

OpenFOAM

introduzione al software e tutorials guidati

ConoscereLinux - Modena Linux User Group

Dr. D. Angeli

diego.angeli@unimore.it

Dr. M. Cavazzuti

marco.cavazzuti@unimore.it

Disclaimer

This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM® and OpenCFD® trade marks.

Sommario

1 Introduzione a OpenFOAM

2 Lid-Driven Cavity

3 Mixing Elbow

4 Hot Mixing Elbow

5 Micro-reference

OpenFOAM /1

- OpenFOAM (Open Field Operation and Manipulation, www.openfoam.com) è un toolbox C++, per la risoluzione di sistemi di equazioni alle derivate parziali tramite il metodo dei *Volumi Finiti*, su griglie poliedriche arbitrarie
- È sviluppato da OpenCFD Ltd., società facente parte di ESI Group, e distribuito dalla OpenFOAM Foundation (www.openfoam.org)
- Il toolbox è strutturato come un insieme di librerie C++, e di applicazioni che si fondano su di esse
- Le applicazioni includono strumenti di *pre-processing* (generatori di griglia, convertitori, manipolatori, . . .), *post-processing* (tramite il programma open source ParaView), e vari solutori CFD e non

OpenFOAM /2

- Una versione estesa di OpenFOAM è sviluppata e mantenuta dall'*OpenFOAM Extend Project* (www.extend-project.de). Essa integra i principali contributi derivanti dalla comunità di utenti e sviluppatori, nello spirito open source
- Le caratteristiche di OpenFOAM sono comparabili a quelle dei principali codici CFD commerciali
- La vera forza di OpenFOAM è la sua natura open source. Modificando il codice sorgente è possibile creare solutori, utilities e librerie ad-hoc, e controllare l'esatta implementazione di diverse funzionalità, cosa essenziale nell'ambito della ricerca
- Il maggior svantaggio è la curva di apprendimento ripida che lo caratterizza, anche a causa dell'assenza di una *Graphical User Interface* (GUI). Ma questa può essere al contempo un'opportunità

Principali Fonti di Documentazione

È vero, manca una guida ufficiale esaustiva di OpenFOAM, ma sono molte le fonti dove poter reperire le informazioni necessarie. Eccone alcune:

- la User's Guide e la Programmer's Guide allegati al codice: sono essenziali ma molto utili, e sono reperibili anche qui
foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf
foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf
- documentazione Doxygen, reperibile anche online all'indirizzo
www.openfoam.org/docs/cpp/
- i Tutorials forniti con l'installazione di OpenFOAM sono molto intuitivi e di facile interpretazione una volta che si hanno le basi
- OpenFOAM Wiki (www.openfoamwiki.net)
- OpenFOAM Forum su CFD Online (www.cfd-online.com/Forums/openfoam/)
- Il materiale di training degli OpenFOAM Workshops, tenuti con cadenza annuale, è disponibile a questa pagina
www.extend-project.de/events-and-meetings/detailed-categories/archive
- Il materiale del corso post-laurea di Håkan Nilsson alla Chalmers University (Svezia), è disponibile alla pagina
www.tfd.chalmers.se/~hani/kurser/OS_CFD/
- Il materiale dei corsi OpenFOAM di Joel Guerrero a UNIGE, è disponibile alla pagina www.dicat.unige.it/guerrero/openfoam.html

Installazione

OpenFOAM nasce per essere utilizzato su sistemi GNU/Linux. Esistono tre modi per installare il software (www.openfoam.org/download/)

- Installare il *pacchetto precompilato*
 - versioni precedenti alla 3.0+: in repository Ubuntu e OpenSUSE
 - dalla versione 3.0+: migrazione al sistema Docker (anche per MacOSx) e pacchetto Windows ufficiale
- Compilare *codice sorgente* della release
- Compilare il sorgente della *distribuzione Git* (ora *OpenFOAM-plus*) per bugfix e ultimi aggiornamenti

Modelli e Solutori

Principali modelli fisici e numerici e tipologie di solutori inclusi in OpenFOAM:

- Modelli di turbolenza RANS e LES
- Modelli termofisici
- Modelli di trasporto/reologici
- Modelli di cinetica chimica e combustione
- Mesh dinamiche
- Solutori di fluidodinamica incomprimibile
- Solutori di fluidodinamica comprimibile
- Solutori di fluidodinamica multifase
- Solutori con scambio termico
- Solutori con combustione
- Solutori di problemi con mesh dinamiche

Pre-Processing e Post-Processing

Nel pacchetto OpenFOAM sono incluse utilites per:

- Creare griglie
- Importare/esportare griglie da fonti esterne
- Visualizzare informazioni sulla griglia
- Controllare e migliorare la qualità della griglia
- Scalare, traslare, ruotare griglie
- Fondere, separare, eliminare zone
- Importare e reinterpolare dati da altri casi
- Calcolare medie temporali, minimi e massimi di variabili
- Calcolare medie e integrali di variabili su insiemi di facce/celle
- Campionare dati lungo punti e/o linee
- Generare superfici ausiliarie (piani, isosuperfici, . . .)
- Calcolare forze e coefficienti di attrito, pressione, portanza, . . .

