Per questa tesi si è scelto di monitorare la correttezza del modello di SA tramite il coefficiente d'attrito $c_f$ che si genera sulla parete inferiore del tubo reativo al test case scelto, variabile fortemente influenzata dallo strato limite che si forma su quella parete e dalla relativa turbolenza, quindi fortemente influenzato dalla bontà del modello. Questa variabile risulta facile da calcolare

$$c_f = \frac{\tau_{wall}}{\frac{1}{2}\rho\vec{v}^2} \qquad \tau_{wall} = [\vec{\tau}_{ij}]\vec{n} \tag{4.6}$$

e, inoltre, si hanno a disposizione i relativi risultati di una simulazione LES sullo stesso condotto scelto. Come già spiegato, le simulazioni LES risultano più costose computazionalmente ed applicabili ai soli casi semplici ma presentano risultati più corretti, data la minor soggezione ad approssimazioni, che si avvicinano molto ai dati sperimentali. Per questo motivo possiamo prendere i relativi dati del $c_f$, visibili nella tabella 4.1 e nel grafico 4.5, ed utilizzarli per effettuare una comparazione coi risultati del modello SA, trattandoli come fossero dati sperimentali a cui puntare con i miglioramenti.
Avendo scelto di tentare di migliorare il modello di SA con il metodo del gradiente, come già più volte detto, non resta che definire la funzione goal con cui monitorare la differenza tra il $c_f$ risultante da SA e il $c_{f,exp}$ precedentemente definito, di cui verrà successivamente calcolato il gradiente, per cui risulta

$$g = \int_l (c_f - c_{f,exp})^2 dl \tag{4.7}$$

dove con $l$ si intende la lunghezza della parete solida inferiore del condotto.

Avendo definito la funzione goal è possibile analizzare più nel dettaglio anche la sua derivata $\frac{\partial g}{\partial U}$, necessaria alla risoluzione del sistema dell'aggiunto, che risulta

$$\frac{\partial g}{\partial U} = 2\int_l (c_f - c_{f,exp})\frac{\partial c_f}{\partial U} dl \tag{4.8}$$

Resta quindi da calcolare $\frac{\partial c_f}{\partial U}$, ovvero la derivata del coefficiente d'attrito in funzione dei gradi di libertà che, come già spiegato, è stata calcolata con lo strumento di differenziazione automatica Tapenade. In particolare si è realizzata una subroutine fortran che calcolasse il $c_f$ in funzione di tutti i gradi di libertà, che risultano essere 15 in quanto vi sono 5 equazioni di governo per 3 grandi di libertà per elemento della mesh, che è stata poi elaborata da Tapendade che ha restituito la subroutine che calcola la derivata voluta.

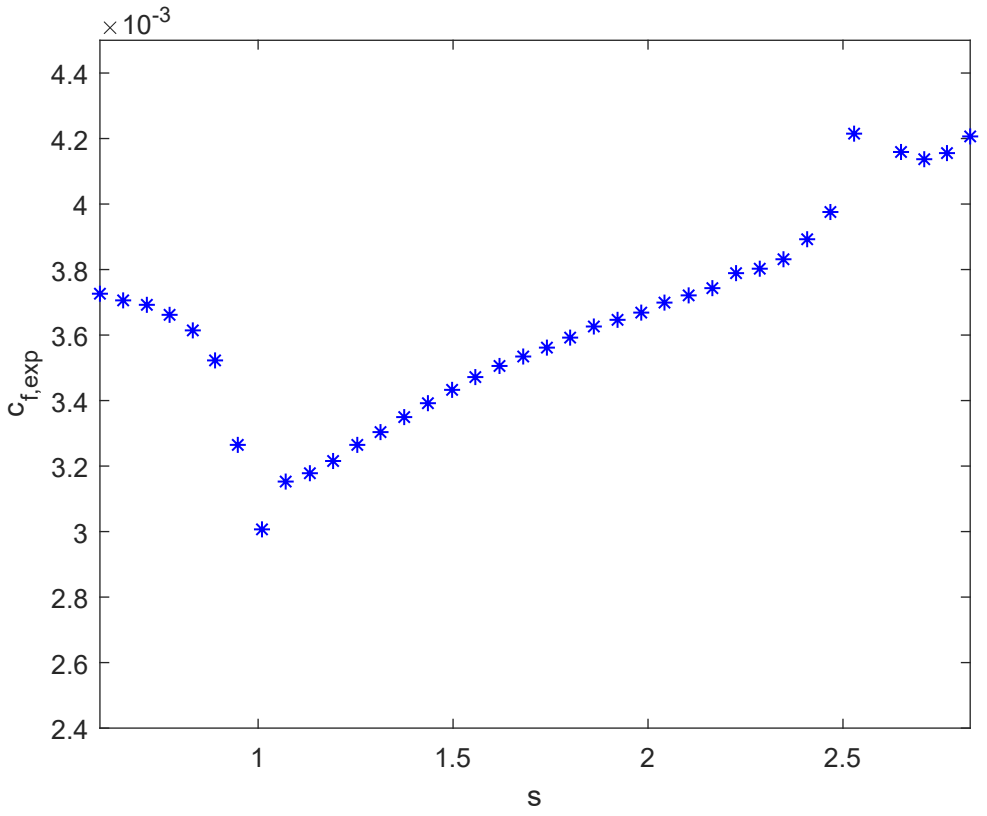


Figura 4.5. Andamento $c_f$ risultante dalla LES in funzione della coordinata curvilinea $s$

Tabella 4.1. Dati $c_f$ risultante dalla LES

| $s$ | $c_{f,exp}$ |
|---|---|
| 0.5942928039702234 | 0.003726244343891402 |
| 0.6538461538461536 | 0.003705882352941176 |
| 0.7146401985111662 | 0.003692307692307692 |
| 0.7729528535980148 | 0.003661764705882352 |
| 0.8325062034739454 | 0.0036142533936651573 |
| 0.889578163771712 | 0.0035226244343891404 |
| 0.9478908188585613 | 0.0032647058823529413 |
| 1.009925558312655 | 0.0030067873303167418 |
| 1.070719602977667 | 0.0031527149321266966 |
| 1.1327543424317617 | 0.003178167420814479 |
| 1.1923076923076918 | 0.0032154977375565604 |
| 1.2543424317617864 | 0.0032647058823529404 |
| 1.3138957816377166 | 0.003303733031674207 |
| 1.3746898263027292 | 0.003349547511312217 |
| 1.4354838709677418 | 0.0033919683257918544 |
| 1.4975186104218359 | 0.0034326923076923067 |
| 1.5570719602977665 | 0.0034717194570135744 |
| 1.6191066997518608 | 0.003505656108597284 |
| 1.6799007444168732 | 0.0035345022624434378 |
| 1.7406947890818856 | 0.0035616515837104067 |
| 1.800248138957816 | 0.003592194570135746 |
| 1.8610421836228286 | 0.0036261312217194556 |
| 1.9218362282878412 | 0.003646493212669683 |
| 1.9826302729528535 | 0.0036685520361990943 |
| 2.042183622828784 | 0.003699095022624434 |
| 2.1042183622828787 | 0.0037211538461538454 |
| 2.16501240694789 | 0.0037432126696832573 |
| 2.225806451612903 | 0.003789027149321266 |
| 2.286600496277915 | 0.0038026018099547505 |
| 2.347394540942928 | 0.0038314479638009045 |
| 2.40818858560794 | 0.003892533936651583 |
| 2.4677419354838706 | 0.003975678733031673 |
| 2.5285359801488827 | 0.004214932126696832 |
| 2.5888833746898255 | 0.004158936651583709 |
| 2.648436724565756 | 0.004136877828054298 |
| 2.708249379652605 | 0.004155542986425339 |
| 2.7663027295285355 | 0.004206447963800904 |

