POLITECNICO
DI MILANO

Scuola di Ingegneria Industriale
Corso di Laurea Magistrale in Ingegneria Aeronautica

Miglioramento di un modello di turbolenza nel codice di calcolo OpenFOAM

Relatore: Prof. Maurizio QUADRIO
Correlatore: Ing. Luca GASPARINI

Tesi di laurea di:
Damiano MOLINARI Matr. 765091

Anno Accademico 2013–2014

Sommario

Questo lavoro di tesi si colloca nell'ambito della fluidodinamica computazionale, e considera modelli di turbolenza per le equazioni di Navier-Stokes incomprimibili mediate alla Reynolds. Il lavoro è svolto all'interno della collaborazione tra il Politecnico di Milano e l'azienda FondTech s.r.l. che utilizza questo tipo di modelli per la consulenza aerodinamica in campo automobilistico, e vede le sue basi su due lavori di tesi precedenti che hanno studiato alcuni aspetti del modello $\overline{v'^2} - f$ con funzioni di parete adattive.

Lo scopo di questo lavoro è quello di proseguire nella direzione intrapresa, validando una nuova versione del modello con e senza l'uso di funzioni di parete adattive. Come riferimento e base di partenza è stato utilizzato il modello $\overline{v'^2} - f$ presente in OpenFOAM, che rappresenta il software utilizzato per l'implementazione e verifica delle caratteristiche del modello.

La tesi contiene una prima parte teorica che chiarisce l'ambito di lavoro e gli attori principali, facendo una breve cronologia sull'evoluzione dei modelli presi in considerazione nell'arco degli anni. Si passa poi all'implementazione del modello, che viene verificato attraverso il confronto di alcuni risultati con i dati reperibili dagli articoli di riferimento e dagli andamenti di set di dati sperimentali e DNS.

La validazione è stata eseguita su una lastra piana ed è stato possibile osservare come il modello preso in considerazione abbia un buon comportamento per la parte di integrazione fino a parete, lasciando buone sensazioni per eventuali prove più specifiche, mentre risulta ancora lievemente carente se utilizzato con funzioni di parete adattive.

Abstract

Subject of this thesis is Computational Fluid Dynamics, and in particular turbulence models are considered for the solution of the incompressible RANS equations. The work stems from an ongoing collaboration between Politecnico di Milano and FondTech s.r.l., where such models are used in automotive application, and continues two preceding thesis, when the $\overline{v'^2} - f$ turbulence model has been considered in conjunction with adaptive wall function.

Aim of the present work is to develop further the model by implementing a new formulation, and to test it with and without wall functions. We start from the $\overline{v'^2} - f$ version available in OpenFOAM, which is the software package used in the thesis.

The document contains a first introduction part, followed by a detailed description of the implementation of the model, and the validation work carried out with comparison with numerical (DNS) and experimental dataset.

The validation, carried out for a boundary layer on a flat plate, reveals that the model performs very well when the near-wall layer is resolved, whereas the wall-function implementation still shows slightly inferior results.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Nel mondo delle competizioni automobilistiche un aspetto di rilevanza fondamentale, e di sempre maggiore interesse, è rappresentato dallo studio dell'aerodinamica. Team e aziende del settore sono così motivate a concentrare tempo, sforzi e denaro nella ricerca degli strumenti e delle soluzioni che permettano di ottenere i migliori risultati possibili in pista sotto questo punto di vista. Per raggiungere questo obiettivo, ingegneri e tecnici possono avvalersi di prove sperimentali, attraverso gallerie del vento, oppure della *CFD (Computational Fluid Dynamics)*, cioè simulazioni numeriche attraverso modelli matematici.

Questo è il lavoro che viene svolto all'interno del gruppo *Fondmetal Technologies (FondTech)*, attraverso l'utilizzo di due gallerie del vento per la parte sperimentale e di notevoli risorse di calcolo per le simulazioni numeriche.

Grazie alla collaborazione fra FondTech e il *Politecnico di Milano* è nato questo lavoro di tesi, che intende continuare il lavoro di ricerca e miglioramento dei modelli di turbolenza utilizzati per la CFD. All'interno di questa categoria rientrano un numero considerevole di metodologie di calcolo e varianti, ma il nostro interesse si focalizzerà sulle RANS, che identificano la soluzione più consona date le caratteristiche del problema in esame. Le basi di questo lavoro sono definite dai precedenti lavori di tesi svolti da Benelli [2] e da Angiolini [1], riguardanti rispettivamente l'implementazione del modello RANS $\overline{v'^2} - f$ con wall-function adattive e la validazione di quest'ultime sul modello $k - \omega$.

Lo scopo di questo lavoro è quello di validare un recente modello di turbolenza selezionato dalla letteratura, utilizzando come base di riferimento la versione utilizzata nel primo lavoro di tesi, cioè un $\overline{v'^2} - f$ proposto originariamente da Davidson [8,24].

Come per i precedenti lavori, lo strumento utilizzato per la verifica delle richieste è *OpenFOAM*, software libero che permette una completa gestione e modifica.

Dalla distribuzione 2.1.0 di questo software è presente già una versione del $\overline{v'^2} - f$ denominata v2f . Di conseguenza, per dotare questo lavoro di una base più possibile comune e ripetibile, si è deciso di utilizzare questa versione come base di partenza. A partire da questo modello si è introdotta una modifica della condizione al contorno della quantità ϵ, come riportato nell'articolo di Davidson, e denominando questa versione come v2fm. Successivamente è stato quindi possibile passare all'obiettivo

principale, cioè quello di valutare un'alternativa più recente, sempre rimanendo all'interno dell'insieme dei modelli RANS incomprimibili a quattro equazioni a base $\overline{v'^2}$. La ricerca preliminare ha evidenziato come candidato il modello $k-\epsilon-\overline{v'^2}/k-\alpha$ proposto da Billard e Laurence ($BL\overline{v'^2}/k$) [6] denominato BLv2k. Per ognuno dei tre modelli è stata poi inserita la possibilità di funzionamento con wall-function adattive come già stato fatto nel primo lavoro di tesi.

La verifica delle prestazioni dei modelli è stata realizzata con delle simulazioni nella semplice geometria della lastra piana, seguendo le specifiche riportate dall'esperimento di Wieghardt [25], cioè una lastra di $5m$ di lunghezza con velocità asintotica di $33m/s$ con fluido di lavoro aria a $\nu = 1.51 \cdot 10^{-5} m^2/s$ e $\rho = 1Kg/m^3$, in modo da avere un massimo $Re_x = 10.9 \cdot 10^6$ ($Re_\theta \approx 15500$).

I dati così ottenuti dalla prove del BLv2k su questa geometria sono stati confrontati con i dati ottenuti dagli articoli degli stessi Billard e Laurance [6] (lastra piana e channel flow a $Re_\tau = 2000$ ($Re_\theta \approx 6500$)) per osservare la corretta scrittura del modello. Invece il confronto complessivo dei tre modelli presi in considerazione è stato eseguito con i dati sperimentali dello stesso Wieghardt svolti nel 1951 a $Re_\theta = 4850, 15500$, i risultati del 1999 ottenuti da Osterlund [19] a $Re_\theta = 5156, 6510$, e anche con l'andamento fornito dalla DNS su lastra piana svolta da Sillero e Jimenez nel 2011 a $Re_\theta = 5000, 6500$ [22].

Quanto appena accennato viene meglio chiarito e descritto nel resto del documento, che è suddiviso come segue:

CAPITOLO 1 - Accenni di teoria: in una prima parte si accenna alle basi che reggono lo studio della fluidodinamica, definendo le possibili strade percorribili a seconda dell'interesse di studio, in modo da fissare il luogo preciso sul quale questo lavoro viene definito, cioè i modelli RANS incomprimibili a 4 equazioni; nella seconda parte si descrivono gli effetti e le principali caratteristiche che si creano all'interno di un fluido a contatto con una parete solida, e la descrizione delle principali tecniche per tenerne conto all'interno di un modello di turbolenza;

CAPITOLO 2 - Modelli a base $\overline{v'^2}$: descrizione dei modelli alla base di questo lavoro, facendo una cronologia delle varianti e delle modifiche apportate nell'arco del tempo, definendo il modello $BL\overline{v'^2}/k$ frutto della collaborazione tra Billard e Laurence;

CAPITOLO 3 - Descrizione di *OpenFOAM*: descrizione dello strumento utilizzato per questa esperienza, dalla definizione dell'architettura, alla definizione dei file necessari al lancio di una simulazione, passando per la scrittura delle modifiche volute;

CAPITOLO 4 - Risultati: i risultati delle simulazioni vengono analizzati per valutare le prestazioni del nuovo modello, suddividendoli in 4 set di confronti:

1. confronto dei riferimenti, in cui si confrontano i due set sperimentali e la DNS negli andamenti della velocità e del coefficiente d'attrito;
2. validazione dell'implementazione del BLv2k, in cui vengono confrontate le curve del nuovo modello implementato con le curve originali fornite negli articoli di riferimento;
3. confronto dei tre modelli v2f, v2fm, BLv2k;
4. confronto per ogni modello di tre simulazioni con wall-function adattive a differenti posizioni di y^+, utilizzando come riferimento la simulazione integrata fino a parete.

CAPITOLO 5 - Conclusioni: Dati gli obiettivi prefissati e dalle validazioni eseguite, vengono tratte le conclusioni di questo lavoro.

Capitolo 1

Cenni di teoria

1.1 Dai principi di conservazione al $\overline{v'^2} - f$

Nell'ambito della fluidodinamica, un settore sempre più utilizzato e studiato riguarda la simulazione numerica delle equazioni di Navier Stokes mediate alla Reynolds attraverso modelli di turbolenza: la cosiddetta *CFD (Computational Fluid Dynamics)*. La risoluzione di un problema fluidodinamico consiste nella risoluzione di equazioni per il calcolo di diverse proprietà del fluido, come per esempio velocità, pressione, densità, e temperatura, in funzione dello spazio e del tempo. La necessità di utilizzare dei modelli semplificati deriva dall'elevata difficoltà di queste equazioni e quindi dall'infattibilità della loro soluzione, per via dei notevoli tempi di calcolo richiesti dai calcolatori. Altro aspetto che favorisce il loro utilizzo è l'impossibilità di riprodurre correttamente esperimenti di laboratorio di particolari condizioni di interesse, sia per motivi tecnologici sia per motivi economici, che altrimenti porterebbero anche loro alla conoscenza delle proprietà del fluido richieste.

Partiamo quindi dall'inizio e vediamo da dove derivano e come nascono questi modelli.

1.1.1 Equazioni di Navier-Stokes e loro decomposizione

Per la corretta descrizione dei fenomeni che avvengono all'interno di un fluido, è necessario definire un sistema di equazioni che tenga conto di tutte le variabili in gioco all'interno del dominio, e che sia in grado di rappresentare correttamente le relazioni che tra di loro si creano. Per fare questo la fluidodinamica ci mette a disposizione le equazioni differenziali di conservazione di massa (scalare), quantità di moto (vettoriale) ed energia (scalare), che sono il risultato del bilancio di grandezze in un generico volume di controllo. Introducendo anche l'equazione di stato, e le necessarie condizioni iniziali e al contorno, otteniamo un sistema chiuso (*equazioni di Navier-Stokes*), risolvibile numericamente. Queste equazioni per un flusso incomprimibile a

proprietà costanti possono essere scritte nella seguente forma:

$$\nabla \cdot \mathbf{U} = 0$$
$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{U}. \tag{1.1}$$

La scelta di risolvere numericamente il sistema 1.1 risulta molto dispendiosa per le dimensioni tipiche di un problema applicativo, per questo motivo si preferisce risolvere le equazioni solamente per le componenti medie delle variabili e trascurando le fluttazioni che queste hanno attorno al valore medio. Questa operazione di decomposizione viene eseguita su tutte le quantità, e prende anche il nome di *decomposizione di Reynolds*. Prendendo ad esempio la velocità, possiamo quindi considerarla come somma di una componente media ($\overline{U}$) e una fluttuazione (U'):

$$U(x,t) = \overline{U}(x) + U'(x,t), \tag{1.2}$$

in cui la componente media è stata ottenuta da:

$$\overline{U}(x) = \lim_{T \to +\infty} \frac{1}{T} \int_0^T U(x,t)dt. \tag{1.3}$$

Sostituendo questa definizione all'interno del sistema di Navier-Stokes 1.1 per tutte le variabili, e mediando tutta l'equazione, si ottiene il sistema di equazioni *RANS (Reynolds Averaged Navier-Stokes)*.

$$\nabla \cdot \overline{\mathbf{U}} = 0$$
$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}}\overline{\mathbf{U}}) = -\frac{1}{\rho}\nabla \overline{p} + \nu \nabla^2 \overline{\mathbf{U}} - \nabla \cdot (\overline{\mathbf{U}'\mathbf{U}'}). \tag{1.4}$$

1.1.2 Modellazione degli sforzi di Reynolds

Confrontando i sistemi di partenza 1.1 e quello decomposto 1.4, la struttura rimane invariata ad eccezione di un solo termine ($-\nabla \cdot (\overline{\mathbf{U}'\mathbf{U}'})$), che presenta ancora le componenti di fluttuazione del campo di moto. Il termine all'interno della divergenza è un tensore di 9 componenti (solo 6 indipendenti essendo un tensore simmetrico), che prende il nome di *tensore degli sforzi di Reynolds*.

$$\mathbf{U}'\mathbf{U}' = \begin{bmatrix} uu & uv & uw \\ uv & vv & vw \\ uw & vw & ww \end{bmatrix}. \tag{1.5}$$

Purtroppo questo termine non è scrivibile come funzione di quantità medie, quindi il sistema 1.4 non è chiuso e il tensore 1.5 deve essere modellato:

RSM (Reynolds Stress Model) **:** modellazione diretta delle componenti del tensore degli sforzi di Reynolds, che richiede quindi di definire 6 equazioni differenziali alle derivate parziali, una per ogni componente del tensore;

EVM (Eddy Viscosity Model) **:** modellazione che si basa sull'*ipotesi di Boussinesq* cioè la possibilità di ricavare l'intero tensore degli sforzi di Reynolds ricavandolo da quantità medie note, e introducendo la definizione della *viscosità turbolenta* (ν_t):

$$-\overline{U'U'}_{ij} = -S_{ij} = 2\nu_t\overline{e_{ij}} - \frac{2}{3}\overline{k}\delta_{ij}, \tag{1.6}$$

con $\overline{e_{ij}} = \frac{1}{2}\left(\frac{\partial\overline{u_i}}{\partial x_i} + \frac{\partial\overline{u_j}}{\partial x_i}\right)$ e $\overline{k}$ =*energia cinetica turbolenta*= $\frac{1}{2}\left(\overline{u'u'} + \overline{v'v'} + \overline{w'w'}\right)$.
Il tensore S_{ij} diventa (in coordinate cartesiane):

$$\begin{bmatrix} 2\nu_t\left(\frac{\partial\overline{u}}{\partial x}\right) - \frac{2}{3}\overline{k} & \nu_t\left(\frac{\partial\overline{u}}{\partial y} + \frac{\partial\overline{v}}{\partial x}\right) & \nu_t\left(\frac{\partial\overline{w}}{\partial x} + \frac{\partial\overline{u}}{\partial z}\right) \\ \nu_t\left(\frac{\partial\overline{u}}{\partial y} + \frac{\partial\overline{v}}{\partial x}\right) & 2\nu_t\left(\frac{\partial\overline{v}}{\partial y}\right) - \frac{2}{3}\overline{k} & \nu_t\left(\frac{\partial\overline{v}}{\partial z} + \frac{\partial\overline{w}}{\partial y}\right) \\ \nu_t\left(\frac{\partial\overline{w}}{\partial x} + \frac{\partial\overline{u}}{\partial z}\right) & \nu_t\left(\frac{\partial\overline{v}}{\partial z} + \frac{\partial\overline{w}}{\partial y}\right) & 2\nu_t\left(\frac{\partial\overline{w}}{\partial z}\right) - \frac{2}{3}\overline{k} \end{bmatrix}. \tag{1.7}$$

In questo modo le equazioni diventano:

$$\begin{aligned} \nabla\cdot\overline{\mathbf{U}} &= 0 \\ \frac{\partial\overline{\mathbf{U}}}{\partial t} + \nabla\cdot(\overline{\mathbf{U}}\overline{\mathbf{U}}) &= -\frac{1}{\rho}\nabla\overline{P} + \nabla\cdot(\nu_e(x)\nabla\overline{\mathbf{U}}), \end{aligned} \tag{1.8}$$

$$\nu_e(x) = \nu + \nu_t(x). \tag{1.9}$$

In cui si nota l'introduzione di un nuovo termine di pressione: $\overline{P} = \overline{p} + 2/3\overline{k}$ necessario per semplificare il termine di energia cinetica turbolenta presente nella diagonale della matrice degli sforzi di Reynolds. Questa operazione è lecita in quanto stiamo lavorando con fluidi a proprietà costanti, quindi la pressione risulta nota a meno di una costante.
Con la definizione introdotta dall'ipotesi di Boussinesq, si sottintendono però due importanti assunzioni:

- la possibilità di esprimere un termine legato alle fluttuazioni turbolente come funzione del campo di moto medio (ipotesi intrinseca) e addirittura come funzione lineare del suo gradiente (ipotesi specifica);
- definire come termine di proporzionalità una viscosità, che usualmente si riferisce ad una proprietà del fluido, ma che in questa definizione diventa una proprietà della corrente.

A questo punto l'attenzione si sposta sulla viscosità turbolenta che rappresenta l'unico termine necessario per la chiusura del problema, e che può essere rappresentata come il prodotto di una lunghezza (l^*) per una velocità (u^*) di

riferimento.

$$\nu_t = u^* l^*. \tag{1.10}$$

I modelli utilizzati per questo lavoro di tesi appartengono al gruppo degli EVM, e per questo si prosegue descrivendo questa categoria di modelli.

1.1.3 Modelli EVM

A seconda del numero di equazioni differenziali necessarie alla formulazione della viscosità turbolenta, possiamo raggruppare i modelli derivanti.

Modelli a zero equazioni: Sono modelli che non introducono ulteriori equazioni differenziali, ma che si limitano a definire dei valori caratteristici per la lunghezza e la velocità di riferimento. Un esempio di questa categoria è la *Prandtl mixing length*, un modello per correnti di parete bidimensionale in cui:

$$\nu_t = l_m^2 \cdot |\frac{d\overline{u}}{dy}| \qquad \text{ed} \qquad l_m = \kappa y^+, \tag{1.11}$$

dove κ rappresenta la *costante di Kolmogorov*, mentre y^+ rappresenta la distanza da parete in direzione verticale adimensionalizzata in unità di parete (vedi Sezione 1.2).

Modelli ad 1 equazione: Appartengono a questa categoria i modelli che fanno dipendere la viscosità turbolenta da grandezze che vengono ricavate attraverso una sola equazione differenziale esterna, come per esempio il modello k, in cui $\nu_t = c\sqrt{k} l_m$. Dove k (energia cinetica turbolenta) viene ottenuta da un'equazione differenziale di trasporto.

Modelli a 2 equazioni: A questa categoria appartengono i più diffusi e comuni modelli di simulazione RANS, perchè rappresenta un buon compromesso tra accuratezza e velocità di soluzione. I principali sono:

- ***modello*** $k-\epsilon$: $\nu_t = c_\mu k^2/\epsilon$ con le relative equazioni differenziali per k ed ϵ, dove quest'ultima rappresenta la velocità di dissipazione dell'energia cinetica per unità di massa, definita come: $\epsilon = 2\nu e_{ij} e_{ij}$;
- ***modello*** $k-\omega$: $\nu_t = c_\mu k/\omega$ con le relative equazioni differenziali per k ed ω, dove quest'ultima rappresenta il rapporto: $\omega = \epsilon/k$.

Modelli a 4 equazioni: Dai modelli classici a 2 equazioni passiamo direttamente a quelli a 4 equazioni, che rappresentano il soggetto di questo lavoro di tesi. In particolare consideriamo il gruppo dei modelli $\overline{v'^2} - f$, che sono una evoluzione dei modelli $k-\epsilon$, in cui vengono appunto aggiunte un'ulteriore equazione di trasporto per $\overline{v'^2}$ ed un'equazione ellittica per f. In questo modo otteniamo il

modello $k-\epsilon-\overline{v'^2}-f$ (denominato appunto $\overline{v'^2}-f$ o più semplicemente *v2f*), ma fanno parte di questa categoria anche varianti di questo modello, come per esempio il $k-\epsilon-\overline{v'^2}/k-\alpha$ (denominato $BL\overline{v'^2}/k$ o più semplicemente *BLv2k*) che sarà la variante scelta per l'implementazione ed il confronto.
La viscosità turbolenta per i modelli $\overline{v'^2}-f$ viene rappresentata generalmente come:

$$\nu_t = c_\mu \overline{v'^2} T(k,\epsilon), \tag{1.12}$$

dove c_μ è una costante, $T(k,\epsilon)$ è una funzione che si differenzia a seconda del tipo di modello, e che dimensionalmente è un tempo, mentre $\overline{v'^2}$ è la varianza delle fluttuazioni turbolente della componente di velocità normale alla parete e garantisce alla ν_t un andamento più corretto vicino alle pareti, permettendo a questi modelli di essere integrati fino a parete.
Una descrizione accurata di questi ultimi modelli seguendo la loro evoluzione temporale verrà trattata nel Capitolo 2, dopo aver concluso la definizione degli strumenti necessari alla comprensione e valutazione degli stessi.

1.2 Comportamento vicino a parete

Specializzando le equazioni RANS per un canale piano di altezza δ, dove il flusso medio è bidimensionale, è possibile definire lo sforzo totale τ come:

$$\tau(y) = \mu\frac{d\overline{u}}{dy} - \rho\overline{uv} \qquad \text{oppure} \qquad \tau(y) = \tau_w(1-\frac{y}{\delta}). \tag{1.13}$$

Nella prima formulazione il primo termine rappresenta il contributo dovuto alla viscosità, mentre il secondo rappresenta l'effetto degli sforzi di Reynolds. Nella seconda formulazione il termine τ_w è lo sforzo alla parete. La definizione di questo termine ci permette di ricavare delle unità caratteristiche di lunghezza e velocità direttamente legate a quello che succede in prossimità della parete. In particolare possiamo definire:

velocità di attrito: $u_\tau = \sqrt{\tau_w/\rho}$;

lunghezza viscosa: $\delta_\nu = \nu/u_\tau$.

A loro volta queste quantità possono essere utilizzate per adimensionalizzare le variabili del nostro modello (come $k, \epsilon, \overline{v'^2}, \omega ...$) oppure quelle geometriche (come per esempio la distanza verticale da parete) ottenendo così le *variabili di parete* che qui riportiamo.

$$y^+ = \frac{y}{\delta_\nu} \quad k^+ = \frac{k}{u_\tau^2} \quad \epsilon^+ = \frac{\epsilon\nu}{u_\tau^4} \quad \overline{v'^2}^+ = \frac{\overline{v'^2}}{u_\tau^2} \quad \omega^+ = \frac{\omega\nu}{u_\tau^2}. \tag{1.14}$$

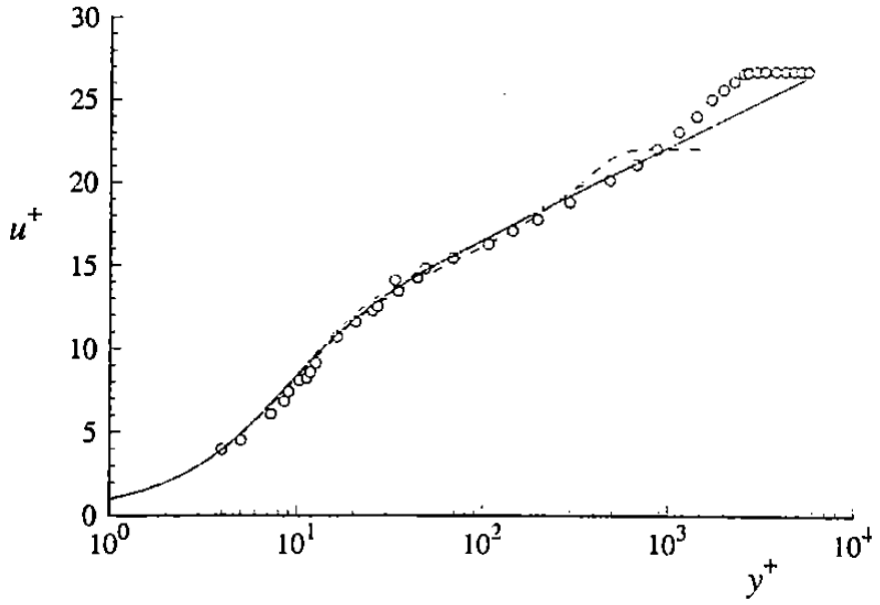

Figura 1.1: Profilo di velocità media in unità di parete [20]

L'importanza di questa riscalatura sta nel fatto che l'andamento delle variabili in coordinate di parete, risulta universale, quindi, a parità di numero di Reynolds, da esperimenti differenti (sia per valore del numero di Reynolds, sia per la geometria: canale piano, lastra piana, ecc.) è possibile osservare gli stessi comportamenti, come per il profilo di velocità media riportato nella Figura 1.1.

