Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di laurea

Strategia *Multi-Fidelity* applicata alla CFD di un flusso transonico attorno a un profilo alare

Davide Iannotta

Relatore: Prof.ssa Sandra Pieraccini
Candidato: Davide Iannotta, 254383

Sommario

In questo lavoro di tesi si applica una strategia *Multi-Fidelity* che ricerca un compromesso tra precisione ed efficienza di calcolo a delle simulazioni CFD attorno a un profilo supercritico in condizioni di flusso transonico con i parametri di *inlet* che variano con una distribuzione di probabilità di tipo *Beta*, per modellare stocasticamente i piccoli e repentini cambiamenti nelle condizioni di monte dovuti ad esempio a una raffica.
Questo algoritmo numerico è stato recentemente introdotto (2014) ed aspira ad un approccio sinergico che possa giovare delle migliori caratteristiche di un modello piuttosto grossolano detto *Low-Fidelity* ed uno decisamente migliore detto *High-Fidelity*, per ottenere una soluzione surrogata *High-Fidelity* affidabile ma molto meno costosa in termini di tempo e risorse computazionali. Le soluzioni LF sono molto rapide da predisporre e ottenere; nonostante la scarsa accuratezza, è comunque necessario che catturino perlomeno l'andamento generale degli output e la sensibilità di variazione rispetto ai parametri scelti per poter anche solo essere considerate un valido modello del problema fisico. Le soluzioni HF, invece, sono molto dispendiose e possono essere ottenute solo su un numero ridotto di configurazioni dei parametri, imposto dalla potenza computazionale a disposizione. L'utilizzo combinato dei due modelli è in grado di arrivare a soluzioni di ottima qualità a scelta nell'intero spazio dei parametri lanciando una quantità relativamente esigua di simulazioni HF.
Questo lavoro implementa un approccio *Bi-Fidelity* basato su un singolo modello LF e uno HF, per quanto una strategia *Multi-Fidelity* con più modelli sia possibile da utilizzare e sia stata discussa in diversi lavori di ricerca.
Viene innanzitutto fornita un'introduzione teorica alla strategia *Multi-Fidelity* dettagliando i vari passaggi interni dell'algoritmo implementato; quest'ultimo viene successivamente testato su alcuni problemi alle derivate parziali accademici con un parametro *random* monodimensionale. Dopo questi casi di validazione, l'approccio MF è infine applicato a delle simulazioni CFD con uno spazio di parametri aleatori funzionali multi-dimensionale dato dal numero di Mach e dall'angolo di attacco del flusso sul profilo. Il principale risultato da analizzare sta nel tasso di decadimento dell'errore riscontrato sulle soluzioni surrogate all'aumentare del numero di soluzioni HF calcolate, e di conseguenza del costo complessivo della ricostruzione.

Abstract

Within the project at issue, we consider a Multi-Fidelity strategy seeking a compromise between accuracy and efficiency in CFD simulations around a supercritical airfoil in transonic flow conditions with the inlet configuration parameters being specified by a Beta probability distribution, in order to stochastically model the small and sudden changes in the far-field conditions due for instance to a gust.
This type of numerical algorithm was recently proposed (2014) pursuing a synergic approach capable of exploiting the best features of a rather coarse Low-Fidelity model and an all around better High-Fidelity model, in order to obtain a surrogate high fidelity reliable solution at a reduced and way more accessible computational cost. The LF solutions are easily attainable and can be very quickly set up; despite this, they must at least be able to capture some general behaviour of the physical problem and its variation sensibility upon the chosen parameters in order to be considered a valid model in the first place. On the other hand, the HF solutions are highly time consuming and can only be performed on a few number of configurations with the constraint given by the available computational power at disposal. The combined use of the models should build high quality solutions in the entire parameter space only using a very modest amount of HF simulations.
The present thesis implements a Bi-Fidelity approach with a single LF and HF model, even though a truely multy-fidelity approach is possible and has been discussed in related scientific papers.
A theoretical introduction to the general Multi-Fidelity strategy is provided going through all its in-between steps, then the implemented algorythm is tested on some controlled academic PDE models with a single-dimension random parameter. After these validation cases, a MF strategy is finally applied to the CFD airfoil simulations with a tensorised multi-dimensional functional paramater space made up by the Mach number and the flow angle of attack on the airfoil. The main focus relies on analysing the error decay rate of the surrogate solution while increasing the number of performed HF simulations, and thus the overall cost of the reconstruction.

Riconoscimenti

Vorrei ringraziare sentitamente il mio relatore, la Professoressa Sandra Pieraccini, per la disponibilità e l'encomiabile professionalità mostrata nel supportarmi e guidarmi attraverso le difficoltà di questo cruciale momento della mia carriera universitaria.
La mia più sincera gratitudine va inoltre a Mariano, per il prezioso contributo concreto dato a questo lavoro, senza il quale la strada sarebbe stata dal principio più tortuosa.

Indice

Elenco delle figure

Introduzione

La simulazione di un fenomeno fisico complesso di interesse tramite l'utilizzo sinergico di più modelli computazionali gradualmente più accurati sta divenendo di uso sempre più comune in diversi campi della ricerca e dell'ingegneria. Una delle strategie in questione è detta *Multi-Fidelity* ed è stata introdotta in un lavoro del 2014 [8].

Alcuni casi per cui l'applicazione della strategia *Multi-Fidelity* può portare a benefici considerevoli nel rapporto tra risorse impiegate (e dunque efficienza della simulazione) e accuratezza dei risultati acquisiti sono ad esempio:

- Sistemi particolarmente complessi che ricevono in input una serie di parametri *random* modellati con delle corrispondenti funzioni di densità di probabilità: l'ambizione di condurre ciascuna simulazione legata alla fluttuazione di ogni parametro del problema con un modello computazionale costoso risulta spesso al livello pratico irrealizzabile. Si pensi ad esempio alle variazioni indotte sulle condizioni di volo di un'ala soggetta a raffiche di vento non modellabili in maniera deterministica.

- Studi di ottimizzazione per il *design* ingegneristico di un componente in cui la stessa simulazione va condotta un numero enorme di volte al variare ad esempio di alcuni parametri di *shaping* per analizzare la sensibilità della soluzione ottenuta alla variazione geometrica introdotta e raggiungere il *design* ottimale desiderato. Si pensi ad esempio alla modellazione della forma di un'ala per massimizzarne la performance aerodinamica: la risposta fisica del sistema sarà sensibilmente soggetta a delle variazioni e per rendere apprezzabile la reale influenza del parametro studiato può risultare ragionevole la scelta di un passo discreto di variazione molto fine da cui consegue necessariamente un'esplosione del numero di simulazioni necessarie per finalizzare lo studio.

Come visto negli esempi sopra menzionati, il numero di configurazioni "simili" da simulare per la caratterizzazione esaustiva di un determinato fenomeno fisico cresce sensibilmente all'aumentare dei parametri in gioco e

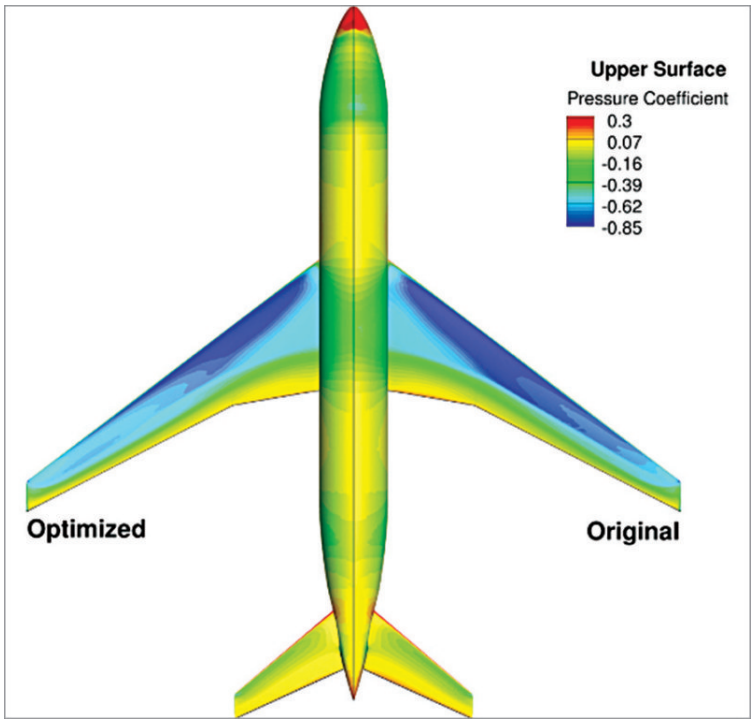


Figura 0.1: Esempio di risultati CFD sull'ottimizzazione di una geometria 3D

immediatamente può diventare fuori scala dal punto di vista pratico oltrepassando la capacità computazionale a disposizione, perlomeno volendo tenersi entro tempi di calcolo ragionevolmente accettabili. Inoltre i costi per la CFD possono essere, anche per una singola simulazione, molto onerosi in termini di risorse temporali ed economiche; basti pensare alla complessità computazionale di modelli di chiusura della turbolenza come RANS (*Reynolds Avereged Navier-Stokes*) o LES (*Large Eddy Simulation*), senza neanche citare la quasi sempre proibitiva simulazione diretta DNS (*Direct Numerical Simulation*). In questi casi risulta dunque evidente che allo stato dell'arte una strategia numerica che permetta di ottimizzare il numero di simulazioni onerose condotte costruendo delle soluzioni approssimate di buona affidabilità a partire da modelli molto più snelli e accessibili è non solo altamente auspicabile, ma spesso assolutamente necessaria.

Il presente lavoro di tesi ha come principale scopo l'applicazione dell'algoritmo *Multi-Fidelity* a una simulazione CFD 2D attorno a un profilo alare supercritico (nello specifico il RAE2822, ma i dettagli del caso di studio *benchmark* preso in esame verranno meglio precisati nel Capitolo 3 che tratterà quest'argomento) per analizzare il rapporto tra l'errore commesso nella ricostruzione delle soluzioni surrogate e il numero di *High-Fidelity* utilizzate

per la ricostruzione.

Le risorse informatiche necessarie alla conduzione dello studio sono state in parte sviluppate *ad hoc* ed hanno richiesto l'ausilio dei seguenti strumenti:

(a) Tutta la procedura di implementazione dell'algoritmo utilizzato è stata sviluppata in MATLAB tramite diversi *script* pubblicati insieme alla tesi.

(b) Le simulazioni CFD sono state condotte tramite il software *open-source* `su2`, sprovvisto di un proprio generatore di *mesh* interno.

(c) Le varie *mesh* utilizzate per la discretizzazione del dominio di calcolo sono state prodotte con un software esterno chiamato `gmsh`.

(d) Il programma che interfaccia gli *input* per le simulazioni generati in MATLAB con lo *scheduler* del *cluster* di calcolo utilizzato lanciando in maniera sequenziale tutte le simulazioni necessarie a caratterizzare lo spazio dei parametri è stato sviluppato in linguaggio `python`.

Il *report* realizzato si articola nel modo di seguito descritto:

Nel primo Capitolo si presenta un'introduzione teorica dettagliata della strategia *Multi-Fidelity* analizzandone le due principali fasi di selezione e ricostruzione. Si pone l'accento sulla riduzione effettiva della complessità computazionale di un problema rispetto alla sua dimensione e si fornisce una maniera di stima dell'errore commesso dall'algoritmo utilizzata poi nel prosieguo del lavoro.

Il secondo Capitolo descrive i *test case* di validazione dell'algoritmo *Multi-Fidelity* implementato per dei semplici modelli ODE e PDE. I due casi di studio saranno quello $[1D \times 1D]$ del filo elastico e quello $[2D \times 1D]$ della membrana elastica; la prima dimensione è quella del dominio di calcolo fisico , mentre la seconda si riferisce allo spazio dei parametri aleatori, in entrambi i casi ridotto a un singolo parametro che varia con legge di distribuzione uniforme.

Nel terzo Capitolo viene descritto il caso di studio vero e proprio dettagliando i vari elementi delle simulazioni effettuate (*mesh*, modello fisico, schemi numerici) alle cui soluzioni - in particolare al coefficiente di pressione C_p sul profilo - viene applicato l'algoritmo *Multi-Fidelity* con annessa analisi dell'errore commesso dalla ricostruzione.

Le Conclusioni tratteranno alcune riflessioni sul lavoro effettuato rispetto ai risultati attesi in partenza e dopo i *test* di validazione. Si darà spazio a possibili sviluppi futuri e margini potenziali di ampliamento del lavoro.

Capitolo 1

La strategia *Multi-Fidelity*

Si è già discusso nell'Introduzione come la complessità di alcuni sistemi nella propagazione dell'incertezza dagli *input* agli *output* renda proibitivo l'uso di simulazioni *High-Fidelity* deterministiche se non in modesta quantità. Per quasi ogni applicazione pratica è però possibile avere a disposizione un modello *Low-Fidelity* approssimato in grado di rendere quantomeno il comportamento globale del sistema; questo può risultare una semplificazione del modello più preciso per diversi motivi [14, 8]:

- Schematizza delle equazioni per delle variabili macroscopiche in un sistema multiscala, come ad esempio la turbolenza in un fluido (modelli LED);
- Schematizza le equazioni di governo di un fenomeno semplificate per trascurare una parte della fisica del problema (è questo il rapporto che intercorre ad esempio tra le equazioni di Eulero e quelle di Navier-Stokes) o ne considera una linearizzazione (come nel caso dello studio della propagazione delle onde acustiche);
- Schematizza semplicemente le stesse equazioni di governo ma su una griglia di calcolo spaziale molto più grossolana;
- Adotta uno schema temporale (*time-stepping*) poco accurato nel caso di discretizzazioni su modelli non-stazionari;
- Adotta uno schema di discretizzazione spaziale di ordine inferiore per le equazioni di governo o per alcune parti di esse: il sistema algebrico risultante da risolvere iterativamente avrà conseguentemente una matrice di coefficienti più sparsa;
- Una combinazione ibrida di due o più delle precedenti motivazioni.

Le simulazioni *Low-Fidelity* sono in grado di predire con buona affidabilità il comportamento del sistema e sono realizzabili con un dispendio relativamente molto di risorse; tuttavia, per risolvere totalmente la complessità degli *output*, è necessario disporre di alcune informazioni più raffinate che possono derivare esclusivamente da un modello *High-Fidelity*, per contro molto costoso.

Narayan, Gittelson e Xiu in [8] pongono le basi teoriche per un algoritmo che riesce ad estrapolare tramite la strategia *Multi-Fidelity* le informazioni più importanti dalle (poche) simulazioni HF effettuate e ad affiancarle alle soluzioni LF a disposizione per ottenere la sinergia tra efficienza e accuratezza ricercata.

L'algoritmo si compone essenzialmente di due *step* che verranno di seguito analizzati approfonditamente:

(1) Selezione, nello spazio discretizzato dei parametri su cui sono state calcolate le LF, di alcuni valori caratteristici detti *pivot* in corrispondenza dei quali verranno calcolate le HF. Per questo motivo il numero di punti selezionati è relativamente basso rispetto alla dimensione dello spazio parametrico a disposizione.

(2) Costruzione di una legge di approssimazione nello spazio delle soluzioni LF per stimare la soluzione in un nuovo punto desiderato; questa legge viene poi applicata alle soluzioni HF per la ricostruzione di una soluzione accurata.

1.1 Modello matematico

Per la descrizione della strategia si fa riferimento a un generico problema alle derivate parziali che presenta dell'incertezza su alcuni parametri [8, 3, 14]:

$$\begin{cases} u_t(x,t,Z) = \mathcal{L}(u) & \text{in } D \times (0,T] \times I_Z, \\ \mathcal{B}(u) = 0 & \text{su } \partial D \times [0,T] \times I_Z, \\ u = u_0 & \text{in } D \times \{t=0\} \times I_Z \end{cases} \tag{1.1}$$

Di seguito si introduce la notazione utilizzata nella (1.1) e necessaria anche per il prosieguo della trattazione:

D	Dominio fisico di calcolo; è un sottoinsieme di $\mathbb{R}^l$, con $l = \{1, 2, 3\}$ a seconda della dimensione del modello fisico utilizzato. Si introduce poi $\mathbf{x}$ generico punto di $\mathbb{R}^l$: nel caso più generale $\mathbf{x} = (x, y, z)$
t	Variabile temporale: $t \in [\,0, T\,]$, con $T > 0$ generico limite superiore del dominio temporale. Il modello è dunque definito come non-stazionario per rappresentare un problema fittizio che sia il più generale possibile
$\mathbf{I_Z}$	Spazio degli *input* del problema affetti da incertezza modellata con una certa distribuzione di probabilità. Può essere in generale multi-dimensionale ($I_Z \subseteq \mathbb{R}^d$, con $d \geq 1$ numero di parametri utilizzati). $\mathbf{Z}$ rappresenta ciascuna delle possibili istanze dell'insieme dei parametri in I_Z: $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_d)$ da cui dipende la soluzione u
$\mathcal{L}$	Generico operatore differenziale alle derivate parziali spaziali che agisce sulla variabile incognita del problema u
$\mathcal{B}$	Operatore di bordo che impone delle condizioni al contorno omogenee su tutto il confine del dominio di calcolo ∂D
$\mathbf{u_0}$	Stato iniziale della soluzione al tempo $t = 0$ da cui poi evolve il problema temporale

Per questo lavoro verrà da ora in poi inteso implicitamente che la (1.1) sia risolta numericamente così come in effetti implementato per i casi di studio dei capitoli 2 e 3: questo perché il focus principale della tesi è sulla riduzione della complessità computazionale di grandi simulazioni, ma in linea di principio nulla vieta che le valutazioni di $u(Z)$ siano effettuate sperimentalmente [8]. Il problema viene inoltre assunto matematicamente "ben posto" per tutti o quasi i valori di $\mathbf{Z} \in I_Z$.

Precisiamo infine che in questa trattazione teorica si assume che i parametri dello spazio I_Z siano degli *input* di tipo *random* distribuiti con una probabilità uniforme nel loro dominio di variazione. In realtà nessuna di questa due ipotesi è necessaria o ha un reale effetto di condizione sul funzionamento dell'algoritmo *Multi-Fidelity* [8]; di fatti:

(a) le variabili del problema potrebbero essere fatte variare con passo uniforme in un *range* scelto a priori deterministicamente ad esempio per un problema di ottimizzazione;

(b) la strategia si adatta molto facilmente a casi in cui la distribuzione

di probabilità delle variabili *random* sia non uniforme. Ad esempio vedremo che il campionamento dei parametri utilizzato nel capitolo 3 di questa tesi è di natura diversa.

1.1.1 Modello *Low-Fidelity*

Si definisce $u^L : I_Z \to V^L$ un'approssimazione *Low-Fidelity* della soluzione alla (1.1), che si assume semplice da ottenere, come precedentemente discusso; V^L è lo spazio approssimato della soluzione LF ed è denotato da un numero di gradi di libertà in genere relativamente piccolo, di seguito definito:

$$\dim(V^L) \stackrel{def}{=} N_L \tag{1.2}$$

u^L è la soluzione del sistema alle derivate parziali

$$\begin{cases} u_t^L(x,t,Z) = \mathcal{L}^L(u^L) & \text{in } D \times (0,T\,] \times I_Z, \\ \mathcal{B}^L(u^L) = 0 & \text{su } \partial D \times [\,0,T\,] \times I_Z, \\ u^L = u_0^L & \text{in } D \times \{t=0\} \times I_Z \end{cases} \tag{1.3}$$

in cui $\mathcal{L}^L$, $\mathcal{B}^L$ e u_0^L sono approssimazioni dei $\mathcal{L}$, $\mathcal{B}$ e u_0 dell'equazione (1.1) per uno specifico valore di $\mathbf{Z}$ e sono definiti in modo del tutto simile.

Si ricorda che il metodo di risoluzione numerico della (1.3) è irrilevante ai fini della discussione sull'algoritmo *Multi-Fidelity* [8] e può essere per comodità pensato come un metodo alle differenze finite, come per i casi del capitolo 2. In questo modo si può pensare che il valore N_L definito nella (1.2) rappresenti il numero di punti di griglia di una discretizzazione LF del dominio di calcolo D.

Si consideri ora un campionamento discreto Γ dello spazio I_Z:

$$\Gamma = \{z_1, z_2, \ldots, z_M\} \subset I_Z, \;\; M \gg 1 \tag{1.4}$$

in cui ciascuna componente z_k, con $1 \leq k \leq M$ è un vettore appartenente a $\mathbb{R}^d$ e rappresenta un'istanza completa dell'intero *set* di parametri incerti (che ha appunto dimensione $d = \dim(I_Z)$); il simbolo di vettore è in questi casi reso implicitamente per non appesantire troppo la notazione.

L'assunzione fatta sulla grandezza di M ha una duplice motivazione:

1. Dare una grande dimensione all'insieme Γ in modo da renderlo sufficientemente ricco perché possa rappresentare lo spazio parametrico con buona risoluzione;

2. Il campionamento di I_Z scelto rappresenta concretamente le configurazioni di parametri aleatori sulle quali risolvere il modello (1.3) e quindi M è il numero designato di simulazioni LF da effettuare. Si presuppone – come già discusso – che le soluzioni *Low-Fidelity* siano semplici da ottenere ma "povere" di informazione ed è quindi auspicabile averne molte a disposizione per potere estrapolarne la maggior quantità possibile di informazioni utili sulla fisica del problema.

Definiamo infine l'insieme delle soluzioni LF su Γ:

$$u^L(\Gamma) = \{u^L(z_1), u^L(z_2), \dots, u^L(z_M)\} \tag{1.5}$$

(in cui ancora una volta le notazioni matriciali, così come l'esplicita dipendenza delle soluzioni dalle coordinate $\mathbf{x}$ e da t, sono tralasciate) e il corrispondente spazio vettoriale da esse formato:

$$U^L(\Gamma) = \text{span}(u^L(\Gamma)) = \text{span}\{u^L(z_1), u^L(z_2), \dots, u^L(z_M)\} \tag{1.6}$$

1.1.2 Modello *High-Fidelity*

Si procede in maniera del tutto simile al caso *Low-Fidelity* descritto nel paragrafo precedente 1.1.1. Si può definire una $u^H : I_Z \to V^H$, soluzione di un corrispondente sistema alle derivate parziali

$$\begin{cases} u_t^H(x,t,Z) = \mathcal{L}^H(u^H) & \text{in } D \times (0,T\,] \times I_Z, \\ \mathcal{B}^H(u^H) = 0 & \text{su } \partial D \times [\,0,T\,] \times I_Z, \\ u^H = u_0^H & \text{in } D \times \{t=0\} \times I_Z \end{cases} \tag{1.7}$$

in cui V^H è lo spazio approssimato della soluzione accurata HF ed è denotato da un numero di gradi di libertà molto grande; ad esempio può essere considerata la rappresentazione della soluzione della (1.1) alle differenze finite su una griglia molto fitta:

$$\dim(V^H) \stackrel{def}{=} N_H, \;\; N_H \gg N_L \tag{1.8}$$

L'unico vincolo esplicitato sul rapporto tra le dimensioni dei due spazi V^H e V^L deriva direttamente dalla natura del modello HF e LF rispettivamente; al di là di questo, non si richiede alcuna particolare restrizione o interconnessione tra gli spazi delle soluzioni che possono anzi anche essere ottenuti con schemi numerici diversi [8] e che di certo generano una discretizzazione diversa e indipendente del dominio di calcolo D.

Analogamente a quanto fatto per le (1.5) e (1.6) si può quindi definire l'insieme e lo spazio delle soluzioni *High-Fidelity* su un particolare campionamento dello spazio parametrico I_Z. La precisa definizione e la dimensione del sottospazio γ introdotto di seguito verranno discusse nel paragrafo 1.2:

$$u^H(\gamma) = \{u^H(z_1), u^H(z_2), \dots, u^L(z_m)\} \tag{1.9}$$

$$U^H(\gamma) = \text{span}(u^H(\gamma)) = \text{span}\{u^H(z_1), u^H(z_2), \dots, u^H(z_m)\} \tag{1.10}$$

Ovviamente, con Γ descritto dalla (1.4), seguono in maniera analoga le definizioni di $u^H(\Gamma)$ e $U^H(\Gamma)$. Queste ultime, però, data la grande ricchezza in punti di Γ e la limitata capacità di valutazione delle u^H, non hanno quasi mai un reale interesse pratico se non in semplici casi di *test* in cui la realizzabilità delle simulazioni HF è sostanzialmente illimitata per quello che concerne gli scopi dell'analisi (vedi Capitolo 2).

1.2 Algoritmo *Bi-Fidelity*

Siamo ora nella situazione in cui si hanno a disposizione due modelli diversi u^L e u^H che approssimano la soluzione alla (1.1). Lo scopo di questa sezione è descrivere la costruzione di un algoritmo che possa combinare in maniera robusta i punti di forza dei modelli sopracitati per ottenere un'approssimazione efficiente e accurata della soluzione u^H sull'intero dominio I_Z. Si ricorda che l'insieme Γ che discretizza I_Z è assunto denso a sufficienza da catturare ogni variazione di u^H qualora si fosse in grado di fare una valutazione completa dell'insieme $u^H(\Gamma)$ [8].

La strategia *Multi-Fidelity* presentata con solo due modelli a disposizione è più propriamente detta *Bi-Fidelity*, ma i due termini sono utilizzati alternativamente nel corso dell'intera trattazione. Per una descrizione approfondita di una strategia che utilizzi più di due modelli e dei possibili vantaggi che se ne ricavano si veda [14, pp. 453–456].

L'algoritmo di riferimento per questo lavoro si articola come segue:

(1) Si effettuano tutte le simulazioni necessarie alla costruzione di $U^L(\Gamma)$, ottenendo quindi un ampio *database* di soluzioni *Low-Fidelity*.

(2) [Algoritmo 1] Si seleziona un sottoinsieme $\gamma \subset \Gamma$ di nodi di interpolazione detti *pivot*, in cui si dovranno valutare le soluzioni u^H. La scelta di questi punti va fatta opportunamente (dato il costo delle simulazioni HF) per poter trarre dalle poche u^H calcolate il maggior numero di informazioni possibili sul comportamento della vera soluzione alla (1.1).

(3) Si effettua una sola valutazione costosa di $U^H(\gamma)$ e si salvano le soluzioni in un *database* HF; è il passaggio computazionalmente più dispendioso dell'algoritmo *Multi-Fidelity*.

(4) [ALGORITMO 2] Si ricostruisce la soluzione per valori di $z \in I_Z$ al di fuori di γ interpolando le soluzioni dell'insieme $u^H(\gamma)$ pre-calcolato. La legge di approssimazione per la ricostruzione è calcolata utilizzando le u^L, quindi per la soluzione ricostruita in un determinato punto z_k è richiesta solo una valutazione di $u^L(z_k)$, poco costosa da ottenere.

1.2.1 Selezione dei nodi *pivot*

La scelta dei nodi in cui calcolare le soluzioni *High-Fidelity* è descritta in [8] con una procedura che implementa un cosiddetto *greedy algorithm*, cioè un algoritmo che cerca di accrescere ad ogni *step* la cardinalità di un insieme aggiungendovi un nodo da un insieme candidato di partenza (nel nostro caso Γ) scelto tramite una precisa regola caratterizzante per l'algoritmo. In pratica bisogna definire il sottospazio γ introdotto nelle (1.9) e (1.10), che sarà formato da una raccolta di punti "speciali" di Γ e avrà dimensione

$$\dim(\gamma) \overset{def}{=} m, \quad m \ll M \tag{1.11}$$

in cui ovviamente il rapporto richiesto tra le dimensioni m ed M deriva direttamente dal loro utilizzo all'interno della strategia. Il valore di m è frutto di una scelta arbitraria che deriva dalla capacità computazionale a disposizione. In genere, nei casi reali, è verosimile che non si abbia la possibilità di andare oltre ad un ordine di grandezza di poche decine o addirittura unità.

Si consideri una funzione distanza $d^L(\cdot,\cdot)$ su V^L; con essa si può formulare la distanza tra una funzione $v \in V^L$ e un sottospazio $W \subset V^L$ come segue

$$d^L(v, W) = \inf_{w \in W} \|v - w\|^L \tag{1.12}$$

La strategia per la selezione del nodo successivo da aggiungere al set corrente consiste nel *massimizzare la distanza tra la soluzione nel nodo candidato e lo spazio formato dalle precedenti soluzioni nel* set *di punti corrente*: in questo modo ad ogni iterazione si ottimizza la nuova quantità di informazioni ottenute dalla valutazione di $u^L(\gamma)$. In pratica, definendo

$$\begin{cases} \gamma^0 = \emptyset \\ \gamma^{n-1} = \{z_1, z_2, \ldots, z_{n-1}\}, \quad n > 1 \end{cases} \tag{1.13}$$

Tabella 1.1: Descrizione della notazione matriciale adottata per l'implementazione del *Selection Step* dell'algoritmo *Multi-Fidelity*

Nome	Dimensione	Descrizione
$\mathbf{V}_{\text{LF}}$	$\in \mathbb{R}^{N_L \times M}$	Matrice delle soluzioni *Low-Fidelity* $u^L(\Gamma)$; sull'i-esima colonna contiene tutti i gradi di libertà definiti per lo spazio approssimato V^L tramite la soluzione $u^L(z_i)$, con $1 \leq i \leq M$
$\mathbf{G}$	$\in \mathbb{R}^{M \times M}$	Matrice di Gram dei prodotti scalari nello spazio V^L: ogni sua componente è definita dalla relazione $(w_{ij})_{i,j \leq M} = \langle u^L(z_i), u^L(z_j) \rangle^L$ sulla cui effettiva implementazione (algebrica prima e numerica poi) si fornisce una spiegazione approfondita nel sottoparagrafo ad essa dedicato
$\mathbf{Z}$	$\in \mathbb{R}^{M \times d}$	Matrice che rappresenta la discretizzazione dello spazio dei parametri I_Z: su ogni riga presenta un insieme di parametri $z_i \in \Gamma$, con $i \leq M$
$\mathbf{P}$	$\in \mathbb{R}^{M \times M}$	Matrice di permutazione da applicare a Z in modo tale che lo spazio γ dei nodi selezionati sia rappresentato dalle prime m righe della matrice permutata $\mathbf{PZ}$

ad ogni iterazione n successiva si arricchisce l'insieme γ di un ulteriore nodo *pivot* tramite la formula ricorsiva:

$$\begin{cases} z^n = \arg\max_{z \in \Gamma} \left(d^L(u^L(z), U^L(\gamma^{n-1})) \right) \\ \gamma^n = \gamma^{n-1} \bigcup \{z^n\} \end{cases} \tag{1.14}$$

Le implementazioni possibili della (1.14) sono molteplici e necessitano di un'applicazione relativamente semplice di operazioni standard di algebra lineare numerica [3, 8]. Tutti gli approcci si basano sulla costruzione di alcune matrici e hanno come *output* principale una permutazione delle soluzioni $u^L(\Gamma)$. La notazione necessaria è introdotta nella tabella 1.1.

La matrice di permutazione $\mathbf{P}$ da cui si possono ricavare i nodi *pivot* selezionati è ottenibile in modo equivalente (perlomeno dal punto di vista puramente algebrico) da uno dei seguenti 3 metodi:

(a) *Si calcola una fattorizzazione di Cholesky della matrice* $\mathbf{G}$*, i.e.* $\mathbf{G} = \mathbf{H}\mathbf{H}^T$*, e poi si effettua una fattorizzazione* $\mathbf{QR}$ *con pivoting su colonne della matrice* ${\mathbf{V}_{\mathrm{LF}}}^T\mathbf{H}$*, del tipo:*

$$ {\mathbf{V}_{\mathrm{LF}}}^T\mathbf{H} = \mathbf{Q}\mathbf{R}\mathbf{P}^T $$

(b) *Si effettua una fattorizzazione LU completa della matrice* ${\mathbf{V}_{\mathrm{LF}}}^T\mathbf{G}\mathbf{V}_{\mathrm{LF}}$*,* i.e.

$$ {\mathbf{V}_{\mathrm{LF}}}^T\mathbf{G}\mathbf{V}_{\mathrm{LF}} = \mathbf{P}^T\mathbf{L}\mathbf{U}\mathbf{P} $$

(c) *Si effettua una fattorizzazione di Cholesky della matrice* ${\mathbf{V}_{\mathrm{LF}}}^T\mathbf{G}\mathbf{V}_{\mathrm{LF}}$*,* i.e.

$$ {\mathbf{V}_{\mathrm{LF}}}^T\mathbf{G}\mathbf{V}_{\mathrm{LF}} = \mathbf{P}^T\mathbf{L}\mathbf{L}^T\mathbf{P} $$

Si noti che in ciascuno dei metodi presentati ogni iterazione "tiene a mente" le precedenti andando ad aggiornare l'insieme γ pre-esistente. In pratica

$$ \gamma^{n-1} \subset \gamma^n \ \forall n $$

Allo stesso modo le matrici *vecG* ed $\mathbf{L}$ ad ogni iterazione successiva solo solo un "ingrandimento" delle precedenti; questo permette di risparmiare memoria di allocazione non ricalcolando ad ogni *step* le matrici per intero. Si garantisce infine in ogni caso, con l'utilizzo di queste operazioni algebriche lineari, la costruzione di un insieme di soluzioni $u^L(\gamma)$ che siano tra loro *linearmente indipendenti*.

