Politecnico di Milano
Facoltà di ingegneria industriale e dell'informazione
Corso di Laurea magistrale in ingegneria aeronautica

Ottimizzazione vincolata adjoint-based: sviluppo di una metodologia open source su piattaforma HPC

Tesi di laurea di:
Mattia Murari
matricola 818146

Relatore:
Prof. Maurizio Quadrio

Correlatore:
Ing. Roberto Pieri

Dicembre 2015

Anno accademico 2014/2015

Abstract

The purpose of this work is to develop a constrained optimization cycle based on the adjoint method, capable of dealing with incompressible external aerodynamic problems for bi-dimensional geometries. The whole loop makes use of open source software only: the adjoint solver has been implemented inside OpenFOAM and the free software Dakota has been chosen as the optimizer.

The approach used is based on the parametrization of the airfoil with a certain number of control points. The presence of both a cost function and a constraint function implies that the adjoint problem must be solved twice, in order to compute the sensitivity distribution on the control points with respect to each function. These values are exploited by an optimization algorithm which provides, as output, the displacement of the control points, from which stems the creation of the configuration for the following step of the cycle.

The results obtained show that, in every condition analysed, the developed process succeeds in reaching a configuration that improves the cost functions in observance of the constraints, pointing out its potentiality for a future development.

Sommario

Questo lavoro si prefigge lo scopo di sviluppare un ciclo di ottimizzazione vincolata basato sul metodo dell'aggiunto, in grado di relazionarsi a problemi di aerodinamica esterna incomprimibile e bidimensionale. Un obbiettivo ricercato è quello di utilizzare esclusivamente strumenti *open source*, infatti il solutore aggiunto è stato implementato all'interno di OpenFOAM, mentre come software di ottimizzazione è stato utilizzato Dakota.
L'approccio utilizzato si basa sulla parametrizzazione del corpo in esame attraverso un certo numero di punti di controllo. La presenza di una funzione obbiettivo e di una funzione vincolo comporta che il problema aggiunto debba essere risolto due volte, fornendo due distribuzioni distinte di sensitività sui punti di controllo del corpo. Successivamente, questi valori fungono da input per un algoritmo di ottimizzazione che fornisce in output gli spostamenti dei punti di controllo, da cui è possibile ricostruire il profilo deformato per il passo successivo del ciclo.
Dall'analisi dei risultati si può concludere che il procedimento sviluppato è in grado di convergere verso una configurazione che ottimizza la funzione obbiettivo nel rispetto dei vincoli imposti, evidenziando delle potenzialità che possono essere sfruttate in vista di un ulteriore sviluppo futuro.

Ringraziamenti

Il ringraziamento più importante lo dedico alla mia famiglia: ai miei nonni, ai miei zii e, in particolare, ai miei genitori che mi hanno sempre sostenuto e motivato e sono stati la colonna portante di tutti questi anni.

Un ringraziamento particolare va a Roberto, Raffaele e Francesco, su cui ho sempre potuto contare in ogni situazione, fornendomi un aiuto che è andato ben oltre i loro doveri professionali, al prof. Quadrio, che è sempre stato in grado di indirizzare questo lavoro nella direzione giusta, e a Leo, che sono felice abbia condiviso con me questa esperienza.

Grazie ad Ambra, che rimane un punto di riferimento certo ogni volta che ho bisogno di aiuto, così come a tutti i miei amici che, tra una partita di basket e una serata in compagnia, mi sono sempre vicini e mi hanno sempre aiutato a far scivolare via anche i momenti più difficili.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Negli ultimi anni l'utilizzo della CFD sia per il progetto aerodinamico di velivoli e veicoli che per la fluidodinamica interna ha assunto un ruolo sempre più importante, anche grazie alla possibilità di poter analizzare in breve tempo un'ampia gamma di soluzioni diverse. Questo dà la possibilità di individuare con anticipo le soluzioni migliori da testare successivamente in galleria del vento, garantendo un notevole risparmio per l'economia del progetto. In quest'ottica, risulterebbe vantaggioso disporre di un processo automatizzabile in grado di ricavare la configurazione ottima tra tutte quelle possibili, considerando i vincoli del progetto. Per soddisfare questa esigenza, dunque, risulta necessario accoppiare l'analisi CFD con metodi di ottimizzazione numerica. Analizzando la letteratura esistente si può constatare che i metodi maggiormente utilizzati per l'ottimizzazione in ambito aerodinamico appartengono a due classi principali: gli algoritmi genetici e i metodi *gradient-based*.

Gli algoritmi genetici sono degli schemi robusti e relativamente semplici da implementare, tuttavia il loro costo computazionale cresce all'aumentare dei parametri di ottimizzazione e non garantiscono una convergenza monotona verso l'ottimo, caratteristiche che non li rendono adeguati a problemi ingegneristici che coinvolgono un elevato numero di variabili di progetto.

Le caratteristiche principali dei metodi a gradiente sono riassunte in [1], in cui gli autori ne evidenziano i principali problemi, connessi fondamentalmente a tre aspetti: *(i)* nel caso siano presenti più ottimi locali, questi metodi localizzano l'ottimo più vicino alla configurazione di partenza, che non necessariamente coincide con l'ottimo globale; *(ii)* il costo e l'accuratezza del metodo dipendono dall'algoritmo per il calcolo dei gradienti; *(iii)* necessitano di un considerevole costo iniziale in termini di tempo e lavoro per l'implementazione di un processo robusto e efficiente. Nel caso si riesca a superare questi tre problemi i metodi *gradient-based* garantiscono una convergenza veloce e monotona verso la configurazione ottima.

Mentre la questione *(i)* è intrinsecamente legata al tipo problema che si deve affrontare e richiede che volta per volta si sia in grado di individuare una condizione di partenza sufficientemente vicina all'ottimo globale, la questione *(ii)* mette in luce l'importanza fondamentale del metodo di calcolo dei gradienti. I due metodi maggiormente utilizzati per questo scopo sono il calcolo attraverso le differenze finite e il metodo aggiunto e, anche in questo caso, ciascuno ha alcuni aspetti positivi come quelli negativi.
Il calcolo basato sulle differenze finite è estremamente semplice da implementare, ma il costo computazionale ad esso associato è direttamente proporzionale al numero di parametri di design n_β. In particolare, assumendo il costo della risoluzione di un problema diretto unitario, il costo del calcolo dei gradienti del funzionale da minimizzare rispetto alle n_β variabili è di $n_\beta + 1$, nel caso si utilizzino le differenze finite al primo ordine, oppure di $2n_\beta$, nel caso invece si usino le differenze finite al secondo ordine. Al contrario, il metodo aggiunto è caratterizzato da una maggiore complessità che si traduce in una maggiore difficoltà iniziale per l'implementazione del metodo, mentre dal punto di vista del costo del calcolo dei gradienti, esso è indipendente dal numero di variabili di progetto ed è circa pari a due simulazioni dirette. Nel caso di problemi complessi in cui è possibile agire su molti parametri, questa differenza sostanziale giustifica l'investimento iniziale per l'implementazione di un ciclo di ottimizzazione basato sul metodo aggiunto. A sua volta, all'interno del metodo aggiunto si possono distinguere due approcci: il metodo aggiunto discreto prevede la derivazione dell'operatore aggiunto partendo dalle equazioni di Navier-Stokes già discretizzate, mentre il metodo aggiunto continuo prevede la derivazione del sistema aggiunto dalle equazioni di Navier-Stokes continue e successivamente la discretizzazione dell'operatore aggiunto. Quest'ultimo è il metodo che sarà utilizzato nel resto del lavoro, per la sua maggior semplicità di implementazione. La differenza tra questi due approcci non verrà approfondita ulteriormente in quanto, nonostante alcune diversità, queste due strade portano a risultati che non differiscono significativamente fra loro. Per una trattazione più precisa di questa questione si può fare riferimento all'articolo [2].
Dopo questo considerazioni, l'ultimo problema *(iii)* rimasto aperto è quello di sviluppare il ciclo di ottimizzazione.

1.1 Scopo del lavoro

Al momento, l'unico esempio di codice nato proprio con lo scopo di servire da tool di ottimizzazione è rappresentato dal software SU^2, di cui è appena uscita la quarta *release* [3]. Questo software *open-source*, sviluppato dall'università di Stanford, ha come target di riferimento problemi di ottimizzazione aerodinamica comprimibile in regime transonico, fornendo a questo scopo vari solver per le equazioni di Eulero e Navier-Stokes e

le relative equazioni aggiunte, insieme a diversi moduli in grado di gestire parametrizzazioni e deformazioni della mesh. Allo stesso tempo, esistono pochi software CFD, sia commerciali che liberi, che contengono al loro interno dei solutori per il sistema aggiunto. Ispirandosi a SU^2, lo scopo a lungo termine di questo progetto è quello di sviluppare un ciclo di ottimizzazione aerodinamica vincolata *gradient-based* che sfrutti un solutore aggiunto delle equazioni di Navier-Stokes incomprimibili per il calcolo dei gradienti della funzione obbiettivo e delle funzioni vincolo rispetto alle variabili di progetto. Con questo obbiettivo in mente, il presente lavoro si prefigge, come primo passo, l'obbiettivo di validare il ciclo di ottimizzazione su profili bidimensionali in regime laminare. Come software all'interno di cui implementare il solutore è stato scelto OpenFOAM, per la possibilità e la semplicità con cui è possibile modificare, adattando ai propri scopi, i solutori già esistenti. Come il resto degli strumenti utilizzati in questo lavoro, esso è liberamente scaricabile in rete, garantendo di poter sviluppare un procedimento completamente *open source*. Inoltre, tutto il processo di simulazione è stato sviluppato sulla piattaforma HPC di CINECA Galileo [4], cosa che, grazie alla possibilità di parallelizzazione dei processi, ha permesso sia la riduzione dei tempi di sviluppo del ciclo che dei tempi di calcolo finali.

La tesi è suddivisa nei seguenti capitoli:

- **Teoria**: partendo dalla equazioni di Navier-Stokes incomprimibili viscose vengono ricavate le rispettive equazioni aggiunte e viene esplicitata la relazione per il calcolo della sensitività superficiale. Il tutto viene successivamente specializzato al caso di ottimizzazione dei coefficienti aerodinamici di un profilo alare 2D.
- **Implementazione**: viene presentato il solutore aggiunto implementato in OpenFOAM, riportando i listati dei moduli più significativi.
- **Metodologia**: viene esposto il ciclo di ottimizzazione vero e proprio, andando ad approfondire i vari elementi che lo costituiscono.
- **Risultati**: vengono riportati i risultati ottenuti applicando il ciclo di ottimizzazione a vari regimi di numero di Reynolds e di angolo d'attacco. Per ogni configurazione viene effettuata una minimizzazione del coefficiente di resistenza e una massimizzazione del coefficiente di portanza.
- **Sviluppi futuri**: sono elencati i passi che logicamente dovrebbero seguire a questo lavoro.
- **Appendice**: nell'appendice A sono riportate tutte le strade che sono state intraprese nel tentativo di costruire una metodologia robusta ed efficiente e i problemi che si sono incontrati prima di raggiungere la configurazione finale.

Capitolo 2

Teoria

Nel seguente capitolo vengono presentate le equazioni di Navier-Stokes incomprimibili, instazionarie e viscose per un fluido newtoniano, distinguendo le diversità tra regime di moto laminare e turbolento. Partendo da esse vengono ricavate le relative equazioni aggiunte, attraverso un procedimento di integrazione per parti, per poi ricavare l'espressione della sensitività. Questi sviluppi seguono il lavoro proposto da Othmer nell'articolo [5].

2.1 Equazioni di Navier-Stokes

Il modello fisico di partenza di cui si vuole ricavare la controparte aggiunta, è rappresentato dalle equazioni di Navier-Stokes incomprimibili, stazionarie e viscose che sono descritte dal sistema:

$$\begin{cases} (\mathbf{u} \cdot \nabla)\mathbf{u}) + \nabla p - \nabla \cdot (2\nu \mathbf{D}(\mathbf{u})) = 0 \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \tag{2.1}$$

dove con $\mathbf{D}(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ viene rappresentato il tensore degli sforzi di deformazione, con $\mathbf{u}$ il vettore velocità e con p il rapporto tra pressione e densità.

Lo scopo del processo di ottimizzazione è quello di minimizzare (o massimizzare) una certa funzione obbiettivo $J(\beta, \mathbf{\Phi})$, che in generale dipenderà dalle variabili di progetto β e dal vettore delle soluzioni del flusso $\mathbf{\Phi} = [\mathbf{u}, p]^T$. Dato che le variabili $\mathbf{\Phi}$ sono soggette al vincolo di essere soluzione delle equazioni 2.1, è possibile fare uso di un classico metodo per trattare problemi di ottimizzazione vincolata: il metodo dei moltiplicatori di Lagrange. Viene quindi introdotta la funzione lagrangiana L, definita come la somma

tra la funzione obbiettivo e un termine proporzionale ai vincoli $\mathbf{R}$ tramite i moltiplicatori di lagrange $\boldsymbol{\lambda}$.

In questo caso i vincoli del problema sono le equazioni di stato, quindi:

$$\begin{cases} [R_1, R_2, R_3]^T = (\mathbf{u} \cdot \nabla)\mathbf{u}) + \nabla p - \nabla \cdot (2\nu \mathbf{D}(\mathbf{u})) \\ R_4 = \nabla \cdot \mathbf{u} \end{cases} \tag{2.2}$$

mentre devono essere introdotte due nuove grandezze che fungeranno da moltiplicatori di lagrange: la velocità aggiunta $\mathbf{v}$ e la pressione aggiunta q, per cui:

$$\boldsymbol{\lambda} = \begin{bmatrix} \mathbf{v} \\ q \end{bmatrix} \tag{2.3}$$

In questo modo la funzione lagrangiana diventa:

$$L(\beta, \boldsymbol{\Phi}, \boldsymbol{\lambda}) = J + \int_\Omega \boldsymbol{\lambda} \cdot \mathbf{R} \; d\Omega \tag{2.4}$$

L'approccio di ottimizzazione che si intende utilizzare è di tipo *gradient-based*: in questi metodi il minimo locale di una funzione viene individuato compiendo passi incrementali rispetto ad una configurazione di partenza nella direzione del gradiente della funzione obbiettivo rispetto alle variabili di progetto, che prende il nome di sensitività. Quindi è necessario caclolare la variazione totale della funzione lagrangiana, ottenuta dalla somma delle variazioni parziali rispetto ad ogni variabile da cui dipende:

$$\delta L = \delta_\beta L + \delta_{\boldsymbol{\Phi}} L + \delta_{\boldsymbol{\lambda}} L \tag{2.5}$$

Dato che la variazione di L rispetto ai moltiplicatori di Lagrange rappresenta le equazioni di stato, l'ultimo termine della 2.5 è identicamente nullo nel momento in cui viene valutato nella soluzione delle equazioni di Navier-Stokes.
Al contrario, il secondo termine $\delta_{\boldsymbol{\Phi}} L$ rappresenta l'elemento più gravoso per il calcolo della sensitività, in quanto richiederebbe di risolvere le equazioni di Navier-Stokes tante volte quante sono le variabili di progetto. Per superare questo inconveniente è possibile scegliere i moltiplicatori di Lagrange in modo che soddisfino la relazione:

$$\delta_{\boldsymbol{\Phi}} L = \delta_{\boldsymbol{\Phi}} J + \int_\Omega \boldsymbol{\lambda} \cdot \delta_{\boldsymbol{\Phi}} \mathbf{R} \; d\Omega = 0 \tag{2.6}$$

In questo modo il costo computazionale del calcolo della sensitività risulta indipendente dal numero di parametri di design β, e il suo calcolo si riduce alla determinazione del

primo termine in 2.5, cioè:

$$\delta L = \delta_\beta L = \delta_\beta J + \delta_\beta \int_\Omega [\mathbf{v}, q]^T \cdot \mathbf{R} \; d\Omega \tag{2.7}$$

Questa scelta si traduce nella formulazione di un set di equazioni per le quantità $\mathbf{v}$ e q, che prendono il nome di equazioni aggiunte.

2.2 Derivazione delle equazioni aggiunte

Ricordando che, per definizione, $\mathbf{\Phi} = [\mathbf{u}, p]^T$ l'equazione 2.6 può essere riscritta come:

$$\delta_\mathbf{u} J + \delta_p J + \int_\Omega [\mathbf{v}, q]^T \cdot \delta_\mathbf{u} \mathbf{R} \; d\Omega + \int_\Omega [\mathbf{v}, q]^T \cdot \delta_p \mathbf{R} \; d\Omega = 0 \tag{2.8}$$

Tenendo presente le relazioni in 2.2, le derivate di $\mathbf{R}$ possono essere facilmente calcolate come:

$$\delta_\mathbf{u} \mathbf{R} = \begin{bmatrix} (\delta\mathbf{u} \cdot \nabla)\mathbf{u} + (\mathbf{u} \cdot \nabla)\delta\mathbf{u} - \nabla \cdot (2\nu \mathbf{D}(\delta\mathbf{u})) \\ -\nabla \cdot \delta\mathbf{u} \end{bmatrix} \tag{2.9}$$

$$\delta_p \mathbf{R} = \begin{bmatrix} \nabla \delta p \\ 0 \end{bmatrix} \tag{2.10}$$

In questa derivazione si è trascurata la variazione della viscosità turbolenta, seguendo l'approssimazione nota come *frozen turbulence*. Questa semplificazione risulta effettivamente corretta solo nel caso di regime laminare in cui la viscosità è uniforme in tutto il dominio, mentre costituisce appunto un'approssimazione nel caso di regime turbolento, in cui la viscosità è funzione della posizione. Tuttavia, questa approssimazione viene generalmente accettata in letteratura e gli studi effettuati al riguardo, come ad esempio l'articolo [6] non evidenziano errori significativi dovuti all'introduzione di questa approssimazione.
A questo punto l'equazione 2.8 diventa:

$$\delta_\mathbf{u} J + \delta_p J + \int_\Omega \mathbf{v} \cdot \big((\delta\mathbf{u} \cdot \nabla)\mathbf{u} + (\mathbf{u} \cdot \nabla)\delta\mathbf{u} - \nabla \cdot (2\nu \mathbf{D}(\delta\mathbf{u})\big) - q\nabla \cdot \delta\mathbf{u} \; d\Omega + \int_\Omega \mathbf{v} \cdot \nabla \delta p \; d\Omega = 0 \tag{2.11}$$

A questo punto è utile suddividere la funzione costo J in un contributo sul bordo del dominio $\Gamma = \partial\Omega$ e in un contributo all'interno del dominio Ω, come:

$$J = \int_\Gamma J_\Gamma \; d\Gamma + \int_\Omega J_\Omega \; d\Omega \tag{2.12}$$

Sostituendo in 2.11 tale scomposizione e integrando per parti si ottiene:

$$\begin{aligned}
&\int_\Gamma d\Gamma\Big(\mathbf{v}\cdot\mathbf{n}+\frac{\partial J_\Gamma}{\partial p}\Big)\delta p+\int_\Omega d\Omega\Big(-\nabla\cdot\mathbf{v}+\frac{\partial J_\Omega}{\partial p}\Big)\delta p\\
&+\int_\Gamma d\Gamma\Big(\mathbf{n}(\mathbf{u}\cdot\mathbf{v})+\mathbf{v}(\mathbf{u}\cdot\mathbf{n})+2\nu\mathbf{n}\cdot\mathbf{D}(\mathbf{v})-q\mathbf{n}+\frac{\partial J_\Gamma}{\partial\mathbf{u}}\Big)\delta\mathbf{u}\\
&-\int_\Gamma d\Gamma 2\nu\mathbf{n}\cdot\mathbf{D}(\delta\mathbf{u})\cdot\mathbf{v}\\
&+\int_\Omega d\Omega\Big(-\nabla\mathbf{v}\cdot\mathbf{u}-(\mathbf{u}\cdot\nabla)\mathbf{v}+-\nabla\cdot\big(2\nu\mathbf{D}(\mathbf{v})\big)+\frac{\partial J_\Omega}{\partial\mathbf{u}}\Big)\delta\mathbf{u}=0
\end{aligned}\tag{2.13}$$

Questa equazione deve essere soddisfatta per ogni variazione $\delta\mathbf{u}$ e δp che soddisfi le equazioni di Navier-Stokes, condizione che viene verificata annullando gli integrandi dei singoli integrali. In questo modo dagli integrali sul dominio Ω dell'equazione 2.13, si ricavano le equazioni aggiunte delle equazioni di Navier-Stokes:

$$\begin{cases}
-2\mathbf{D}(\mathbf{v})\cdot\mathbf{u}=-\nabla q+\nabla\cdot\big(2\nu\mathbf{D}(\mathbf{v})\big)-\frac{\partial J_\Omega}{\partial\mathbf{u}}\\
\nabla\cdot\mathbf{v}=\frac{\partial J_\Omega}{\partial p}
\end{cases}\tag{2.14}$$

dove si è fatto uso della relazione: $-2\mathbf{D}(\mathbf{v})\cdot\mathbf{u}=-\nabla\mathbf{v}\cdot\mathbf{u}-(\mathbf{u}\cdot\nabla)\mathbf{v}$

Allo stesso modo le condizioni al contorno si ricavano dagli integrali sul bordo del dominio Γ:

$$\begin{cases}
\int_\Gamma d\Gamma\Big(\mathbf{n}(\mathbf{u}\cdot\mathbf{v})+\mathbf{v}(\mathbf{u}\cdot\mathbf{n})+2\nu\mathbf{n}\cdot\mathbf{D}(\mathbf{v})-q\mathbf{n}+\frac{\partial J_\Gamma}{\partial\mathbf{u}}\Big)\delta\mathbf{u}-\int_\Gamma d\Gamma 2\nu\mathbf{n}\cdot\mathbf{D}(\delta\mathbf{u})\cdot\mathbf{v}=0\\
\int_\Gamma d\Gamma\Big(\mathbf{v}\cdot\mathbf{n}+\frac{\partial J_\Gamma}{\partial p}\Big)\delta p=0
\end{cases}\tag{2.15}$$

Si nota subito che la struttura delle equazioni aggiunte ricorda molto quella delle equazioni dirette, a partire dalla linearità. Inoltre, nel caso la funzione costo J non dipenda da un integrale all'interno del dominio ($J_\Omega=0$), si ottiene che anche il campo vettoriale rappresentato dalla velocità aggiunta risulta a divergenza nulla, così come il campo di velocità diretta. La differenza fondamentale tra la struttura del problema diretto e aggiunto sta nel segno meno che precede il termine convettivo: questo sta ad indicare che le informazioni trasportate dal problema aggiunto fluiscono controcorrente rispetto alla direzione del moto del problema diretto.

2.3 Sensitività

Risolte le equazioni aggiunte nelle variabili **v** e q è possibile calcolare il valore della sensitività partendo dalla relazione 2.7. Il primo termine esprime la dipendenza della funzione obbiettivo rispetto alle variabili di design e può essere calcolato facilmente attraverso semplici considerazioni geometriche, mentre il secondo termine richiede di essere in grado di calcolare le perturbazioni indotte nel campo di moto diretto dalle perturbazioni geometriche delle variabili di progetto, oltre che alla risoluzione del sistema delle equazioni aggiunte.

Per fare ciò si segue la metodologia introdotta da Soto e Löhner in [7], che permette di riformulare l'integrale di volume in 2.7 in un integrale di superficie sul bordo del dominio, da cui è possibile calcolare il valore di sensitività di superficie, una volta note le grandezze dirette e aggiunte.

Ricordando che **R** rappresenta il sistema delle equazioni di Navier-Stokes, possiamo affermare che per ogni variazione ammissibile $\delta\mathbf{u}$ e δp, la variazione totale $\delta\mathbf{R}$ è identicamente nulla:

$$\delta\mathbf{R} = \delta_\beta\mathbf{R} + \delta_\mathbf{u}\mathbf{R} + \delta_p\mathbf{R} = 0 \tag{2.16}$$

Quindi si può riformulare la relazione per il calcolo della sensitività, sfruttando la relazione ricavata sopra:

$$\delta_\beta L = \delta_\beta J - \int_\Omega [\mathbf{v}, q]^T \cdot \delta_\mathbf{v}\mathbf{R}\ d\Omega - \int_\Omega [\mathbf{v}, q]^T \cdot \delta_p\mathbf{R}\ d\Omega \tag{2.17}$$

Gli ultimi due integrali dell'equazione sono già stati incontrati nel procedimento di derivazione delle equazioni aggiunte, quindi è possibile sfruttare i risultati ricavati nelle equazioni 2.9 e 2.10 ottenendo direttamente la relazione:

$$\begin{aligned}\delta_\beta L \quad = \quad & \delta_\beta J - \int_\Gamma d\Gamma\ \mathbf{v}\cdot\mathbf{n}\delta p + \int_\Omega d\Omega\ \nabla\cdot\mathbf{v}\delta p \\ & - \int_\Gamma d\Gamma \big(\mathbf{n}(\mathbf{u}\cdot\mathbf{v}) + \mathbf{v}(\mathbf{u}\cdot\mathbf{n}) + 2\nu\mathbf{n}\mathbf{D}(\mathbf{v} - q\mathbf{n})\big)\delta\mathbf{u} \\ & + \int_\Gamma d\Gamma\ 2\nu\mathbf{n}\cdot\mathbf{D}(\delta\mathbf{u})\cdot\mathbf{v} - \int_\Omega d\Omega\big(-\nabla\mathbf{v}\cdot\mathbf{u} - (\mathbf{u}\cdot\nabla)\mathbf{v} - \nabla\cdot(2\nu\mathbf{D}(\mathbf{v})) + \nabla q\big)\delta\mathbf{u}\end{aligned} \tag{2.18}$$

2.4 Specializzazione all'ottimizzazione di flussi esterni

I risultati ottenuti fino ad ora hanno una validità generale, in quanto non è stata ancora avanzata nessuna ipotesi sulla funzione costo o sulle condizioni al contorno imposte sulle variabili dirette. In questa sezione verranno riformulate le stesse equazioni specializzandole a un ottimizzazione aerodinamica attorno ad un profilo. In particolare, saranno ricavate le condizioni al contorno e la sensitività scegliendo come funzionale da minimizzare la componente di pressione del coefficiente di resistenza:

$$J = \frac{1}{\frac{1}{2}|\mathbf{u}_\infty|^3} \int_{\Gamma_{wall}} p\ \mathbf{n} \cdot \mathbf{u}_\infty\ d\Gamma \tag{2.19}$$

2.4.1 Condizioni al contorno

Generalmente, il tipico bordo del dominio di questi problemi è costituito da tre diverse zone: inlet, outlet e superficie solida (wall), a cui vengono associate delle specifiche condizioni al contorno di velocità e pressione. È un approccio comune fissare l'andamento della velocità imponendo un gradiente nullo di pressione all'inlet, fissare un valore di pressione con gradiente di velocità nullo all'outlet mentre in corrispondenza del profilo si considera una condizione di *no-slip* per la velocità e di gradiente nullo per la pressione. In questo sistema, le funzioni costo che tipicamente vengono scelte sono formulate come un integrale lungo i bordi del dominio, quindi la componente J_Ω dipendente dall'interno del dominio è nulla. In questo modo, le equazioni aggiunte e le relative condizioni al contorno possono essere semplificate nella forma:

$$\begin{cases} -2\mathbf{D}(\mathbf{v}) \cdot \mathbf{u} = -\nabla q + \nabla \cdot \big(2\nu \mathbf{D}(\mathbf{v})\big) \\ \nabla \cdot \mathbf{v} = 0 \end{cases} \tag{2.20}$$

Il grande vantaggio di questa semplificazione è che le equazioni aggiunte diventano indipendenti dalla funzione costo J, quindi se sarà necessario cambiare il funzionale da ottimizzare il solutore aggiunto non dovrà subire modifiche, l'unico cambiamento sarà sulle condizioni al contorno da applicare alle variabili aggiunte.