La Figura 1.1 fa anche intuire la presenza di differenti regioni al variare della distanza dalla parete. Queste regioni vengono suddivise come riportato nella Figura 1.2. L'andamento universale del profilo di velocità fa si che questo profilo venga spesso chiamato *legge di parete.*

Questa può essere divisa in due macro regioni: l'*inner layer* e l'*outer layer.* Partendo dalla regione più vicino a parete troviamo il *sub-strato viscoso*, dove le caratteristiche della corrente sono appunto dettate unicamente dalla viscosità del fluido, e la velocità varia linearmente con la distanza da parete ($u^+ = y^+$). Spostandoci dalla parete troviamo il *buffer layer*, ossia una zona di raccordo tra la zona viscosa e lo *strato logaritmico*, dove appunto la velocità varia logaritmicamente con la distanza ($u^+ = 1/\kappa \ln(y^+) + A$, dove κ è la già citata costante di Kolmogorov= 0.41 e $A = 5$).

1.2.1 Damping function e Wall function

Le *damping functions*, o funzioni di smorzamento, hanno lo scopo di adeguare il comportamento del modello vicino a parete a quello corretto, non comportando cambiamenti invece lontano da parete. Queste funzioni vengono imposte direttamente

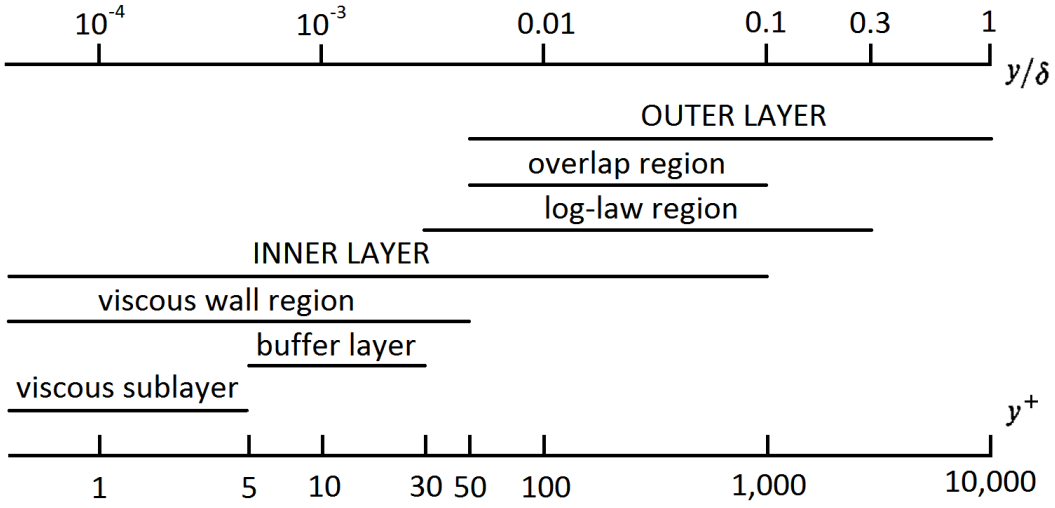

Figura 1.2: Suddivisione della legge di parete [20]

sulla viscosità turbolenta, e per esempio, per il modello $k-\epsilon$ si può scrivere:

$$\nu_t = f_\mu c_\mu \frac{k^2}{\epsilon} \qquad \text{con} \qquad f_\mu = \exp(\frac{-2.5}{1+Re_L/50}). \tag{1.15}$$

Queste funzioni non vengono utilizzate invece per i modelli $\overline{v'^2}-f$ in quanto la viscosità turbolenta presenta la quantità $\overline{v'^2}$, che tendendo a zero molto rapidamente, ha lo stesso effetto di una dumping function, come verrà meglio chiarito nel Capitolo 2.

Calcoli RANS basati sui modelli fino a qui descritti, con o senza funzioni di smorzamento potrebbero comunque essere molto onerosi dal punto di vista computazionale. Infatti risulta evidente che la mesh utilizzata deve arrivare sufficientemente vicino a parete per descrivere ad esempio il profilo di Figura 1.1. Questo comporta la necessità di avere un numero elevato di celle vicino a parete, comportando un tempo di calcolo che cresce velocemente al crescere della complessità geometrica del caso e del valore del numero di Reynolds.

Per fornire un esempio numerico, la zona più vicina a parete corrisponde al sub-strato viscoso, cioè $y^+ < 5$. Se volessimo usare una mesh che abbia il suo primo centro cella almeno a circa $y^+ = 1$, questo comporta, utilizzando un flusso d'aria con $\nu = 1 \cdot 10^{-5} m^2/s, \rho = 1 Kg/m^3$ e circa $\tau_w = 1 Kg/ms^2$ (plausibile con una velocità della corrente di $20-30 m/s$ e una lastra piana di qualche metro di lunghezza), avere una cella di altezza $h = 2 \cdot 10^{-5} m = 0.02 mm$, quindi molto piccola.

Per ovviare a questo potenziale problema è possibile ricorrere all'uso delle *wall functions*. La tecnica delle funzioni di parete consiste nel fissare il primo centro cella, non nel sub-strato viscoso, ma nella zona logaritmica del profilo di velocità, dove conosciamo dalla teoria il comportamento della corrente. A questo punto, i

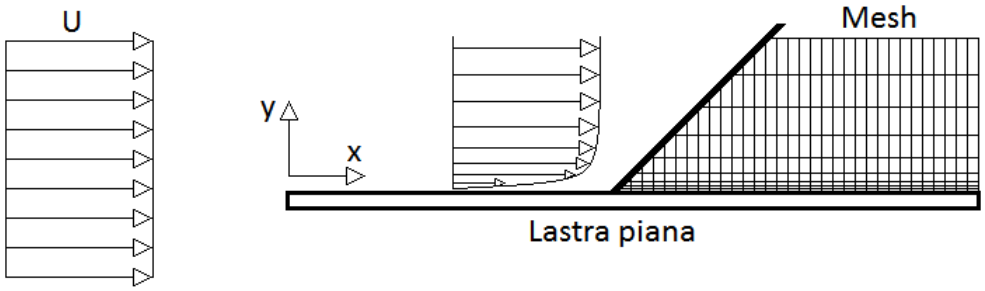

Figura 1.3: Schema della lastra piana e della mesh

parametri nella prima cella (che risulterebbero errati, in quanto non si è tenuto conto del sub-strato viscoso), non vengono calcolati, ma ottenuti dalle funzioni di parete note dalla teoria. Il vantaggio di questa tecnica è quindi una riduzione notevole della dimensione complessiva della mesh. Il primo centro cella deve però sempre cadere all'interno della zona di validità della funzione di parete utilizzata, cosa non sempre facile, soprattutto nelle zone di separazione della corrente.

In questo lavoro verranno invece utilizzate delle *wall function adattive*, cioè che si possono utilizzare quando la prima cella della mesh viene a cadere in una qualsiasi posizione. Invece di utilizzare una funzione analitica teorica per ricavare i valori delle incognite nel primo centro cella, si interpolano valori predeterminati contenuti in una tabella ottenuta per esempio da una DNS, oppure con una prova svolta con integrazione fino a parete. In questo caso quindi, la procedura di esecuzione di una prova è la seguente:

- viene eseguita una prova su una geometria molto semplice (una lastra piana per esempio) con una mesh molto fitta, che quindi vede il suo primo centro cella all'interno del sub-strato viscoso. Questo tipo di simulazione, con integrazione delle equazioni fino a parete, prende anche il nome di *Low-Reynolds (Low-Re)*;

- dalla prova Low-Reynolds viene ricavata una tabella contenente i valori (in unità di parete) delle variabili del modello. Viene anche inserito il numero di Reynolds locale (calcolato con la distanza da parete del centro cella), fondamentale per i passaggi successivi.

- ora si può passare alla creazione della mesh *High-Reynolds (High-Re)* per il problema da studiare (cioè quella da usare con le wall function) mettendo il primo centro cella ad una distanza molto più elevata (anche y^+ dell'ordine delle centinaia). In questo calcolo lanciando il modello con questa mesh meno raffinata si procede come segue:

  - stima della velocità nel primo centro cella: U_1;

- calcolo del numero di Reynolds locale: $Re_1 = U_1 y_1/\nu$ (dove y_1 è l'altezza del primo centro cella);
- lettura di tutte le variabili del modello associate al Re_1 dalla tabella tramite interpolazione;
- calcolo di u_τ avendo a disposizione y_1 dalla mesh e y_1^+ dalla tabella;
- trasformazione delle variabili lette dalla tabella in forma dimensionale e assegnamento nel centro cella.

Questo procedimento è possibile, utilizzando una sola tabella, perchè i valori al suo interno sono in unità di parete, quindi, come già detto, i profili delle variabili sono universali.

Capitolo 2

Modelli $\overline{v'^2} - f$

Riprendendo quanto descritto alla fine della prima parte del Capitolo 1, i modelli $\overline{v'^2} - f$ sono modelli RANS a 4 equazioni ($k - \epsilon - \overline{v'^2} - f$), e sono una evoluzione del modello a 2 equazioni ($k - \epsilon$). Rappresentano una soluzione al problema della chiusura delle equazioni RANS già descritte che vengono qui riportate:

$$\nabla \cdot \overline{\mathbf{U}} = 0$$
$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}\mathbf{U}}) = -\frac{1}{\rho}\nabla \overline{P} + \nabla \cdot (\nu_e(x)\nabla \overline{\mathbf{U}}), \tag{2.1}$$

$$\nu_e(x) = \nu + \nu_t(x) \quad ; \quad \nu_t = c_\mu \overline{v'^2} T(k, \epsilon). \tag{2.2}$$

A differenza del modello $k - \epsilon$ in cui la viscosità turbolenta (ν_t) viene definita utilizzando l'energia cinetica turbolenta k per definire la necessaria scala di velocità, in questo modello si utilizza la $\overline{v'^2}$. Questa può essere identificata come la componente di sforzo normale a parete e rappresenta il giusto smorzamento del trasporto turbolento vicino a parete. Infatti, $\overline{v'^2}$ tendendo a zero molto velocemente (tende a zero come y^4, a differenza di $\overline{u'^2}$ e $\overline{w'^2}$ che tendono come y^2), possiede lo stesso effetto di una damping-function, adeguando il valore della ν_t a parete, come visibile nella Figura 2.1.

Per questo motivo il $\overline{v'^2} - f$ è definito un modello Low-Reynolds in quanto avendo un buon comportamento vicino a parete, è possibile eseguire integrazioni senza l'obbligo di utilizzare wall-function. L'equazione per f è invece un'equazione ellittica per modellare gli effetti anisotropi di parete, la cui dipendenza è inserita nel termine sorgente dell'equazione di trasporto di $\overline{v'^2}$.

In questo modo il $\overline{v'^2} - f$ è in grado di fornire una migliore predizione della separazione della corrente e della resistenza viscosa. L'aspetto negativo risiede però nella maggiore complessità, dovuta all'introduzione di due ulteriori equazioni, ma nonostante questo gode nel complesso di un'ottima stabilità numerica.

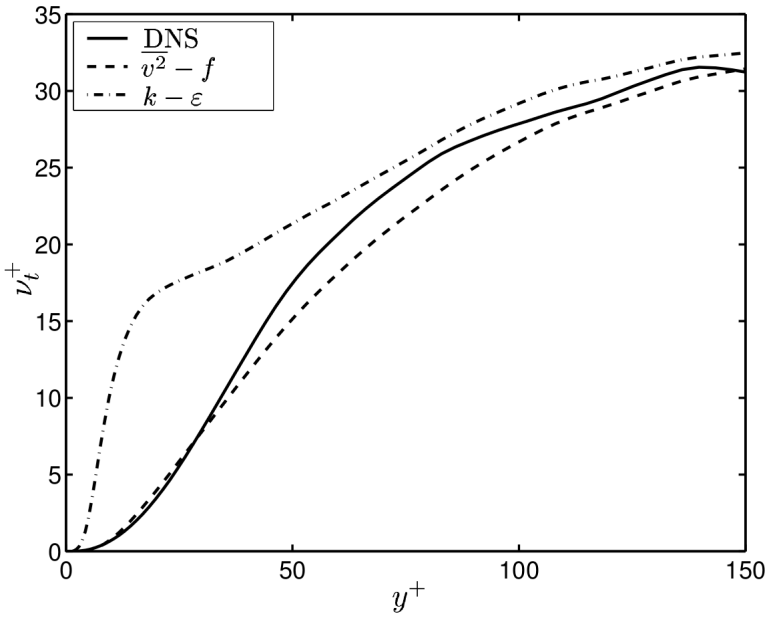

Figura 2.1: Comportamento della ν_t per i modelli $k - \epsilon$ e $\overline{v'^2} - f$

2.1 Evoluzione del $\overline{v'^2} - f$

La prima versione del $\overline{v'^2} - f$ è stata introdotta da Durbin nel 1991 [9], e col passare del tempo ha subito numerose variazioni ed influenze. Questa sezione vuole riassumere le principali varianti proposte da allora fino ai giorni nostri col modello $BL\overline{v'^2}/k$ definito da Billard e Laurence nel 2012 [6].

Di seguito viene riportata la prima versione del modello:

$$\begin{cases} \frac{Dk}{Dt} & = P - \epsilon + \partial_j \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \partial_j k \right] \\ \frac{D\epsilon}{Dt} & = \frac{C_{\epsilon 1} P - C_{\epsilon 2} \epsilon}{T} + \partial_j \left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon} \right) \partial_j \epsilon \right] \\ \frac{D\overline{v'^2}}{Dt} & = kf - \overline{v'^2} \frac{\epsilon}{k} + \partial_j \left[(\nu + \frac{\nu_t}{\sigma_{v^2}}) \partial_j \overline{v'^2} \right] \\ f_h & = f - L^2 \partial_j \partial_j f \end{cases}, \tag{2.3}$$

$$\nu_t = C_\mu \overline{v'^2} T, \qquad f_h = \frac{1}{T} (C_1 - 1) \left(\frac{2}{3} - \frac{\overline{v'^2}}{k} \right) + C_2 \frac{P}{k}, \tag{2.4}$$

$$\lim_{y \to 0} \epsilon = 2\nu \left(\frac{k}{y^2} \right) \qquad \lim_{y \to 0} f = \frac{-20\nu^2}{\epsilon} \left(\frac{\overline{v'^2}}{y^4} \right) , \tag{2.5}$$

in cui le costanti sono riportate nella tabella 2.1, mentre le scale di tempo e spazio turbolente sono definite da:

Tabella 2.1: Costanti del modello Durbin del 1991

C_1	C_2	$C_{\epsilon 1}$	$C_{\epsilon 2}$	σ_k	σ_ϵ	σ_{v^2}	C_L	C_η	C_T	C_μ
1.2	0.3	1.7	2.0	1.3	1.6	1.3	0.17	80	6	0.2

Tabella 2.2: Costanti del modello Lien-Kalitzin del 2001

C_1	C_2	$C_{\epsilon 1}$	$C_{\epsilon 2}$	σ_k	σ_ϵ	σ_{v^2}	C_L	C_η	C_T	C_μ
1.4	0.3	$1.4\left(1+0.05\sqrt{k/\overline{v'^2}}\right)$	1.9	1	1.3	1	0.23	70	6	0.22

$$\begin{cases} L = & C_L \max\left[\frac{k^{3/2}}{\epsilon}, C_\eta\left(\frac{\nu^3}{\epsilon}\right)^{1/4}\right] \\ T = & \max\left[\frac{k}{\epsilon}, C_T\left(\frac{\nu}{\epsilon}\right)^{1/2}\right] \end{cases}. \tag{2.6}$$

Questo modello utilizza un valore abbastanza alto di $C_{\epsilon 1}$ rispetto alla consuetudine del $k-\epsilon$, e sarà l'oggetto di successive modifiche. Infatti per meglio rappresentare la dissipazione anisotropa nella regione logaritmica questo termine viene reso proporzionale a P/ϵ [10], successivamente viene fatto dipendere dalla distanza da parete, per arrivare infine alla formulazione definitiva [11], in cui viene fatto dipendere da $\sqrt{k/\overline{v'^2}}$ per motivi di risoluzione numerica.

Per lo stesso motivo dal modello originale vengono a dipendere delle modifiche alle condizioni al contorno. Infatti originalmente la condizione al contorno per $\overline{v'^2}$ ed f sono accoppiate, ed il fatto che f sia dettata da un'equazione ellittica, fa si che l'implementazione risulti molto instabile. Per risolvere questo problema, si pensò di ridefinire f come $\overline{f} = f + 5\epsilon\overline{v'^2}/k^2$ [15] permettendo di fissare le condizioni a parete separatamente ed in particolare potendo imporre direttamente $\overline{f} = 0$.

Queste modifiche vennero definite singolarmente dai rispettivi autori, e solo successivamente [16] vennero inserite in un'unica formulazione, di cui vengono riportate la nuova f_h, le costanti (nella tabella 2.2), ed il termine $-\overline{v'^2}\frac{\epsilon}{k}$ nell'equazione di $\overline{v'^2}$ divenne $-6\overline{v'^2}\frac{\epsilon}{k}$.

$$f_h = \frac{1}{T}\left[(C_1 - 1)\frac{2}{3} - (C_1 - 6)\frac{\overline{v'^2}}{k}\right] + C_2\frac{P}{k}. \tag{2.7}$$

Di particolare rilevanza è anche la versione introdotta da Davidson, Nielsen e Svenningsson del 2003 [8,24] che si basa sulla formulazione appena descritta, introducendo un limite sulla ν_t. Infatti la formulazione standard della viscosità turbolenta lontano da parete permetterebbe alla $\overline{v'^2}$ di crescere fino ad essere $\overline{v'^2} > \frac{2}{3}k$, cosa che non rispecchia il reale comportamento della $\overline{v'^2}$. La soluzione risiede nel definire un limite alla stessa $\overline{v'^2}$ che si ripercuote nello scegliere $\nu_t = \min\left\{0.09k^2/\epsilon, 0.22\overline{v'^2}T\right\}$. Questa formulazione del $\overline{v'^2} - f$ è quella definita nell'introduzione come *v2f* cioè

Tabella 2.3: Costanti del modello $\varphi - f$

C_1	C_2	$C_{\epsilon 1}$	$C_{\epsilon 2}$	σ_k	σ_ϵ	σ_φ	C_L	C_η	C_T	C_μ
1.4	0.3	$1.4\left(1+0.05\sqrt{1/\varphi}\right)$	1.85	1	1.3	1	0.25	110	6	0.22

la versione implementata in OpenFOAM dalla versione 2.1.0 e che sarà la base di partenza per l'implementazione del nuovo modello.

Le condizioni al contorno per questa formulazione sono le seguenti:

$$k = \overline{v'^2} = f = 0,\ \epsilon = 2\nu k/y^2, \tag{2.8}$$

e quella per ϵ presuppone la conoscenza dei valori della viscosità turbolenta, dell'energia cinetica turbolenta e della distanza da parete nella prima cella. In questo documento, si è quindi deciso di inserire questa condizione esternamente al modello prendendo i valori necessari nella prima cella e calcolandone il valore per ϵ. Il modello con questa variante è stato denominato *v2fm* e rappresenta il secondo modello disponibile nei confronti del Capitolo 4, mentre i dettagli di come è stata eseguita questa modifica sono riportati nel Capitolo 3.

Proseguendo nella descrizione delle varianti dei modelli, di particolare interesse risultano le formulazioni introdotte da Uribe (UMIST) e Hanjalic (TUD) intorno al 2005 [12, 13], introducendo praticamente gli stessi concetti indipendentemente. Questi modelli rappresentano una via di mezzo tra i consueti $\overline{v'^2} - f$ e il futuro $BL\overline{v'^2}/k$, dal momento che utilizzano al posto della variabile $\overline{v'^2}$, la variabile $\varphi = \overline{v'^2}/k$, e prendono il nome di modelli $\varphi - f$.

Qui di seguito vengono riportate le equazioni del modello $\varphi - f$ definito da Uribe:

$$\begin{cases} \frac{Dk}{Dt} & = P - \epsilon + \partial_j \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \partial_j k \right] \\ \frac{D\epsilon}{Dt} & = \frac{C_{\epsilon 1} P - C_{\epsilon 2} \epsilon}{T} + \partial_j \left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon} \right) \partial_j \epsilon \right] \\ \frac{D\varphi}{Dt} & = \overline{f} - P\frac{\varphi}{k} + \frac{2}{k}\frac{\nu_t}{\sigma_k}\partial_j\varphi\partial_j k + \partial_j \left[\frac{\nu_t}{\sigma_k}\partial_j\varphi \right] \\ L^2\nabla^2\overline{f} - \overline{f} & = \frac{1}{T}\left(C_1 - 1\right)\left[\varphi - \frac{2}{3}\right] - C_2\frac{P}{k} - 2\frac{\nu}{k}\partial_j\varphi\partial_j k - \nu\nabla^2\varphi \end{cases}, \tag{2.9}$$

$$\overline{f} = f + \frac{2\nu\left(\nabla\varphi\nabla k\right)}{k} + \nu\nabla^2\varphi, \qquad \nu_t = C_\mu\varphi k T, \tag{2.10}$$

in cui le costanti sono riportate nella Tabella 2.3, mentre le scale di tempo e spazio turbolente sono definite da:

$$\begin{cases} L & = C_L \max\left[\frac{k^{3/2}}{\epsilon}, C_\eta\frac{\nu^{3/4}}{\epsilon^{1/4}}\right] \\ T & = \max\left[\frac{k}{\epsilon}, C_T\sqrt{\frac{\nu}{\epsilon}}\right] \end{cases}. \tag{2.11}$$

A differenza di quanto visto fino ad ora, le equazioni di φ ed ϵ sono disaccoppiate, mentre è interessante notare che φ sia adimensionale e rappresenta la percentuale di $\overline{v'^2}$ rispetto al totale k e può assumere valori da 0 fino al suo valore isotropico 2/3.

Nell'equazione originale di $\overline{v'^2}$ il termine sorgente era rappresentato da $\epsilon\overline{v'^2}/k$, che è difficile da riprodurre correttamente vicino a parete, dal momento che ϵ tende a crescere, mentre $\overline{v'^2}/k$ tende a zero. Per quanto riguarda invece questa formulazione, il termine sorgente contiene la produzione di energia P che va gradualmente a zero, lasciando solamente il meccanismo di diffusione viscosa come dominante, che risulta stabile e maggiormente predicibile.

Per quanto riguarda le condizioni al contorno di f si ha un miglioramento della stabilità dello schema computazionale, in quanto numeratore e denominatore sono entrambi proporzionali a y^2 e non a y^4 come accadeva nell'originale, essendo:

$$\lim_{y\to 0} f = \frac{-2\nu\left(\overline{v'^2}/k\right)}{y^2}. \tag{2.12}$$

Il modello ideato da Hanjalic invece, non tiene conto dei termini che si ottengono andando a derivare l'equazione di φ da quella di k e $\overline{v'^2}$, essendo trascurabili, e differisce anche per quanto riguarda l'impiego di un modello quasi-lineare per la pressione (SSG) [23] all'interno dell'equazione di f, che garantisce un miglioramento per il flusso di parete in non equilibrio. Queste due differenze influiscono poco sull'andamento complessivo del modello, e quanto appena descritto per il modello di Uribe continua a valere anche per questo modello.

2.2 Billard - Laurence 2012 (BLv2k)

Nonostante gli ottimi risultati raggiunti dalle formulazioni precedenti, i modelli a base $\overline{v'^2}$ (sia $\overline{v'^2} - f$ che $\varphi - f$) soffrono ancora di alcuni problemi che alcune versioni tendono a limitare, esponendosi però ad altri problemi. Infatti un difetto non ancora completamente risolto risiede nella stabilità e accuratezza della condizione al contorno per f, che alcuni modelli come l'ultimo citato di Hanjalic migliorano, ma non ancora definitivamente. Altro problema delle versioni trattate fino ad ora è la sovrastima di $\overline{v'^2}$ o φ nella regione logaritmica, e l'interdipendenza dei coefficienti, cioè il fatto che un coefficiente possa modificare più aspetti del comportamento del modello, rendendo difficoltosa la scelta degli stessi.

Per questi motivi Billard e Laurence [3–6] vedono necessaria la definizione di un nuovo modello le cui equazioni e costanti vengono riportate di seguito e nella Tabella 2.4.