L'algoritmo effettivamente implementato per tutti i casi di studio è quello della *full pivoted Cholesky Decomposition* (c). Una sua completa descrizione tramite l'utilizzo di uno *pseudo-codice* ispirato al linguaggio di MATLAB è riportata nell'Algoritmo 1. L'implementazione in MATLAB vera è propria utilizzata per l'analisi degli errori è realizzata nello *script* `Selection_Step_Cholesky.m`.

La scelta è effettuata sulla base di considerazioni di risparmio di memoria di allocazione delle variabili e di complessità computazionale [3, 8]. Infatti ad esempio il metodo (b) costringe ad utilizzare l'eliminazione gaussiana su matrici di grandi dimensioni facendo aumentare di molto il costo computazionale. L'algoritmo (a), invece, lavora sull'intera matrice $\mathbf{G}$ a differenza di (c) che permette di troncarne la valutazione all'n-esima riga quando si è giunti all'n-esimo *step* dell'Algoritmo 1.

Algoritmo 1: Selezione dei punti *pivot* dal *set* Γ tramite il metodo di decomposizione di Cholesky

Input: Matrice $\mathbf{V}_{\mathrm{LF}}$ delle soluzioni LF $[N_L \times M]$, Numero di nodi da selezionare m, Numero di soluzioni da considerare N (facoltativo, di *default* $= M$), Matrice dei prodotti scalari $\mathbf{A}$ (facoltativo, di *default* $\mathbf{A} = \mathbf{I}[N_L \times N_L]$).

Output: Matrice di Gram troncata alla riga m $\mathbf{G_L}$ e il suo fattore di Cholesky L; vettore della permutazione $\mathbf{P}$ contenente gli indici dei *pivot* selezionati.

begin

/* inizializzo il vettore dei prodotti scalari */

$\mathbf{w}(n) = (\mathbf{u}_n^L)^T \mathbf{u}_n^L \quad \forall\, n \leq M$

/* inizializzo il vettore P e il fattore di Cholesky inferiore L */

$P = 1 : M, \qquad L = \mathrm{zeros}(M, m)$

for $n = 1 : m$ **do**

/* trovo il successivo pivot e il suo indice */

$[e, p] = \underset{n \leq i \leq M}{\arg\max} \left(\mathbf{w}(i)\right)$

/* blocco che controlla la condizione di ill-conditioning */

if $e < \varepsilon$ **then**

$n = n - 1$ // mi arresto al passo precedente

break

/* aggiorno il vettore w */

$\mathbf{w}([n, p]) = \mathbf{w}([p, n])$

/* permuto le colonne di V e P */

$\mathbf{V}_{\mathrm{LF}}(:, [n, p]) = \mathbf{V}_{\mathrm{LF}}(:, [p, n]), \qquad \mathbf{P}([n, p]) = \mathbf{P}([p, n])$

/* blocco che aggiorna l'n-esima colonna di L */

$\forall\, t |\ n + 1 \leq t \leq m \qquad$ compute

$\mathbf{r}(t) = \mathbf{V}_{\mathrm{LF}}{}^T \mathbf{A} \mathbf{V}_{\mathrm{LF}} - \sum_{j=1}^{m-1} \mathbf{L}(t, j)\mathbf{L}(n, j)$

$\mathbf{L}(n, n) = \sqrt{\mathbf{w}(n)}$ // elemento diagonale

$\forall\, t |\ n + 1 \leq t \leq m \qquad$ compute

$\mathbf{L}(t, n) = \frac{\mathbf{r}(t)}{\mathbf{L}(n,n)}$ // elementi sotto-diagonale

$\mathbf{w}(t) = \mathbf{w}(t) - \mathbf{L}^2(t, n)$

/* produzione degli output */

$\mathbf{L} = \mathbf{L}(1 : n, :)$

$\mathbf{P} = \mathbf{P}(1 : n)$ // troncati ad n in caso di uscita per ill-conditioning

$\mathbf{G_L} = \mathbf{L}\mathbf{L}^T$

Si precisa infine che la matrice $\mathbf{V}_{\text{LF}}$ è assunta di rango di massimo per evitare di incorrere in problemi numerici, mentre la matrice di Gram $\mathbf{G}$ potrebbe avere un rango minore del numero di nodi *pivot* m scelto da selezionare, portando l'algoritmo di selezione ad arrestarsi anticipatamente per sopraggiunta condizione di *ill-conditioning*; in ogni caso, un controllo per questa situazione è semplice da implementare ed è stato considerato nell'Algoritmo 1. Come si vedrà, questo accorgimento sarà particolarmente rilevante per i casi di studio del Capitolo 2.

Calcolo della matrice di Gram

Nell'implementazione pratica dell'algoritmo di selezione, un ruolo cruciale è giocato dalla matrice $\mathbf{G}$; bisogna dunque discutere di una strategia che permetta di calcolarla. Teoricamente, seguendo la definizione standard fornita in tabella 1.1 di matrice dei prodotti scalari delle soluzioni *Low-Fidelity*, e considerando una base dello spazio V^L

$$\mathbf{b_k},\ k = \{1, 2, \dots, N_L\} \mid \forall z_i \in \Gamma \quad u_L(z_i) = \sum_{k=1}^{N_L} \hat{v}_k^L(z_i) b_k$$

si possono calcolare direttamente le componenti di $\mathbf{G}$ con il prodotto matriciale

$$\mathbf{G} = {\mathbf{V}_{\text{LF}}}^T \mathbf{B} \mathbf{V}_{\text{LF}} \tag{1.15}$$

in cui $\mathbf{B} \in \mathbb{R}^{N_L \times N_L}$ è la matrice dei prodotti scalari della base $\mathbf{b_k}$ ed è dunque semplicemente ottenibile utilizzando ancora una volta la definizione di prodotto interno a V^L:

$$(b_{ij})_{i,j \leq N_L} = \langle\, b_i, b_j \,\rangle^L \tag{1.16}$$

Tuttavia, è abbastanza frequente che vi siano casi in cui le soluzioni $u^L(\gamma)$ siano conosciute per punti su una discretizzazione del dominio fisico $\mathbf{x}$ in cui $N_L = \dim(V^L)$ rappresenta il numero di punti di griglia utilizzati per risolvere numericamente l'equazione modello – è questo ad esempio il caso di una soluzione ottenuta con un metodo alle differenze finite ed è di fatto la situazione in cui ci si trova in tutti i casi di studio presentati in questa tesi –. Con delle soluzioni di questo tipo non è assolutamente banale esplicitare la forma di una base dello spazio V^L, cruciale nel calcolo di $\mathbf{G}$ tramite la (1.15).

Bisogna allora trovare delle strade alternative che consistono nell'approssimare il prodotto scalare continuo su V^L $\langle \cdot, \cdot \rangle^L$ con una sua versione discreta $[\cdot, \cdot]_\alpha^L$ [14]. Agendo in tal senso, si perviene a una nuova definizione della

matrice di Gram, più utile al livello pratico:

$$\begin{aligned} &\mathbf{G} \simeq \left[u^L(z_i), u^L(z_j)\right]_\alpha^L = {\mathbf{V}_{\mathrm{LF}}}^T \mathbf{A} \mathbf{V}_{\mathrm{LF}} \\ &\text{con} \quad \mathbf{A} = \mathrm{diag}(\alpha_1, \alpha_2, \dots, \alpha_{N_L}) \end{aligned} \tag{1.17}$$

in cui gli α_i, per $1 \leq i \leq N_L$, sono dei pesi scelti appropriatamente per discretizzare il prodotto scalare $\langle \cdot, \cdot \rangle^L$, vale a dire che una scelta adeguata di questi coefficienti rende l'approssimazione per la matrice $\mathbf{G}$ quasi esatta. A questo punto si può utilizzare il risultato ottenuto con la (1.17) per la scomposizione già discussa in precedenza.

Anche la scelta degli α_i è nella pratica tutt'altro che banale: in prima approssimazione ci si può accontentare di prenderli semplicemente tutti uguali ad 1 e di definire quindi $\mathbf{A}$ come la matrice identità $\mathbf{I}^{N_L \times N_L}$. Questa soluzione è presentata nella descrizione del metodo riportata nell'Algoritmo 1.

Per il codice effettivamente utilizzato in `Selection_Step_Cholesky.m` si sono contemplate due alternative possibili tramite l'utilizzo di un *input* facoltativo che legge una matrice $\mathbf{B}$:

(a) Quando l'*input* viene omesso si considera $\mathbf{B} = \mathbf{A} = \mathbf{I}^{N_L \times N_L}$; in sostanza il programma trascura completamente la matrice di mezzo delle (1.15) e (1.17) e calcola

$$\mathbf{G} = {\mathbf{V}_{\mathrm{LF}}}^T \mathbf{V}_{\mathrm{LF}}$$

(b) Il codice utilizza effettivamente la (1.15) con una matrice B che è la matrice di massa della discretizzazione effettuata. Quest'ultima è una matrice tridiagonale definita come segue:

$$\mathbf{B} = \mathrm{diag}(1/6;\, 2/3;\, 1/3) = \begin{pmatrix} 2/3 & 1/6 & 0 & \dots\dots & 0 \\ 1/6 & 2/3 & 1/6 & 0 \;\dots & 0 \\ 0 & \ddots & \ddots & \ddots \;\dots & 0 \end{pmatrix}$$

Sarà chiaro nei test effettuati nel Capitolo 2 come questo accorgimento non farà la differenza nell'accuratezza delle ricostruzioni ottenute.

1.2.2 Ricostruzione *Bi-Fidelity*

A questo punto della trattazione, dopo l'applicazione dell'Algoritmo 1 e il calcolo delle soluzioni *High-Fidelity* necessarie si hanno a disposizione:

1. Un insieme discreto di punti Γ grande abbastanza da rappresentare fedelmente lo spazio I_Z e le corrispondenti soluzioni *Low-Fidelity* $u^L(\Gamma)$ su di esso calcolate inizialmente.

2. Un sottoinsieme deliberatamente piccolo $\gamma \subset \Gamma$ di nodi selezionati con la *ratio* formulata nella (1.14) sui quali si sono calcolate anche le dispendiose simulazioni *High-Fidelity*.

3. Inoltre, se per il *selection step* si è utilizzata la procedura descritta dall'Algoritmo 1, abbiamo la garanzia che $u^L(\gamma)$ sia un *set* di vettori linearmente indipendenti – ne avremo bisogno perchè useremo le $u^L(\gamma)$ come base dello spazio $U^L(\gamma)$ – e abbiamo già a disposizione la matrice di fattorizzazione di Gram $\mathbf{L}$ come *output* aggiuntivo "gratuito" della selezione [1].

Il nostro scopo resta quello di sintetizzare tutte queste informazioni raccolte e salvate in memoria per costruire in un qualunque nodo $z \in I_Z$ desiderato una soluzione *High-Fidelity* surrogata senza dover fare ulteriori valutazioni di u^H; la soluzione ricostruita in z sarà di qui in avanti denominata $v^H(z)$. Naturalmente l'ambizione è quella di ottenere un'approssimazione il più accurata possibile che ricalchi la $u^H(z)$ (che a sua volta per come è costruita dovrebbe essere una rappresentazione fedele di $u(z)$ soluzione della (1.1) nel continuo).

Si precisa che il punto z in cui si cerca la ricostruzione *Multi-Fidelity* è a priori *totalmente indipendente dagli insiemi discreti precedentemente definiti* Γ *e* γ: infatti in linea di principio, se Γ è stato scelto denso a sufficienza (questo requisito è ribadito con insistenza perché da esso dipende sensibilmente la bontà delle ricostruzioni ottenute al di fuori di Γ), le informazioni combinate estrapolate dagli spazi $U^L(\Gamma)$ e $U^H(\gamma)$ sono sufficienti per cogliere il comportamento della soluzione in tutto l'insieme *continuo* I_Z. L'unico ulteriore requisito necessario è una valutazione *on the fly* di $u^L(z)$ che però dovrebbe essere, per ipotesi di modello *Low-Fidelity*, molto semplice da ottenere.

La strategia proposta da Narayan, Gittelson e Xiu si articola come segue:

(1) Sfrutta la conoscenza del comportamento della soluzione u^L sn V^L al variare di z_k per costruire una legge interpolante di approssimazione in V^L

(2) Applica senza restrizione la stessa legge appena ottenuta allo spazio V^H utilizzando le soluzioni $u^H(\gamma)$

Quest'idea può essere formulata matematicamente in modo abbastanza semplice e diretto: dopo aver calcolato la $u^L(z)$ nel punto desiderato (ma

[1] Nell'Algoritmo 2, alla riga 5 si implementa la risoluzione del sistema (1.21) con il comando standard MATLAB \. In realtà, avendo già a disposizione una fattorizzazione triangolare inferiore di Cholesky L, converrebbe programmare "a mano" una risoluzione con una *forward substitution* che calcoli $\mathbf{c} = \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{g}$ [si veda 8, Algorithm 2]

la procedura si estende molto naturalmente a un insieme di punti e relative soluzioni $u^L(z_i)$ come presentato nell'Algoritmo 2) si cerca un vettore $v^L \in U^L(\gamma)$ che risolva

$$\langle v^L, \varphi(z)\rangle^L = \langle u^L(z), \varphi(z)\rangle^L, \qquad \forall\, \varphi \in U^L(\gamma) \tag{1.18}$$

Si dimostra che la soluzione all'equazione (1.18), per un *set* $u^L(\gamma)$ linearmente indipendente e uno spazio V^L in cui è propriamente definito un prodotto scalare $\langle\cdot,\cdot\rangle^L$, esiste ed è unica [si veda 8, pp. 501–502]. Si noti inoltre che la (1.18) rappresenta una proiezione di $u^L(z)$ sullo spazio $U^L(\gamma)$, *i.e* è possibile scrivere il vettore v^L come un insieme di m componenti nella base formata da $u^L(\gamma)$. Queste coordinate rappresentano i coefficienti c_k nella seguente equazione, che definisce l'operatore di proiezione $\mathcal{P}_{U^L_\gamma}(\cdot)$ (in altre parole la legge interpolante cercata):

$$v^L(z) = \mathcal{P}_{U^L_\gamma}(u^L(z)) = \sum_{k=1}^{m} c_k\, u^L(z_k) \tag{1.19}$$

A questo punto, con l'operatore di interpolazione appena introdotto per $U^L(\gamma)$, si può "trapiantare" la stessa logica su $U^H(\gamma)$ giungendo finalmente a definire la legge di ricostruzione *Bi-Fidelity*:

$$v^H(z) = \hat{u}^H \overset{def}{=} \sum_{k=1}^{m} c_k\, u^H(z_k), \qquad \forall\, z \in I_Z \tag{1.20}$$

La procedura implementata, dettagliatamente descritta con uno *pseudo-codice* ispirato al linguaggio MATLAB nell'Algoritmo 2, è programmata tramite lo *script* `Reconstruction_Step.m`

Calcolo dei coefficienti $\mathbf{c_k}$

L'ultima cosa che resta da definire per poter arrivare a un'implementazione completa della (1.20) descritta nell'Algoritmo 2 è un modo che ci consenta di calcolare i coefficienti $\{c_k\}_{k=1}^m$.

Definendo il vettore $\mathbf{c} \in \mathbb{R}^{m\times 1}$, si vede che questo è soluzione del sistema lineare

$$\begin{cases} \mathbf{G^L c} = \mathbf{g} \\ \{g_k\}_{k=1}^m = \langle u^L(z), u^L(z_k)\rangle^L, \qquad 1 \le k \le m \end{cases} \tag{1.21}$$

in cui $\mathbf{G^L} = \mathbf{L}\mathbf{L}^T$ è la matrice di Gram troncata all'm-esima riga ottenuta in *output* dall'Algoritmo 1.

È bene sottolineare che il sistema (1.21) potrebbe incorrere in un forte malcondizionamento, in particolare se si è sperimentato *ill-conditioning* – e dunque basso rango di $\mathbf{G}$ – durante la fase di selezione nodale.

Algoritmo 2: *Recontruction step* della strategia *Multi-Fidelity*

Input: Matrice $\mathbf{V}_{\mathrm{LF}}(\boldsymbol{\gamma}) \in \mathbb{R}^{N_L \times m}$ permutata e troncata con le LF nei nodi *pivot* in *output* dal *selection step*; matrice $\mathbf{u}^L(z_i) \in \mathbb{R}^{N_L \times q}$ con sulle colonne i q vettori LF da ricostruire in HF; matrice di Gram $\mathbf{G^L} \in \mathbb{R}^{m \times m}$ troncata in *output* dal *selection step*; matrice $\mathbf{V}_{\mathrm{HF}}(\boldsymbol{\gamma}) \in \mathbb{R}^{N_H \times m}$ che ha sulle colonne le HF calcolate nei dodi *pivot*; Matrice dei prodotti scalari $\mathbf{A}$ (facoltativo, di *default* $\mathbf{A} = \mathbf{I} \in \mathbb{R}^{N_L \times N_L}$).

Output: Matrice $\hat{\mathbf{u}}^H(z_i) \in \mathbb{R}^{N_H \times q}$ che ha sulle colonne le soluzioni ricostruite in HF per ciascun vettore LF richiesto in *input*; matrice $\mathbf{C_k} \in \mathbb{R}^{m \times q}$ con sulle colonne i vettori $\mathbf{c_i}$ con i coefficienti di ciascuna ricostruzione(facoltativo).

1 **begin**

/* inizializzazione delle matrici di output */

2 $\hat{\mathbf{u}}^H(z_i) = \mathrm{zeros}\,(N_H, q), \quad \mathbf{C_k} = \mathrm{zeros}\,(m, q)$

/* calcolo i termine noti g di ciascun sistema lineare */

3 $\mathbf{g} = (\mathbf{u}^L(z_i)^T \mathbf{A} \mathbf{V}_{\mathrm{LF}}(\boldsymbol{\gamma}))^T$

/* per ciascun vettore da ricostuire */

4 **for** $i = 1 : q$ **do**

/* calcolo i coefficienti c */

5 $\mathbf{C_k}(:, i) = \mathbf{G^L} \backslash\, \mathbf{g}(:, i)$

6 $\mathbf{c} = \mathbf{C_k}(:, i)$

/* ricostruisco le soluzioni in HF */

7 $\hat{\mathbf{u}}^H(z_i)(:, i) = \mathbf{V}_{\mathrm{HF}}(\boldsymbol{\gamma})\mathbf{c}$

1.3 Analisi dell'algoritmo

Questo paragrafo è dedicato a un breve *focus* sul costo computazionale per l'ottenimento di una soluzione ricostruita con la strategia *Multi-Fidelity* appena descritta e all'introduzione di alcune formule per stimare l'errore commesso dall'algoritmo nell'approssimare le vere soluzioni *High-Fidelity*.

Gli errori di seguito introdotti torneranno molto utili per visualizzare la crescita asintotica della precisione di ricostruzione dell'algoritmo al crescere del numero di nodi HF calcolati. È bene precisare, però, che si tratta solo di indici tangibili e semplicemente manipolabili nella pratica e non di una definizione rigorosa dell'errore di approssimazione commesso dell'algoritmo *Bi-Fidelity*. Una tale trattazione teorica esula dagli scopi di questo lavoro e può essere trovata approfonditamente [8, pp. 503–508], in cui gli autori si soffermano su una dimostrazione rigorosa della convergenza dell'algoritmo di

selezione e proiezione descritto nella sezione 1.2 al modello HF u^H. [2]

Per completezza, si menzionano in questa sede senza dettagliarli ulteriormente i due passaggi principali della dimostrazione, che arriva fra l'altro a definire un limite superiore che rappresenta il massimo errore possibilmente commesso dall'algoritmo di ricostruzione:

(1) Si dimostra che anche se la selezione nodale viene guidata dal modello *Low-Fidelity* come descritto e implementato nella procedura standard, la proiezione *High-Fidelity* fatta sullo spazio $U^H(\gamma)$ è comunque accurata.

(2) Si deriva un risultato teorico che limita superiormente l'errore commesso dalla legge interpolante per l'approssimazione delle soluzioni *High-Fidelity* descritta dall'equazione (1.20).

Questi due importanti risultati, combinati tra loro, portano in maniera diretta alla deduzione della convergenza dell'algoritmo *Multi-Fidelity* al modello u^H.

1.3.1 Complessità computazionale

Per rendere più apprezzabile la riduzione dell'ordine del modello che la strategia *Multi-Fidelity* è in grado di ottenere, ci si sofferma sull'analisi dei costi computazionali dell'intera procedura per ottenere, ad esempio, una soluzione ricostruita.

Abbiamo appena visto come l'algoritmo garantisca accuratezza e convergenza verso il modello *High-Fidelity*: è dunque il costo per ottenere una reale valutazione di $u^H(z)$ (per un qualsiasi punto $z \in I_Z$) l'appropriato termine di paragone da tenere a mente per questa valutazione di impiego di risorse.

Ripercorrendo ciascuno dei passaggi della strategia *Multi-Fidelity* si possono valutare i costi dell'algoritmo completo – si ricorda che si prende a titolo esemplificativo per quest'analisi la ricostruzione di una singola soluzione $u^L(z) \to \hat{u}^H(z)$ –:

1. Calcoli preliminari:
   Eseguiti solo *una tantum* e poi allocati in memoria [3]. Comprendono a loro volta:

[2] Il fatto che poi u^H converga alla reale soluzione u della (1.1) è ovviamente auspicabile ma dipende invece dalle proprietà di consistenza e convergenza del modello u^H scelto e messo in atto; senza perdita di generalità, si assume nel seguito che tali proprietà siano garantite

[3] può essere problematico – talvolta una complessità computazionale leggermente maggiore (ottenuta ad esempio rieseguendo iterativamente operazioni vettoriali *on the fly* invece di una singola valutazione matriciale) è addirittura preferibile a fronte di un risparmio di memoria di allocazione quando si ha a che fare con problemi dalle enormi dimensioni

M simulazioni *Low-Fidelity* effettuate per costruire il database di soluzioni salvate in $\mathbf{V}_{\mathrm{LF}}$ e utilizzato per la selezione dei nodi interpolanti. Per costruzione del modello u^L, non dovrebbero costituire un problema al livello di costo computazionale. Il valore di M è in teoria una scelta arbitraria; in pratica, rappresentando la cardinalità di Γ, ha il vincolo di dover rappresentare esaustivamente lo spazio dei parametri I_Z e le variazioni $\frac{\partial u}{\partial z}$ per non degradare l'accuratezza delle successive proiezioni $\mathcal{P}_{U^L_\gamma}$ su $U^L(\gamma)$ di soluzioni che non hanno contribuito alla selezione nodale.

La fattorizzazione di Cholesky Rappresenta il cuore dell'Algoritmo 1. Con qualche accorgimento adottato nella procedura, *e.g.* il troncamento della matrice all'm-esima riga (o n-esima con $n < m$ nel caso in cui il rango di $\mathbf{G}$ fosse strettamente minore del numero di nodi *scelto*) si può giungere ad una complessità dell'algebra lineare da eseguire dell'ordine di $O(Mm^2)$

m simulazioni *High-Fidelity* Sono di certo la fonte di costo maggiore nell'intera procedura. Tutto il senso della strategia consiste nel tentare di fare meno valutazioni possibili di u^H, *i.e.* poter scegliere un m molto piccolo. A tal proposito è bene insistere ancora sulla bontà della scelta dell'insieme Γ, da cui dipende la selezione dei nodi interpolanti e di conseguenza la quantità di informazione sul comportamento della soluzione al variare dei parametri estrapolata da ogni singola soluzione $u^H(z_i)$ calcolata: in questo senso, se il modello u^L è davvero economico come dovrebbe, conviene far salire molto il valore di M (o ottimizzare il modo in cui i nodi di discretizzazione vengono scelti per formare Γ) per poter far scendere il più possibile il limite inferiore di m. Per quanto riguarda quello superiore, invece, dipende ovviamente dalla capacità di effettuare simulazioni HF a disposizione. Nonostante questo, si noterà nei casi di studio dei prossimi capitoli che spesso la ricostruzione smette di migliorare sensibilmente all'aumentare di m al di là di una certa soglia oltre la quale non conviene quindi spingersi vista l'onerosità delle simulazioni rapportata al guadagno esiguo in accuratezza.

2. Simulazione LF su richiesta:
Una valutazione di $u^L(z)$ nel punto $z \in I_Z$ che si desidera ricostruire. Abbiamo visto come a priori sia indipendente dal *set* Γ e sia dunque da calcolare all'occorrenza. Non contribuisce in maniera sostanziale al costo complessivo della procedura.

3. Ricostruzione *Multi-Fidelity*:
   L'interpolazione vera e propria che approssima la soluzione. Il suo cuore è l'Algoritmo 2. Consiste di 3 passaggi principali:

   Calcolo di g Valutazione del vettore dei termini noti nel sistema lineare (1.21): prevede il calcolo di m prodotti scalari nello spazio delle soluzioni *Low-Fidelity* (di dimensione N_L)

   $$\langle u^L(z), u^L(z_k)\rangle_{k\leq m}^L$$

   per un costo complessivo dell'ordine $O(mN_L)$

   Risoluzione del sistema lineare È lo *step* che permette di ottenere il vettore dei coefficienti $\{c_k\}_{k\leq m}$; necessita dell'inversione della matrice di $\mathbf{G}$ che però non presenta grandi dimensioni perché è troncata all'm-esima riga; inoltre il fattore triangolare inferiore di Cholesky $\mathbf{L}$ è già calcolato e salvato in *output* dall'Algoritmo 1 nel *selection step* snellendo di molto la procedura di risoluzione che è dell'ordine $O(m^2)$

   Calcolo della soluzione interpolata L'applicazione del vettore $\mathbf{c}$ allo spazio $U^H(\gamma)$ tramite la (1.20). Si tratta di eseguire dei prodotti scalari per un costo dell'ordine $O(mN_H)$

Appare subito chiaro il grande vantaggio ricavato dalla procedura appena analizzata in tutte le sue fasi: l'ottenimento di una soluzione $\hat{u}^H(z)$ per qualunque punto z desiderato ha richiesto solamente, al netto delle operazioni preliminari necessarie e già smaltite in una fase precedente:

(a) *una singola valutazione del modello Low-Fidelity*

(b) una serie di operazioni standard di algebra lineare dal costo complessivo

$$O(m^2 + m(N_L + N_H))$$

1.3.2 Stima dell'errore

Si consideri un nuovo *set* di punti di cardinalità N

$$\tilde{z_k} \subset I_Z, \quad 1 \leq k \leq N \tag{1.22}$$

che sia completamente indipendente da Γ *e, di conseguenza,* γ. Si noti che in questo caso la scelta dei nodi in $\tilde{z_k}$ è davvero arbitraria e non deve essere compiuta con nessuna particolare *ratio*, a patto che ovviamente facciano parte dello spazio parametrico di partenza I_Z. Va da sè che un campionamento

adeguato, denso ed omogeneo, sia indicato per ottenere una valutazione *globale* dell'errore in I_Z che sia il più significativa possibile.

Questi nodi ci serviranno infatti da campione per testare l'errore di ricostruzione commesso dall'algoritmo per una valutazione della soluzione $\hat{u}^H$ in un generico punto prescritto z_i; ovviamente se prendessimo come riferimento un punto $z \in \{\Gamma \setminus \{\gamma\}\}$ e provassimo a ricostruire in HF ua soluzione che fa parte dello spazio $U^L(\Gamma)$ utilizzato per guidare la selezione nodale, sarebbe lecito attendersi un'accuratezza elevata per via dell'ottimo funzionamento dell'operatore di proiezione $\mathcal{P}^L_\gamma$ in queste particolari condizioni. L'errore stimato sarebbe però "drogato" dalla scelta dei vettori su cui è calcolato facendo perdere all'approccio generalità e attendibilità.

In questo senso, ricostruire in modalità *High-Fidelity* tutti i vettori $u^L(z)$, con $z \in \{\Gamma \setminus \{\gamma\}\}$ per cui si è scelto di non fare una reale valutazione di u^H può essere utile a dare una prova di forza della strategia numerica ma non è sufficientemente generale come valutazione globale dell'errore di approssimazione commesso.

Questa valutazione di $\hat{u}^H(\Gamma)$ è comunque piuttosto economica da ottenere e potrebbe essere interessante da utilizzare per costruirne un modello *Medium-Fidelity* interposto tra i due a disposizione per una strategia *Tri-Fidelity* tra quelle proposte in [14] e [8].

Fatte queste precisazione sulla necessità di effettuare un nuovo campionamento di I_Z per stimare gli errori commessi, è chiaro che una valutazione di discrepanza globale o puntuale della soluzione ricostruita rispetto alla reale *High-Fidelity* presupponga la realizzabilità delle soluzioni HF che permettano di costruire l'intero insieme $u^H(\tilde{z_k})$. Questo è garantito per casi di studio dal costo sufficientemente ridotto da permettere di effettuare delle simulazioni considerate *High-Fidelity* in maniera piuttosto agevole, ma è totalmente fuori questione per simulazioni reali su larga scala.

Si vedrà come per i modelli accademici del Secondo Capitolo sia stato addirittura possibile valutare interamente senza problemi $u^H(\Gamma)$ e considerare un insieme $\tilde{z_k}$ dalle dimensioni ancora maggiori di Γ. Se il modello matematico PDE che rappresenta u è semplice abbastanza si può in qualche caso conoscere la soluzione esatta di (1.1) e confrontare ad essa le ricostruzioni o fare un'analisi di convergenza di u^H su u. Si noti che, ovviamente proprio per la logica con cui è costruita, *l'approssimazione bi-fidelity non può in nessun caso migliorare il modello u^H nella rappresentazione di u.*

Invece, per la CFD, seppure in un caso relativamente semplice e con una geometria 2D, non si è potuto spingersi oltre un $N \sim m$ e ci si è dovuti accontentare di una stima dell'errore medio su I_Z sicuramente non esaustiva a causa dei vincoli computazionali imposti. In famosi casi di studio *benchmark* come quello analizzato in questo lavoro, è sempre possibile riferirsi a dati

sperimentali e/o simulazioni numeriche reperibili nella letteratura scientifica sull'argomento per avere un'idea della bontà dei modelli implementati.

Le formule utilizzate per la stima degli errori commessi in tutti i casi analizzati sono le seguenti:

$$\mathcal{E}_{L^\infty} = \|\hat{u}^H - u^H\|_{L^\infty(D\times I_Z)} \tag{1.23a}$$

$$\mathcal{E}_{L^2} = \|\hat{u}^H - u^H\|_{L^2(D\times I_Z)} \tag{1.23b}$$

$$\begin{cases} \|f(\mathbf{x})\|_{L^2(\mathbb{D})} = \sqrt{\int_{\mathbb{D}} |f|^2 d\mathbf{x}} \\ \|f(\mathbf{x})\|_{L^\infty(\mathbb{D})} = \sup_{\mathbf{x}\in\mathbb{D}} |f(\mathbf{x})| \end{cases} \tag{1.24}$$

Le (1.23) seguono le definizioni standard per il calcolo della norma $L^\infty \to \|\cdot\|_{L^\infty(D)}$ ed $L^2 \to \|\cdot\|_{L^2(D)}$ su un dominio fisico D (1.24), e vengono poi estese a I_Z – da cui l'intero dominio di calcolo fisico-parametrico $D \times I_Z$ – visto che a priori possono essere introdotte e sono ben definite per *qualunque* $z \in I_Z$.