# Capitolo 5

# Risultati e conclusioni

## 5.1 Risultati del modello base di SA

L'ultima fase dello studio si incentra sull'analisi dei risultati raggiunti e dei progressi fatti, in relazione agli obbiettivi che si erano posti, per poter quindi trarre delle conclusioni e valutare eventuali sviluppi futuri ed applicazioni del lavoro svolto. Dopo aver scelto e messo a punto il modello fisico e fluidodinamico, descritto nel dettaglio la metodologia di lavoro, scelto il caso di studio da utilizzare e scelto le condizioni di lavoro, quelle al contorno e quelle iniziali, è possibile finalmente svolgere i calcoli lanciando le simulazioni numeriche tramite il codice generato. Le prove e le modifiche fatte sono state moltissime, tutti gli aspetti e tutte le scelte effettuate sono stati analizzati nel dettaglio per valutarne la correttezza e l'efficienza e, ovviamente, anche i risultati sono stati ottenuti prima in forma grezza e poi perfezionati migliorando il codice e la scelta dei vari parametri fluidodinamici. In un primo momento l'attenzione è stata volta al raggiungimento di una soluzione stabile e corretta del modello di Spalart-Allmaras base, cioè senza modifiche o migliorie. Ovviamente risulta logico cercare di far funzionare un modello prima di poter pensare di migliorarlo, per questo ci si è dedicati alla messa a punto dei paramentri fluidodinamici, delle condizioni iniziali e dell'inizializzazione delle variabili, delle condizioni al contorno e della mesh di calcolo che discretizza il test case scelto. Sistemati questi fattori, si è cercato anche di ottimizzare i calcoli e rendere le simulazioni di durata ragionevole sfruttando in modo efficiente il codice e le sue potenzialita. Una prima soluzione stabile, da utilizzare come verifica e come punto di partenza per il miglioramento del modello, è stata raggiunta dopo circa 144000 iterazioni nel tempo, per una durata dei calcoli di qualche ora con il codice parallelizzato su 20 core. I risultati ottenuti sono visibili nei grafici delle immagini 5.1, 5.2 e 5.3 che rappresentano rispettivamente la distribuzione del Mach, della pressione e della densità all'interno del condotto. Osservando il grafico del Mach si nota che sulla parete inferiore è presente lo strato limite ben sviluppato, come previsto avendo imposto la condizione di parete viscosa per cui la velocità del flusso

su di essa è nulla e poi aumenta gradualmente fino a raggiungere il valore del resto del flusso, mentre sulla parete superiore, posta inviscida, non vi sono gradienti di velocità che infatti resta costante; si nota in oltre una leggera accelerazione del flusso vicino alla parete superiore nella zona di curvatura dovuto proprio alla curvatura stessa. Osservando il secondo ed il terzo grafico si nota invece che sia la pressione che la densità non variano di molto lungo il tubo, solo nella zona della curvatura sulla parete superiore vi è una leggera diminuzione della pressione e della densità e su quella inferiore un leggero aumento di entrambe le grandezze. In figura

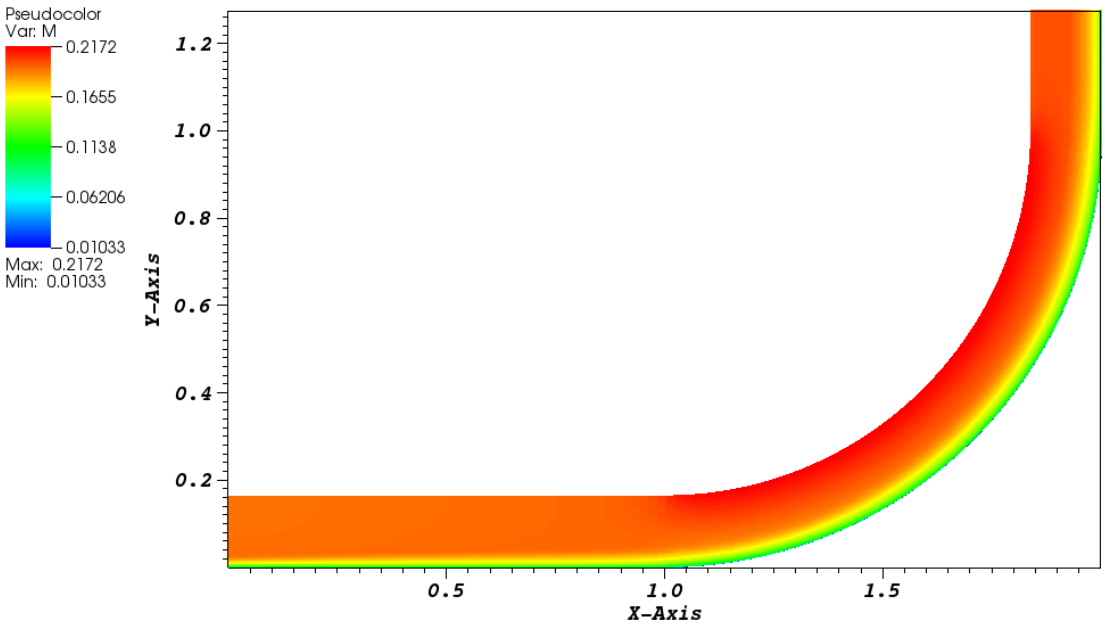


Figura 5.1. Distribuzione del Mach nel condotto

5.4 invece è visibile l'andamento del coefficiente d'attrito sulla parete inferiore del condotto in funzione della coordinata curvilinea che, come già detto, è la variabile scelta per il controllo della bontà del modello, quindi quella più importante per il raggiungimento dello scopo prefissato. Osservando il grafico si nota che inizialmente il $c_f$ tende a $-\infty$ a causa degli effetti dell'ingresso sul flusso ma successivamente si stabilizza e dimiusce fino al punto in cui finisce il primo tratto rettilineo, dove si ha il minimo globale, per poi aumentare fino alla fine, tranne che per una piccola oscillazione corrispondente alla fine della curvatura in cui diminuisce e poi aumenta nuovamente, subendo gli effetti dell'uscita. Per questa prima fase si è utilizzato il modello di Spalart-Allmaras privo di modifiche di cui sono già note correttezza ed applicabilità al caso scelto, osservando infatti il grafico in figura 5.5 è possibile confrontare il $c_f$ ricavato con quello ricavato da Duraisamy e Singh con lo stesso modello e per lo stesso test case: l'andamento risulta comparabile. Sempre nello