Tabella 2.4: Costanti del modello $BL\overline{v'^2}/k$

$C_{\epsilon 1}$	$C_{\epsilon 2}$	$C_{\epsilon 3}$	$C_{\epsilon 4}$	σ_k	σ_{ϵ_h}	σ_φ	C_μ	C_T	C_L	C_η	C_1	C_2
1.44	1.83	2.3	0.4	1	1.5	1	0.22	4	0.164	75	1.7	0.9

$$\begin{cases} \frac{Dk}{Dt} & = P - \epsilon_h + \partial_j \left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_k} \right) \partial_j k \right] - C_{\epsilon 3} \left(1-\alpha\right)^3 \frac{k}{\epsilon_h} 2\nu\nu_t \left(\partial_k \partial_j U_i\right) \left(\partial_k \partial_j U_i\right) \\ \frac{D\epsilon_h}{Dt} & = \frac{C_{\epsilon 1} P - C^*_{\epsilon 2} \epsilon_h}{T} + \partial_j \left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_{\epsilon_h}} \right) \partial_j \epsilon_h \right] \\ \frac{D\varphi}{Dt} & = \left(1-\alpha^3\right) f_w + \alpha^3 f_h - P\frac{\varphi}{k} + \frac{2}{k}\frac{\nu_t}{\sigma_k} \partial_j \varphi \partial_j k + \partial_j \left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_\varphi} \right) \partial_j \varphi \right] \\ 1 & = \alpha - L^2 \partial_j \partial_j \alpha \end{cases}, \tag{2.13}$$

$$\begin{cases} L & = \sqrt{C_L^2 \left(\frac{k^3}{\epsilon_h^2} \right) + C_\eta^2 \frac{\nu^{3/2}}{\epsilon_h^{1/2}}} \\ T & = \sqrt{\frac{k^2}{\epsilon_h^2} + C_T^2 \frac{\nu}{\epsilon_h}} \\ T_{lim} & = \frac{0.6}{\sqrt{6} C_\mu \varphi \sqrt{S_{ij} S_{ij}}} \end{cases}, \tag{2.14}$$

$$C^*_{\epsilon 2} = C_{\epsilon 2} + \alpha^3 \left(C_{\epsilon 4} - C_{\epsilon 2}\right) \tanh \left(\left| \frac{\partial_j \left(\nu_t / \sigma_k \partial_j k\right)}{\epsilon_h} \right|^{3/2} \right), \tag{2.15}$$

$$f_w = -\frac{\epsilon_h}{2}\frac{\varphi}{k}, \qquad f_h = -\frac{1}{T}\left(C_1 - 1 + C_2 \frac{P}{\epsilon_h} \right)\left(\varphi - \frac{2}{3} \right), \tag{2.16}$$

$$\nu_t = C_\mu \varphi k \min\left(T, T_{lim}\right). \tag{2.17}$$

L'equazione ellittica è sempre presente, ma questa volta è relativa al parametro α, parametro adimensionale compreso tra 0 a parete e tendente a 1 lontano da essa (eliminando così la difficoltà numerica con il limite di parete).

La variabile ϵ invece ha subito una doppia ridefinizione, infatti possiamo definire:

$$\epsilon = \underbrace{\epsilon_h + \frac{k}{\epsilon} E}_{\epsilon'_h} + \frac{1}{2}\nu \partial_j \partial_j k, \tag{2.18}$$

$$E = 2\nu\nu_t \left(\partial_k \partial_j U_i\right)\left(\partial_k \partial_j U_i\right). \tag{2.19}$$

E toglie parte delle non linearità, ed è un modo alternativo per mantenere dipendente dall'anisotropia la costante di produzione di dissipazione ($C^*_{\epsilon 1}$). Normalmente però questo termine introduce anche delle difficoltà numeriche, per questo è stato inserito nell'equazione di k tramite ϵ'_h in modo da poterlo trattare implicitamente. La seconda ridefinizione invece ($\epsilon = \epsilon'_h + 1/2\nu\partial_j\partial_j k$) è stata introdotta con lo scopo

di ridurre la dissipazione di ϵ nell'equazione di k, quando $k \to 0$. In conclusione la condizione al contorno risulta:

$$\lim_{h\to 0} \epsilon_h = \frac{\nu k}{y^2}. \tag{2.20}$$

ϵ_h valutata vicino a parete va ad influenzare anche le scale di lunghezza (L) e tempo (T), quindi possiamo affermare che queste non vengono influenzate da quanto succede lontano da parete, facendo perdere tutti i comportamenti che sono di natura dipendenti da Re_τ. Questo si viene a notare nell'incorretta riproduzione del picco di energia cinetica turbolenta nel buffer-layer, che è notoriamente un fenomeno dipendente da Re_τ.

Capitolo 3

Implementazione

OpenFOAM (Open Field Operation And Manipulation) è un codice CFD libero e gratuito formato da librerie scritte in linguaggio C++, orientato alla soluzione numerica di problemi inerenti alla meccanica del continuo attraverso la discretizzazione ai volumi finiti. La caratteristica di essere un codice libero rende possibile la completa modifica dei sorgenti, e la struttura gerarchica delle librerie combinata con l'elevato livello di astrazione del codice permette di eseguire le modifiche volute con relativa facilità.

Le librerie C++ hanno lo scopo di creare degli eseguibili che prendono il nome di *applicazioni*. Queste applicazioni si suddividono in due categorie: i *solutori* che risolvono i problemi della meccanica del continuo, e le *utilities* che provvedono alla manipolazione dei dati. Il solutore incomprimibile e stazionario utilizzato per questo lavoro è *simpleFoam*, mentre un esempio di utilities sono tutte quelle applicazioni come la generazione della mesh o il sampling dei dati che svolgono le operazioni di pre/post-processing e che verranno descritte in seguito.

3.1 Procedure di discretizzazione

Lo scopo di questo programma è la soluzione di equazioni alla derivate parziali (PDE), che corrisponde ad eseguire operazioni di derivazione ed integrazione su quantità tensoriali nello spazio e nel tempo. E' necessario quindi specificare come i tensori vengono creati e come le derivate vengono discretizzate in equazioni algebriche, e questo viene di seguito eseguito seguendo l'ordine riportato dalla guida programmatori disponibile in OpenFOAM [17].

Il metodo a volumi finiti definisce la discretizzazione in tre parti: discretizzazione *spaziale*, *temporale* e delle *equazioni*. Innanzitutto è necessario creare delle classi di tensori (*Field<Type>*, dove Type identifica il tipo di campo da creare: scalare, vettoriale, ecc.) che vengono rinominate a seconda del tipo come: *scalarField*, *vectorField*, *tensorField*, *symmTensorField*, *tensorThirdField*, *symmTensorThirdField*.

Discretizzazione del dominio: Il dominio di calcolo tipicamente non strutturato (senza un allineamento predeterminato) della mesh viene definito attraverso la classe *polyMesh* (file da definire nell'atto della creazione di una simulazione, come vedremo nel seguito), nella quale vengono definiti i parametri fondamentali come i punti, le facce, le celle e i contorni. Per la discretizzazione a volumi finiti, queste informazioni vengono estese creando la classe *fvMesh*. A questo punto per definire un campo tensoriale riferito alla mesh, si fa riferimento all'entità *geometricField*<*Type*>, ed in particolare ci si riferisce a *volField*<*Type*>, *surfaceField*<*Type*> o *pointField*<*Type*> a seconda che il valore sia riferito rispettivamete al centro, sulla faccia o sul vertice della cella.

Discretizzazione temporale: avviene semplicemente suddividendo il dominio voluto in istanti temporali di dimensione Δt, specificabili all'interno dell'oggetto *controlDict*.

Discretizzazione delle equazioni: corrisponde in una conversione delle equazioni PDE in equazioni algebriche che sono espresse generalmente come : $[A][x] = [b]$, dove $[A]$ rappresenta una matrice quadrata di coefficienti definita tramite la classe *fvMatrix*<*Type*>. Tutti i termini in un'equazione PDE vengono rappresentati individualmente utilizzando le classi *finiteVolumeMethod* (*fvm*) e *finiteVolumeCalculus* (*fvc*), che contengono funzioni che rappresentano gli operatori differenziali che descrivono i *geometricField*<*Type*>. Nello specifico la classe *fvm* restituisce delle classi *fvMatrix*<*Type*> ed esegue derivazioni implicite, mentre *fvc* restituisce classi di tipo *geometricField*<*Type*> eseguendo calcoli espliciti . La discretizzazione ai volumi finiti viene eseguita integrando i termini sul volume di una cella o sulla superficie. Questi integrali vengono poi linearizzati, e l'elenco degli schemi da utilizzare per ogni termine è definita dall'oggetto *fvSchemes* (altro elemento presente nelle cartelle di una simulazione da creare).

I termini sottilineati sono rispettivamente i file o *dizionari* associati ad ogni discretizzazione che devono essere definiti alla creazione di una nuova simulazione, recando appunto le informazioni necessarie, che vengono descritte nei successivi paragrafi.

3.2 Scrittura del modello di turbolenza BLv2k

All'interno della cartella di installazione di OpenFOAM, oltre alle cartelle delle applicazioni, è possibile trovare anche la sotto-cartella *src*, all'interno della quale sono presenti le librerie riguardanti per esempio l'implementazione degli schemi a volumi finiti, degli schemi di soluzione delle equazioni differenziali e l'implementazione dei modelli di turbolenza.

Per accedere ai modelli di turbolenza RANS stazionari incomprimibili presenti in OpenFOAM è sufficiente portarsi in:

../openfoam/src/tubolenceModel/incompressible/RAS/

Qui possiamo trovare molti modelli, come il *kEpsilon*, il *kOmega* e il *v2f*, quest'ultimo disponibile dalla distribuzione 2.1.0. Il nostro scopo è partire dalla versione già implementata del v2f, per apportarne le modifiche evidenziate nel capitolo precedente al fine di ottenere il modello BLv2k. Aprendo la cartella del modello troviamo l'header (.H) e il file sorgente (.C) che definiscono l'oggetto v2f. All'interno del primo vi troviamo i prototipi delle funzioni e la dichiarazione delle variabili, mentre nel secondo troviamo la creazione delle funzioni stesse.

Per creare il nuovo modello è conveniente copiare e incollare all'interno della cartella RAS la cartella del modello v2f, e rinominarla BLv2k. Rinominati anche i file all'interno e sostituito il nome dell'oggetto v2f con BLv2k all'interno degli stessi file, abbiamo un nuovo v2f con il nome cambiato, che ora dobbiamo modificare.

Come prima cosa andiamo ad aggiornare all'interno dell'header la dichiarazione delle variabili, il che significa sia le costanti, che gli oggetti *volScalarField* che rappresentano le variabili del modello: da *k_* , *epsilon_* , *v2_* , *f_* si passa a *k_* , *epsilon_* , *z_* , *alpha_* (Nella definizione del modello data nei capitoli precedenti si è utilizzata la variabile φ, ma all'interno dei sorgenti si è preferito utilizzare z, per evitare confusione con la *phi* già presente). Proseguendo si aggiunge la definizione della funzione che restituisce la *Tlim()* e si aggiornano le funzioni che ritornano il valore della diffusività effettiva e delle variabili del modello.

Per quanto riguarda il file .C, all'interno della definizione delle funzioni private, è necessario modificare le funzioni *T()*, *L()*, *Tlim()*, e all'interno del costruttore dell'oggetto aggiornare le costanti e le variabili del modello. Fatto questo, si arriva alla definizione delle equazioni della turbolenza:

Equazione per k:

$$\frac{Dk}{Dt} = P - \epsilon_h + \partial_j \left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_k} \right) \partial_j k \right] - C_{\epsilon 3} \left(1 - \alpha\right)^3 \frac{k}{\epsilon_h} 2\nu\nu_t \left(\partial_k \partial_j U_i\right) \left(\partial_k \partial_j U_i\right), \tag{3.1}$$

che viene tradotto nel seguente codice:

```
tmp<fvScalarMatrix> kEqn
(       fvm::ddt(k_)
      + fvm::div(phi_, k_)
      - fvm::laplacian(DkEff(), k_)
     ==
      G
```

```
      - fvm::Sp(epsilon_/k_,k_)
      - fvm::Sp(Ceps3_*pow((1.0-alpha_),3.0)*E/epsilon_, k_)
);
```

Equazione per ϵ:

$$\frac{D\epsilon_h}{Dt} = \frac{C_{\epsilon 1}P - C^*_{\epsilon 2}\epsilon_h}{T} + \partial_j\left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_{\epsilon_h}}\right)\partial_j\epsilon_h\right], \tag{3.2}$$

che viene tradotto nel seguente codice:

```
tmp<fvScalarMatrix> epsEqn
(       fvm::ddt(epsilon_)
      + fvm::div(phi_, epsilon_)
      - fvm::laplacian(DepsilonEff(), epsilon_)
     ==
        fvm::Sp(Ceps1_*G/k_,epsilon_)
      - fvm::Sp(Ceps2*epsilon_/k_, epsilon_)
);
```

Equazione per φ:

$$\frac{D\varphi}{Dt} = \left(1-\alpha^3\right)f_w + \alpha^3 f_h - P\frac{\varphi}{k} + \frac{2}{k}\frac{\nu_t}{\sigma_k}\partial_j\varphi\partial_j k + \partial_j\left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_\varphi}\right)\partial_j\varphi\right], \tag{3.3}$$

che viene tradotto nel seguente codice:

```
tmp<fvScalarMatrix> zEqn
(       fvm::ddt(z_)
      + fvm::div(phi_, z_)
      - fvm::laplacian(DzEff(), z_)
     ==
        (1.0-pow(alpha_,3.0))*fw
      + pow(alpha_,3.0)*fh
      - fvm::Sp(G/k_, z_)
      + 2.0/k_*nut_/sigmaK_*(fvc::grad(z_) & fvc::grad(k_))
);
```

Equazione per α:

$$1 = \alpha - L^2\partial_j\partial_j\alpha, \tag{3.4}$$

che viene tradotto nel seguente codice:

```
tmp<fvScalarMatrix> alphaEqn
```

```
(     - fvm::laplacian(alpha_)
      - 1.0/sqr(L_)
     ==
      - fvm::Sp( 1.0/sqr(L_), alpha_ )
);
```

dove si è definito:

$$E = 2\nu\nu_t \left(\partial_k \partial_j U_i\right) \left(\partial_k \partial_j U_i\right), \tag{3.5}$$

```
const volScalarField E = 2.0*nu()*nut_*fvc::magSqrGradGrad(U_);
```

$$G = 2\nu_t S_{ij} S_{ij}, \tag{3.6}$$

```
const volScalarField G = nut_*2.0*magSqr(symm(fvc::grad(U_)));
```

$$C^*_{\epsilon 2} = C_{\epsilon 2} + \alpha^3 \left(C_{\epsilon 4} - C_{\epsilon 2}\right) \tanh\left(\left|\frac{\partial_j \left(\nu_t/\sigma_k \partial_j k\right)}{\epsilon_h}\right|^{3/2}\right), \tag{3.7}$$

```
const volScalarField Ceps2 = Ceps20_ + pow(alpha_,3)*(Ceps4_-Ceps20_)*...
      ...tanh(pow(mag(fvc::div(nut_/sigmaK_*fvc::grad(k_))/epsilon_),1.5));
```

$$f_w = -\frac{\epsilon_h}{2}\frac{\varphi}{k} \quad ; \quad f_h = -\frac{1}{T}\left(C_1 - 1 + C_2 \frac{P}{\epsilon_h}\right)\left(\varphi - \frac{2}{3}\right), \tag{3.8}$$

```
const volScalarField fw = -epsilon_/2.0*z_/k_;
const volScalarField fh = -1.0/T_*(C1_-1.0+C2_*G/epsilon_) * (z_-2.0/3.0);
```

Come si nota dalle righe di codice sopra riportate, il fatto di avere un linguaggio astratto di questo tipo ci permette di modificare sostanzialmente il codice senza avere la necessità di preoccuparci di come i volumi finiti vadano poi ad operare, o di come sia necessario spezzare il lavoro per renderlo parallelizzabile.

A questo punto abbiamo completato il grosso della modifica del modello, e possiamo far compilare a *wmake* le librerie in modo che vengano creati gli eseguibili necessari al funzionamento del modello. Per fare questo è necessario aggiungere la riga *BLv2k/Blv2k.C* assieme agli altri modelli all'interno del file *Make/files* e da terminale, portandosi nella cartella *../RAS/*, lanciare il comando *wmake libso*, che provvederà alla compilazione fornendo brevi spiegazioni di eventuali errori.

3.2.1 Correzione del modello per simulazioni Low-Re

Per il corretto funzionamento del modello per simulazioni Low-Reynolds sono necessarie anche le corrette condizioni al contorno. Queste normalmente vengono assegnate all'interno della cartella della simulazione, ma sia per il v2f che per il BLv2k, la condizione al contorno per la ϵ dipende dai valori a parete di k, ν e della posizione normale alla parete del centro cella. Nello specifico, abbiamo:

$$v2f : \lim_{y\to 0} \epsilon = 2\nu k/y^2, \qquad BLv2k : \lim_{y\to 0} \epsilon = \nu k/y^2. \tag{3.9}$$

In questo modo si è pensato di correggere direttamente all'interno del modello il valore di ϵ, facendolo calcolare istante per istante.

```
const fvPatchList& patches = mesh_.boundary();
double ni;
double epsilon_h;

forAll(patches, patchi)
  { const fvPatch& curPatch = patches[patchi];
    if (isA<wallFvPatch>(curPatch))
      { forAll(curPatch, facei)
          { label faceCelli = curPatch.faceCells()[facei];
            scalar y1 = y()[patchi][facei];
            ni = nu()().internalField()[faceCelli];
            scalar k1 = k_.internalField()[faceCelli];
            epsilon_h = ni*k1/sqr(y1);
            epsilon_[faceCelli] = epsilon_h;
          }
      }
  }
```

Come si nota, viene definito l'oggetto patches (contenente tutte le patch del bordo della mesh), ed eseguendo un ciclo su tutte, è possibile salvarle una alla volta nell'oggetto curPatch. Tramite l'*if* si verifica che siano delle pareti e si opera su ognuna singolarmente attraverso il secondo ciclo. A questo punto possiamo ricavare i valori necessari al calcolo di ϵ e aggiornare il valore della variabile del modello.

Questo codice viene salvato in un file chiamato *SetEpsWall.H* e messo nella cartella del BLv2k. Per essere richiamato è necessario inserire la riga di codice *#include"SetEpsWall.H"* appena prima della definizione dell'equazione dei modelli di turbolenza nel file .C del modello. Questa correzione forzata della condizione sulla ϵ non è tuttavia presente nella formulazione del v2f di OpenFOAM, come si nota osservando il comportamento della variabile stessa vicino a parete nella Figura 3.1. A seguito di alcune prove eseguite su lastra piana con integrazione a parete, si è notato un migliore comportamento della ϵ a seguito della modifica, e si è quindi deciso di mantenere entrambe le versioni per la parte di validazione dei modelli, identificando la versione modificata con il nome di *v2fm* (si sottolinea che il file della correzione per il v2f risulta identico a quello del BLv2k ad eccezione della moltiplicazione per 2 nella formula del calcolo di ϵ).

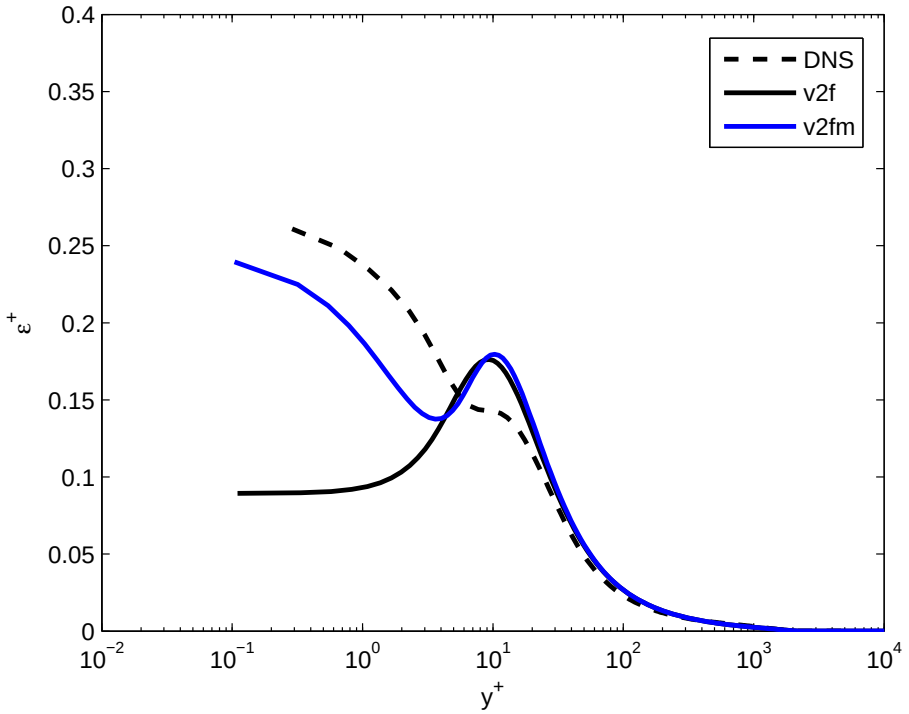

Figura 3.1: Comportamento della ϵ^+ per v2f e v2fm

Tabella 3.1: Struttura dell'n-esima riga della tabella per simulazione High-Reynolds per BLv2k

$Re^+_{y,n} = U^+_n y^+_n$	y^+_n	U^+_n	k^+_n	ϵ^+_n	z^+_n	α^+_n

3.2.2 Correzione del modello per simulazioni High-Re

Per i casi High-Reynolds si ricorre alle wall-function adattive. Queste presuppongono la creazione di una tabella dalla simulazione Low-Reynolds, recante i valori di tutte le variabili del modello adimensionalizzate in unità di parete, ordinati per il numero di Reynolds calcolato con la distanza normale a parete. La struttura della tabella utilizzata in questo documento è quindi la seguente:

```
149 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
1.168e-02 1.080e-01 1.080e-01 4.591e-03 2.389e-01 9.113e-05 4.621e-03
1.090e-01 3.302e-01 3.301e-01 3.697e-02 1.256e-01 5.061e-04 1.396e-02
3.187e-01 5.646e-01 5.644e-01 8.457e-02 8.968e-02 1.035e-03 2.370e-02
```

in cui una generica riga della tabella è riportata nella Tabella 3.1, mentre la prima riga contiene tutti zeri ad eccezione del primo elemento, che rappresenta il numero di righe totali della tabella.

Avendo a disposizione questa tabella, la modifica al modello di turbolenza si suddivide in tre parti assegnate a tre differenti file:

ReadWFTable.H: questo file legge i valori della tabella salvati nel file *WF_ TABLE.txt* e posizionato nella cartella della simulazione *../constant/polyMesh/* realizzata dal caso High-Reynolds. Successivamente assegna questi valori a delle variabili, che devono essere definite all'interno dell'header del modello, in modo che le suddette siano disponibili per i file successivi.

```
int i=0;
FILE *fTable=0;
fTable = fopen("constant/polyMesh/WF_TABLE.txt","r");
fscanf(fTable,"%d %le %le %le %le %le\n",...
      ...&N_tbl,&y_plus,&u_plus,&k_plus,&epsilon_plus,...
      ...&z_plus,&alpha_plus);
RE_plus = (double*) malloc(N_tbl * sizeof(double));
Y_plus = (double *) malloc(N_tbl * sizeof(double));
U_plus = (double *) malloc(N_tbl * sizeof(double));
K_plus = (double *) malloc(N_tbl * sizeof(double));
EPSILON_plus = (double *) malloc(N_tbl * sizeof(double));
Z_plus = (double *) malloc(N_tbl * sizeof(double));
ALPHA_plus = (double *) malloc(N_tbl * sizeof(double));
//Reading Table
for(i=0; i< N_tbl; i++)
{ fscanf(fTable,"%le %le %le %le %le %le\n",...
      ...&RE_plus[i],&Y_plus[i],&U_plus[i],&K_plus[i],...
      ...&EPSILON_plus[i],&Z_plus[i],&ALPHA_plus[i]);
}
fclose(fTable);
```

InterWallFunc.H: esegue la ricerca all'interno dei valori della tabella salvati dal precedente file e li assegna nel centro della prima cella. Dovendo assegnare il valore delle variabili nel primo centro cella vicino a parete, viene eseguito il medesimo procedimento descritto per la correzione della condizione al contorno di ϵ alla Sezione 3.2.1, soltanto che questa volta, all'interno del secondo ciclo abbiamo:

```
label faceCelli = curPatch.faceCells()[facei];
vector n=NN[facei];
n = n / mag(n);
vector p = U_.internalField()[faceCelli];
p = p - (p & n)*p;
scalar y1 = y()[patchi][facei];
ni = nu()().internalField()[faceCelli];
```

```
scalar u1y1 = mag(p)*y1/ni;

//Table interpolation
i=0;
while ((RE_plus[i]<u1y1) && (i+1<=N_tbl-1))
  { i=i+1;
  }
if (u1y1 >= RE_plus[N_tbl-1])
  { y_plus=Y_plus[N_tbl-1];
    u_plus=U_plus[N_tbl-1];
    k_plus=K_plus[N_tbl-1];
    epsilon_plus=EPSILON_plus[N_tbl-1];
    z_plus=Z_plus[N_tbl-1];
    alpha_plus=ALPHA_plus[N_tbl-1];
  }
else
  { y_plus=exp((log(Y_plus[i])-log(Y_plus[i-1]))*...
           ...(log(u1y1)-log(RE_plus[i-1]))/...
           ...(log(RE_plus[i])-log(RE_plus[i-1]))+...
           ...log(Y_plus[i-1]));
    u_plus=exp((log(U_plus[i])-log(U_plus[i-1]))*...
           ...(log(u1y1)-log(RE_plus[i-1]))/...
           ...(log(RE_plus[i])-log(RE_plus[i-1]))+...
           ...log(U_plus[i-1]));
    k_plus=...
    epsilon_plus=...
    z_plus=...
    alpha_plus=...
  }

//u_tau Computing
double u_tau = ni*y_plus/y1;
double u_tau2 = pow(u_tau,2);

//Boundary Condition Changing: first cell center
z_[faceCelli] = z_plus;
alpha_[faceCelli] = alpha_plus;
k_[faceCelli] = u_tau2*k_plus;
epsilon_[faceCelli] = pow(u_tau,4)/ni*epsilon_plus;
```

in cui si leggono i valori della velocità parallela alla parete, della viscosità e della posizione verticale del primo centro cella. Grazie a questi è possibile calcolare il valore del numero di Reynolds in quel punto (u1y1) ed interpolare le variabili da assegnare. Si sottolinea che viene eseguita una interpolazione lineare sui logaritmi, in quanto l'andamento delle variabili risulta più costante, quindi l'interpolazione più precisa.