Per il post-processing grafico, è incluso nel pacchetto il programma open source ParaView

Settare l'Ambiente di OpenFOAM

Per agevolare l'utilizzo di OpenFOAM è opportuno eseguire il *sourcing* dello *script*

- `. //etc/bashrc`

L'operazione permette di impostare una serie di *alias* (ossia comandi abbreviati) e *variabili d'ambiente* (contenenti principalmente percorsi di cartelle) funzionali all'uso comune del software.

Inoltre, lo script inserisce nella variabile d'ambiente \$PATH (che contiene il percorso degli eseguibili richiamabili a riga di comando) il percorso di tutte le applicazioni eseguibili di OpenFOAM.

Principali Variabili di Ambiente e Alias

Le variabili d'ambiente si dividono tra cartelle di progetto e cartelle utente

Variabile	Percorso Cartella	Alias
● \$FOAM_INST_DIR	/opt/OpenFOAM	
● \$WM_PROJECT_DIR	/opt/OpenFOAM/OpenFOAM-v3.0+	foam
● \$WM_THIRD_PARTY_DIR	/opt/OpenFOAM/ThirdParty-v3.0+	
● \$FOAM_TUTORIALS	/opt/OpenFOAM/OpenFOAM-v3.0+/tutorials	tut
● \$FOAM_APP	/opt/OpenFOAM/OpenFOAM-v3.0+/applications	app
● \$FOAM_SOLVERS	/opt/OpenFOAM/OpenFOAM-v3.0+/applications/solvers	sol
● \$FOAM_UTILITIES	/opt/OpenFOAM/OpenFOAM-v3.0+/applications/utilities	util
● \$FOAM_SRC	/opt/OpenFOAM/OpenFOAM-v3.0+/src	src
● \$FOAM_APPBIN	\$WM_PROJECT_DIR/platforms/linux64GccDPInt320pt/bin	
● \$FOAM_LIBBIN	\$WM_PROJECT_DIR/platforms/linux64GccDPInt320pt/lib	
● \$WM_PROJECT_USER_DIR	\$HOME/OpenFOAM/username-v3.0+	
● \$FOAM_RUN	\$HOME/OpenFOAM/username-v3.0+/run	run

I percorsi cartella cui le variabili si riferiscono possono variare a seconda del percorso di installazione effettivo di OpenFOAM.

Le cartelle utente vengono create col comando

- `mkdir -p $FOAM_RUN`

Organizzazione delle Cartelle /1

Analizziamo la struttura delle cartelle di OpenFOAM col comando

- `tree -L 1 -d $WM_PROJECT_DIR`

Cartella	Contenuto
<code>\$WM_PROJECT_DIR</code>	
- <code>applications</code>	codice sorgente di solvers e utilities
- <code>bin</code>	shell scripts di uso generale
- <code>doc</code>	documentazione
- <code>etc</code>	file di setup dell'ambiente e altro
- <code>platforms</code>	file binari di applicazioni e librerie
- <code>src</code>	codice sorgente delle librerie
- <code>tutorials</code>	casi di esempio per la maggior parte delle applicazioni
- <code>wmake</code>	impostazioni per la compilazione

Organizzazione delle Cartelle /2

Allo stesso modo diamo uno sguardo alla cartella \$FOAM_SOLVERS col comando

- `tree -L 1 -d $FOAM_SOLVERS`

I solutori sono catalogati secondo la classe di problemi fisici a cui si riferiscono, troviamo così le seguenti sottocartelle:

- basic
- combustion
- compressible
- discreteMethods
- DNS
- electromagnetics
- financial
- heatTransfer
- incompressible
- lagrangian
- multiphase
- stressAnalysis

La stessa suddivisione si trova nella cartella dei tutorials \$FOAM_TUTORIALS

Queste cartelle contengono le cartelle dei solutori, le quali a loro volta contengono files con lo stesso nome del solutore e con estensione .C. Questi sono i sorgenti del main dei solutori e includono anche una breve descrizione del solver stesso.