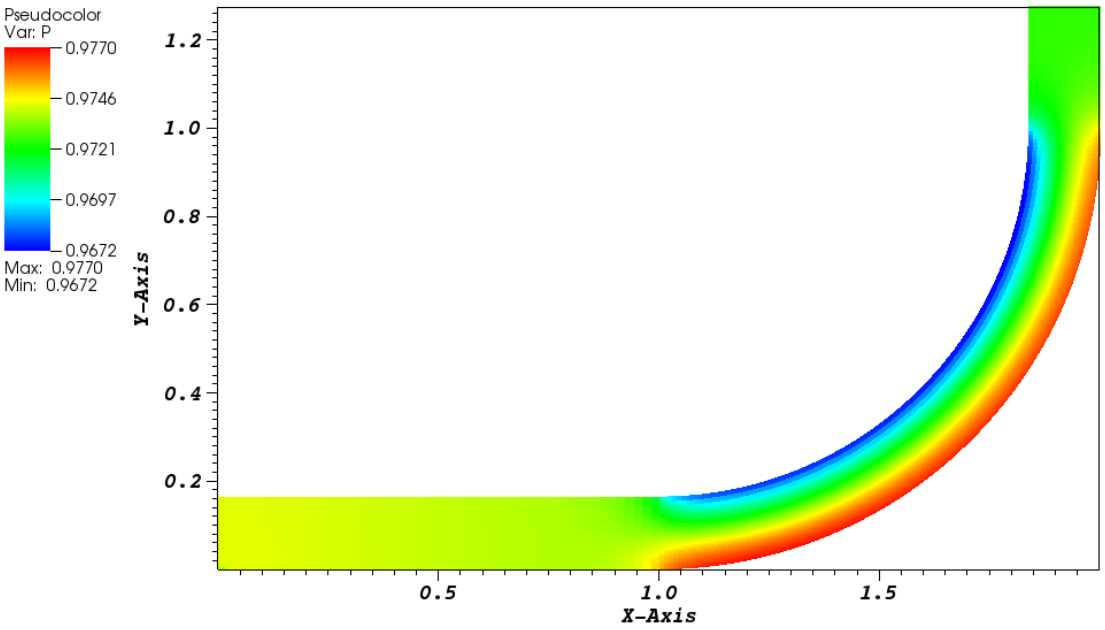


Figura 5.2. Distribuzione di pressione nel condotto

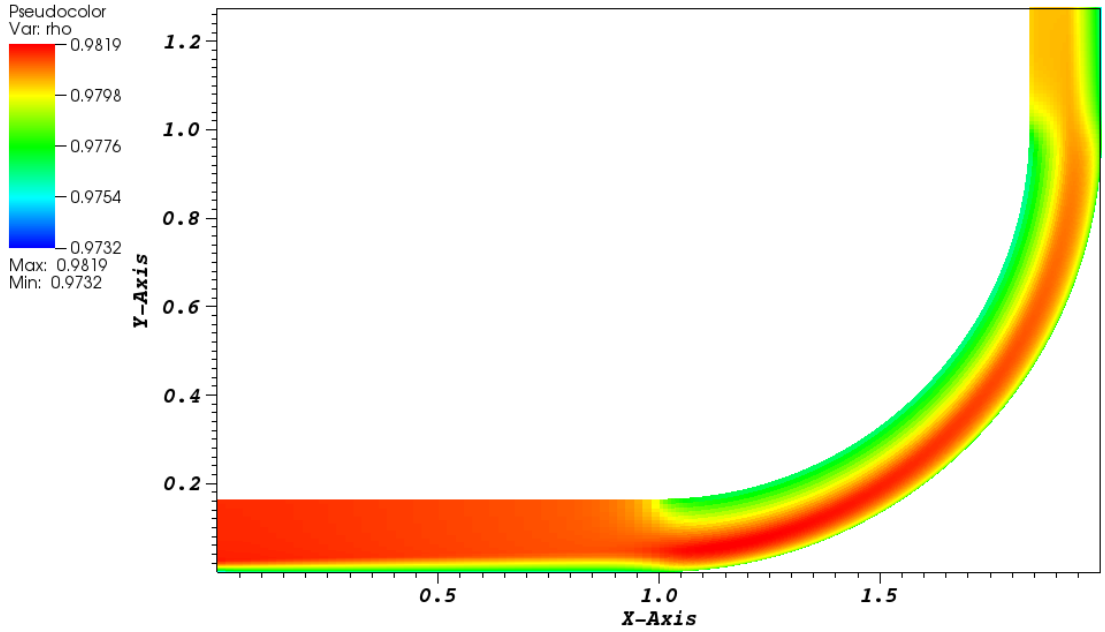


Figura 5.3. Distribuzione di densità nel condotto

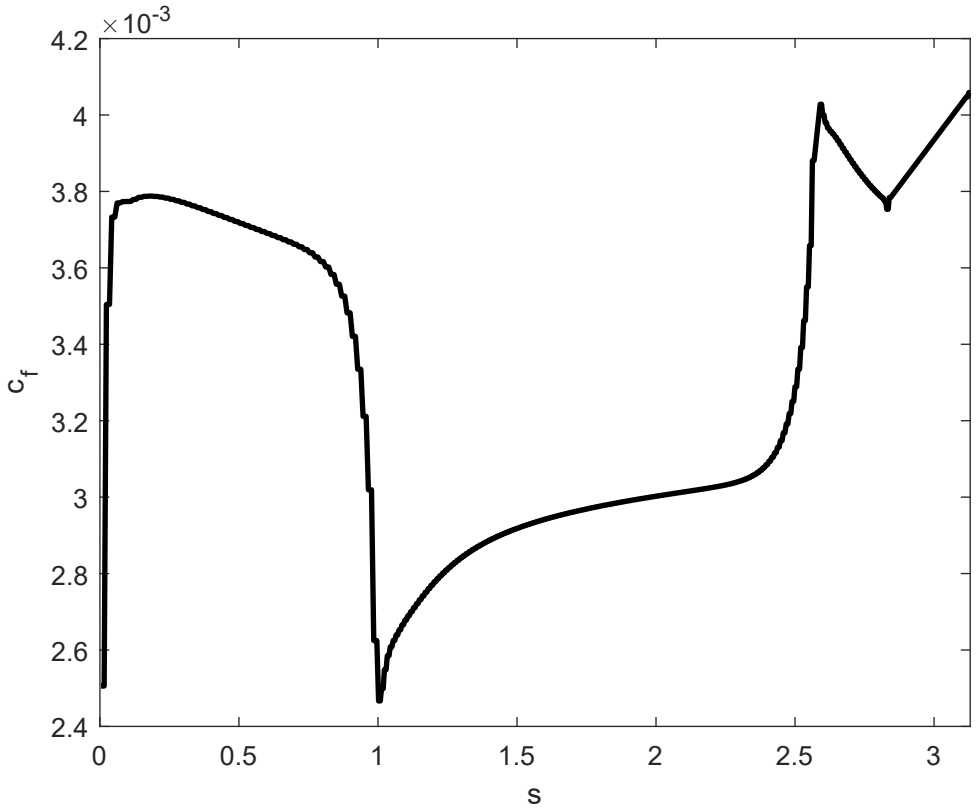


Figura 5.4. Andamento del coefficiente d'attrito in funzione della coordinata curvilinea

stesso grafico è possibile anche confrontare entrambi i risultati con il $c_f$ ottenuto con la simulazione LES ed osservare che l'andamento è simile ma, tranne che per la prima zona, vi è una differenza non trascurabile nei valori ottenuti. Il dominio del grafico risulta ridotto rispetto alla totale lunghezza del condotto in quanto per le zone iniziali e finali, che risultano comunque di più difficile analisi a causa dei disturbi dovuti agli effetti di bordo di ingresso ed uscita, non si disponeva dei dati LES, ma in generale si evince comunque che il modello di SA tende a sottostimare il valore del coefficiente d'attrito lungo buona parte del tubo, generando quindi un errore rispetto ai dati reali. Proprio questo errore ha generato l'idea di tentare di migliorare il modello, lavorando sulla differenza nei dati per cercare di minimizzarla con un metodo integrabile all'interno del modello stesso ed applicabile anche ad altri casi di studio.

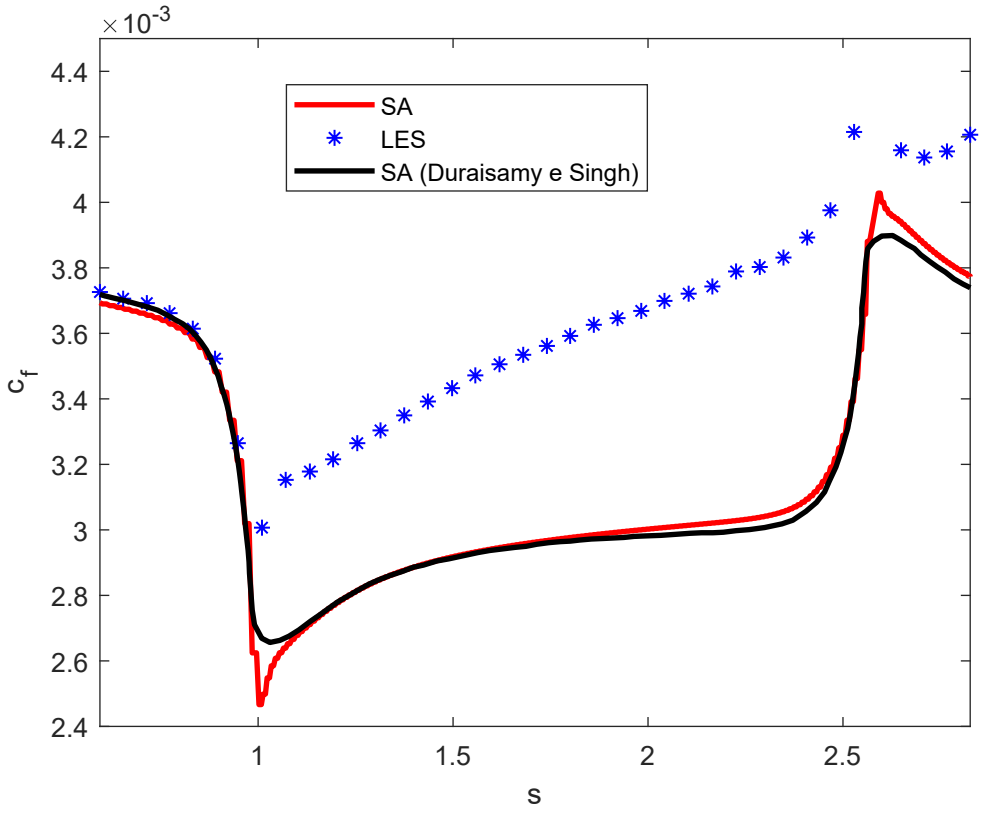


Figura 5.5. Confronto andamento coefficiente d'attrito

## 5.2 Risultati del modello di SA migliorato con l'aggiunto

Nella prima fase del lavoro si è raggiunta una prima e fondamentale soluzione stabile ottenuta col modello base di SA che ha permesso di valutare l'effettiva differenza nei risultati rispetto al modello LES, dovuta alle approssimazioni e assunzioni su cui SA si basa. Partendo da questa soluzione è possibile quindi lavorare col metodo dell'aggiunto per tentare di migliorare il modello scelto, per cui nella seconda fase di calcolo ci si è concetrati proprio su questo obbiettivo. Dopo aver raggiunto la soluzione stazionaria il codice calcola il gradiente della funzione goal scelta con il metodo del gradiente e genera quindi una perturbazione di entità proporzionale ad esso, come spiegato più dettagliatamente nel capitolo precedente. Il modello di SA viene quindi modificato tramite il fattore moltiplicativo $\beta$ applicato al termine di produzione che, dopo un'altra fase di iterazioni temporali, porta ad una nuova soluzione stabile di cui viene estrapolato il coefficiente d'attrito. Nella figura 5.6 è infatti possibile osservare il campo di $\beta$ nel canale dopo l'applicazione del metodo

del gradiente che evidenzia come fuori dallo strato limite il suo valore resti costantemente pari a 1, mentre nello strato limite il suo valore sia fortemente variabile. Questo indica che nella prima zona il modello di base è già sufficientemente preciso e non necessita di modifiche, mentre all'interno dello strato limite SA risulta poco preciso e necessita di forti correzioni, evidenziando quindi l'importante lavoro effettuato col metodo descritto. Un primo modo per verificare se si sta effettiva-