Nella riformulazione delle condizioni al contorno 2.15 è possibile sfruttare il fatto che sia la velocità che la velocità aggiunta sono campi vettoriali a divergenza nulla, ottenendo le due equazioni:

$$\int_\Gamma d\Gamma \big(\mathbf{n}(\mathbf{v}\cdot\mathbf{u}) + \mathbf{v}(\mathbf{u}\cdot\mathbf{n}) + \nu(\mathbf{n}\cdot\nabla)\mathbf{v} - q\mathbf{n} + \frac{\partial J_\Gamma}{\partial \mathbf{u}}\big) \cdot \delta\mathbf{u} - \int_\Gamma d\Gamma\ \nu(\mathbf{n}\cdot\nabla)\delta\mathbf{u}\cdot\delta\mathbf{v} = 0 \tag{2.21}$$

$$\int_{\Gamma} d\Gamma \big(\mathbf{v}\cdot\mathbf{n} + \frac{\partial J_{\Gamma}}{\partial p}\big)\delta p = 0 \tag{2.22}$$

Partendo da queste, nel seguito verranno ricavate le specifiche condizioni al contorno per inlet, outlet e wall.

2.4.1.1 Inlet

Come già specificato in precedenza, su questa superficie la velocità è solitamente specificata, quindi il primo integrale dell'equazione 2.21 è identicamente nullo. Dal fatto che $\delta\mathbf{u}$ è un campo a divergenza nulla si ricava:

$$\nabla\cdot\delta\mathbf{u} = (\mathbf{n}\cdot\nabla)\delta u_n + \nabla_{||}\cdot\delta\mathbf{u}_t = 0 \tag{2.23}$$

dove u_n e $\mathbf{u}_t$ rappresentano rispettivamente la componente normale e tangenziale del vettore $\mathbf{u}$, mentre $\nabla_{||}$ rappresenta la componente tangenziale dell'operatore gradiente. Dato che $\delta\mathbf{u}_t$ è nullo all'inlet, l'equazione precedente implica che $(\mathbf{n}\cdot\nabla)\delta u_n = 0$, da cui $(\mathbf{n}\cdot\nabla)\delta\mathbf{u} = (\mathbf{n}\cdot\nabla)\delta\mathbf{u}_t$. In conclusione, la condizione al contorno 2.21 si riduce a:

$$\int_{\Gamma} \nu(\mathbf{n}\cdot\nabla)\delta\mathbf{u}_t\cdot\mathbf{v}_t \quad d\Gamma = 0 \tag{2.24}$$

che a sua volta equivale ad imporre:

$$\mathbf{v}_t = 0 \tag{2.25}$$

Invece, la seconda condizione al contorno 2.22, tenendo conto che il funzionale J non ha nessuna dipendenza dall'inlet diventa semplicemente:

$$\mathbf{v}\cdot\mathbf{n} = \frac{\partial J_{\Gamma_{inlet}}}{\partial p} = 0 \tag{2.26}$$

Le due relazioni così ottenute affermano che all'inlet la velocità aggiunta deve essere fissata a valore nullo.
Si nota che queste condizioni al contorno non forniscono nessuna informazione sul valore di q al bordo. Data la sua dualità con la pressione per il problema aggiunto, allo stesso modo verrà imposto un gradiente nullo.

2.4.1.2 Wall

Anche sulla parete solida del profilo rimane valido il ragionamento che ha portato alla relazione 2.24, quindi persiste la condizione:

$$\mathbf{v}_t = 0 \tag{2.27}$$

Diversamente da quanto osservato in precedenza, il funzionale J è espresso come un integrale proprio sulla superficie del profilo, quindi la relazione per la componente normale della velocità aggiunta sarà non nulla e uguale a:

$$\mathbf{v} \cdot \mathbf{n} = -\frac{\partial J_{\Gamma_{wall}}}{\partial p} = -\frac{1}{\frac{1}{2}|\mathbf{u}_\infty|^3}\mathbf{n} \cdot \mathbf{u}_\infty \tag{2.28}$$

Come nel caso precedente la condizione al contorno per la pressione aggiunta è quella di gradiente nullo.

2.4.1.3 Outlet

All'outlet la condizione al contorno per la velocità è di gradiente nullo, quindi il secondo integrale di 2.21 si annulla. Uguagliando a 0 l'integrando del rimanente integrale si ottiene:

$$\mathbf{n}(\mathbf{v} \cdot \mathbf{u}) + \mathbf{v}(\mathbf{u} \cdot \mathbf{n}) + \nu(\mathbf{n} \cdot \nabla)\mathbf{v} - q\mathbf{n} + \frac{\partial J_\Gamma}{\partial \mathbf{u}} = 0 \tag{2.29}$$

La componente normale di questa equazione fornisce la condizione al contorno per la pressione aggiunta:

$$q = \mathbf{v} \cdot \mathbf{u} + v_n u_n + \nu(\mathbf{n} \cdot \nabla)v_n \tag{2.30}$$

in cui si è tenuto conto che la funzione costo non dipende dalla superficie dell'outlet. Invece, la componente tangenziale dell'equazione fornisce la condizione al contorno per la grandezza $\mathbf{v}_t$:

$$u_n \mathbf{v}_t + \nu(\mathbf{n} \cdot \nabla)\mathbf{v}_t = 0 \tag{2.31}$$

che viene completata dalla relazione ricavabile dall'equazione 2.22, che in questo caso diventa:

$$\mathbf{v} \cdot \mathbf{n} = 0 \tag{2.32}$$

2.4.2 Sensitività

Per il calcolo della sensitività è possibile riprendere il risultato ottenuto nell'equazione 2.18 e constatare che, nel caso in cui effettivamente non ci sia un contributo del volume

interno del dominio nella funzione costo J, gli integrali su Ω si annullano per garantire il soddisfacimento delle equazioni aggiunte.
Il calcolo della sensitività si riduce quindi a:

$$\delta_\beta L = \delta_\beta J - \int_\Gamma d\Gamma\ \mathbf{v}\cdot\mathbf{n}\delta p - \int_\Gamma d\Gamma \big(\mathbf{n}(\mathbf{u}\cdot\mathbf{v}) + \mathbf{v}(\mathbf{u}\cdot\mathbf{n}) + 2\nu\mathbf{n}\mathbf{D}(\mathbf{v}-q\mathbf{n})\big)\cdot\delta\mathbf{u} + \int_\Gamma d\Gamma\ 2\nu\mathbf{n}\cdot\mathbf{D}(\delta\mathbf{u})\cdot\mathbf{v} \tag{2.33}$$

Tuttavia, questa forma non è può essere utilizzata direttamente per il calcolo della sensitività in quanto le variazioni delle variabili dirette $\delta\mathbf{u}$ e δp non sono note a priori. Dato che le variabili di progetto β sono costituite dagli spostamenti dei punti del profilo in direzione normale alla superficie, è ragionevole approssimare le variazioni delle grandezze in questione con una espansione in serie di Taylor arrestata al primo ordine:

$$\begin{cases} \delta\mathbf{u} = \beta(\mathbf{n}\cdot\nabla)\mathbf{u} \\ \delta p = \beta(\mathbf{n}\cdot\nabla)p \end{cases} \tag{2.34}$$

Quindi è possibile riscrivere la relazione 2.33 come:

$$\delta_\beta L = \delta_\beta J - \int_\Gamma d\Gamma \mathbf{v}\cdot\mathbf{n}(\mathbf{n}\cdot\nabla)p - \int_\Gamma d\Gamma \big(\mathbf{n}(\mathbf{u}\cdot\mathbf{v}) + \mathbf{v}(\mathbf{u}\cdot\mathbf{n}) + 2\nu\mathbf{n}\mathbf{D}(\mathbf{v}-q\mathbf{n})\big)\cdot(\mathbf{n}\cdot\nabla)\mathbf{u} + \int_\Gamma 2\nu\mathbf{n}\cdot\mathbf{D}(\mathbf{n}\cdot\nabla)\mathbf{u})\cdot\mathbf{v} \tag{2.35}$$

Ricordando che la sensitività necessita di essere calcolata solo sulla superficie del profilo e richiamando le condizioni al contorno ricavate per questa superficie:

$$\begin{cases} \mathbf{u} = 0 \\ \frac{\partial p}{\partial \mathbf{n}} = 0 \\ \mathbf{v}_t = 0 \\ \frac{\partial q}{\partial \mathbf{n}} = 0 \\ \mathbf{v}\cdot\mathbf{n} = -\frac{1}{\frac{1}{2}|\mathbf{u}_\infty|}\mathbf{n}\cdot\mathbf{u}_\infty \end{cases} \tag{2.36}$$

L'equazione 2.35 si riduce ai termini:

$$\delta_\beta L = \delta_\beta J - \int_\Gamma d\Gamma \big(2\nu\mathbf{n}\mathbf{D}(\mathbf{v}-q\mathbf{n})\big)\cdot(\mathbf{n}\cdot\nabla)\mathbf{u} + \int_\Gamma d\Gamma\ 2\nu\mathbf{n}\cdot\mathbf{D}(\mathbf{n}\cdot\nabla)\mathbf{u})\cdot\mathbf{v} \tag{2.37}$$

Dalla definizione di J in 2.19, è possibile derivare il primo termine dell'equazione precedente:

$$\delta_\beta J = \nabla p \cdot \frac{\mathbf{u}_\infty}{\frac{1}{2}|\mathbf{u}_\infty|^3} = \frac{\nabla p \cdot \mathbf{d}}{\frac{1}{2}|\mathbf{u}_\infty|^2} \tag{2.38}$$

in cui il vettore $\mathbf{d}$ rappresenta la direzione della forza aerodinamica che si desidera ottimizzare.

Per esplicitare il calcolo dei rimanenti integrali è necessario considerare la relazione:

$$2\mathbf{n} \cdot \mathbf{D}(\mathbf{v}) = (\mathbf{n} \cdot \nabla)\mathbf{n} + \nabla(\mathbf{v} \cdot \mathbf{v}) \tag{2.39}$$

In questo modo il secondo integrale di 2.37 diventa:

$$\int_\Gamma d\Gamma \; 2\nu \mathbf{n} \cdot \mathbf{D}(\mathbf{v}) \cdot (\mathbf{n} \cdot \nabla)\mathbf{u} = \int_\Gamma d\Gamma \; \nu(\mathbf{n} \cdot \nabla)\mathbf{v} \cdot (\mathbf{n} \cdot \nabla)\mathbf{u} = \int_\Gamma d\Gamma \; \nu \frac{\partial \mathbf{v}}{\partial \mathbf{n}} \cdot \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \tag{2.40}$$

dove si è fatto uso della notazione $\frac{\partial \mathbf{u}}{\partial \mathbf{n}}$ per indicare la derivata di $\mathbf{u}$ in direzione normale.

Infine è possibile riscrivere l'ultimo integrale di 2.37 nel seguente modo:

$$\int_\Gamma d\Gamma \; 2\nu \mathbf{n} \cdot \mathbf{D}\big((\mathbf{n} \cdot \nabla)\mathbf{u}\big) \cdot \mathbf{v} = \int_\Gamma d\Gamma \; \nu \frac{\partial^2 \mathbf{u}}{\partial \mathbf{n}^2} \cdot \mathbf{v} = -\int_\Gamma d\Gamma \; \nu \frac{\partial^2 \mathbf{u}}{\partial \mathbf{n}^2} \cdot \frac{\mathbf{d}}{\frac{1}{2}\mathbf{u}_\infty^2} \tag{2.41}$$

Dove si è utilizzato la condizione al contorno sul profilo per la velocità aggiunta in 2.36.

Inserendo le relazioni trovate nell'equazione di partenza 2.37 si ottiene l'espressione finale per il calcolo della sensitività:

$$\delta_\beta L = \int_\Gamma \Big(\frac{\nabla p \cdot \mathbf{d}}{\frac{1}{2}|\mathbf{u}_\infty|^2} - \nu \frac{\partial \mathbf{v}}{\partial \mathbf{n}} \cdot \frac{\partial \mathbf{u}}{\partial \mathbf{n}} + q\mathbf{n} \cdot \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - \nu \frac{\partial^2 \mathbf{u}}{\partial \mathbf{n}^2} \cdot \frac{\mathbf{d}}{\frac{1}{2}\mathbf{u}_\infty^2} \Big) d\Gamma \tag{2.42}$$

Concludendo, bisogna osservare che, in accordo con la convenzione di scegliere le normali come entranti nel profilo, ad un valore positivo di sensitività corrisponde uno spostamento normale entrante del profilo e, viceversa, uno spostamento normale uscente dal profilo per valori di sensitività negativa.

Capitolo 3

Implementazione

Il solutore delle equazioni aggiunte di Navier-Stokes è stato implementato all'interno di OpenFOAM [8]. OpenFOAM è un software CFD *open source* ai volumi finiti che, negli ultimi anni, ha acquistato sempre maggiore successo sia in ambito accademico che in ambito industriale. La semplicità con cui possono essere implementate le equazioni differenziali e la facilità con cui è possibile adattare solutori già esistenti alle proprie esigenze, sono stati i due fattori principali in questa scelta. Questi aspetti sono resi possibili dal linguaggio di programmazione utilizzato, il C++, e da una spinta programmazione ad oggetti, che ha consentito agli sviluppatori di definire una serie di *classi* che rendono più flessibile e adattabile alle proprie esigenze il codice esistente.

3.1 Solutore aggiunto

Come punto di partenza per l'implementazione del nuovo solutore è stato utilizzato il codice del solver *adjointShapeOptimizationFoam*, scritto da C. Othmer, già presente nella distribuzione standard di OpenFOAM e finalizzato all'ottimizzazione topologica di flussi interni. Alcune informazioni sull'implementazioni di questo solutore vengono fornite in [9].

A sua volta, *adjointShapeOptimizationFoam*, ripropone lo schema utilizzato dal solutore delle equazioni di Navier-Stokes incomprimibili *simpleFoam*, che si basa su un algoritmo di tipo SIMPLE (Semi-Implicit Method for Pressure-Linked Equations). Tale algoritmo risolve le equazioni di Navier-Stokes incomprimibili seguendo un'approccio *segregato*: viene, cioè, sfruttato il fatto che la seconda equazione del sistema riportato in 2.1, più che un'equazione dinamica, rappresenta un vincolo cinematico, quindi il campo di pressione può essere calcolato iterativamente in modo che il campo di velocità soddisfi

tale vincolo di incomprimibilità.
Nello specifico, partendo dalle condizioni iniziali per velocità e pressione, per prima cosa viene calcolato un campo di velocità utilizzando l'equazione per la quantità di moto come nella prima equazione del sistema 2.1 (*predictor*); successivamente, calcolando la divergenza dell'equazione della quantità di moto si ottiene un'equazione di Poisson per la pressione, che può essere risolta con i metodi per le equazioni ellittiche; infine la soluzione del campo di velocità viene corretta in modo da soddisfare il vincolo di incomprimibilità (*corrector*).

Dato che solutore diretto e solutore aggiunto sono basati sullo stesso algoritmo e problema diretto e aggiunto coinvolgono gli stessi operatori differenziali, è possibile applicare gli stessi schemi di discretizzazione per i due problemi.

Ad eccezione dello schema risolutivo, il nuovo solutore implementato presenta delle diversità sostanziali rispetto al solutore introdotto da Othmer.
Per prima cosa, il solutore implementato per questo lavoro può essere utilizzato sia per flussi esterni che per flussi interni. La differenza, in questo caso, sta nel fatto che la logica all'interno del quale il solutore *adjointShapeOptimizationFoam* è stato sviluppato è quella dell'ottimizzazione topologica. Questa filosofia si basa sull'introduzione di una grandezza, detta porosità, che in base al valore assunto cella per cella, ha il compito di modificare la topologia del dominio interessato: se il valore di porosità in una certa zona è elevato, quella porzione di dominio dovrà comportarsi come una parete solida. In questo modo, attraverso successive modifiche della topologia del dominio, ci si aspetta di convergere verso una geometria ottimizzata. Al contrario, l'approccio seguito in questa tesi è quello di un'ottimizzazione di forma: in questo caso l'output del solutore aggiunto è costituito dalla sensitività superficiale, che rappresenta lo spostamento dei punti del corpo in esame in direzione normale alla superficie stessa. In questo modo la configurazione successiva si ottiene da una deformazione della configurazione precedente, senza modificare la topologia del dominio.

In più, mentre il solutore già esistente è in grado di funzionare solo con la funzione costo con cui è stato implementato, il solutore in esame è in grado di gestire sia una massimizzazione che una minimizzazione di una forza aerodinamica orientata arbitrariamente. Ciò è possibile modificando a piacimento il dizionario chiamato *ptimizationProperties*, la cui struttura verrà introdotta nella sezione successiva.

Infine, nel solutore scritto da Othmer, equazioni dirette e aggiunte vengono risolte insieme ad ogni iterazione dell'algoritmo, mentre nello sviluppo del nuovo solutore si è deciso di disaccoppiare la soluzione del problema diretto da quella del problema aggiunto. Secondo questa filosofia di implementazione, un ciclo completo per la soluzione diretto-aggiunto segue il seguente schema:

- partendo dalle condizioni iniziali viene lanciato il solver diretto;
- raggiunta la convergenza del diretto, le soluzioni di velocità e pressione vengono congelate e utilizzate come condizioni iniziali per il solver aggiunto;
- alla fine del ciclo viene calcolata la sensitività superficiale.

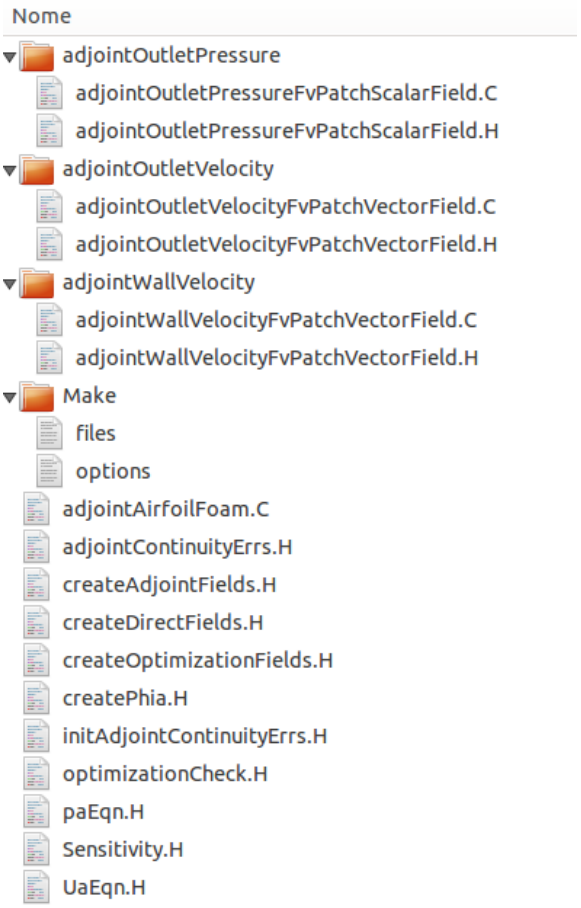
Questo procedimento risulta particolarmente vantaggioso, dal punto di vista del costo computazionale, nel caso in cui sia necessario risolvere più volte il sistema aggiunto con funzioni costo diverse, come ad esempio avviene per ottimizzazioni multiobbiettivo o ottimizzazioni vincolate. Infatti, ipotizzando che la risoluzione del problema diretto e del problema aggiunto abbiano costo uguale e unitario, il costo totale per risolvere un problema con N funzioni costo diverse con il metodo proposto sarebbe di N+1, mentre lo stesso problema avrebbe un costo di 2N tenendo accoppiati la risoluzione dei due set di equazioni.

Ora che sono state evidenziate le principali caratteristiche del solutore, nel prossimo capitolo vengono spiegate nel dettaglio tutte le sue componenti.

3.2 Implementazione del solutore

La struttura del solutore *adjointOptFoam* è schematizzata in Figura 3.1. Il file *adjointAirfoilFoam.C* rappresenta il file *main* e al suo interno vengono richiamati i vari *header*, a ciascuno dei quali corrisponde un'azione distinta: *createDirectField.H*, *createAdjointField.H*, *createOptimizationField.H* e *createPhia.H* servono ad inizializzare tutte le grandezze che verranno utilizzate nel solutore; *initAdjointContinuityErrs.H* inizializza i *continuity errors* della soluzione aggiunta, che verranno poi aggiornati ad ogni iterazione del solutore dal file *adjointContinuityErrs.H*; l'*header optimizationCheck.H* ha una funzione di *error handling*, cioè controlla che i dizionari e i file di input forniti dall'utente non contengano errori e le parole chiave siano coerenti con quelle che si aspetta il solver; all'interno di *UaEqn.H* e *paEqn.H* vengono risolte le equazioni per ricavare rispettivamente la velocità aggiunta e la pressione aggiunta; infine in *Sensitivity.H* viene calcolata la sensitività. Le tre cartelle *adjointOutletPressure*, *adjointOutletVelocity* e *adjointWallVelocity* contengono i file in cui vengono definite le condizioni al contorno in corrispondenza di outlet e wall. All'interno delle semplificazioni in cui ci siamo posti, in cui la funzione costo J non presenta una dipendenza dall'interno del dominio Ω, l'implementazione delle condizioni al contorno rappresenta l'unico elemento che è necessario modificare nel momento in cui sia ha l'esigenza di modificare la funzione costo, mentre il resto del solutore rimane invariato.