Visto che una rappresentazione continua di ogni possibile variazione $\frac{\partial u}{\partial z}$ in I_Z non è di alcuna applicazione pratica, bisogna procurarsi una discretizzazione da implementare effettivamente nel codice che faccia ricorso al campionamento $\tilde{z_k}$ definito nella (1.22), costruito appositamente per stimare gli errori. Per cui, con buona approssimazione (a seconda delle caratteristiche di $\{\tilde{z_k}\}$), si possono scrivere:

$$\mathcal{E}_{L^\infty} \approx \max_{k\leq N} \|\hat{u}^H(z_k) - u^H(z_k)\|_{L^\infty(D)} \tag{1.25a}$$

$$\mathcal{E}_{L^2} \approx \frac{1}{N} \sum_{k=1}^{N} \|\hat{u}^H(z_k) - u^H(z_k)\|_{L^2(D)} \tag{1.25b}$$

che saranno le versioni di (1.23) utilizzate all'atto pratico nel codice MATLAB. Senza alcuna perdita di generalità, come vedremo, nei casi di studio affrontati e presentati successivamente il campionamento $\{z_k\}$ è stato costruito con degli *input* di tipo random con distribuzione di probabilità uniforme su tutto il *range* di variazione.

Vedremo come la definizione integrale di norma L^2 presentata nella (1.24) sarà declinata e interpretata numericamente caso per caso a seconda delle caratteristiche del dominio fisico in esame. Per quanto riguarda la $\|\cdot\|_{L^\infty(D)}$, invece, nel caso di un vettore definito per punti su una griglia numerica di cui i punti x_i rappresentano le coordinate, la definizione mostrata in (1.24) si

interpreta semplicemente come

$$\|v(x_i)\|_{L^\infty(\mathbb{D})} = \max_{i \leq \dim(\mathbb{D})} |f(x_i)|$$

Si noti come la (1.25b) faccia riferimento a un errore medio globale commesso dalla ricostruzione sull'intero campionamento dello spazio parametrico I_Z e venga infatti calcolata come una media aritmetica; al contrario, la (1.25a) è una stima dell'errore massimo commesso sulla specifica soluzione maggiormente "problematica" che corrisponde a un determinato valore di z testato.

Capitolo 2

Test Case accademici

Abbiamo visto come l'algoritmo di ricostruzione *Bi-Fidelity* presentato ed analizzato nel Capitolo 1 sia stato implementato in MATLAB tramite le due *function* contenute negli *script* `Selection_Step_Cholesky.m` e `Reconstruction_Step.m`. Questo capitolo sarà dedicato a mettere alla prova la strategia realizzata su due semplici problemi *benchmark* con un duplice intento:

(a) Analizzare l'andamento degli errori di ricostruzione così come introdotti nel paragrafo 1.3 per dei problemi numerici computazionalmente molto leggeri in modo da poter fare numerose valutazioni di un modello *High-Fidelity* a un costo ridotto ed accessibile: questo ci consentirà di campionare molto densamente lo spazio parametrico e ottenere una stima degli errori commessi molto più attendibile.

(b) Fare una validazione del codice in un caso di studio "controllato" prima di applicarlo alle simulazioni CFD. Le soluzioni dei modelli presentati sono intuitive, note, ampiamente discusse in letteratura e facilmente visualizzabili; questa semplificazione permette la validazione tramite il confronto dei risultati ottenuti sul decadimento dell'errore con altri casi di studio *benchmark* presenti in letteratura a cui è stata applicata la strategia *Multi-Fidelity*, oltre che la definizione di un riferimento proprio che assesti la qualità dell'algoritmo messo in atto al netto di "condizionamenti esterni" che potrebbero degradarne il funzionamento (*e.g.* qualità della *mesh*, convergenza del metodo numerico utilizzato, forti discontinuità delle soluzione, etc.)

I due modelli fisici considerati saranno

1. Caso 1D-1D: Il filo elastico; dominio fisico unidimensionale (x) e spazio parametrico rappresentato dal coefficiente di elasticità μ, *input* considerato affetto da incertezza.

2. Caso 2D-1D: La membrana elastica; naturale generalizzazione 2D del modello del filo. Dominio bidimensionale (x, y) e spazio parametrico rappresentato dal coefficiente di elasticità μ, *input* considerato affetto da incertezza.

2.1 Il Filo elastico

2.1.1 Modello matematico

Sia dato il filo elastico *sottile* rappresentato in figura 2.1a di sezione circolare costante A e lunghezza L (*i.e.* $D/L \ll 1$, con D diametro di A). In assenza di forze esterne applicate, l'asse del filo (figura 2.1b) giace lungo la coordinata x di una terna di riferimento cartesiana (x, y, z) occupandone una porzione corrispondente all'intervallo $[0, L]$ e i suoi estremi – posti in $x = 0$ e $x = L$ – sono per il momento vincolati all'asse x (quest'ultima ipotesi si tramuterebbe in termini matematici nell'applicazione al nostro problema di condizioni al contorno di Dirichelet *omogenee*, ma vedremo che non sarà necessaria e potrà essere generalizzata nella soluzione proposta).

Considerando ora che il filo sia soggetto alle forze [1]

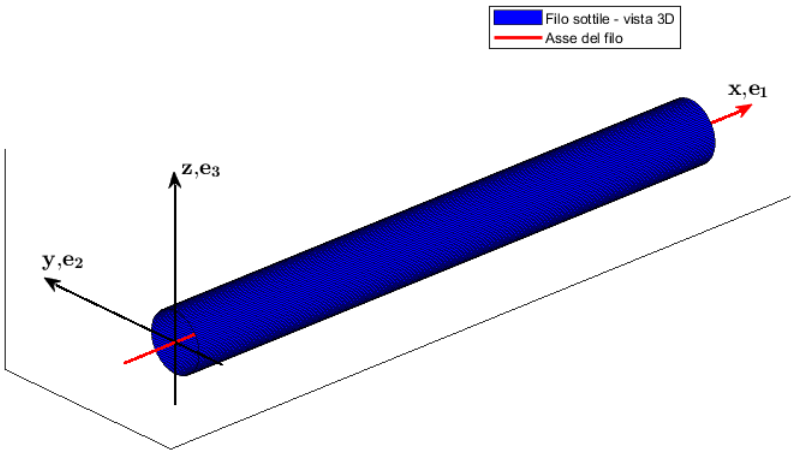
(a) $\mathbf{f} \rightarrow$ forza esterna che risiede nel piano $\{x, z\}$ ed è diretta verticalmente, cioè in maniera normale all'asse del filo. Le sue componenti rispetto ad una terna di versori $\{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}\}$ diretti come gli assi del sistema di riferimento sono, di conseguenza, $\mathbf{f} = (0, 0, f_3)$; D'ora in avanti, vista l'assenza d'ambiguità e per semplicità di notazione, si ometterà il pedice della componente f_3 scrivendo semplicemente f.

(b) $\mathbf{r} \rightarrow$ forza di richiamo elastico, definita quindi come proporzionale allo spostamento indotto $\mathbf{u}$ ma diretta in verso opposto;

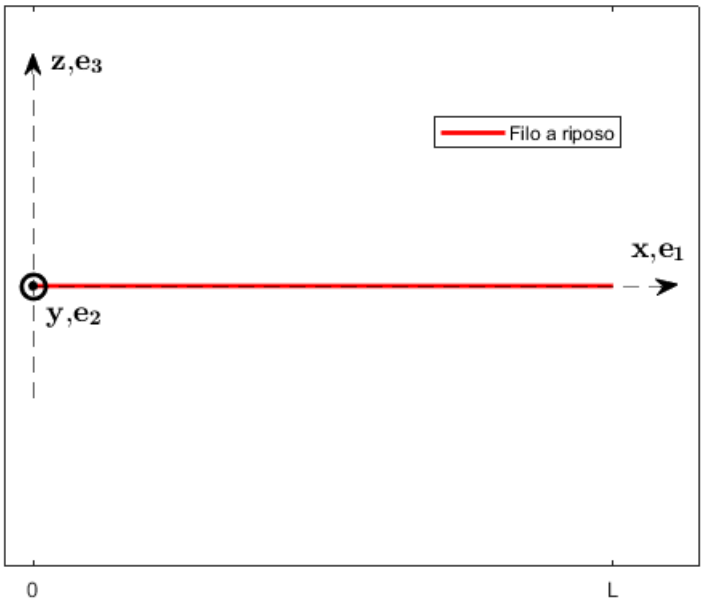
$$\mathbf{r} = -\gamma \mathbf{u}$$

con $\gamma \geq 0$ coefficiente elastico di richiamo.

[1] tutte le forze trattate in questo capitolo sono in realtà dimensionalmente delle densità volumiche di forza [N/m^3], e il termine *forza* – preso atto di questa doverosa precisazione – verrà comunque usato consapevolmente in maniera (leggermente) impropria per semplicità d'esposizione



(a) *Vista 3D del filo sottile*



(b) *Asse longitudinale del filo sottile*

Figura 2.1: Modello fisico del filo elastico – il filo è rappresentato a riposo e fissato agli estremi, con la relativa terna cartesiana considerata

e a condizione che siano rispettate le seguenti (ragionevoli) semplificazioni ed ipotesi

1. Il filo è composto di un materiale isotropo (*e.g* una lega metallica).
2. La forza esterna applicata $\mathbf{f}$ è considerata di "piccola" entità, *i.e.* il modello considerato si mantiene in campo lineare e valgono le relazioni costitutive lineari di Hooke tra sforzi e deformazioni del filo.
3. Lo spostamento $\mathbf{u} = (u_1, u_2, u_3)$ indotto dalle forze applicate è a sua volta considerato piccolo e si può assumere:
   - che sia complanare a $\mathbf{f}$, *i.e.* $u_2 = 0$
   - che la componente longitudinale u_1 sia ampiamente trascurabile rispetto a quella u_3 diretta come $\mathbf{f}$

   In formule si può cioè scrivere
   $$\mathbf{u} \simeq u_3 \, \mathbf{e_3}$$
4. Si sfrutta l'ipotesi di *filo sottile* per trascurare tutte le variazioni $\partial(\cdot)/\partial y$ e $\partial(\cdot)/\partial z$ e ridursi ad un modello *unidimensionale* in cui le uniche variazioni possibili avvengano lungo l'asse del filo x. In pratica si riduce il filo al suo asse e si individua il dominio fisico $D = \{x|\ x \in [0, L]\}$. Allo stesso modo lo spostamento u è definito come una funzione di una sola variabile lungo l'asse x: $\mathbf{u} = \mathbf{u}(x)$. $u(x)$ è assunta continua (il filo non può spezzarsi) e derivabile (la deformata indotta è "dolce"): si veda la figura 2.2.

L'equazione di equilibrio delle forze del sistema proiettata lungo la componente z (cioè moltiplicata scalarmente per $\mathbf{e_3}$) diventa un'equazione differenziale alle derivate ordinarie del primo ordine (con condizioni al contorno) che governa lo spostamento e si può scrivere come

$$\begin{cases} \dfrac{d\tau}{dx} + f - \gamma\, u = 0 & \text{con } x \in (0, L) \\ \tau = \mu \dfrac{du}{dx} & \text{con } x \in (0, L) \\ \{u(0), u(L)\} = \mathbf{0} \end{cases} \tag{2.1}$$

in cui la seconda equazione richiama la definizione di componente del tensore degli sforzi data dalla legge di Hooke (trascurando la componente longitudinale come discusso) e la terza definisce delle condizioni al contorno di Dirichelet

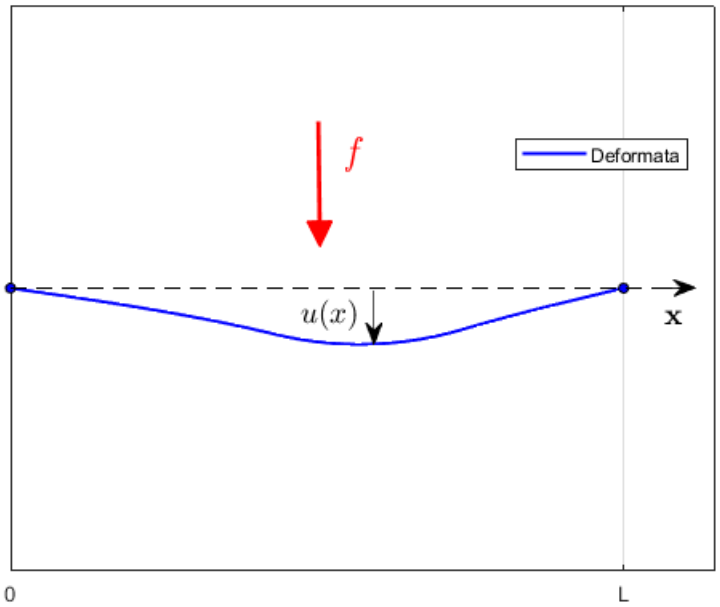


Figura 2.2: Deformata $u(x)$ del filo elastico soggetto al carico f

omogenee. μ rappresenta il modulo di taglio del materiale di cui è composto il filo e segue per i materiali isotropi una formulazione in funzione di modulo di Young E e coefficiente di Poisson $\nu \approx 0.3$

$$\mu = \frac{E}{2(1+\nu)}$$

Scrivendo la (2.1) in forma più compatta (*i.e.* sostituendo la seconda equazione all'interno della prima) e considerando una generalizzazione del modello che ammetta condizioni di Dirichelet non omogenee, si arriva al problema differenziale di riferimento per il *test* che stavamo cercando.

La (2.1) infatti diventa:

$$\begin{cases} -\dfrac{d}{dx}\left(\mu \dfrac{du}{dx}\right) + \gamma\, u = f \quad \text{con } x \in (0, L) \\ \{u(0), u(L)\} = \{g_0, g_L\} \end{cases} \tag{2.2}$$

2.1.2 Discretizzazione e implementazione nel codice

Il modello del filo elastico è stato risolto tramite la *function* implementata in MATLAB nello *script* `Filo_Dir_FinElem_Solver.m` che implementa il

metodo degli elementi finiti. Per discretizzare la (2.2) agli elementi finiti, però, dobbiamo considerarne una riformulazione integrale, chiamata *formulazione variazionale (o debole)* del problema:

$$\begin{cases} u \in V \\ \int_0^L \mu \frac{du}{dx}\frac{dv}{dx}\,dx + \int_0^L \gamma uv\,dx = \int_0^L fv\,dx \quad \forall\, v \in V \end{cases} \tag{2.3}$$

in cui V rappresenta l'insieme degli spostamenti ammissibili v che il filo elastico può compiere quando soggetto a dei carichi e che devono godere delle seguenti proprietà: continuità, derivabilità (eventualmente a tratti), rispetto delle condizioni di vincolo agli estremi.

La formulazione variazionale (2.3) è per i casi "regolari" equivalente a quella differenziale (2.2) (*i.e.* entrambe forniscono la medesima soluzione $u(x)$), ma permette anche di trattare i casi più generali in cui le funzioni $f(x)$ e $\mu(x)$ sono continue solo a tratti su $[0, L]$ oppure la forza applicata sia un carico concentrato in un punto.

Il *solver* agli elementi finiti utilizzato è costruito con le seguenti caratteristiche:

- Considera una forza f che carica il filo passata in *input* che può essere:
  (a) una funzione continua definita su tutto il dominio $D = [0, L]$; in tal caso è passata al *solver* tramite un *function handle*
  (b) una forza definita per punti sulla griglia considerata nella formulazione discreta del problema; questo permette di ammettere nella soluzione che f sia una funzione continua solo a tratti. In questo caso, l'*input* è passato tramite un semplice vettore.

- Può considerare eventualmente un coefficiente di taglio $\mu = \mu(x)$ variabile (in modo continuo) lungo l'asse del filo e che viene passato in x tramite un *function handle* che ne definisce l'andamento. Nell'utilizzo pratico del codice, però, visto che μ funge da parametro affetto da incertezza su cui costruire la strategia *Multi-Fidelity*, si ci è sempre limitati a un caso di coefficiente costante $\forall\, x \in D$ (e dunque sull'intera griglia numerica).

- Il coefficiente di richiamo elastico γ è supposto costante a priori.

- La discretizzazione in elementi finiti considera una griglia di punti non necessariamente equispaziata in cui l'i-esimo elemento ha lunghezza pari a $h_i = x_i - x_{i-1}$. Il *solver* legge la griglia tramite un semplice vettore di punti contenente le coordinate x_i.

- Considera delle condizioni al contorno di Dirichelet eventualmente *non omogenee*, cioè lo spostamento agli estremi del filo può essere fissato a un valore diverso da 0, come già assunto matematicamente nella (2.2)
- Restituisce la deformata statica $u(x)$ su tutti i punti di griglia forniti in *input* x_i.

La formulazione discreta della (2.3) conduce alla costruzione del sistema algebrico lineare

$$\mathbf{Au} = \mathbf{f} \tag{2.4}$$

di dimensioni $N-2$, se N è il numero di punti di griglia utilizzati per la soluzione; questo perchè gli spostamenti nel primo e ultimo nodo della griglia sono fissati dalle condizioni al contorno e non aggiungono equazioni al sistema.

Una derivazione completa delle equazioni discrete che conducono alla (2.4) presupporrebbe un approfondimento teorico sul metodo degli elementi finiti che esula dallo scopo di questa tesi e viene dunque omessa (si vedano come riferimenti [10, 7]). Questi modelli sono infatti usati strumentalmente come *test case* per l'algoritmo *Multi-Fidelity* analizzato e vengono brevemente presentati solo per completezza e linearità di esposizione.

In ogni caso, si può citare la forma della matrice $\mathbf{A}$ infine ottenuta per comprenderne la relativa semplicità. Essa è formata da due componenti, relative agli effetti di taglio e di richiamo

$$\mathbf{A} = \mathbf{A}_\mu + \mathbf{A}_\gamma$$

in cui:

(a) Matrice di rigidezza $\mathbf{A}_\mu$
è la stessa che si otterrebbe senza considerare la forza elastica di richiamo (le due componenti del sistema sono cioè nel nostro semplice modello completamente *disaccoppiate*). Nel caso particolare di griglia equispaziata $h_i = \text{cost}$ e modulo di taglio $\mu = \text{cost}$ in $[0, L]$, $\mathbf{A}_\mu$ è una matrice tridiagonale del tipo

$$\mathbf{A}_\mu = \frac{\mu}{h}\,\text{diag}(-1; 2; -1)$$

(b) Matrice di massa $\mathbf{A}_\gamma$
si presenta anch'essa sotto forma tridiagonale, ed ha valori costanti su ciascuna diagonale nel caso particolare in cui la griglia sia equispaziata e il coefficiente di proporzionalità γ sia definito costante in $[0, L]$:

$$\mathbf{A}_\mu = \gamma h\,\text{diag}(1/6; 2/3; 1/6)$$

Si noti che, ovviamente, la somma di due matrici tridiagonali sarà anch'essa tridiagonale. Dal punto di vista numerico questo è un vantaggio considerevole: le matrici tridiagonali sono un particolare tipo di *matrice sparsa*, in cui cioè il numero delle componenti diverse da 0 è molto piccolo rispetto al totale degli elementi. Questo tipo di matrici può essere allocato in memoria con grande efficienza (conservando solo le entrate non nulle e la loro posizione indicizzata) e permette di poter raffinare enormemente la griglia continuando ad avere un problema di inversione del sistema computazionalmente gestibile in tempi brevissimi.

2.1.3 Applicazione della strategia *Multi-Fidelity*

Con riferimento alla notazione introdotta e ampiamente utilizzata nel Capitolo 1, decliniamo le varie componenti della strategia *Multi-Fidelity* per il caso in esame del filo elastico per poi analizzarle in sequenza:

(a) la soluzione del problema u deve risolvere la (2.3) o la (2.2). In particolare si tratta di un problema *tempo-indipendente* e con un dominio unidimensionale rappresentato da un intervallo di numeri reali.

(b) i modelli u^L e u^H sono le discretizzazioni dell'equazione di riferimento in modalità *Low-Fidelity* e *High-Fidelity*, rispettivamente. La strategia messa in atto declina le diverse fedeltà dei modelli mediante un forte raffinamento della griglia di calcolo su cui si risolvono le equazioni numeriche, a parità di modello matematico (2.3).

(c) lo spazio parametrico I_Z è rappresentato dall'unico parametro incerto considerato nell'equazione modello: il modulo di taglio μ.

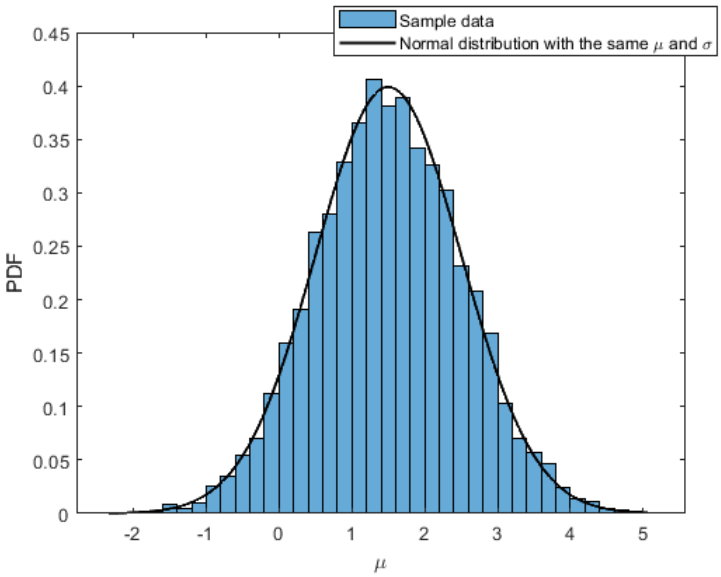
1 Campionamento dello spazio I_Z

Si descrivono di seguito le scelte fatte per la costruzione degli insiemi Γ, che guida la selezione nodale, e $\{\tilde{z}_k\}$, utile al calcolo degli errori presentato in un successivo paragrafo dedicato.

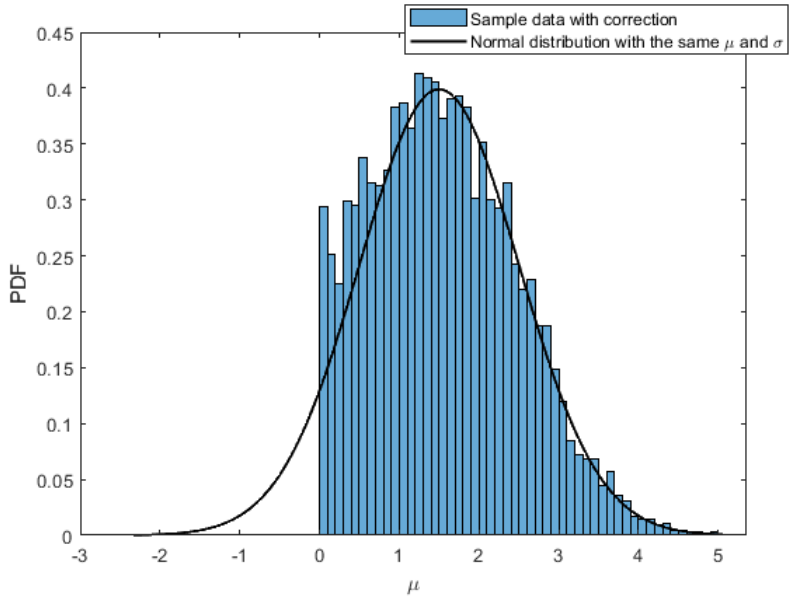
L'insieme Γ su cui si è valutato il modello *Low-Fidelity* è costruito con un campionamento *random* della variabile affetta da incertezza μ con una distribuzione di probabilità di tipo *gaussiano (o normale)*.

Si è proceduto a ritroso:

(1) in primo luogo è stato calcolato un vettore di valori di μ che segue una distribuzione normale tramite la funzione di MATLAB `randn`. La *media* e la *deviazione standard* della distribuzione valgono rispettivamente 1.5 ed 1.



(a) *Dati grezzi*



(b) *Correzione per escludere i valori negativi*

Figura 2.3: Distribuzione gaussiana del campionamento sul parametro μ – Confronto con la PDF analitica

(2) i valori campionati grezzi così ottenuti sono stati corretti in valore assoluto per evitare possibili istanze di coefficienti di taglio negativi (vista la vicinanza della media allo 0) ovviamente privi di alcun senso fisico.

(3) a questo punto si è potuto valutare lo spazio parametrico monodimensionale I_Z come l'intervallo numerico che ha per estremo inferiore il valore minimo del vettore μ corretto, e per estremo superiore il valore massimo.

Il valore di M considerato (numero di simulazioni LF calcolate per costruire lo spazio $U^L(\Gamma)$) è stato impostato pari a 1000.

La scelta della distribuzione normale invece di una più omogenea distribuzione *random* uniforme vuole dare una logica di fondo fisica al caso di studio piuttosto astratto: è lecito assumere che le acquisizioni del modulo di taglio di un materiale vengano ottenute tramite campagne sperimentali i cui dati sono di certo affetti da errori aleatori che conducono inevitabilmente a distribuzioni a campana del tipo prescritto dalla (2.5). Del resto si è già asserito nel corso della trattazione teorica nel Capitolo 1 che in linea di principio, con i dovuti accorgimenti, il funzionamento dell'algoritmo *Multi-Fidelity* non dipende dalla metodologia di costruzione degli insiemi discreti su I_Z; in [14, 8] si considerano punti selezionati *alla Monte Carlo* e/o distribuzioni uniformi, ma non si preclude a priori alcun tipo di scelta purché consona.

La figura 2.3 mostra il confronto tra gli istogrammi delle distribuzioni normali sui dati campionati e la corrispondente ben nota funzione analitica gaussiana per i valori $\mu = 1.5,\ \sigma = 1$ [2], che viene di seguito ricordata:

$$\mathcal{G}(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.5}$$

Per quanto riguarda il *set* indipendente degli $\{\tilde{z}_k\}$, la scelta è ricaduta su una distribuzione con densità di probabilità uniforme (funzione `rand`) su tutto l'intervallo I_Z definito a seguito della creazione di Γ (figura 2.4). La *ratio* in questo caso è data dalla ricerca di una rappresentazione abbastanza omogenea dello spazio parametrico che possa portare a una stima significativa dell'errore medio. Data la semplicità del caso trattato, come anticipato in 1.3, possiamo permetterci di considerare una cardinalità $N = \dim(\{\tilde{z}_k\})$ di 10^4 su cui valutare le u^H, di un ordine di grandezza superiore a M.

[2] *n.b.*: in questo specifico contesto il simbolo μ si riferisce al valor medio della curva gaussiana e non al modulo di taglio del filo finora trattato. Quest'ambiguità di notazione – che riguarda solo questo breve passaggio – è stata accettata per non ridefinire il simbolo di media storicamente utilizzato in letteratura per la (2.5)

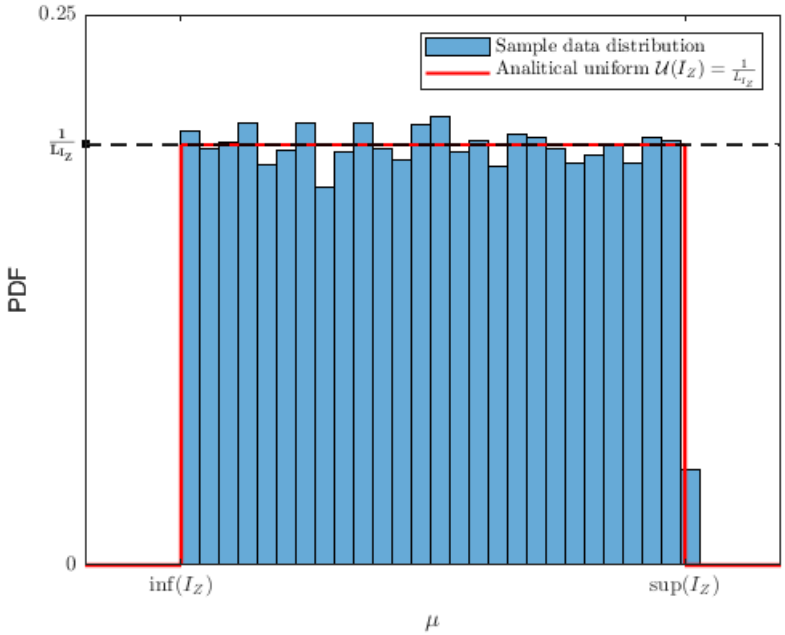


Figura 2.4: Distribuzione di probabilità dei dati campionati $\{z_k\}$ – confronto con la *step function* analitica

La PDF analitica della distribuzione uniforme su un intervallo $\mathbb{I}$ si indica con $\mathcal{U}(\mathbb{I})$ ed è semplicemente una *step function* il cui valore di gradino si può calcolare con la relazione:

$$\mathcal{U}(\mathbb{I}) = \frac{1}{b_{\mathbb{I}} - a_{\mathbb{I}}}, \quad \text{con } a_{\mathbb{I}} = \inf(\mathbb{I}) \text{ e } b_{\mathbb{I}} = \sup(\mathbb{I}) \tag{2.6}$$

Nel nostro caso, definendo L_{I_Z} l'ampiezza dell'insieme I_Z ricavato:

$$\mathcal{U}(I_Z) = \frac{1}{L_{I_Z}} \approx 0.2$$

Il confronto tra le distribuzioni di probabilità discreta e analitica uniforme è mostrato in figura 2.4.

La figura 2.5 mostra invece una rappresentazione parziale (per ragioni di qualità di visualizzazione) degli insiemi discretizzati creati per dare un'intuizione della distribuzione dei campioni sullo spazio parametrico. Si noti che, come richiesto, i punti in rosso che rappresentano $\{\tilde{z}_k\}$ sono distribuiti abbastanza omogeneamente sull'intervallo, mentre quelli in blu (*set* Γ) si addensano sensibilmente attorno al valor medio precisato per μ pari ad 1.5.

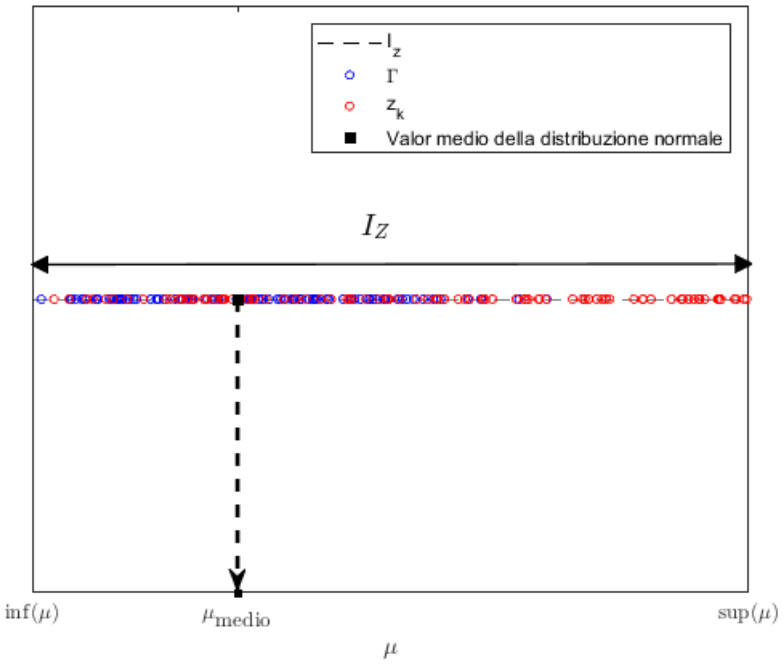


Figura 2.5: Collocazione dei campionamenti effettuati Γ e $\{\tilde{z_k}\}$ sull'intervallo I_Z

2 Modello *Low-Fidelity* e selezione nodale

Abbiamo precisato come il modello matematico di riferimento e lo schema numerico utilizzato nella nostra analisi restino gli stessi già discussi in precedenza per la costruzione di entrambi i modelli LF e HF. A questo punto dunque, al fine di caratterizzare le soluzioni u^H, non ci resta che specificare la griglia di calcolo su cui sono state valutate.