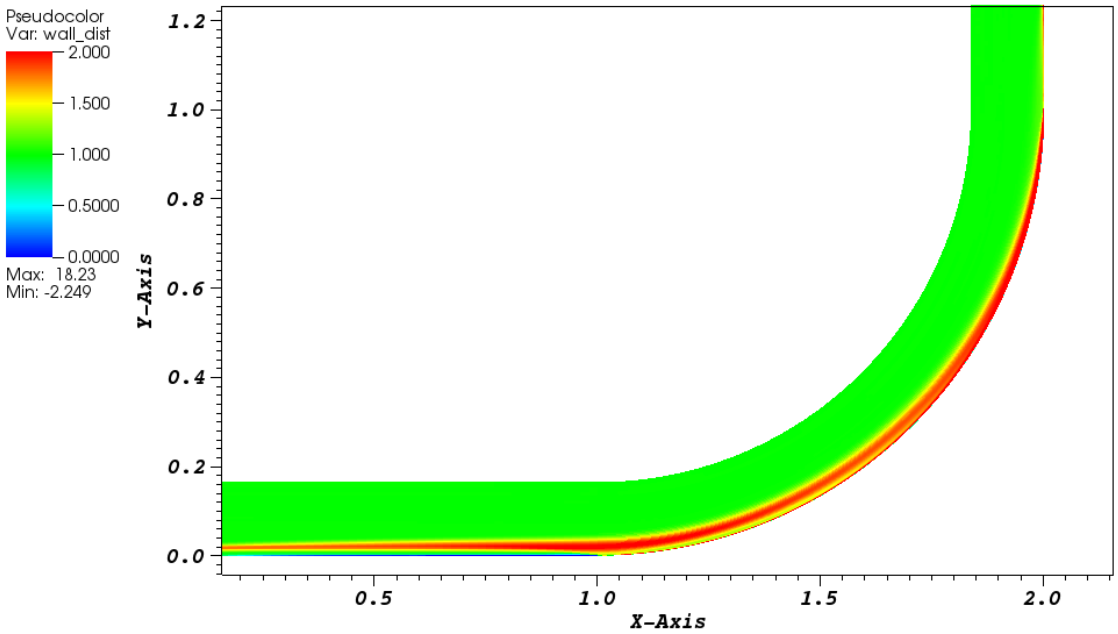


Figura 5.6. Distribuzione di $\beta$ nel condotto

mente lavorando in modo corretto è controllare l'evoluzione della funzione goal che, volendo cercare di minimizzarla per ottenere l'ottimizzazione, dovrebbe diminuire dopo l'applicazione dell'inversione di campo col metodo dell'aggiunto. Osservando la tabella 5.1 si nota facilmente come, effettivamente, vi sia una diminuzione della funzione goal all'aumentare del numero di passi effettuati col metodo del gradiente, con una variazione complessiva notevole che conferma il raggiungimento della minimizzazione voluta e l'avvicinamento dei risultati ai dati di riferimento. Anche il grafico in figura 5.7, che mostra l'andamento di $g$ in funzione del numero di passi del metodo del gradiente, conferma visivamente l'andamento decrescente della funzione goal ed evidenzia, inoltre, come inizialmente vi sia una diminuzione molto rapida di quest'ultima che poi si riduce drasticamente e diventa minima, tanto da sembrare quasi nulla alla fine. In realtà la diminuzione continua sempre, anche dopo moltissimi passi del metodo del gradiente, ma il beneficio che se ne trae diminuisce esponenzialmente rendendo sempre più costoso, a livello di calcoli, ottenere anche il minimo miglioramento apprezzabile sul coefficiente d'attrito, per cui non saranno

Tabella 5.1. Funzione goal

| $k_{grad}$ | $g$ |
|---|---|
| 1 | $7.5192 * 10^{-7}$ |
| 10 | $4.4812 * 10^{-7}$ |
| 25 | $2.1960 * 10^{-7}$ |
| 50 | $8.7141 * 10^{-8}$ |
| 75 | $4.4738 * 10^{-8}$ |
| 100 | $2.5486 * 10^{-8}$ |
| 150 | $9.6395 * 10^{-9}$ |
| 200 | $3.8383 * 10^{-9}$ |
| 500 | $1.3005 * 10^{-10}$ |
| 1000 | $9.7018 * 10^{-11}$ |
| 2000 | $8.9474 * 10^{-11}$ |

più sufficienti pochi passi ma ne serviranno molti. Parallelamente, però, diminuirà di molto anche la perturbazione generata sulla soluzione stabile dal nuovo valore di $\beta$ derivante dall'aggiunto, renendo quindi necessarie meno iterazioni temporali per tornare a convergenza ed ottenere una nuova soluzione.

Nel grafico in figura 5.8 si può osservare il confronto tra il $c_f$ precedentemente calcolato dalla prima soluzione, quello ricavato dalla nuova soluzione e quello della LES, si nota subito che il nuovo $c_f$ risulta identico nell'andamento ma più vicino ai valori della LES per cui effettivamente si è ottenuto un miglioramento nei risultati e quindi nell'accuratezza del modello. I nuovi valori del coefficiente d'attrito restano però ancora inferiori rispetto a quelli reali, per cui permangono delle imprecisioni nel modello e il margine di differenza nei dati su cui lavorare è ancora considerevole. Tutto il processo precedentemente descritto viene infatti ripetuto innumerevoli volte, raggiungendo sempre nuove soluzioni stazionarie derivanti da nuovi valori di $\beta$ che generano valori di $c_f$ sempre più accurati e simili ai risultati della LES. In figura 5.9 infatti è ben visibile come ad ogni passo del metodo dell'aggiunto la funzione del $c_f$ aumenti e si avvicini sempre più ai valori della curva dei risultati reali, certificando quindi la correttezza del metodo scelto per migliorare il modello di SA, con ulteriore conferma deducibile dai dati numerici visibili nella tabella 5.2.
Ovviamente si osserva anche che permangono comunque delle imprecisioni e degli errori nell'andamento del $c_f$, che non raggiunge mai la perfetta coincidenza con i dati LES in quanto permangono delle approssimazioni e semplificazioni anche dopo i miglioramenti apportati al metodo. Inoltre, una maggior quantità di dati sperimentali avrebbe aiutato molto il raggiungimento di risultati migliori, permettendo di sfruttare maggiormente le potenzialità del metodo illustrato in questa tesi, che

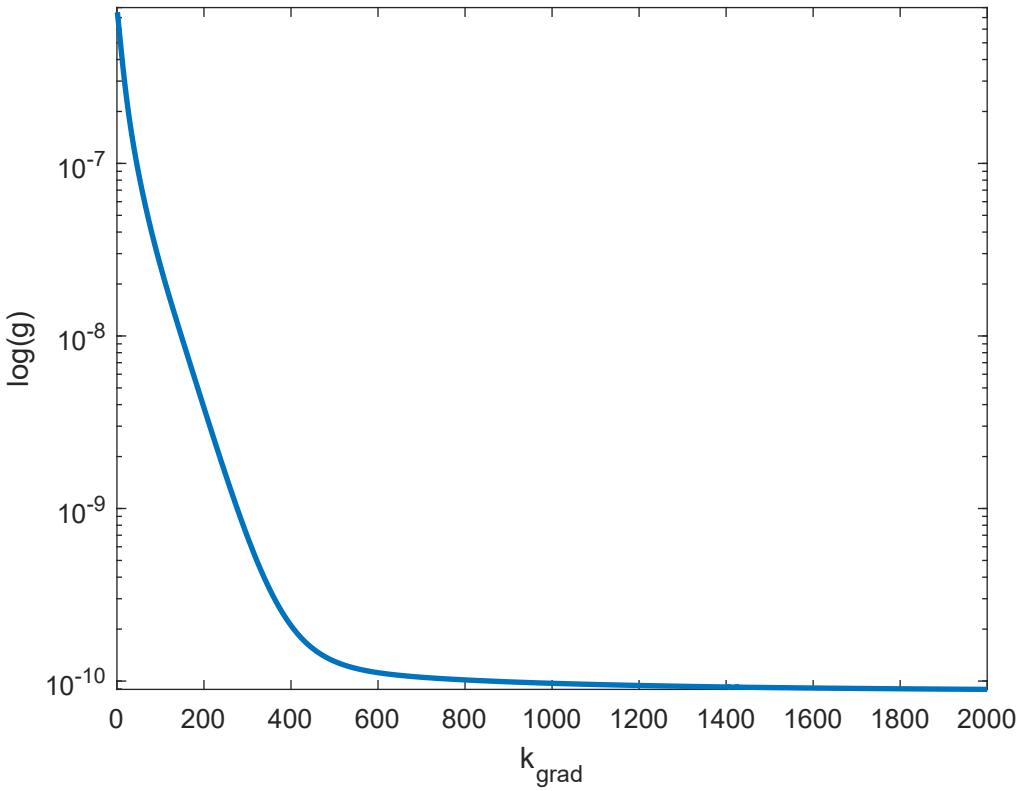


Figura 5.7. Andamento della funzione goal

comunque ha già dato buoni risultati. I punti più critici dei grafici, in cui l'andamento dei risultati rimane ancora distante o differente da quello di riferimento, corrispondono infatti alle zone in cui la quantità dei dati LES è molto bassa, come per esempio vicino al punto di massimo finale, mentre dove vi sono più dati, come per esempio nella zona lineare centrale, si raggiunge la corrispondenza quasi perfetta. L'ottimizzazione migliore è stata raggiunta dopo 2000 passi del metodo del gradiente effettuati in circa 36 ore di calcolo parallelizzato su 96 core, si sarebbe potuto andare ancora oltre ma, come già spiegato precedentemente, il miglioramento del $c_f$ risulta sempre meno apprezzabile e il tempo di calcolo necessario sempre maggiore. Si deduce, quindi, che è stato raggiunto il limite di applicabilità del metodo per i dati a disposizione e le scelte fatte, risultando insensato tentare di proseguire ed andare oltre per raggiungere una maggiore accuratezza. Concludendo è possibile, quindi, dire che il risultato prefissato per questa tesi è stato raggiunto con successo in quanto si è effettivamente migliorato il modello di turbolenza di Spalart-Allmaras che, corretto col valore di $\beta$ ricavato con l'inversione di campo effettuata tramite gradiente calcolato con il metodo dell'aggiunto, risulta infine