FIGURA 3.1: Struttura del solutore *adjointOptFoam*



Di seguito vengono descritti singolarmente i file principali, seguiti dal loro listato:

adjointAirfoilFoam.C : file principale in cui viene gestito il ciclo del solver e vengono richiamati i vari *header*. Nella prima parte del codice vengono inizializzate tutte le grandezze che verranno coinvolte, dopo di che inizia il loop all'interno di cui, ad ogni iterazione, viene risolta l'equazione per la velocità aggiunta $\mathbf{U}_a$ e l'equazione per la pressione aggiunta p_a. Una volta raggiunta la convergenza si esce dal ciclo e viene calcolata la sensitività.

```
1  /*---------------------------------------------------------------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4     \\    /   O peration     |
5      \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
6       \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
```

```
13      the Free Software Foundation, either version 3 of the License, or
14      (at your option) any later version.
15
16      OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17      ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18      FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
19      for more details.
20
21      You should have received a copy of the GNU General Public License
22      along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
23
24  Application
25      airfoilAdjointFoam
26
27  Description
28      Steady-state incompressible frozen viscosity adjoint solver.
29
30  \*---------------------------------------------------------------------------*/
31
32  #include "fvCFD.H"
33  #include "singlePhaseTransportModel.H"
34  #include "RASModel.H"
35  #include "simpleControl.H"
36
37  template<class Type>
38  void zeroCells
39  (
40      GeometricField<Type, fvPatchField, volMesh>& vf,
41      const labelList& cells
42  )
43  {
44      forAll(cells, i)
45      {
46          vf[cells[i]] = pTraits<Type>::zero;
47      }
48  }
49
50
51  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
52
53  int main(int argc, char *argv[])
54  {
55      #include "setRootCase.H"
56
57      #include "createTime.H"
58      #include "createMesh.H"
59      #include "createDirectFields.H"
60      #include "createAdjointFields.H"
61      #include "createOptimizationFields.H"
62      #include "initAdjointContinuityErrs.H"
63      #include "optimizationCheck.H"
64
65      simpleControl simple(mesh);
66
67      // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
68
69      Info<< "\nStarting time loop\n" << endl;
70
71      while (simple.loop())
72      {
73          Info<< "Time = " << runTime.timeName() << nl << endl;
74
75          // Adjoint Pressure-velocity SIMPLE corrector
76          #include "UaEqn.H"
77          #include "paEqn.H"
78
79          // Sensitivity calculation
80          #include "Sensitivity.H"
81
82          runTime.write();
83
84          //write adjoint surface sensitivity
85          Info << "\nWriting surface sensitivity\n" << endl;
86
87
88          Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
89          << "  ClockTime = " << runTime.elapsedClockTime() << " s"
90          << nl << endl;
91      }
92
93  //      #include "Sensitivity.H"
94
95      Info<< "End\n" << endl;
96
97      return(0);
98  }
99
100
101 // ************************************************************************* //
```

adjointAirfoilFoam.C

UaEqn.H : nel file seguente viene risolta la prima equazione del sistema aggiunto 2.14 per $\mathbf{U}_a$ e viene applicata una condizione di sottorilassamento per ottenere il valore di velocità aggiunta all'iterazione successiva.

```
1  // Adjoint Momentum predictor
2
3              volVectorField adjointTransposeConvection((fvc::grad(Ua) & U));
4
5              zeroCells(adjointTransposeConvection, inletCells);
6
7              tmp<fvVectorMatrix> UaEqn
8              (
9                  fvm::div(-phi, Ua)
10               - adjointTransposeConvection
11               + turbulence->divDevReff(Ua)
12             );
13
```

```
14              UaEqn().relax();
15
16              solve(UaEqn() == -fvc::grad(pa));
```

UaEqn.H

paEqn.H : successivamente, si passa alla soluzione dell'equazione per la pressione aggiunta, ottenuta applicando la divergenza all'equazione precedente. Dopo aver rilassato il valore di p_a ottenuto viene applicato un *corrector* sulla velocità aggiunta che impone ad $\mathbf{U}_a$ il vincolo di incomprimibilità.

```
1  // Pressure corrector
2  {
3  pa.boundaryField().updateCoeffs();
4               volScalarField rAUa(1.0/UaEqn().A());
5               Ua = rAUa*UaEqn().H();
6               UaEqn.clear();
7               phia = fvc::interpolate(Ua) & mesh.Sf();
8               adjustPhi(phia, Ua, pa);
9
10              // Non-orthogonal pressure corrector loop
11              while (simple.correctNonOrthogonal())
12              {
13                  fvScalarMatrix paEqn
14                  (
15                      fvm::laplacian(rAUa, pa) == fvc::div(phia)
16                  );
17
18                  paEqn.setReference(paRefCell, paRefValue);
19                  paEqn.solve();
20
21                  if (simple.finalNonOrthogonalIter())
22                  {
23                      phia -= paEqn.flux();
24                  }
25              }
26
27              #include "adjointContinuityErrs.H"
28
29              // Explicitly relax pressure for adjoint momentum corrector
30              pa.relax();
31
32              // Adjoint momentum corrector
33              Ua -= rAUa*fvc::grad(pa);
34              Ua.correctBoundaryConditions();
35 }
```

paEqn.H

Sensitivity.H : ottenuti i valori di $\mathbf{U}_a$ e p_a a convergenza, si calcola il valore di sensitività sulla superficie del profilo secondo la relazione 2.42. Estraendo il verso dell'ottimizzazione (massimizzazione o minimizzazione) dal dizionario *optimizationProperties*, viene

scelto il segno di sensitività.

```
// Sensitivity
forAll(wallGradU.boundaryField(), patchi)
{
        if(mesh.boundary()[patchi].type() == "wall")
        {
                wallGradU.boundaryField()[patchi] ==
                -U.boundaryField()[patchi].snGrad();
                wallGradUa.boundaryField()[patchi] ==
                -Ua.boundaryField()[patchi].snGrad();
                wallGrad2U.boundaryField()[patchi] ==
                -wallGradU.boundaryField()[patchi].snGrad();
                normal.boundaryField()[patchi] ==
                mesh.Sf().boundaryField()[patchi]/mesh.magSf().boundaryField()[
    patchi];
        }
}

volScalarField nuEff = turbulence->nuEff()();

wallSens == (d & fvc::grad(p))/C_inf - nuEff * (wallGradUa & wallGradU) - nuEff/
    C_inf * (wallGrad2U & d) + pa * (normal & wallGradU);

if (optimization == "maximize") {
        wallSens == -wallSens;
}
```

Sensitivity.H

adjointOutletPressureFvPatchScalarField.C : nel file seguente viene imposta la condizione al contorno sulla pressione all'outlet secondo la relazione 2.30.

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
```

```
23
24  \*---------------------------------------------------------------------------*/
25
26  #include "adjointCdOutletPressureFvPatchScalarField.H"
27  #include "RASModel.H"
28  #include "addToRunTimeSelectionTable.H"
29  #include "fvPatchMapper.H"
30  #include "volFields.H"
31  #include "surfaceFields.H"
32
33  // * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * * //
34
35  Foam::adjointCdOutletPressureFvPatchScalarField::
36  adjointCdOutletPressureFvPatchScalarField
37  (
38      const fvPatch& p,
39      const DimensionedField<scalar, volMesh>& iF
40  )
41  :
42      fixedValueFvPatchScalarField(p, iF)
43  {}
44
45
46  Foam::adjointCdOutletPressureFvPatchScalarField::
47  adjointCdOutletPressureFvPatchScalarField
48  (
49      const adjointCdOutletPressureFvPatchScalarField& ptf,
50      const fvPatch& p,
51      const DimensionedField<scalar, volMesh>& iF,
52      const fvPatchFieldMapper& mapper
53  )
54  :
55      fixedValueFvPatchScalarField(ptf, p, iF, mapper)
56  {}
57
58
59  Foam::adjointCdOutletPressureFvPatchScalarField::
60  adjointCdOutletPressureFvPatchScalarField
61  (
62      const fvPatch& p,
63      const DimensionedField<scalar, volMesh>& iF,
64      const dictionary& dict
65  )
66  :
67      fixedValueFvPatchScalarField(p, iF)
68  {
69      fvPatchField<scalar>::operator=
70      (
71          scalarField("value", dict, p.size())
72      );
73  }
74
75
76  Foam::adjointCdOutletPressureFvPatchScalarField::
77  adjointCdOutletPressureFvPatchScalarField
```

```
78  (
79      const adjointCdOutletPressureFvPatchScalarField& tppsf,
80      const DimensionedField<scalar, volMesh>& iF
81  )
82  :
83      fixedValueFvPatchScalarField(tppsf, iF)
84  {}
85
86
87  // * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //
88
89  void Foam::adjointCdOutletPressureFvPatchScalarField::updateCoeffs()
90  {
91      if (updated())
92      {
93          return;
94      }
95
96      const fvsPatchField<scalar>& phip =
97          patch().lookupPatchField<surfaceScalarField, scalar>("phi");
98
99      const fvsPatchField<scalar>& phiap =
100         patch().lookupPatchField<surfaceScalarField, scalar>("phia");
101
102     const fvPatchField<vector>& Up =
103         patch().lookupPatchField<volVectorField, vector>("U");
104
105     const fvPatchField<vector>& Uap =
106         patch().lookupPatchField<volVectorField, vector>("Ua");
107
108     //first get nuEff
109     const incompressible::RASModel& ras
110     = db().lookupObject<incompressible::RASModel>("RASProperties");
111
112     scalarField nuw
113     = ras.nuEff()().boundaryField()[patch().index()];
114
115     //patch normal vector
116     vectorField np = patch().Sf()/patch().magSf();
117
118     //patch-adjacent normal adjoint velocity Uac_n
119     scalarField Uac_n = (Uap.patchInternalField() & np);
120
121     //patch normal adjoint velocity Uap_n
122     scalarField Uap_n = phiap/patch().magSf();
123
124     //patch deltas
125     const scalarField& delta = patch().deltaCoeffs();
126
127     //snGrad Ua_n
128     scalarField snGradUan = delta*(Uap_n - Uac_n);
129
130     // Eqn.(54) in Othmer[2008]
131
132     operator==(
```

```
133                 (Up & Uap)
134                 + (phiap/patch().magSf())*(phip/patch().magSf())
135                 + nuw*snGradUan
136                 );
137
138
139     fixedValueFvPatchScalarField::updateCoeffs();
140 }
141
142
143 void Foam::adjointCdOutletPressureFvPatchScalarField::write(Ostream& os) const
144 {
145     fvPatchScalarField::write(os);
146     writeEntry("value", os);
147 }
148
149
150 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
151
152 namespace Foam
153 {
154     makePatchTypeField
155     (
156         fvPatchScalarField,
157         adjointCdOutletPressureFvPatchScalarField
158     );
159 }
160
161 // ************************************************************************* //
```

adjointOutletPressureFvPatchScalarField.C

adjointOutletVelocityFvPatchVectorField.C : le condizioni al contorno sulla patch di outlet sono completate con la condizione al contorno sulla componente tangenziale della velocità aggiunta così come indicato in 2.31.

```
1  /*--------------------------------------------------------------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4     \\    /   O peration     |
5      \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
6       \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
13     the Free Software Foundation, either version 3 of the License, or
14     (at your option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
```

```
19      for more details.
20
21      You should have received a copy of the GNU General Public License
22      along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
23
24  \*---------------------------------------------------------------------------*/
25
26  #include "adjointCdOutletVelocityFvPatchVectorField.H"
27  #include "volFields.H"
28  #include "RASModel.H"
29  #include "addToRunTimeSelectionTable.H"
30  #include "surfaceFields.H"
31  #include "fvPatchFieldMapper.H"
32
33  // * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * * //
34
35  Foam::adjointCdOutletVelocityFvPatchVectorField::
36  adjointCdOutletVelocityFvPatchVectorField
37  (
38      const fvPatch& p,
39      const DimensionedField<vector, volMesh>& iF
40  )
41  :
42      fixedValueFvPatchVectorField(p, iF)
43  {}
44
45
46  Foam::adjointCdOutletVelocityFvPatchVectorField::
47  adjointCdOutletVelocityFvPatchVectorField
48  (
49      const fvPatch& p,
50      const DimensionedField<vector, volMesh>& iF,
51      const dictionary& dict
52  )
53  :
54      fixedValueFvPatchVectorField(p, iF)
55  {
56      fvPatchVectorField::operator=(vectorField("value", dict, p.size()));
57  }
58
59
60  Foam::adjointCdOutletVelocityFvPatchVectorField::
61  adjointCdOutletVelocityFvPatchVectorField
62  (
63      const adjointCdOutletVelocityFvPatchVectorField& ptf,
64      const fvPatch& p,
65      const DimensionedField<vector, volMesh>& iF,
66      const fvPatchFieldMapper& mapper
67  )
68  :
69      fixedValueFvPatchVectorField(ptf, p, iF, mapper)
70  {}
71
72
73  Foam::adjointCdOutletVelocityFvPatchVectorField::
```

```
74  adjointCdOutletVelocityFvPatchVectorField
75  (
76      const adjointCdOutletVelocityFvPatchVectorField& pivpvf,
77      const DimensionedField<vector, volMesh>& iF
78  )
79  :
80      fixedValueFvPatchVectorField(pivpvf, iF)
81  {}
82
83
84  // * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //
85
86  // Update the coefficients associated with the patch field
87  void Foam::adjointCdOutletVelocityFvPatchVectorField::updateCoeffs()
88  {
89      if (updated())
90      {
91          return;
92      }
93
94      const fvsPatchField<scalar>& phiap =
95          patch().lookupPatchField<surfaceScalarField, scalar>("phia");
96
97      const fvPatchField<vector>& Uap =
98      patch().lookupPatchField<volVectorField, vector>("Ua");
99
100
101     //first get nuEff
102     const incompressible::RASModel& ras
103     = db().lookupObject<incompressible::RASModel>("RASProperties");
104
105     scalarField nuw
106     = ras.nuEff()().boundaryField()[patch().index()];
107
108     //patch normal vector
109     vectorField np = patch().Sf()/patch().magSf();
110
111     //patch deltas
112     const scalarField& delta = patch().deltaCoeffs();
113
114     //patch normal adjoint velocity Uap_n
115     scalarField Uap_n = phiap/patch().magSf();
116
117     //patch-adjacent adjoint velocity Uac
118     vectorField Uac = (Uap.patchInternalField());
119
120     //patch-adjacent normal adjoint velocity Uac_n
121     scalarField Uac_n = (Uap.patchInternalField() & np);
122
123     //patch-adjacent tangential adjoint velocity Uac_n
124     vectorField Uac_t = Uac - Uac_n * np;
125
126         vectorField::operator=(
127                                 - (nuw * delta)/(Uap_n + nuw*delta) * Uac_t
128                             );
```

```
129
130     fixedValueFvPatchVectorField::updateCoeffs();
131 }
132
133
134 void Foam::adjointCdOutletVelocityFvPatchVectorField::write(Ostream& os) const
135 {
136     fvPatchVectorField::write(os);
137     writeEntry("value", os);
138 }
139
140
141 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
142
143 namespace Foam
144 {
145     makePatchTypeField
146     (
147         fvPatchVectorField,
148         adjointCdOutletVelocityFvPatchVectorField
149     );
150 }
151
152
153 // ************************************************************************* //
```

adjointOutletVelocityFvPatchVectorField.C

adjointWallVelocityFvPatchVectorField.C : nel file viene introdotta la condizione al contorno sulla componente normale della velocità aggiunta sulla superficie del profilo come è stato ricavato nell'equazione 2.28.

```
1  /*--------------------------------------------------------------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4     \\    /   O peration     |
5      \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
6       \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
13     the Free Software Foundation, either version 3 of the License, or
14     (at your option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
19     for more details.
20
21     You should have received a copy of the GNU General Public License
22     along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
```

```
23
24  \*---------------------------------------------------------------------------*/
25
26  #include "adjointCdWallVelocityFvPatchVectorField.H"
27  #include "volFields.H"
28  #include "addToRunTimeSelectionTable.H"
29  #include "surfaceFields.H"
30  #include "fvPatchFieldMapper.H"
31
32  // * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * * //
33
34  Foam::adjointCdWallVelocityFvPatchVectorField::
35  adjointCdWallVelocityFvPatchVectorField
36  (
37      const fvPatch& p,
38      const DimensionedField<vector, volMesh>& iF
39  )
40  :
41      fixedValueFvPatchVectorField(p, iF)
42  {}
43
44
45  Foam::adjointCdWallVelocityFvPatchVectorField::
46  adjointCdWallVelocityFvPatchVectorField
47  (
48      const fvPatch& p,
49      const DimensionedField<vector, volMesh>& iF,
50      const dictionary& dict
51  )
52  :
53      fixedValueFvPatchVectorField(p, iF)
54  {
55      fvPatchVectorField::operator=(vectorField("value", dict, p.size()));
56  }
57
58
59  Foam::adjointCdWallVelocityFvPatchVectorField::
60  adjointCdWallVelocityFvPatchVectorField
61  (
62      const adjointCdWallVelocityFvPatchVectorField& ptf,
63      const fvPatch& p,
64      const DimensionedField<vector, volMesh>& iF,
65      const fvPatchFieldMapper& mapper
66  )
67  :
68      fixedValueFvPatchVectorField(ptf, p, iF, mapper)
69  {}
70
71
72  Foam::adjointCdWallVelocityFvPatchVectorField::
73  adjointCdWallVelocityFvPatchVectorField
74  (
75      const adjointCdWallVelocityFvPatchVectorField& pivpvf,
76      const DimensionedField<vector, volMesh>& iF
77  )
```

```
78  :
79      fixedValueFvPatchVectorField(pivpvf, iF)
80  {}
81
82
83  // * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //
84
85  // Update the coefficients associated with the patch field
86  void Foam::adjointCdWallVelocityFvPatchVectorField::updateCoeffs()
87  {
88
89      if (updated())
90      {
91          return;
92      }
93
94
95      // Extract the dictionary from the database
96      const dictionary& optimizationProperties = db().lookupObject<IOdictionary>
97      (
98       "optimizationProperties"
99       );
100
101
102     // Extracting value
103     vector d(optimizationProperties.lookup("d"));
104     dimensionedScalar U_inf(optimizationProperties.lookup("U_inf"));
105
106     scalar C_inf(.5*pow(U_inf.value(), 2));
107
108         vectorField::operator=(
109                             - patch().nf()*( (1/C_inf)*(patch().nf() & d) )
110                             );
111
112     fixedValueFvPatchVectorField::updateCoeffs();
113 }
114
115
116 void Foam::adjointCdWallVelocityFvPatchVectorField::write(Ostream& os) const
117 {
118     fvPatchVectorField::write(os);
119     writeEntry("value", os);
120 }
121
122
123 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
124
125 namespace Foam
126 {
127     makePatchTypeField
128     (
129         fvPatchVectorField,
130         adjointCdWallVelocityFvPatchVectorField
131     );
132 }
```