Fermo restando il valore di $M = 1000$ e un dominio di calcolo di ampiezza unitaria ($L = 1$), si è fatto uso di 3 griglie *Low-Fidelity* diverse, dal crescente numero di nodi (e dunque dalla crescente accuratezza):

Modello A: Il più grossolano a disposizione; utilizza una griglia equispaziata fatta di 22 nodi totali, quindi 20 nodi interni, *i.e.* $N_L = 20$. La quantità di punti dovrebbe essere insufficiente a risolvere il dominio D anche per un modello appositamente LF (a titolo di esempio si noti la scarsità di punti agli estremi del domini per le soluzioni A mostrate in figura 2.6)

Modello B: di finezza intermedia; utilizza una griglia equispaziata con 52 nodi totali, quindi impone un valore $N_L = 50$

Modello C: il più fine a disposizione; la griglia resta equispaziata ma il numero di nodi è un ordine di grandezza superiore rispetto al Modello A, *i.e.* un totale di 102 punti di griglia, con $N_L = 100$

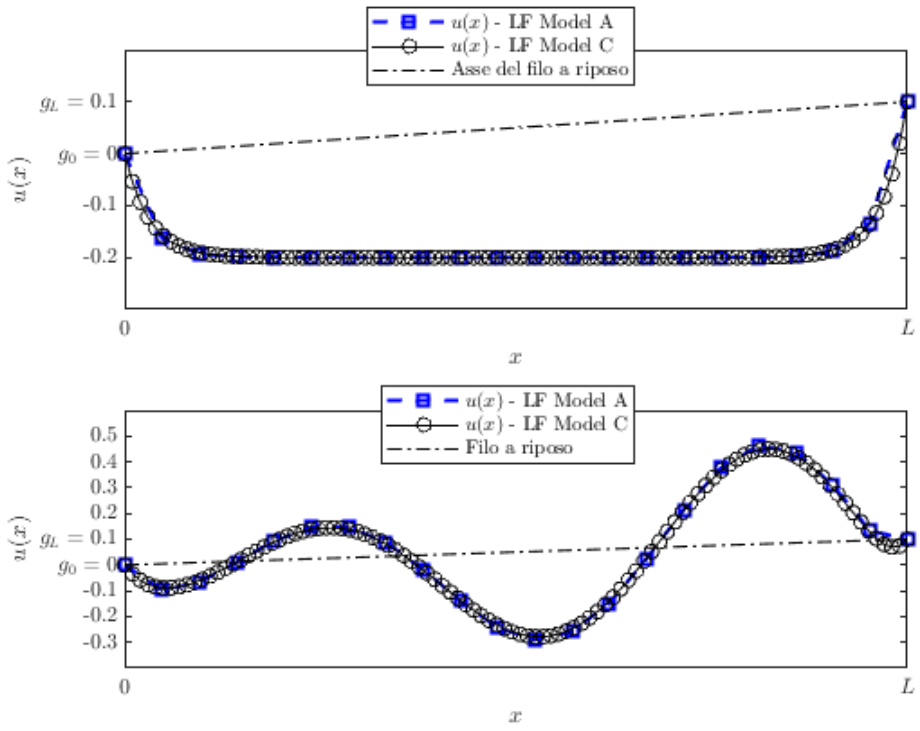


Figura 2.6: Soluzioni LF ottenute con i modelli A e C, per le stesse condizioni. *In alto*: forza f uniforme. *In basso*: forza f continua combinazione di funzioni polinomiali e trigonometriche

La griglia resta equispaziata in tutti i casi perchè è infatti stata testata per distribuzioni di forza $f(x)$ molto eterogenee tra loro, non dando modo di intuire a priori la collocazione dei maggiori gradienti della soluzione in cui eventualmente avrebbe avuto senso applicare uno *stretching*.

Si noti anche che l'implementazione di 3 modelli LF diversi ha il solo scopo di testare le variazioni di qualità dell'algoritmo di ricostruzione al crescere del numero di gradi di libertà dello spazio V^L; i modelli sono stati utilizzati uno per volta e *l'uno indipendentemente dall'altro* per la costruzione di 3 leggi interpolanti diverse che sono state poi confrontate solo in una fase conclusiva. Questa precisazione è importante al fine di sgombrare il campo da possibili ambiguità con una strategia *Multi-Fidelity* che utilizza dei modelli *Medium-Fidelity* intermedi tra quello LF e quello HF [14, 8]. L'algoritmo implementato, come chiarito nella sezione 1.2 a pagina 6, è quello *Bi-Fidelity*.

Per ciascuno dei modelli LF considerati si è costruito l'insieme delle M soluzioni $u^L(\Gamma)$ salvate sulle colonne della matrice $\mathbf{V}^{\mathrm{LF}}_{mod} \in \mathbb{R}^{N^L_{mod} \times M}$ in cui $mod = \{A, B, C\}$. Le soluzioni usate per la ricostruzioni sono state ottenute per un caso in cui la forza f è uniforme, *i.e,* è una funzione continua e costante sull'intervallo $[0, L]$.

La matrice così formata è stata quindi passata in *input* alla *function* `Selection_Step_Cholesky.m` (Algoritmo 1) insieme a un valore prescritto "piccolo" di m. In realtà, come accennato nell'introduzione a questo *test case*, per un problema numerico computazionalmente così leggero ci si può permettere di effettuare un numero sostanzialmente illimitato di valutazioni di u^H (e si sfrutterà la cosa per il calcolo degli errori); detto ciò, il valore di m è comunque limitato superiormente dalle seguenti considerazioni:

(a) la validità dell'algoritmo va testata significativamente per un valore di $m \ll M$, al fine di non "drogare" i risultati ottenuti con ricostruzioni che non sarebbero realizzabili nella pratica

(b) al crescere del valore di m si calcola una matrice di Gram $\mathbf{G}$ dalle dimensioni via via maggiori, fino all'eventuale superamento del suo rango e dunque al raggiungimento della condizione di uscita per *ill-conditioning* dell'Algoritmo 1

(c) non è in ogni caso garantita una decrescita monotòna dell'errore all'aumentare senza vincoli di m, *i.e.* non vale comunque la pena spingersi oltre una certa soglia.

Si è tentata la ricostruzione per un valore massimo di $m = 15$, ma vedremo graficamente nei risultati ottenuti che il verificarsi della suddetta condizione "patologica" ha costretto la selezione ad arrestarsi prima.

La figura 2.7 mostra la collocazione dei nodi interpolanti selezionati dall'algoritmo guidato dallo spazio U^L_{B} delle soluzioni *Low-Fidelity* di tipo B. Come si può notare l'algoritmo è stato in grado di selezionare solo 9 nodi sui 15 richiesti inizialmente prima di incorrere nell'uscita forzata per *ill-conditioning*; si tratta comunque di un valore di HF ragionevole per ottenere delle ricostruzione accurate, come si vedrà.

3 Modello *High-Fidelity*

Le soluzioni considerate HF sono costruite su una griglia numerica equispaziata molto fitta formata da 1002 nodi, e dunque da $N_H = 10^3$ punti interni. Si noti che tra la griglia HF e la migliore delle LF a disposizione (modello C) passa un ordine di grandezza di differenza; questo dovrebbe essere sufficiente

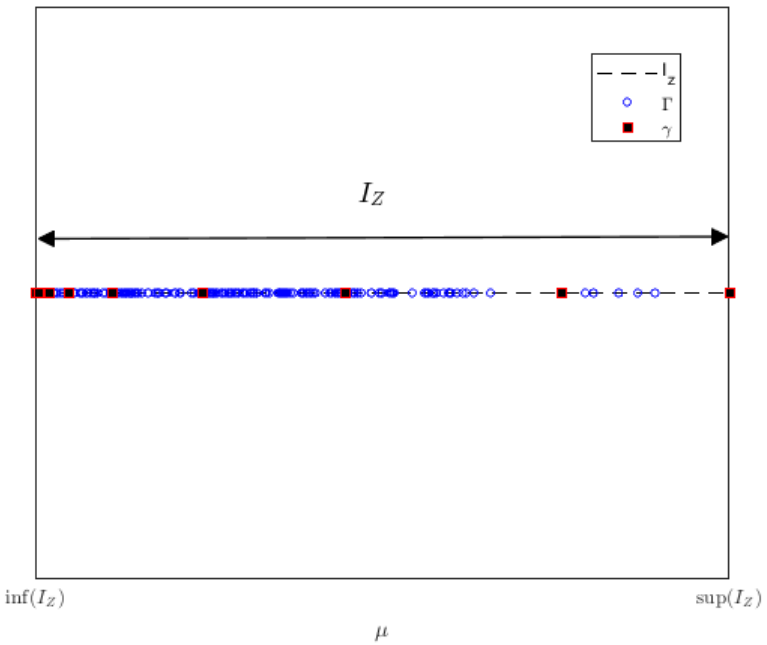


Figura 2.7: nodi *pivot* selezionati per il modello LF B

a raccogliere informazioni sulla variazione della soluzione con una risoluzione molto maggiore, che sfugge a tutti i modelli *Low-Fidelity* costruiti. In questo caso semplice il modello *High-Fidelity* che ci siamo procurati può essere applicato all'intero spazio discreto Γ senza problemi, per cui si è proceduto in maniera leggermente diversa da quanto descritto nel caso più generale nella trattazione del Capitolo 1:

(1) una volta ottenuto il campionamento Γ, si è in primo luogo valutata e allocata in memoria l'intera matrice delle soluzioni *High-Fidelity* $\mathbf{V}^{\mathrm{HF}}$ tramite l'applicazione iterativa del *solver* agli elementi finiti costruito. Nello stesso *script* `Main_Filo_HF.m` si procede contestualmente al calcolo di una matrice analoga a $\mathbf{V}^{\mathrm{HF}}$ che però salva nelle sue colonne le soluzioni HF per ogni punto dell'insieme $\{\tilde{z}_k\}$.

(2) a questo punto, avendo a disposizione il vettore $\mathbf{p}$ con il posizionamento dei nodi interpolanti in Γ in *output* dallo *step* di selezione, si è potuta effettuare una permutazione delle colonne di $\mathbf{V}^{\mathrm{HF}}$ per poi estrarne la sotto-matrice $\mathbf{V}^{\mathrm{HF}}(\gamma)$ che definisce lo spazio V^H desiderato.

Il modello HF realizzato si può vedere in figura 2.8, dove è messo a confronto con il modello HF di tipo A. Lo spostamento raffigurato è indotto

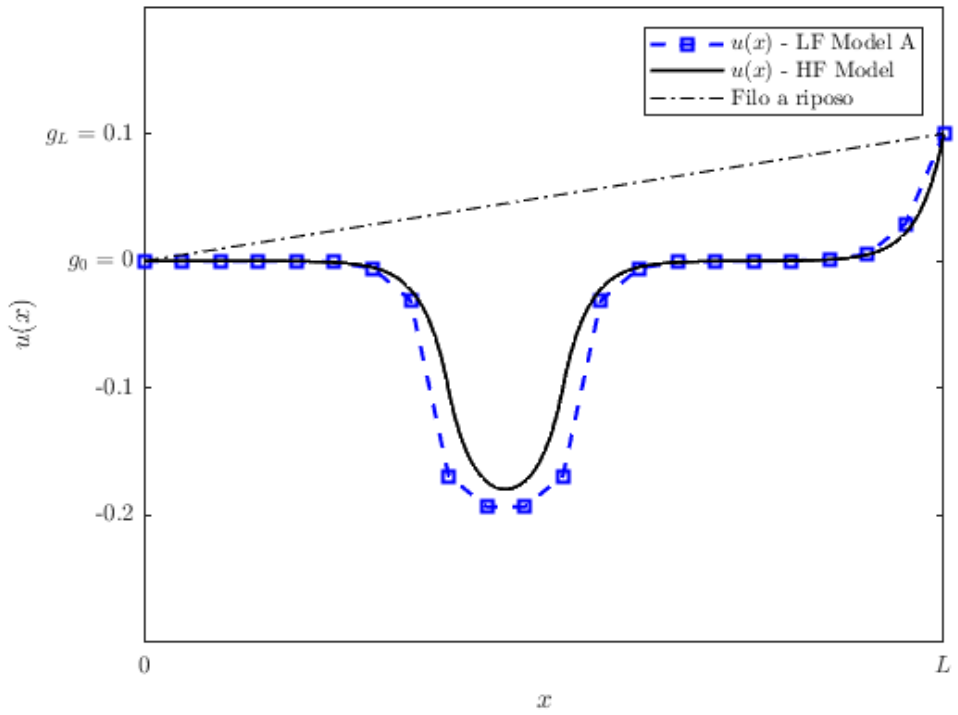


Figura 2.8: Confronto tra i modelli HF e LF tipo A – f concentrata

da un carico concentrato in una zona molto ristretta del dominio; si noti come la grande densità di punti della soluzione HF (in nero) riesca a risolvere efficacemente anche i gradienti maggiori al centro della griglia. Per una situazione del genere, lo scostamento rispetto a una griglia grossolana come quella di tipo A appare evidentissimo: le due soluzioni si sovrappongono solo negli intervalli in cui sono praticamente nulle in assenza di forze applicate.

4 Risultati: convergenza ed errori

A questo punto dell'analisi si possono ricapitolare gli elementi a disposizione, pre-calcolati e allocati in memoria nelle apposite matrici:

1. Le soluzioni LF calcolate su Γ e caricate nelle matrici $\mathbf{V}_{mod}^{\mathrm{LF}} \in \mathbb{R}^{N_{mod}^{L} \times M}$ (si ricorda che $M = 1000$) per ciascuna delle tre griglie A,B e C costruite.

2. Le soluzioni LF e HF calcolate su $\{\tilde{z}_k\}$ caricate nelle corrispondenti matrici $\tilde{\mathbf{V}}_{mod}^{\mathrm{LF}} \in \mathbb{R}^{N_{mod}^{L} \times N}$ e $\tilde{\mathbf{V}}_{\mathrm{HF}} \in \mathbb{R}^{N_H \times N}$ (si ricorda che $N = 10^4$).

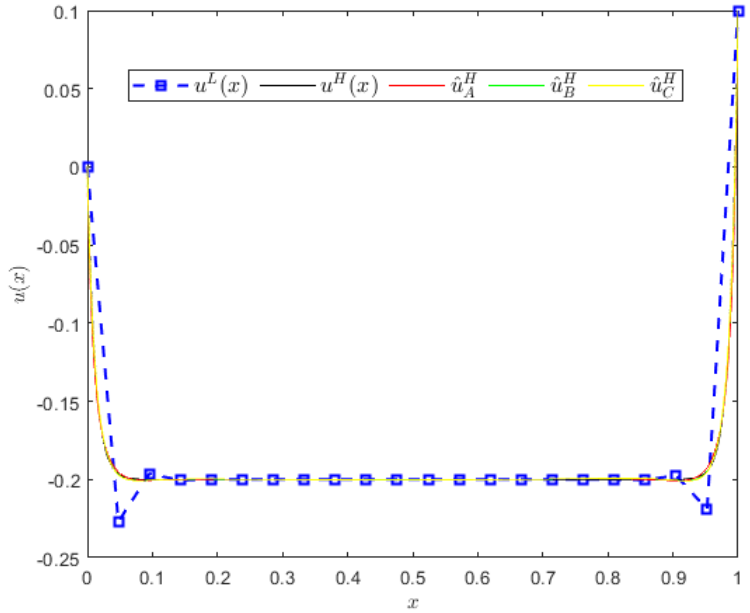


Figura 2.9: Convergenza delle ricostruzioni ottenute sul modello *High-Fidelity*

3. I sotto-spazi γ_A,γ_B e γ_C contenenti i nodi selezionati da ciascuno spazio LF, la cui dimensione m è al più pari a 15.

4. Le soluzioni HF calcolate sull'intero insieme Γ (e a maggior ragione si hanno quelle su $\gamma \subset \Gamma$) caricate nella matrice $\mathbf{V}_{\text{HF}} \in \mathbb{R}^{N_H \times M}$.

Non resta che applicare la legge di interpolazione costruita e definita nella (1.20) per ottenere le ricostruzioni in modalità *High-Fidelity* del vettore $u^L(z)$ per qualunque valore di $z \in I_Z$ desiderato. In particolare, si sono ricostruite:

(a) tutte le soluzioni LF in $\mathbf{V}_{mod}^{\text{LF}}$ per analizzarne la convergenza sulle HF corrispondenti, visto che siamo stati in grado di calcolarle. Ricordiamo che l'algoritmo di ricostruzione (essendo sostanzialmente definito da una proiezione sullo spazio U_γ^L) dovrebbe funzionare meglio su vettori che appartengono all'insieme delle soluzioni che hanno guidato la selezione nodale.

(b) tutte le soluzioni salvate nelle $\tilde{\mathbf{V}}_{mod}^{\text{LF}}$, in modo da poter calcolare una stima degli errori medi e massimi commessi in un caso più generale, utilizzando le (1.25).

L'ultima annotazione da fare riguarda la prima delle (1.24), che definisce il calcolo della norma di tipo L^2 utilizzata poi nella (1.25b) tramite un integrale che si riporta di seguito per maggiore linearità di esposizione

$$\|f(\mathbf{x})\|_{L^2(\mathbb{D})} = \sqrt{\int_{\mathbb{D}} |f|^2 d\mathbf{x}}$$

Chiaramente, a seconda delle caratteristiche topologiche dell'insieme $\mathbb{D}$, l'integrale si declina in maniera diversa. In questo caso unidimensionale in cui il nostro dominio fisico è un semplice intervallo $I_Z \subset \mathbb{R}$, bisogna calcolare un integrale singolo di una funzione definita per punti. *i.e.*:

$$\|f(\mathbf{x})\|_{L^2(D)} = \sqrt{\int_0^L |f|^2 dx}$$

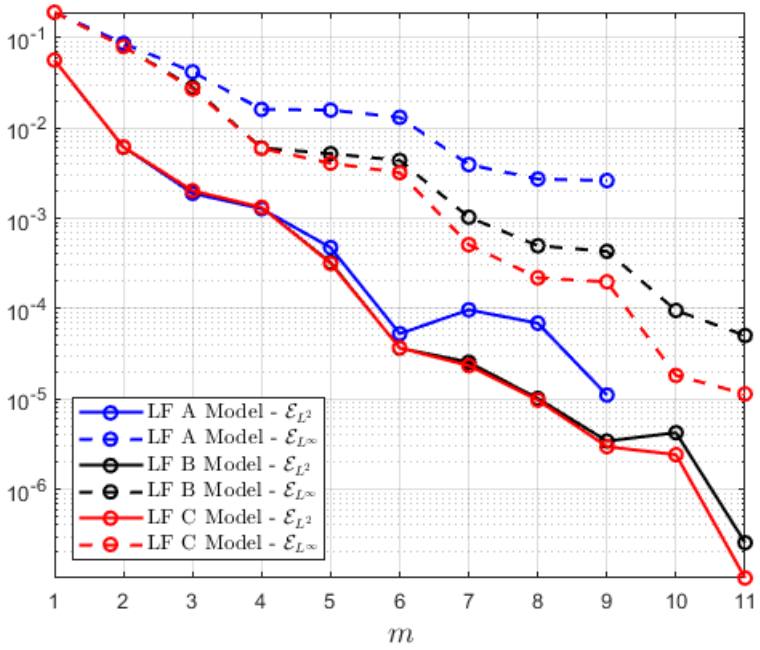
che è stata implementata nel codice tramite una semplice *function* `Normf2.m` che utilizza una quadratura numerica (il metodo dei trapezi, preciso al primo ordine) per calcolare l'integrale sotto radice.

La figura 2.9 mostra la convergenza delle ricostruzioni ottenute con i 3 modelli *Low-Fidelity* considerati sulla soluzione *High-Fidelity* per un vettore appartenente all'insieme delle $u^H(\Gamma)$.

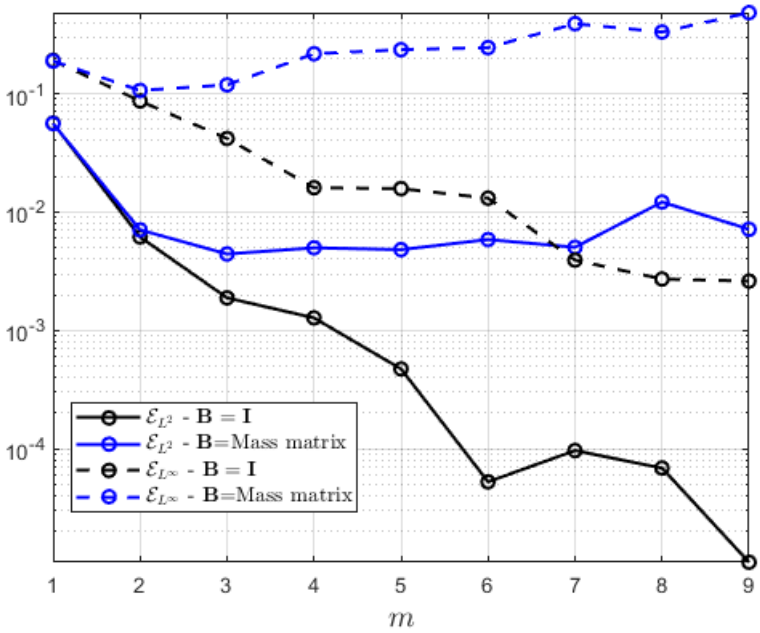
- Le varie u sono ovviamente selezionate per lo stesso valore di μ scelto casualmente con la sola accortezza che non fosse un nodo di interpolazione presente in nessuno degli insiemi $\gamma_{\text{A,B,C}}$ [3].
- Le ricostruzioni sono effettuate tutte per lo stesso numero di nodi *pivot* $m = 4$.
- L'unica soluzione *Low-Fidelity* mostrata per confronto è quella più grossolana di tipo A.

Si nota come per il valore di μ rappresentato (piuttosto basso) la soluzione LF abbia grossi problemi di risoluzione il gradiente agli estremi che si mostrano tramite delle piccole oscillazioni di natura completamente numerica. In verità, nella maggior parte dei casi per degli spostamenti indotti da una forza f uniforme su D, anche le LF si comportano piuttosto bene approssimando con buona accuratezza la soluzione. Si vede infatti anche in questo caso problematico che per una buona porzione centrale del dominio la soluzione è pressoché costante e il modello *Low-Fidelity* risulta preciso.

[3] in cui banalmente l'algoritmo fornisce una ricostruzione "esatta" selezionando il singolo vettore di base in $U^H(\gamma)$ tramite un vettore dei coefficienti $\mathbf{c} = \mathbf{e_i}$ i-esima componente della base canonica, con i posizione del nodo *pivot* in γ



(a) *N_L parametro libero* – $\mathbf{B} = \mathbf{I}$



(b) *scelta di* $\mathbf{B}$ *parametro libero* – $N_L = cost$

Figura 2.10: Filo elastico – Errori di ricostruzione su $\{\tilde{z}_k\}$

Le ricostruzioni sono invece totalmente sovrapposte e (perlomeno al livello grafico) indistinguibili dalla u^H su cui convergono; d'altronde ci aspettavamo che la qualità delle approssimazioni *Multi-Fidelity* fosse ottima in questo caso, sia per il buon comportamento delle LF appena menzionato, sia perché il vettore ricostruito è in $u^L(\gamma)$.

In figura 2.10 è invece mostrato in scala logaritmica il decadimento degli errori di ricostruzione commessi su $\{\tilde{z_k}\}$:

(a) la 2.10a mostra le ricostruzioni ottenute passando all'Algoritmo 1 una matrice B che è semplicemente la matrice identità di dimensioni $N^L_{mod} \times N^L_{mod}$, variando invece il modello utilizzato per ricostruire: il decadimento degli errori è pressoché monotòno esponenziale in tutti i casi, diminuendo di diversi ordini di grandezza per una crescita contenuta del numero di nodi interpolanti in ascissa. Si noti inoltre come la scelta del modello LF non influenzi in modo cruciale la qualità della ricostruzione, a meno che non sia davvero troppo grossolano, come nel caso del modello A; anche la condizione di *ill-conditioning* viene raggiunta per un m più basso (il rango della matrice $\mathbf{V}^{\mathrm{HF}}_{\mathrm{A}}$ è infatti più basso rispetto alle altre due).

(b) la 2.10b tiene invece fermo il numero di nodi della griglia *Low-Fidelity* al caso A, e mostra il confronto tra le ricostruzioni ottenute passando all'Algoritmo 1 prima la matrice identità, come nel caso precedente, e poi la matrice di massa introdotta nella sezione 1.2. Le curve in blu mostrano chiaramente che la scelta della matrice di massa non porta nessun vantaggio, ma è anzi controproducente per la strategia in questo caso degradandone le prestazioni.

2.2 La membrana elastica

Il problema della membrana elastica è la naturale generalizzazione al caso 2D1D del modello del filo appena discusso nel paragrafo 2.1 a pagina 24 ed è stato di conseguenza affrontato nell'approccio numerico e nella strategia *Multi-Fidelity* in maniera (quasi) del tutto analoga. Per via di queste considerazioni e al fine di snellire la trattazione epurandola da eventuali parti ridondanti, questa sezione conserverà la struttura della precedente dandone però per acquisiti i concetti chiave e passandovi dunque più concisamente. Il *focus* maggiore e più dettagliato verrà dedicato alle differenze significative tra i due modelli.

2.2.1 Modello matematico

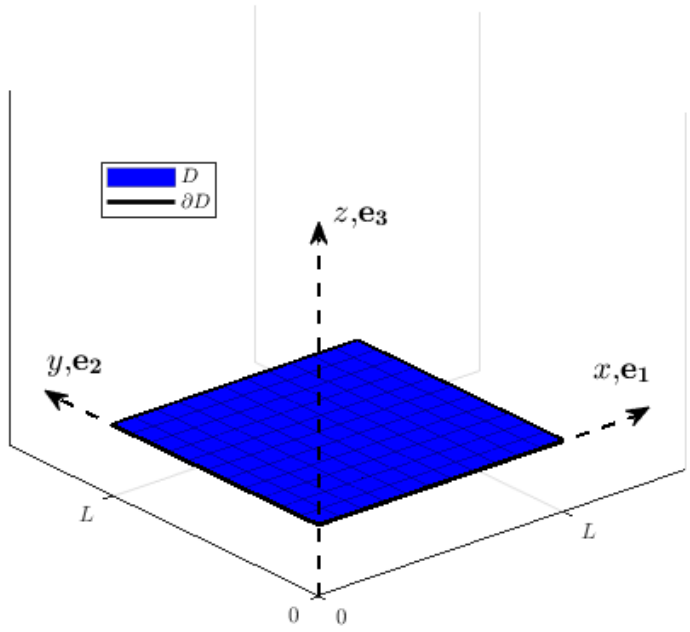


Figura 2.11: Membrana elastica a riposo – modello del piano mediale

Si consideri una membrana *sottile* (*i.e.* dallo spessore trasversale trascurabile rispetto alla sezione in pianta) la cui sezione centrale, in assenza di forze esterne applicate, giace nel piano (x, y) con riferimento alla stessa terna cartesiana utilizzata per il modello del filo e raffigurata in figura 2.11. La regione di piano occupata dalla membrana a riposo rappresenta il dominio di calcolo D su cui valutare gli spostamenti.

Il dominio D rappresentato in 2.11 è un quadrato in $\mathbb{R}^2$ definito semplicemente dalle coppie di punti

$$D = \{(x, y) \mid 0 \leq x \leq L \quad \wedge \quad 0 \leq y \leq L\}$$

o in notazione più breve: $D = [0, L] \times [0, L]$; esso rappresenta l'effettivo dominio di calcolo sul sarà realizzata la discretizzazione e la risoluzione del problema numerico, anche se tutta la notazione utilizzata in questo paragrafo è valida più in generale, per un qualsiasi $D \in \mathbb{R}^2$.

Le membrana si suppone soggetta alle stesse forze **f** e **r** del caso precedente, con le medesime ipotesi e semplificazioni. Stavolta si possono trascurare tutte le variazioni $\partial(\cdot)/\partial z$ mentre si considererà che le variabili in gioco abbiano una distribuzione bidimensionale, *i.e.* che siano delle funzioni di (x, y) definite su D.

L'unica componente di spostamento apprezzabile è ancora una volta $u_3 = \mathbf{u} \cdot \mathbf{e_3}$, per cui si definisce la funzione spostamento soluzione del modello presentato come

$$u_3 \simeq u = u(x, y) : D \to \mathbb{R}$$

e allo stesso modo

$$f_3 \simeq f = f(x, y) : D \to \mathbb{R}$$

che può essere una funzione continua anche a tratti.

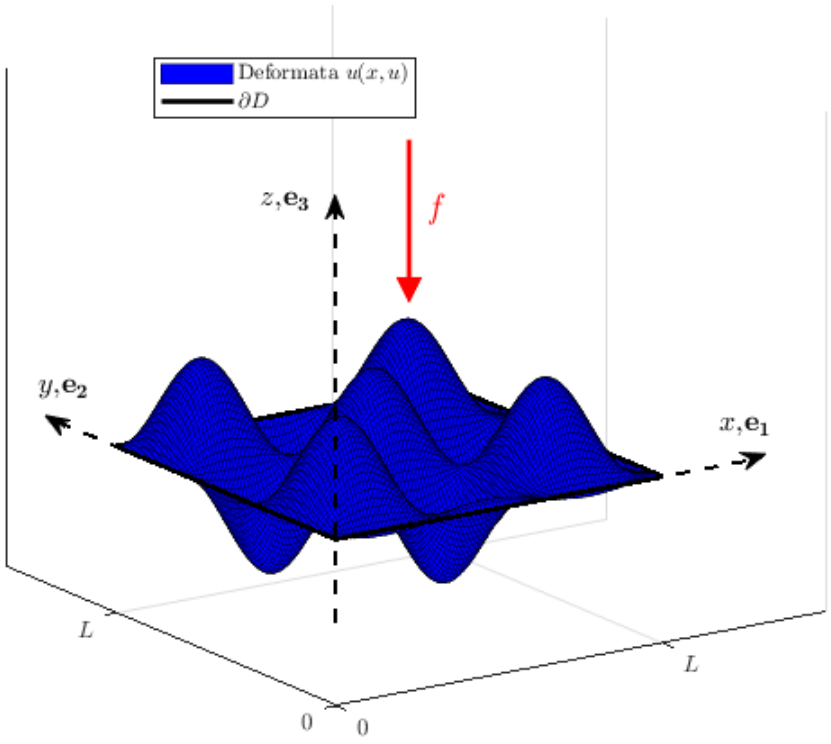


Figura 2.12: Deformata statica sotto l'azione di una forza di tipo sinusoidale

Tenuto conto di queste ipotesi, l'equazione di equilibrio della membrana in forma differenziale proiettata lungo la direzione z (cioè moltiplicata scalarmente per $\mathbf{e_3}$) diventa:

$$\begin{cases} \nabla \cdot \boldsymbol{\tau} + f - \gamma u = 0 & \text{in } D \\ \boldsymbol{\tau} = \mu \nabla u & \text{in } D \\ u = 0 & \text{su } \partial D \end{cases} \tag{2.7}$$

in cui $\boldsymbol{\tau}$ è il vettore degli sforzi di taglio che estrae dal versore delle tensioni $\boldsymbol{\sigma}$ solo le componenti

$$\boldsymbol{\tau} = (\tau_{31}, \tau_{32})$$

la cui definizione riportata in forma vettoriale nella seconda delle (2.7) richiama di nuovo la relazione lineare di Hooke approssimata trascurando come detto gli spostamenti u_1 e u_2 rispetto ad u_3.

In forma più compatta, e considerando delle condizioni al contorno eventualmente non omogenee sul bordo di D, (2.7) diventa

$$\begin{cases} -\nabla \cdot (\mu \nabla u) + \gamma u = f & \text{in } D \\ u = g(x, y) & \text{su } \partial D \end{cases} \tag{2.8}$$

che rappresenterà l'effettivo modello matematico di riferimento per l'implementazione nel codice. Si noti che $g(x, y) : \partial D \to \mathbb{R}$ è una funzione assegnata che prescrive la posizione di tutti i punti $(x, y) \in \partial D$.

Nella figura 2.12 a fronte è mostrata a titolo di esempio una soluzione $u(x, y)$ indotta da una forza distribuita in maniera continua su D formata da più onde sinusoidali, per condizioni al contorno di Dirichelet *omogenee*.