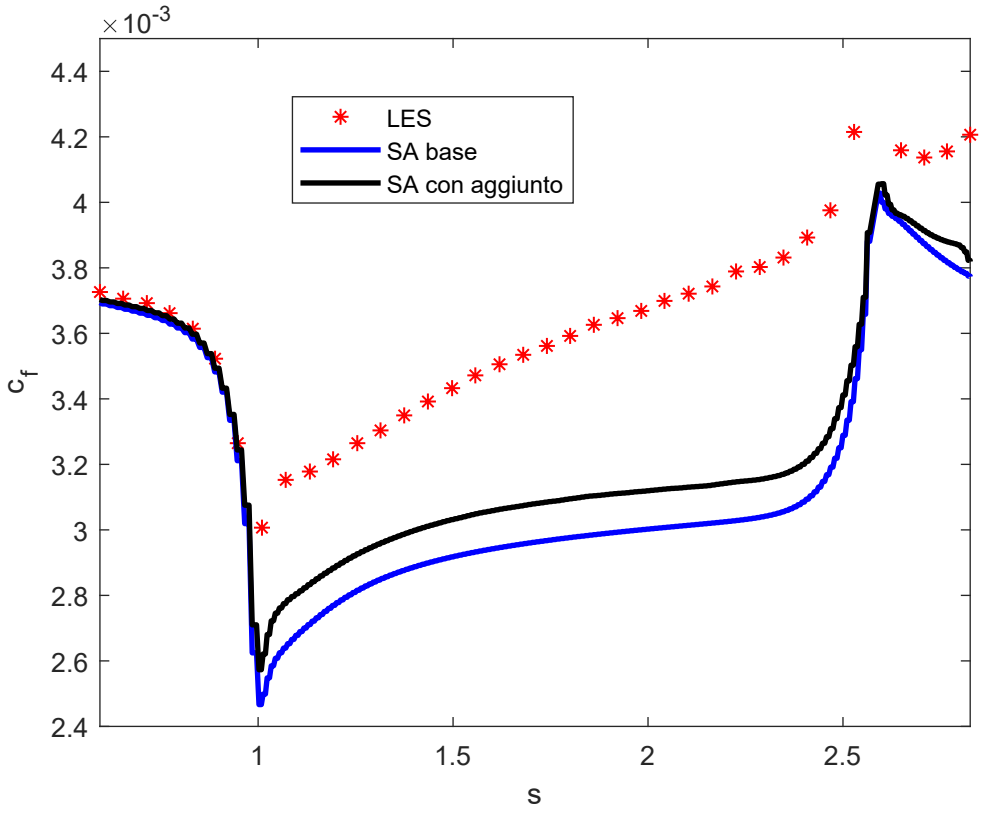


Figura 5.8. Prova preliminare del metodo con 10 passi di ottimizzazione

nettamente più corretto e accurato nel calcolo del coefficiente d'attrito agente sulla parete inferiore del condotto studiato.

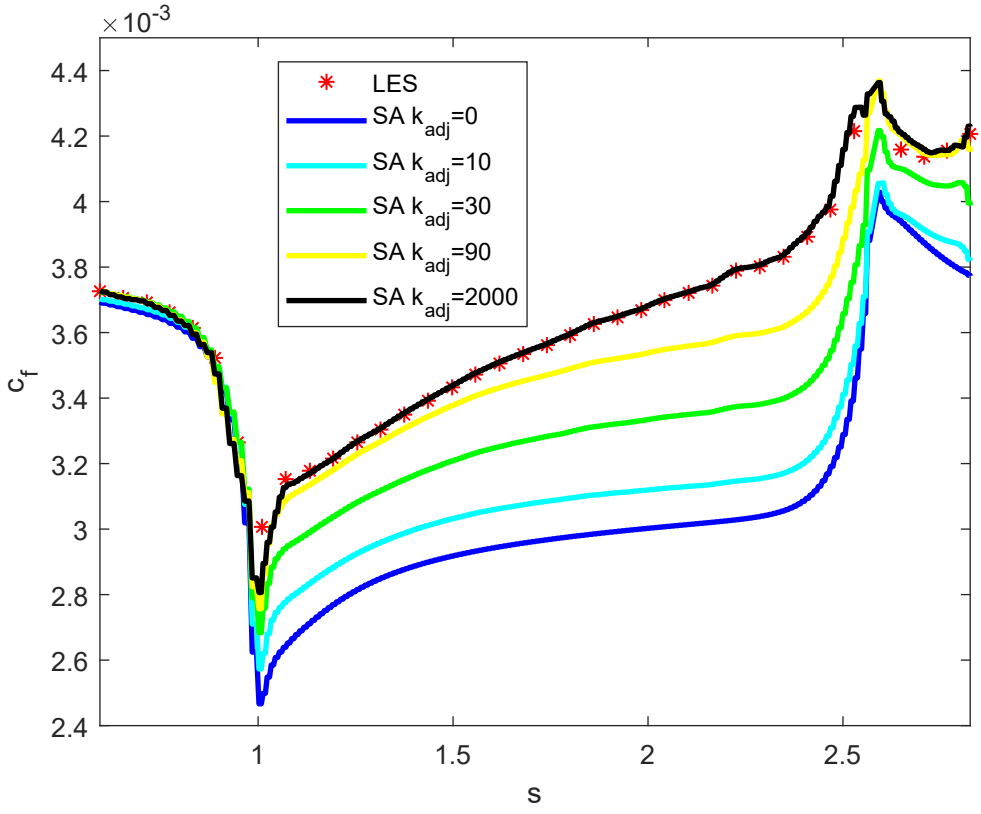


Figura 5.9. Miglioramento andamento del coefficiente d'attrito

Tabella 5.2. Confronto risultati del $c_f$

| $s$ | $c_f$ (SA base) | $c_f$ (SA migliorato) | $c_{f,exp}$ (LES) |
|---|---|---|---|
| 0.594292803970223 | 0.003700487130720 | 0.003723963204128 | 0.003726244343891 |
| 0.653846153846153 | 0.003686496631366 | 0.003703893069310 | 0.003705882352941 |
| 0.714640198511166 | 0.003669742816553 | 0.003687880531092 | 0.003692307692308 |
| 0.772952853598014 | 0.003643751726607 | 0.003652508071880 | 0.003661764705882 |
| 0.832506203473945 | 0.003597472214126 | 0.003595383562638 | 0.003614253393665 |
| 0.889578163771712 | 0.003493696497638 | 0.003473751983698 | 0.003522624434389 |
| 0.947890818858561 | 0.003244050653189 | 0.003163398428111 | 0.003264705882353 |
| 1.009925558312650 | 0.002621198351527 | 0.002895403895184 | 0.003006787330317 |
| 1.070719602977660 | 0.002786851851156 | 0.003136327400483 | 0.003152714932127 |
| 1.132754342431760 | 0.002841911008001 | 0.003176667853724 | 0.003178167420814 |
| 1.192307692307690 | 0.002891555233842 | 0.003225384162404 | 0.003215497737557 |
| 1.254342431761780 | 0.002933448355081 | 0.003276690944379 | 0.003264705882353 |
| 1.313895781637710 | 0.002961626037715 | 0.003311898810029 | 0.003303733031674 |
| 1.374689826302720 | 0.002990420957437 | 0.003359744221199 | 0.003349547511312 |
| 1.435483870967740 | 0.003014317598537 | 0.003402914946568 | 0.003391968325792 |
| 1.497518610421830 | 0.003034164346783 | 0.003444301406249 | 0.003432692307692 |
| 1.557071960297760 | 0.003049287682356 | 0.003478115807134 | 0.003471719457014 |
| 1.619106699751860 | 0.003063476770894 | 0.003512073709203 | 0.003505656108597 |
| 1.679900744416870 | 0.003075379678168 | 0.003541626190411 | 0.003534502262443 |
| 1.740694789081880 | 0.003086322128208 | 0.003569969188275 | 0.003561651583710 |
| 1.800248138957810 | 0.003096475001283 | 0.003602671104824 | 0.003592194570136 |
| 1.861042183622820 | 0.003104217199832 | 0.003631100325071 | 0.003626131221719 |
| 1.921836228287840 | 0.003111538386425 | 0.003651396940689 | 0.003646493212670 |
| 1.982630272952850 | 0.003118083098363 | 0.003676194481548 | 0.003668552036199 |
| 2.042183622828780 | 0.003125570207845 | 0.003706746287447 | 0.003699095022624 |
| 2.104218362282870 | 0.003131439591593 | 0.003728784759004 | 0.003721153846154 |
| 2.165012406947890 | 0.003137069581928 | 0.003748906925257 | 0.003743212669683 |
| 2.225806451612900 | 0.003147087971573 | 0.003795144913296 | 0.003789027149321 |
| 2.286600496277910 | 0.003155317423732 | 0.003810055089216 | 0.003802601809955 |
| 2.347394540942920 | 0.003173986080514 | 0.003846481911389 | 0.003831447963801 |
| 2.408188585607940 | 0.003221008211218 | 0.003921524365183 | 0.003892533936652 |
| 2.467741935483870 | 0.003309165228669 | 0.004016024437589 | 0.003975678733032 |
| 2.528535980148880 | 0.003559376353293 | 0.004287427019064 | 0.004214932126697 |
| 2.648883374689820 | 0.003957157935684 | 0.004204681989523 | 0.004158936651584 |
| 2.708436724565750 | 0.003907071706219 | 0.004152536149575 | 0.004136877828054 |
| 2.648883374689820 | 0.003876063232186 | 0.004163387161613 | 0.004155542986425 |
| 2.708436724565750 | 0.003784601463635 | 0.004211285837792 | 0.004206447963801 |