SetWallNut.H: va a correggere istante per istante il valore della viscosità turbolenta del modello sempre nel primo centro cella, con le variabili in unità di parete disponibili dalla tabella. Questo è necessario perchè è stato saltato il sub-strato viscoso, quindi lo sforzo calcolato a parete risulterebbe differente da quello reale. La correzione avviene tramite la seguente formula:

$$\nu_t = \nu(y^+/u^+ - 1), \tag{3.10}$$

come descritto da Kalitzin [14]. In questo modo il valore di viscosità turbolenta calcolato non è il valore realmente esistente in quella posizione, ma viene calcolato un valore di viscosità tale per cui si ottiene il giusto sforzo a parete, con il differente gradiente che risulta dall'aver posto il primo centro cella molto più in alto.
Ancora una volta la procedura risulta uguale a quanto fatto per l'InterWallFunc.H e per il SetEpsWall.H e quello che cambia è all'interno del secondo ciclo, una volta selezionato le patch di parete. In particolare la prima parte risulta del tutto uguale a quella già mostrata per il file precedente, solamente che in questo caso è sufficiente interpolare solamente la y^+ e la u^+.

```
label faceCelli = curPatch.faceCells()[facei];
vector n=NN[facei];
n = n / mag(n);
vector p = U_.internalField()[faceCelli];
p = p - (p & n)*p;
scalar y1 = y()[patchi][facei];
ni = nu()().internalField()[faceCelli];
scalar u1y1 = mag(p)*y1/ni;

//Table interpolation
i=0;
while ((RE_plus[i]<u1y1) && (i+1<=N_tbl-1))
  { i=i+1;
  }
if (u1y1 >= RE_plus[N_tbl-1])
```

```
  { y_plus=Y_plus[N_tbl-1];
    u_plus=U_plus[N_tbl-1];
  }
else
  { y_plus=exp((log(Y_plus[i])-log(Y_plus[i-1]))*...
        ...(log(u1y1)-log(RE_plus[i-1]))/...
        ...(log(RE_plus[i])-log(RE_plus[i-1]))+...
        ...log(Y_plus[i-1]));
    u_plus=exp((log(U_plus[i])-log(U_plus[i-1]))*...
        ...(log(u1y1)-log(RE_plus[i-1]))/...
        ...(log(RE_plus[i])-log(RE_plus[i-1]))+...
        ...log(U_plus[i-1]));
  }
//Changing Nut
scalarField& nutw = nut_.boundaryField()[patchi];
nutw[facei] = ni*(y_plus/u_plus - 1);
```

Il primo di questi file va richiamato all'interno del corpo dell'oggetto del modello in questione, il secondo appena prima della definizione delle equazioni turbolente, mentre l'ultimo va inserito subito dopo ogni valutazione della viscosità turbolenta del modello.

3.3 Creazione di una simulazione

La creazione di un caso di simulazione consiste nella definizione dei file contenenti le informazioni necessarie agli eseguibili di OpenFOAM di procede nella simulazione. Questi file devono rispettare la corretta sintassi e devono anche essere organizzati in cartelle. Nella Figura 3.2 viene riportata la struttura delle cartelle di una simulazione necessaria al funzionamento del modello BLv2k, che può essere ottenuta partendo da quelle presenti nei tutorial di OpenFOAM per i problemi incomprimibili con solutore simpleFoam.

Di seguito vengono riportate le informazioni basilari che descrivono i file sopra riportati, e si ricorda che maggiori dettagli sulle possibilità di definizine di questi dizionari e sulle varianti degli schemi presenti, sono disponibili nella UserGuide [18] e nella programmersGuide [17] di OpenFOAM.

3.3.1 constant

Come riportato dal diagramma della Figura 3.2, in questa cartella sono presenti dei file per la caratterizzazione di alcune proprietà:

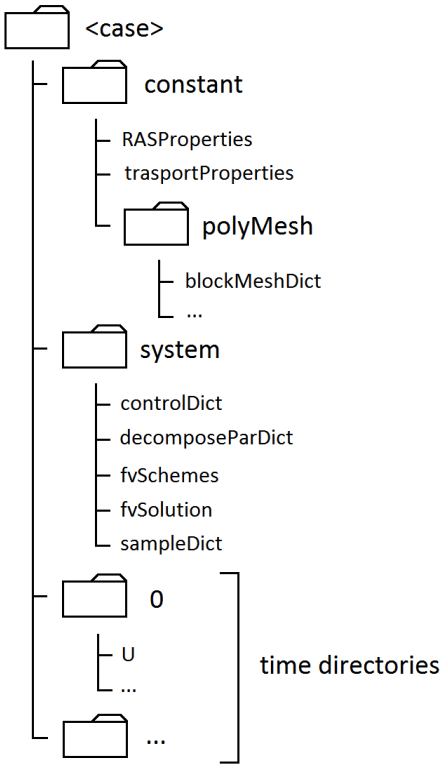

Figura 3.2: Gerarchia delle cartelle di una simulazione

RASProperties: consiste nello specificare il modello di turbolenza voluto per la simulazione;

transportProperties: specifica le proprietà del fluido in esame.

Oltre a questi file è presente anche la cartella *poliMesh* nella quale sono riportati tutti i file necessari a caratterizzare il dominio di calcolo e la mesh. Nello specifico questa descrizione è assegnata solamente al file di seguito descritto, mentre gli altri file vengono ricavati automaticamente da OpenFOAM quando si compila la mesh.

blockMeshDict: prima vengono definiti i vertici della mesh, successivamente si definisce l'insieme di punti che va a comporre un blocco, specificando in che modo infittire le celle (grading). Per ultimo vengono definite le *patches*, cioè le superfici, che saranno fondamentali nel momento di imporre le condizioni al contorno.

```
convertToMeters 1.0;
vertices
( (0.0000 0.0000 0.0000)
  (0.5000 0.0000 0.0000)
  (5.5000 0.0000 0.0000)
```

```
  (0.0000 0.1500 0.0000)
  (0.5000 0.1500 0.0000)
  (5.5000 0.1500 0.0000)
  (0.0000 0.0000 0.1000)
  (0.5000 0.0000 0.1000)
  (5.5000 0.0000 0.1000)
  (0.0000 0.1500 0.1000)
  (0.5000 0.1500 0.1000)
  (5.5000 0.1500 0.1000)
);
blocks
( hex (0 1 4 3 6 7 10 9) (150 150 1) simpleGrading (0.1000 3000.0000 1)
  hex (1 2 5 4 7 8 11 10) (1500 150 1) simpleGrading (16.0000 3000.0000 1)
);
patches
( patch inlet
   (  (0 6 9 3) )
  patch outlet
   (  (2 5 11 8) )
  patch top_us
   (  (9 10 4 3) )
  patch top
   (  (10 11 5 4) )
  patch bottom_us
   (  (0 1 7 6) )
  wall bottom_wall
   (  (1 2 8 7) )
  empty side_left
   ( (0 3 4 1)
     (1 4 5 2) )
  empty side_right
   ( (7 10 9 6)
     (8 11 10 7) )
);
mergePatchPairs ( );
```

L'esempio riportato si riferisce ad un lastra piana, che rappresenta il dominio utilizzato in questa tesi per la fase di verifica dei risultati. Data la semplicità di definizione del dominio utilizzato in questa esperienza è stato possibile definire direttamente la mesh dal file, altrimenti, per geometrie più complicate, è pos-

sibile avvalersi di software appositi per la generazione della mesh ed esportare da questi le informazioni nei file richiesti da OpenFOAM.

3.3.2 system

All'interno della cartella system troviamo 5 file:

controlDict: presenta le principali e basilari informazioni per la simulazione, come il solutore voluto, l'istante temporale di inizio (che tipicamente è 0, e per il quale dovremo definire le condizioni iniziali), il passo temporale (per il nostro caso sarà unitario essendo stazionari), l'istante finale (a cui si arresta la simulazione se non arriva prima a convergenza) e l'intervallo di scrittura su file della simulazione.

```
application     simpleFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         30000;
deltaT          1;
writeControl    timeStep;
writeInterval   1000;
purgeWrite      40;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
```

decomposeParDict: serve esclusivamente se si vuole effettuare la simulazione in parallelo, e reca il numero di processori a disposizione, il metodo di decomposizione e come suddividere il dominio.

```
numberOfSubdomains 4;
method hierarchical;
simpleCoeffs
{ n ( 2 1 1 );
  delta 0.001;
}
hierarchicalCoeffs
{ n ( 4 1 1 );
```

```
    delta 0.001;
    order xyz;
}
metisCoeffs
{ }
manualCoeffs
{ dataFile "";
}
distributed no;
roots ( );
```

fvSchemes: già accennato in precedenza, questo dizionario, ci permette di definire gli schemi di discretizzazione di ogni operatore matematico utilizzato.

```
ddtSchemes
{ default steadyState;
}
gradSchemes
{ default Gauss linear;
}
divSchemes
{ default Gauss linear;
}
laplacianSchemes
{ default Gauss linear corrected;
}
interpolationSchemes
{ default linear;
}
snGradSchemes
{ default corrected;
}
fluxRequired
{ default no;
  p;
}
```

fvSolution: serve per impostare i solutori delle equazioni, le tolleranze e altri algoritmi.

```
solvers
```

```
{ p
  { solver PCG;
    preconditioner DIC;
    tolerance 1e-15;
    relTol 0.01;
  }
  U
  { solver PBiCG;
    preconditioner DILU;
    tolerance 1e-15;
    relTol 0.01;
  }
  k ...
  epsilon ...
  z ...
  alpha ...
  R ...
  nuTilda ...
}
SIMPLE
{ nNonOrthogonalCorrectors 0;
  residualControl
   { p                     1e-5;
     U                     1e-5;
     "(k|epsilon|omega)"   1e-5;
   }
}
relaxationFactors
{ fields
  { p            0.3; }
  equations
  { U            0.4;
    k            0.4;
    epsilon      0.4;
    z            0.4;
    alpha        0.4;
    R            0.4;
    nuTilda      0.4;
  }
}
```

sampleDict: serve per la fase di post-processing, in cui è possibile specificare le variabili da estrarre e la posizione. Si sottolinea che le coordinate richieste da OpenFOAM sono quelle assolute e quindi l'asse delle x ha lo zero posto non all'inizio della lastra piana, ma all'inizio del dominio, quindi bisogna tenere conto di eventuali zone iniziali, come vedremo nel nostro caso.

```
setFormat raw;
surfaceFormat vtk;
interpolationScheme cell;
fields
(    p
     U
     k
     epsilon
     z
     alpha
);
sets
(    profile1
     { type midPoint;
       axis xyz;
       start (2.264 0 0.05);
       end   (2.264 0.15 0.05);
     }
     profile2 ...
       .
       .
       .
);
surfaces ( );
```

3.3.3 time directories

Con questo nome si identificano tutte le cartelle create dal programma durante la simulazione, assieme alla cartella di condizioni iniziali che tipicamente è 0. All'interno di questa cartella devono essere presenti un file per ogni variabile del modello di cui è necessario definire le condizioni al contorno. Nel nostro caso avremo i file per: k, *epsilon*, z, *alpha*, *nut*, *nuTilda*, p ed U. La struttura di questi file è la medesima e consiste nell'assegnare ad ogni patch definita dalla mesh il tipo di condizione al contorno ed il valore se necessario. A titolo di esempio si riporta la struttura del file per la U:

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (33 0 0);
boundaryField {
 inlet
  {   type            fixedValue;
      value           uniform (33 0 0);
  }
 outlet
  {   type            zeroGradient;
  }
 top_us
  {   type            zeroGradient;
  }
 top
  {   type            zeroGradient;
  }
 bottom_us
  {   type            slip;
  }
 bottom_wall
  {   type            fixedValue;
      value           uniform (0 0 0);
  }
 side_left
  {   type            empty;
  }
 side_right
  {   type            empty;
  }
}
```

Una volta definiti tutti i file è possibile lanciare la simulazione, portandosi da terminale all'interno della cartella della simulazione e lanciare i seguenti comandi:

blockMesh: compila il file *blockMeshDict* generando la mesh tramite la creazione dei file sopra citati;

decomposePar: crea le cartelle *processor0*, *processor1...* per la simulazioe in parallelo (nel caso si volesse lanciare la simulzione non in parallelo, questo comando si può evitare);

mpirun -np 4 simpleFoam -parallel: avvia la simulazione in parallelo (altrimenti è sufficiente il comando *simpleFoam*), dove 4 rappresenta il numero dei processori a disposizione;

reconstructPar: ricostruisce le cartelle degli istanti temporali a partire dalle informazioni memorizzate nelle cartelle *processor*, una volta terminata la simulazione (anche questo comando è da utilizzare solo in caso di simulazione in parallelo);

sample: primo comando di post-processing che estrapola i dati voluti secondo quanto specificato nel file apposito e li salva per tutti gli istanti temporali nella cartella *postProcessing*;

wallShearStress: altro comando di post-processing tipicamente utilizzato, che esegue il calcolo dello sforzo a parete e lo memorizza in un file che viene salvato all'interno di tutte le cartelle degli istanti temporali.

Capitolo 4

Risultati

Per la validazione dei modelli v2f, v2fm, BLv2k presi in considerazioni in questo lavoro, si è cominciato con la creazione di un caso, che sarà comune a tutte le simulazioni. La scelta della geometria è andata nella direzione dell'esperimento svolto da Wieghardt [25], mentre per le condizioni di prova si è fatto riferimento all'articolo di Bosh e Rodi [7], con un rapporto tra viscosità turbolenta e cinematica ($r_\mu = \nu_t/\nu = 100$) e un livello di turbolenza ($T_u = \sqrt{\overline{u'^2}} = 0.02$). In particolare la geometria bidimensionale utilizzata è schematizzata nella Figura 4.1.

Consiste di una lastra piana lunga 5 metri, preceduta da 0.5 metri di aria in modo da far stabilizzare il flusso dalle condizioni al contorno, prima di incontrare la lastra piana. A differenza di quanto riportato nella figura, lo spessore della lamina è nullo, in modo da eliminare gli effetti dovuti alla deviazione della corrente e considerare solamente quelli viscosi. Questo è realizzabile imponendo sulla superficie occupata della lastra piana le corrette condizioni al contorno e iniziali di velocità nulla, mentre gli altri valori sono riportati nella Tabella 4.3. L'altezza del dominio è di 0.15 metri e la mesh costruita su questa geometria è stata suddivisa in due zone:

- *1° zona*: consiste nella zona anteriore alla lamina dove esiste sola aria;
- *2° zona*: inizia dal bordo d'attacco della lastra piana, fino alla fine.

Nella mesh per simulazioni Low-Reynolds la prima zona è stata suddivisa con 150 celle sia orizzontalmente che verticalmente, mentre la seconda zona ha 1500 e 150

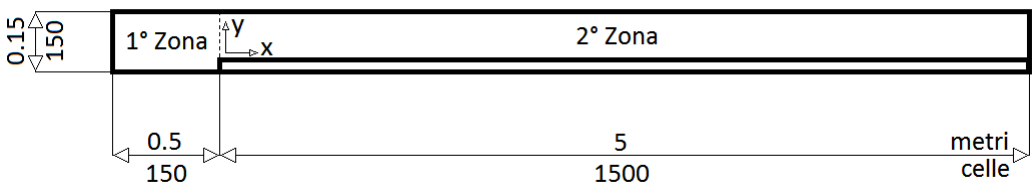

Figura 4.1: Geometria del dominio di calcolo

Tabella 4.1: Caratteristiche della mesh per simulazioni Low/High-Reynolds

	n° celle				grading			
	1° parte		2°parte		1° parte		2°parte	
y_1^+	x	y	x	y	x	y	x	y
0.1	150	150	1500	150	0.1	3000	16	3000
10	150	25	1500	25	0.1	100	16	100
50	150	20	1500	20	0.1	16	16	16
100	150	20	1500	20	0.1	6	16	6

celle rispettivamente per la parte orizzontale e verticale. Per arrivare a valori molto piccoli di y^+, come richiesto dal problema, è stato necessario l'uso di un grading (rapporto tra la dimensione dell'ultima cella e la prima in una direzione) abbastanza elevato. Orizzontalmente si è scelto 0.1 nella prima zona di dominio in modo da avere le celle sempre più piccole man mano che ci si avvicina al bordo d'attacco della lastra e 16 per lo stesso motivo nella seconda zona. La scelta corretta di questi termini va fatta tenendo anche in considerazione che deve preferibilmente mantenersi la continuità tra la dimensione delle celle tra una zona e l'adiacente. Verticalmente è stato invece utilizzato un grading di 3000 per entrambe le zone, che porta ad avere una $y^+ \approx 0.1$.

Nonostante questa simulazione sia bidimensionale, è necessario introdurre anche una profondità del dominio, di qualsiasi dimensione, ma con una sola cella, in modo da vincolare OpenFOAM a eseguire i calcoli solamente in quella coordinata di profondità. Per maggiore chiarezza il file blockMeshDict (che genera la geometria e la mesh), riportato nella Sezione 3.3.1, si riferisce proprio a questa geometria.

Per quanto riguarda invece le simulazioni High-Reynolds si sono utilizzati differenti numeri di celle e differenti valori di grading, che sono riassunti nella Tabella 4.1.

Oltre alle caratteristiche di geometria e mesh, è importante controllare anche i parametri dichiarati nei file fvSchemes e fvSolution presenti nella cartella system. Per motivi di coerenza sono stati utilizzati gli stessi parametri e gli stessi schemi per tutti e tre i modelli, sia per le simulazioni Low-Reynolds che High-Reynolds. La scelta effettuata è possibile visionarla nei file riportati nella Sezione 3.3.2.

Rimangono da assegnare solamente le condizioni iniziali e al contorno, tramite i file contenuti nella cartella 0 del caso. Infatti, come già accennato, all'interno possiamo trovare un file per ogni variabile del modello, all'interno del quale possiamo specificare le condizioni per ogni patch della mesh. Le caratteristiche dell'aria per queste simulazioni sono state ricavate dalle prove di Wieghardt, come riportato nella Tabella 4.2, mentre le condizioni iniziali e al contorno sono riportate nella Tabella 4.3.

Tabella 4.2: Caratteristiche del fluido di lavoro

Velocità asintotica	$U = 33m/s$
Viscosità cinematica	$\nu = 1.51 \times 10^{-5} m^2/s$
Densità	$\rho = 1Kg/m^3$

Tabella 4.3: Condizioni iniziali ed al contorno assegnate

Variabile	Condizione internal field	Condizione inlet	Condizione a parete
k	$(r_\mu U)^2$	$(r_\mu U)^2$	1×10^{-11}
ϵ	$\frac{2}{3}\frac{C_\mu}{\nu}(r_\mu U)^4$	$\frac{2}{3}\frac{C_\mu}{\nu}(r_\mu U)^4$	zeroGradient
$\overline{v'^2}$	$\frac{2}{3}(r_\mu U)^2$	$\frac{2}{3}(r_\mu U)^2$	1×10^{-11}
φ	$\frac{2}{3}$	$\frac{2}{3}$	1×10^{-11}
f	1×10^{-11}	zeroGradient	1×10^{-11}
α	1	1	1×10^{-11}

4.1 Simulazioni Low-Re

In questa sezione vengono riportati i risultati ottenuti, cioè vengono riportate le figure relative alle variabili dei modelli confrontati, con riferimenti DNS e sperimentali. Le variabili sono tutte adimensionalizzate in unità di parete e riportate in relazione alla coordinata di parete anch'essa adimensionalizzata. Invece il grafico del cf viene sempre riportato in funzione del numero di Reynolds calcolato con la coordinata x (distanza del punto considerato dal bordo d'attacco della lamina piana: $Re_x = \frac{Ux}{\nu}$).

4.1.1 Confronto dei riferimenti