```
133
134
135 // ************************************************************************* //
```

adjointWallVelocityFvPatchVectorField.C

Per quanto riguarda i file rimanenti che non sono stati riportati come listati o non è stato necessario modificarli rispetto alla versione esistente nel solutore *adjointShapeOptimizationFoam* oppure non determinano un contributo fondamentale per la spiegazione dell'algoritmo su cui si basa questo solutore, per cui sono stati omessi.

Capitolo 4

Metodologia

Sviluppato il solutore per il problema aggiunto, per raggiungere lo scopo desiderato è necessario integrare questo tool all'interno di un ciclo di ottimizzazione. Richiamando quanto già anticipato nell'introduzione, lo scopo generale di questo lavoro è quello di sviluppare un processo di ottimizzazione vincolata, per il momento limitato a profili 2D. L'introduzione di un vincolo nell'ottimizzazione è necessario per far in modo che l'output dell'intero processo sia in grado di rispettare certe specifiche di performance o di realizzabilità.

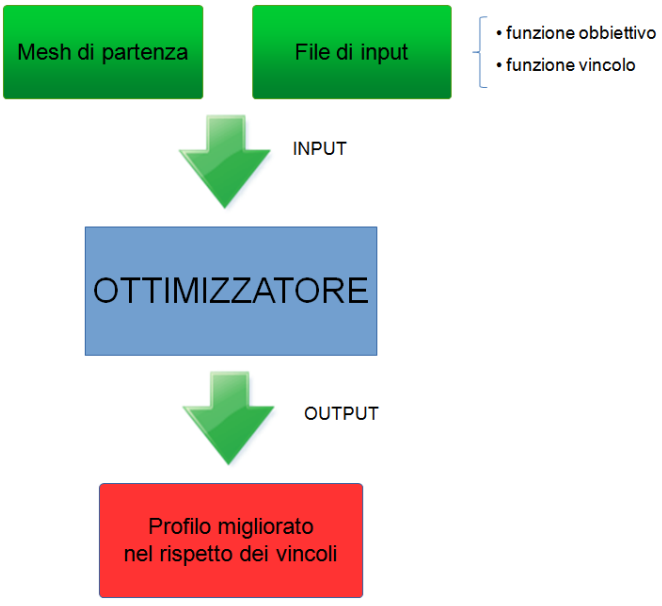


FIGURA 4.1: Schema riassuntivo del processo

Facendo riferimento alla Figura 4.1, gli input richiesti da questo processo sono due: una mesh di partenza sull'oggetto dell'ottimizzazione e un file di input, in cui l'utente deve essere in grado di specificare la funzione costo, le funzioni vincolo e scegliere se i vari funzionali devono essere minimizzati o massimizzati. Questi due elementi innescheranno

il ciclo di ottimizzazione che avrà come risultato finale un CAD del corpo ottimizzato nel rispetto dei vincoli.

Per gestire un ciclo di ottimizzazione vincolato, il solo programma OpenFOAM non risulta più sufficiente, in quanto si rende necessario l'utilizzo di un programma esterno di ottimizzazione che sia in grado di pilotare il solver diretto e quello aggiunto e, sulla base delle informazioni fornite dagli stessi, individuare la direzione ottima per il passo successivo del ciclo.
Portando avanti la scelta di operare unicamente con strumenti *open source*, si è deciso di utilizzare per questo scopo Dakota (Design Analysis Kit for Optimization and Terascale Applications), software sviluppato presso i Sandia National Laboratories. Di seguito verrà introdotto il software Dakota evidenziandone le caratteristiche che lo rendono lo strumento adeguato per gli scopi descritti. Per una descrizione più approfondita si rimanda al manuale utente [10].

4.1 Dakota

4.1.1 Introduzione a Dakota

Dakota è uno strumento per l'ottimizzazione in grado di fornire un gran numero di metodi iterativi e algoritmi in grado di rapportarsi, principalmente, con i seguenti problemi:

- **Ottimizzazione**: lo scopo è quello di massimizzare o minimizzare una funzione costo, il cui valore viene ricavato da un modello di simulazione. All'interno di Dakota sono presenti algoritmi *gradient-based*, *derivative-free* e di ottimizzazione globale. Inoltre Dakota permette di gestire ottimizzazioni multi-obbiettivo e vincolate.
- **Studi parametrici** (DOE): tali metodi hanno lo scopo di esplorare l'effetto di cambiamenti parametrici all'interno di un modello di simulazione, esplorando l'intero spazio dei parametri del problema. L'output ricercato in questo tipo di studi è l'analisi di sensitività di un funzionale rispetto alle variabili di design.
- **Quantificazione di incertezza**: i metodi di quantificazione di incertezza calcolano informazioni probabilistiche sulla funzione costo basandosi sulle distribuzioni di probabilità fornite come parametri di input.
- **Calibrazione**: gli algoritmi di calibrazione cercano di massimizzare l'accordo tra gli output del modello di simulazione e i dati sperimentali.

Tutti questi algoritmi sono stati implementati in modo da poter sfruttare il calcolo parallelo.

4.1.2 Interfaccia con solutori esterni

Queste capacità possono essere usate come *stand-alone* all'interno di Dakota oppure possono essere facilmente interfacciate con un codice di simulazione esterno.

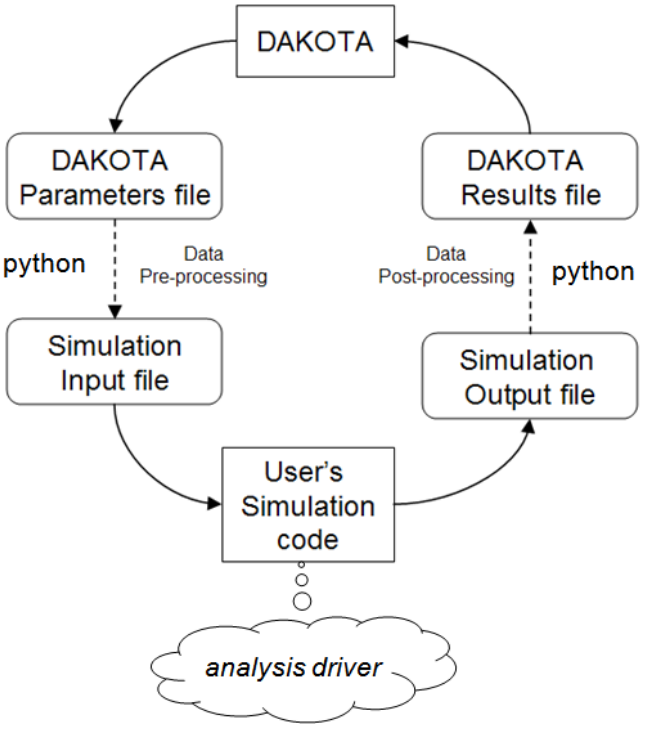


FIGURA 4.2: Interfaccia tra Dakota e software esterno

In figura 4.2 è schematizzato il funzionamento del processo di interfaccia tra Dakota e un generico modello di simulazione esterno. Nello schema, le frecce continue definiscono delle operazioni di input/output realizzate all'interno di un unico ambiente, sia esso Dakota o il codice di simulazione, mentre le frecce tratteggiate rappresentano uno scambio di dati tra i due sistemi, che avviene attraverso la scrittura e lettura di brevi file di dati. La conversione di questi file dalla sintassi di Dakota a quella del solutore esterno deve essere codificata dall'utente e, per le specifiche esigenze di questo lavoro, questo compito viene realizzato da dei semplici script python.

4.1.3 File di input

Ciascun algoritmo di Dakota, per iniziare, ha bisogno di alcune informazioni contenute in un file di input, chiamato *dakota.in*, all'interno di cui l'utente può specificare:

- algoritmo di ottimizzazione: esistono diversi algoritmi tra cui poter scegliere, ma al momento l'algoritmo *conmin*, è l'unico in grado di poter gestire una ottimizzazione vincolata;
- funzione costo: è necessario scegliere il numero delle funzioni costo (c'è la possibilità di gestire ottimizzazioni multi-obbiettivo) e indicare il nome del file all'interno di cui, ad ogni passo del ciclo, verrà scritto il valore assunto dalla funzione stessa;
- tolleranza sulla convergenza: dopo ogni passo, se la valutazione della funzione costo nelle ultime due configurazioni differisce di un valore minore di quanto specificato, il ciclo viene considerato a convergenza;
- funzione vincolo: è necessario scegliere il numero delle funzioni e indicare il nome del file all'interno di cui, ad ogni passo del ciclo, verrà scritto il valore assunto dalla funzione stessa;
- tolleranza sul rispetto dei vincoli;
- variabili di design: fornire un nome di identificazione, un valore iniziale e range di valori ammissibili per ogni variabile;
- specificare se si ricerca la massimizzazione o la minimizzazione della funzione costo;
- modalità di calcolo dei gradienti: *numerical gradients*, se si vuole che i gradienti siano calcolati da Dakota, o *analytical gradients*, se si desidera utilizzare un codice di calcolo esterno.
- *analysis driver*: infine, è necessario specificare il nome del file all'interno di cui viene gestito il blocco dedicato al software esterno. Esso contiene al suo interno i comandi del codice di simulazione nella loro giusta sequenza.

Specificate queste informazioni, Dakota dà inizio al ciclo creando una nuova directory associata al primo passo e generando, al suo interno, un file chiamato *params.txt*, che contiene i valori assunti dalle variabili di design al passo corrente. A questo punto entra in gioco il primo file python che, leggendo i valori in *params.txt*, genera i file di input necessari per il codice di simulazione. Una volta eseguita la serie di comandi riportati in *analysis driver*, un secondo script python esegue il post-processing dei risultati della simulazione per estrarre i dati di cui necessita l'ottimizzatore per generare il successivo file dei parametri e ricominciare il ciclo.

Queste caratteristiche fanno di Dakota un ambiente flessibile ed efficace per il design e l'ottimizzazione, rendendolo lo strumento adatto per pilotare il ciclo di ottimizzazione sotto esame.

4.2 Ciclo di ottimizzazione

Dopo aver approfondito il funzionamento del motore di tutto il processo è possibile entrare nei dettagli del ciclo di ottimizzazione vero e proprio e dell'interfaccia con OpenFOAM.
In figura 4.3 è riportato uno schema dettagliato del ciclo di ottimizzazione.

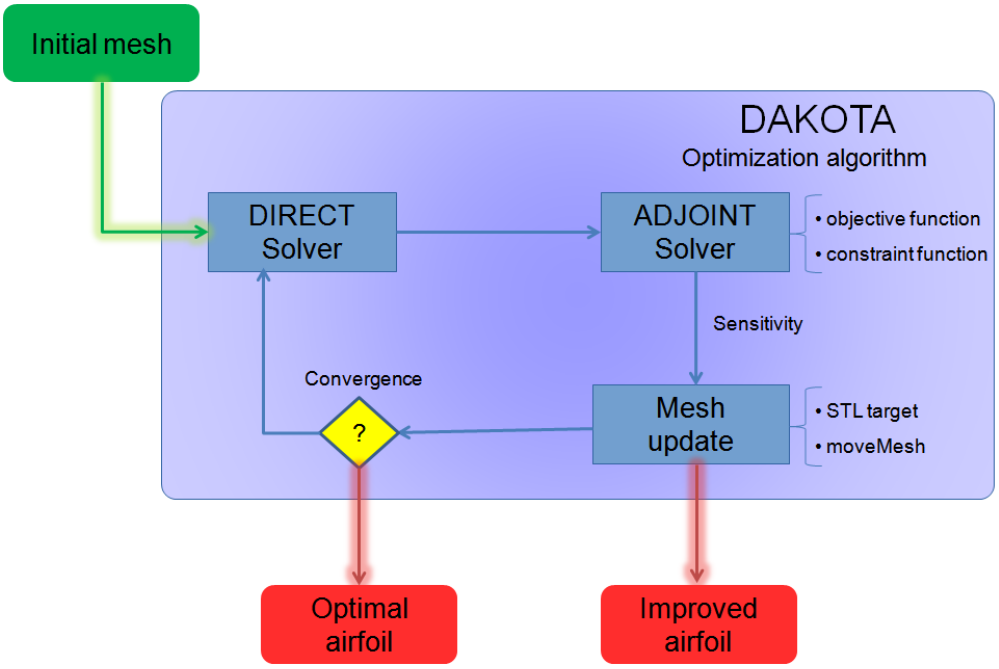


FIGURA 4.3: Schema del ciclo di ottimizzazione

Come già anticipato, il punto di partenza è rappresentato da una mesh iniziale del profilo, mentre il ciclo di ottimizzazione si compone dei seguenti passi:

- Solutore diretto: viene utilizzato il solutore di OpenFOAM per le equazioni di Navier-Stokes incomprimibili *simpleFoam*.
- Solutore aggiunto: viene utilizzato il solutore *adjointOptFoam*, descritto nel Capitolo 2. Nel caso sia presente un vinvolo, il problema aggiunto deve essere risolto due volte: una per la funzione obbiettivo e una per la funzione vincolo.
- Deformazione della mesh: i due passi precedenti forniscono come output i valori della funzione costo e della funzione vincolo valutate sulla configurazione corrente e il relativo andamento della sensitività sul profilo. Sulla base di questi dati l'ottimizzatore fornisce i nuovi valori dei parametri di design per il passo successivo, su cui viene riaggiornata la mesh, e il loop può ricominciare dal primo punto.

I primi due punti sono già stati sviluppati in precedenza, al contrario il terzo, riguardante la deformazione della mesh, solleva la presenza di due questioni aperte: quali variabili

di design scegliere per descrivere il profilo e quale strumento utilizzare per modificare la mesh.

4.2.1 Variabili di design: punti di controllo

Dato che il solutore aggiunto *adjointOptFoam* è in grado di calcolare la variazione di una funzione obbiettivo rispetto allo spostamento normale alla superficie di un qualsiasi punto del profilo, la scelta che sembrerebbe più logica sarebbe quella di scegliere coma variabili di design tutti i punti del profilo sulla mesh. Tuttavia, seguendo questa strada, che è dettagliata con maggior precisione nell'Appendice A, ci si deve confrontare con diversi problemi, tra cui il principale è rappresentato dalla forma irregolare assunta dai profili modificati.

Per risolvere questi problemi si è deciso di parametrizzare la forma del profilo attraverso l'inserimento di un certo numero di punti di controllo sulla sua superficie, in numero molto minore rispetto ai punti sulla mesh. Per questa parametrizzazione sono state implementate, nello script python che interfaccia il sistema OpenFOAM con quello di Dakota, le funzioni di Hicks-Henne. Queste funzioni, che per la loro forma vengono anche chiamate *bump function*, vengono sommate al profilo indeformato per ottenere il profilo al passo successivo, secondo l'equazione:

$$y^{n+1} = y^n + \sum_{k=1}^{N_C} d_k f_k(x) \tag{4.1}$$

con

$$f_k(x) = \sin^3(\pi x^{\epsilon_k})$$

$$\epsilon_k = \log(0.5)/\log(x_k)$$

In cui d_k sono le ampiezze delle bump function $f_k(x)$ e rappresentano quindi le variabili di design del problema di ottimizzazione, mentre x_k è l'ascissa in cui si colloca il massimo della funzione k-esima.

Queste funzioni, rappresentate in Figura 4.4, sono particolarmente adatte per il design di profili alari, in quanto ciascuna di esse si raccorda a 0 in corrispondenza del bordo d'attacco (x=0) e del bordo d'uscita (x=1) [11].

Questa scelta implica la necessità di trasformare l'output del solutore aggiunto, fornito come gradiente della funzione costo rispetto allo spostamento normale di tutti punti

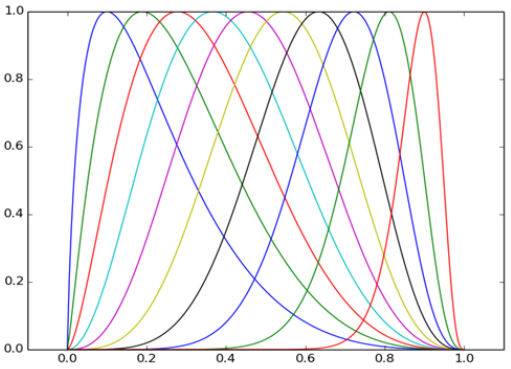


FIGURA 4.4: Funzioni di Hicks-Henne

del profilo appartenenti alla griglia di calcolo, nel gradiente della funzione obbiettivo rispetto ad uno spostamento normale di ciascun punto di controllo. Questo può essere facilmente ottenuto con la regola della derivazione di funzioni composte, o *chain rule*. Infatti, indicando con x_j i punti di griglia e con X_i i punti di controllo, si ricava:

$$\frac{\partial J}{\partial X_i} = \sum_{j=1}^{N_G} \frac{\partial J}{\partial x_j} \frac{\partial x_j}{\partial X_i} \tag{4.2}$$

Dove la sommatoria in j va estesa a tutti i punti di griglia appartenenti alla superficie del profilo N_G. Le grandezze $\frac{\partial J}{\partial x_j}$ sono proprio i valori forniti in output dal solutore aggiunto, mentre i valori $\frac{\partial x_j}{\partial X_i}$ si possono ricavare dalla definizione delle funzioni di Hicks-Henne in 4.1. Facendo riferimento ad uno spostamento infinitesimo δ_i del punto di controllo X_i si ottiene:

$$\frac{\partial x_j}{\partial X_i} = \delta_i \sin^3(\pi x_j^{\epsilon_i})$$

Sostituendo nella relazione precedente si ottiene la formula da implementare nello script di interfaccia:

$$\frac{\partial J}{\partial X_i} = \sum_{j=1}^{N_G} \frac{\partial J}{\partial x_j} \delta_i \sin^3(\pi x_j^{\frac{\log 0.5}{\log X_i}}) \tag{4.3}$$

4.2.2 Deformazione della mesh: moveDynamicMesh

Per modificare la mesh tra un passo del ciclo e il successivo si utilizza il tool *built-in* di OpenFOAM *moveDynamicMesh*. Questo strumento, partendo da una mesh e da un CAD di un oggetto di riferimento in formato .stl, permette di deformare la mesh in moda da farla aderire all'stl target. Questa procedura può essere facilmente inserita e automatizzata nel ciclo, attraverso il secondo script python che interfaccia Dakota con OpenFOAM. Infatti, come si è già visto, l'ottimizzatore di Dakota restituisce ad ogni passo un file dei parametri in cui sono contenuti gli spostamenti, calcolati come ottimi,

dei punti di controllo, espressi in termini di ampiezza delle rispettive *bump function* d_k. Sommando tutte le funzioni di Hicks-Henne si ottiene il profilo deformato e, di conseguenza, le coordinate dei nuovi punti di griglia che descrivono il profilo. Da essi, attraverso un algoritmo di triangolazione dei punti, è possibile generare un file stl del profilo deformato. A questo punto la routine *moveDynamicMesh* deforma la mesh in modo da far coincidere la *patch* corrispondere al profilo con l'stl target. Inoltre, per far in modo di poter confrontare le caratteristiche aerodinamiche del profilo iniziale con quelli deformati, ad ogni passo il profilo ottenuto dopo la deformazione viene riscalato a corda unitaria.