# Capitolo 6

# Possibilità di sviluppo futuro

I risultati raggiunti in questo lavoro sono molto promettenti ed interessanti dato che verificano la correttezza dell'idea di applicare il metodo dell'inversione di campo, con implementazione del metodo dell'aggiunto, per migliorare la precisione dei risultati di un modello di turbolenza molto utilizzato in ambito CFD. In questo caso si è lavorato sostanzialmente solo su un parametro del metodo, le uniche modifiche sono state apportate infatti tramite il fattore moltiplicativo $\beta$ al termine di produzione $P$ di SA, e si è utilizzata una sola variabile, il coefficiente d'attrito $c_f$, come parametro di confronto e verifica ma è ipotizzabile pensare di applicare lo stesso metodo ad altri parametri del modello utilizzando altre variabili fluidodinamiche come controllo. Idealmente più parametri vengono corretti e più variabili fluidodinamiche vengono monitorate, più si riuscirà alla fine ad ottenere un modello accurato e corretto in tutti gli aspetti, proporzionalmente però aumenta anche la complessità del tenere conto di tutti i risultati e di derivarne le modifiche appropriate da apportare al modello che si sta studiando. Allo stesso modo si può pensare di applicare lo stesso metodo di miglioramento ad altri modelli di turbolenza diversi da Spalart-Allmaras o, addirittura, anche ad altri tipi di modelli, per cui il campo di applicazione di questo studio è teoricamente molto vasto. Un'altra possibilità di miglioramento e modifica del metodo presentato riguarda l'implementazione di algoritmi di ottimizzazione più efficienti rispetto al metodo di discesa del gradiente, quali per esempio possono essere il metodo di Newton [21], molto complesso da implementare, o i metodi quasi-Newton come il BFGS [22], che mantiene alcune caratteristiche del metodo di Newton ma risulta più facilmente utilizzabile.

Un problema di questo metodo è, però, legato alla generalizzazione ed al riutilizzo del modello migliorato che si ottiene, l'ottimizzazione che si effettua è infatti tanto buona quanto legata allo specifico caso di studio utilizzato. Per cercare di generalizzare il modello descritto è necessario legare i miglioramenti ottenuti alle variabili fluidodinamiche del campo di moto tramite relazioni generiche, non dipendenti dal test case e dalla sua geometria e complessità. Questa operazione può essere fatta tramite le reti neurali e gli algoritmi del machine learning, che

permettono infatti di cercare le relazioni nascoste tra diverse variabili e renderle riapplicabili. In particolare l'idea di base è quella di addestrare una rete neurale a trovare le relazioni tra il fattore moltiplicativo $\beta$ e tutte le variabili conservative ed i loro gradienti ricavate dal relativo campo di moto del fluido. Una volta che la rete è addestrata a dovere è in grado di ricavare il campo di $\beta$ in funzione delle grandezze fluidodinamiche da cui dipende, facilmente rilevabili per qualsiasi caso di studio, per cui, implementandola all'interno del modello di turbolenza, rende ques'utlimo generalizzato e riutilizzabile per altri problemi. Per poter effettuare l'addestramento è necessario disporre di molti dati relativi ai parametri di cui si vogliono ricavare le correlazioni, più se ne hanno più l'addestramento sarà accurato, per cui l'ideale è applicare lo stesso metodo a diversi test case ed unire tutti i dati per generare il set da addestramento. Con il machine learning è possibile anche pensare di lavorare parallelamente su più parametri del modello in esame e su più variabili fluidodinamiche parallelamente, come precedentemente spiegato, ottenendo quindi idealmente dei modelli molto accurati ed efficienti utilizzabili in diversi casi di studio differenti. Ovviamente, anche in questo caso più parametri si prendono in analisi, più accurato sarà il risultato ma più complesso sarà determinare la corretta architettura della rete ed addestrarla adeguatamente.

## 6.1 Un'esempio di rete neurale

Per lo sviluppo e lo studio delle reti neurali artificiali e dei relativi algoritmi di machine learning esistono diversi strumenti e software, uno dei più famosi ed utilizzati è sicuramente TensorFlow [23] che è una piattaforma molto potente e versatile, basata su linguaggio Python, sviluppata da Google, che risultà però complessa da utilizzare direttamente per i neofiti del settore. Una libreria molto utile, utilizzabile con TensorFlow, è Keras [24] che permette di semplificare molto la realizzazione del codice per la programmazione della rete e l'implementazione dei relativi algoritmi e, tra l'altro, contiene degli utilissimi strumenti che danno la possibilità di valutare le prestazioni delle reti create ed ottimizzarne l'architettura. Come già accennato precedentemente, la struttra di una rete neurale è simile a quella del cervello umano, ovvero si basa su moltissimi neuroni interconnessi tra loro e, nella versione più comune di architettura, organizzati in strati e sottostrati, alcuni definiti nascosti in quanto non si interfacciano direttamente con l'esterno della rete. Come visibile in figura 6.1 vi sono due principali strati dedicati all'input ed all'output dei dati, mentre quelli nascosti sono dedicati all'elaborazione dei dati ed al calcolo dei risultati. Ogni neurone, detto nodo, è collegato a tutti i nodi dello strato successivo e di quello precedente, ovviamente se presenti, ed ogni connessione è caratterizzata da una variabile chiamata peso che permette di trasferire le informazioni tra i nodi. Per il training della rete il set di dati scelto, composto da input ed output, viene fornito ai relativi strati e viene poi elaborato dai diversi nodi parallelamente seguendo l'algoritmo di addestramento che, quando completato, permette di determinare il

valore corretto di tutti i pesi della rete. Dopo che la rete risulta addestrata è possibile testarla con un nuovo set di dati formato da soli input, che verranno elaborati con i pesi determinati in addestramento, i cui risultati verranno forniti dallo strato di output.

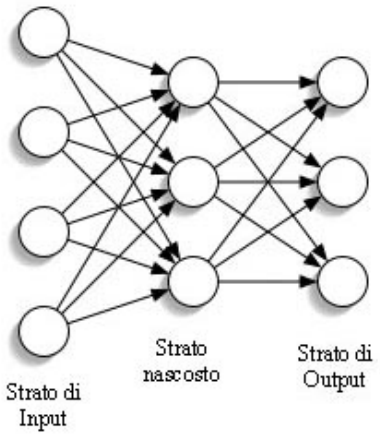


Figura 6.1. Arichettura di una rete neurale multistrato (Fonte:[25])

Si ipotizzi, per esempio, di utilizzare come set di dati per il training di una rete neurale un database sul diabete, in cui sono immagazzinate molte informazioni cliniche di un elevato numero di pazienti, presi come gruppo di studio, di cui alcuni affetti da diabete [26]. Lo scopo dello studio è di cercare una correlazione tra i parametri clinici rilevati nei pazienti e il fato che siano o meno diabetici, per cui è possibile utilizzare una rete neurale per questo scopo. Il database contiene tutte le informazioni in formato numerico, cosa che facilita il lavoro di preparazione della rete, infatti la variabile che indica se il paziente abbia o meno il diabete vale 1 nel primo caso e 0 nel secondo, facendo ricadere lo studio nei casi di classificazione binaria. Un esempio del database è visibile in tabella 6.1, in cui i dati clinici elencati

Tabella 6.1. Database sul diabete

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

sono:

1. numero di gravidanze
2. concentrazione di glucosio nel plasma dopo un test di tolleranza specifico
3. pressione diastolica del sangue
4. spessore della pelle
5. quantità di insulina sierica a 2 ore dall'assunzione di glucosio
6. indice di massa corporea
7. indice di familiarità col diabete
8. età
9. paziente affetto da diabete (1 si 0 no)

per cui i dati delle prime sette colonne saranno gli input, mentre quelli dell'ultima colonna saranno gli output. Volendo, quindi, elaborare questi dati con una rete neurale è necessario prima di tutto sceglierne l'architettura, processo che risulta molto complesso e può essere perfezionato solo con l'esperienza, non basandosi su linee guida predeterminate ed essendo difficilmente oggetivabile. Per questo semplice esempio si è scelto di utilizzare una semplice rete multistrato formata da 4 layer formati rispettivamente da 8 12 8 ed 1 nodo. Il codice risulta quindi:

```
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy
# fix random seed for reproducibility
numpy.random.seed(7)
# load pima indians dataset
dataset = numpy.loadtxt("dati_diabete.csv", delimiter=",")
# split into input (X) and output (Y) variables
X = dataset[:,0:8]
Y = dataset[:,8]
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, kernel_initializer='uniform', \
activation='relu'))
model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
model.add(Dense(1, kernel_initializer='uniform', \
activation='sigmoid'))
```

Dopo aver importato tutte le librerie necessarie ed il database di dati, Keras permette di costruire molto facilmente una rete neurale multistrato, scegliendo infatti il modello di tipo sequenziale è possibile aggiungere manualmente ogni strato definendone il numero di nodi e, nel primo, anche il numero di variabili di input a cui corrisponderà la dimensione dello strato di input. Con il primo comando viene creato quindi il layer di input e un hidden layer, a cui se ne aggiungono altri dello stesso tipo con ogni istruzione di aggiunta, mentre l'ultimo strato aggiunto diventerà quello di output, di dimensione pari al numero di variabili di output. I nodi scelti sono totalmente connessi tra loro poichè appartenenti alla classe Dense ed è necessario, inoltre, decidere il tipo di inizializzazione e la funzione di attivazione di ogni strato, scegliendo tra quelle standard disponibili (relu = rectified linear unit activation function; sigmoid = Sigmoid function). A questo punto l'architettura della rete è pronta ed è possibile compilare il modello, per farlo è necessario scegliere la funzione di perdita e l'ottimizzatore da usare per calcolare i pesi della rete e un'eventuale parametro di controllo da rilevare e riportare alla fine dell'addestramento. Il codice risulta:

```
# compile model
model.compile(loss='binary_crossentropy', optimizer='adam', \
metrics=['accuracy'])
```

In questo caso si è scelta come funzione di perdita l'entropia incrociata [27], come parametro di controllo l'accuratezza della rete e come ottimizzatore un'algoritmo chiamato adam basato sul metodo del gradiente discendente. Dopo aver preparato la rete si può passare, quindi, all'addestramento (fit) vero e proprio, realizzabile molto semplicemente grazie a Keras:

```
# fit the model
history = model.fit(X, Y, validation_split=0.33, epochs=150, \
batch_size=10, verbose=2)
```

Il numero di "epochs" indica il numero di iterazioni imposte per l'addestramento, mentre il "bach size" indica ogni quante colonne del database di dati l'algoritmo aggiorna i pesi del modello; per ognuna delle 150 iterazioni di addestramento, quindi, verranno lette tutte le colonne del set di dati ma i pesi verranno aggiornati solo ogni 10 colonne. Il termine "validation split" indica invece la percentuale di dati del database di training che Keras userà per generare automaticamente un database di dati di test. La rete risulta a questo punto addestrata e può essere utilizzata su un nuovo database di cui non si conoscono i valori di output, ma in questo caso, per semplicità e per valutare la correttezza del lavoro fatto, si è scelto di testare la rete sullo stesso database di addestramento:

```
# evaluate the model
_, accuracy = model.evaluate(X, Y)
```

```
print('Accuracy: %.2f' % (accuracy*100))
# make class predictions with the model
predictions = model.predict_classes(X)
# summarize the first 20 cases
for i in range(20):
  print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], \
          Y[i]))
```

Anche per questo operazione il codice risulta molto semplice utilizzando Keras, che permette inoltre di visualizzare il valore dell'accuratezza della rete precedentemente salvato. Il codice restituisce quindi in questo caso:

```
.....
Epoch 149/150
 - 0s - loss: 0.4939 - acc: 0.7490 - val_loss: 0.5044
 - val_acc: 0.7756
Epoch 150/150
 - 0s - loss: 0.4902 - acc: 0.7607 - val_loss: 0.5057
 - val_acc: 0.7795
32/768 [>.............................] - ETA: 0s
768/768 [==============================] - 0s 43us/step

Accuracy: 76.17

[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0] => 0 (expected 1)
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0] => 0 (expected 0)
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0] => 1 (expected 1)
[1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0] => 0 (expected 0)
[0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0] => 1 (expected 1)
[5.0, 116.0, 74.0, 0.0, 0.0, 25.6, 0.201, 30.0] => 0 (expected 0)
[3.0, 78.0, 50.0, 32.0, 88.0, 31.0, 0.248, 26.0] => 0 (expected 1)
[10.0, 115.0, 0.0, 0.0, 0.0, 35.3, 0.134, 29.0] => 1 (expected 0)
[2.0, 197.0, 70.0, 45.0, 543.0, 30.5, 0.158, 53.0] => 1 (expected 1)
[8.0, 125.0, 96.0, 0.0, 0.0, 0.0, 0.232, 54.0] => 0 (expected 1)
[4.0, 110.0, 92.0, 0.0, 0.0, 37.6, 0.191, 30.0] => 0 (expected 0)
[10.0, 168.0, 74.0, 0.0, 0.0, 38.0, 0.537, 34.0] => 1 (expected 1)
[10.0, 139.0, 80.0, 0.0, 0.0, 27.1, 1.441, 57.0] => 0 (expected 0)
[1.0, 189.0, 60.0, 23.0, 846.0, 30.1, 0.398, 59.0] => 1 (expected 1)
[5.0, 166.0, 72.0, 19.0, 175.0, 25.8, 0.587, 51.0] => 1 (expected 1)
[7.0, 100.0, 0.0, 0.0, 0.0, 30.0, 0.484, 32.0] => 0 (expected 1)
[0.0, 118.0, 84.0, 47.0, 230.0, 45.8, 0.551, 31.0] => 0 (expected 1)
[7.0, 107.0, 74.0, 0.0, 0.0, 29.6, 0.254, 31.0] => 0 (expected 1)
[1.0, 103.0, 30.0, 38.0, 83.0, 43.3, 0.183, 33.0] => 0 (expected 0)
```

```
[1.0, 115.0, 70.0, 30.0, 96.0, 34.6, 0.529, 32.0] => 0 (expected 1)
```