2.2.2 Discretizzazione e implementazione nel codice

Per il caso della membrana non si è riformulata l'equazione modello in maniera integrale per ottenere la formulazione variazionale del problema; infatti, a differenza del filo elastico, si è implementato un *solver* alle *differenze finite* nello *script* `Membrana_Dir_FinDiff_Solver.m` che si serve proprio dell'equazione in forma differenziale così come presentata nella (2.8) per poi discretizzarne le derivate tramite uno schema centrato del secondo ordine. Ancora una volta, una derivazione completa e rigorosa del metodo numerico adottato esula dagli scopi di questa tesi e per tutti gli approfondimenti si rimanda ai numerosi testi che discutono i metodi numerici di base presenti in letteratura [*e.g.* 7, 10].

Il *solver* utilizzato ha caratteristiche molto simili al caso del filo elastico:

- Considera una forza f che carica la membrana passata in *input* tramite un *function handle* che è assunta continua. Con la formulazione differenziale (e la conseguente discretizzazione alle differenze finite) si perde infatti la possibilità di trattare il caso del carico concentrato. Ancora una volta, in ogni caso, la ricostruzione è stata effettuata per delle soluzioni ottenute con una f uniformemente distribuita su D.

- Può considerare eventualmente un coefficiente di taglio $\mu = \mu(x, y)$ variabile (in modo continuo) su D che viene passato in *input* tramite un *function handle* che ne definisce l'andamento. Visto che comunque il parametro μ è l'unico nello spazio I_Z, verrà sempre considerato come un valore costante per permettere all'algoritmo di cogliere la sensibilità delle variazioni $\partial u/\partial \mu$. In particolare, con quest'assunzione si può effettuare una riscrittura dell'equazione modello (2.8) del tipo:

  $$-\mu\,\Delta u + \gamma u = f$$

  in cui il simbolo $\Delta(\cdot) = \partial^2(\cdot)/dx^2 + \partial^2(\cdot)/dy^2$ è il *Laplaciano*.

- Il coefficiente di richiamo elastico γ è supposto costante a priori.

- La discretizzazione alle differenze finite considera un dominio rettangolare di dimensioni $L_x \times L_y$ (di *default* si assume un quadrato $L_x = L_y = L$) e discretizza ciascuna delle due dimensioni indipendentemente con due griglie costituite da N_x ed N_y nodi, rispettivamente. Per valori uguali si utilizza la notazione N senza pedice. L'intera griglia di calcolo bidimensionale (i, j) – di cui si può vedere un esempio in figura 2.13 – è ottenuta ripetendo le due griglie monodimensionali per tutta la lunghezza di L, nelle due direzioni.

- Considera delle condizioni al contorno di Dirichelet eventualmente *non omogenee*, come descritto matematicamente nella (2.2); in particolare, per le ricostruzioni effettuate, su ciascun lato del quadrato D si è imposta una funzione g diversa di tipo costante, polinomiale o trigonometrica, con l'unico vincolo di congruità agli angoli del dominio (la membrana non può "spezzarsi").

- Restituisce in *output*:

  (a) la deformata statica sotto forma di vettore $u \in \mathbb{R}^{N^2}$, che riordina i punti di griglia in modo *lessicografico* per passare dalla notazione a due indici (i, j) a quella a unico indice l. Questo è necessario

per risolvere il sistema algebrico derivante dalla discretizzazione; si ricorda inoltre che le soluzioni che formano gli spazi V^L e V^H della strategia *Low-Fidelity* sono vettori unidimensionali, quindi la soluzione del sistema così indicizzata è già comoda per la costruzione delle matrici da passare al *selection step*.

(b) una matrice $U(i,j) \in \mathbb{R}^{N\times N}$ che contiene la deformata statica nei nodi della griglia bidimensionale. È sostanzialmente u ricostruito sotto forma matriciale, ed è utile per la visualizzazione delle soluzioni ottenute.

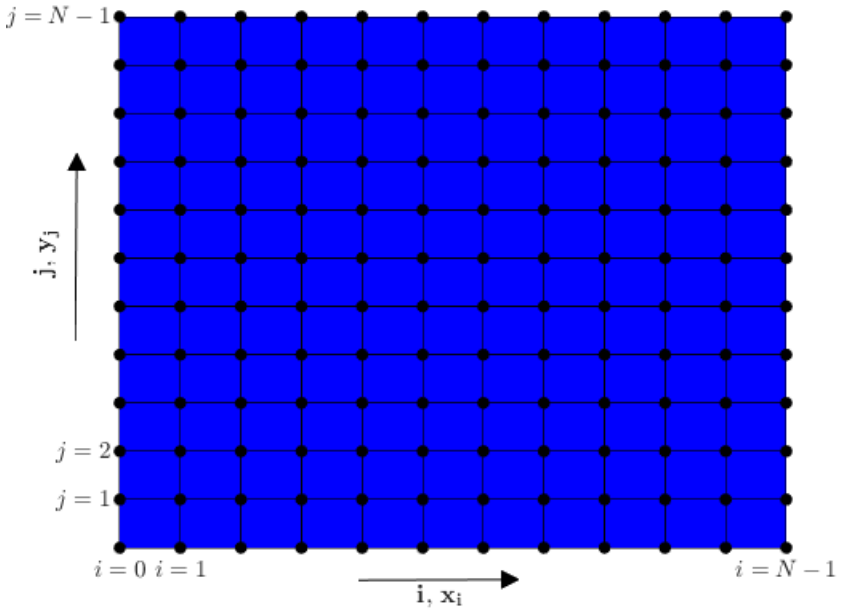


Figura 2.13: Griglia di calcolo 2D equispaziata

La figura 2.13 mostra un esempio di griglia equispaziata, rappresentativa di quelle su cui poi sono state calcolate le soluzioni u^L ed u^H. Per passare dalla notazione a doppio indice al vettore u cercato si utilizza come detto un orientamento lessicografico, che seleziona solo i punti *interni* a partire dalla riga più in basso della griglia ($j = 1$, senza cioè considerare il bordo del dominio $j = 0$ su cui valgono le condizioni di Dirichelet), da sinistra verso destra, *i.e.* per i crescenti. Il passaggio da un'indicizzazione all'altra si ottiene con

$$(i,j) \longleftrightarrow (l): \quad l = i + (j-1)(N-2), \quad 1 \leq i,j \leq N-2 \tag{2.9}$$

da cui deriva direttamente la lunghezza del vettore u dei nodi interni

$$l_{\max} = (N-2)^2 = N_{\text{int}}^2$$

che definisce la dimensione del problema numerico, di ordine quadraticamente superiore rispetto a quello del filo elastico, come era naturale attendersi.

Il sistema algebrico lineare che risolve il problema numerico è formalmente identico al precedente:

$$\mathbf{A}\mathbf{u} = \mathbf{f} \tag{2.10}$$

in cui ancora una volta $\mathbf{A}$ è simmetrica e data dalla somma dei due contributi disaccoppiati:

(a) Matrice di rigidezza $\mathbf{A}_\mu$
ha in questo caso una forma leggermente più complessa di tipo *pentadiagonale*. Per un coefficiente di taglio uniforme μ e una griglia equispaziata di passo $h = \frac{L}{N-1}$ il generico elemento a_{jk} è dato da:

$$a_{jk} = \frac{\mu}{h^2} \begin{cases} 4 & \text{se } k = j \\ -1 & \text{se } k = j \pm 1, \text{ con } j \not\equiv 0 \mod (N_{\text{int}}) \\ -1 & \text{se } k = j \pm (N_{\text{int}}) \\ 0 & \text{altrove} \end{cases} \tag{2.11}$$

(b) Matrice di massa $\mathbf{A}_\gamma$
è una semplice matrice diagonale, che addirittura nel caso considerato di $\gamma = \text{cost}$ si riduce a

$$\mathbf{A}_\gamma = \gamma\, \mathbf{I}_{N_{\text{int}}^2} \tag{2.12}$$

Il problema si riduce dunque anche in questo caso all'inversione di una matrice piuttosto sparsa dalla struttura pentadiagonale.

2.2.3 Applicazione della strategia *Multi-Fidelity*

Le scelte effettuate nella declinazione della strategia *Multi-Fidelity* ricalcano fedelmente quelle già introdotte per il caso del filo, per cui si rimanda alla sezione 2.1.3 per una discussione più approfondita al riguardo; in questo paragrafo verranno invece brevemente presentati gli insiemi e i modelli costruiti al fine di mostrare i risultati ottenuti.

1 Campionamento dello spazio I_Z

Con i valori di media e deviazione standard (pari rispettivamente a 1.5 ed 1) scelti a priori si è proceduto come segue:

(a) in primo luogo si è ottenuto il *set* di punti Γ distribuiti normalmente attorno alla media (al netto della correzione in valore assoluto apportata per scartare i valori eventualmente negativi).

(b) i valori minimo e massimo dell'insieme così costruito sono stati assegnati rispettivamente al $\sup(I_Z)$ e all'$\inf(I_Z)$, definendo lo spazio parametrico come un intervallo su $\mathbb{R}$.

(c) una volta conosciuto I_Z si è potuto procedere a un campionamento con distribuzione di probabilità uniforme che lo descrivesse, ottenendo dunque in questo modo l'insieme $\{\tilde{z_k}\}$

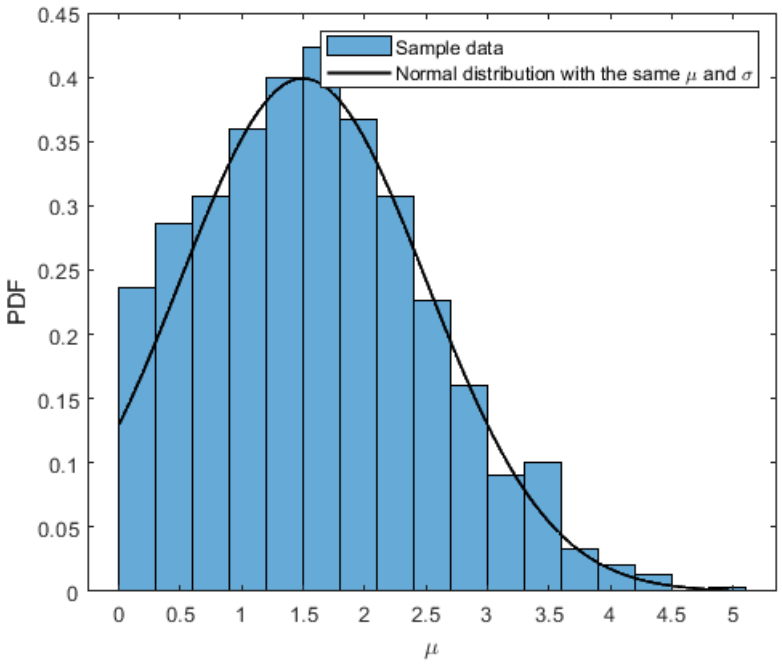


Figura 2.14: Distribuzione gaussiana del campionamento Γ sul parametro μ

L'insieme Γ già corretto in valore assoluto ha una cardinalità di $M = 10^3$; la distribuzione dei suoi punti è mostrata in figura 2.14, in cui contestualmente si mostra la gaussiana analitica per confronto.

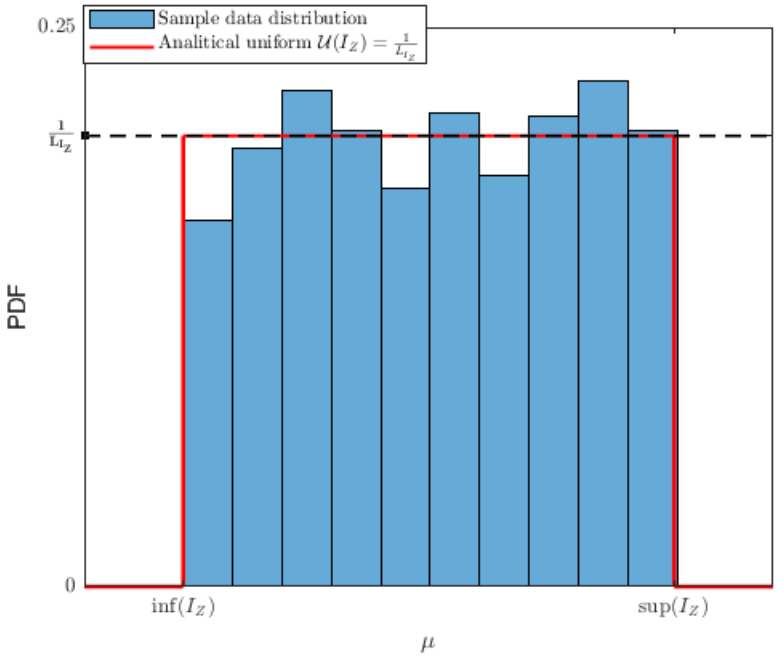


Figura 2.15: Distribuzione uniforme del campionamento $\{\tilde{z}_k$ sul parametro μ

Questa volta si è preferito costruire l'insieme $\{\tilde{z}_k\}$ per il calcolo degli errori con la stessa dimensione di Γ (*i.e.* $N = M = 10^3$) [4], visto l'andamento quadratico con la dimensione di griglia della complessità computazionale nel caso della membrana. Questo dovrebbe comunque essere sufficiente ad ottenere una stima degli errori significativa e mediata su tutto l'intervallo I_Z.

La figura 2.15 mostra la distribuzione di probabilità dell'insieme così campionato a confronto con la *step function* analitica definita al solito da $\mathcal{U}(I_Z) = 1/L_{I_Z}$. Si nota come il campionamento sia più "grezzo" rispetto al caso del filo elastico (figura 2.4), per via della cardinalità scelta di un ordine di grandezza inferiore.

Infine, in figura 2.16 è mostrata la collocazione dei punti campionati sull'intervallo I_Z, che come si vede anche graficamente rispetta le prescrizioni fatte nella scelta delle distribuzioni. Γ rappresenta l'errore di misura sul coefficiente di taglio μ e si attesta attorno alla media che dovrebbe rappresentare il valore reale, mentre $\{\tilde{z}_k\}$ copre omogeneamente l'intero dominio parametrico per

[4] *n.b.*: in questo contesto N si riferisce alla dimensione dell'insieme di punti $\{\tilde{z}_k\}$ e non al numero di punti di griglia del precedente paragrafo. L'ambiguità di notazione deriva dalla volontà di conservare una consistenza di nomenclatura con quella introdotta per il caso generale nel Capitolo 1

una buona rappresentazione dell'errore di costruzione su una qualsiasi $u(z)$ desiderata.

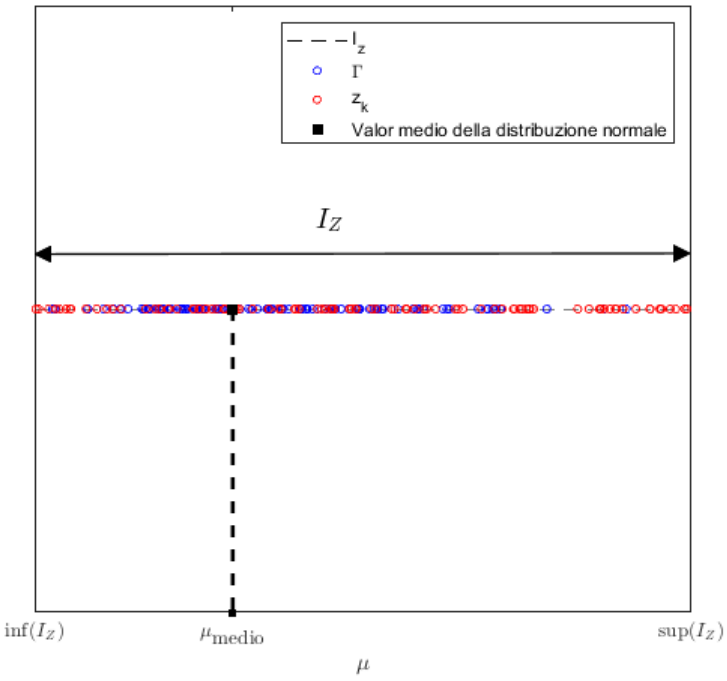


Figura 2.16: Collocazione dei campionamenti effettuati Γ e $\{\tilde{z}_k$ sull'intervallo I_Z

2 Modello *Low-Fidelity* e selezione nodale

La classificazione di un modello come HF o LF dipende solo dalla finezza della griglia numerica sulla quale è stata discretizzata l'equazione modello. Si sono costruiti tre modelli *Low-Fidelity* indipendenti:

Modello A: il più grossolano tra quelli a disposizione; la griglia conta 12 nodi per ciascuna dimensione del dominio, quindi il valore di $N_L = \dim(V^L)$ è pari a $(N-2)^2 = 100$. Si noti come sebbene le due dimensioni di D siano entrambe discretizzate con un passo più grezzo rispetto al modello tipo A del filo, la dimensione numerica del problema è pari a quella del filo tipo C. Questo rende bene l'idea della maggiore complessità derivante dal passaggio al caso bidimensionale.

Modello B: di finezza intermedia; $N = 32 \Rightarrow N_L = (N-2)^2 = 900$, circa della stessa dimensione del modello u^H nel caso del filo elastico.

Modello C: il più fine costruito tra i LF; $N = 52 \Rightarrow N_L = (N-2)^2 = 2500$

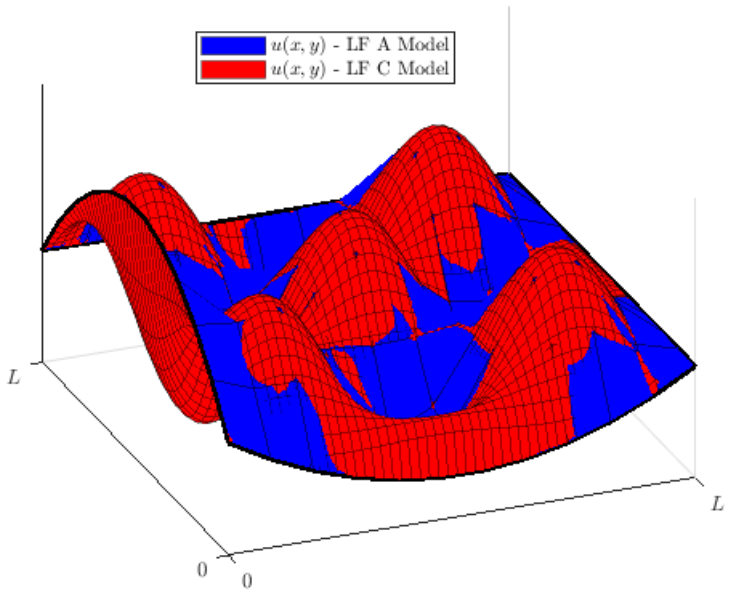


Figura 2.17: Confronto su una soluzione per f variabile dei modelli LF A e C

In figura 2.17 si può apprezzare la differenza di risoluzione tra i due modelli LF A e C, per un caso in cui le condizioni di Dirichelet sono assegnate da funzioni paraboliche e sinusoidali e la forza f è anch'essa una combinazione di polinomi e funzioni trigonometriche.

Per ciascun modello LF si è di seguito costruita la matrice $\mathbf{V}_{\text{LF}}$ con tutte le soluzioni $u^L(\Gamma)$ nello spazio discreto dei parametri. Questa viene quindi passata in *input* al *selection step* (Algoritmo 1) insieme a un valore massimo di nodi *pivot* m fissato pari a 15. Anche in questo caso si incorrerà nell'uscita per malcondizionamento della matrice di Gram prima di arrivare all'm desiderato; in ogni caso i nodi selezionati sono tutt'altro che insufficienti a ricostruire accuratamente la soluzioni, in generale e a maggior ragione per un caso di studio semplice come quello in esame.

La figura 2.18 mostra la collocazione degli 11 *pivot* selezionati guidando il processo con la matrice LF di tipo B. La maggiore concentrazione di nodi interpolanti presso l'estremo inferiore dell'intervallo I_Z si spiega notando che per valori di μ prossimi allo 0 la sensibilità di variazione delle soluzioni rispetto al parametro è molto elevata. Questo, per quanto in realtà non realistico

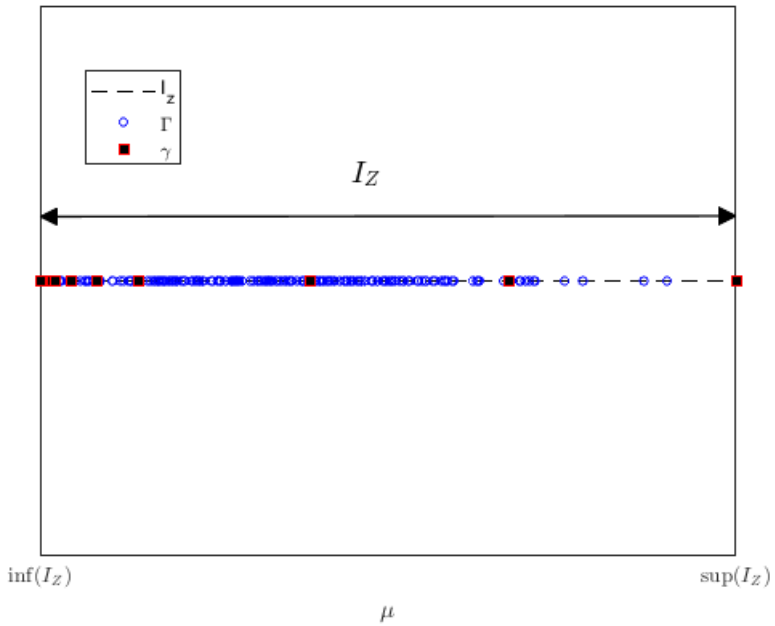


Figura 2.18: Collocazione dei nodi *pivot* su I_Z - LF tipo C

dal punto di vista fisico per un coefficiente di taglio, è molto interessante numericamente[5].

I punti rimanenti sono poi distribuiti ciascuno in una parte diversa del dominio a grande distanza dagli altri; l'algoritmo di selezione sta cioè "sforzandosi" di raccogliere informazioni su ciascun grande intervallo di variazione del parametro.

3 Modello *High-Fidelity*

Le soluzioni HF sono calcolate su una griglia fatta da 152 punti per ciascuna dimensione. Il numero di nodi interni alla griglia, e dunque la dimensione dello spazio V^H costruito dalle u^H è pari a $N_H = N_{\text{int}}^2 = 22500$. Nonostante le dimensioni notevolmente maggiori del problema, anche in questo caso si è potuto procedere senza particolari difficoltà al calcolo dell'intero insieme $u^H(\Gamma)$, salvato in colonna nella matrice $\mathbf{V}_{\text{HF}} \in \mathbb{R}^{N_H \times M}$ e successivamente

[5] Tutta la trattazione di questo Capitolo si configura infatti come esperimento puramente numerico dall'alto livello di astrazione; si noti infatti come le unità di misura delle quantità "fisiche" in gioco siano state volutamente omesse al fine della creazione di un *framework* matematico. I modelli fisici analizzati vanno intesi solo come riferimento concettuale e sono usati strumentalmente al solo scopo di testare le potenzialità dell'algoritmo implementato

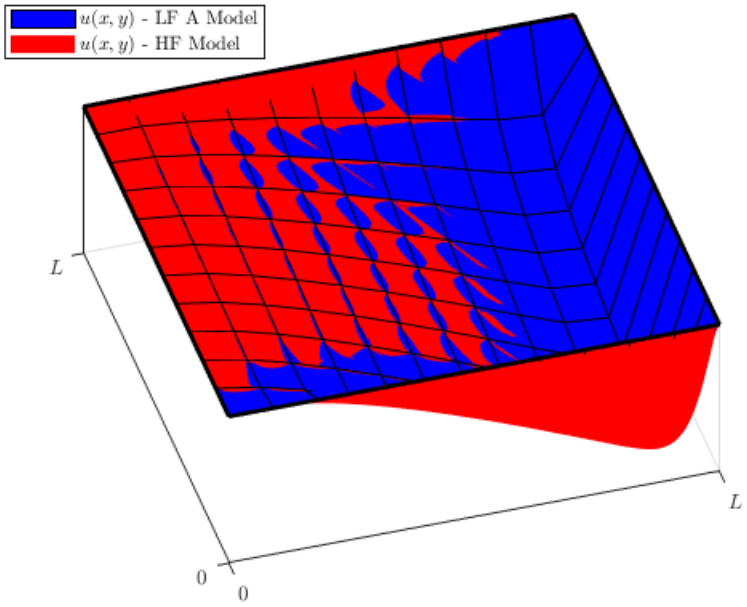


Figura 2.19: Modelli HF e LF tipo A a confronto su una particolare soluzione

permutato dal vettore $\mathbf{p}$ in *output* dal *selection step* per poterne estrarre le $u^H(\gamma)$.

L'insieme $\{\tilde{z}_k\}$, fatto anch'esso da 1000 elementi, è stato trattato nello stesso modo all'interno dello *script* `Main_Membrana_HF.m`.

La figura 2.19 mostra l'applicazione del modello HF costruito in uno specifico caso *test*; se ne può apprezzare la risoluzione per confronto al LF di tipo A mostrato in sovrapposizione per riferimento. La soluzione è calcolata per una forza f distribuita di tipo logaritmico, le condizioni al contorno sono di Dirichelet omogenee su tutto il bordo del dominio ∂D.

4 Risultati: convergenza ed errori

Le matrici e i vettori pre-allocati in memoria e utilizzati come *input* dell'interpolazione per la ricostruzione delle soluzioni sono i seguenti, definiti esattamente come nel paragrafo 2.1.3 a pagina 38:

- $\mathbf{V}^{\text{LF}}_{mod} \in \mathbb{R}^{N^L_{mod} \times M}$, con $M = 1000$ e $mod = \{\text{A,B,C}\}$
- $\tilde{\mathbf{V}}^{\text{LF}}_{mod} \in \mathbb{R}^{N^L_{mod} \times N}$ e $\tilde{\mathbf{V}}_{\text{HF}} \in \mathbb{R}^{N_H \times N}$, con $N = 1000$.
- γ_{A},γ_{B} e γ_{C}, con $m \leq 15$.

- $\mathbf{V}_{\text{HF}} \in \mathbb{R}^{N_H \times M}$.

Applicando la legge di interpolazione (1.20) (Algoritmo 2) si sono ricostruite le soluzioni salvate nelle $\tilde{\mathbf{V}}^{\text{LF}}_{mod}$, in modo da poter quantificare gli errori utilizzando ancora una volta le (1.25).

Per il caso della membrana, avendo già precedentemente appurato che l'algoritmo funziona ottimamente per i vettori $u^L(\Gamma)$ proprio per come è costruito (*i.e.* le soluzioni convergono accuratamente verso quelle del modello HF), si vuole confrontare il decadimento degli errori con le ricostruzioni del *set* indipendente $\{\tilde{z}_k\}$; potrebbe infatti essere di interesse la costruzione di uno spazio Γ comprendente alcuni punti inseriti *ad hoc* per i quali si sa già che si è interessati a conoscere delle soluzioni in modalità HF surrogata.

Il calcolo della norma L^2 sul dominio D si riduce in questo caso bidimensionale alla valutazione dell'integrale doppio di una funzione definita per punti:

$$\|f(\mathbf{x})\|_{L^2(D)} = \sqrt{\int_D |f(x,y)|^2 \, dxdy}$$

in cui si ricorda che $D = [0, L]^2$.

Prima di poter utilizzare una quadratura per approssimare il valore dell'integrale – si è optato ancora una volta per la formula dei trapezi applicata su entrambe le direzioni della griglia – bisogna ricostruire le soluzioni discrete u^H e $\hat{u}^H$ in forma matriciale per riottenere i punti posizionati sui nodi interni della griglia di calcolo mostrata in figura 2.13; si può ottenere facilmente il risultato desiderato invertendo la procedura di ordinamento lessicografico della (2.9).

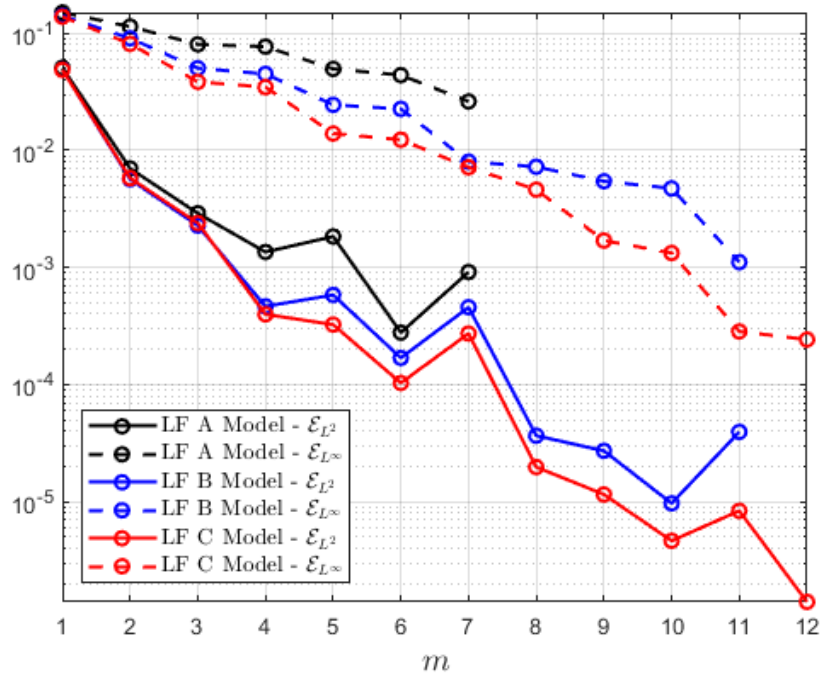
Il metodo appena descritto è implementato tramite la *function* `NormL22D` dello *script* omonimo.

La figura 2.20 mostra in scala logaritmica l'andamento degli errori al crescere del numero degli elementi *pivot* m utilizzati per la ricostruzione. Si osserva che:

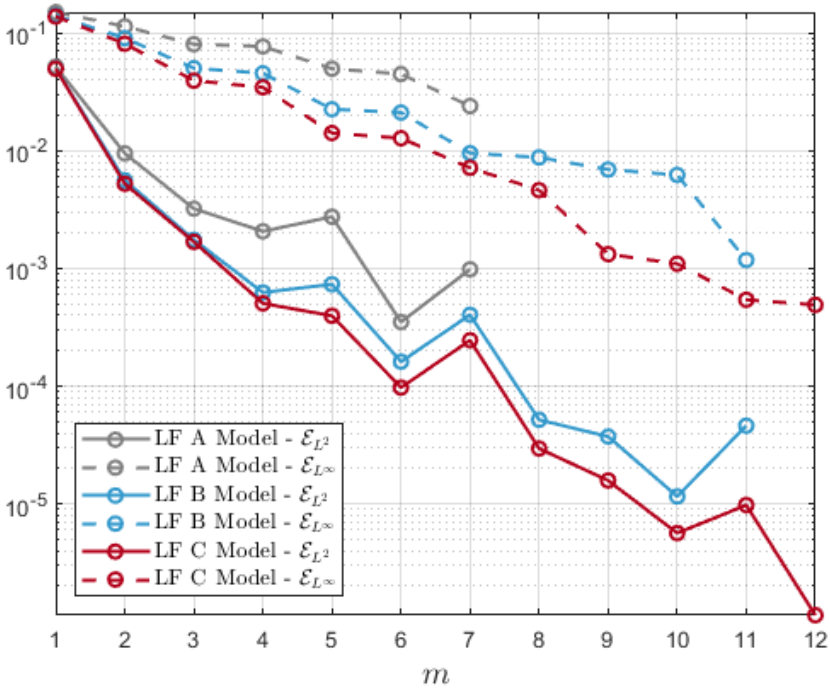
- Lo studio sul caso del filo elastico ci ha mostrato che l'utilizzo di una matrice di massa tridiagonale $\mathbf{B}$ per il calcolo delle componenti di $\mathbf{G}$ non era stato di alcun vantaggio; per la membrana, allora, si è preferito direttamente utilizzare la matrice identità con cui si ottenevano risultati decisamente migliori.

- Il decadimento degli errori resta globalmente monotòno – al netto di qualche estemporanea inversione di tendenza per taluni valori di m – e di natura esponenziale: a partire dall'ovviamente impreciso caso $m = 1$[6]

[6] Si noti che con un unico *pivot* l'algoritmo di ricostruzione non può far altro che selezionare la sola u^H a disposizione eventualmente riscalandola con un coefficiente c



(a) *Ricostruzioni su* $\{\tilde{z}_k\}$



(b) *Ricostruzioni su* Γ

Figura 2.20: Membrana elastica – Errori di ricostruzione

si guadagnano diversi ordini di grandezza in accuratezza utilizzando solo una manciata di elementi *pivot*.