Ad esempio, il file

- `$FOAM_APP/solvers/incompressible/icoFoam/icoFoam.C`

sorgente del solutore icoFoam, contiene la descrizione

- transient solver for incompressible, laminar flow of Newtonian fluids

Setup ed Esecuzione di un Caso

I passi da seguire per la risoluzione di un'analisi CFD con OpenFOAM sono:

- Costruzione della griglia
- Scelta del solutore opportuno
- Impostazioni delle condizioni iniziali ed al contorno necessarie
- Impostazioni dei *dictionaries*
- Lancio del calcolo
- Controllo dei residui della simulazione
- Post-processing dei dati

Struttura di un Caso

- A differenza di altre applicazioni, come molti codici commerciali, in OpenFOAM la griglia, le impostazioni e i risultati sono suddivisi tra un certo numero di files
- Il caso è memorizzato in una cartella con struttura fissa, contenente:

`case-dirname`

- | - 0 definizione delle condizioni iniziali e al contorno
- | - constant scelta dei modelli e definizione delle proprietà dei materiali
 - | | - polyMesh files della mesh
- | - system impostazioni del solutore (schemi di discretizzazione, tolleranze di convergenza, ...)
- | - iter-number cartelle dei risultati della simulazione organizzate in base ai diversi istanti di tempo o alle iterazioni

- Tipicamente, ad ogni oggetto ad alto livello (un campo scalare o vettoriale, un entità geometrica, ...) corrisponde un file

Formato dei Dati

- Il principale sistema di input in OpenFOAM è il *dictionary*, ovvero liste di coppie *keyword / value*, in cui l'ordine delle voci è indifferente e sono ammessi *dictionaries* annidati (delimitati da parentesi graffe)

Si noti come in molti codici commerciali, dietro all'interfaccia grafica, i dati vengono gestiti allo stesso modo

- Il *dictionary* principale, che controlla l'esecuzione della simulazione, è il `controlDict` nella cartella `system`
- La definizione degli altri *dictionaries* può variare a seconda dei solutori usati
- Esempio: nel solutore `icoFoam` il *dictionary* in cui vengono definite le proprietà di trasporto del fluido si chiama `transportProperties` e si trova nella cartella `constant`. Al suo interno si trova la definizione della viscosità cinematica del fluido ν come
- `nu [0 2 -1 0 0 0 0] 0.01;`
- Il modo migliore per imparare ad usare OpenFOAM resta quello di scorrere qualche tutorial. Nel resto di questa dispensa analizzeremo in dettaglio tre casi applicativi

Sommario

- 1 Introduzione a OpenFOAM
- 2 Lid-Driven Cavity
- 3 Mixing Elbow
- 4 Hot Mixing Elbow
- 5 Micro-reference

Tutorial 1: Lid-Driven Cavity

- Oggetto dell'analisi:

Flusso laminare in una cavità bi-dimensionale quadrata indotto dal moto della parete superiore della cavità stessa

- Dimensioni e proprietà:

Dominio 0.1×0.1 m

Velocità della parete $U = 1.0$ m/s

Viscosità cinematica $\nu = 0.01$ m²/s

Numero di Reynolds $Re = \frac{UL}{\nu} = 10$

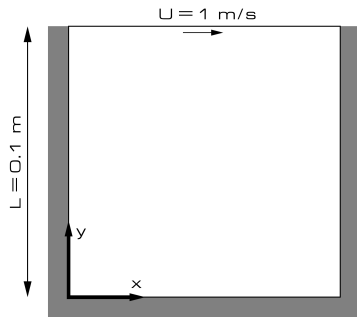
- Impostazioni del calcolo:

Tempo di simulazione $t \in [0, 0.05]$ s

Passo temporale $\Delta t = 0.005$ s

Griglia di calcolo $20 \times 20 \times 1$

Solutore icoFoam



Cavity Tutorial: Impostazione del Caso

Questo esempio è già impostato nel tutorial cavity del solutore icoFoam, recuperiamo quindi i file del tutorial copiandoli nella cartella utente run, per poi accedervi

- `cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity $FOAM_RUN`
- `cd $FOAM_RUN/cavity`

La struttura della cartella è quella già vista nelle slides precedenti, eccetto il fatto che la cartella polyMesh non esiste ancora in quanto non è ancora stata generata la mesh del caso