L'algoritmo di deformazione non si limita a spostare i soli punti della *patch* associata al profilo, ma coinvolge anche i punti nelle vicinanze, in modo da garantire una buona qualità della mesh deformata.
Una limitazione che si è riscontrata nell'utilizzo della routine *moveDynamicMesh*, è stata la degradazione che subisce la qualità della griglia di calcolo durante la deformazione, nel momento in cui sono presenti degli scostamenti significativi tra il profilo di partenza e il profilo target. Per evitare problemi di questo tipo lo spostamento massimo ammissibile per i punti di controllo viene imposto pari a 10^{-2}.

Infine, per rendere il più flessibile e *user-friendly* possibile l'utilizzo di questo strumento, è stata introdotta la possibilità di compilare un file di input, denominato *opt.in*, in cui è possibile specificare la funzione obbiettivo dell'ottimizzazione, il tipo di vincolo e il numero di punti di controllo desiderati, che verrano distribuiti sulla superficie del profilo seconda una distribuzione di Chebychev, in modo da avere una maggior capacità di controllo sul bordo d'attacco e sul bordo d'uscita.
Bisogna tenere presente che, al momento, è possibile specificare solo due tipi di ottimizzazione: o una minimizzazione del coefficiente di resistenza con un vincolo sul coefficiente di portanza oppure una massimizzazione del coefficiente di portanza con un vincolo sul coefficiente di resistenza.

4.2.3 Criterio di convergenza

La convergenza del ciclo è controllata da Dakota stesso al termine di ogni passo. Come è già stato anticipato, all'interno del file di input *dakota.in* deve essere specificata la parola chiave *convergence_tolerance*, che nel caso specifico di queste simulazioni è stato fissato pari a 10^{-5}. Se le valutazioni delle funzioni obbiettivo delle ultime tre configurazioni differiscono l'uno dalla precedente di una quantità minore della tolleranza specificata e la funzione obbiettivo si trova all'interno del range di valori ammessi, allora il ciclo è considerato a convergenza e viene arrestato.

Esiste un'altra parola chiave, *max_iterations*, grazie a cui è possibile specificare un numero massimo di passi che si vuole compiere. Nel caso in analisi si è scelto di non specificare tale parametro in modo da essere sicuri di raggiungere la configurazione migliore possibile.

4.3 Settaggio di una simulazione in OpenFOAM

In figura 4.5, è schematizzata la struttura del caso di partenza in OpenFOAM, a cui viene attribuito il nome *casebase*.

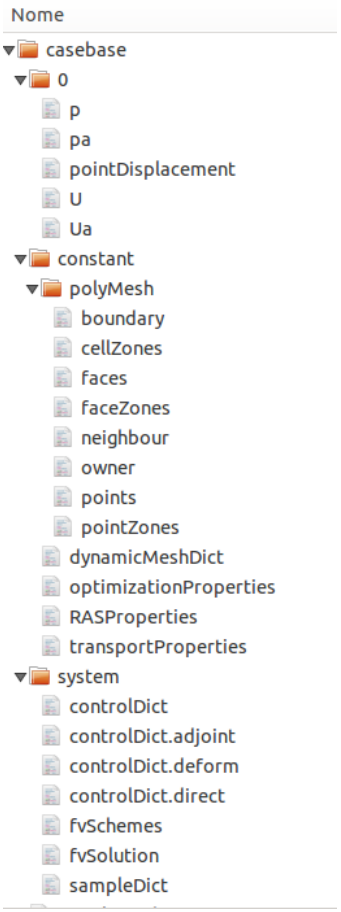


FIGURA 4.5: Struttura del caso iniziale in OpenFOAM

Ovviamente, sono presenti le tre cartelle necessarie per una simulazione standard:

- **0**: contiene le condizioni iniziali di tutte le grandezze coinvolte, sia per il problema diretto che per quello aggiunto;
- **constant**: contiene i file in cui viene definito il regime di moto e, nel caso di regime turbolento, il modello di turbolenza utilizzato, mentre nella cartella *polyMesh* sono presenti tutte le informazioni riguardati la griglia di calcolo;

- **system**: contiene i dizionari in cui vengono specificati i parametri della simulazione e gli schemi utilizzati nel problema:
  - *controlDict*: contiene i parametri che servono a gestire la simulazione, come ad esempio il numero di iterazioni massime e il numero di cifre significative che saranno utilizzate. Esistono tre versioni diverse di questo file per la necessità di variare questi parametri a seconda del tipo di problema che si sta risolvendo: deformazione della mesh, diretto o aggiunto: sono stati utilizzati degli schemi al secondo ordine sia per il problema diretto che per quello aggiunto;
  - *fvSchemes*: al suo interno vengono specificati gli schemi numerici da applicare per la discretizzazione di ogni operatore differenziale;
  - *fvSolution*: al suo interno vengono specificati i solutori utilizzati per ciascuna grandezza, i criteri d'arresto della simulazione e i fattori di rilassamento;
  - *sampleDict*: è il dizionario per tutto ciò che riguarda il *post-processing*. Vanno specificati il nome delle grandezze di interesse e la *patch* su cui si desidera estrarle.

Dato che, all'interno del ciclo, non è richiesto generare una nuova mesh ad ogni passo, tutti i file relativi al *pre-processing* non sono necessari.

Oltre a questi componenti standard esistono dei file specifici e necessari per la particolare procedura in esame. Questi sono:

- *pointDisplacement*: al suo interno sono specificate quali *patch* dovranno essere modificate nel processo di deformazione della griglia e il nome del relativo CAD target e quali dovranno invece rimanere fissate;
- *dinamicMeshDict*: contiene i parametri con cui viene settato l'algoritmo di deformazione della mesh;
- *optimizationProperties*: questo file contiene alcune informazioni importanti per la definizione del problema aggiunto. In particolare l'utente deve definire all'interno di questo file il modulo della velocità asintotica, la direzione della forza aerodinamica oggetto dell'ottimizzazione e indicare se si vuole ottenere una minimizzazione o una massimizzazione di tale forza.

Di seguito viene riportato un esempio del file *optimizationProperties*, per un caso a velocità unitaria ed incidenza nulla in cui si vuole minimizzare il coefficiente di resistenza.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      optimizationProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


U_inf                           U_inf [ 0 1 -1 0 0 0 0 ] 1;

d                               (1.0 0.0 0.0);

optimization                    minimize;

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

optimizationProperties

Capitolo 5

Risultati

In questo capitolo vengono riportati i risultati ottenuti applicando la metodologia di ottimizzazione descritta al capitolo precedente. Come primo test per questo nuovo strumento si è deciso di condurre delle simulazioni su un profilo NACA 0012 in regime laminare, per evitare eventuali problemi introdotti dai modelli di turbolenza. Sono state effettuate diverse prove a differenti numeri di Reynolds e angoli di incidenza e ciascuna di esse ha evidenziato il buon funzionamento del ciclo di ottimizzazione.

Per ogni configurazione sono state portate a termine due ottimizzazioni diverse:

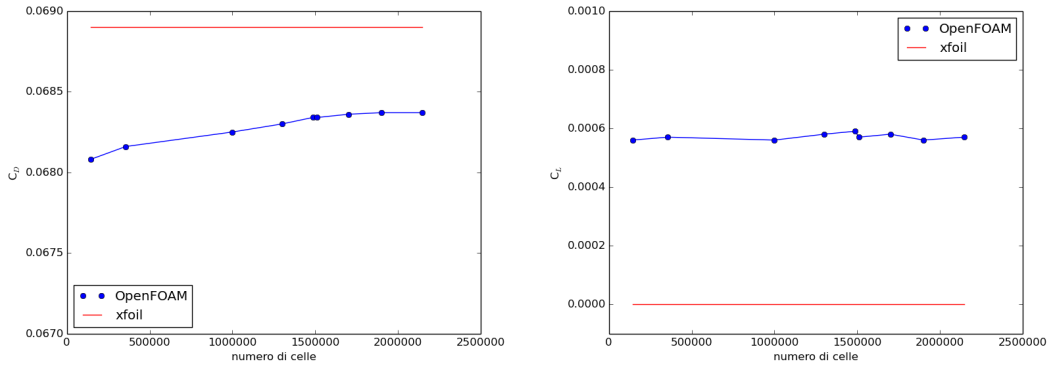
- una minimizzazione del coefficiente di resistenza, vincolando il coefficiente di portanza a non scendere al di sotto del valore iniziale;
- una massimizzazione del coefficiente di portanza, vincolando il coefficiente di resistenza a non superare il valore iniziale.

La scelta dei vincoli applicati risiede nella volontà di non peggiorare le qualità di efficienza aerodinamica dei profili ottimizzati.
Si ricordi che in ogni ciclo è implicitamente presente un ulteriore vincolo geometrico con cui la corda viene mantenuta unitaria.

5.1 Studio di sensitività della mesh

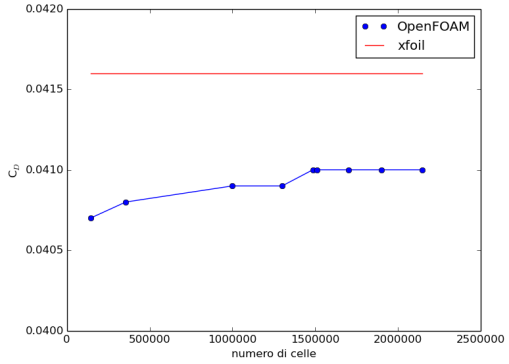
Prima di procedere con l'esposizione dei risultati, è necessario presentare le prove svolte e le considerazioni che hanno determinato la scelta della mesh di partenza.
Come validazione della mesh è stata effettuato uno studio di sensitività sui risultati ottenuti in termini di C_D e C_L al variare del numero di celle componenti la griglia di

calcolo. Le mesh utilizzate per questo studio sono mesh esaedriche strutturate e sono state generate attraverso il software Pointwise. Le simulazioni sono state svolte per due condizioni di moto diverse: la prima ad un numero di Reynolds pari a 3000 e incidenza nulla e la seconda a Reynolds 10000 e incidenza 2 gradi. Non essendo presenti dati sperimentali riferiti a prove condotte in regime laminare, si sono dovuti confrontare i risultati ottenuti con i valori forniti dal programma Xfoil [12] sullo stesso profilo con un calcolo viscoso allo stesso valore di numero di Reynolds e incidenza.

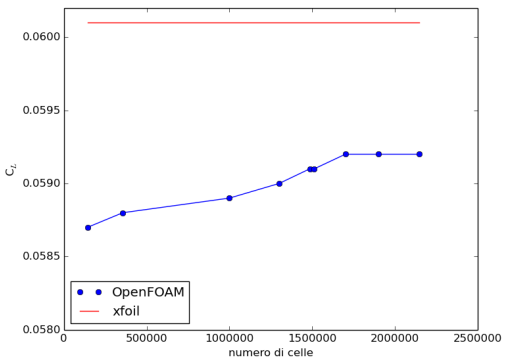


(A) *Re = 3000, α = 0 deg*

(B) *Re = 3000, α = 0 deg*



(C) *Re = 10000, α = 2 deg*



(D) *Re = 10000, α = 2 deg*

FIGURA 5.1: Convergenza dei coefficienti aerodinamici al variare del numero di celle della mesh

Dai risultati di convergenza in figura 5.1 si nota come, oltre le 1500000 celle, i valori dei coefficienti aerodinamici si attestino a convergenza. Le differenze tra questi valori di convergenza e i relativi risultati forniti da Xfoil sono riassunti in tabella 5.1.

TABELLA 5.1: Confronto tra OpenFOAM e Xfoil

Simulazione	$\Delta C_D(xfoil)$	$\Delta C_L(xfoil)$
$Re = 3000, \alpha = 0deg$	0.8%	0%
$Re = 10000, \alpha = 2deg$	1.4%	1.3%

In particolare, la differenza in termini di C_L può essere indagata attraverso il confronto dell'andamento del coefficiente di pressione lungo la corda. Per rendere più evidenti le differenze tra la simulazione svolta in OpenFOAM e i risultati di Xfoil, in figura 5.2, è proposto il confronto tra i coefficienti di pressione ad un numero di Reynolds di 10000 e incidenza 4 gradi dove, sulla mesh più lasca, si ha una differenza del 20% tra i due valori.

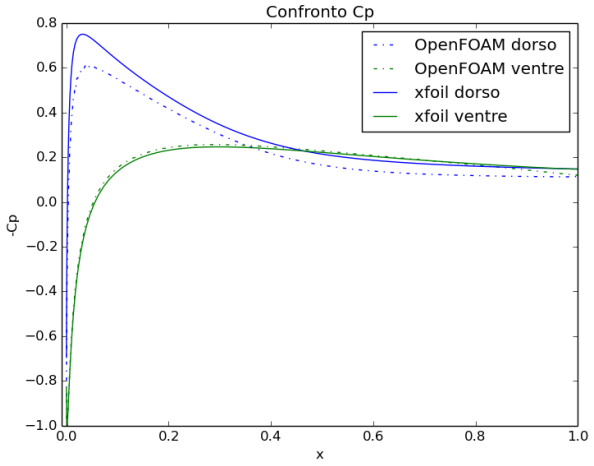


FIGURA 5.2: Confronto tra coefficienti di pressione

Da questo confronto risulta evidente che la differenza principale tra le due soluzioni sta nell'individuazione dell'espansione sul dorso del profilo, che risulta maggiore nel risultato fornito da Xfoil. Questa differenza potrebbe essere attribuita ad una differenza dello spessore dello strato limite calcolato dai due diversi codici, in particolare ci si dovrebbe aspettare che la presenza della sovraespansione sul dorso presente nella soluzione di Xfoil sia dovuta alla previsione di uno strato limite più spesso rispetto a quello calcolato da OpenFOAM.

In [13], lo stesso sviluppatore di Xfoil, professor Mark Drela, descrive dettagliatamente l'algoritmo utilizzato nel suo software per il calcolo dello strato limite. Esso si basa su un metodo iterativo che accoppia una formulazione non viscosa, che risolve le equazioni di Eulero, e una formulazione integrale viscosa, che utilizza l'equazione di von Karman 5.1 per risolvere lo strato limite in termini di spessore di spostamento δ^* e spessore di quantità di moto θ. Ad ogni iterazione, il calcolo inviscido eredita dal calcolo viscoso al passo precedente una condizione al contorno di traspirazione che, sulla base del valore di spessore di spostamento δ^* calcolato calcolato lungo la corda, impone una condizione di non compenetrazione ad una distanza δ^* dallo spessore del profilo, andando di fatto ad imporre uno spostamento delle linee di corrente adiacenti al profilo. Questo accoppiamento viene iterato fino a convergenza.

$$\frac{1}{\rho}\tau_0 = \frac{d}{dx}\left(U_e^2\theta\right) + U_e\delta^*\frac{dU_e}{dx} \tag{5.1}$$

Una conferma della validità di questo approccio iterativo può essere ritrovato nel metodo delle espansioni asintotiche, con cui è possibile ricavare il modello delle equazioni dello strato limite riportato nel sistema 5.2. Infatti questo approccio si basa su una suddivisione del calcolo del flusso in un problema interno e di un problema esterno rispetto allo strato limite, interfacciati da opportune condizioni di interfaccia.
Una trattazione dettagliata di questo metodo può essere trovata, ad esempio, in [14].

$$\begin{cases} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} \\ \frac{\partial p}{\partial y} = 0 \end{cases} \tag{5.2}$$

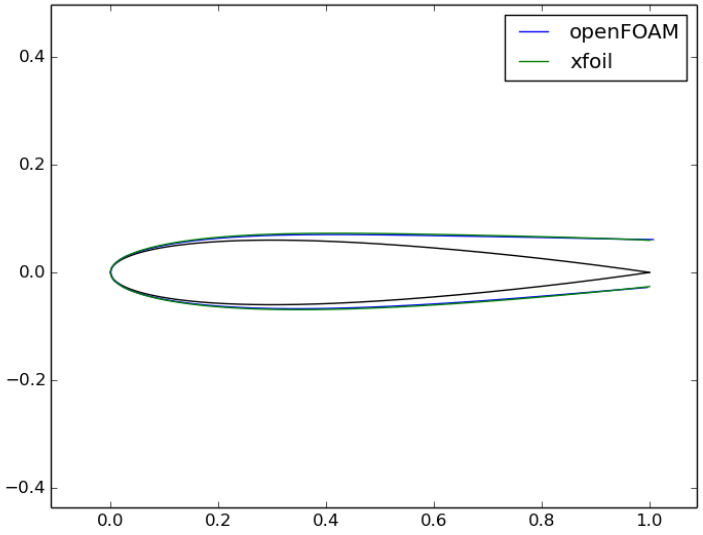
Xfoil permette di estrarre facilmente in un file di output il profilo di spessore di spostamento così calcolato sul profilo, quindi per rendere possibile il confronto non rimane che estrarre le stesse informazioni dalla simulazione effettuata con OpenFOAM.
Per fare ciò si è fatto uso di una funzione presente nel software di visualizzazione ParaView in grado di estrarre i valori assunti dal campo di velocità lungo una linea normale al profilo in un punto definito. Una volta in possesso di queste informazioni il valore di spessore di spostamento dello strato limite in corrispondenza del punto in esame è calcolabile tramite la formula:

$$\delta^* = \int_0^H \left(1 - \frac{u(y)}{U_\infty}\right) dy \tag{5.3}$$

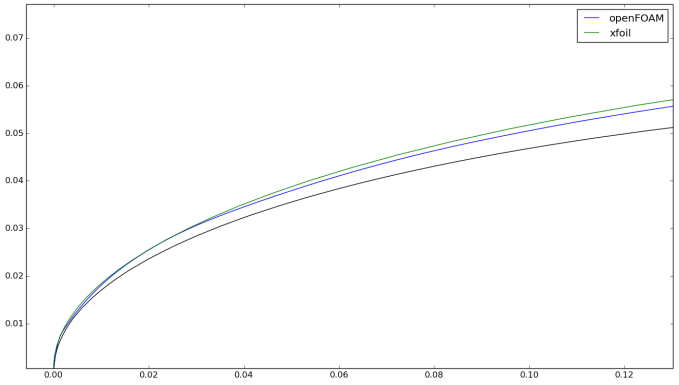
in cui y rappresenta l'ascissa lungo la normale al profilo e si estende dalla superficie del profilo stesso ($y = 0$) fino al bordo del dominio ($y = H$), $u(y)$ il modulo della velocità nel punto y e U_∞ il valore di velocità asintotica.

In figura 5.3 vengono confrontati i profili di spessore di spostamento dello strato limite così come calcolati da Xfoil e OpenFOAM nella stessa configurazione precedente a Reynolds 10000 e incidenza 4 gradi. Si può osservare, come ci si aspettava, che pur essendo i due andamenti molto simili, sul dorso del profilo lo strato limite calcolato da Xfoil risulta più spesso di quello calcolato in OpenFOAM. Questa differenza può essere attribuita proprio al metodo di calcolo iterativo utilizzato in xfoil che lo stesso Drela in [15] afferma essere meno accurato per bassi numeri di Reynolds.

In conclusione, questo confronto non permette di giudicare l'effettiva correttezza dei risultati ottenuti rispetto ai valori reali, in quanto gli stessi risultati forniti da Xfoil



(A) *Profilo completo*



(B) *Particolare del dorso del profilo*

FIGURA 5.3: Confronto tra strato limite calcolato da OpenFOAM e xfoil

saranno caratterizzati da una certa percentuale di incertezza, tuttavia può essere considerato come un indizio del fatto che questi non siano molto lontani dagli effettivi valori esatti.