Si nota subito come l'accuratezza della rete risulti abbastanza elevata ma non perfetta, infatti, utilizzando anche solo i primi 20 pazienti come database di test, si osserva come i risultati in alcuni casi siano errati ma per la maggior parte venga effettuata una previsione corretta. Inoltre, per valutare ulteriormente la qualità dell'addestramento della rete, è possibile visualizzare i dati relativi all'evoluzione dell'accuratezza e della perdita del modello [28]:

```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Nei grafici delle immagini 6.2 e 6.3 è possibile, infatti, osservare l'andamento dell'accuratezza e delle perdite del modello di rete neurale scelto all'aumentare del numero di iterazioni di addestramento. Le perdite sono espresse dalla funzione di entropia incrociata, come già precedentemente spiegato, mentre l'accuratezza è definita utilizzando gli stessi dati di addestramento per effettuare una predizione con la rete neurale, condrontando poi i risultati con quelli reali del database. In entrambi i grafici vi sono due linee, una riferita al set di dati di addestramento scelto e l'altra al set di dati di test creato automaticamente da Keras precedentemente citato. Come prevedibile, l'accuratezza aumenta col crescere dei cicli di addestramento mentre le perdite diminuiscono, certificando la bontà dell'addestramento, che però potrebbe essere prolungato auentando il numero di iterzioni dato che la curva dell'accuratezza non ha ancora raggiunto la stazionarietà. Tutte le scelte legate ai parametri ed all'architettura della rete, come già detto, sono fortemente legate all'esperienza del progettista, ma Keras permette comunque di utilizzare degli specifici strumenti per poterle ottimizzare, i quali, uniti ad un elevato numero di test e prove, possono portare a migliorare notevolmente le prestazioni e raggiungere un grado di accuratezza molto elevato.

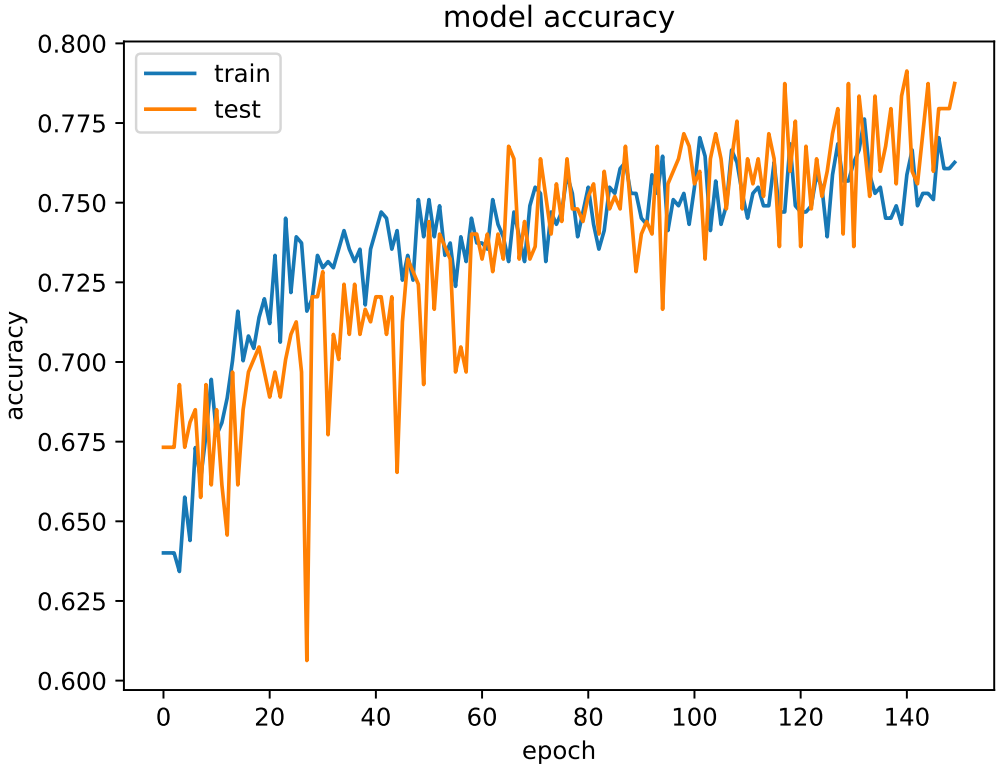


Figura 6.2. Andamento dell'accuratezza durante l'addestramento della rete

In poche righe di codice è stato possibile creare una rete neurale artificiale ed addestrarla a trovare una relazione tra il diabete e una serie di dati clinici, cosa che sarebbe stata molto complessa da fare diversamente, mentre con l'ausilio di questo potente strumento risulta relativamente semplice. Questo era solamente un piccolo e semplice esempio delle potenzialità delle reti neurali e del machine learning, applicando lo stesso principio ai dati fluidodinamici ricavati da questa tesi, è facile quindi dedurre come sia effettivamente possibile addestrare una rete neurale per migliorare il modello di turbolenza di SA ed implementarla poi direttamente al suo interno, come spiegato nei capitoli precedenti.

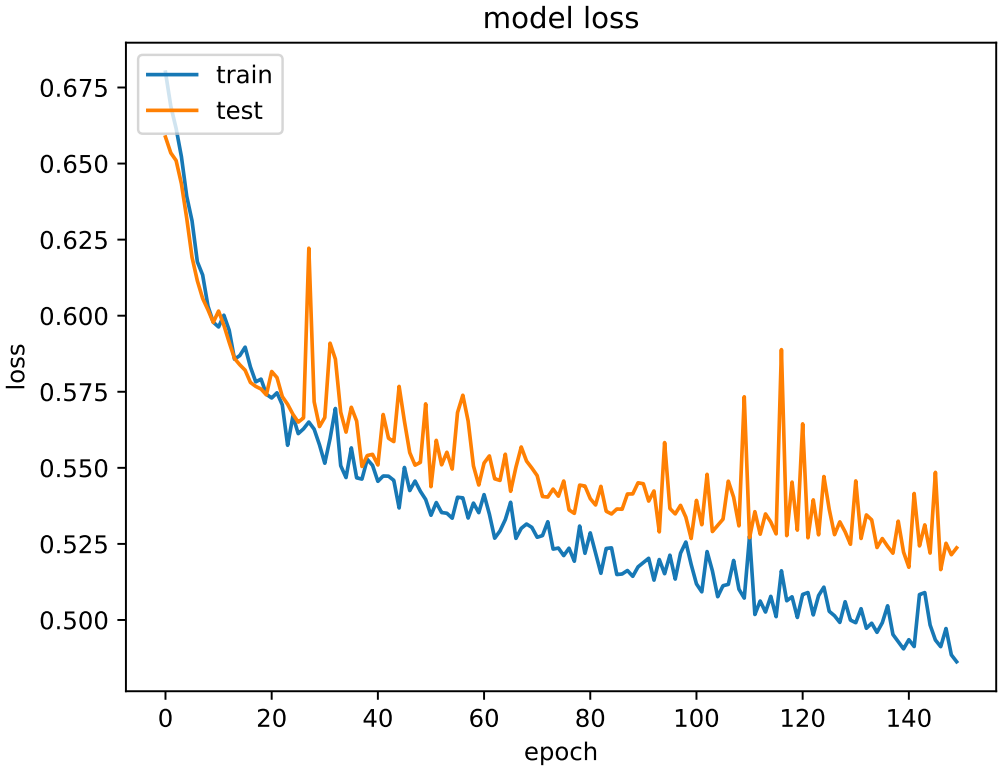


Figura 6.3. Andamento delle perdite durante l'addestramento della rete

# Bibliografia


[1] Nicoletta Boldrini. *Cos'è l'Intelligenza Artificiale, perché tutti ne parlano e quali sono gli ambiti applicativi*. 2019. URL: https://www.ai4business.it/intelligenza-artificiale/intelligenza-artificiale-cose/.

[2] Nicoletta Boldrini. *Cos'è il Machine Learning, come funziona e quali sono le sue applicazioni*. 2018. URL: https://www.ai4business.it/intelligenza-artificiale/machine-learning/machine-learning-cosa-e-applicazioni/.

[3] Nicoletta Boldrini. *Reti neurali: cosa sono e a cosa servono*. 2019. URL: https://www.ai4business.it/intelligenza-artificiale/deep-learning/reti-neurali/.

[4] *Sito di articoli sulle reti neurali*. URL: https://wagenaartje.github.io/neataptic/docs/builtins/perceptron/.

[5] Enrico Nobile. *Introduzione ai modelli di turbolenza in CFD*. Presentazione. Facoltà di Ingegneria, università di Trieste, 2017.

[6] *Sito di Ansys, software commerciale di simulazione CFD*. URL: https://www.ansys.com/it-it/products/fluids.

[7] Zhang Weiwei et al. *Machine learning methods for turbulence modeling in subsonic flows over airfoils*. Relazione. School of aeronautics, Northwestern Polytechnical University, Xi'an 710072, China, 2018.

[8] Milano Michele e Koumoutsakos Petros. *Neural network modeling for near wall turbulent flow*. Relazione. Institute of computational sciences, Zürich, Switzerland, 2002.

[9] Hocevar Marco, Sirok Brane e Grabec Igor. *A turbulent-wake estimation using radial basis function neural networks*. Relazione. Faculty of mechanical engineering, University of Ljubljana, Slovenia, 2005.

[10] Singh Anand Pratap e Duraisamy Karthik. *Using field inversion to quantify functional errors in turbulence closures*. Relazione. Department of aerospace engineering, University of Michigan, USA, 2016.

[11] Singh Anand Pratap, Duraisamy Karthik e Medida Shivaji. *Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils*. Relazione. Department of aerospace engineering, University of Michigan, USA, 2017.

[12] Ferrero Andrea. «Computational fluid dynamics for aerospace propulsion system: an approach based on discontinuous finite elements». Tesi di dottorato. Politecnico di Torino, 2014.

[13] Blazek Josef. *Computational fluid dynamics: principles and applications*. 2001.

[14] Spalart P.R e Allmaras S.R. «A one-equation turbulence model for aerodynamic flows». In: *AIAA 30th aerospace science meeting and exhibit* (1993).

[15] Spalart P.R, Allmaras S.R. e Johnson F.T. «Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model». In: *Seventh international conference on computational fluid dynamics (ICCFD7)* (2012).

[16] Bassi F. et al. «On the flexibility of agglomeration based physical space discontinuous Galerking discretizations». In: *Journal of computational physics* (2012).

[17] Pironneau O. «On optimum design in fluid mechanics». In: *Journal of fluid mechanics* (1974).

[18] Jameson Antony. *Optimum aerodynamic design using CFD and control theory*. Articolo. Department of Mechanical e Aerospace Engineering, Princeton University, USA, 1995.

[19] Giles Michael B. e Pierce Niles A. «An introduction to the adjoint approach to design». In: *Flow, turbulence and combustion*. 2000.

[20] Hascoet Laurent e Pascual Valèrie. «The Tapenade automatic differentiation tool: principles, model and specification». In: *ACM transactions on mathematical software (TOMS)* (2013).

[21] Boyd Stephen e Vandenberghe Lieven. *Convex optimization*. 2004.

[22] Yu-Hong Dai. «A perfect example for the BFGS method». In: *Mathematical Programming* (2012).

[23] *Sito di TensorFlow*. URL: `https://www.tensorflow.org/about`.

[24] *Sito di Keras*. URL: `https://keras.io/`.

[25] *Sito dell'enciclopedia "Il Diogene"*. URL: `https://www.ildiogene.it/Ency=retineurali.html`.

[26] Browniee Jason. *Keras tutorial: develop your first neural network in python step-by-step*. 2016. URL: `https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/`.

[27] Vermorel Joannès. *Cos'è l'entropia incrociata*. 2018. URL: https://www.lokad.com/it/definizione-entropia-incrociata.

[28] Browniee Jason. *Display deep learning model training history in keras*. 2016. URL: https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/.