- `cavity`
 - | - 0
 - | - constant
 - | - system

Cavity Tutorial: blockMeshDict /1

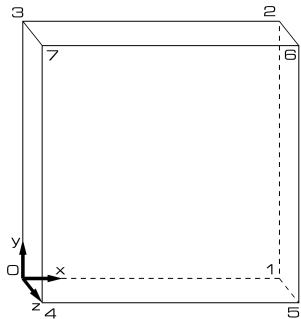
La cartella `system` contiene il `blockMeshDict`, dictionary necessario per la generazione della mesh con l'applicazione `blockMesh`, un generatore di mesh strutturate molto basilare incluso in `OpenFOAM`. Vediamo la struttura del file

- `cat constant/polyMesh/blockMeshDict`

Il file è così strutturato:

- definizione di un fattore di scala
`convertToMeters 0.1;`
- definizione dei vertici del dominio
(vedi figura a lato)

```
vertices
(
  (0 0 0)
  (1 0 0)
  (1 1 0)
  (0 1 0)
  (0 0 0.1)
  (1 0 0.1)
  (1 1 0.1)
  (0 1 0.1)
);
```



Cavity Tutorial: blockMeshDict /2

- definizione dei blocchi 3D (formati da 8 vertici ciascuno)

Nota: OpenFOAM usa sempre griglie 3D, quando la simulazione è di tipo 2D la griglia deve comunque essere 3D ed avere 1 sola cella di spessore

```
blocks
```

```
(
```

```
  hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
```

```
);
```

- la keyword `hex` indica che il blocco è esaedrico e strutturato
- `(0 1 2 3 4 5 6 7)` è la lista dei vertici che definiscono il blocco. Il loro ordine deve essere tale da formare un sistema destrorso
- `(20 20 1)` indica il numero di elementi in ogni direzione ($x y z$)
- `simpleGrading (1 1 1)` indica il rapporto di espansione degli elementi (ampiezza dell'ultima cella fratto ampiezza della prima). Un valore unitario indica quindi che la mesh è uniforme in una data direzione

Cavity Tutorial: blockMeshDict /3

- definizione di eventuali spigoli non rettilinei nei blocchi (nessuno nel nostro caso)

```
edges
(
);
```

- definizione dei tipi di condizioni al contorno sulle facce del dominio (vedi a destra)

ad ogni *patch* sono associati: un tipo (e.g. wall), un nome (e.g. fixedWalls), una lista di facce con i vertici ordinati in senso orario quando visti dall'interno del blocco. In un caso 2D le facce nella terza direzione devono essere di tipo empty

- definizione di eventuali interfacce sovrapposte (utilizzato in domini multiblocco, ove le interfacce tra blocchi attigui siano definite tramite vertici distinti)

```
mergePatchPairs
(
);
```

```
boundary
(
  movingWall
  {
    type wall;
    faces
    (
      (3 7 6 2)
    );
  }
  fixedWalls
  {
    type wall;
    faces
    (
      (0 4 7 3)
      (2 6 5 1)
      (1 5 4 0)
    );
  }
  frontAndBack
  {
    type empty;
    faces
    (
      (0 3 2 1)
      (4 5 6 7)
    );
  }
);
```

Cavity Tutorial: Generazione della Mesh

- dalla cartella `$FOAM_RUN/cavity` si genera la mesh digitando il comando

`blockMesh`

con questa operazione viene creata cartella `constant/polyMesh` (nel caso non esistesse già) ed al suo interno vengono creati alcuni files che definiscono punti, facce, connettività delle celle della mesh

```
boundary    faces    neighbour
owner      points
```

- è di interesse il file `boundary` in quanto mostra le definizioni delle *patches* (vedi a destra)
- si può controllare la validità e la bontà della griglia generata col comando `checkMesh`

```
3
(
movingWall
{
    type wall;
    inGroups 1(wall);
    nFaces 20;
    startFace 760;
}
fixedWalls
{
    type wall;
    inGroups 1(wall);
    nFaces 60;
    startFace 780;
}
frontAndBack
{
    type empty;
    inGroups 1(empty);
    nFaces 800;
    startFace 840;
}
)
```

Cavity Tutorial: Cartelle constant e system

- La cartella `constant`, oltre alla sottocartella `polyMesh`, contiene il file `transportProperties`, `dictionary` per lo scalare dimensionale viscosità cinematica (ν), unica proprietà richiesta dal solutore `icoFoam`

```
nu [ 0 2 -1 0 0 0 0 ] 0.01;
```

i numeri tra parentesi individuano l'unità di misura della grandezza. Ogni valore corrisponde alla potenza cui vanno elevate le unità base del SI secondo l'ordine [kg m s K kgmol A cd]. In questo caso siamo quindi davanti ad una grandezza che si esprime in m^2/s
- La cartella `system`, oltre al file `blockMeshDict` già visto, contiene i tre files:
 - `controlDict` definisce i controlli generali per l'esecuzione del caso
 - `fvSchemes` definisce gli schemi di discretizzazione da usare per i diversi termini delle equazioni
 - `fvSolution` definisce i metodi risolutivi per i sistemi lineari associati alle equazioni discrete, e le istruzioni per l'algoritmo di accoppiamento velocità-p pressione (PISO in questo caso)