- Il modello A si dimostra troppo grossolano per cogliere efficacemente le variazioni della soluzione nello spazio dei parametri. La curva corrispondente è interrotta molto precocemente per un valore di $m = 7$, per via di una matrice $\mathbf{V}_{\mathrm{A}}^{\mathrm{LF}}$ di rango troppo basso.

- La figura 2.20b mostra gli errori calcolati ricostruendo in modalità HF tutte le soluzioni $u^L(\Gamma)$; diversamente da quanto atteso, non si riscontrano sostanziali miglioramenti rispetto alle ricostruzioni su $\{\tilde{z_k}\}$ (2.20a), nonostante al *set* Γ appartengano anche i punti in γ per cui si dovrebbe commettere un errore praticamente nullo. Questo si può spiegare con le seguenti considerazioni:

  (a) la semplicità del caso in esame e la bontà dei modelli LF a disposizione rendono il comportamento dell'algoritmo praticamente indipendente dalla scelta di $z \in I_Z$. Infatti le soluzioni $\hat{u}^H(z_k)$ si comportano già egregiamente nell'approssimare le corrispondenti *High-Fidelity* e mostrano probabilmente la migliore prestazione possibile dell'algoritmo per questo caso di studio

  (b) I nodi del campionamento Γ hanno una distribuzione che riesce efficientemente a rappresentare lo spazio I_Z rendendo poi la ricostruzione ugualmente accurata anche per punti diversi da quelli che hanno guidato la selezione.

2.3 Confronto con altri casi notevoli

Le prestazioni dell'algoritmo *Bi-Fidelity* osservate per i due *test case* effettuati serviranno – come precisato nella presentazione di questo Capitolo – da riferimento per poter valutare la qualità delle ricostruzione sulla CFD; perciò, prima di passare al *framework* principale, è bene confrontare i risultati ottenuti sugli errori di ricostruzione con altri casi *benchmark* presenti nella letteratura di riferimento per questa tesi.

Si tratta sempre di problemi numerici standard di natura e complessità diversa dai due presentati qui e dunque significativi per valutare la riproducibilità dei risultati ottenuti. L'interesse principale è sulle possibili generalizzazioni delle varie "debolezze" dell'approccio proposto in questo lavoro per sottolineare il loro eventuale condizionamento sulla strategia.

Sensibilità parametrica. Un possibile problema potrebbe derivare dalla troppo esigua variazione delle soluzioni ottenute rispetto al parametro scelto. In [14] l'*equazione di Burgers* viene modellata alle differenze finite con approccio *Bi-Fidelity* in cui lo spazio parametrico è formata da una singola perturbazione (*random* uniforme) su una delle condizioni al contorno; in questo caso, le soluzioni sono affette da un'elevatissima sensibilità al parametro scelto.

I risultati ottenuti (che non vengono mostrati per ragioni di *copyright* non essendo stati prodotti autonomamente) sono perfettamente in linea con quelli ottenuti per filo e membrana.

Dimensione dello spazio parametrico. Il valore del numero di parametri incerti d – posto finora per tutta questa trattazione pari ad 1 – può influenzare significativamente il comportamento dell'algoritmo in particolare se combinato con una scarsa finezza del modello LF utilizzato; in [14] si propone una soluzione a un problema ellittico con diffusività *random*. Si nota molto bene l'appiattimento delle curve degli errori al crescere di m quando la dimensione d diventa molto grande; l'accuratezza delle ricostruzioni arriva ad essere di circa due ordini di grandezza inferiore a quella valutata per i nostri casi di studio.

Concordanza tra i modelli LF e HF. I due modelli *Low-Fidelity* ed *High-Fidelity* utilizzati per risolvere le equazioni modello in questo Capitolo fanno variare solo la finezza della griglia di calcolo di un metodo numerico che resta lo stesso in tutti casi. Una possibile generalizzazione è proposta in [8]: si affronta il problema della ricostruzione polinomiale troncata di una funzione *esatta* nota a priori. Si presentano due modelli LF ed uno HF:

1. il primo dei LF è della stessa natura di quello HF ed è ottenuto trascurando una parte della variazione funzionale di u.
2. il secondo dei LF ha un andamento totalmente diverso, ed ha il solo proposito di essere in grado di cogliere la dipendenza parametrica $\partial u/\partial z$

Le due ricostruzioni ottenute $\hat{u}_1^H$ e $\hat{u}_2^H$ sono coincidenti. La conclusione che se ne può trarre è interessante: la strategia *Bi-Fidelity* richiede *solo* che un modello LF sia in grado di approssimare la dipendenza parametrica di un corrispondente HF, anche se è completamente inaccurato nel rappresentare quella spaziale sul dominio di calcolo D. Ancora una volta, inoltre, si attesta la totale indipendenza della ricostruzione dal metodo numerico utilizzato.

Anche in questo caso, però, gli errori ottenuti a parità di m sono maggiori di tre ordini di grandezza rispetto a quelli ottenuti ad esempio in figura 2.20

o in altre parole, per raggiungere valori 10^{-6} su $\mathcal{E}_{L^2}$ bisogna utilizzare circa il doppio delle simulazioni HF.

In generale, soprattutto con riferimento a modelli più complessi, possiamo dire che si corre il rischio di sovrastimare l'accuratezza dell'algoritmo *Multi-Fidelity* valutandolo nei casi di studio presentati.

Capitolo 3

Applicazione alle simulazioni CFD

Questo Capitolo utilizza tutti gli strumenti finora costruiti, analizzati e validati in questo lavoro di tesi per andare al cuore del problema e adottare una strategia *Multi-Fidelity* per la riduzione dell'ordine della simulazione si un campo di moto fluidodinamico. Verranno riportati, ove ritenuto necessario, gli opportuni approfondimenti teorici sugli argomenti presentati.

3.1 Presentazione del caso di studio

3.1.1 Modello fisico

Si consideri il campo di moto bidimensionale di una corrente uniforme U_∞ che investe un profilo alare posto ad un'incidenza α mostrato in figura 3.1. Ricordando la definizione di *numero di Mach* come rapporto tra velocità (locale) della corrente e velocità del suono nelle specifiche condizioni termodinamiche

$$\mathrm{M} = \frac{u}{a}$$

si possono distinguere i flussi in *subsonici* ($\mathrm{M} < 1$) e *supersonici* ($\mathrm{M} > 1$); la condizione che li separa è ovviamente quella sonica ottenuta per $\mathrm{M} = 1$. Questo valore soglia è caratterizzante per il flusso in maniera cruciale, in quanto la velocità del suono rappresenta la velocità di propagazione delle perturbazioni di pressione nel campo di moto e regola cioè il passaggio di "informazioni" fisiche tra le particelle fluide: a diversi regimi di M possono corrispondere campi di moto totalmente diversi al livello fenomenologico.

Un campo di moto può definirsi sub(super)sonico solo se *tutti i suoi punti* sono localmente in condizioni sub(super)soniche. Per un profilo alare, sappiamo che il campo di velocità varia sensibilmente mentre la corrente fluida

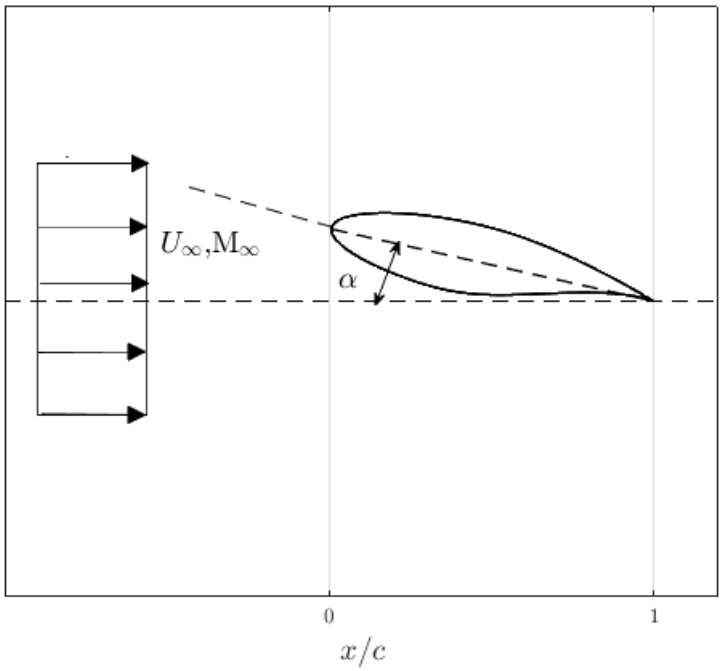


Figura 3.1: Profilo alare in incidenza investito da una corrente uniforme

ne lambisce la geometria; occorre dunque fare una distinzione più pertinente, basata sul riferimento della corrente indisturbata a monte (U_∞, M_∞) ma ponendo l'attenzione a ciò che sta accadendo sul profilo. Definendo x/c come la coordinata sulla corda del profilo normalizzata dalla sua lunghezza (*i.e.* $x/c \in [0, 1]$):

(a) Regime subsonico $\Rightarrow M_\infty < 0.6 \div 0.7$
tutti i punti sul profilo sono in condizioni subsoniche; è a sua volta ulteriormente suddivisibile in

- subsonico incompressibile: $M_\infty < 0.3$, per cui si considera l'informazione propagata istantaneamente in tutto il campo di moto e si assume densità $\rho = \text{cost}$
- subsonico compressibile: $M_\infty > 0.3$, per cui gli effetti di compressibilità non sono più trascurabili.

(b) Regime transonico $\Rightarrow M_\infty > 0.6 \div 0.7$
sulla superficie del profilo si trovano contemporaneamente zone subsoniche e supersoniche. In pratica l'accelerazione della corrente sul dorso porta il flusso a raggiungere le condizioni soniche per una stazione

$x/c < 1$. Da ciò consegue la formazione di un'*onda d'urto*. Nell'intorno del punto sonico e fino alla posizione dell'urto si forma la cosiddetta *bolla supersonica* all'interno di un campo subsonico. A valle dell'urto il campo di moto è subsonico, turbolento, con possibile separazione della vena fluida (dovuta al forte gradiente di pressione $dp/d(x/c)$ avverso generato dalla compressione dell'urto) e dunque stallo aerodinamico. La compresenza di due campi di moto così diversi tra loro contribuisce a complicare considerevolmente la risoluzione di un flusso di questo genere, sia da un punto di vista teorico che pratico (numerico). Il regime transonico si instaura a partire da un valore di M_∞ detto *critico*:

Mach critico inferiore $M_{\mathbf{Cr}}^{\boldsymbol{inf}}$, è il minimo valore di M_∞ per cui il primo punto sul profilo raggiunge localmente condizioni soniche. Il suo valore ha grande rilevanza in fase di progetto e si può stimare preliminarmente con le relazioni inviscide tra coefficiente di pressione C_p e M che sfruttano il fatto che al punto di massima velocità $M = 1$ corrisponda quello di minima pressione [1]

La maggior parte degli aerei per il trasporto civile sono progettati oggi per avere delle condizioni di crociera transoniche ($M_\infty \approx 0.75$).

(c) Regime supersonico $\Rightarrow M_\infty \geq 1$
non è possibile invece fornire una nozione altrettanto precisa di limite superiore del regime di coesistenza tra i flussi subsonico e supersonico, cioè di un eventuale M_{Cr}^{sup}. Per valori di M_∞ di poco inferiori all'unità gli urti su dorso e sul ventre (che si è nel frattempo formato) si spostano verso valle lasciando in campo supersonico la quasi totalità della superficie del profilo. Quando $M_\infty = 1$, teoricamente, i due urti si uniscono sul bordo di fuga e il flusso sul profilo è ovunque transonico tranne che per una piccola zona sul bordo d'attacco detta in questo caso *bolla subsonica*. Per $M > 1$ la fenomenologia cambia nuovamente, con la formazione di urto curvo staccato (*bow shock*) a monte del profilo, *i.e.* per una coordinata $x/c < 0$; appena a valle dell'urto il campo di moto è in alto subsonico con la formazione di una (piccola) bolla subsonica sul bordo d'attacco, mentre in tutti i restanti punti sulla superficie profilo il M locale è maggiore di 1. Una soglia per cui la regione subsonica scompaia del tutto non esiste – per via della presenza di raggio di curvatura al bordo d'attacco – ma dal punto di vista degli effetti globali per l'analisi del campo di moto possiamo considerare in regime totalmente supersonico i profili immersi in una corrente $M_\infty > 1$.

In figura 3.2 le topologie appena descritte per M_∞ nell'intorno dell'unità sono mostrate con delle visualizzazioni di flusso in galleria del vento. Le imma-

gini sono state tratte dal materiale didattico pubblicamente a disposizione sul sito del dipartimento di ingegneria meccanica ed aerospaziale dell'Università di Roma *La Sapienza*.

Si ricorda che qualunque valutazione quantitativa sui valori dei M_{Cr}^{inf} dipende sensibilmente dalla geometria del profilo utilizzato e dall'incidenza della corrente che lo investe α.

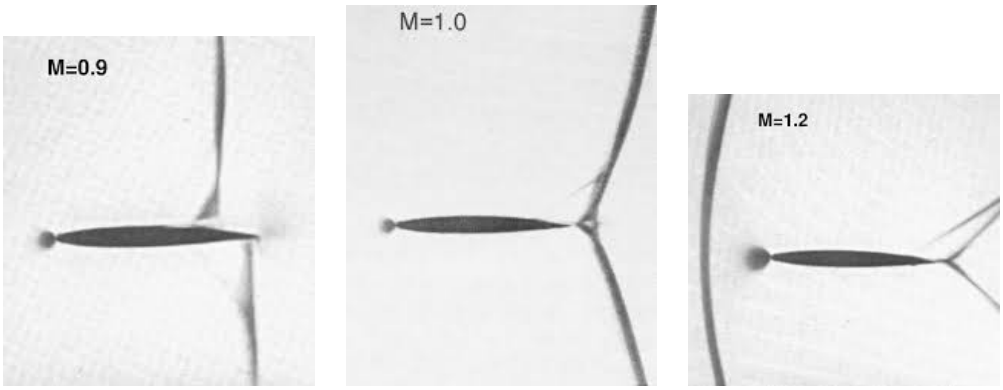


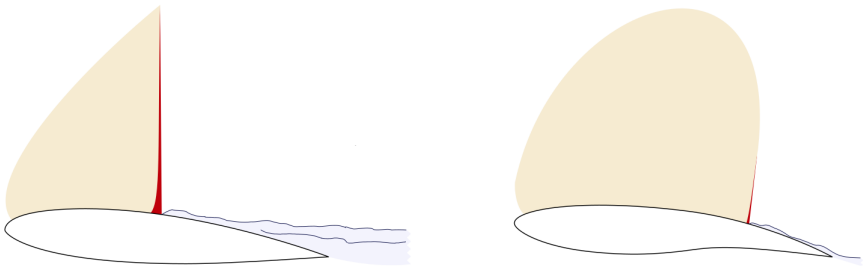
Figura 3.2: Visualizzazione del flusso su un profilo alare in regime transonico al variare di M_∞ – **fonte:** http://dma.ing.uniroma1.it

Con riferimento all'eterogeneità della fenomenologia appena descritta, non stupisce che il *design* della geometria di un profilo alare debba essere pensato *ad hoc* per adattarsi alle particolari esigenze del regime di flusso per il quale è progettato. Possiamo infatti distinguere due tipologie di geometria adottate:

I profili tradizionali o *subcritici* progettati con l'intento di spostare il più in alto possibile il valore di M_{Cr}^{inf} in modo da poter volare ad elevati M_∞ senza entrare in regime transonico. Le caratteristiche geometriche sono quelle convenzionali, *i.e.* piccoli raggi di curvatura e spessore massimo posizionato per $x/c \approx 0.5$. Si comportano molto male se utilizzati in regime transonico, presentando un forte *stallo d'urto* per via di uno *shock* molto spostato verso monte che degrada l'efficienza aerodinamica (vedi figura 3.3).

I profili *supercritici* progettati specificamente per lavorare a $M_\infty > M_{Cr}^{inf}$, presentano caratteristiche geometriche molto peculiari; sono infatti dotati di raggi di curvatura al bordo d'attacco molto acuti in modo da accelerare il flusso immediatamente raggiungendo parecchio a monte la condizione sonica. La loro geometria è disegnata in modo che a progetto

la ricompressione sul dorso avvenga *isoentropicamente*, senza quindi incorrere nelle dissipazioni dovute alla formazione dell'onda d'urto; il dorso è particolarmente piatto per la ricompressione graduale, il ventre compensa in portanza assottigliandosi molto nella zona posteriore. Anche nel caso della formazione di un urto, questo dovrebbe essere di intensità minore, collocato più verso valle e con meno stallo aerodinamico rispetto a quello formatosi su un profilo subcritico (figura 3.3)



(a) *profilo subcritico* **(b)** *profilo supercritico*

Figura 3.3: Bolla supersonica in condizioni supercritiche per due tipi di profilo alare

In figura 3.4 si mostrano a confronto i momenti di beccheggio di un profilo standard NACA a 5 cifre – il NACA 23018 – e del profilo supercritico NASA SC(2)0414. I dati sono ottenuti con il software dedicato di aerodinamica 2D `xfoil`. Si nota come la geometria supercritica abbia lo svantaggio di generare momenti di beccheggio più grandi (in valore assoluto) costringendo all'equilibrio ad avere una superficie di coda più deportante e quindi complessivamente diminuendo la sostentazione dell'aereomobile a parità di superfici portanti.

Il caso di studio considerato farà riferimento a un problema *benchmark* transonico che ha trovato grandi applicazioni in campagne di validazioni di codici di calcolo CFD condotte dalla NASA [11]. L'interesse sta nel poter attingere a numerosi dati di confronto in letteratura sia ottenuti sperimentalmente [4] sia numericamente [9]. Il profilo utilizzato sarà quindi il supercritico RAE2822, la cui geometria (mostrata in dettaglio in figura 3.5) è stata ottenuta per punti dall'archivio *online* http://airfoiltools.com/.

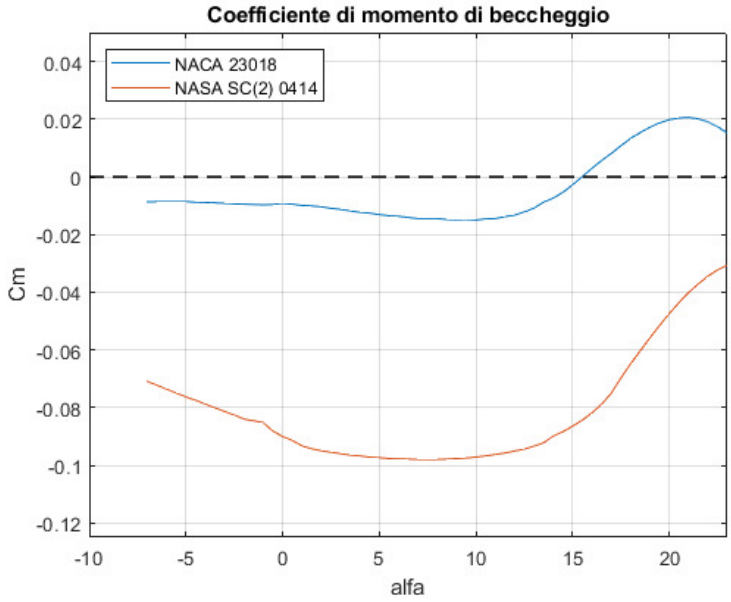


Figura 3.4: Momento di beccheggio $c_m(\alpha)$ per due tipi di profilo

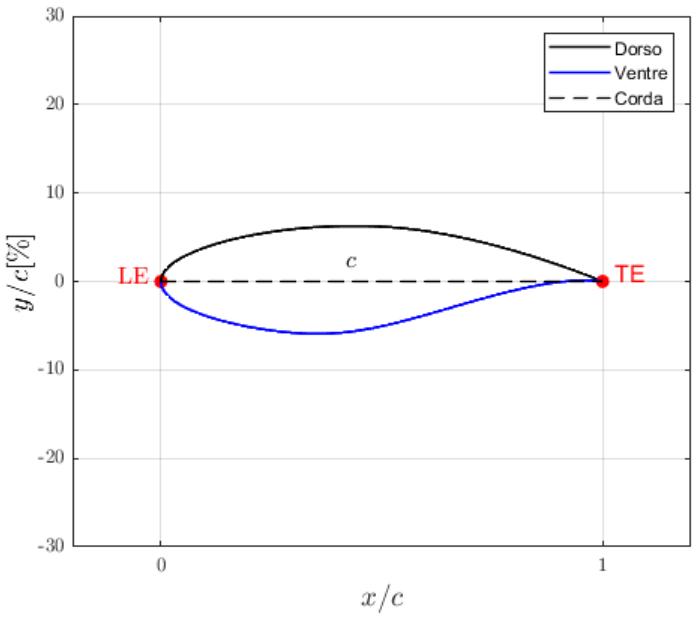


Figura 3.5: Geometria del profilo supercritico di riferimento RAE2822

3.1.2 Equazioni di governo

Nel caso più generale di un sistema continuo, in assenza di reazioni chimiche e apporto volumico di calore dall'esterno le leggi che governano la fluidodinamica sono le *equazioni di Navier-Stokes*, che si scrivono:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \\ \dfrac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{f} \\ \dfrac{\partial E}{\partial t} + \nabla \cdot [(E+p)\mathbf{v}] - \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{v}) + \nabla \cdot \mathbf{q} = \rho \mathbf{f} \cdot \mathbf{v} \end{cases} \tag{3.1}$$

Sebbene formalmente molto complesse, le (3.1) sono l'espressione per la particella fluida di tre principi fisici fondamentali e la loro derivazione, che non verrà descritta in questa sede, è piuttosto semplice. Nell'ordine:

1. la prima equazione prende anche il nome di *equazione di continuità* e rappresenta l'espressione della conservazione della massa all'interno del sistema

2. la seconda è un'equazione vettoriale e rappresenta il bilancio di quantità di moto, cioè la condizione di equilibrio delle forze del sistema; sostanzialmente è una riscrittura della seconda legge di Newton $\mathbf{F} = m\mathbf{a}$ in un caso in cui m non è assunta costante

3. la terza formula il primo principio della termodinamica per un fluido in movimento; rappresenta cioè la conservazione dell'energia totale del sistema $E = \rho(e + \frac{1}{2}|\mathbf{v}|^2)$

La forma delle Navier-Stokes presentata è quella differenziale conservativa (formulazione della divergenza) ed adotta una notazione compatta non del tutto convenzionale in termini matematici per questioni di brevità di formulazione.

Le equazioni di Navier-Stokes non hanno, nel caso più generale, soluzione analitica in forma chiusa [1]. In verità, si possono trovare soluzioni esatte solamente per delle semplificazioni talmente estreme da non trovare quasi alcuna applicazione pratica, pur conservando un grande interesse didascalico.

[1] Non sono, nella forma presentata, neppure un problema ben posto matematicamente. Il numero di variabili infatti eccede quello di equazioni a disposizione. Modelli di semplice chiusura del problema possono essere ottenuti tenendo conto di relazioni costituive e termodinamiche e facendo l'assunzione di gas perfetto; questo permette di trovare altre relazioni tra le variabili in gioco

La complessità di queste equazioni di governo è stata decisiva per lo sviluppo e l'affinamento di tecniche di simulazione numerica CFD sempre più accurate e sofisticate, oltre che del moderno approccio integrato numerico-sperimentale.

Le equazioni di governo che verranno utilizzate per la risoluzione numerica del flusso attorno al profilo alare sono le *equazioni di Eulero* (3.2), che si possono ottenere direttamente dalle (3.1) trascurando i termini di viscosità e conducibilità termica (cioè l'intera fisica diffusiva legata al trasporto molecolare). Per l'aerodinamica, inoltre, viene in genere trascurato il termine legato alla distribuzione volumica di forza per unità di massa $\mathbf{f}$ (*e.g.* l'accelerazione gravitazionale $\mathbf{g}$); viste le basse densità in gioco, infatti, potrebbe intervenire in maniera rilevante solo per un campo di moto a bassissima velocità che non rappresenta un caso di nostro interesse per questo studio.

Da tali considerazioni si ottiene dunque immediatamente:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \\ \dfrac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p \\ \dfrac{\partial E}{\partial t} + \nabla \cdot [(E+p)\mathbf{v}] = 0 \end{cases} \tag{3.2}$$

Il sistema (3.2) rappresenta le equazioni di Eulero nella forma più generale, per un fluido compressibile e instazionario. Conviene infine farne un'ultima riscrittura per presentare il modello matematico effettivamente utilizzato nell'implementazione del codice a disposizione, che come per la maggior parte dei codici CFD discretizza le equazioni di governo con un approccio ai *volumi finiti* basato quindi su una formulazione integrale invece che differenziale.

Si precisa che, anche se le formulazioni matematiche delle equazioni di governo (sia per Eulero che per Navier-Stokes) sono tutte equivalenti e intercambiabili con delle semplici identità vettoriali e l'applicazione del teorema della divergenza di Gauss, la forma integrale può considerarsi quella fondamentale perché permette di trattare casi più generali con eventuali discontinuità nel campo di moto. La forma differenziale, invece, assume che le variabili in gioco siano derivabili nel dominio di calcolo in cui sono definite. Le stesse considerazioni erano state fatte per l'equazione modello del filo elastico, nel Capitolo 2.

Nel nostro caso un approccio integrale è quanto mai necessario, visto che il flusso transonico su un profilo supercritico in condizioni diverse da quelle per cui il profilo è stato progettato porta alla formazione di un'onda d'urto che introduce delle forti discontinuità nella soluzione ottenuta. Al netto di

queste considerazioni, dunque, le equazioni che governano il caso di studio in esame saranno le seguenti:

$$\begin{cases} \dfrac{\partial}{\partial t}\displaystyle\int_V \rho \, dV + \int_S \rho \, \mathbf{v} \cdot \mathbf{n} \, dS = 0 \\ \dfrac{\partial}{\partial t}\displaystyle\int_V \rho \mathbf{v} \, dV + \int_S \rho \mathbf{v} \, \mathbf{v} \cdot \mathbf{n} \, dS = -\int_S p \, \mathbf{I} \cdot \mathbf{n} \, dS \\ \dfrac{\partial}{\partial t}\displaystyle\int_V E \, dV + \int_S (E + p) \, \mathbf{v} \cdot \mathbf{n} \, dS = 0 \end{cases} \tag{3.3}$$

che derivano direttamente dalle (3.2) tramite l'integrazione su un generico volume di controllo V (che ha per bordo una superficie S) e l'applicazione del già citato teorema di Gauss per trasformare gli integrali tripli di divergenza in integrali di flusso.

Adottare un modello matematico basato sulle equazioni (3.3) comporta di certo un compromesso tra semplicità computazionale e livello di dettaglio della descrizione del campo di moto. Di seguito è riportata una breve discussione sulle motivazioni, i limiti e le implicazioni di tale scelta.

- L'aver trascurato la parte diffusiva delle (3.1) implica la rinuncia a una trattazione dello *strato limite* che si forma su una qualunque superficie fisica quando lambita da una corrente fluida. In realtà, il modello di Eulero è "esatto" per il *campo esterno* perché gli effetti diffusivi risultano apprezzabili solo in una zona di spessore δ molto vicino alla parete; in tutto il resto del campo di moto, la parte convettiva dell'equazione è fortemente preponderante. Anche gli sforzi sia teorici – come ad esempio la soluzione esatta dello strato limite su una lamina piana dovuta a Blasius [2] – che numerici vanno in questa direzione: si calcola una soluzione di strato limite riadattando le (3.1) per una coordinata normale alla parete nelle sue immediate vicinanze e poi si cerca di raccordare l'andamento così ottenuto con il campo di moto esterno.

- Per quanto molto limitata nelle dimensioni, la regione dello strato limite ha un'enorme importanza fisica tenendo conto di tutti gli effetti dissipativi dovuti alla perdita di quantità di moto della corrente per la condizione di aderenza a parete. Senza un modello di strato limite non è possibile analizzare ad esempio il coefficiente di resistenza completo del profilo, che per un corpo affusolato *deve* tenere conto degli effetti di sforzo d'attrito per effetto del gradiente di velocità a parete. Una trattazione di questo tipo basata solo sulle (3.3) non avrebbe senso.

- Nel nostro caso, però, la variabile di interesse per lo studio e la ricostruzione delle soluzioni è – come si vedrà in seguito – la distribuzione del

coefficiente di pressione c_p sul profilo, definito come normalizzazione del campo di pressioni p

$$c_p = \frac{p}{\frac{1}{2}\rho U_\infty^2} \tag{3·4}$$

Richiamando il modello di equazioni dello strato di limite di Prandtl (lo stesso usato da Blasius per la già citata soluzione esatta), si vede che per l'equazione di bilancio della quantità di moto lungo la direzione normale alla parete l'analisi degli ordini di grandezza dei termini normalizzati porta ad ottenere (nel caso bidimensionale)

$$\frac{\partial p}{\partial y} \simeq 0 \Rightarrow \quad p = p(x) \tag{3·5}$$

il che significa che l'informazione sulla pressione viene trasportata ed "entra" nello strato direttamente dal flusso esterno, e che quindi con buona approssimazione si possono considerare esatti i valori di p ricavati dal modello di Eulero che considera solo una condizione di tangenza del flusso a parete.

- Per una risoluzione più precisa della turbolenza del campo di moto che si genera dietro l'urto bisognerebbe utilizzare un modello come le RANS che tenga conto delle fluttuazioni accoppiandolo con uno schema di chiusura che modella la turbolenza (per il calcolo del *tensore degli sforzi di Reynolds*). L'interesse è invece concentrato su una valutazione *media* del campo di moto e si rinuncia a una ricostruzione dell'informazione istantanea [2]. Infine, si considererà una soluzione *stazionaria*, cioè l'effetto costante nel tempo di un campo di moto completamente instaurato che trascura le derivate $\partial d/\partial t$ nelle (3·3).

3.1.3 Parametri di studio

1 Scelta dei parametri

I parametri affetti da incertezza che costituiscono lo spazio I_Z bidimensionale sono:

1. il numero di Mach della corrente indisturbata a monte M_∞

[2] La distribuzione del coefficiente di pressione viene comunque in genere calcolata al fine di essere integrata per ottenere la portanza sviluppata dal profilo che è dunque un'informazione globale e mediata nel tempo

2. l'angolo di incidenza *geometrico* della corrente rispetto alla direzione orizzontale x (nel codice AOA – *angle of attack*)

Come si è discusso in precedenza, queste particolari variabili influenzano fortemente la condizione critica del flusso e dunque la fenomenologia che ne deriva; in questo tipo di campo di moto, in altre parole, la sensibilità variazionale delle soluzioni ai due parametri considerati è molto elevata. Un altro importante elemento caratterizzante di un flusso aerodinamico come quello studiato sarebbe il numero di Reynolds di monte Re_∞

$$\mathrm{Re}_\infty = \frac{U_\infty c}{\nu}$$

che è una rappresentazione molto efficace del rapporto di importanza tra le forze di inerzia e quelle viscose, e ha dunque influenza sugli effetti diffusivi e sulla transizione turbolenta. Tuttavia, utilizzare le equazioni di governo di Eulero significa assumere che l'inerzia (parte convettiva delle (3.1)) sia preponderante sugli effetti viscosi, che vengono come detto trascurati; in termini matematici questo significa considerare un $\mathrm{Re} \to \infty$ o, più praticamente, il parametro Re_∞ non gioca nessun ruolo nelle simulazioni ed è perciò lasciato invariato.

I parametri considerati sono di tipo *operazionale*, agiscono cioè sulle condizioni fisiche del campo di moto, ad esempio per un possibile studio preliminare della perturbazione delle condizioni a progetto di un velivolo che monta questo profilo. L'approccio *Multi-Fidelity* è in realtà molto utile anche per problemi di ottimizzazione, come segnalato nell'Introduzione a questo lavoro. Sarebbe di interesse la possibilità di effettuare uno studio parametrico sulla geometria del profilo che permettesse di considerare le variazioni della risposta ottenuta a perturbazioni di carattere geometrico.