Prima di tutto vengono confrontati i riferimenti che si intenderanno utilizzare nel proseguo del documento. In particolare si fa riferimento ai set sperimentali di Wieghardt [25] e Osterlund [19] e un set di dati DNS realizzati da Sillero, Jimenez, Moser nel 2011 [22]. Per quanto riguarda i set sperimentali, il primo è più datato (1951), ma ancora molto valido e sicuramente più comune, mentre il secondo è più recente (1999) e più completo, sia in termini di campionamenti per ogni condizione di misura, che per numero di condizioni di misure. Vengono quindi riportati gli andamenti del profilo di velocità con le rispettive leggi analitiche e del c_f con la legge analitica di Blasius e quella semiempirica di Prantl - Von Karman che qui sono riportate:

$$c_{f,Blasius} = \frac{0.664}{\nu\sqrt{x}}, \qquad c_{f,Prantl-Von\,Karman} = \frac{2\,(0.41)^2}{\log^2(3.5 c_f Re_x)}.$$

La figura del profilo di velocità vede quindi il confronto dei due set sperimentali e della DNS a $Re_\theta \approx 5000$. Si preferisce utilizzare il numero di Reynolds riferito allo

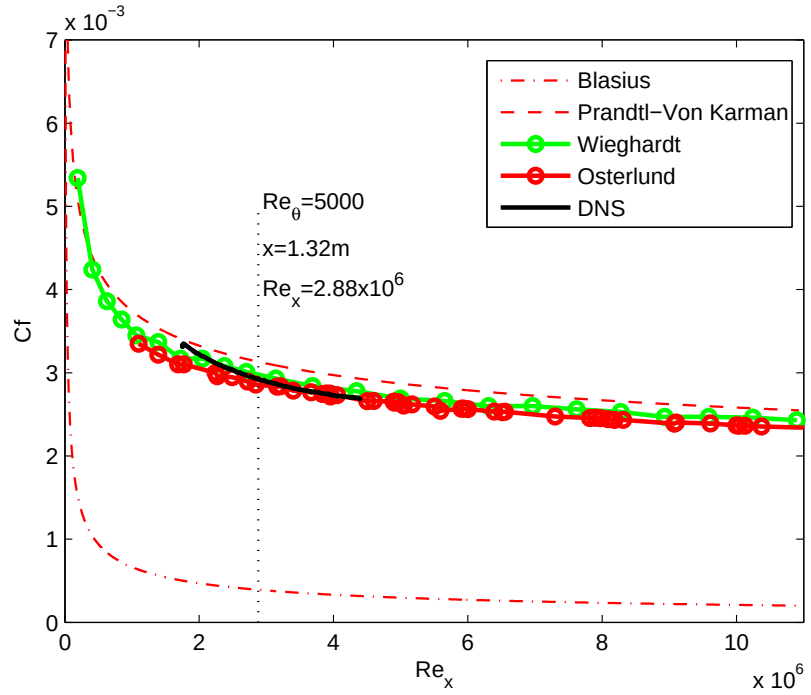

Figura 4.2: Andamento del cf dei riferimenti.

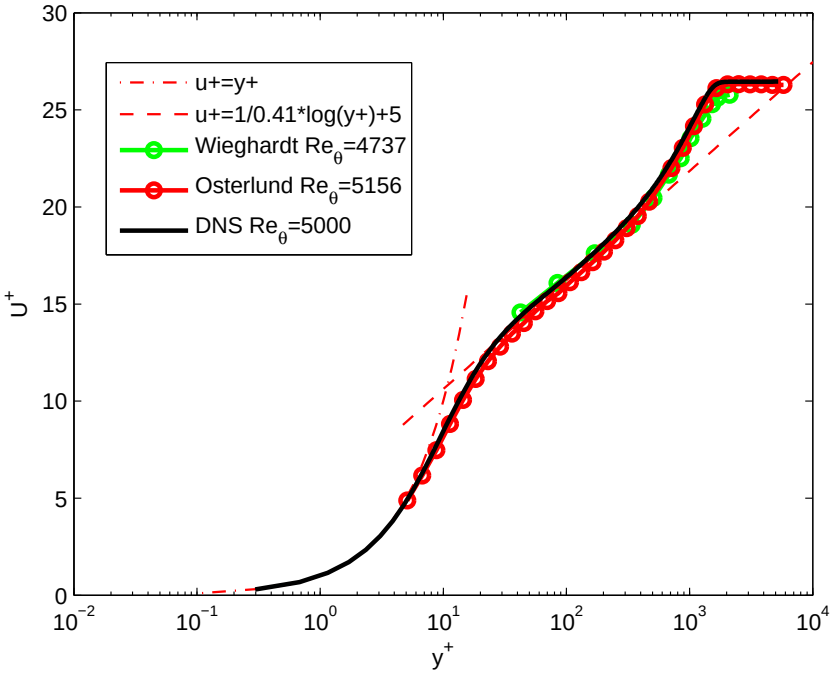

Figura 4.3: Andamento di U^+ dei riferimenti.

spessore di quantità di moto θ, invece della coordinata x, per identificare la posizione sulla lastra piana alla quale stiamo visualizzando i risultati, perchè in letteratura è più comune trovare riferimenti con questa definizione, anche per differenti geometrie, come per esempio il canale piano. Per maggiore chiarezza, in questa e nelle successive curve del cf, viene inserito il riferimento della posizione al Re_θ corrispondente delle curve dei successivi coefficienti, riportando anche indicativamente l'equivalente in Re_x e la coordinata x corrispondente.

Nel grafico del coefficiente d'attrito le curve presentano una leggera differenza, sopratutto i due set sperimentali che è possibile osservare in tutto l'estensione del grafico, e che discostano maggiormente dal profilo di Prantl-Von Karman. Nel grafico del profilo di velocità media le tre curve presentano la stessa pendenza nella regione logaritmica, ma leggermente traslata, per poi discostarsi maggiormente nella regione lontana da parete, dove $y^+ > 2000$. La spiegazione principale di questa differenza è da imputare per gran parte alla differenza di Re_θ delle diverse prove a disposizione, che risulterebbero più vicine.

4.1.2 Validazione implementazione BLv2k

In questa sezione vengono confrontati i dati ottenuti dalle prove eseguite con il BLv2k e i dati originali forniti negli articoli di Billard [3–6] per il canale piano e la lastra piana per il solo cf, riportando inoltre per i coefficienti anche i dati della DNS.

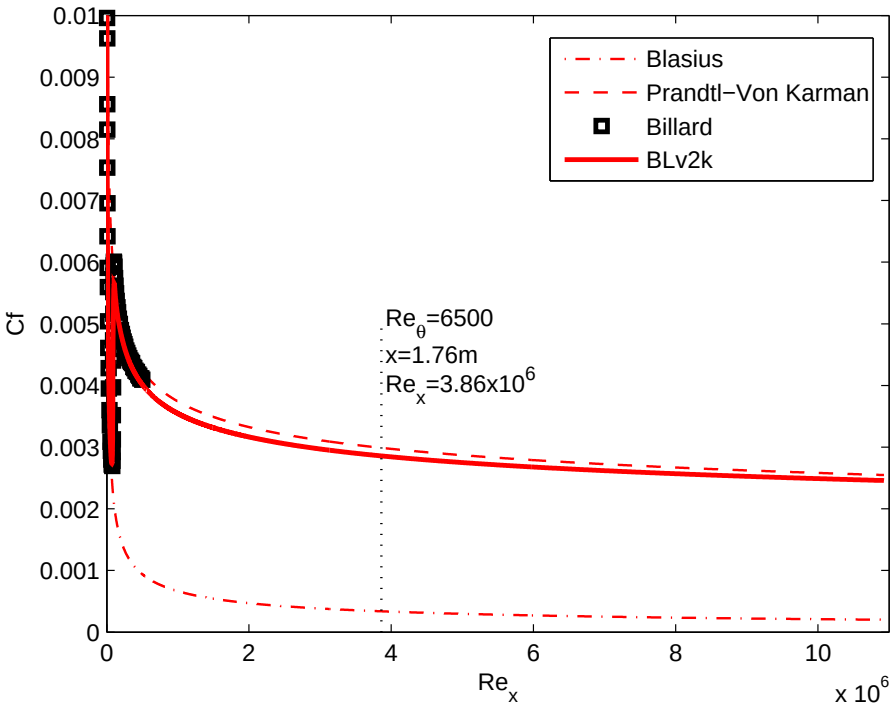

Figura 4.4: Validazione del BLv2k: andamento del cf

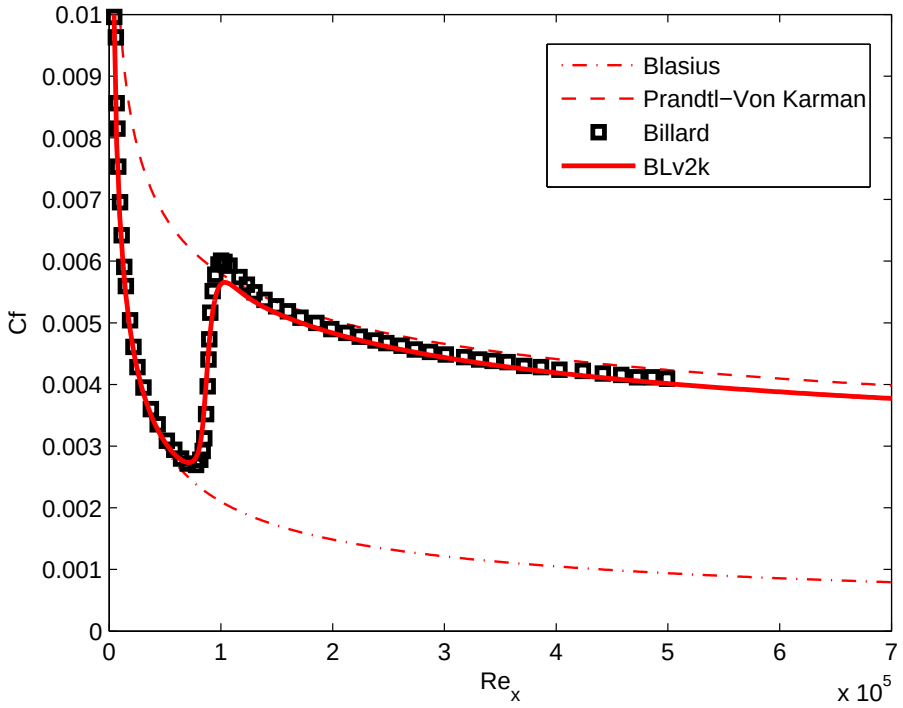

Figura 4.5: Validazione del BLv2k: zoom sulla transizione del cf

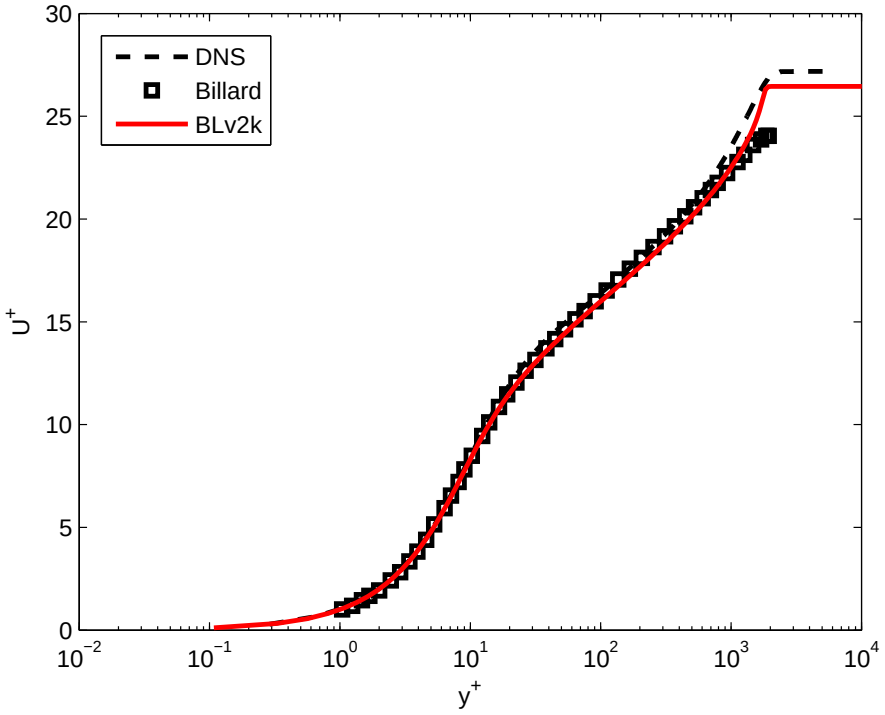

Figura 4.6: Validazione del BLv2k: andamento di U^+

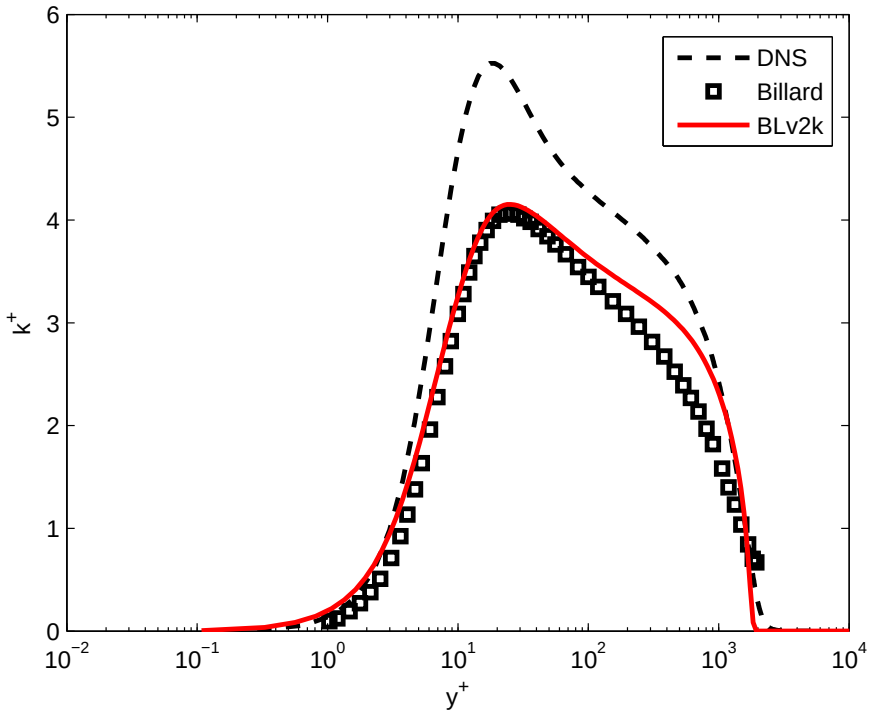

Figura 4.7: Validazione del BLv2k: andamento di k^+

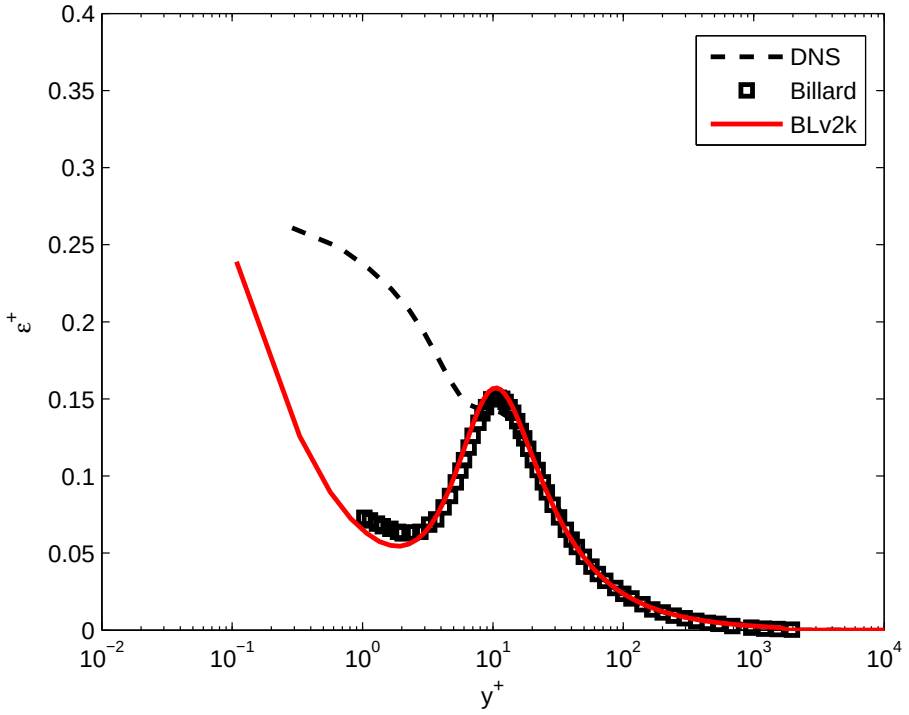

Figura 4.8: Validazione del BLv2k: andamento di ϵ^+

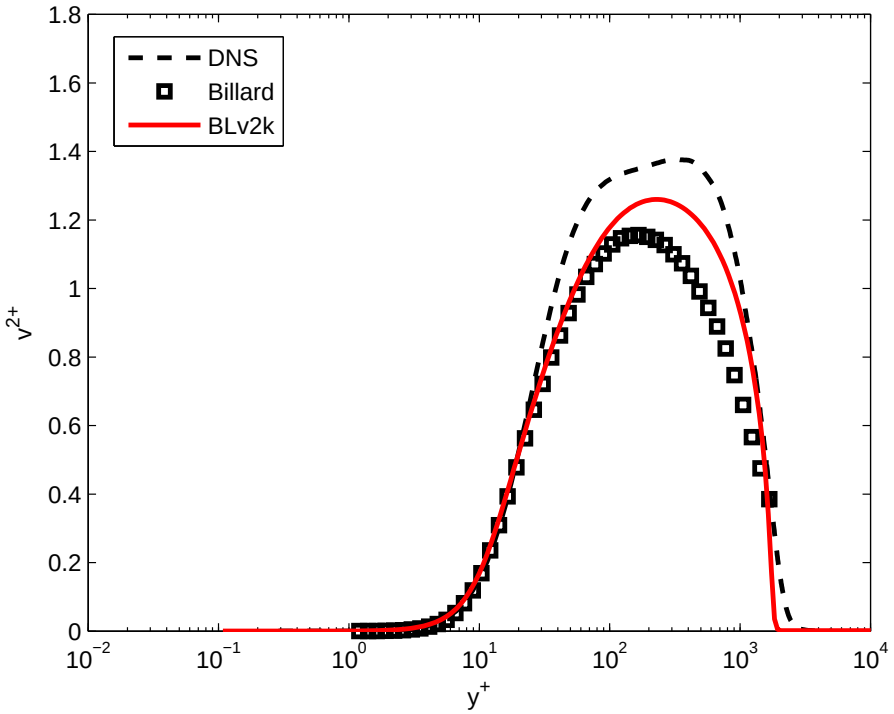

Figura 4.9: Validazione del BLv2k: andamento di $\overline{v'^2}^+$

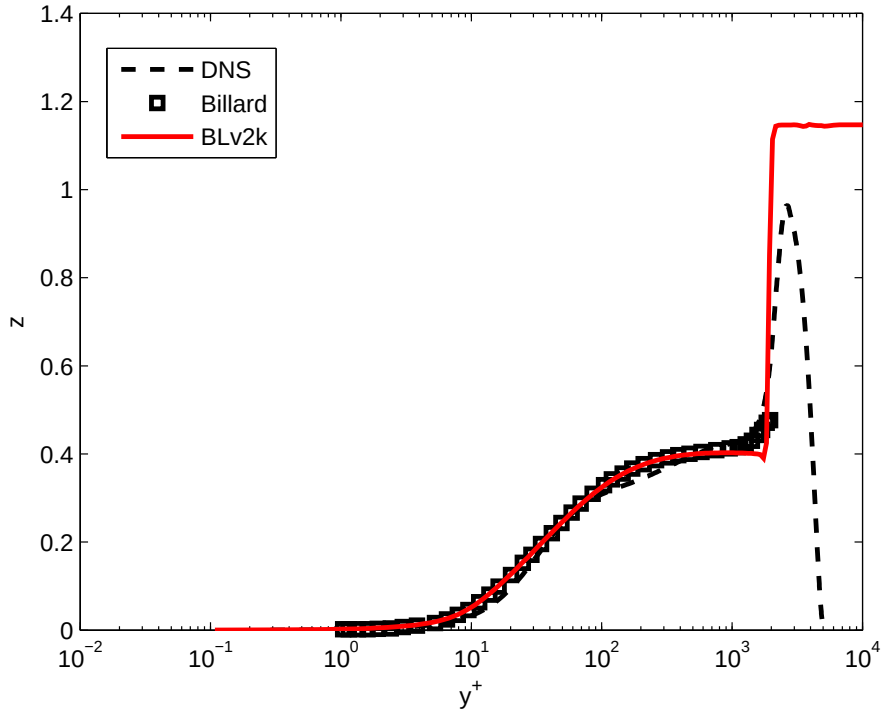

Figura 4.10: Validazione del BLv2k: andamento di φ (z)

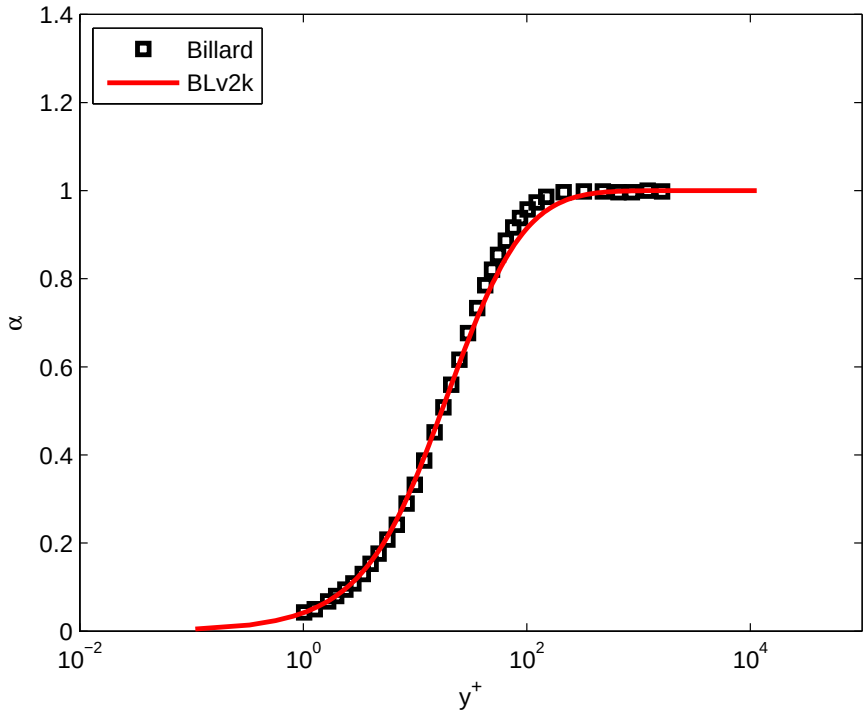

Figura 4.11: Validazione del BLv2k: andamento di α

Le simulazioni sono state eseguite a $Re_\theta = 6500$ che corrisponde circa a $Re_\tau = 2000$ delle prove di Billard su canale piano, e questo punto viene evidenziato anche nel grafico del cf come fatto precedentemente.

Tra le due implementazioni si nota un'ottima sovrapposizione, anche per quanto riguarda la transizione dello strato limite da laminare a turbolento, che normalmente è molto suscettibile di variazioni considerevoli. Leggermente più distanti sono invece le curve delle figure dei coefficienti k e $\overline{v'^2}$, ma questo può essere spiegato dal fatto che le curve ottenute dagli articoli riguardano un canale piano, quindi lontano da parete potrebbero diventare non più trascurabili gli effetti della seconda parete. Altra spiegazione risiede nel differente software utilizzato per la simulazione, e alle differenti condizioni di prova, comprendendo differenti schemi numerici e parametri utilizzati.

4.1.3 Confronto v2f-v2fm-BLv2k

Una volta validato il modello implementato con le curve originali si passa al confronto diretto e principale, paragonando i tre differenti modelli analizzati fino a questo punto (v2f, v2fm, BLv2k). Come i precedenti confronti si riporta il grafico del cf e quello delle principali variabili, assieme ai riferimenti DNS e ai dati sperimentali.

Per i grafici delle variabili si sottolinea che visualizzando l'andamento ad una data posizione x della lamina, ogni curva possiede un differente Re_θ, avendo uno strato limite differente. Reciprocamente, visualizzando l'andamento al medesimo

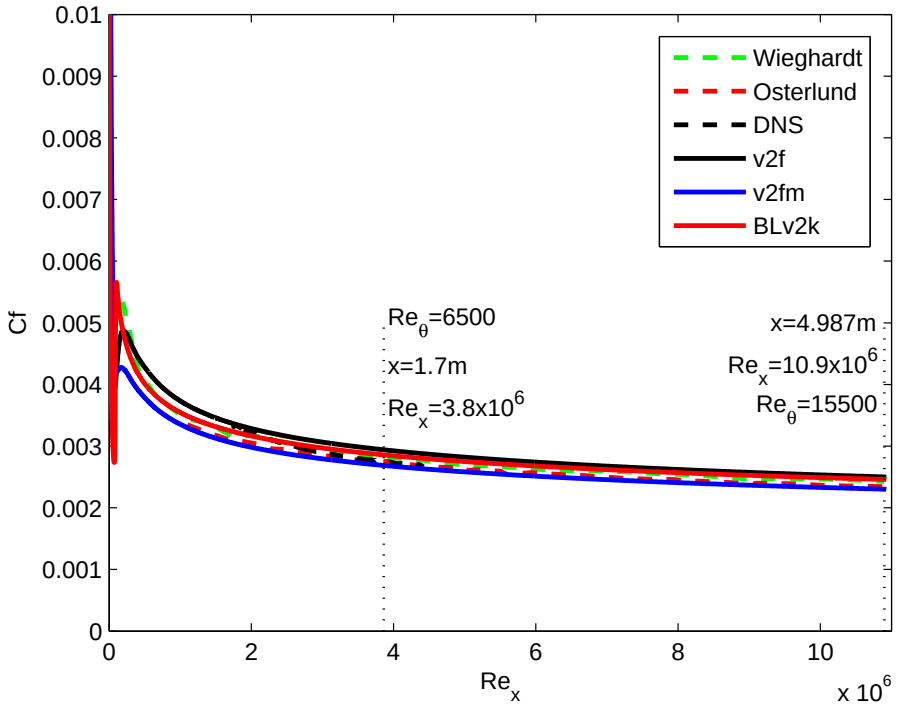

Figura 4.12: Confronto di v2f-v2fm-BLv2k: andamento del cf

Re_θ, risulta differente la coordinata x. Per questo motivo vengono riportate due curve del profilo di velocità in due differenti stazioni della lamina: la Figura 4.14 è riferita alla coordinata $x = 4.987m$, mentre la Figura 4.15 è riferita a $Re_\theta = 6500$. Le stazioni sono state scelte anche in modo che nella prima sia possibile confrontare gli andamenti con entrambi i riferimenti sperimentali, e nella seconda confrontare i riferimenti con la DNS e i dati sperimentali di Osterlund. Per la seconda coordinata ($Re_\theta = 6500$) vengono poi riportati tutti i coefficienti.

Dai grafici del cf (Figura 4.12 e 4.13) possiamo notare come le curve dei tre modelli presentino delle differenze. In particolare vi è una sovrastima del coefficiente per il v2f originale, che risulta superiore a tutti i riferimenti. Apportando la modifica alla ϵ a parete il coefficiente cambia molto, andando a sottostimare i valori suggeriti dai riferimenti. Il BLv2k invece si trova tra i due, ma molto prossimo al v2f e ai dati di Wieghardt. Per quanto riguarda la transizione, i modelli si allineano pressapoco nelle stesse vicinanze, avendo però differente rapidità nel passaggio, che vede nel BLv2k il più rapido.

Nonostante la differente coordinata, le curve della velocità sono coerenti tra di loro, e si riscontrano le medesime differenze osservate nel grafico del cf, cioè una stima rispettivamente crescente per v2f, BLv2k e v2fm. Si nota inoltre come tutti i modelli tendano alla stessa pendenza nella zona logaritmica, ma con rapidità differente, portando poi a sovrastimare o sottostimare il valore lontano da parete della

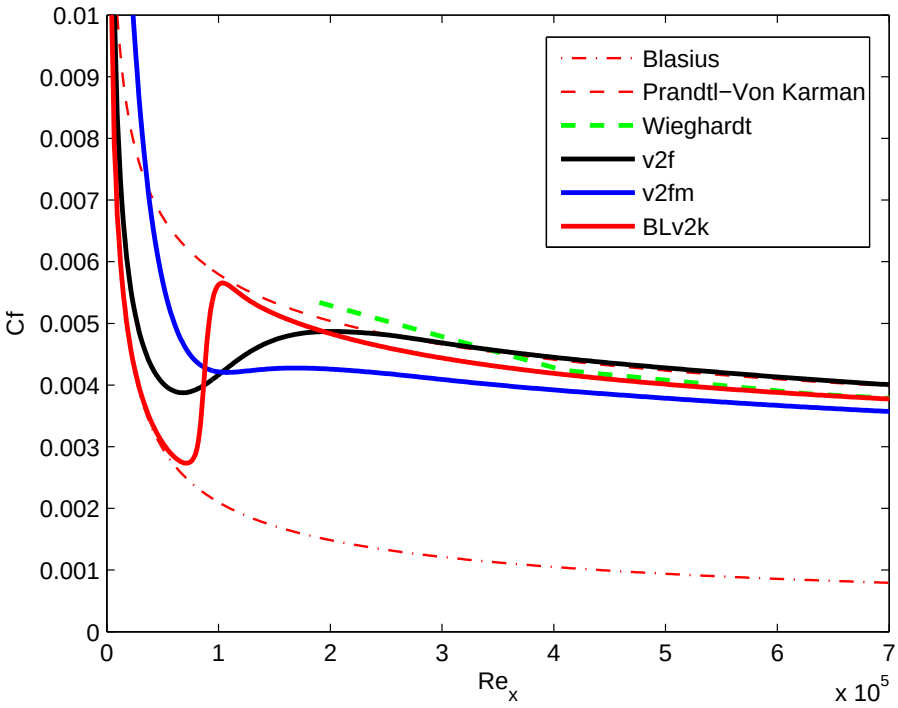

Figura 4.13: Confronto di v2f-v2fm-BLv2k: zoom dell'andamento del cf

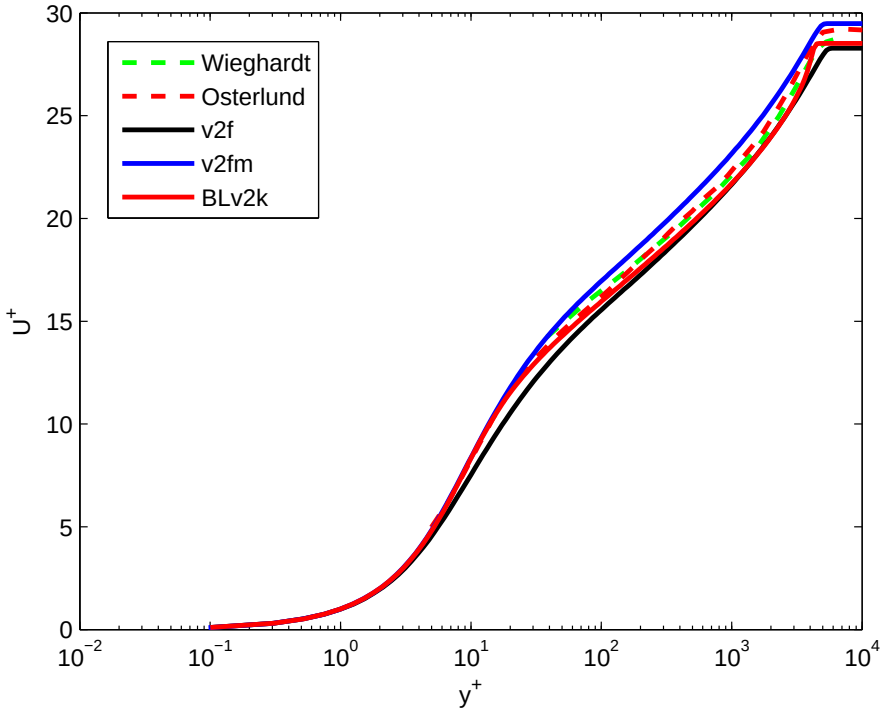

Figura 4.14: Confronto di v2f-v2fm-BLv2k: andamento di U^+ a $x = 4.987m$

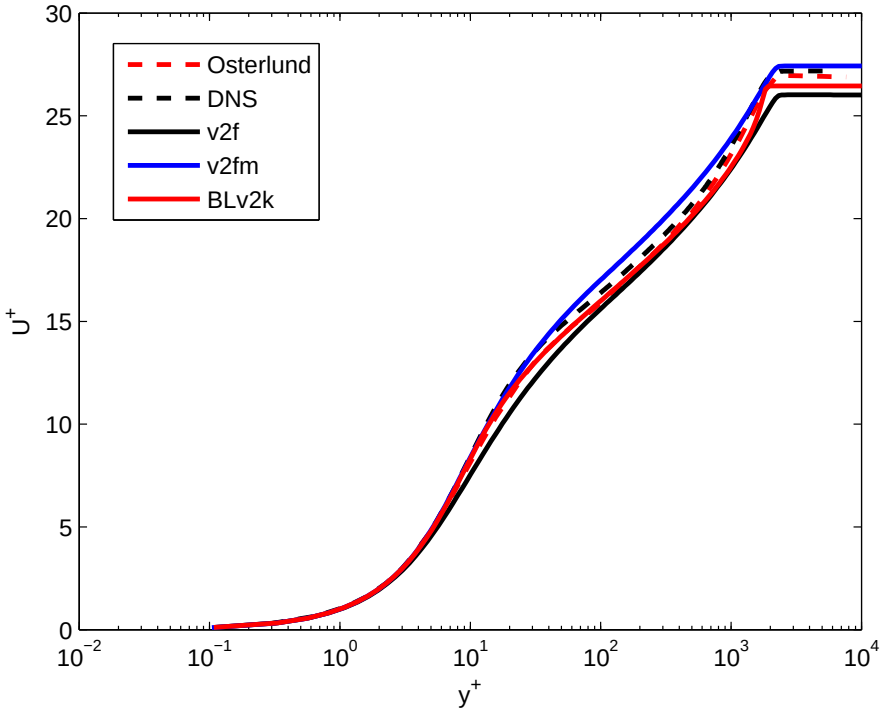

Figura 4.15: Confronto di v2f-v2fm-BLv2k: andamento di U^+ a $Re_\theta = 6500$

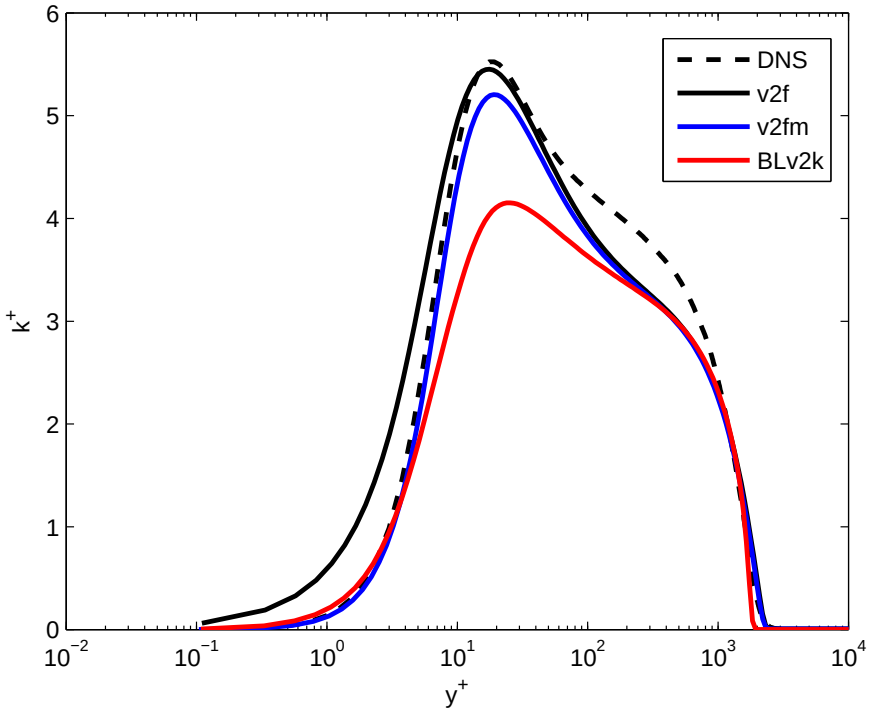

Figura 4.16: Confronto di v2f-v2fm-BLv2k: andamento di k^+ a $Re_\theta = 6500$

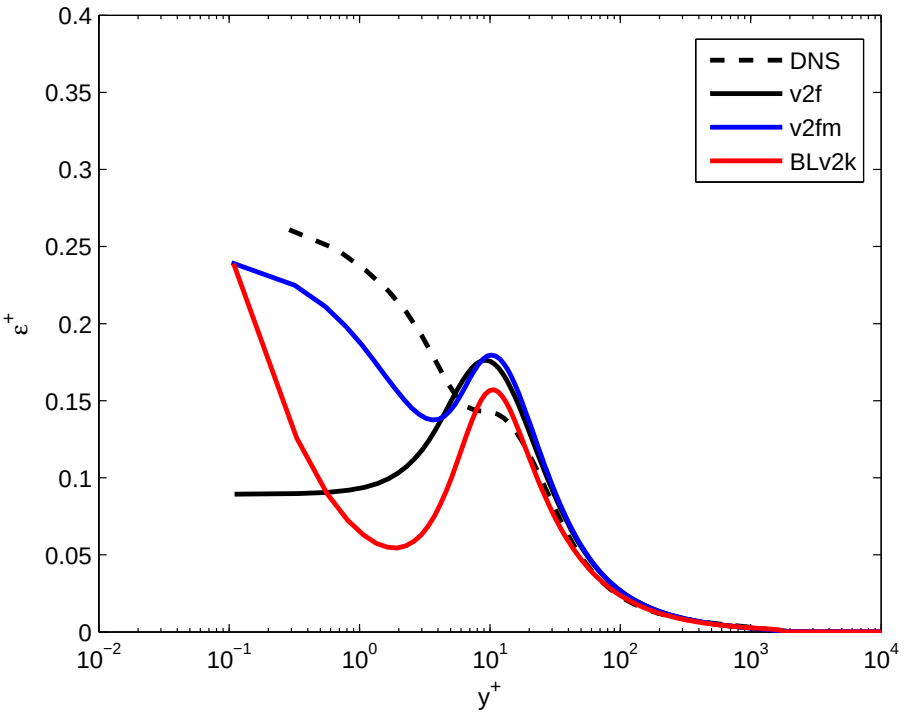

Figura 4.17: Confronto di v2f-v2fm-BLv2k: andamento di ϵ^+ a $Re_\theta = 6500$

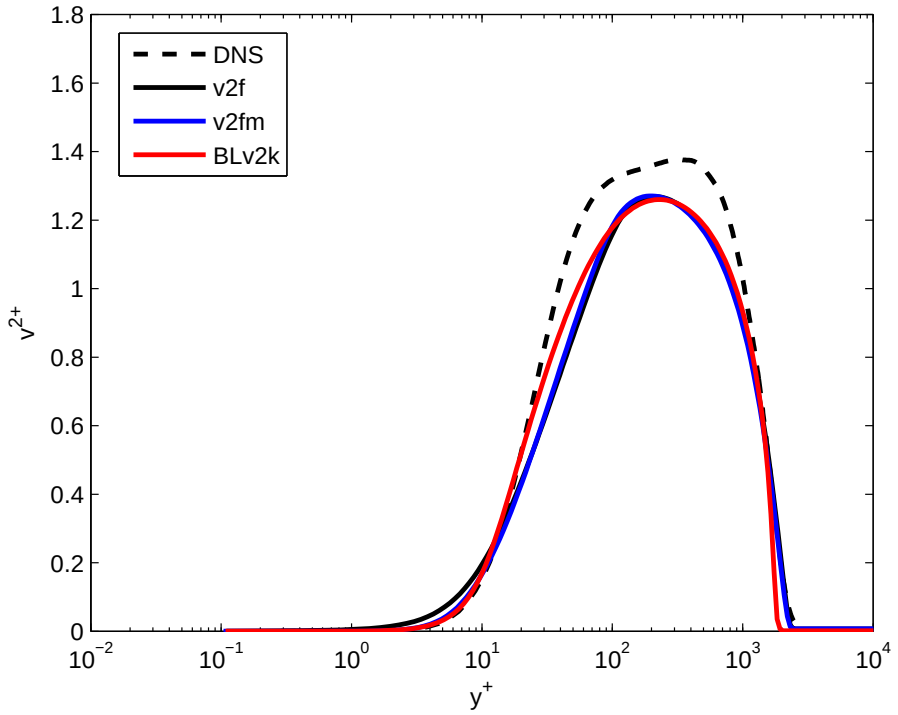