Cavity Tutorial: controlDict

- Il file controlDict è formato come segue:

application	icoFoam;	nome del solutore
startFrom	startTime;	criterio di inizio simulazione
startTime	0;	tempo iniziale della simulazione
stopAt	endTime;	criterio di fine simulazione
endTime	0.5;	tempo finale della simulazione
deltaT	0.005;	passo temporale
writeControl	timeStep;	criterio per la scrittura dei risultati
writeInterval	20;	intervallo di scrittura dei risultati
purgeWrite	0;	eliminare i salvataggi precedenti? 0=no, n=mantieni gli ultimi n
writeFormat	ascii;	formato dei files dei risultati
writePrecision	6;	cifre significative usate nei risultati
writeCompression	off;	compressione dei files di output
timeFormat	general;	formato delle cartelle dei risultati
timePrecision	6;	cifre significative della variabile tempo
runTimeModifiable	true;	controllo a <i>run-time</i> del calcolo

Cavity Tutorial: fvSchemes

Il file `fvSchemes` definisce gli schemi di discretizzazione, nel nostro caso (vedi a destra):

- discretizzazione temporale Euleriana implicita
- discretizzazione dei gradienti e del termine convettivo con schema lineare (alle differenze centrali)
- discretizzazione del termine diffusivo con schema lineare e senza correzioni non ortogonali dei gradienti normali alle superfici
- interpolazione lineare dei valori alle facce delle celle
- discretizzazione dei gradienti normali alle superfici di tipo ortogonale

```

ddtSchemes
{
    default Euler;
}
gradSchemes
{
    default Gauss linear;
    grad(p) Gauss linear;
}
divSchemes
{
    default none;
    div(phi,U) Gauss linear;
}
laplacianSchemes
{
    default Gauss
        linear orthogonal;
}
interpolationSchemes
{
    default linear;
}
snGradSchemes
{
    default orthogonal;
}

```

Cavity Tutorial: fvSolution

Il file `fvSolution` definisce le procedure di risoluzione, si sottolinea quanto segue:

- all'interno di ogni iterazione le equazioni (e.g. pressione `p`, o velocità `U`) vengono considerate arrivate a convergenza se sono raggiunti residui dell'ordine della `tolerance` imposta o se i residui si riducono rispetto al valore di partenza a un fattore pari a `relTol`. Col valore 0 tale controllo è disabilitato
- nel file `fvSolution` è anche possibile definire coefficienti di rilassamento o criteri di convergenza globali (cosa omessa in questo tutorial ma che vedremo in seguito)

```
solvers
{
  p
  {
    solver PCG;
    preconditioner DIC;
    tolerance 1e-06;
    relTol 0;
  }
  U
  {
    solver smoothSolver;
    smoother symGaussSeidel;
    tolerance 1e-05;
    relTol 0;
  }
}
PISO
{
  nCorrectors 2;
  nNonOrthogonalCorrectors 0;
  pRefCell 0;
  pRefValue 0;
}
```

Cavity Tutorial: Cartella 0

La cartella 0 contiene le condizioni iniziali e al contorno per tutte le variabili primarie e per ogni *patch* definita. Nel nostro caso queste si riducono al campo scalare di pressione p e al campo vettoriale di velocità U . Consideriamo il file U:

- si stabilisce che la variabile U ha la dimensione di una velocità (m/s)
- viene inizializzato il campo di moto (`internalField`) a 0 m/s
- vengono fissati (`fixedValue`) i valori di velocità alle *patches* di *boundary*: velocità uniforme di 1 m/s lungo x per la `movingWall`, 0 m/s per la `fixedWalls`
- il tipo `empty` associato alla *patch* `frontAndBack` indica che non è richiesta la soluzione in direzione z , essendo il caso 2D
- analogamente nel file 0/p vengono imposti gradienti di pressione nulli alle pareti

```
dimensions [0 1 -1 0 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
    movingWall
    {
        type fixedValue;
        value uniform (1 0 0);
    }
    fixedWalls
    {
        type fixedValue;
        value uniform (0 0 0);
    }
    frontAndBack
    {
        type empty;
    }
}
```