Le geometrie sofisticate dei profili supercritici vengono infatti spesso ottenute con la risoluzione numerica *inversa*, vale a dire prescrivendo un andamento desiderato del campo di pressioni e ricavando la superficie che meglio si adatta ai requisiti proposti. Inoltre, con delle perturbazioni di lievissima entità e distribuite in modo aleatorio, si potrebbe tener conto di eventuali difetti costruttivi (tolleranze) o della deformazione della geometria in volo per effetto dei carichi subiti.

L'introduzione di parametri di tipo geometrico accanto a quelli operazionali, però, sarebbe stata fonte di alcune considerevoli complicazioni:

(a) la geometria del profilo ottenuta dall'archivio http://airfoiltools.com/ è definita per punti; non si conosce cioè, come per il caso semplice dei NACA, un andamento funzionale (eventualmente dipendente dai

parametri incerti desiderati) che leghi l'andamento delle curve di dorso e ventre alla coordinata x/c. La parametrizzazione PARSEC [12] – utilizzata in uno dei *test case* dello studio [6] che fa da riferimento per questo lavoro e in molti altri studi di ottimizzazione – fornisce un metodo molto efficiente di descrizione del profilo tramite le variabili geometriche mostrate in figura 3.6

(b) il codice CFD `su2` utilizzato non è dotato di un proprio *mesher* interno con la possibilità di riadattare la griglia di calcolo per un'assegnata geometria in *input*. Questo significa che per ogni istanza *random* di un assegnato parametro geometrico si sarebbe dovuta costruire una *mesh* adattiva diversa o, più ragionevolmente, implementare un codice *ad hoc* che potesse farlo in modo programmatico.

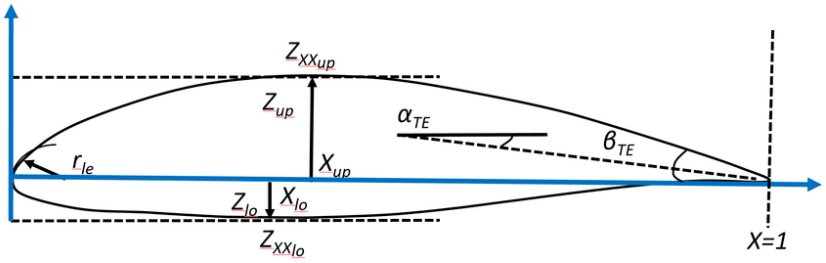


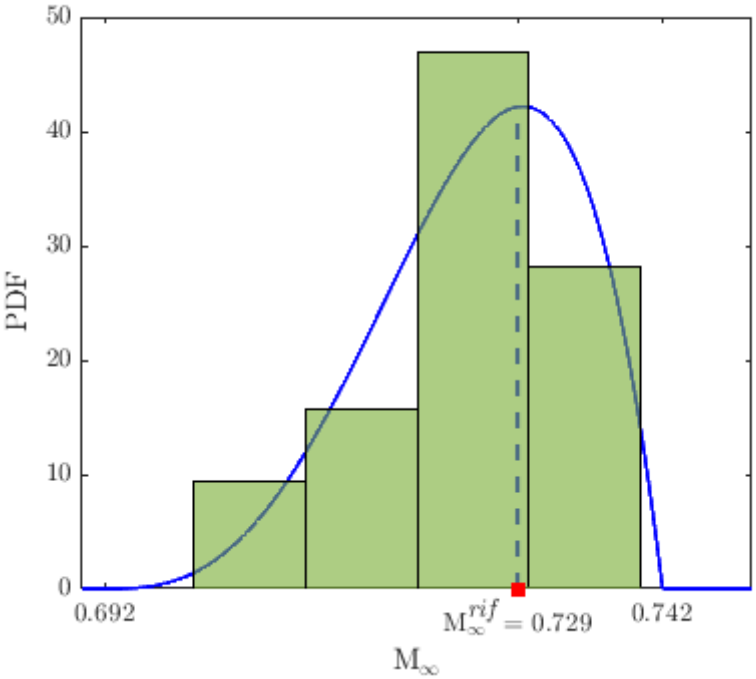
Figura 3.6: Parametri della discretizzazione PARSEC su un profilo alare

Tenendo come utile confronto in letteratura l'*experimental case 6* dell'archivio di validazione NASA [13], le condizioni di riferimento da perturbare per lo studio effettuato sono riportate in tabella 3.1, insieme alla distribuzione dei vari insiemi campionati per l'algoritmo *Bi-fidelity*.

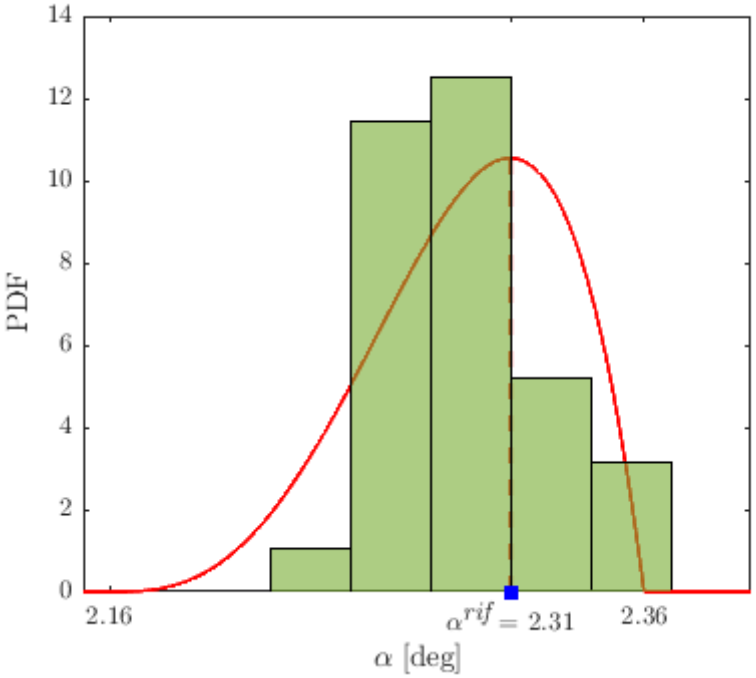
2 Campionamento ed incertezze

Si è scelto un campionamento dell'insieme I_Z con delle distribuzioni di probabilità che possano essere rappresentative di una condizione di volo in cui una raffica introduca delle piccole perturbazioni sui valori di riferimento per M_∞ e α, che fanno da centro alle rispettive distribuzioni.

Come nei *test case* del Capitolo 2 si è dunque proceduto a ritroso definendo prima i *set* discreti con ragioni di tipo "fisico", e poi considerando $I_Z \in \mathbb{R}^2$ come il più piccolo insieme continuo del tipo $[I_\mathrm{M}] \times [I_\alpha]$ che li contiene. Una volta così ricavato il dominio parametrico, si è potuto procedere anche alla creazione dello spazio $\{\tilde{z}_k\}$ per la stima degli errori di ricostruzione.



(a) *Distribuzione dei campioni* M



(b) *Distribuzione dei campioni* α

Figura 3.7: Campionamento indipendente dei due parametri di I_Z – PDF = B

Tabella 3.1: Valori di riferimento e incertezze dei parametri per le simulazioni

Parametro	NASA Case 6	Tipo di incertezza	
		Γ	$\{\tilde{z}_k\}$
Re_∞	$6{,}5 \times 10^6$	–	–
M_∞	0,729	B(32, 4, 2, 0,692, 0,05)	$\mathcal{U}$(0,692, 0,742)
α	2,31°	B(32, 4, 2, 2,16°, 0,2°)	$\mathcal{U}$(2,16°, 2,36°)

Per l'ottenimento dello spazio Γ è stata utilizzata per entrambi i parametri operazionali una distribuzione di tipo Beta $\mathrm{B}(a, b)$ ben nota in statistica, di cui si introduce di seguito la funzione di densità di probabilità:

$$\mathrm{B}(a, b, x) = \frac{x^{a-1}\,(1-x)^{b-1}}{\displaystyle\int_0^1 x^{a-1}\,(1-x)^{b-1}\,dx} \tag{3.6}$$

che è parametrizzata dai valori di a e b e definita sull'intervallo $[0, 1]$.

Per poter ottenere un insieme campionato distribuito nel modo descritto dalla (3.6) si è creata una semplice *function* `randbeta` che utilizza il metodo dell'inversione della funzione CFD (calcolata per integrazione numerica a partire dalla PDF) su un vettore di punti a distribuzione uniforme. Lo *script* emula il funzionamento delle funzioni standard MATLAB `rand` e `randn` producendo in *output* un vettore delle dimensioni richieste contenente punti distribuiti secondo la (3.6) ma su un intervallo eventualmente riscalato.

La notazione usata in tabella 3.1 si riferisce proprio a questo:
$\mathrm{B}(M, a, b, \mathit{inf}, L)$ descrive M punti campionati con incertezza data dalla distribuzione $\mathrm{B}(a, b)$ sull'intervallo che ha come estremo inferiore *inf* e come ampiezza L.

In figura 3.7 si mostrano le distribuzioni degli intervalli di valori campionati di M e α che formano l'insieme Γ. Si noti che le due discretizzazioni avvengono in maniera totalmente *indipendente*: Γ viene cioè creato a partire dai due intervalli tramite una tensorizzazione. Per ogni valore di M campionato si sono effettuate le simulazioni LF corrispondenti ad ogni valore campionato di α a disposizione, e così via fino ad ottenere quindi un *set* di coppie di valori di dimensioni $M = M_{\mathrm{M}_\infty} \times M_\alpha = 1024$. La distribuzione discreta dei punti ottenuti fatica a seguire accuratamente quella teorica per lo scarso numero di punti su ognuno dei due intervalli indipendenti; la scelta su M_{M_∞} e M_α è stata effettuata per ottenere un valore di M dello stesso ordine di grandezza di quello dei *test case* del Capitolo 2 una volta effettuata la tensorizzazione.

Per i punti in $\{\tilde{z}_k\}$ si è invece adottata la solita distribuzione uniforme $\mathcal{U}$ all'interno dei due intervalli per M_∞ e α definiti da Γ. Le coppie sono stavolta

selezionate definendo a priori la dimensione dell'intero insieme ($N = 20$) e scegliendo contemporaneamente due valori che vadano a formare il k-esimo punto $\tilde{z}_k$, per $1 \leq k \leq N$.

Si noti che, come anticipato, ci si è dovuti limitare ad una cardinalità piuttosto esigua su cui calcolare gli errori, visto che il metodo di stima dell'errore di ricostruzione ha come presupposto la costosa valutazione dell'intero insieme $u^H(\{\tilde{z}_k\})$. In questo caso si è scelto dunque un $N \sim m$.

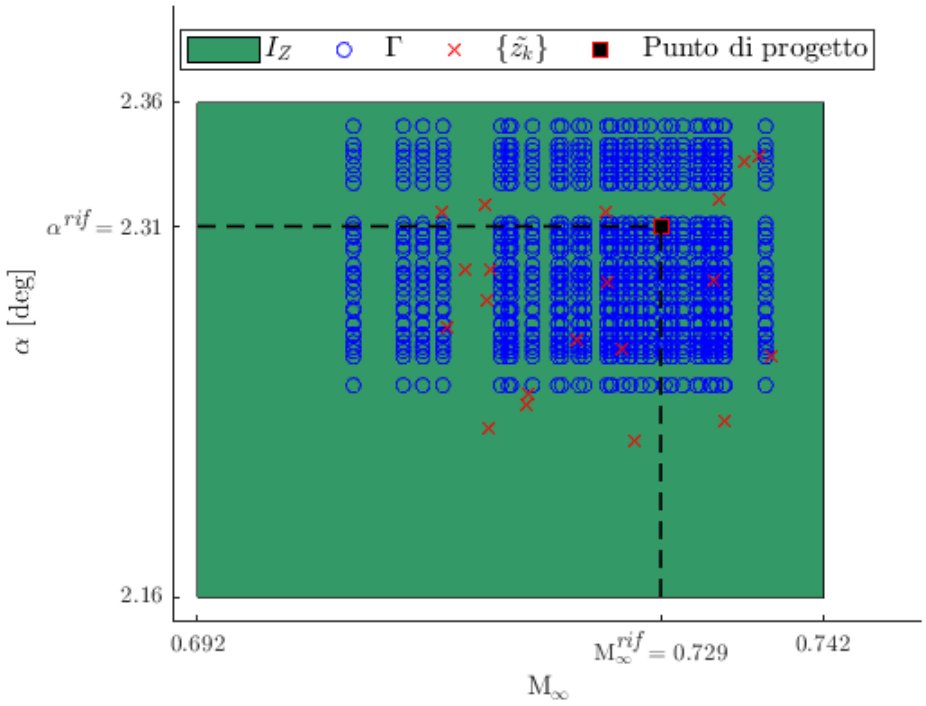


Figura 3.8: Collocazione degli insiemi discreti sullo spazio parametrico

La figura 3.8 confronta la collocazione sullo spazio dei parametri dei due insiemi discreti costruiti. La rappresentazione ottenuta di I_Z potrebbe essere troppo poco densa e/o uniforme per cogliere la variazione delle soluzioni sull'intero insieme. Si noti anche l'evidente differenza di costruzione dei due insiemi:

- Γ è una tensorizzazione di due variabili indipendenti e appare perciò "ordinatamente" sotto forma di una griglia rettangolare
- $\{\tilde{z}_k\}$ è selezionato prendendo simultaneamente delle coppie di punti *random* ed è infatti sparso all'interno dell'insieme di definizione

3.2 Simulazioni effettuate

3.2.1 Metodo numerico

Tutti i parametri che definiscono il metodo numerico adottato sono passati al *solver* CFD di `su2` tramite un *file* di testo che per mette di impostare con buona flessibilità le specifiche della simulazione desiderata: il *configuration file* `.cfg`. Di seguito si elencano e descrivono brevemente le principali scelte effettuate, omettendo ovviamente quelle per i valori di M_∞, α e Re appena discusse.

Solver. Chiave = `EULER`
Come già discusso nella sezione precedente, si sceglie di adottare il modello matematico di Eulero come equazioni di governo del problema.

Condizioni infinito a monte. Il flusso indisturbato "lontano" dal profilo è caratterizzato dalle sue proprietà termodinamiche. Per chiudere il sistema bisogna impostare il valore di almeno un'altra variabile: alla pressione di riferimento p_∞ viene assegnato il valore di *default* di 101 325 Pa. A questo punto, avendo fissato il numero di Mach, con la definizione di velocità del suono e l'assunzione di gas perfetto (la relativa costante universale è modificale nel *file* `.cfg`) il sistema *free stream* non ha più gradi di libertà ed è completamente definito.

Condizioni al contorno. Come si mostrerà a breve, la geometria del dominio di calcolo utilizzato prevede l'utilizzo di due sole superfici: la parete del profilo alare e il bordo esterno del dominio; su di esse vanno assegnate delle condizioni al contorno per definire il problema. Si è scelto

- Per il profilo alare: Chiave = `MARKER_EULER`
  la condizione di tangenza del flusso a parete tipica delle equazioni di eulero: $\mathbf{v} \cdot \mathbf{n} = 0$
- Per il bordo esterno: Chiave = `MARKER_FAR`
  Sul confine del dominio di calcolo è assunta la condizione di corrente indisturbata nelle condizioni impostate in precedenza.

Schema convettivo. Chiave = `JST`
Schema per la discretizzazione della parte convettiva dell'equazione di governo (nel nostro caso l'unica presente). La scelta ricade sullo Jameson-Schmidt-Turkel(JST), di tipo centrato e del secondo ordine; questo schema introduce

della diffusività artificiale tramite un termine sorgente numerico al fine di stabilizzare la soluzione.

Schema temporale. Chiave = `EULER_IMPLICIT`
Discretizzazione temporale con lo schema implicito di Eulero intrinsecamente stabile. In realtà, come anticipato, le simulazioni effettuate sono stazionarie, quindi in realtà quello discretizzato è un *pseudo-tempo* che serve solo a permettere al programma di avanzare alla successiva iterazione fino al raggiungimento della condizione di stazionarietà cercata.

Solutore lineare. Chiave = `FGMRES`
Metodo iterativo per l'inversione del sistema di equazioni ad ogni *step* temporale. Si sceglie il *Flexible Generalized Minimum Residual* (FGMRES) che è l'opzione di *default* del codice per le simulazioni CFD; si seleziona anche un pre-condizionatore del sistema di tipo `ILU`. Per un problema come questo, il *linear solver* dovrebbe andare a convergenza dopo una decina di iterazioni: il numero massimo è comunque impostato cautelativamente a 50.

Criterio di convergenza. Chiave = `RMS_DENSITY`
La simulazione si arresta quando il valore dei residui di una variabile conservativa (che sono un indice di quanto il campo numerico costruito sia vicino a soddisfare le equazioni di governo) scende sotto una soglia prefissata; la variabile considerata è la deviazione standard di ρ e il valore soglia (Chiave = `CONV_RESIDUAL_MINVAL`) è impostato a $10^{-7,5}$. Sulla base di alcuni *test* effettuati, si è scelto anche un numero massimo di iterazioni entro il quale la soluzione deve andare a convergenza secondo il primo criterio o essere forzatamente arrestata (Chiave = `ITER`), pari a 300.

3.2.2 Discretizzazione del dominio di calcolo

Anche per le ricostruzioni sul profilo, la strategia *Multi-Fidelity* è stata applicata considerando dei modelli LF e HF che risolvano numericamente le stesse equazioni di governo (quelle di Eulero, appunto) ma su griglie di calcolo dalla qualità molto diversa. Di seguito sono presentate e visualizzate le *mesh* costruite per i vari modelli considerati.

La geometria del dominio di calcolo è comune in tutti i casi. Si sono definite le due curve sulle quali sono state imposte le condizioni al contorno, di seguito descritte:

(a) Il RAE2822 è importato nel software dedicato per punti. Su queste coordinate sono state costruite e raccordate diverse curve per ottenere la geometria completa che rappresenta la parete del profilo alare:

- una *spline* per i punti sulla zona centrale del dorso
- una *spline* per i punti sulla zona centrale del ventre
- due *spline* per il bordo di fuga, unite da un segmento che raccorda i due punti estremi di coordinata $x/c = 1$ (il profilo presenta un TE troncato)
- due *spline* per il bordo attacco (per definire meglio la forte curvatura presente) unite da un segmento che chiude la geometria raccordando l'ultimo punto a disposizione con il primo

(b) Il confine del dominio di calcolo è definito da una circonferenza di raggio R (il cui valore può essere cambiato all'interno del *file* `.geo` parametrizzando la *mesh*) che costruisce dunque una cosiddetta *O-Grid*, tipica per le applicazioni 2D di Eulero.

Successivamente si è assegnato il campo di moto 2D alla superficie piana compresa tra le due curve appena introdotte.

Ancora una volta, a parità di modello *High-Fidelity*, sono stati utilizzati diversi modelli LF (mostrati in figura 3.9):

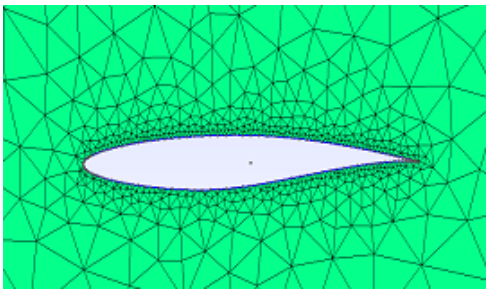
Modello tipo A (figura 3.9a) il meno accurato a disposizione, conta solo 1989 elementi.

Modello tipo B (figura 3.9b) ha la stessa distribuzione di nodi (e dunque lo stesso numero di elementi) sulla superficie del profilo del modello A, ma presenta uno *stretching* della griglia verso l'esterno meno accentuato, risolvendo meglio la zona attorno al profilo; è formato da 3081 elementi.

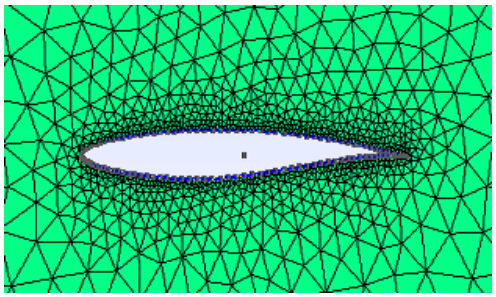
Modello tipo C (figura 3.9c) una griglia LF decisamente più fine con 7330 elementi

La *mesh* per il modello HF è mostrata in figura 3.10 – presenta 38033 elementi e come si vede chiaramente risolve la superficie del profilo con una densità piuttosto elevata, se comparata con quelle dei modelli *Low-Fidelity*. In figura 3.11 si mostra invece in dettaglio la risoluzione al bordo d'attacco, per questioni di chiarezza di visualizzazione.

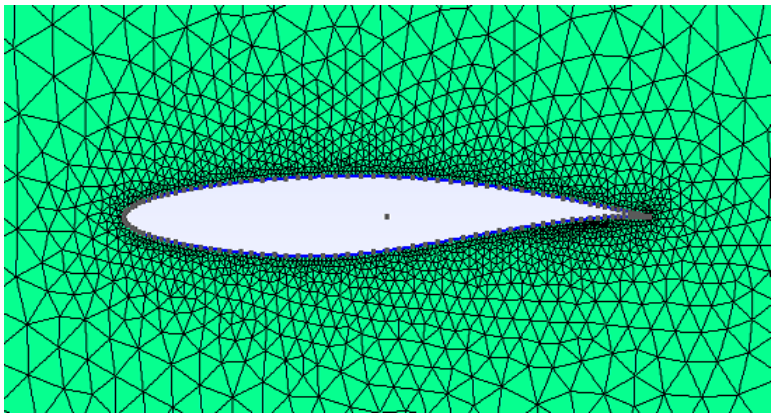
Si tengano presenti le seguenti annotazioni sulle *mesh* presentate in questa sezione:



(a) *LF A Model*



(b) *LF B Model*



(c) *LF C Model*

Figura 3.9: *Mesh* LF sulla superficie del profilo

- Il raggio R delle griglie considerate ha un valore pari a 20 volte la corda del profilo alare (che fa da riferimento per tutte la metrica adimensionale della geometria). Questo valore, scelto per ragioni di realizzabilità delle simulazioni, risulta verosimilmente troppo basso per assicurare perfettamente la condizione di corrente indisturbata ai confini del dominio. Infatti è probabile che agendo in questo modo si sia creato un effetto di bloccaggio numerico, simile concettualmente all'errore sperimentale introdotto in galleria del vento quando l'ingombro del modello è eccessivo rispetto alle dimensioni della galleria. Si ricorda che infatti, per un caso di studio tra l'alto subsonico e il transonico, gli effetti di valle hanno influenza sui punti a monte (cioè la corrente si "accorge" da lontano della perturbazione indotta dalla presenza del profilo) e quindi per riprodurre le condizioni di corrente indisturbata è necessario ampliare di molto il dominio considerato.

- Tutte le griglie presentate per la discretizzazione ai volumi finiti sono

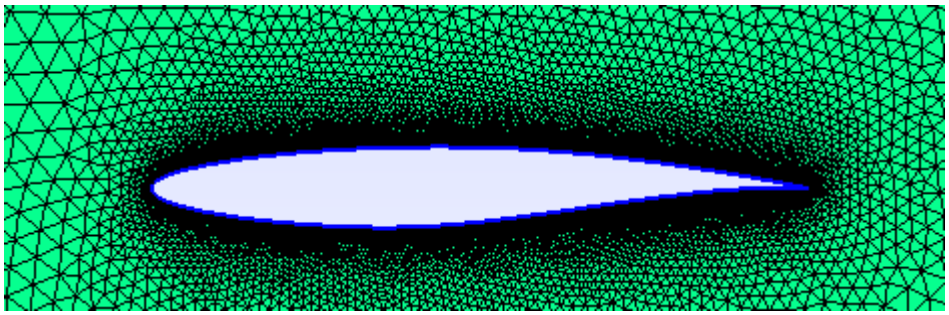


Figura 3.10: *Mesh* HF sulla superficie del profilo

di tipo *non-strutturato* [3] e formate da elementi triangolari. Questi ultimi non sarebbero in realtà ideali per la CFD, ma sono molto rapidi da costruire tramite semplici algoritmi di triangolazione implementati nativamente in `gmsh`.

- La previsione sul campo di moto descritta nella sezione 3.1.1 ci porta alla quasi certezza a priori della presenza di un'onda d'urto più o meno intensa sulla superficie del profilo per la maggior parte delle configurazioni in Γ. Si è visto come questo fenomeno, fortemente dissipativo, porti a delle discontinuità sulla soluzione e in particolare a un repentina compressione a cavallo della superficie dello *shock*. È dunque lecito attendersi che attorno alla zona dell'urto le *mesh* possano non essere abbastanza ricche di elementi da risolvere le discontinuità; si è preferita per il momento la costruzione di griglie (compresa la HF) abbastanza omogenee per privilegiare la semplicità di costruzione e la robustezza, visto che la stessa griglia viene utilizzata per ciascuna delle configurazioni e a priori non è possibile conoscere la collocazione dell'urto che di certo varierà *sensibilmente* al variare dei parametri operazionali considerati.

3.3 Procedura e strategia *Multi-Fidelity*

Alla stregua di quanto realizzato e descritto per i *test* di validazione, una volta procuratici i due modelli per la strategia *Bi-Fidelity* si procede alla costruzione delle matrici che costituiranno da *database* numerico per la successiva applicazione dell'algoritmo. Questa volta però, avendo *effettivamente* la possibilità di valutare solo un numero piuttosto esiguo di soluzioni

[3] La posizione di una cella non è cioè univocamente determinata da quelle delle celle contigue tramite una precisa indicizzazione, come avviene al contrario per le *mesh strutturate*

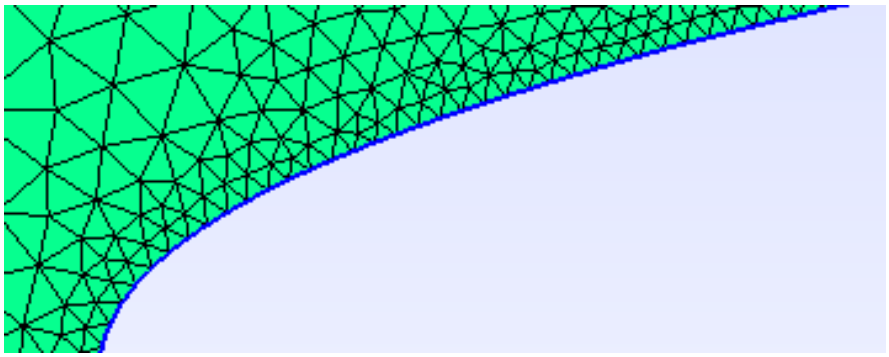


Figura 3.11: *Mesh* HF – dettaglio

High-Fidelity, si procede esattamente con il modus operandi descritto nel Capitolo 1; vale a dire, nel dettaglio:

(1) Si lanciano tramite un apposito *script* `python` in maniera seriale le M simulazioni *Low-Fidelity* predisposte, ciascuna targata da una coppia di valori $(M_\infty, \alpha) \in \Gamma$. Il codice `su2` crea in *output* due tipi di *file* diversi:

- il *volume file* in formato `.vtu` contenente i valori delle variabili fisiche su tutto il dominio di calcolo, compatibile con molti *software* di visualizzazione e dunque utile per il *post-processing* delle soluzioni.
- il *surface file* in formato `.csv` contenente la soluzione solo per gli elementi sulla superficie del profilo; si tratta quindi dei valori delle variabili a parete.

I *volume file* sono parecchio ingombranti e non strettamente necessari ai fini dell'analisi in corso, quindi vengono eliminati appena prima del lancio della simulazione successiva. Si è in questo modo costruito e allocato in memoria un archivio di M soluzioni del flusso a parete.

(2) Si importano nel codice MATLAB sotto forma di tabella i *surface file* appena salvati e si eseguono su di essi le seguenti operazioni:

- per prima cosa si effettua un riordinamento dei punti di calcolo (i centri cella dei volumi finiti costruiti) che vengono restituiti da `su2` con un ordinamento interno al codice dipendente dal modo particolare in cui viene letto il file di *mesh* passato in *input*. Si ottiene così una nuova tabella in cui ciascuna variabile della soluzione è ordinata sul profilo alare in senso orario a partire dal bordo d'attacco:

$$\text{LE} \rightarrow \text{Dorso} \rightarrow \text{TE} \rightarrow \text{Ventre}$$

Tabella 3.2: Numero di celle nei modelli costruiti

Modello	Numero di celle	
	Volume	Superficie
LF Tipo A	1989	173
LF Tipo B	3081	173
LF Tipo C	7330	258
HF	38033	1021

- si estrapola per ciascuna tabella di soluzione il vettore contenente i valori del coefficiente di pressione c_p, la variabile scelta per la ricostruzione; Si salva anche il vettore delle coordinate riordinate $(x/c, y/c)$, *i.e.* l'ascissa curvilinea dei punti sul profilo alare.

(3) Sistemando per colonna i vettori con i valori di c_p disposti in senso orario sul profilo, si costruisce (per ciascun modello *Low-Fidelity*) la matrice $\mathbf{V}^{\mathrm{LF}} \in \mathbb{R}^{N_L \times M}$, in cui N_L non va confuso con il numero di elementi indicato per ciascun modello nel paragrafo 3.2.2; infatti quello rappresentava l'intero numero di celle presenti nel dominio di calcolo, questo è solo il numero di punti discreti presenti sulla superficie del profilo. Si veda la tabella riassuntiva 3.2 per chiarezza.

(4) Si lancia l'Algoritmo 1 guidato dalla matrice $\mathbf{V}^{\mathrm{LF}}$ con un numero massimo di nodi interpolanti da selezionare fissato a $m = 20$; in questo modo si ottengono le matrici $\gamma \in \mathbb{R}^{m \times 2}$ contenenti i *pivot* selezionati per il calcolo delle soluzioni HF.

(5) Avendo a disposizione le m coppie di valori $(\mathrm{M}_\infty, \alpha)$ in γ, si ripetono per la griglia *High-Fidelity* gli *step* (1) e (2) di questa procedura. A questo punto si può costruire la matrice $\mathbf{V}^{\mathrm{HF}} \in \mathbb{R}^{N_H \times m}$ delle soluzioni $u^H(\gamma)$ e allocarla in memoria.

(6) Si ripetono gli *step* (1) e (2) anche per gli N punti dell'insieme $\{\tilde{z}_k\}$ *sia* per i modelli LF *sia* per quelli HF. Si ottengono così le matrici $\tilde{\mathbf{V}}^{\mathrm{LF}} \in \mathbb{R}^{N_L \times N}$ e $\tilde{\mathbf{V}}^{\mathrm{HF}} \in \mathbb{R}^{N_H \times N}$.

(7) A questo punto il *database* per l'ottenimento delle ricostruzioni e la stima degli errori fatti su di esse è completamente allocato in memoria. Si può procedere all'applicazione dell'Algoritmo 2, da cui si ottengono finalmente le soluzioni $\hat{u}^H(\{\tilde{z}_k\})$ nelle N configurazioni *random* richieste.

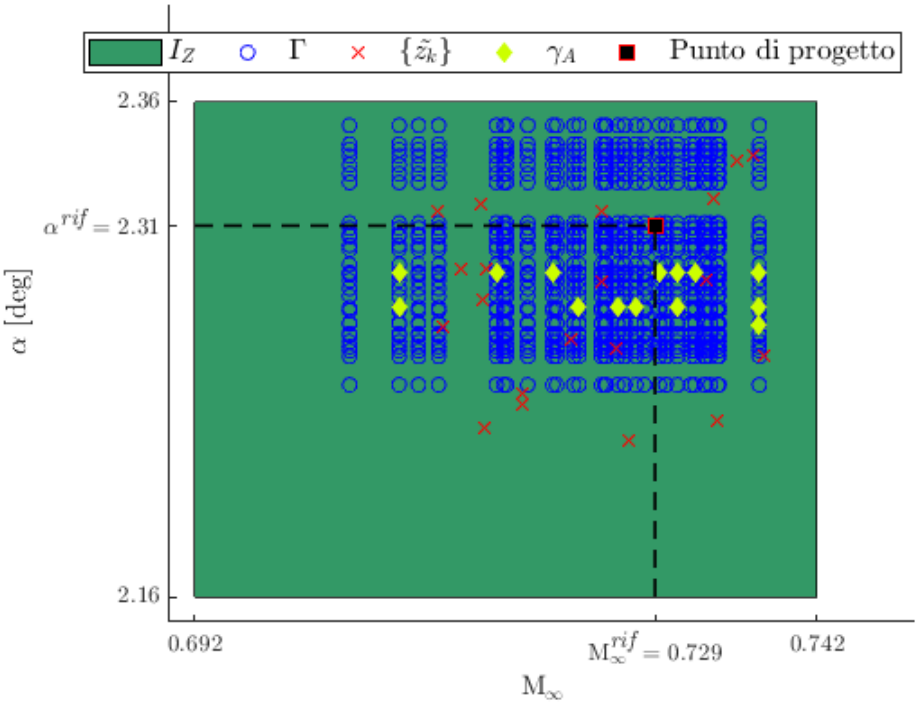


Figura 3.12: Insiemi discreti su I_Z – nodi interpolanti

Per il calcolo dell'errore si è proceduto come sempre, stimando un errore massimo e uno medio sullo spazio V^H, a partire dalle (1.25). In particolare, la (1.25b) pone il solito problema di approssimazione dell'integrale su un insieme fisico D [4]. Per il caso del profilo, la definizione di norma si declina come calcolo di un integrale curvilineo su un campo scalare (*i.e.* di prima specie).