Figura 4.18: Confronto di v2f-v2fm-BLv2k: andamento di $\overline{v'^2}^+$ a $Re_\theta = 6500$

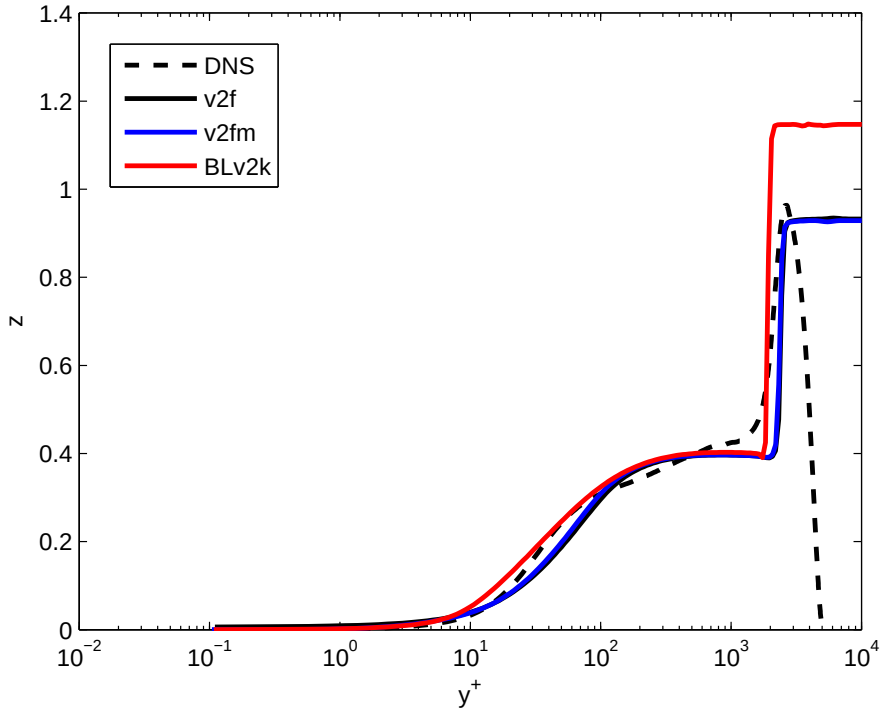

Figura 4.19: Confronto di v2f-v2fm-BLv2k: andamento di φ (z) a $Re_\theta = 6500$

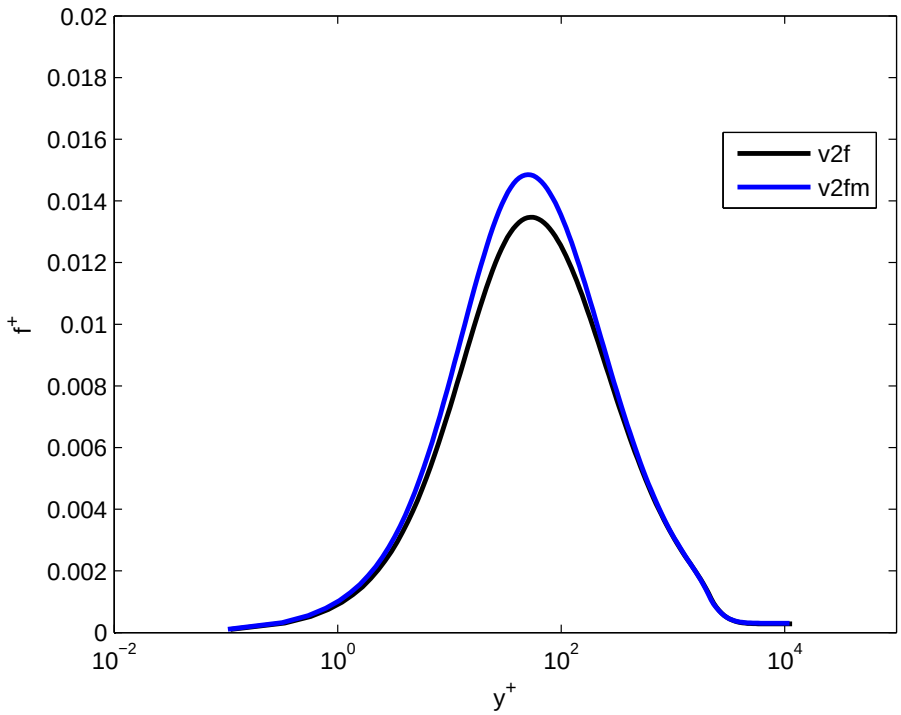

Figura 4.20: Confronto di v2f-v2fm-BLv2k: andamento di f^+ a $Re_\theta = 6500$

velocità. Nei grafici successivi si può notare come il BLv2k tenda a sottostimare il picco di energia cinetica turbolenta, essendo una caratteristica del modello, e a possedere un differente avvicinamento alla condizione a parete nella ϵ. Sempre nel grafico dell'energia cinetica turbolenta possiamo notare come la diversa condizione al contorno per la ϵ del v2fm gli conferisca un comportamento molto più prossimo a quello della DNS rispetto a quello del v2f originale. Per quanto riguarda gli altri coefficienti non si notano sostanziali differenze, ad eccezione del differente picco del BLv2k nella φ (z), dovuto alla già citata sottostima di k.

4.2 Simulazioni High-Re

Dai risultati Low-Reynolds è stato quindi possibile ricavare le tabelle per il funzionamento dei casi High-Reynolds, che sono stati condotti a tre differenti distanze da parete: $y^+ = 10, 50, 100$. I confronti vengono riportati di seguito per ogni modello paragonandoli alle rispettive simulazioni Low-Reynolds.

4.2.1 v2f

Nella figura del cf riportata si può notare come il comportamento assunto dalle simulazioni High-Reynolds sia nell'intorno del comportamento della simulazione con integrazione fino a parete.

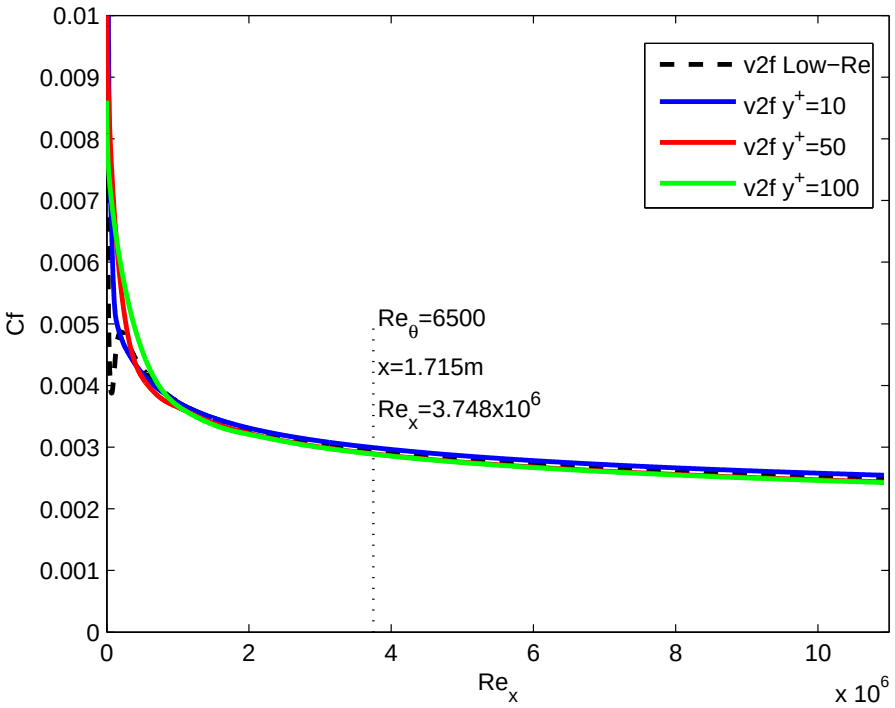

Figura 4.21: Confronto High-Reynolds v2f: andamento del cf

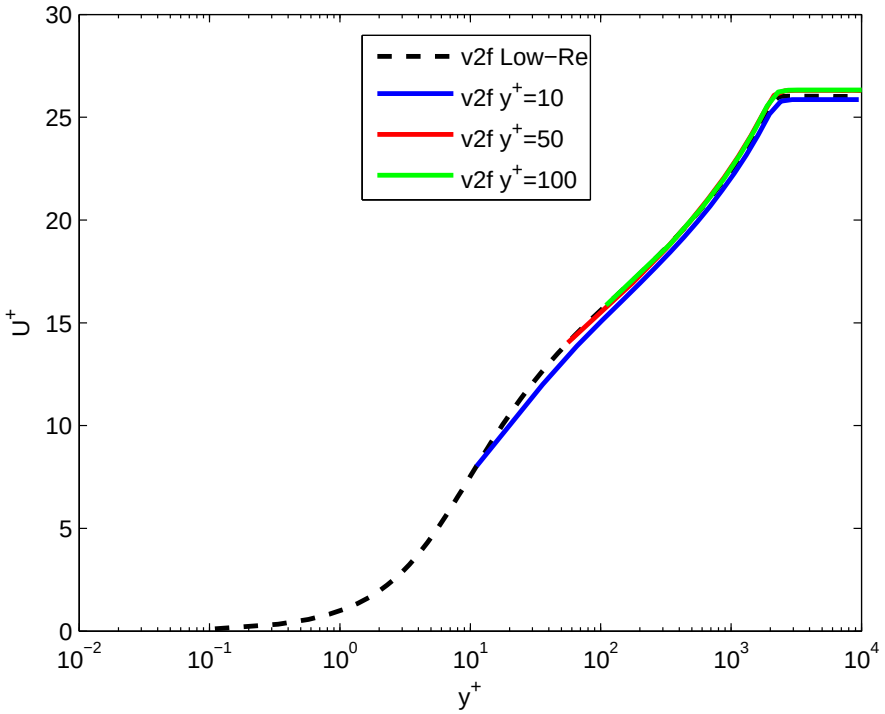

Figura 4.22: Confronto High-Reynolds v2f: andamento di U^+

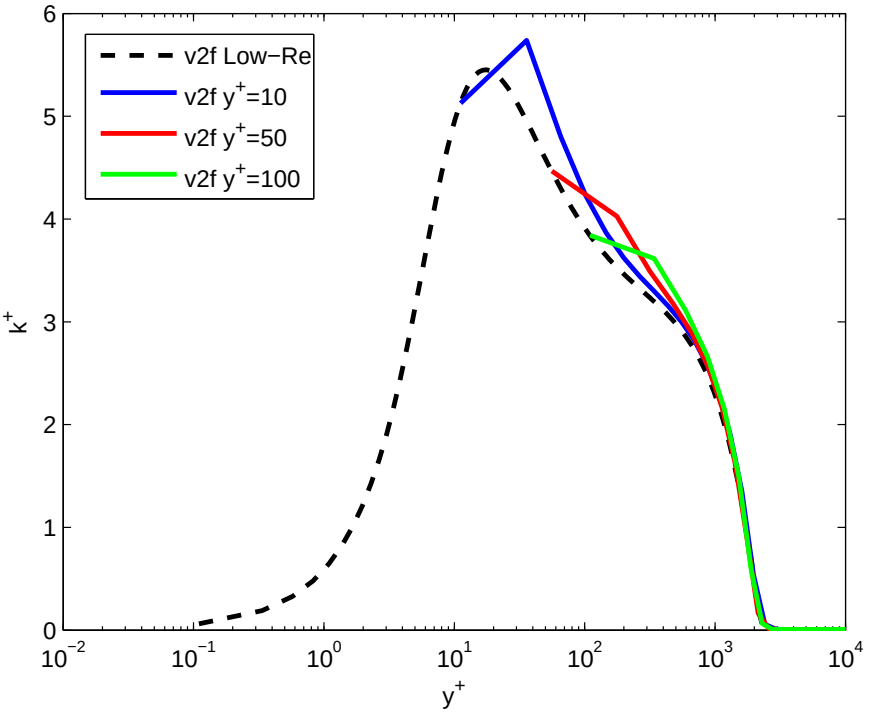

Figura 4.23: Confronto High-Reynolds v2f: andamento di k^+

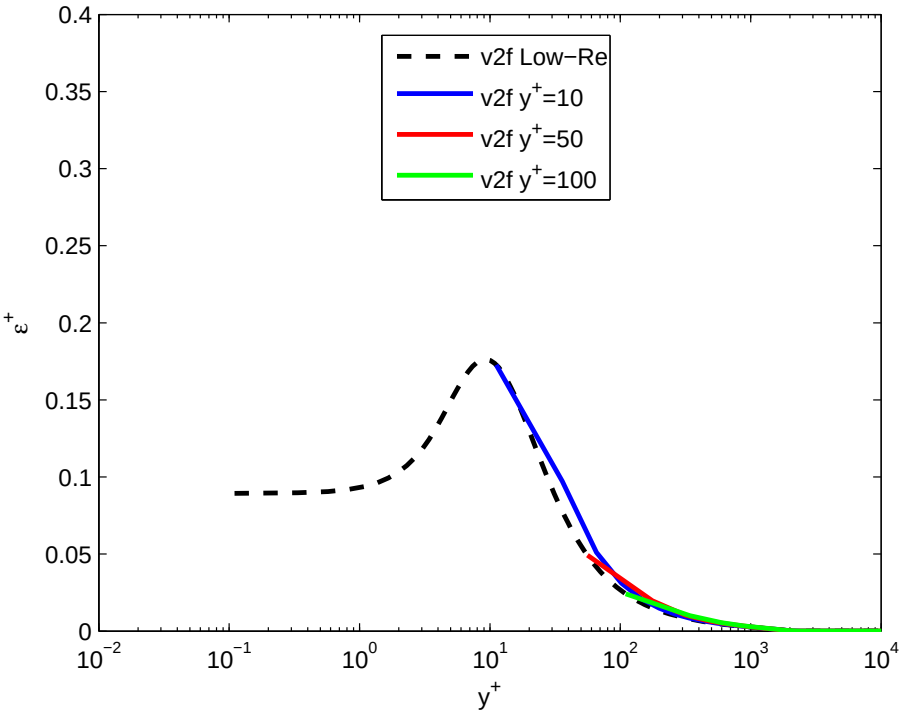

Figura 4.24: Confronto High-Reynolds v2f: andamento di ϵ^+

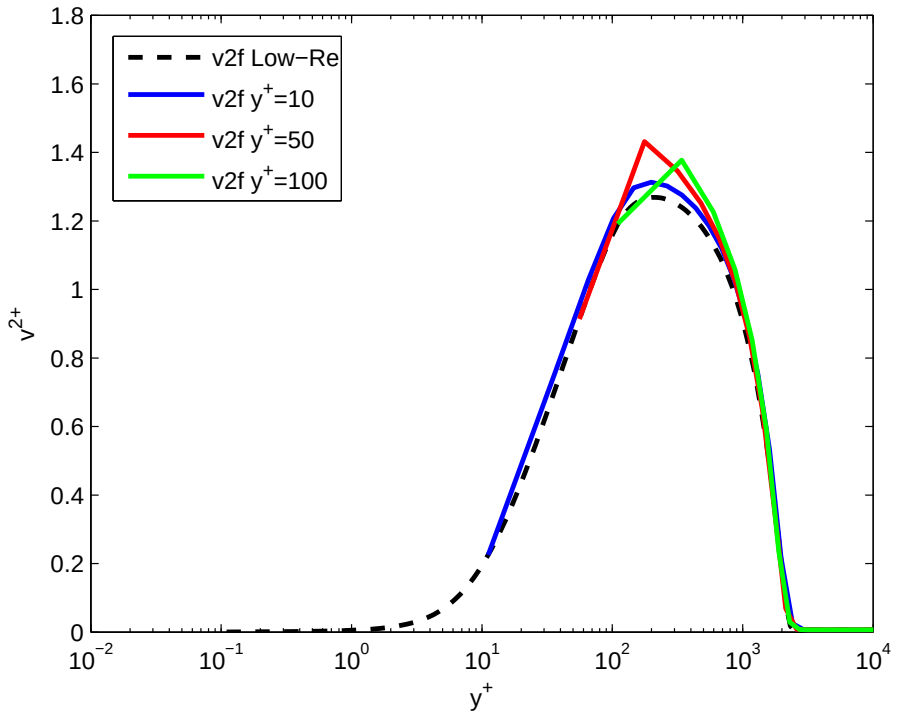

Figura 4.25: Confronto High-Reynolds v2f: andamento di $\overline{v'^2}^+$

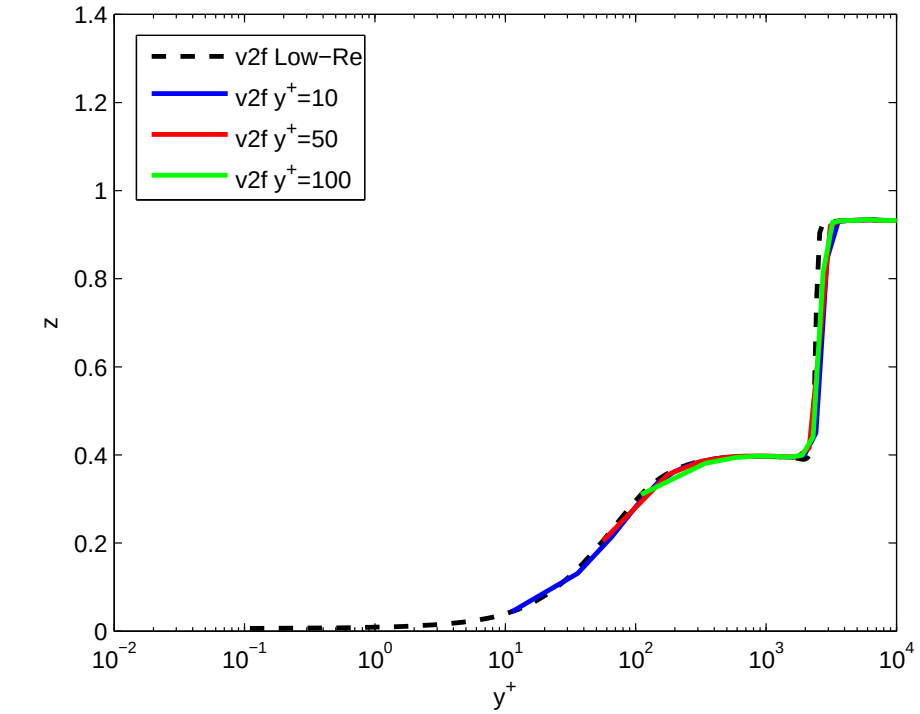

Figura 4.26: Confronto High-Reynolds v2f: andamento di φ (z)

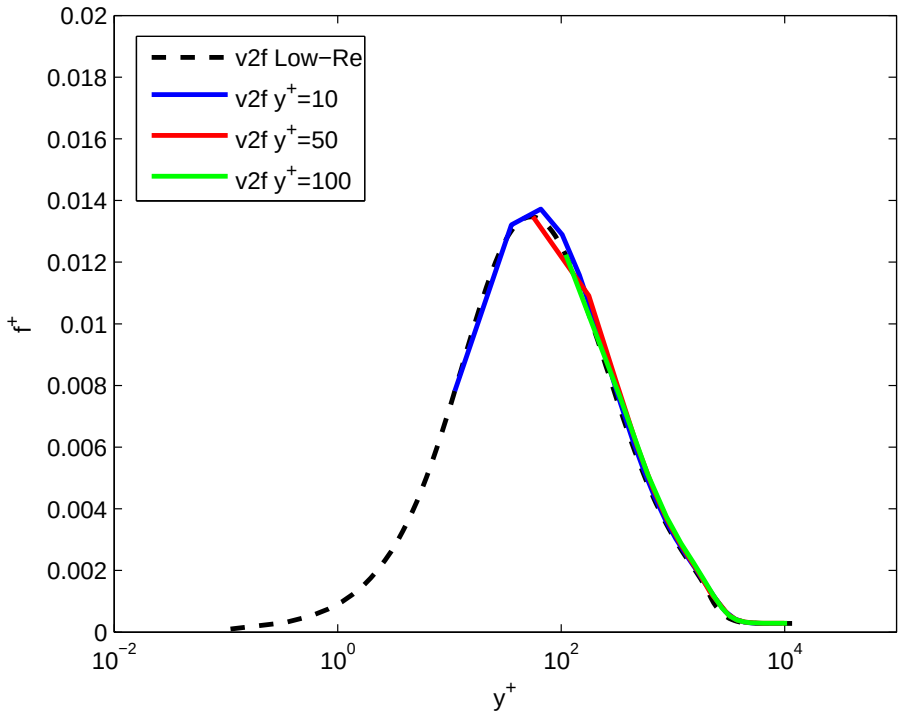

Figura 4.27: Confronto High-Reynolds v2f: andamento di f^+

Nello specifico la simulazione a $y^+ = 10$ tende a sovrastimare la resistenza, mentre le altre due curve tendono a sottostimarla. Il comportamento all'inizio della lamina invece risulta più vario, dove le tre varianti partono già con un flusso turbolento. Osservando la curva del profilo di velocità possiamo notare come la curva a $y^+ = 10$ cada nella regione di transizione tra lineare e logaritmico. Questa caratteristica potrebbe essere la ragione per cui questa curva si discosti dalle altre nella regione logaritmica, ed abbia un comportamento differente anche nell'andamento del cf. Osservando gli altri coefficienti risulta evidente come il primo punto cada effettivamente sempre sopra la curva Low-Reynolds, ma nel calcolo del secondo punto vengono introdotti degli errori notevoli, in particolar modo nelle curve di k^+ e $\overline{v'^2}^+$. Queste differenze si ripercuotono poi negli altri punti successivi, impiegando anche due decadi di y^+ prima di riportarsi sulla curva di riferimento.

4.2.2 v2fm

Per il v2fm possiamo notare lo stesso comportamento del v2f. Nel grafico del cf notiamo sempre una differenza tra le tre versioni High-Reynolds e la versione Low-Reynolds, che vede la curva a $y^+ = 10$ andare nella parte opposta delle altre due. Questa volta però possiamo notare come questa curva riesca ad accennare anche la transizione del flusso,

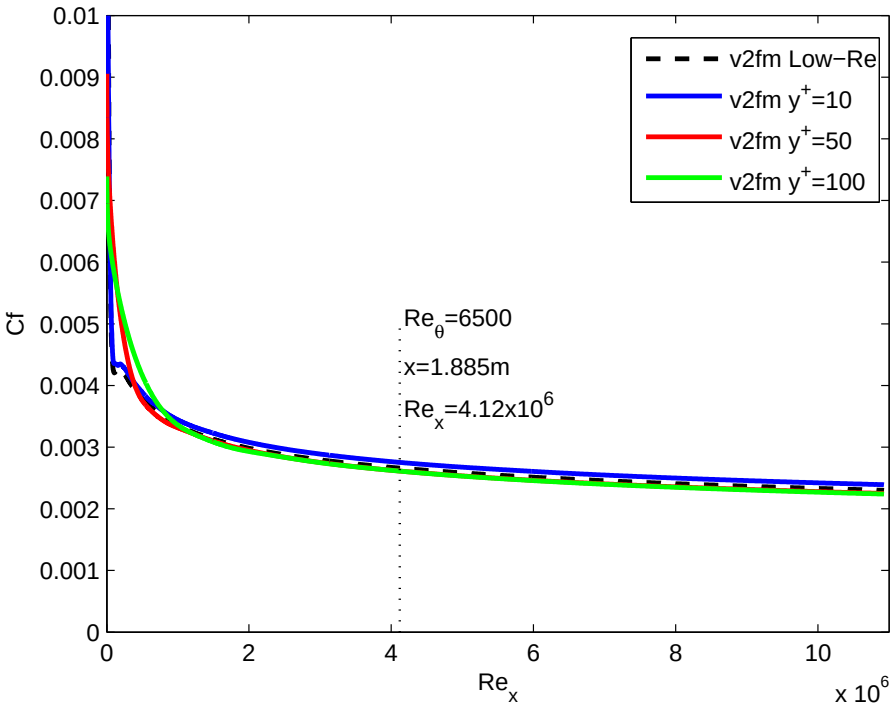

Figura 4.28: Confronto High-Reynolds v2fm: andamento del cf

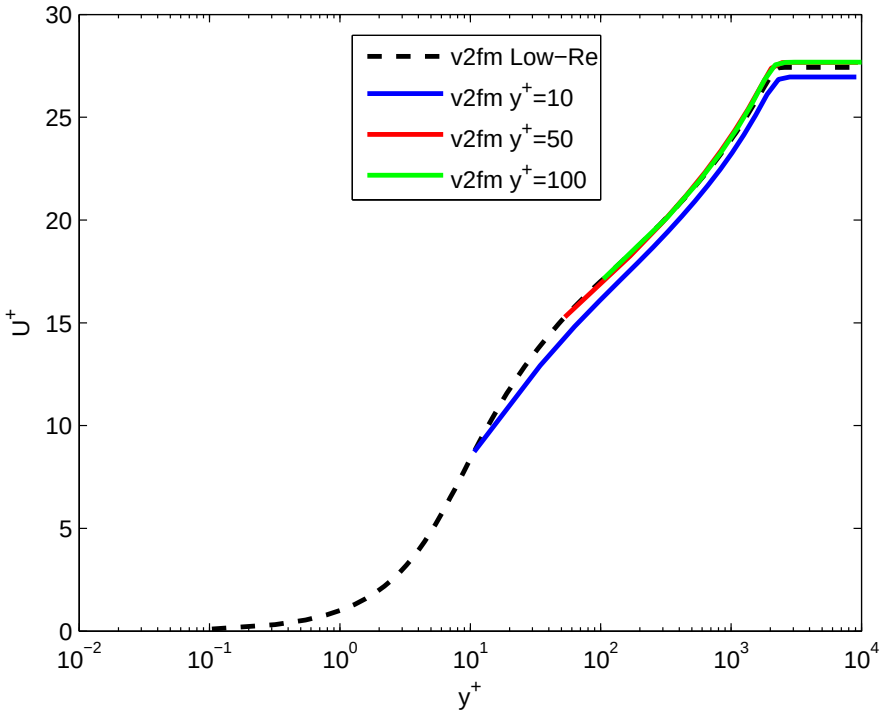

Figura 4.29: Confronto High-Reynolds v2fm: andamento di U^+

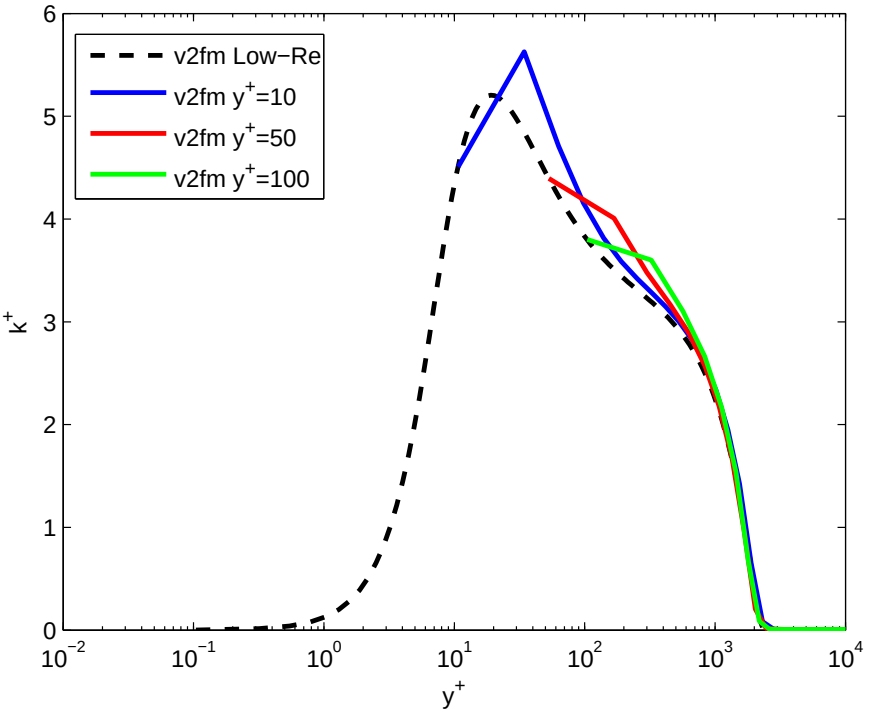

Figura 4.30: Confronto High-Reynolds v2fm: andamento di k^+

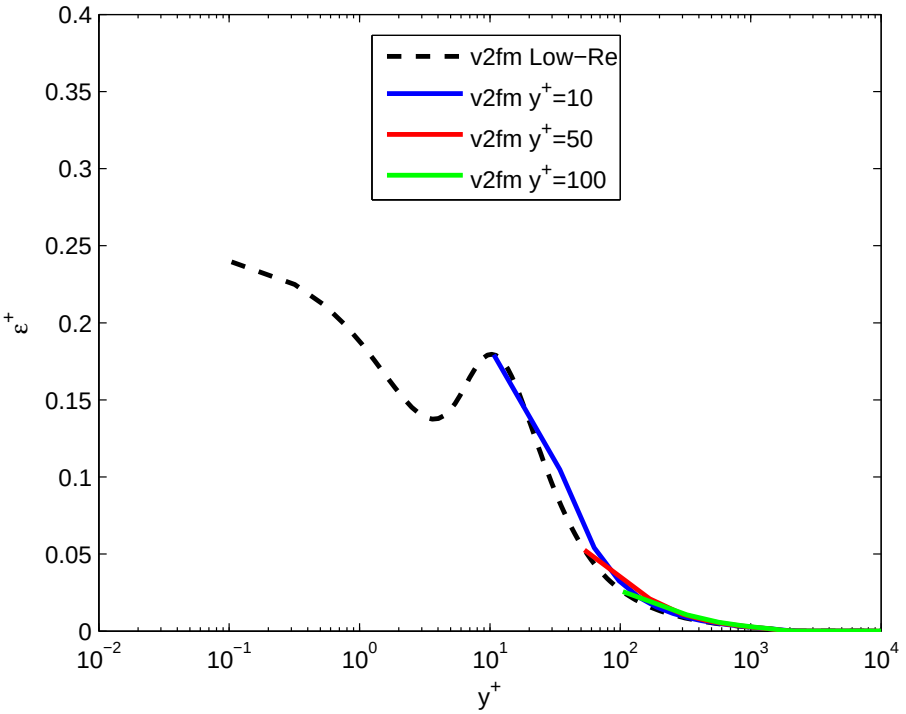

Figura 4.31: Confronto High-Reynolds v2fm: andamento di ϵ^+

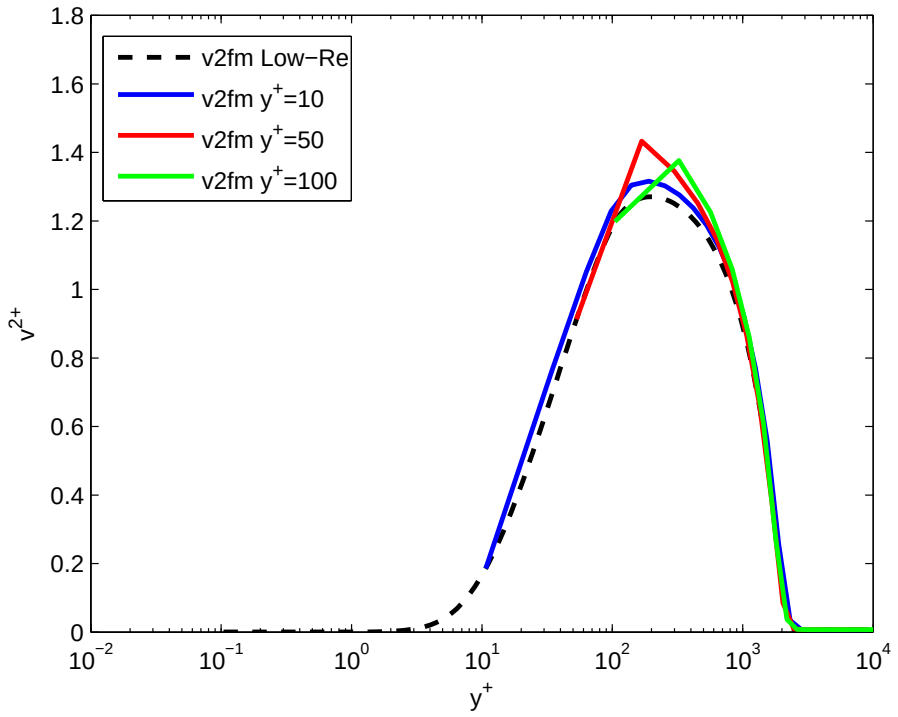

Figura 4.32: Confronto High-Reynolds v2fm: andamento di $\overline{v'^2}^+$

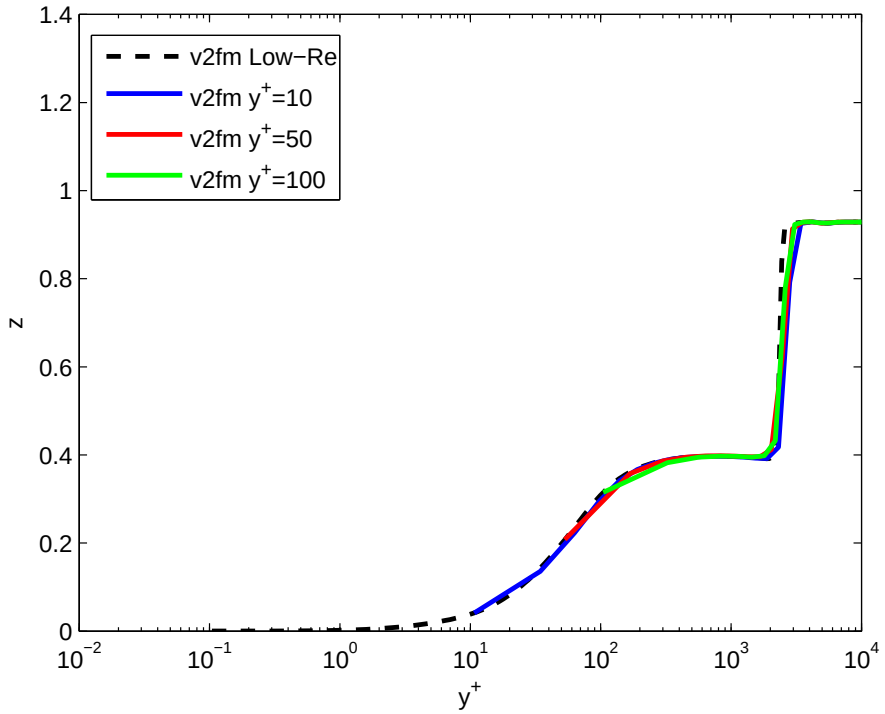

Figura 4.33: Confronto High-Reynolds v2fm: andamento di φ (z)

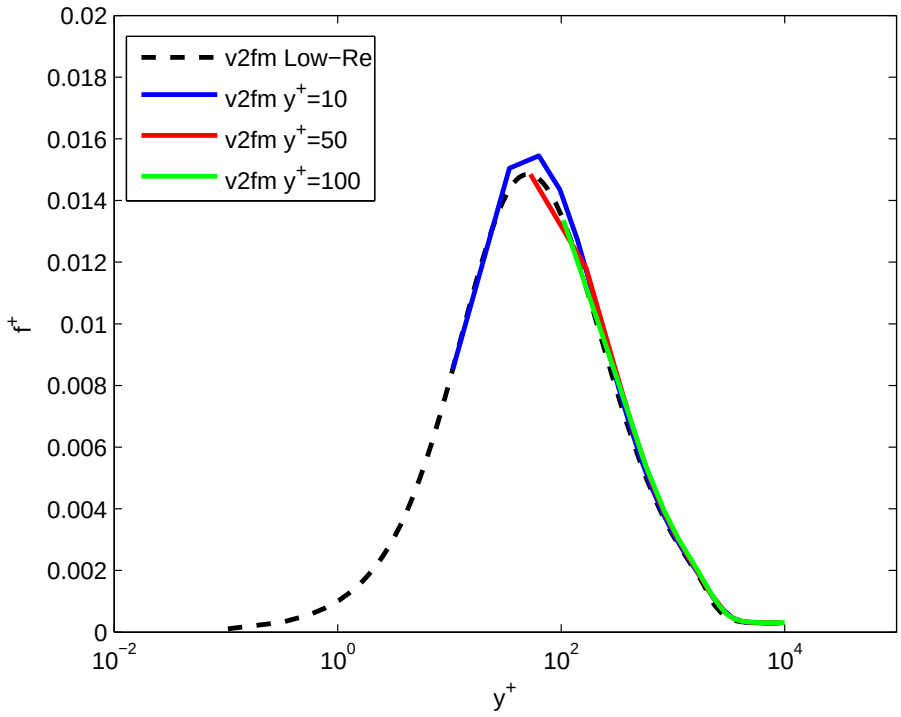

Figura 4.34: Confronto High-Reynolds v2fm: andamento di f^+

caratteristica che però non può essere affidabile in quanto non rientra in generale nelle caratteristiche di un modello RANS, sopratutto se si utilizzano wall-function. Nel profilo di velocità invece l'errore della curva a $y^+ = 10$ tende ad aumentare, sia come pendenza nella zona logaritmica, sia nella regione più esterna dove la curva diventa orizzontale, mentre sembrano rimanere invariate le altre due curve a $y^+ = 50$ e $y^+ = 100$. Gli altri coefficienti hanno gli stessi comportamenti riscontrati con il v2f, dove le curve di k^+ e $\overline{v'^2}^+$ hanno maggiori problemi e dove la curva di f^+ a $y^+ = 10$ è leggermente peggiorata.

4.2.3 BLv2k

Per questo modello il grafico del coefficiente d'attrito sembra essere leggermente meno coerente con l'andamento rilevato dalla prova High-Reynolds, sempre conservando la differenza tra la curva a $y^+ = 10$ e le altre. All'inizio della lamina le tre curve partono già turbolente e differentemente da prima hanno una oscillazione intorno a $Re_x = 0.5 - 1 \cdot 10^6$. Nel profilo di velocità si nota come tutte le curve siano più distanti rispetto a quanto visto per gli altri modelli, sopratutto nella regione più lontana da parete. Per quanto riguarda i coefficienti, nella curva di k^+ troviamo nuovamente i picchi dei secondi punti di integrazione, come anche nella curva di $\overline{v'^2}^+$ dove il comportamento generale risulta leggermente peggiore, al contrario delle altre variabili che risultano prossime al riferimento della prova Low-Reynolds.

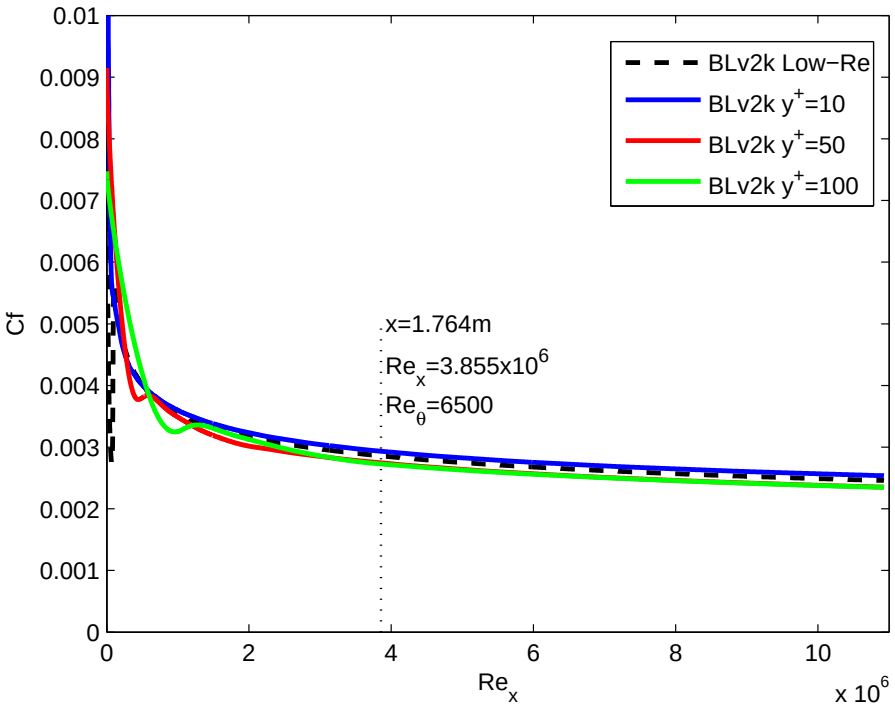

Figura 4.35: Confronto High-Reynolds BLv2k: andamento del cf

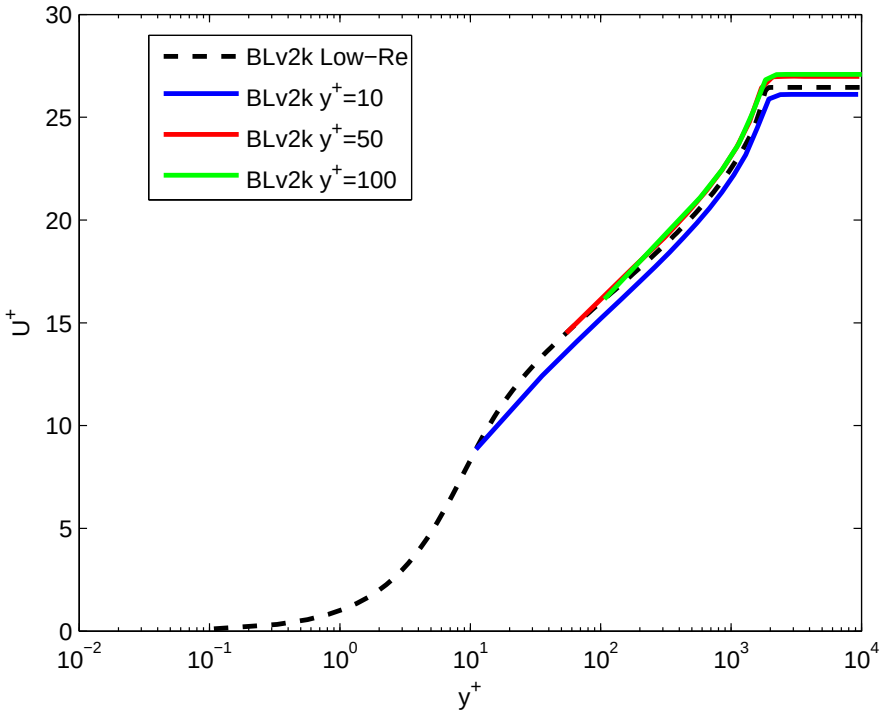

Figura 4.36: Confronto High-Reynolds BLv2k: andamento di U^+

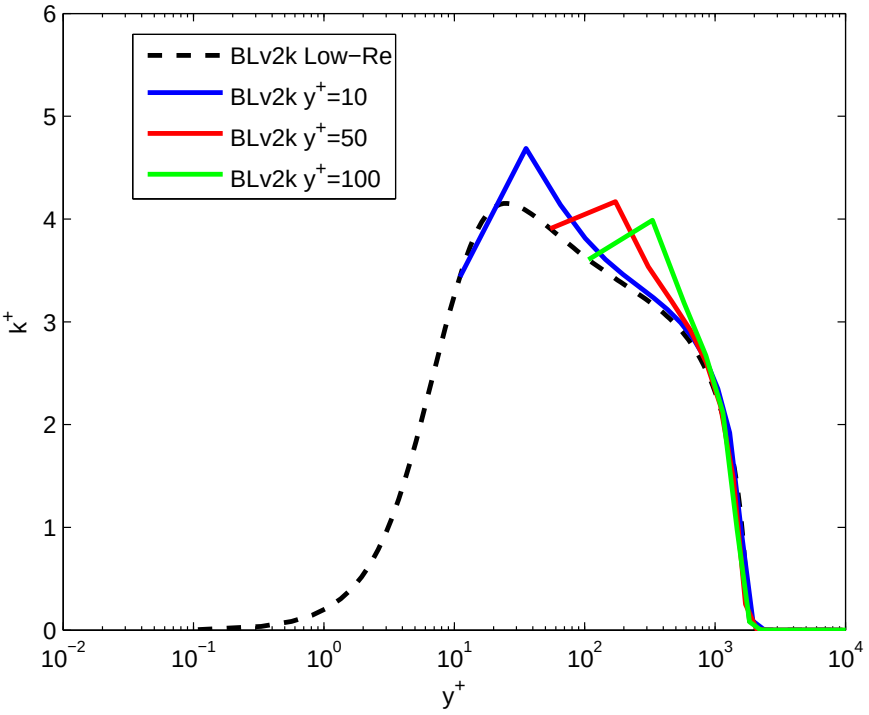

Figura 4.37: Confronto High-Reynolds BLv2k: andamento di k^+

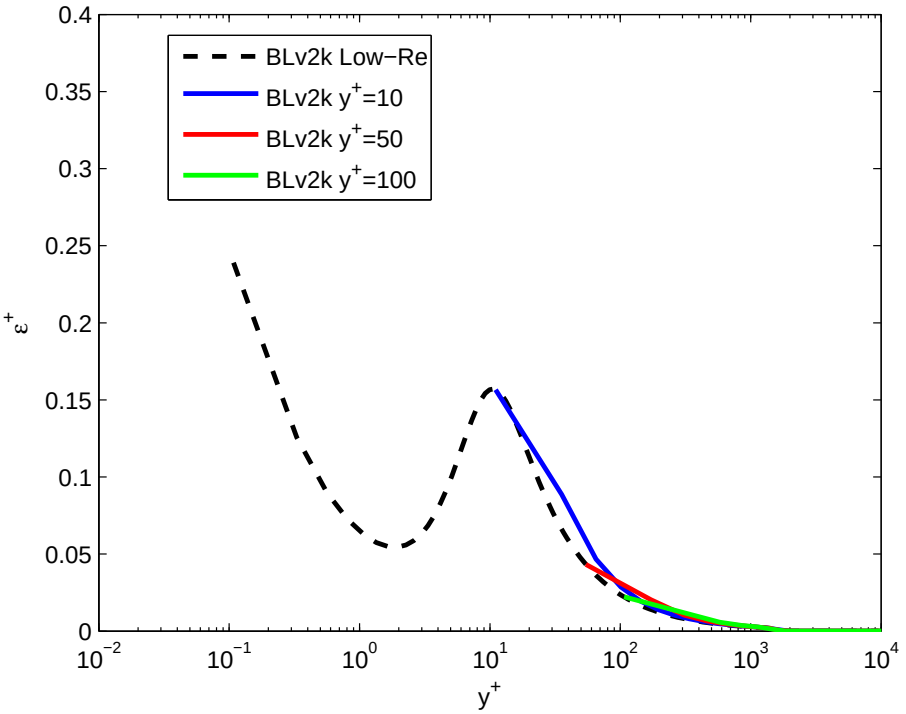

Figura 4.38: Confronto High-Reynolds BLv2k: andamento di ϵ^+

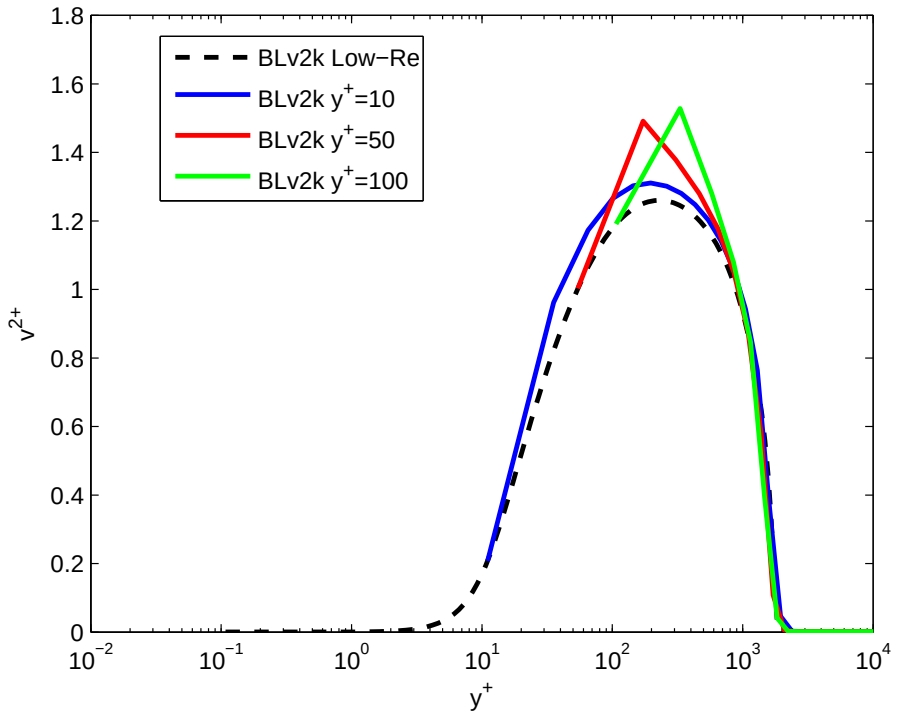

Figura 4.39: Confronto High-Reynolds BLv2k: andamento di $\overline{v'^2}^+$

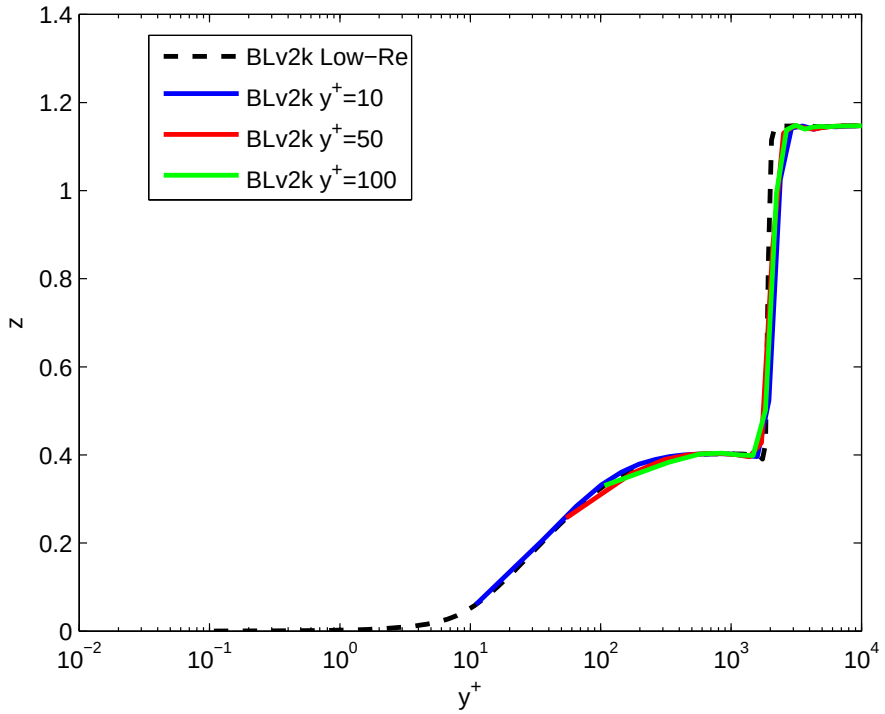

Figura 4.40: Confronto High-Reynolds BLv2k: andamento di φ (z)

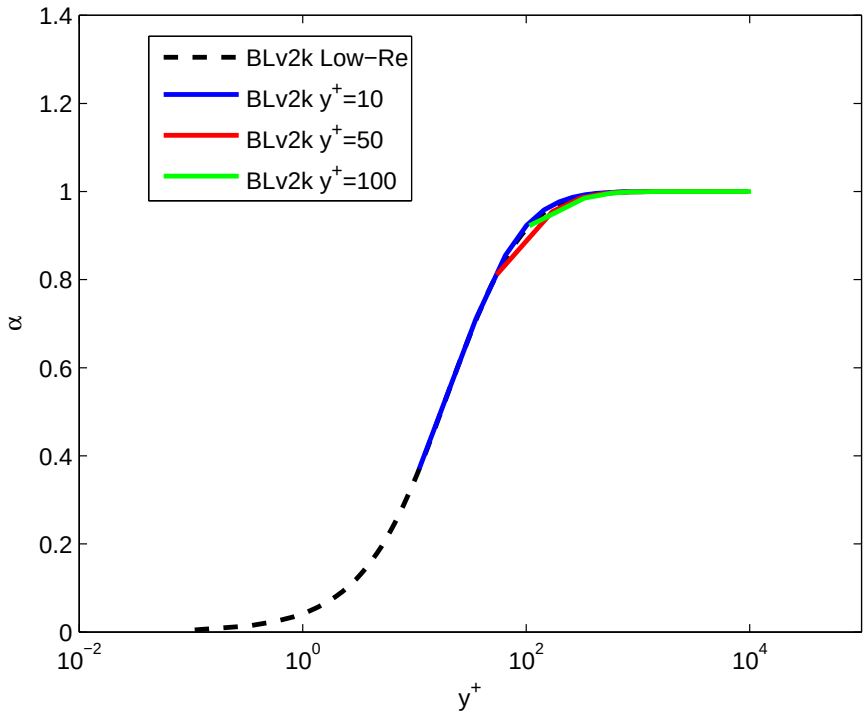

Figura 4.41: Confronto High-Reynolds BLv2k: andamento di α

Conclusioni

In questo lavoro di tesi, che costituisce la continuazione di due lavori precedenti, si è lavorato sul tema dei modelli di turbolenza per la chiusura del sistema delle equazioni di Navier-Stokes mediate alla Reynolds. In particolare lo scopo è quello di validare il nuovo modello $BL\overline{v'^2}/k$, scelto a valle di una iniziale ricerca bibliografica sui recenti sviluppi di modelli di turbolenza a base $\overline{v'^2}$, argomento dei lavori precedenti assieme allo studio di wall-function adattive.

All'interno di OpenFOAM sono state quindi confrontate le prestazione dei tre modelli, attraverso l'analisi dei risultati di numerose simulazioni e il loro confronto con dati di letteratura sia numerici che sperimentali.