Cavity Tutorial: Esecuzione

- come già specificato, il caso `cavity` è un tutorial per il solver `icoFoam`. Per eseguirlo, occorre lanciare, dalla cartella radice del caso, il comando:

```
icoFoam | tee log
```

- il piping (`|`) ridireziona l'output di `icoFoam` al comando `tee`, il quale a sua volta lo invia sia al terminale, sia ad un file `log`, utile a tenere traccia dell'andamento della simulazione
- se si controlla il contenuto della cartella `cavity`, si nota che essa, una volta completato il calcolo, contiene 5 nuove sottocartelle `0.1`, `0.2`, `0.3`, `0.4`, `0.5`, contenenti i risultati (ovvero i campi di p e U) ai rispettivi time step
- il contenuto di tali cartelle è del tutto simile a quello della cartella `0`, ma, per ciascuna delle variabili, l'`internalField` è ora una `nonuniform List<scalar/vector>` contenente i risultati cella per cella
- in esse compare anche un file `phi`, contenente i flussi che attraversano le facce delle celle, necessari per un eventuale restart della simulazione
- compaiono inoltre informazioni riguardanti il tempo nel file `uniform/time`

Cavity Tutorial: log

Il file log contiene indicazioni sui numeri di Courant e i residui ad ogni time step:

```
Time = 0.15
Courant Number mean: 0.22205 max: 0.852089
smoothSolver: Solving for Ux, Initial residual = 3.17056e-05, Final residual = 6.41062e-06, No Iterations 3
smoothSolver: Solving for Uy, Initial residual = 6.70591e-05, Final residual = 7.81984e-06, No Iterations 4
DICPCG: Solving for p, Initial residual = 5.56904e-05, Final residual = 6.02159e-07, No Iterations 12
time step continuity errors : sum local = 6.85695e-09, global = 1.11744e-18, cumulative = 1.67107e-18
DICPCG: Solving for p, Initial residual = 4.22564e-05, Final residual = 8.09147e-07, No Iterations 11
time step continuity errors : sum local = 9.0193e-09, global = -4.94654e-20, cumulative = 1.62161e-18
ExecutionTime = 0.05 s ClockTime = 0 s
```

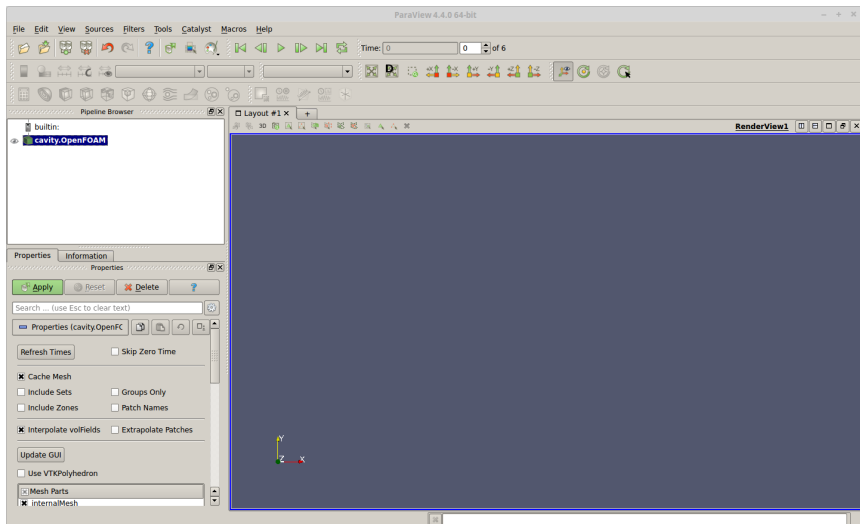
Per quanto riguarda il time step a 0.15 s, per esempio, si vede che

- per U_x e U_y è stato utilizzato il solutore `smoothSolver`, che è arrivato a convergenza in 3 e 4 iterazioni rispettivamente
- il residuo iniziale è calcolato prima della soluzione del sistema lineare, quello finale dopo ed è sempre inferiore alla tolleranza indicata nel file `fvSolution` (`tolerance 1e-05`)
- i residui di p e gli errori di continuità sono riportati due volte in quanto nel file `fvSolution`, tra i controlli PISO, sono prescritti 2 passi di correzione della pressione (`nCorrectors 2`)