5.2 Mesh iniziale

L'aspetto fondamentale che si è dovuto tenere presente nella scelta della mesh di partenza per il ciclo di ottimizzazione è stato il *trade-off* tra il ridurre la dimensione delle celle a parete in modo da risolvere il più precisamente possibile lo strato limite sul profilo, e quindi ottenere risultati più precisi, e la necessità di non infittire troppo la mesh in

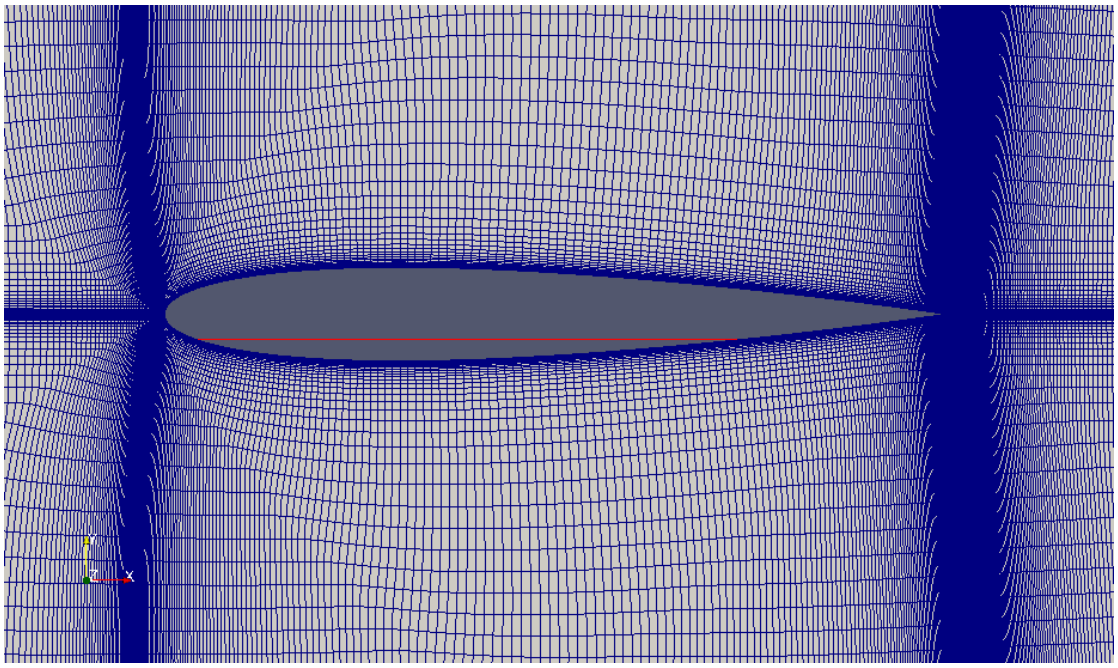


FIGURA 5.4: Particolare della mesh iniziale sul profilo NACA 0012

corrispondenza del profilo stesso in modo da poter deformare la mesh ad ogni passo del ciclo senza degradarne la qualità. In particolare, il vincolo introdotto da quest'ultima considerazione risulta molto stringente e ha determinato la scelta di utilizzare come mesh di partenza la griglia composta da 150000 elementi. Tuttavia questa scelta non penalizza eccessivamente la qualità dei risultati ottenuti: infatti dalle figure 5.1 si può constatare come i valori di C_D e C_L ottenuti sulla mesh più lasca differiscano di meno dell'1% rispetto ai valori raggiunti a convergenza sulle mesh più fitte.

In figura 5.4 è riportato uno zoom della mesh considerata sul profilo.

Per le simulazioni seguenti si è scelto di utilizzare 50 punti di controllo spaziati con una distribuzione di Chebychev. Il perché di questa scelta verrà spiegata più avanti nel capitolo, quando verrà effettuata uno studio delle *performance* dell'ottimizzazione al variare del numero dei punti di controllo.

5.3 Sensitività e confronto con le differenze finite

Per verificare l'accuratezza dell'andamento della sensitività fornita in output dal solutore *adjointOptFoam* si è deciso di confrontarla con i risultati forniti dalle differenze finite nelle stesse condizioni sul profilo di partenza.

Prendendo spunto dall'articolo di Brezzillon e Gauger [16], sono state implementate le differenze finite al secondo ordine e applicate per ricavare gli andamenti della sensitività

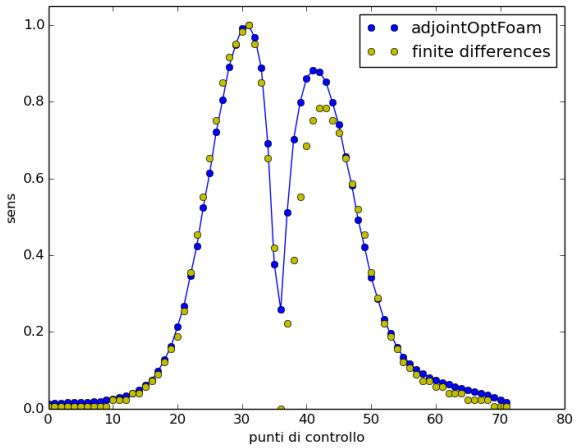
in una condizione di riferimento a Reynolds 3000 a due diverse incidenze: si è ricavato il profilo di sensitività che massimizza il C_L a incidenza nulla e il profilo corrispondente a una minimizzazione del C_D ad un incidenza di 2 gradi.
Grazie alla parametrizzazione adottata per il profilo questa implementazione risulta immediata. Dopo aver scelto uno spostamento δs sufficientemente piccolo, per ogni punto di controllo i è necessario effettuare due simulazioni del problema diretto: la prima sul profilo ottenuto spostando il punto di controllo i-esimo in direzione normale alla superficie ed entrante nel profilo di una lunghezza δs, ricavando così i coefficienti $C_{D_i}^-$ e $C_{L_i}^-$; la seconda sul profilo ottenuto spostando lo stesso punto di controllo di una lunghezza δs ma in direzione normale uscente dal profilo, ottenendo i coefficienti $C_{D_i}^+$ e $C_{L_i}^+$. In questo modo è possibile calcolare il gradiente del coefficiente aerodinamico considerato rispetto al punto di controllo i-esimo come:

$$\begin{aligned} sens_{C_{D_i}} &= \frac{C_{D_i}^+ - C_{D_i}^-}{2\delta_s} \\ sens_{C_{L_i}} &= \frac{C_{L_i}^+ - C_{L_i}^-}{2\delta_s} \end{aligned} \tag{5.4}$$

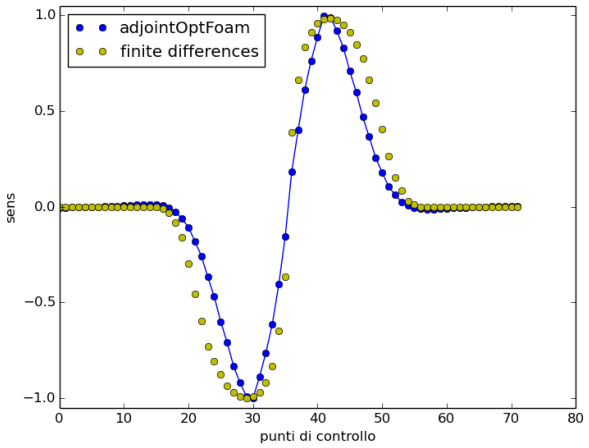
La scelta dell'ampiezza degli spostamenti è una questione critica per questa implementazione: infatti per valori di δs troppo grandi si perde l'ipotesi di linearità che sta alla base della formula utilizzata per il calcolo dei gradienti, mentre se si utilizza un valore di δs troppo piccolo si rischia di introdurre delle modifiche talmente marginali sul profilo di partenza da non andare a modificare i coefficienti aerodinamici. Per il caso in esame, ricordando che la corda del profilo è unitaria, il valore ottimale di spostamento, trovato a seguito di una serie di prove, è di 10^{-4}.

I valori così calcolati sono confrontati con i risultati forniti dal solutore *adjointOptFoam* alla fine del primo passo del ciclo di ottimizzazione, in figura 5.5. Per questo confronto si è deciso di utilizzare un numero sufficientemente elevato di punti di controllo, pari a 72, in modo da mappare con maggior precisione il profilo di sensitività lungo il profilo. In queste due figure, in ascissa è presente unn successione di numeri intero associati alla posizione del punto di controllo, il primo punto si trova sul bordo d'uscita e la numerazione procede in senso antiorario, quindi i primi punti di controllo si trovano sul dorso del profilo.

Dal confronto si può notare che esiste un buon accordo tra l'andamento di sensitività estratto dal solutore in esame e quello calcolato attraverso le differenze finite. Pur non essendoci una sovrapposizione precisa di tutti i valori calcolati in ogni punto di controllo, l'andamento qualitativo calcolato da *adjointOptFoam* rispecchia quello delle differenze finite e, in particolare, c'è una perfetta corrispondenza per quanto riguarda la posizione dei punti di massimo e di minimo.



(A) *Confronto sensitività C_D*



(B) *Confronto sensitività C_L*

FIGURA 5.5: Confronto tra sensitività del solutore e differenze finite

Inoltre, è interessante cercare di dare una spiegazione fisica a questi due andamenti di sensitività ricordando che, per la convenzione assunta di considerare le normali alla superficie entranti nel profilo, un valore di sensitività positiva corrisponde ad uno spostamento entrante nel corpo. Il profilo corrispondente alla minimizzazione del coefficiente di resistenza, essendo sempre positivo, indica che il profilo dovrà ridurre il suo spessore lungo tutta la corda. In particolare, la presenza dei due picchi di sensitività in corrispondenza del bordo d'attacco individua questa zona come regione di maggiore sensibilità alla funzione costo. In più, il fatto che il picco sul dorso sia maggiore del picco sul ventre ci può suggerire che la deformazione sul bordo d'attacco porterà il profilo ad allinearsi con la corrente esterna, inclinata di 2 gradi rispetto ad esso.
Diverso è il caso del profilo relativo alla massimizzazione del coefficiente di portanza in cui, la simmetria del campo di moto ad incidenza nulla, si traduce in una simmetria del profilo di sensitività tra dorso e ventre. Prima di tutto si nota che al ventre del profilo

corrispondono valori di sensitività positive, quindi una deformazione entrante, mentre al dorso valori di sensitività negative, quindi deformazioni uscenti, portando, di fatto, ad un aumento dell'incidenza del profilo.

I trend individuati sono in accordo con l'obbiettivo dell'ottimizzazione quindi, avendo verificato la correttezza del solutore aggiunto, è possibile procedere con sicurezza all'applicazione di un ciclo completo.

Come è già stato sottolineato in precedenza, considerando un numero n_β di punti di controllo, il costo per ottenere il profilo di sensitività per un passo del ciclo attraverso le differenze finite al secondo ordine è di $2n_\beta$ simulazioni, mentre per ottenere la sensitività con il solutore aggiunto *adjointOptFoam* è necessario il costo di sole due simulazioni, con un evidente risparmio in termine di costo computazionale.

5.4 Re = 3000

Nella tabella seguente vengono riportati i valori iniziali dei coefficienti aerodinamici del profilo NACA 0012 corrispondenti ad un numero di Reynolds pari a 3000, ricavati sulla mesh di partenza. Questi sono confrontati, come avvenuto in precedenza, con i valori forniti dal programma Xfoil sullo stesso profilo alle stesse condizioni.

Tabella 5.2: Caratteristiche del profilo di partenza

$\alpha(deg)$	C_D	C_L	$\Delta C_D(xfoil)$	$\Delta C_L(xfoil)$
0	0.0679	0.0	1.4%	0%
2	0.0690	0.090	1.4%	0.1%

Di seguito vengono riportati sinteticamente i risultati ottenuti dalle due ottimizzazioni su C_L e C_D, ai vari angoli di incidenza.

Tabella 5.3: Risultati per la minimizzazione del C_D

$\alpha(deg)$	$C_{D_{opt}}$	ΔC_D
0	0.0646	-4.9%
2	0.0631	-8.6%

Tabella 5.4: Risultati per la massimizzazione del C_L

$\alpha(deg)$	$C_{L_{opt}}$	ΔC_L
0	0.01	-
2	0.150	+67%

Si noterà subito come i risultati raggiunti per la massimizzazione del C_L superino in proporzione quelli per la riduzione del C_D. Ciò è dovuto principalmente a due fattori: prima di tutto bisogna ricordare che nel tentativo di massimizzare il coefficiente di portanza non si ha un limite superiore impostosto dalla fisica, mentre nella minimizzazione del coefficiente di resistenza questo non potrà mai scendere al di sotto dello 0; in più, per quanto riguarda la composizione del coefficiente di resistenza, esiste una componente dovuta agli sforzi viscosi confrontabile con quella di pressione, mentre il coefficiente di portanza è costituito quasi esclusivamente dalla componente di pressione.

Nel seguito di questa sezione sono esposti alcuni risultati qualitativi che permettono di giudicare l'andamento dell'ottimizzazione.

5.4.1 Minimizzazione C_D ad incidenza nulla

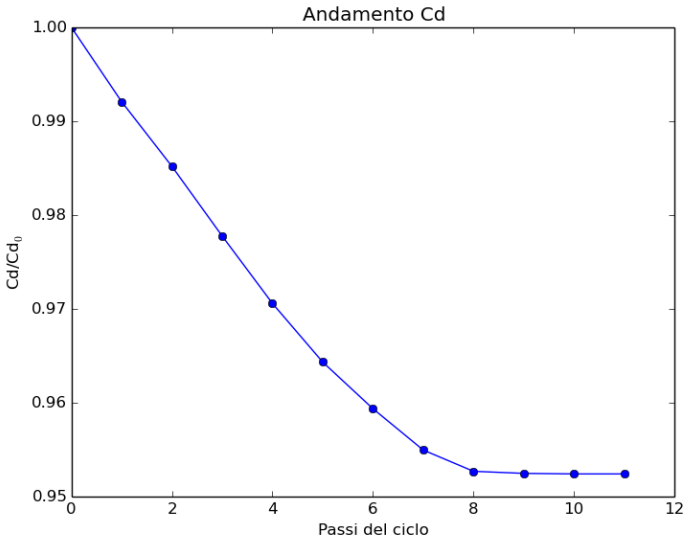


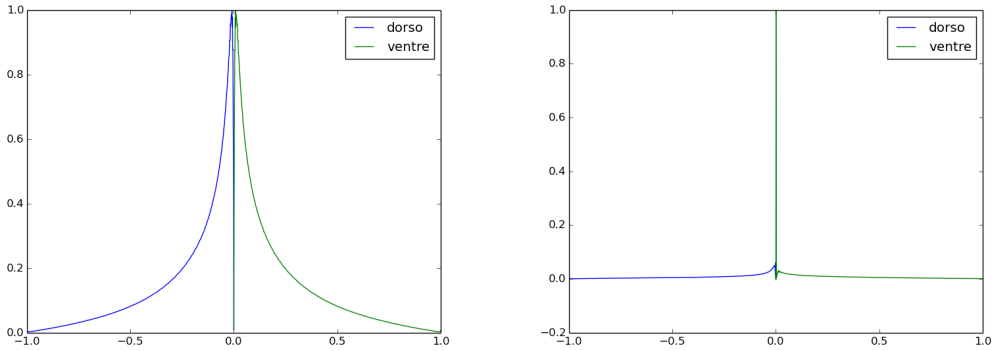
FIGURA 5.6: Risultati del ciclo di minimizzazione C_D

Come primo caso viene analizzato il ciclo relativo alla minimizzazione del coefficiente di resistenza ad incidenza nulla. La convergenza in termini di C_D del ciclo verso il valore finale è riportato in figura 5.6.

Oltre a questi risultati quantitativi, per valutare l'operato del ciclo di ottimizzatore è necessario considerare alcuni risultati qualitativi che vengono riportati inseguito.

Sensitività

In figura 5.7 vengono confrontati i profili di sensitività rispetto alla funzione obbiettivo tra il primo e l'ultimo passo del ciclo.



(A) *Prima iterazione* (B) *Ultima iterazione*

FIGURA 5.7: Sensitività corrispondenti alla minimizzazione del C_D

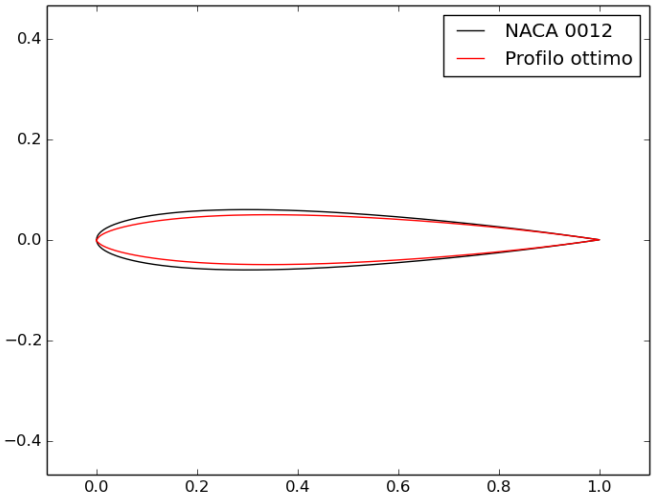
Dato che il suo significato fisico è stato già discusso in precedenza, in questa occasione è interessante notare che, mentre sulla configurazione di partenza i valori di sensitività assumono un valore non nullo lungo tutto il profilo, eccetto che in corrispondenza del punto di bordo d'attacco e di bordo d'uscita, all'ultimo passo il profilo risulta spianato ad un valore nullo, escluso un picco sul bordo d'attacco, dovuto al fatto che la sensitività viene riscalata ogni volta rispetto al suo valore massimo. Il fatto che, all'ultimo passo, il profilo di sensitività, che fornisce le informazioni sulle deformazioni da applicare il profilo, sia ovunque nullo è una conferma ulteriore della buona convergenza raggiunta dal ciclo di ottimizzazione.

Confronto tra profili

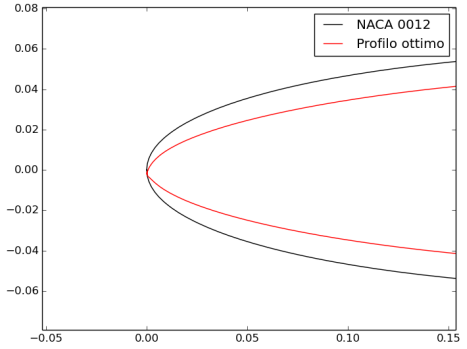
In figura 5.8 è possibile confrontare la forma del profilo ottimizzato rispetto a quella del profilo di partenza. Come ci si aspetta, il trend che si può osservare è quello di una riduzione dello spessore del profilo, mantenendo la simmetria tra dorso e ventre.
Dall'ingrandimento del bordo d'uscita si nota che questo non è soggetto a cambiamenti significativi. Questo è dovuto alla forma delle funzioni di Hicks-Henne con cui viene parametrizzato il profilo, e ha un effetto positivo sul mantenimento di una buona qualità nelle mesh deformate. Infatti, dato che profilo termina con un bordo appuntito, il punto che rappresenta il bordo d'uscita è l'elemento più critico nella deformazione della griglia.

Confronto dei campi di moto

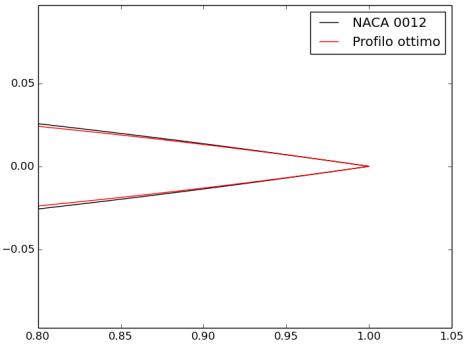
Le figure 5.17 e 5.18 mettono a confronto i campi di velocità e pressione, sia diretti che aggiunti, ottenuti dalle simulazioni sul profilo iniziale e ottimizzato. Questo serve



(A) *Profili*



(B) *Particolare bordo d'attacco*



(C) *Particolare bordo d'uscita*

FIGURA 5.8: Confronto tra profilo di partenza e profilo ottimizzato

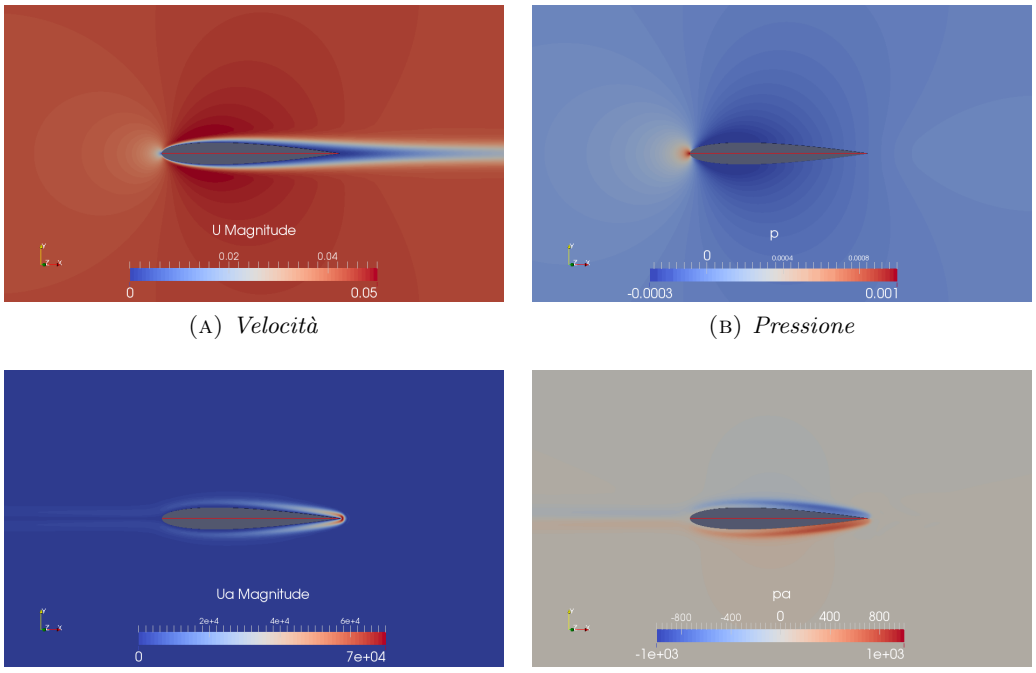
per constatare che le deformazioni apportare al profilo, pur cambiandone la forma, non vanno a modificare radicalmente il campo di moto.

Convergenza delle soluzioni

La qualità delle simulazioni svolte viene confermata analizzando l'andamento della convergenza dei coefficienti aerodinamici e dei residui di velocità e pressione sia per il primo passo del ciclo in figura 5.11, così come all'ultimo passo in figura 5.19.

Confronto dei coefficienti di pressione

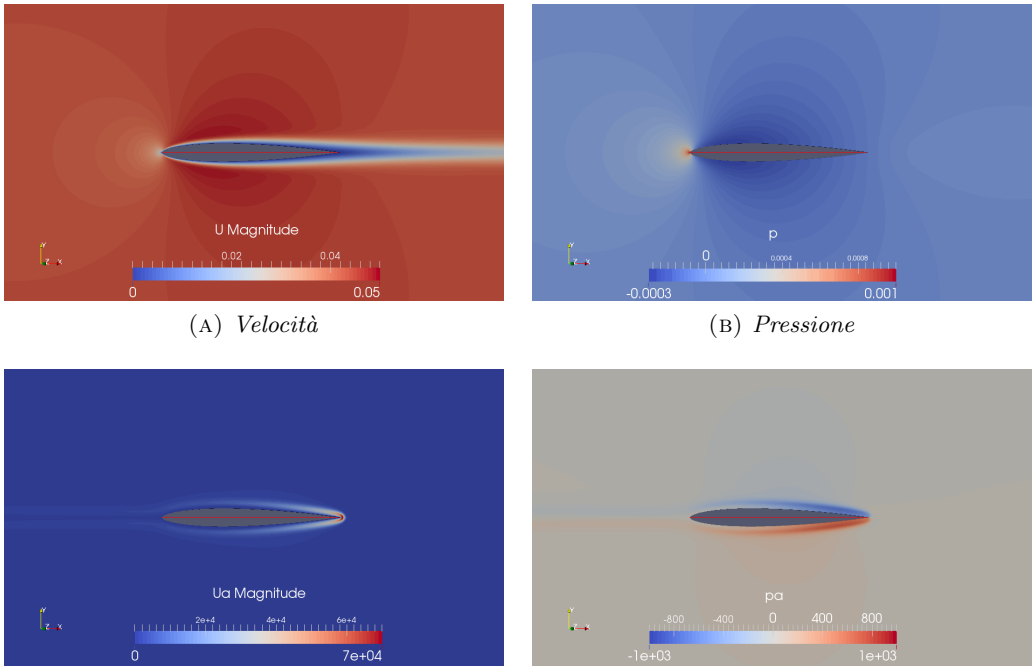
Confrontando i coefficienti di pressione in figura 5.13 si nota che, come effetto della riduzione di pressione, il profilo è soggetto a delle depressioni più contenute di quelle



(A) *Velocità* (B) *Pressione*

(C) *Velocità aggiunta* (D) *Pressione aggiunta*

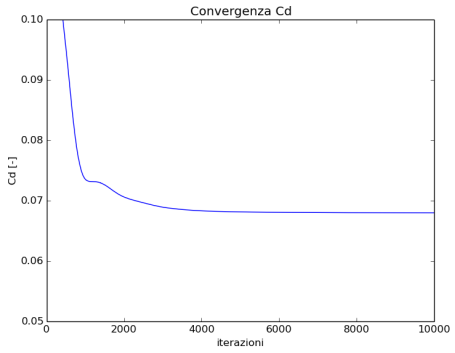
FIGURA 5.9: Campo di moto sul profilo iniziale



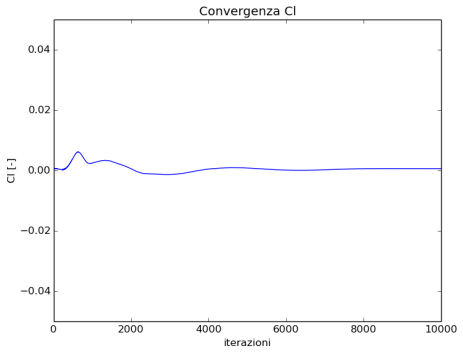
(A) *Velocità* (B) *Pressione*

(C) *Velocità aggiunta* (D) *Pressione aggiunta*

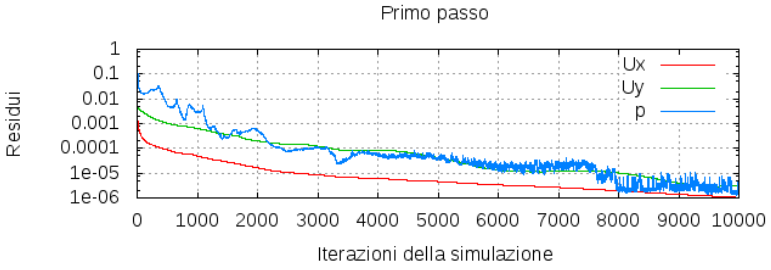
FIGURA 5.10: Campo di moto sul profilo ottimizzato



(A) *Coefficiente di resistenza*

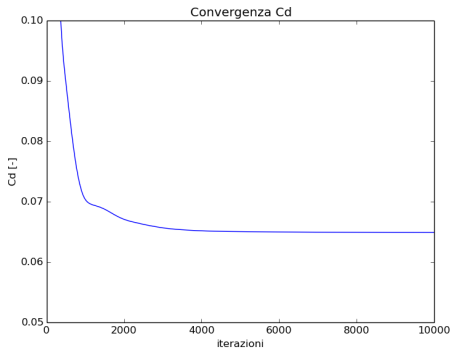


(B) *Coefficiente di portanza*

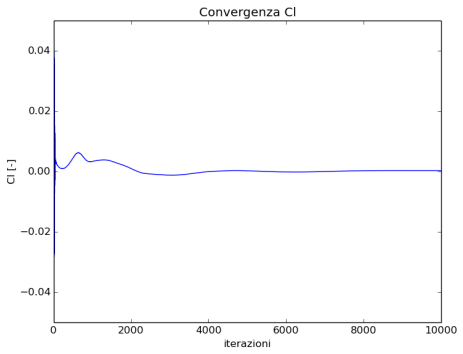


(C) *Residui*

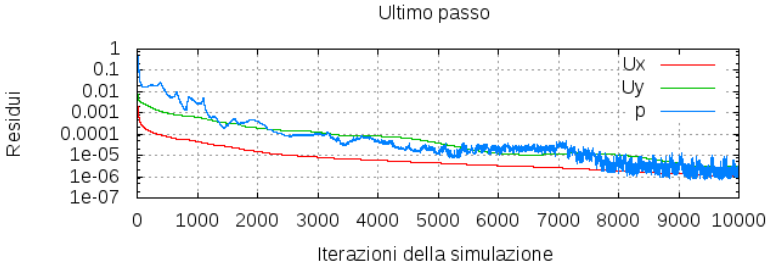
FIGURA 5.11: Convergenza delle soluzioni al primo passo



(A) *Coefficiente di resistenza*



(B) *Coefficiente di portanza*



(C) *Residui*

FIGURA 5.12: Convergenza delle soluzioni all'ultimo passo

che caratterizzano il NACA 0012, pur mantenendo la simmetria di distribuzione della pressione tra dorso e ventre che garantisce una portanza nulla.

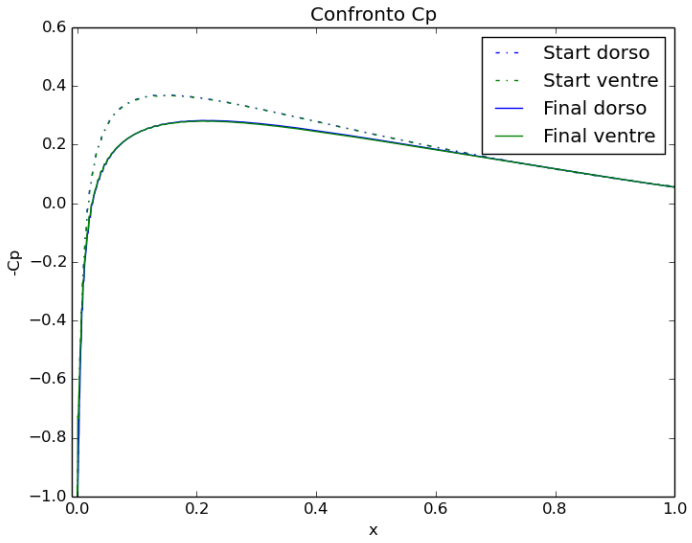


FIGURA 5.13: Confronto tra C_P del profilo iniziale e ottimizzato

5.4.2 Massimizzazione C_L ad incidenza nulla