Possiamo notare in primo luogo che la nostra implementazione del modello BLv2k in OpenFOAM è in perfetta coerenza con quella originale di Billard e Laurence, nonostante vi siano numerose variabili modificabili, come le condizioni test, i metodi risolutivi delle equazioni e i parametri di soluzione che fanno variare considerevolmente la soluzione.

Dal confronto dei modelli Low-Reynolds $\overline{v'^2} - f$ (v2f) e $\overline{v'^2} - f$ con modifica alla condizione al contorno per ϵ (v2fm) è possibile osservare come la modifica eseguita porti alcune differenze. In primo luogo vicino a parete nel grafico della ϵ e in quello di k, ma anche nel comportamento generale delle curve al crescere della distanza da parete, fino alle distanze più lontane, passando da una condizione di sottostima ad una di sovrastima dei valori. Tra i due modelli è quindi evidente un migliore comportamento da parte del v2fm vicino a parete, a scapito di una differente soluzione al di fuori. Il prezzo da pagare per questa modifica è quello di aggiungere una condizione aggiuntiva a parete che va a bypassare le condizioni imposte dai file appositi di OpenFOAM. Forse per tale motivo nell'atto di inserire questo modello in una distribuzione ufficiale si è preferito privilegiare la semplicità alla maggiore precisione vicino a parete, che risulta pressochè fine a se stessa per simulazioni di carattere generale sopratutto se eseguite con wall-function.

Le caratteristiche del BLv2k Low-Reynolds sono invece un buon compromesso tra le caratteristiche appena descritte: grazie alla sua stessa definizione il modello risulta più stabile numericamente, e il comportamento rilevato dai coefficienti risulta corretto vicino a parete e compreso tra i comportamenti del v2f e del v2fm. Va anche

sottolineato come le differenze fino a qui osservate tra le versioni dei modelli siano dello stesso ordine delle differenze tra i riferimenti (sperimentali e DNS). Questo a dimostrare che è possibile solamente affermare che un modello sia più o meno coerente con i riferimenti considerati, ma non si può definirne un comportamento ottimale a priori. Globalmente questo modello ha quindi soddisfatto le aspettative, ed è pronto ora per una valutazione più dettagliata attraverso prove più specifiche, come prove bidimensionali e tridimensionali dove questo modello dovrebbe garantire le sue massime performance.

Per quanto riguarda l'utilizzo delle wall-function adattive si è notato un comportamento simile da parte dei tre modelli, ma in questa fase il BLv2k mostra leggere difficoltà in più rispetto agli altri, sopratutto nella condizione in cui il primo punto della cella ricada nel buffer-layer, caratteristiche che potrebbero compromettere la sua teorica superiorità nel caso di utilizzo per problemi industriali. Altra difficoltà comune a tutti i modelli è la soluzione errata del secondo punto della mesh che può essere alleviata pensando di inserire una mesh in cui il secondo punto ricada più vicino al primo, oppure direttamente imponendo il valore della tabella anche per il secondo punto della mesh, come descritto nel lavoro di tesi precedente. Questo mitigherebbe gli errori riscontrati, ma dovrebbe mantenere la differenza esistente tra i modelli.

Bibliografia

[1] M. Angiolini. Funzioni di parete adattive per il modello di turbolenza $k-\omega$. Master's thesis, Politecnico di Milano, 2011/2012.

[2] S. Benelli. Implementazione e verifica di un modello di turbolenza $\overline{v'^2}-f$ in openfoam. Master's thesis, Politecnico di Milano, 2009/2010.

[3] F. Billard. Development of a near-wall turbulence model and applications. 2011.

[4] F. Billard and D. Laurence. Near-wall turbulence rans modelling and its applications to industrial cases. 2007.

[5] F. Billard and D. Laurence. Development of a robust elliptic-blending turbulence model for near-wall, separated and buoyant flows. 2011.

[6] F. Billard and D. Laurence. A robust $k-\epsilon-\overline{v^2}/k$ elliptic blending turbulence model applied to near-wall, separated and buoyant flows. *International Journal of Heat and Fluid Flow*, 33(1):45–58, 2012.

[7] G. Bosch and W. Rodi. Simulation of vortex shedding past a square cylinder with different turbulence models. *International Journal for numerical methods in fluids*, 28(4):601–616, 1998.

[8] L. Davidson, P.V. Nielsen, and A. Sveningsson. Modifications of the v2 model for computing the flow in a 3d wall jet. 2003.

[9] P.A. Durbin. Near-wall turbulence closure modeling without 'damping functions'. *Theoretical and Computational Fluid Dynamics*, 3(1):1–13, 1991.

[10] P.A. Durbin. Application of a near-wall turbulence model to boundary layers and heat transfer. *International Journal of Heat and Fluid Flow*, 14(4):316–323, 1993.

[11] P.A. Durbin. On the $k-\epsilon$ stagnation point anomaly. *International journal of heat and fluid flow*, 17(1):89–90, 1996.

[12] K. Hanjalic, D. Laurence, M. Popovac, and J.C. Uribe. $(\overline{v^2}/k) - f$ turbulence model and its application to forced and natural convection. In *6th Int. Symp. Engineering Turbulence Modelling and Measurements*, pages 23–25, 2005.

[13] K. Hanjalić, M. Popovac, and M. Hadžiabdić. A robust near-wall elliptic-relaxation eddy-viscosity turbulence model for cfd. *International Journal of Heat and Fluid Flow*, 25(6):1047–1051, 2004.

[14] G. Kalitzin, G. Medic, G. Iaccarino, and P.A. Durbin. Near-wall behavior of rans turbulence models and implications for wall functions. *Journal of Computational Physics*, 204(1):265–291, 2005.

[15] F.S. Lien and P.A. Durbin. Non-linear $k - \overline{v'^2}$ modelling with application to high-lift. *Proceedings of the Summer Program Centre for Turbulence Research. Stanford University*, pages 5–22, 1996.

[16] F.S. Lien and G. Kalitzin. Computations of transonic flow with the $\overline{v'^2} - f$ turbulence model. *International Journal of Heat and Fluid Flow*, 22(1), 2001.

[17] CFD Open. Openfoam programmer's guide. *OpenFOAM Foundation*, 2011.

[18] CFD Open. Openfoam user guide. *OpenFOAM Foundation*, 2011.

[19] J.M. Österlund. Experimental studies of zero pressure-gradient turbulent boundary layer flow. 1999.

[20] Stephen B. Pope. Turbulent flows. 2000.

[21] M. Quadrio. Dispense del corso di turbolenza. 2012.

[22] J. Sillero, J. Jiménez, R.D. Moser, and N.P. Malaya. Direct simulation of a zero-pressure-gradient turbulent boundary layer up to $re_\theta = 6650$. In *Journal of Physics: Conference Series*, volume 318, page 022023. IOP Publishing, `http://torroja.dmt.upm.es/`, 2011.

[23] C.G. Speziale, S. Sarkar, and T.B. Gatski. Modelling the pressure–strain correlation of turbulence: an invariant dynamical systems approach. *Journal of Fluid Mechanics*, 227:245–272, 1991.

[24] A. Sveningsson and L. Davidson. Assessment of realizability constraints in v2-f turbulence models. *International journal of heat and fluid flow*, 25(5):785–794, 2004.

[25] K. Wieghardt and W. Tillman. On the turbulent friction layer for rising pressure. 1951.

Ringraziamenti

I ringraziamenti vanno doverosamente all'Ing. Luca Gasparini di FondTech s.r.l. e al Professor Maurizio Quadrio per aver reso possibile questo lavoro e per il supporto fornito con estrema competenza e rapidità durante lo svolgimento.

In secondo luogo ringrazio la mia famiglia che mi ha permesso di raggiungere questo traguardo, e mi ha supportato durante l'arco degli studi.

Ringrazio anche tutti gli amici, quelli di sempre, il gruppo dell'università e in generale tutti.