Cavity Tutorial: Post-Processing /1

- `paraFoam` è il principale post-processor distribuito con OpenFOAM
- il comando `paraFoam` è in realta un *wrapper* (un'interfaccia) al prodotto di terze parti `ParaView`
- nella pratica, lo script `paraFoam` genera un file `casename.OpenFOAM` necessario per aprire correttamente i file della griglia e dei dati, e per modificare l'interfaccia grafica di `ParaView` secondo le specifiche di OpenFOAM
- per post-processare il caso `cavity` digitare
`paraFoam`
- Il nome del caso appare nell'oggetto `cavity.OpenFOAM` del Pipeline Browser
- L'oggetto può essere controllato usando il menu `Object Inspector`
- Si noti che nella cartella del caso appare un file temporaneo chiamato `cavity.OpenFOAM`

Cavity Tutorial: Post-Processing /2



Cavity Tutorial: Post-Processing /3

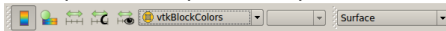
Object Inspector: *Properties*

- nella casella **Mesh Parts** si selezionano le parti della mesh che si desidera importare (e.g. *patches, groups e/o internal mesh*)
 - nella casella **Volume Fields** si selezionano i campi di variabili che si desidera importare (e.g. *p, U*)
 - selezionando **Patch Names** nella parte alta dell'**Object Inspector** i nomi delle *patches* selezionate vengono visualizzati nella finestra di visualizzazione
 - per leggere mesh e dati cliccare su **Apply**
 - se nel frattempo il contenuto della cartella del caso è cambiato (e.g. il calcolo è in corso ed ha effettuato un nuovo salvataggio) selezionare **Update GUI** e **Apply** o **Refresh Times** per importare nuovamente i dati
- ⇒ selezionare tutte le **Mesh Parts** e i **Volume Fields** nel tab *Properties*, poi cliccare **Apply**

Cavity Tutorial: Post-Processing /4

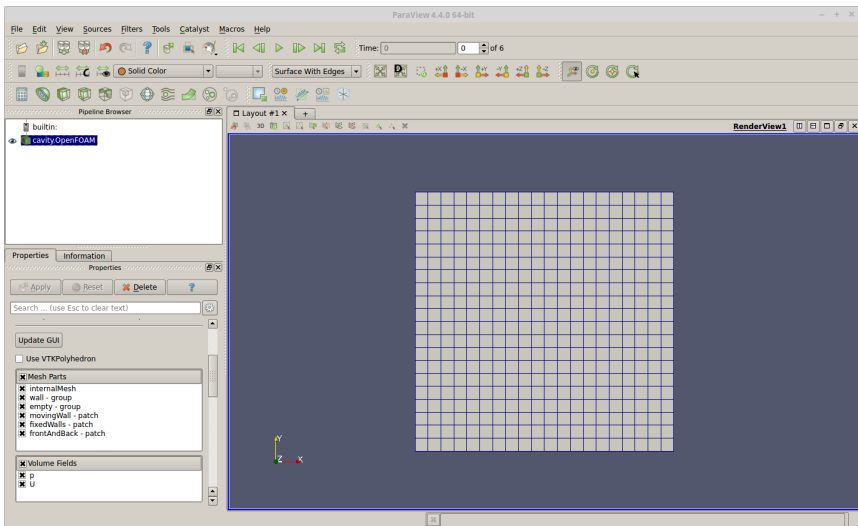
Object Inspector: *Properties*

- la parte bassa del menu *Properties* mostra i controlli della visualizzazione
- l'opzione *Representation* permette la scelta del tipo di rappresentazione dell'oggetto, e.g. *outline*, *wireframe*, *surface*, ...
- il menu *Coloring* permette la colorazione dell'oggetto nella finestra di visualizzazione utilizzando i campi delle variabili (si possono scegliere sia i valori di cella, calcolati, o i valori sui punti di griglia, interpolati); il menu permette inoltre di configurare la legenda
- i menu *Styling* e *Lighting* gestiscono la trasparenza e la luminosità
- l'estensione dell'oggetto può essere visualizzata con *Edit Axes Grid*
- il menu *Background* permette di gestire la colorazione dello sfondo
- alcune delle opzioni presenti in questo tab possono essere raggiunte anche con la toolbar






- ⇒ selezionare *Surface with edges* nel menu *Representation* e *Solid Color* nel menu *Coloring*

Cavity Tutorial: Post-Processing /5



Cavity Tutorial: Post-Processing /6

Finestra di Visualizzazione:





- per ruotare la visuale muovere il mouse tenendo premuto il tasto sinistro
- per zoomare la visuale muovere il mouse su e giù tenendo premuto il tasto destro
- per spostare la visuale muovere il mouse tenendo premuto il tasto centrale
- le icone in alto a sinistra  servono per impostare visuali personalizzate o di default ed effettuare selezioni mirate di celle, punti, blocchi
- le icone in alto a destra  servono per suddividere la finestra in orizzontale o in verticale, o anche massimizzare, ripristinare, chiudere le finestre (solo con più finestre attive)
- alcuni controlli per la gestione rapida della visuale si trovano anche nella toolbar 

Cavity Tutorial: Post-Processing /7

Il menu **Filters/Alphabetical** contiene le opzioni per manipolare e gestire i dati.