L'approssimazione di questo integrale è implementata nella *function* `NormL2Airfoil` dell'omonimo *script*. Il metodo utilizzato è una segmentazione della curva definita per punti e il calcolo dell'integrale di linea su ciascun segmento tramite un'approssimazione del primo ordine. In formule:

$$\int_C f ds = \int_a^b f(\mathbf{r}(t))\|\mathbf{r}'(t)\|\, dt \simeq \sum_{i=1}^{N_H} \left(\frac{f_i + f_{i+i}}{2} \|\mathbf{s}_i'\| \right)$$

in cui $\mathbf{r}(t)$ è una parametrizzazione analitica della curva C (assolutamente non banale per il profilo alare supercritico) e $\|\mathbf{s}_i'\|$ è la lunghezza dell'i-esimo elemento della segmentazione che approssima la curva.

[4] L'insieme D non è da confondere in questo caso con l'intero dominio fisico di calcolo considerato, con cui invece coincideva nei *test case*: visto che il vettore ricostruito dei c_p è definito solo a parete, il dominio sarà appunto costituito dalla superficie del profilo

In figura 3.12 si mostra la collocazione dei nodi *pivot* selezionati dall'algoritmo guidato dalle soluzioni LF di tipo A (ma una situazione analoga occorre per gli altri casi). Si noti come i valori "importanti" di α siano davvero pochi, disposti in una banda orizzontale dall'ampiezza ridotta. Il campionamento utilizzato per α nella costruzione di Γ è probabilmente fin troppo fine, e le informazioni rilevanti si concentrano in una manciata di valori perdendo qualche informazione sulla sensibilità globale alla variabile nelle altre zone dello spazio parametrico.

3.4 Analisi dei risultati ottenuti

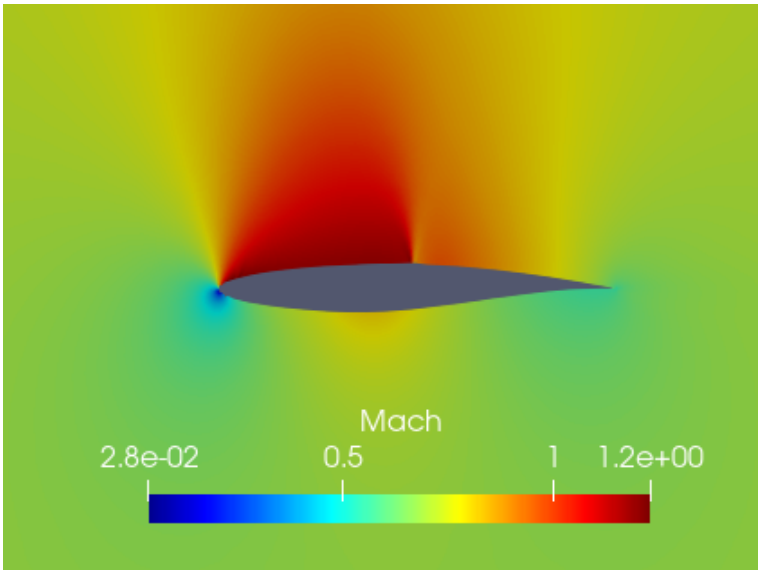
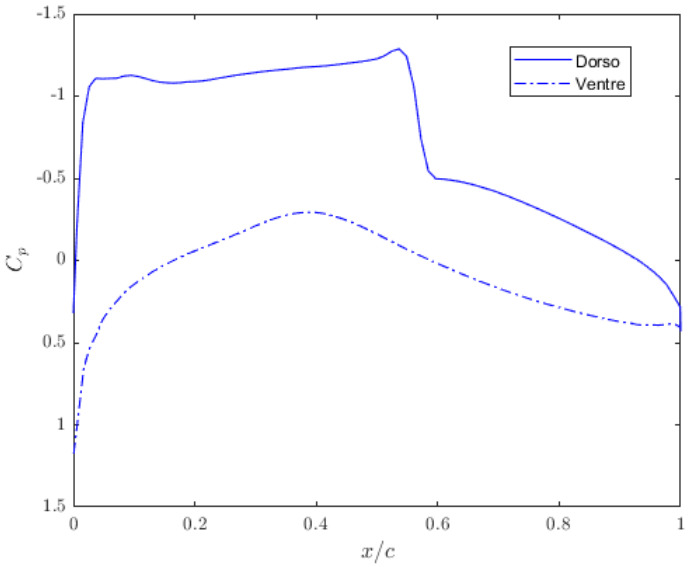


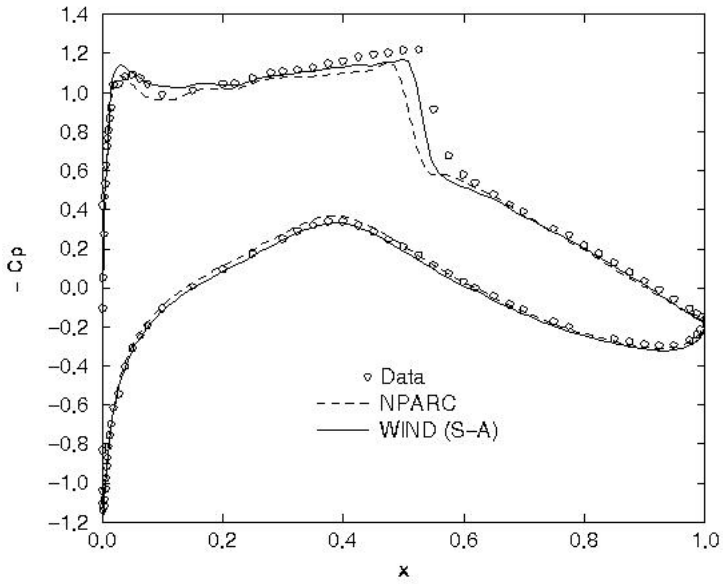
Figura 3.13: Simulazione di riferimento – campo di Mach

Le figure 3.13 e 3.14 visualizzano le soluzioni ottenute per i valori di riferimento (tabella 3.1) per il confronto con il *Test Case* 6 della NASA [11] e la validazione dei risultati. In particolare:

(a) In figura 3.13 è mostrata la *colormap* per il campo di Mach nei pressi del profilo, in modo da poterlo comparare anche con le visualizzazioni sperimentali della figura 3.2 e capire qualitativamente in che regime di flusso ci troviamo: è chiarissima la presenta della bolla supersonica sulla prima parte del profilo (fino a circa il 60% della corda, ascissa in cui è collocato lo *shock*). Come da *design*, il profilo supercritico accelera notevolmente il flusso al bordo d'attacco raggiungendo molto a monte



(a) *Simulazione* `su2`



(b) *Simulazione* WIND [11] *e dati sperimentali* [13]

Figura 3.14: Simulazione di riferimento – c_p

la condizione sonica e generando un urto piuttosto debole, che mantiene le dissipazioni contenute; si vede infatti che a valle dell'urto il regime è ancora alto subsonico. Il punto d'arresto è perfettamente visibile e spostato appena sul ventre a causa dell'incidenza della corrente che investe il profilo.

(b) La figura 3.14 mostra la curva ottenuta per la distribuzione dei c_p attorno a confronto con i dati numerici e sperimentali di riferimento per queste simulazioni. Si nota una leggera discrepanza nelle zone dell'urto e della successiva compressione isoentropica.

In figura 3.15 viene rappresenta una visualizzazione della sensibilità parametrica della soluzione rispetto ad I_Z: la zona colorata in grigio mostra il *range* delle soluzioni in $\mathbf{V}^{\mathrm{LF}}(\Gamma)$ compreso tra il minimo e il massimo valore di c_p tra tutte le simulazioni effettuate ottenuto per ciascun punto discreto $(x/c)_i$ sulla corda. Le variazioni sono mostrate solo sul dorso, dove sono molto più accentuate a causa della fenomenologia fisica discussa nella sezione 3.1. Si noti come il *range* vari sensibilmente da una soluzione isoentropica (priva di urto) fino a degli urti molto intensi, nonostante l'intervallo di variazione dei parametri considerato sia piuttosto piccolo (si ricorda che $L_{I_{\mathrm{M}}} = 0{,}05$ e $L_{I_\alpha} = 0{,}2°$).

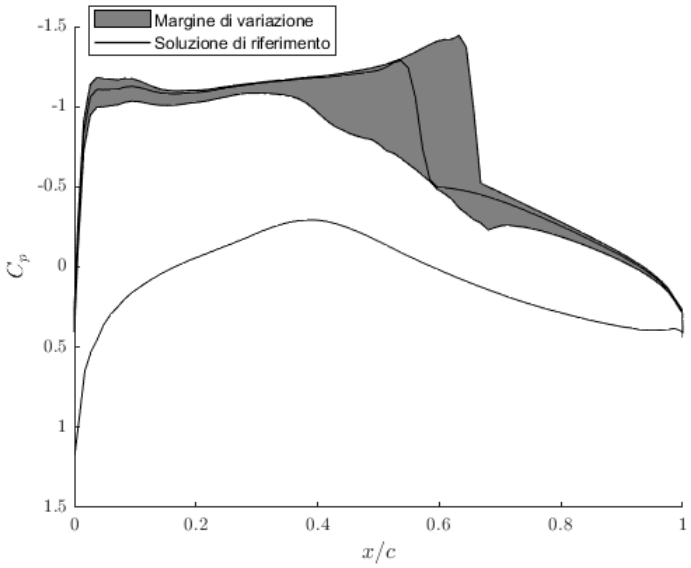


Figura 3.15: *Range* di variazione delle soluzioni in Γ

Le curve in figura 3.16 rappresentano in scala logaritmica l'andamento degli errori di ricostruzione sul *set* $\{\tilde{z}_k\}$ e sono costruite esattamente alla stregua delle figure 2.10 e 2.20.

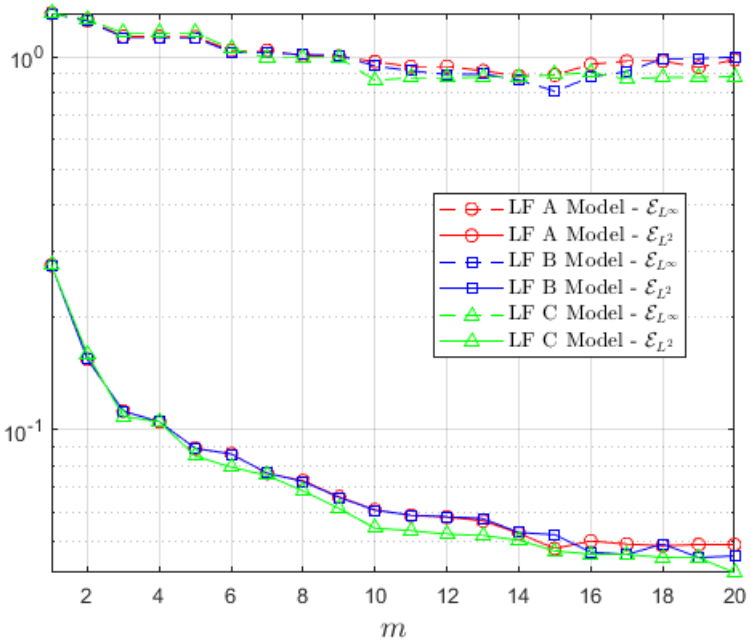
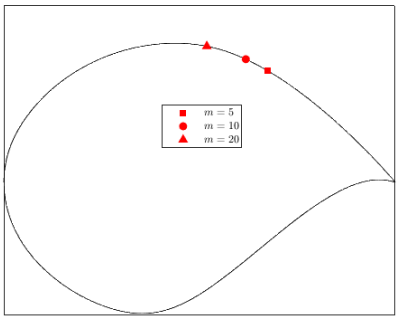


Figura 3.16: Errori di ricostruzione sul profilo

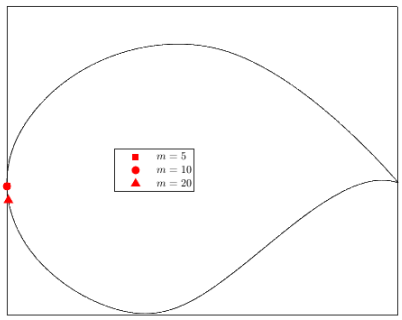
Si nota un rapido appiattimento asintotico delle curve per gli errori medi in norma L^2 su un valore piuttosto alto (≈ 0.05), di diversi ordini di grandezza maggiore rispetto al crollo esponenziale evidenziato nei *test case*. Le curve degli errori massimi in norma L^∞ sono invece addirittura ferme a dei valori dell'ordine dell'unità, comportamento che risulta sintomatico della presenza di alcuni sporadici picchi di errore localizzato che "sporcano" l'intera ricostruzione.

Dopo una fase di *test* preliminari per assicurarsi che l'algoritmo di ricostruzione stesse funzionando correttamente, si è proceduto ad un'analisi separata su dorso e ventre del profilo per risalire alla principale fonte di errore. Come discusso in fase di presentazione delle *mesh* utilizzate (3.2.2), infatti, la quantità di elementi delle griglie nei punti di gradiente più intenso delle soluzioni (*i.e.* la rapidissima espansione al bordo d'attacco e la discontinuità a cavallo dell'urto) potrebbe non essere sufficiente a risolvere efficacemente il campo di moto sul dorso. Sul ventre, per cui in questo regime di M il flusso si mantiene subsonico e l'andamento delle curve è più "dolce", la situazione dovrebbe invece essere meno problematica.

A conferma di questa ipotesi, in figura 3.17 si mostrano i risultati della ricerca della collocazione sul profilo dell'ascissa di errore massimo commesso su tutte le soluzioni in Γ [5]; ci si concentra a titolo di esempio sul modello LF di tipo A.



(a) *Sull'intero profilo*



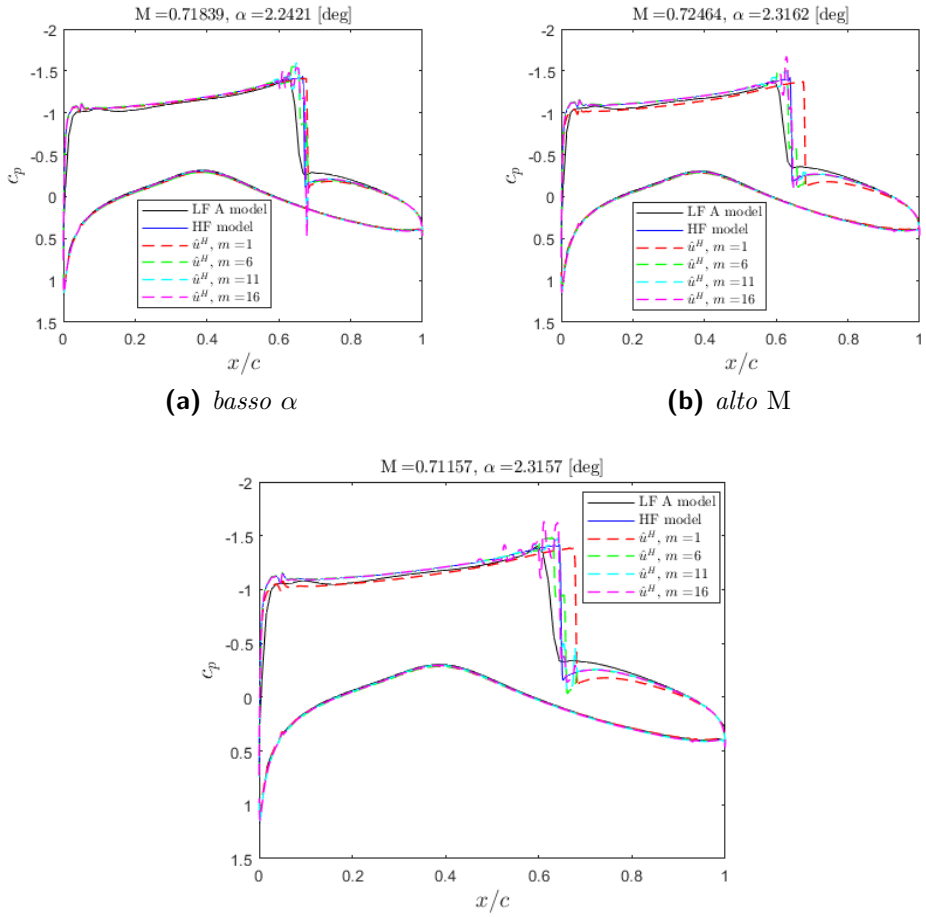
(b) *sul ventre*

Figura 3.17: Collocazione dell'errore di ricostruzione massimo

(a) la figura 3.17a è prodotta cercando il massimo errore su tutti i punti del profilo per ricostruzioni ottenute per diversi valori di m, e mostra con evidenza che l'ascissa più problematica è stanziata nella zona di competenza dell'urto.

(b) forzando invece il programma a cercare l'errore massimo solo tra i valori sul ventre si ottiene la figura 3.17b; in assenza dell'urto si vede bene come la posizione dell'errore si sposti verso il bordo d'attacco, in corrispondenza del punto di arresto della corrente.

[5] Si tratta di un risultato intermedio per il calcolo della funzione $\mathcal{E}_{L^\infty}$

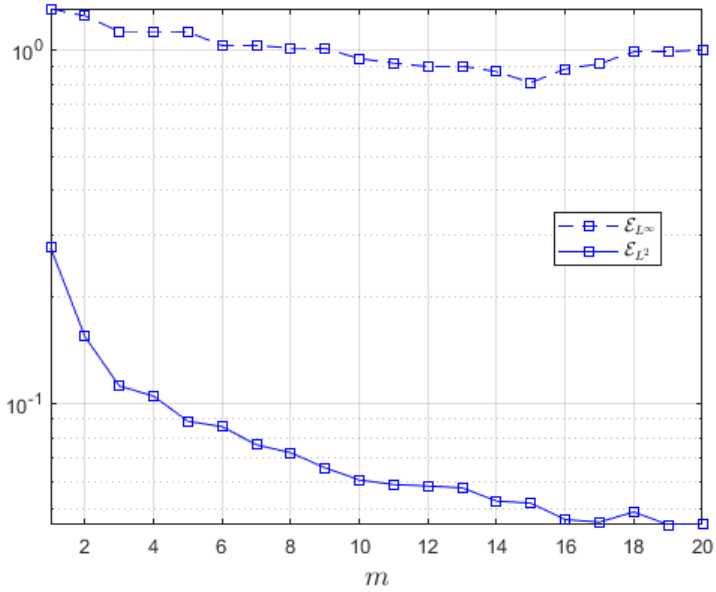


(a) *basso* α **(b)** *alto* M

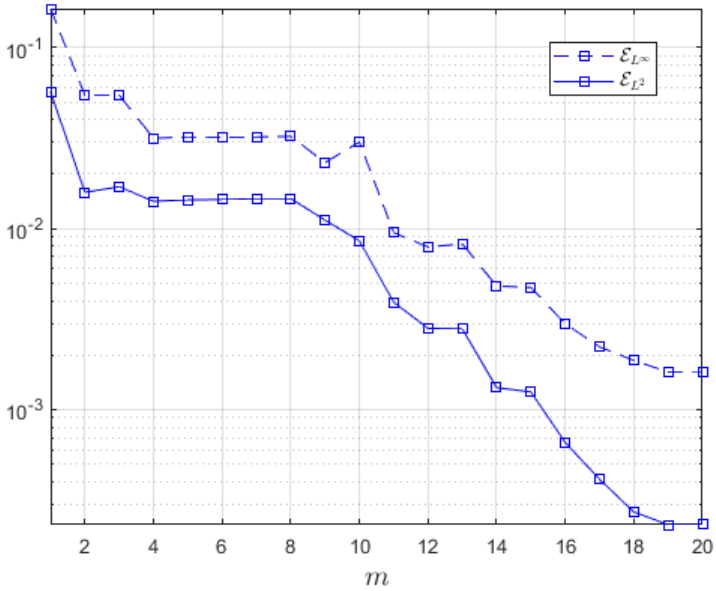
(c) *basso* M

Figura 3.18: Convergenza delle ricostruzioni

La figura 3.18 mostra la convergenza delle ricostruzioni sulla u^H per tre coppie di punti $(\mathrm{M}_\infty, \alpha)$ selezionate dal *set* $\{\tilde{z}_k\}$ e per diversi valori di m; la LF è ancora una volta quella del modello A, utilizzato come base delle ricostruzioni mostrate. Si nota come le ricostruzioni siano soggette ad oscillazioni (più o meno intense a seconda della soluzione considerata) di natura non fisica in corrispondenza della discontinuità data dall'urto. Sul ventre le curve sono invece praticamente indistinguibili, come era lecito attendersi. L'intensità delle oscillazioni cresce con il numero di nodi interpolanti considerato non consentendo il decadimento dell'errore nonostante il miglioramento dell'accuratezza nel restante parte del dominio (andamento asintotico della 3.16).



(a) *dorso*



(b) *ventre*

Figura 3.19: Errori di ricostruzione locali

In figura 3.19 si mostra infine l'andamento degli errori per i punti della ricostruzione collocati solo sul dorso e solo sul ventre del profilo. La figura 3.19a mostra che sul dorso si ha un andamento che ripercorre quello dell'intero profilo, mentre nella 3.19b si ritrova la decrescita quasi-esponenziale che aveva caratterizzato le *performance* dell'algoritmo nei *test* di validazione.

Conclusioni e sviluppi

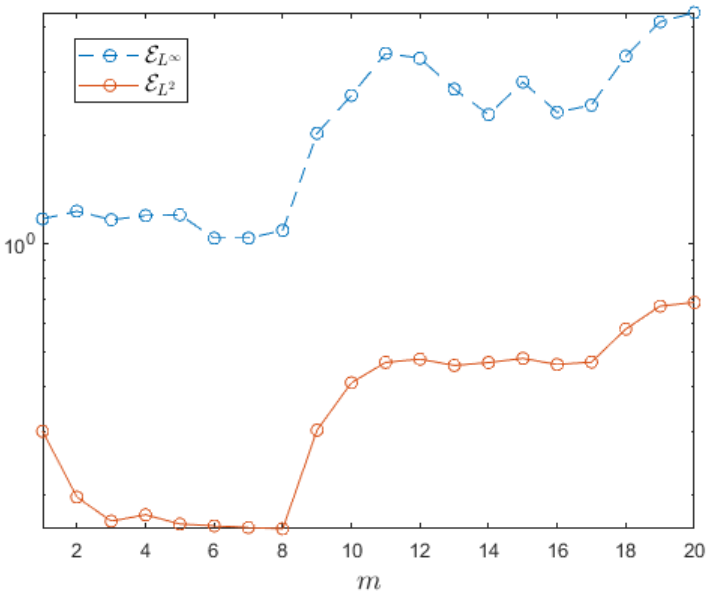
Analizzando i risultati ottenuti per l'applicazione CFD in questo caso di studio transonico, è parso evidente come l'andamento molto eterogeneo della soluzione sulle diverse zone del profilo si rifletta sulla qualità delle ricostruzioni che varia anch'essa fortemente a seconda del tratto considerato.

Una possibile modo di procedere allora potrebbe essere quello di ricostruire due soluzioni diverse – una per il dorso e una per il ventre del profilo – e valutare distintamente gli errori commessi. Si noti però che non è possibile adottare una tale strategia *a posteriori*: le ricostruzioni sono infatti ottenute con la proiezione (1.20) sullo spazio $U^H(\gamma)$ che è lo span delle soluzioni *High-Fidelity* nei nodi interpolanti; la selezione di questi nodi è stata guidata dalle soluzioni sull'*intero profilo* in modo da raccogliere informazioni sulla sensibilità globale della soluzione rispetto ai parametri. A questo punto, ricostruire utilizzando ad esempio solo la parte di $u^L(\gamma)$ e $u^H(\gamma)$ inerenti al ventre dopo aver selezionato le γ perché fossero rappresentative anche delle variazioni sul dorso – che sono peraltro molto più significative – rende l'algoritmo inefficace.

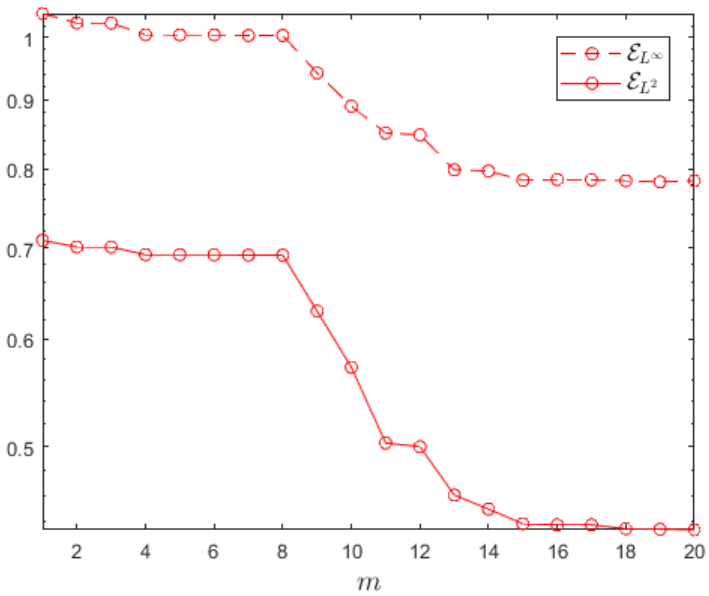
La ricostruzione separata va quindi condotta *a priori*, considerando indipendentemente due modelli u_L^{Ventre} e u_L^{Dorso} e costruendo per ciascuno di essi il relativo campionamento γ *ad hoc*.

A titolo di esempio, si riporta in figura 3.20 il risultato di un approccio di questo tipo condotto a posteriori per la stima degli errori di ricostruzione sulle due parti del profilo. Si può notare immediatamente la completa discordanza dell'andamento di questi risultati rispetto a quelli discussi e mostrati nel corpo centrale della tesi. In particolare si nota la divergenza dell'errore per il dorso all'aumentare dei *pivot* utilizzati m.

In realtà, la strategia *Multi-Fidelity* non ha a priori ragione di non funzionare con l'introduzione di una discontinuità nell'andamento della soluzione considerata (si veda [3]), a patto che come sempre il modello LF coniugato alle caratteristiche dello spazio Γ campionato riesca a raccogliere sufficienti informazioni sulla variazione parametrica. Nel caso di studio in questione, il problema deriva direttamente dai modelli considerati e sarebbe stata pro-



(a) *dorso*



(b) *ventre*

Figura 3.20: Errori di ricostruzione – ricostruzioni separate ottenute impropriamente a posteriori

babilmente necessaria l'introduzione di un nuovo modello HF ancora più accurato (*i.e.* una *mesh* localmente molto fine nella zona della formazione dell'urto). Un indizio sul buon funzionamento del modello nella selezione nodale è dato dall'ottima qualità delle ricostruzioni sul ventre del profilo – dove in verità gli stessi modelli LF sono dei surrogati abbastanza accurati –; le approssimazioni sono state poi formate proiettando su uno spazio di soluzioni u^H che presentavano anch'esse talvolta, per particolari configurazioni dei parametri, delle oscillazioni non fisiche nella soluzione.

L'analisi sull'algoritmo *Multi-Fidelity* condotta in questo lavoro di tesi lascia spazio per diversi possibili margini di ampliamento e sviluppi futuri, tra i quali si annoverano:

- L'utilizzo della strategia *Bi-Fidelity* per l'applicazione a un problema CFD *time-dependent*, ad esempio per un *framework* simile a quello analizzato ma che consideri le equazioni di governo di Eulero instazionarie così come presentate nelle (3.3).
- Un approccio *Multi-Fidelity* di altra natura, che non si risolva nel raffinamento della *mesh* ma che consideri una fisica diversa per ciascun modello (si vedano i possibili rapporti che intercorrono tra un modello HF e il corrispettivo LF nell'Introduzione a questa tesi). Un esempio naturale in questo contesto potrebbe essere rappresentato da un modello LF basato sulle equazioni di Eulero – che dunque trascura una parte di fisica del problema – e uno HF che invece implementa una discretizzazione delle Navier-Stokes (3.1) (ad esempio le RANS); sarebbe interessante capire in questo caso, con un'opportuna scelta della variabile da ricostruire, se Eulero è in grado quantomeno di raccogliere informazioni globali sull'andamento delle soluzioni per poter guidare adeguatamente la scelta delle (costose) Navier-Stokes da valutare.
- Un approccio che utilizzi la strategia *Tri-Fidelity*, nella maniera proposta in [14] già menzionata nel Capitolo 1. A partire dalla costruzione di più modelli LF come si è fatto in questo lavoro, se ne potrebbero utilizzare due simultaneamente considerando l'esistenza di un modello cosiddetto *Medium-Fidelity*.
- Un ampliamento dello spazio I_Z considerato a delle variabili che parametrizzino la geometria del profilo come quelle descritte nella sezione 3.1 [12, 6] per poter ad esempio condurre uno studio di ottimizzazione della forma del profilo per dei parametri operazionali fissati diversi da quelli di progetto.

Bibliografia

[1] John David Anderson Jr. *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.

[2] Renzo Arina. *Fondamenti di aerodinamica*. Levrotto & Bella, 2015.

[3] Claudio Canuto, Sandra Pieraccini e Dongbin Xiu. «Uncertainty quantification of discontinuous outputs via a non-intrusive bifidelity strategy». In: *Journal of Computational Physics* 398 (2019), p. 108885.

[4] PH Cook, MA McDonald e MCP Firmin. «Aerofoil rae 2822-pressure distributions, and boundary layer and wake measurements. experimental data base for computer program assessment». In: *AGARD Report AR* 138 (1979).

[5] Giorgio Graziani. «Aerodinamica». In: *Aerodinamica* (2007).

[6] S Krumscheid, Fabio Nobile e M Pisaroni. «Quantifying uncertain system outputs via the multilevel Monte Carlo method—Part I: Central moment estimation». In: *Journal of Computational Physics* 414 (2020), p. 109466.

[7] Giovanni Monegato. *Metodi e algoritmi per il calcolo numerico*. Clut, 2008.

[8] Akil Narayan, Claude Gittelson e Dongbin Xiu. «A stochastic collocation algorithm with multifidelity models». In: *SIAM Journal on Scientific Computing* 36.2 (2014), A495–A521.

[9] Pramudita S Palar e Koji Shimoyama. «Multi-fidelity uncertainty analysis in CFD using hierarchical kriging». In: *35th AIAA Applied Aerodynamics Conference*. 2017, p. 3261.

[10] Alfio Quarteroni. *Modellistica numerica per problemi differenziali*. Vol. 2. Springer Science & Business Media, 2009.

[11] John W Slater, Julianne C Dudek e Kenneth E Tatum. «The NPARC alliance verification and validation archive». In: *Proceedings of ASME FEDSM00* (2000).

[12] Helmut Sobieczky. «Parametric airfoils and wings, notes on numerical fluid mechanics». In: *edited by K. Fujii and GS Dulikravich* 68 (1998), pp. 71–88.

[13] JJ Thibert, M Grandjacques e LH Ohman. «Experimental data base for computer program assessment». In: *Report of the Fluid Dynamics Panel Working Group* 4 (1979).

[14] Xueyu Zhu, Akil Narayan e Dongbin Xiu. «Computational aspects of stochastic collocation with multifidelity models». In: *SIAM/ASA Journal on Uncertainty Quantification* 2.1 (2014), pp. 444–463.