Visti i buoni risultati per la minimizzazione del C_D è interessante vedere come si comporta l'ottimizzazione nel caso di massimizzazione del C_L nelle stesse condizioni. L'andamento della convergenza del ciclo è mostrata in figura 5.14.

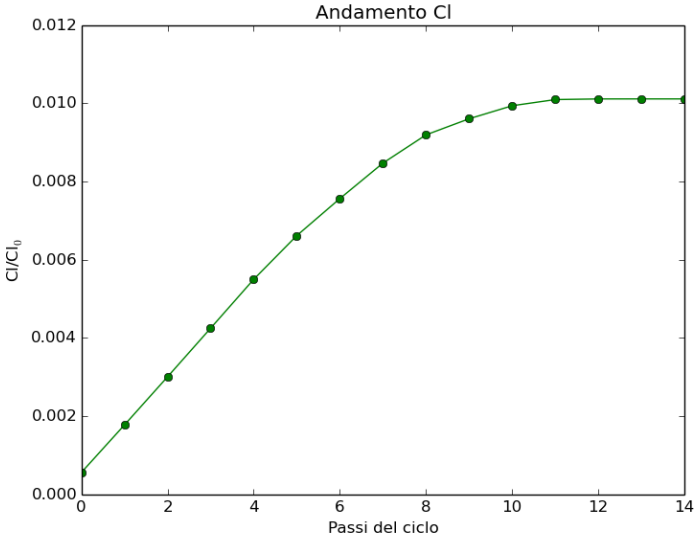
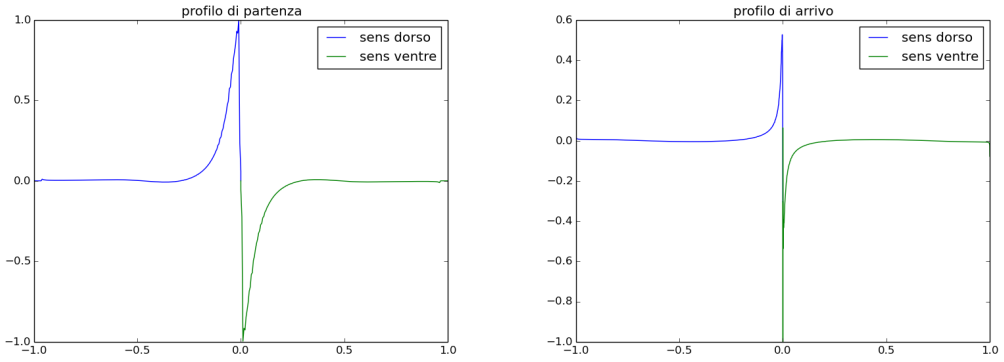


FIGURA 5.14: Risultati del ciclo di massimizzazione del C_L

Sensitività

Dal confronto dei profili di sensitività tra primo e ultimo passo in figura 5.15 si nota che, mentre sulla configurazione iniziale circa il 30% della corda è interessato da valori di sensitività non nulla, nella configurazione finale questa percentuale è ridotta a meno del 5% della corda.



(A) *Prima iterazione* (B) *Ultima iterazione*

FIGURA 5.15: Sensitività corrispondenti alla massimizzazione del C_L

Confronto tra profili

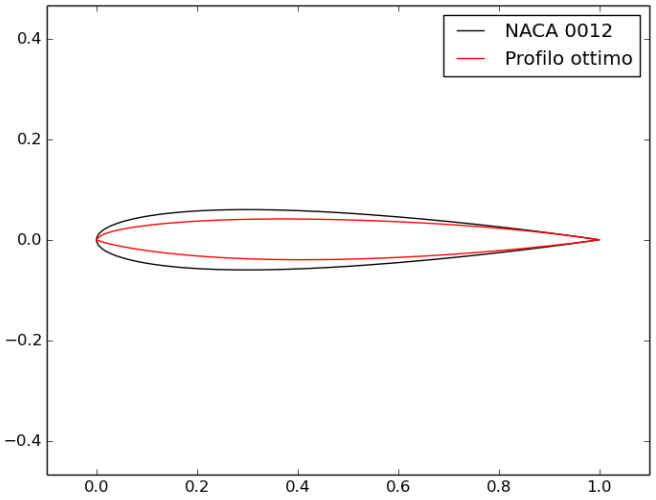
Dal confronto tra profilo iniziale e profilo ottimizzato in figura 5.16 si notano due tendenze principali: da un lato, il profilo perde la propria simmetria riducendo maggiormente lo spessore sul ventre, con l'effetto di portare in incidenza il profilo; dall'altro, per compensare l'aumento di resistenza che deriva dall'incremento dell'angolo d'attacco, lo spessore tende a diminuire.

Confronto dei campi di moto

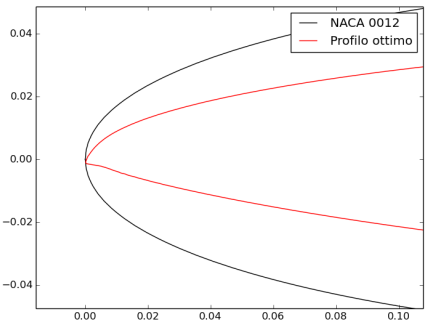
Dal confronto dei campi di moto nelle figure 5.17 e 5.18 si nota come il flusso attorno al profilo ottimizzato non sia più simmetrico tra dorso e ventre, per effetto delle modifiche subite.

Convergenza delle soluzioni

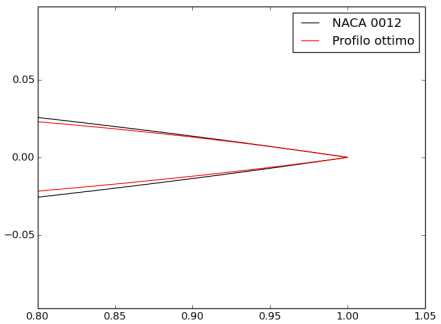
Anche in questo caso l'andamento dei residui e della convergenza dei coefficienti aerodinamici per il passo finale in figura 5.19 conferma la validità delle simulazioni condotte.



(A) *Profili*

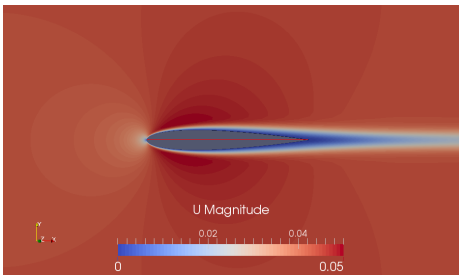


(B) *Particolare bordo d'attacco*

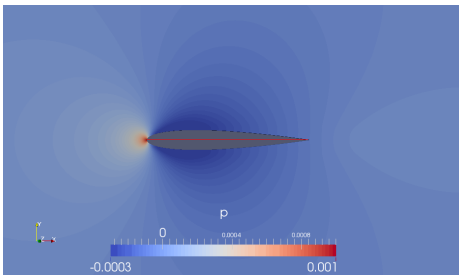


(C) *Particolare bordo d'uscita*

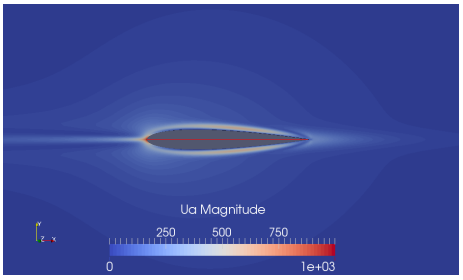
FIGURA 5.16: Confronto tra profilo di partenza e profilo ottimizzato



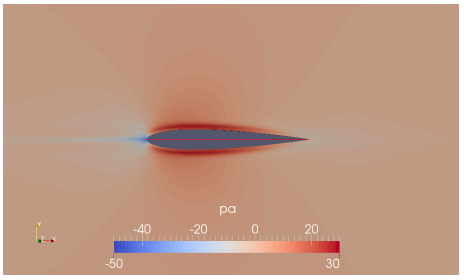
(A) *Velocità*



(B) *Pressione*

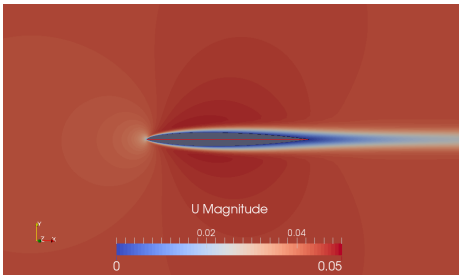


(C) *Velocità aggiunta*

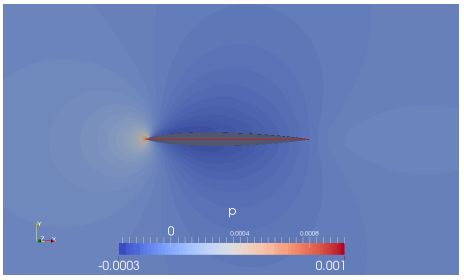


(D) *Pressione aggiunta*

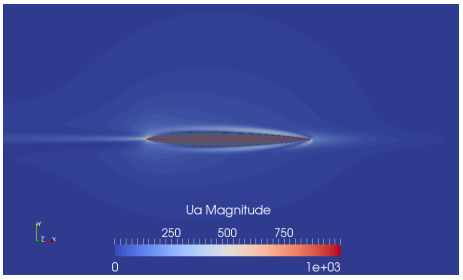
FIGURA 5.17: Campo di moto sul profilo iniziale



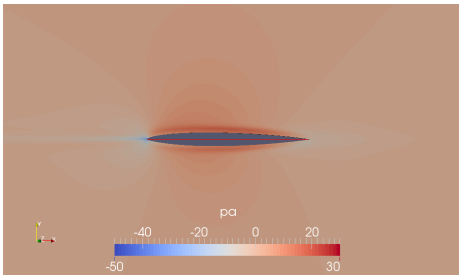
(A) *Velocità*



(B) *Pressione*

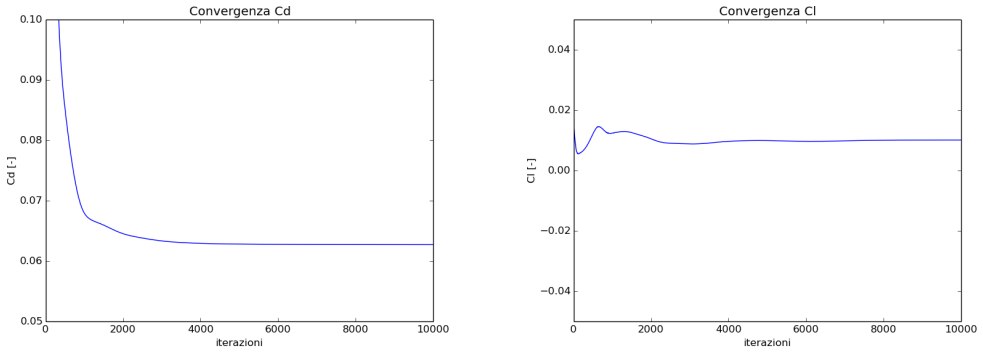


(C) *Velocità aggiunta*

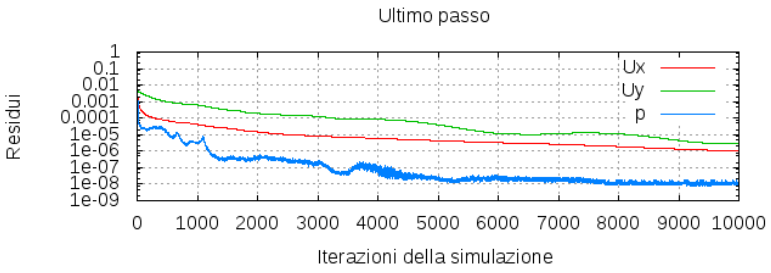


(D) *Pressione aggiunta*

FIGURA 5.18: Campo di moto sul profilo ottimizzato



(A) *Coefficiente di resistenza* (B) *Coefficiente di portanza*



(C) *Residui*

FIGURA 5.19: Convergenza delle soluzioni all'ultimo passo

Confronto dei coefficienti di pressione

Il fatto che il profilo si trovi ad incidenza non nulla è evidenziato dal confronto dei coefficienti di pressione in figura 5.20, infatti la presenza di un'area non nulla tra il profilo associato al dorso e quello associato al ventre indica che il coefficiente di portanza del profilo ottimizzato assume effettivamente un valore positivo.

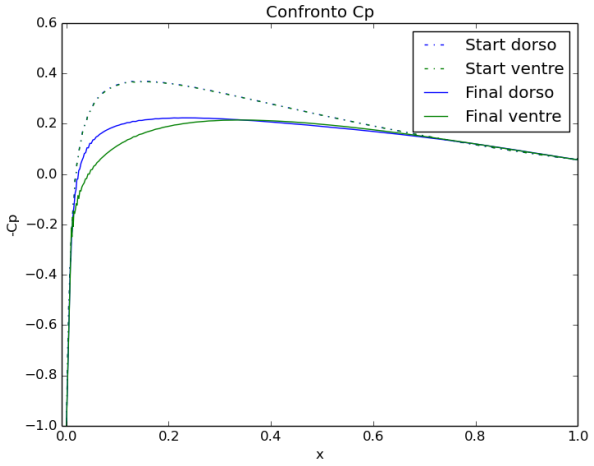


FIGURA 5.20: Confronto tra C_P del profilo iniziale e ottimizzato

5.5 Regime laminare: Re 10000

Per verificare la robustezza del ciclo, nel proseguimento vengono cambiate le condizioni di moto, portando il numero di Reynolds a 10000. La tabella 5.5 mostra il valore iniziale dei coefficienti aerodinamici del profilo NACA 0012 per le condizioni di moto considerate.

TABELLA 5.5: Caratteristiche del profilo di partenza

$\alpha(deg)$	C_D	C_L	$\Delta C_D(xfoil)$	$\Delta C_L(xfoil)$
0	0.0387	0.0	1.8%	0%
2	0.0407	0.0587	2.2%	2.1%

Di seguito vengono riportati sinteticamente i risultati ottenuti dalle due ottimizzazioni su C_L e C_D, ai vari angoli di incidenza.

TABELLA 5.6: Risultati per la minimizzazione del C_D

$\alpha(deg)$	$C_{D_{opt}}$	ΔC_D
0	0.0365	-5.6%
2	0.0372	-8.6%

TABELLA 5.7: Risultati per la massimizzazione del C_L

$\alpha(deg)$	$C_{L_{opt}}$	ΔC_L
0	0.01	-
2	0.113	+93%

Anche in questo caso i risultati ottenuti sono incoraggianti. Sempre nell'ottica di testare la flessibilità del processo, si analizzerà come questo si comporta partendo da una condizione non simmetrica, a 2 gradi di incidenza.

5.5.1 Minimizzazione C_D ad incidenza 2 gradi

La figura 5.21 mostra i risultati in termine di convergenza ottenuti ad ogni passo del ciclo.

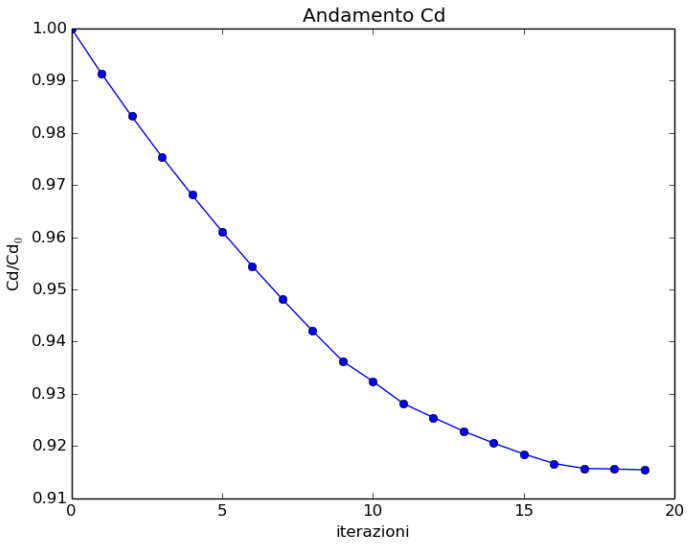
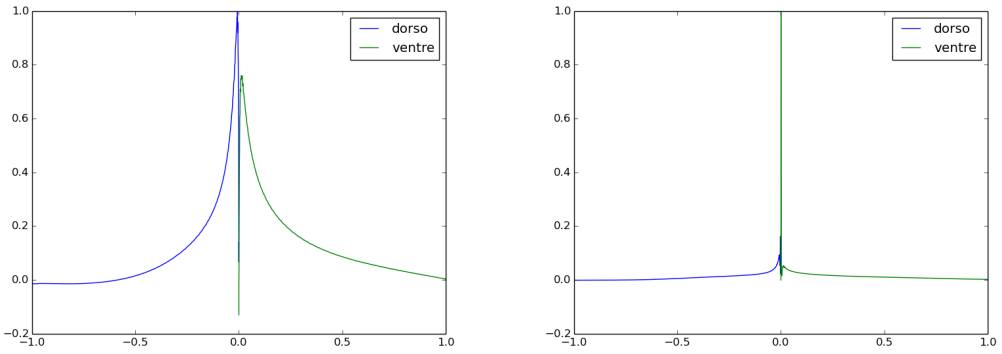


FIGURA 5.21: Risultati del ciclo di minimizzazione del C_D

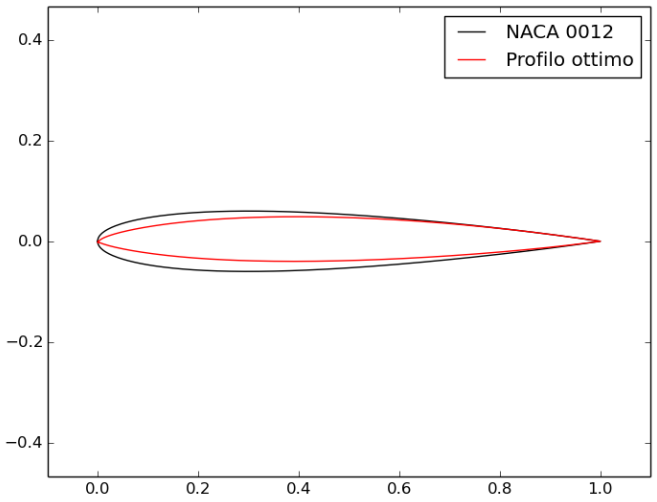
Sensitività

Dal confronto tra sensitività sulla configurazione iniziale e finale in figura 5.22 si nota come sul profilo di partenza il profilo di sensitività assume valori non nulli a partire dal 60% della corda mentre sul profilo finale lo stesso profilo di sensitività è ovunque nulla, ad eccezione del picco sul bordo d'attacco.

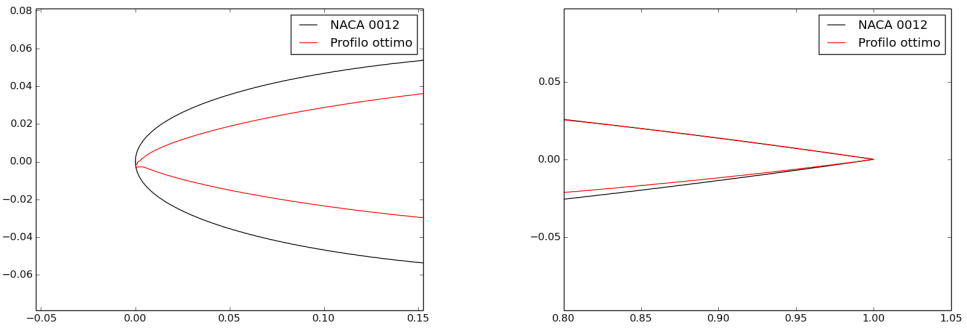


(A) *Prima iterazione* (B) *Ultima iterazione*

FIGURA 5.22: Sensitività corrispondenti alla minimizzazione del C_D



(A) *Profili*



(B) *Particolare bordo d'attacco* (C) *Particolare bordo d'uscita*

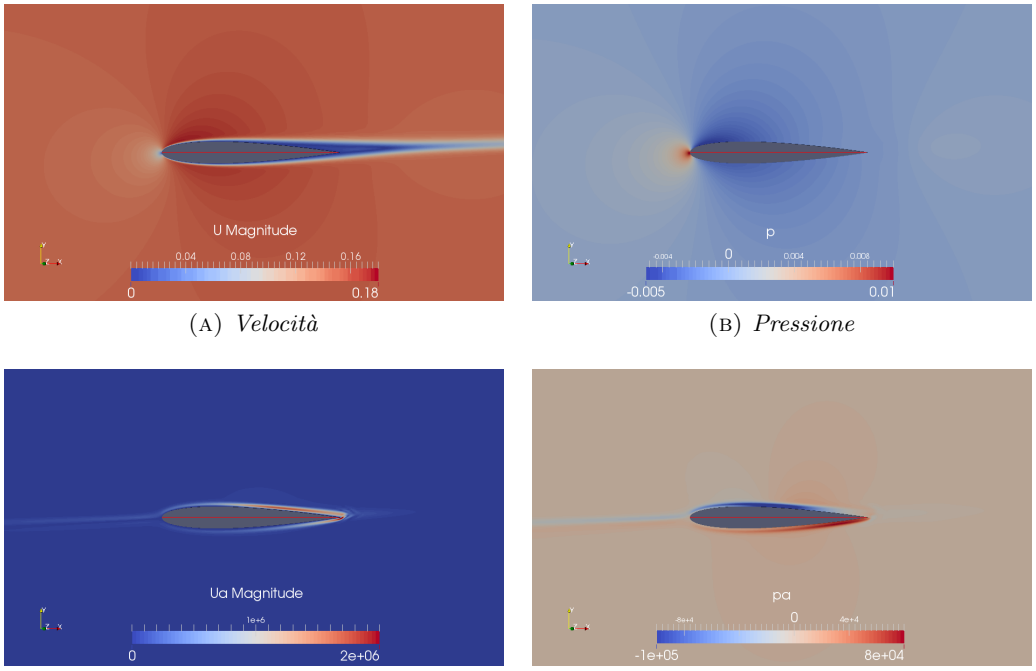
FIGURA 5.23: Confronto tra profilo di partenza e profilo ottimizzato

Confronto tra profili

Come per il caso precedente, la minimizzazione del coefficiente di resistenza comporta una riduzione significativa dello spessore. In particolare in figura 5.23 si nota che il bordo d'attacco è la zona più interessata dalle deformazioni, ed è visibile il tentativo di allineare il profilo con la corrente incidente, inclinando verso il basso la parte anteriore del profilo. Infine, come evidenziato in precedenza, il bordo d'uscita rimane allineato con quello del profilo iniziale.

Confronto dei campi di moto

Dal confronto dei campi di moto nelle figure 5.24 e 5.25 non si osservano cambiamenti radicali introdotti dalla deformazione della mesh.



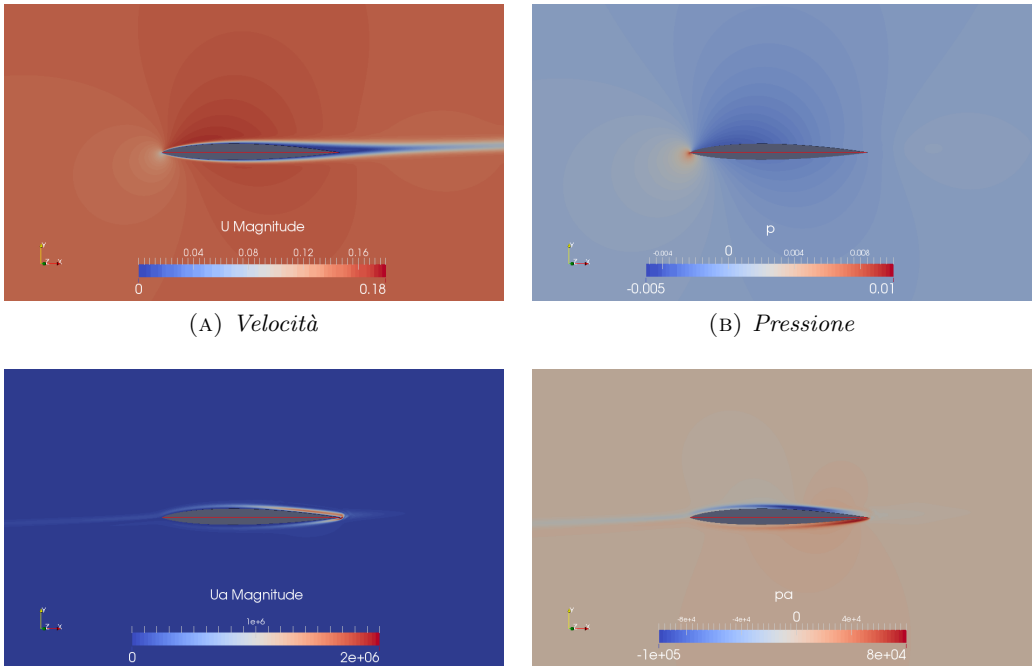
(A) *Velocità* (B) *Pressione*

(C) *Velocità aggiunta* (D) *Pressione aggiunta*

FIGURA 5.24: Campo di moto sul profilo iniziale

Convergenza delle soluzioni

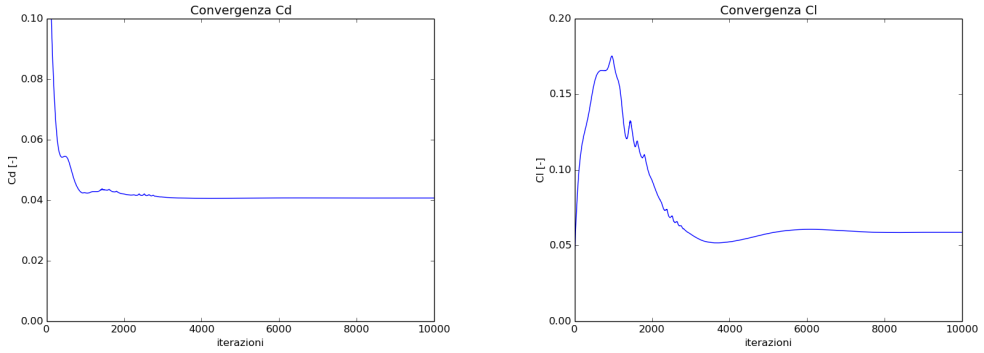
Anche in questo caso è stata garantita una buona convergenza delle soluzioni, verificabile in figura 5.26 e 5.27.



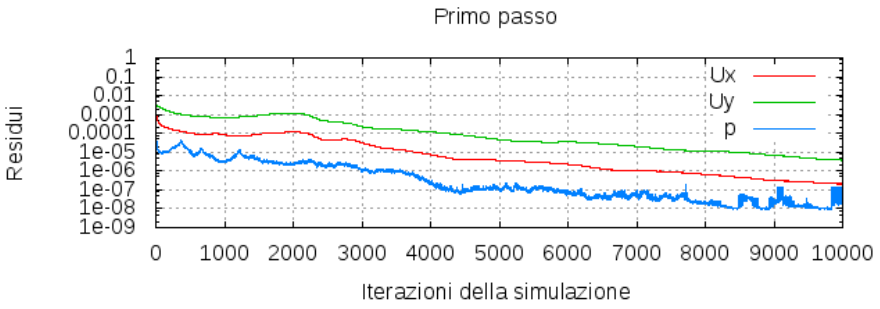
(A) *Velocità* (B) *Pressione*

(C) *Velocità aggiunta* (D) *Pressione aggiunta*

FIGURA 5.25: Campo di moto sul profilo ottimizzato

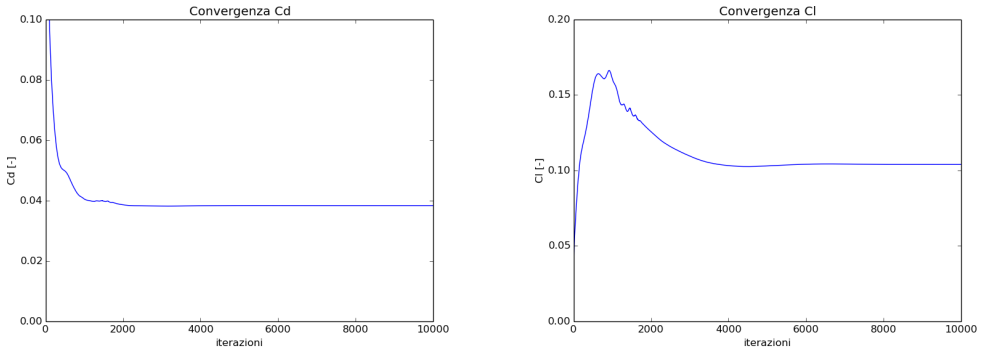


(A) *Coefficiente di resistenza* (B) *Coefficiente di portanza*

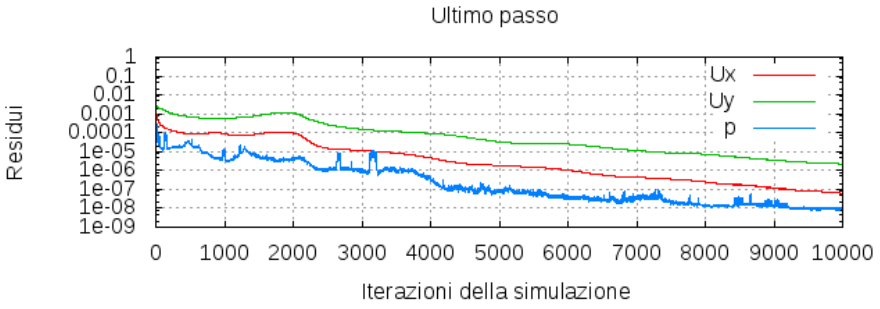


(C) *Residui*

FIGURA 5.26: Convergenza delle soluzioni al primo passo



(A) *Coefficiente di resistenza* (B) *Coefficiente di portanza*



(C) *Residui*

FIGURA 5.27: Convergenza delle soluzioni all'ultimo passo

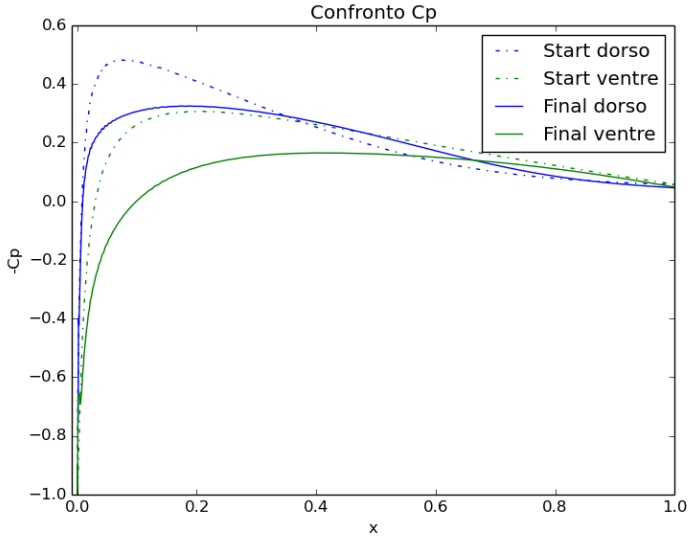


FIGURA 5.28: Confronto tra C_P del profilo iniziale e ottimizzato

Confronto dei coefficienti di pressione

Come osservato per la configurazione a incidenza nulla, la riduzione dello spessore implica una minore depressione sia sul dorso che sul ventre. Tuttavia, come conferma visiva del rispetto del vincolo, in figura 5.28 è evidente che l'area compresa tra le curve di dorso e ventre del profilo ottimizzato non è inferiore a quella compresa tra le curve associate al profilo di partenza.

5.5.2 Massimizzazione C_L ad incidenza 2 gradi

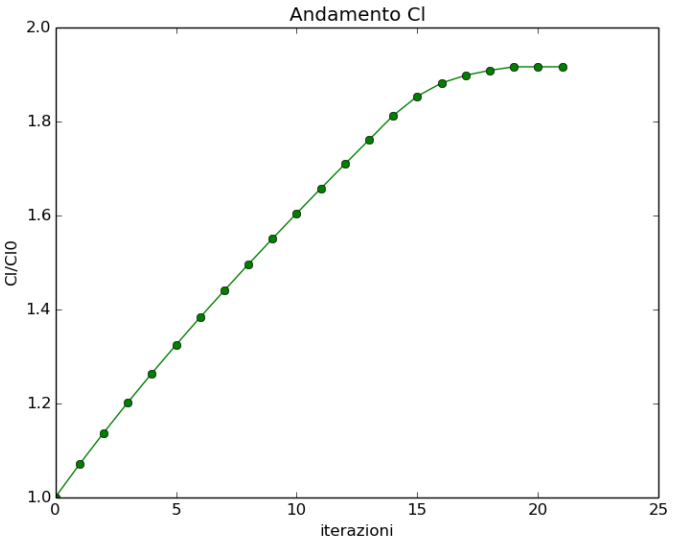


FIGURA 5.29: Risultati del ciclo di massimizzazione del C_L

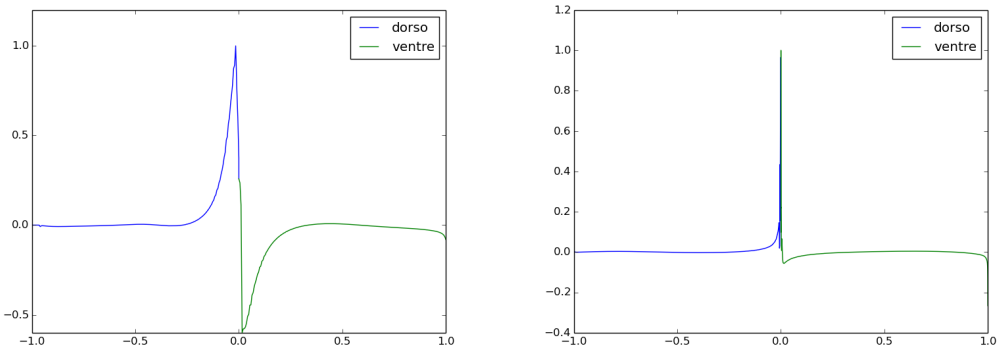
Mantenendo le stesse condizioni di incidenza e numero di Reynolds, si investiga il comportamento per la massimizzazione del coefficiente di portanza.

Sensitività

In figura 5.30 sono confrontati i profili di sensitività del primo passo del ciclo, in cui il 25% della corda del profilo è interessata da valori di sensitività non nulli, e l'ultimo passo del profilo a convergenza, in cui si può osservare che il profilo di sensitività è ovunque nullo, eccetto per il picco sul bordo d'attacco.

Confronto tra profili

Analogamente a quanto visto in precedenza, in figura 5.31 si osserva che la soluzione ricercata dal ciclo per aumentare il coefficiente di portanza è quella di aumentare



(A) *Prima iterazione* (B) *Ultima iterazione*

FIGURA 5.30: Sensitività corrispondenti alla massimizzazione del C_L

l'incidenza del profilo, che viene accompagnata da una diminuzione dello spessore per abbassare il corrispondente coefficiente di resistenza.

Confronto dei campi di moto

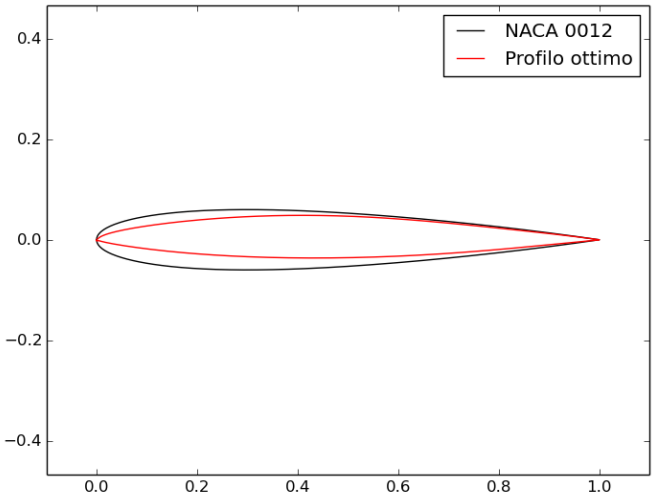
Nelle figure 5.32 e 5.33 vengono confrontati i campi di moto corrispondenti al profilo iniziale e al profilo finale.

Convergenza delle soluzioni

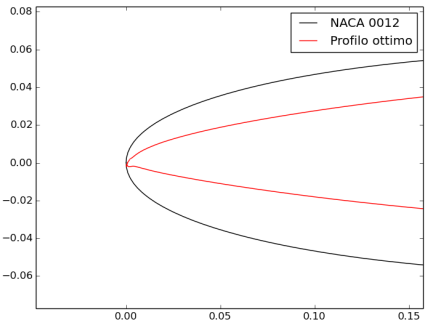
Anche in questo caso è stata garantita una buona convergenza delle soluzioni, riscontrabile in figura 5.34.

Confronto dei coefficienti di pressione

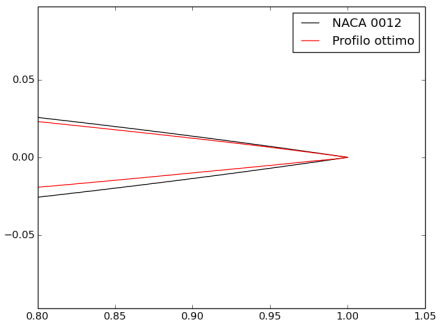
Dal confronto dei due grafici del coefficiente di pressione in figura 5.35, è evidente come l'area compresa tra le due curve associate al profilo finale sia maggiore di quella associata al profilo di partenza.



(A) *Profili*

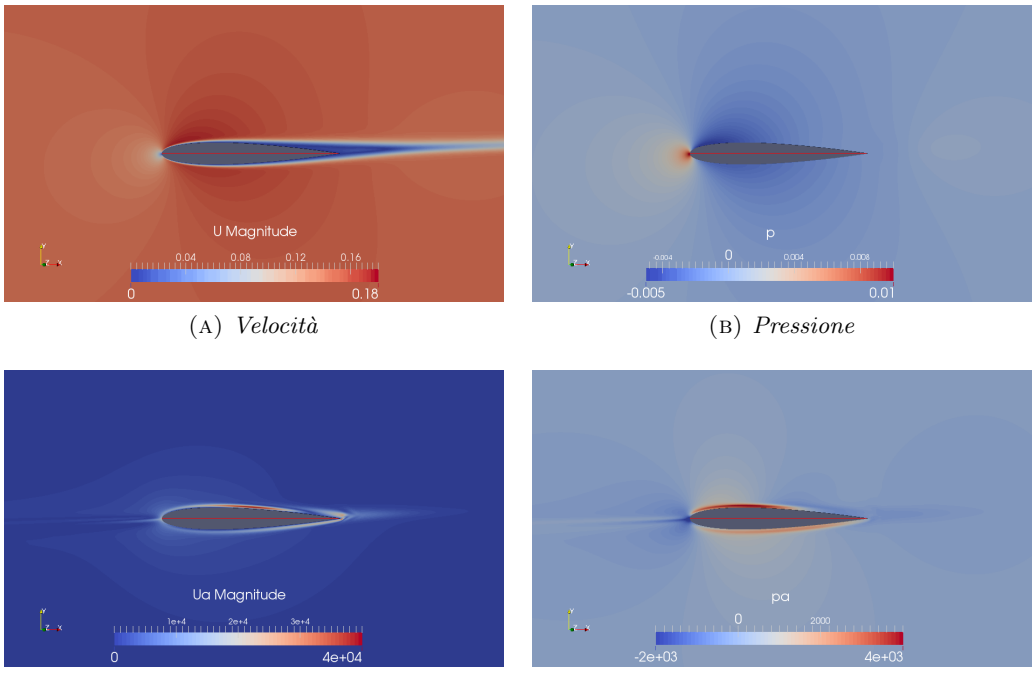


(B) *Particolare bordo d'attacco*



(C) *Particolare bordo d'uscita*

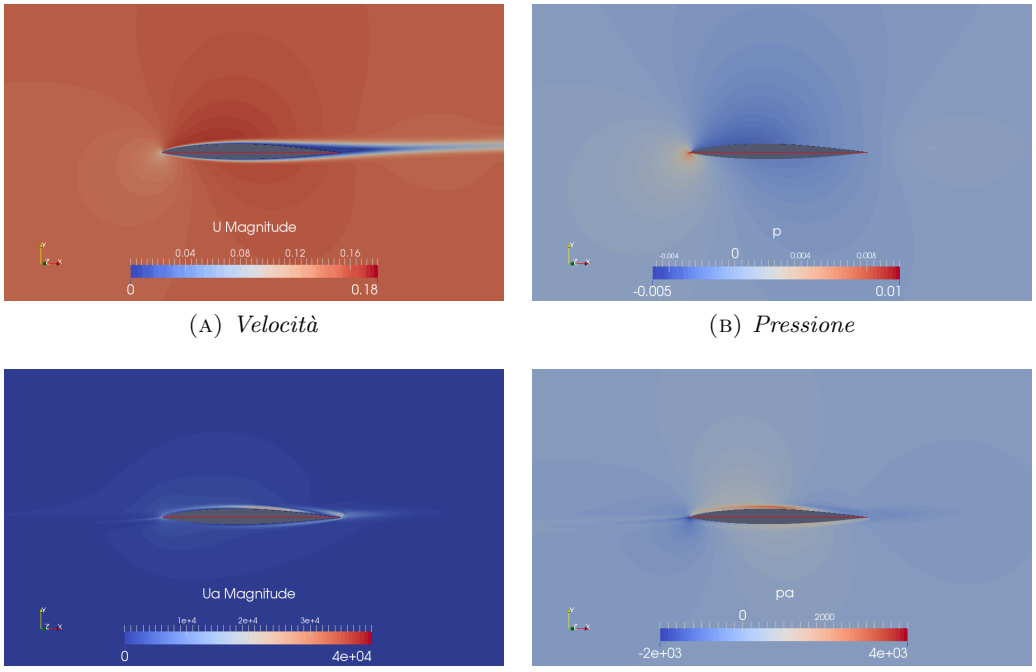
FIGURA 5.31: Confronto tra profilo di partenza e profilo ottimizzato



(A) *Velocità* (B) *Pressione*

(C) *Velocità aggiunta* (D) *Pressione aggiunta*

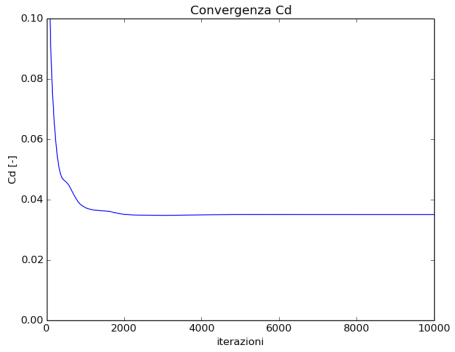
FIGURA 5.32: Campo di moto sul profilo iniziale



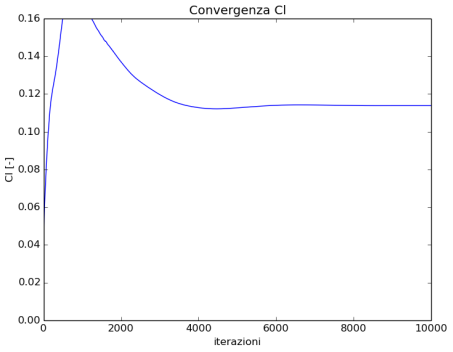
(A) *Velocità* (B) *Pressione*

(C) *Velocità aggiunta* (D) *Pressione aggiunta*

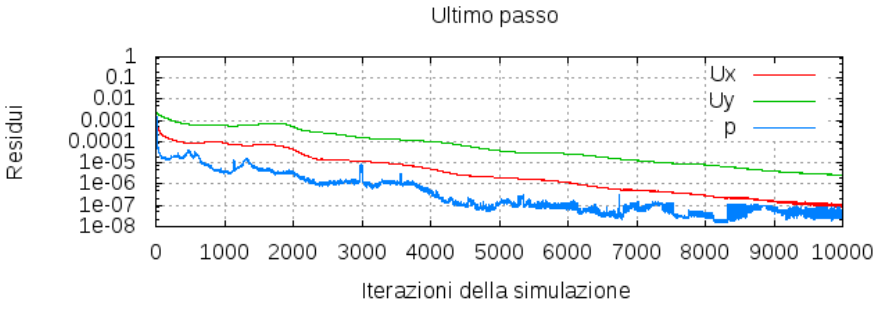
FIGURA 5.33: Campo di moto sul profilo ottimizzato



(A) *Coefficiente di resistenza*



(B) *Coefficiente di portanza*



(C) *Residui*

FIGURA 5.34: Convergenza delle soluzioni all'ultimo passo

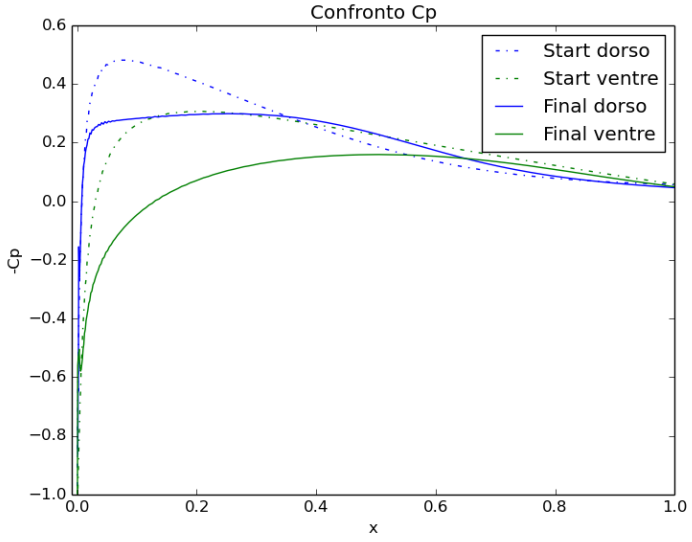


FIGURA 5.35: Confronto tra C_P del profilo iniziale e ottimizzato

5.6 Studio al variare dei punti di controllo

Per scegliere il numero di punti di controllo ottimale per svolgere i cicli proposti in precedenza è stato condotto uno studio per indagare come questo parametro di progetto vada ad incidere sui risultati ottenibili.

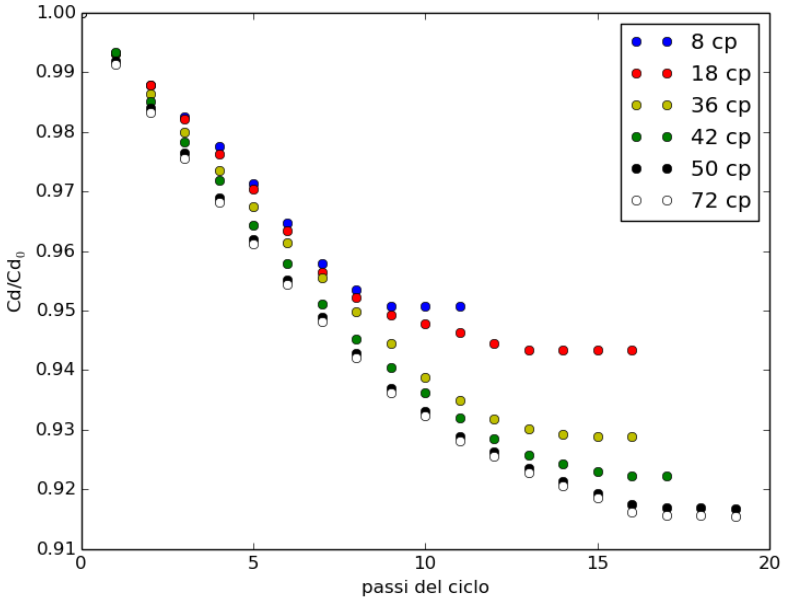


FIGURA 5.36: Risultati al variare del numero di punti di controllo

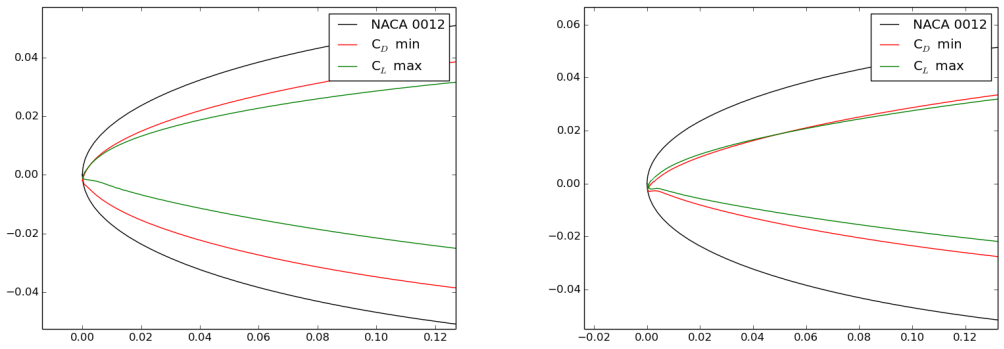
Dalla figura 5.36 si osserva che, in generale, aumentare il numero dei punti di controllo permette di ottenere una riduzione maggiore del coefficiente di resistenza. Questo risultato è intuitivo, in quanto aumentando il numero di parametri di design si amplia lo spettro di configurazioni possibili all'interno di cui il processo in questione è in grado di muoversi, e quindi si hanno maggiori possibilità di raggiungere un risultato migliore. Il ciclo con 8 punti di controllo si interrompe anticipatamente rispetto agli altri per problemi di scarsa qualità della mesh deformata. Questo indica che un numero eccessivamente ridotto di punti di controllo, oltre a precludere il raggiungimento del miglior risultato possibile in termini dell'ottimizzazione della funzione obbiettivo, comporta un peggioramento qualitativi dei profili ottenuti attraverso la deformazione. Allo stesso tempo, si può constatare che le curve associate ai 50 e ai 72 punti di controllo tendono a sovrapporsi, questo porta a concludere che inserire più di 50 punti di controllo non si traduce in un guadagno dal punto di vista dei risultati ottenibili. Queste considerazioni stanno alla base della scelta di condurre le simulazioni sopra presentate con questo numero dei punti di controllo.

5.7 Discussione dei risultati

Sulla base dei risultati esposti in precedenza, si può concludere che il ciclo di ottimizzazione tende a percorrere due strade diverse a seconda della funzione obbiettivo:

- la minimizzazione del C_D è perseguita attraverso una riduzione progressiva dello spessore e un allineamento del bordo d'attacco con il flusso esterno;
- la massimizzazione del C_L è perseguita attraverso un inarcamento della linea media che porta il profilo ad incrementare la propria incidenza.

Ciò è reso evidente dalle immagini di figura 5.37 in cui vengono messi a confronto i profili finali dei rispettivi cicli alle stesse condizioni di numero di Reynolds e incidenza.



(A) *Re = 3000, α = 0 deg* (B) *Re = 10000, α = 2 deg*

FIGURA 5.37: Confronto tra massimizzazione C_L e minimizzazione C_D

Le configurazioni di profilo alare ottenute a convergenza del processo raggiungono un ottimo della funzione obbiettivo nel rispetto dei vincoli imposti, inoltre risultano sensate per quanto riguarda la fisica del problema e realizzabili praticamente, grazie alla regolarità delle forme dei profili e il mantenimento di un certo spessore lungo tutta la corda, garantita dalla presenza dei vincoli stessi.
Per questi motivi si può concludere che il ciclo di ottimizzazione sviluppato per questo caso bidimensionale risulta uno strumento efficace, robusto e flessibile a diversi regimi di moto.

Capitolo 6

Sviluppi futuri

Il presente lavoro rappresenta solo il primo passo verso l'obbiettivo di sviluppare un ciclo di ottimizzazione robusto in grado di relazionarsi a problemi ingegneristici.

In una tesi sviluppata parallelamente a questa, è stato utilizzato lo stesso ciclo di ottimizzazione sviluppato in questo lavoro applicato a flussi turbolenti. I risultati ottenuti hanno evidenziato che, anche con l'introduzione dei modelli di turbolenza, lo stesso processo di ottimizzazione basato sul solutore aggiunto sia in grado di portare a termine con successo gli stessi tipi di ottimizzazione effettuate sul profilo alare NACA 0012.

Grazie a questi risultati, nel passo logicamente successivo, dovrà essere introdotta la possibilità di trattare geometrie tridimensionali. Mentre il solutore per le equazioni aggiunte e, in generale, tutta la parte di simulazione affidata ad OpenFOAM non necessita di cambiamenti per effettuare questo passaggio, è possibile che il resto del ciclo o parte di esso debba essere modificato o, perlomeno, verificato. In particolare emergono due questioni principali:

- *Parametrizzazione*: le funzioni di Hicks-Henne si sono rivelate un'ottima scelta per la parametrizzazione di un profilo alare, ovviamente questo non continua ad essere vero in generale per ogni corpo in 3D. Per questo motivo sarebbe opportuno sviluppare una parametrizzazione della geometria di riferimento in grado di adattarsi ad ogni applicazione. Una soluzione potrebbe essere quella di implementare il metodo *free-form deformation* (FFD), metodo utilizzato dal software SU^2.
- *Deformazione della mesh*: è stato già evidenziato in precedenza come il tool di OpenFOAM per la deformazione della mesh abbia difficoltà nel gestire importanti deformazioni della mesh rispetto alla condizione di partenza. Questo problema potrebbe amplificarsi nel caso tridimensionale, portando alla necessità di abbandonare il tool *moveDynamicMesh* a favore di uno strumento più efficiente.

Per quanto riguarda il software di ottimizzazione, il suo utilizzo non viene modificato. Tuttavia, è stato verificato nel corso di questo lavoro che l'ottimizzatore di Dakota non è in grado di gestire più di una funzione vincolo, quindi questa limitazione è una cosa da tenere presente nel caso si abbia la necessità di sviluppare un ottimizzazione più complessa.

Appendice A

Sviluppi della metodologia

La metodologia illustrata al capitolo 4 è il frutto di un percorso in cui, nel tentativo di costruire un ciclo di ottimizzazione basato sul metodo aggiunto, sono state percorse diverse strade, alcune delle quali hanno evidenziato dei problemi che ne hanno decretato l'abbandono in favore del procedimento che è stato adottato in questo lavoro. Di seguito vengono riportati i passi di questo processo di *trial and error*, con l'intento di mostrare i principali problemi che si possono incontrare nell'implementazione *ex-novo* di un ciclo di ottimizzazione e di proporre eventuali soluzioni al riguardo.

A.1 Scelta delle variabili di progetto

Dato che il solutore aggiunto che viene utilizzato fornisce come output i valori di sensitività alla funzione costo in tutti i punti della mesh con cui viene discretizzato il profilo, la decisione più intuitiva nella scelta delle variabili di progetto sembrerebbe essere quella di prendere tutti i punti a disposizione. Per questo motivo, questa strada è stata temporalmente la prima ad essere seguita.

Per generare una mesh in grado di risolvere lo strato limite in corrispondenza del corpo, si è dovuto discretizzare il profilo con 500 punti di griglia. Pensando di dover estendere questo procedimento a geometrie più complesse, appare evidente che ciò si tradurrebbe in un eccessivo numero di variabili per il ciclo di ottimizzazione. Inoltre, mantenere così tanti gradi di libertà, nel momento in cui si passa alla deformazione del profilo, porta alla generazione di superfici dalla forma piuttosto irregolare, proprio per la possibilità che viene lasciata a punti adiacenti di potersi muovere in maniera svincolata tra di loro. Questi problemi contribuiscono, in aggiunta, a degradare la qualità della mesh di arrivo, costringendo il ciclo di ottimizzazione ad arrestarsi dopo pochi passi.

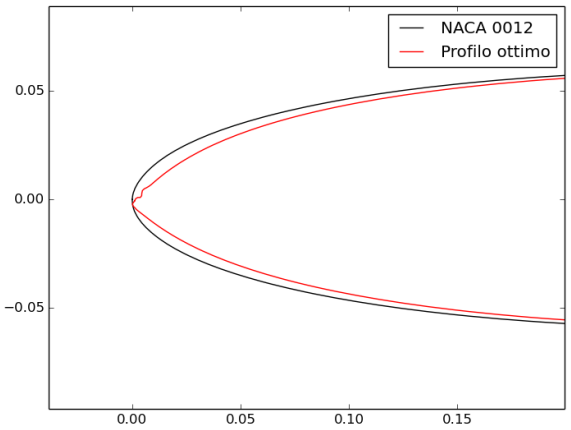


FIGURA A.1: Particolare del bordo d'attacco del confronto tra profilo iniziale e migliorato

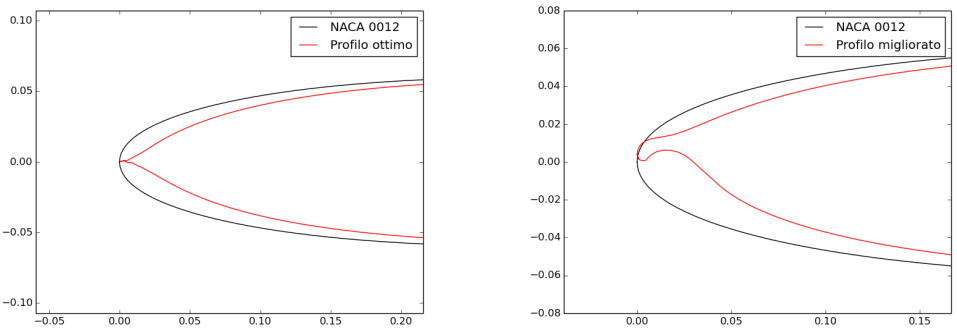
In figura A.1 è riportato un esempio del risultato di un ciclo di ottimizzazione condotto a Re 3000 e incidenza 2 gradi utilizzando come variabili di progetto tutti i punti della mesh. I problemi di deformazione risultano essere evidenti soprattutto sul bordo d'attaco, dove i valori di sensitività, e quindi anche i relativi spostamenti, sono maggiori.

Questi problemi possono essere risolti inserendo sulla superficie del profilo un certo numero di punti di controllo, il cui spostamento determina una deformazione dell'intero profilo. Proprio la dipendenza tra lo spostamento di un punto di controllo e la deformazione indotta sul resto del profilo apre spazio a diversi scenari.

A.2 Interpolazione con RBF

La prima soluzione che si è deciso di adottare era basata esclusivamente sulle radial basis function(RBF). Secondo l'approccio seguito, la sensitività fornita in output dal solutore aggiunto calcolata su tutti i punti di griglia veniva interpolata nei punti di controllo tramite le funzioni RBF e, allo stesso modo, gli spostamenti forniti dall'ottimizzatore di Dakota venivano reinterpolati su tutti i punti del profilo.

Nelle figure A.2 vengono riportati due risultati ottenuti con questa strategia, il primo con l'obbiettivo di minimizzare il coefficiente di resistenza, e il secondo di massimizzare il coefficiente di portanza. Da esse si può concludere che l'introduzione dei punti di controllo ha risolto i problemi di regolarità dei profili deformati, tuttavia emerge un nuovo problema: se lasciati liberi di muoversi secondo il profilo di sensitività, le deformazioni tendono a concentrarsi unicamente sul bordo d'attacco, dove la sensitività assume i valori maggiori. Nel primo caso, il trend dell'ottimizzazione sta portando il



(A) *Minimizzazione* C_D (B) *Massimizzazione* C_L

FIGURA A.2: Risultati con RBF: particolari del bordo d'attacco

profilo ad assomigliare ad una lamina piana, soluzione degenere per la minimizzazione della resistenza, mentre nel secondo caso è come se l'ottimizzatore cercasse di creare un ipersostentatore di bordo d'attacco per incrementare la portanza. Questi aspetti possono diventare critici per quanto riguarda la realizzabilità del risultato dell'ottimizzazione, per questo uno strumento di ottimizzazione valido deve essere in grado di evitare questo tipo di configurazioni.

Questo è il motivo che ha portato all'introduzione della parametrizzazione del profilo attraverso le funzioni di Hicks-Henne: proprio la forma di queste funzioni garantisce che per ogni spostamento dei punti di controllo si otterrà una geometria regolare e realizzabile.

Di fatto, questa rappresenta la metodologia che ha portato i risultati migliori, sia dal punto di vista quantitativo che qualitativo, tra tutte quelle che sono state percorse, motivo per cui è stata adottata per l'implementazione finale del ciclo di ottimizzazione.

Appendice B

Confronto con SU^2

Come anticipato nell'Introduzione 1, il software SU^2 ha rappresentato un punto di riferimento per questo lavoro, per questo sarebbe interessante poter confrontare i risultati ottenuti con il software dell'università di Stanford e con il ciclo sviluppato in questa tesi partendo dallo stesso problema iniziale. Nella *home* page del codice SU^2 è possibile trovare una sezione dedicata a semplici tutorial [17] che permettono all'utente di acquisire familiarità con le capacità del software: tra questi, nella *release* 3.2, è presente un problema di minimizzazione del coefficiente di resistenza di un profilo NACA 0012, ad un numero di Reynolds di 3000, numero di Mach pari a 0.3 e incidenza 2 gradi, con la possibilità di inserire un vincolo sul coefficiente di portanza. Grazie al basso numero di Mach, che rende la correzione sui coefficienti aerodinamici dovuta agli effetti di comprimibilità trascurabile, è possibile confrontare questo tutorial con il relativo caso analizzato in questo lavoro. Nonostante questa analogia, il modello di equazioni utilizzato in SU^2 per l'aerodinamica comprimibile risulta diverso dal modello incomprimibile utilizzato per questa tesi, quindi ci si aspetta la presenza di qualche elemento di diversità tra le due soluzioni.

Nella tabella B.1 vengono confrontati i valori del coefficiente di resistenza, calcolato nelle stesse condizioni di moto, da SU^2 e da OpenFOAM. Si nota che la differenza tra i due valori è inferiore del 2.2%, dato che attesta l'uniformità della condizione di partenza del processo di ottimizzazione.

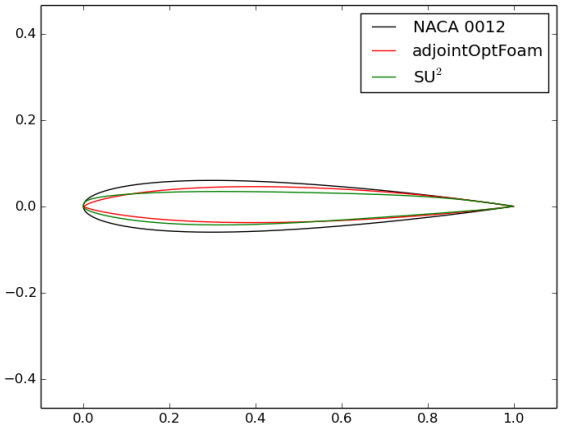
Tabella B.1: Confronto sui profili iniziali

software	C_D
SU^2	0.0707
OpenFOAM	0.0690

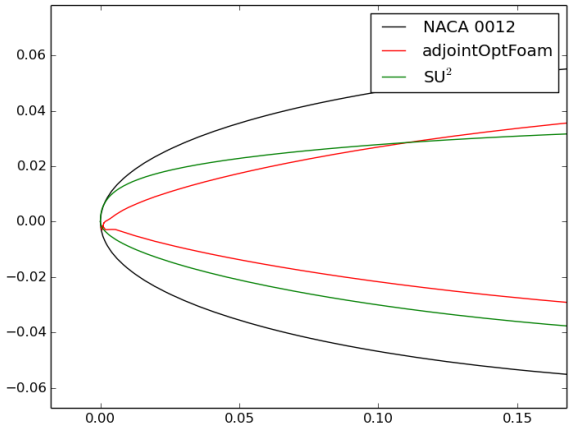
TABELLA B.2: Confronto sui profili ottimizzati

software	C_D	ΔC_D
SU^2	0.0657	-7.1%
OpenFOAM	0.0631	-8.6%

Nella tabella B.2 viene riportato il risultato dell'ottimizzazione eseguita da SU^2 confrontati con i risultati ottenuti in precedenza. Si nota come il ciclo sviluppato per questo lavoro sia in grado di ottenere dei risultati confrontabili con SU^2 dal punto di vista del valore assunto dalla funzione obbiettivo, raggiungendo un valore finale leggermente migliore.



(A) *Profili*



(B) *Particolare del bordo d'attacco*

FIGURA B.1: Confronto tra ciclo sviluppato e SU^2

In figura B.1 si possono notare anche alcune differenze qualitative nei profili ottimizzati

ottenuti: le modifiche apportate dall'ottimizzazione guidata da SU^2 tendono sostanzialmente a mantenere il profilo simmetrico assottigliando lo spessore del profilo uniformemente lungo la corda, mentre il ciclo sviluppato, insieme alla riduzione dello spessore, tende a curvare la linea media del profilo in modo da allineare il bordo d'attacco con la corrente esterna.
Dai risultati ottenuti si può affermare che quest'ultima strategia rappresenta la più efficiente nell'ottica della simulazione, mostrando le potenzialità associate allo strumento sviluppato.

Queste diversità possono essere spiegate dal fatto che il ciclo presentato è stato implementato nell'ottica di risolvere problemi di ottimizzazione aerodinamica incomprimibile, come di fatto è quello proposto, mentre il codice SU^2 è ottimizzato per affrontare problemi di aerodinamica incomprimibile in regime transonico. In questa differenza di problema di riferimento si trova la diversità principale tra questi due codici e trova quindi giustificazione il progetto seguito in questa tesi.

Bibliografia

[1] J. E. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers and Fluids*, 39:373–391, December 2010.

[2] S. K. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA paper 2000-0667*, 2000.

[3] . URL http://su2.stanford.edu/index.html.

[4] URL http://www.cineca.it/it/content/galileo.

[5] C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *Int. J. Numer. Meth. Fluids*, 58:861–877, January 2008.

[6] C. A. Marta and S. Shankaran. On the handling of turbulence equations in rans adjoint solvers. *Computer and Fluids*, 74:102–113, March 2013.

[7] C. Soto and R. Löhner. On the boundary computation of flow sensitivities. *AIAA paper 2004-112*, January 2004.

[8] URL www.openfoam.com.

[9] C. Othmer, E. de Villiers, and H. G. Weller. Implementation of a continuous adjoint for topology optimization of ducted flows. *18th AIAA Computational Fluid Dynamics Conference*, June 2007.

[10] B. M. Adams, L. E. Bauman, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, P. D. Hough, K. T Hu, J. D. Jakeman, L. P. Swiler, and D. M. Vigil. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.2 user's manual. *Sandia Technical Report SAND2010-2183*, December 2009. URL https://dakota.sandia.gov/content/manuals.

[11] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *J. Aircraft*, 15:407–412, July 1978.

[12] M. Drela. Xfoil: An analysis and design system for low reynolds number airfoils. URL http://web.mit.edu/drela/Public/web/xfoil/.

[13] M. Drela. Two-dimensional transonic aerodynamic design and analysis using the euler equations. *Massachusetts Institute of Technology, Gas Turbine Laboratory Rept.*, February 1986.

[14] L. Quartapelle and F. Auteri. Fluidodinamica incomprimibile. *Casa Editrice Ambrosiana*, pages 244–252, 2013.

[15] M. Drela and M. B. Giles. Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA Journal*, 25(10):1347–1355, October 1987.

[16] J. Brezillon and N. R. Gauger. 2d and 3d aerodynamic shape optimization using the adjoint approach. *Aerospace science and technology*, 8:715–727, September 2004.

[17] . URL http://su2.stanford.edu/training.html.

[18] M. Schramm, B. Stoevesandt, and J. Peinke. Adjoint optimization of 2d-airfoils in incompressible flows. *11th World Congress on Computational Mechanics*, November 2014.

[19] M. B. Giles and N. A. Pierce. On the handling of turbulence equations in rans adjoint solvers. *Flow, Turbulence and Combustion*, 65:393–415, 2000.

[20] C. Othmer, T. Kaminski, and R. Giering. Computation of topological sensitivities in fluid dynamics: cost function versatility. *European Conference on Computational Fluid Dynamics*, 2006.

[21] Z. Lyu, Z. Xu, and J. Martins. Benchmarking optimization algorithms for wing aerodynamic design optimization. *The Eighth International Conference on Computational Fluid Dynamics*, July 2014.

[22] F. Palacios, M. R. Colonno, A. C. Aranake, A. Campos, S. R. Copeland, T. D. Economon, A. K. Lonkar, T. W. Lukaczyk, and J. J. Alonso. Stanford university unstructured (su2): An open-source integrated computational environment for multi-physics simulation and design. *AIAA Paper 2013-0287, 51st AIAA Aerospace Sciences Meeting and Exhibit*, January 2013.