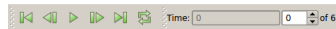
Le più comuni si trovano nella toolbar



-  **Calculator:** crea nuovi campi da variabili esistenti utilizzando semplici funzioni matematiche
-  **Contour:** visualizza isolinee (in 2D) e isosuperfici (in 3D)
-  **Clip:** taglia l'oggetto visualizzato usando un piano, un parallelepipedo, una sfera, o il valore di uno scalare
-  **Slice:** crea piani di taglio, definibili in diversi modi, su cui visualizzare i risultati

Selezionando un filtro, si genera un nuovo oggetto nel **Pipeline Browser** a partire dall'oggetto attivo.

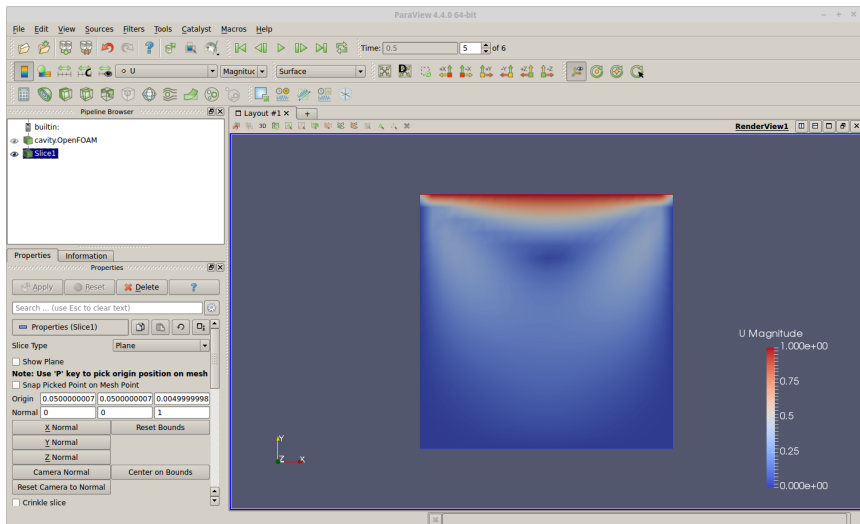
La toolbar







permette di spostarsi tra i diversi istanti di tempo salvati

- ⇒ portarsi all'ultimo istante di tempo
- ⇒ selezionare l'oggetto `cavity.OpenFOAM` nel **Pipeline Browser**
- ⇒ selezionare il filtro **Slice** e impostare il piano di taglio come **Z-normal**
- ⇒ visualizzare come **Surface**, colorare secondo la velocità nei nodi
- ⇒ visualizzare la legenda e disattivare la visualizzazione del piano di taglio

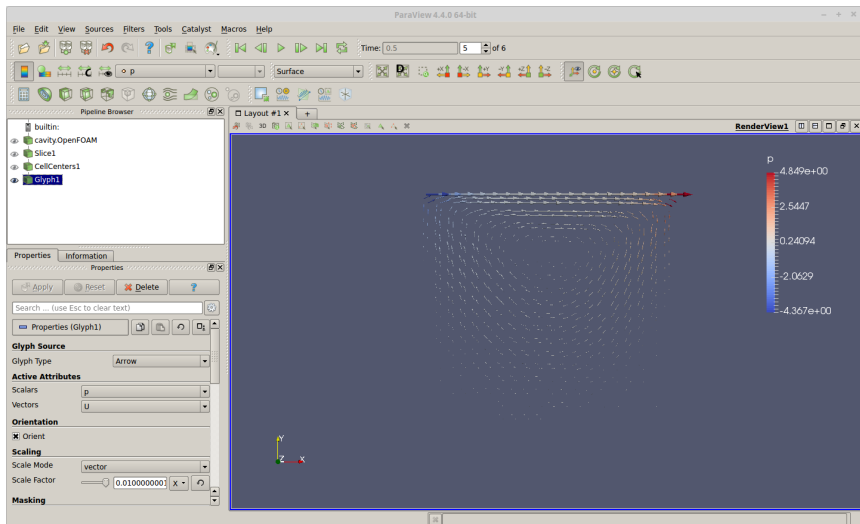
Cavity Tutorial: Post-Processing /8



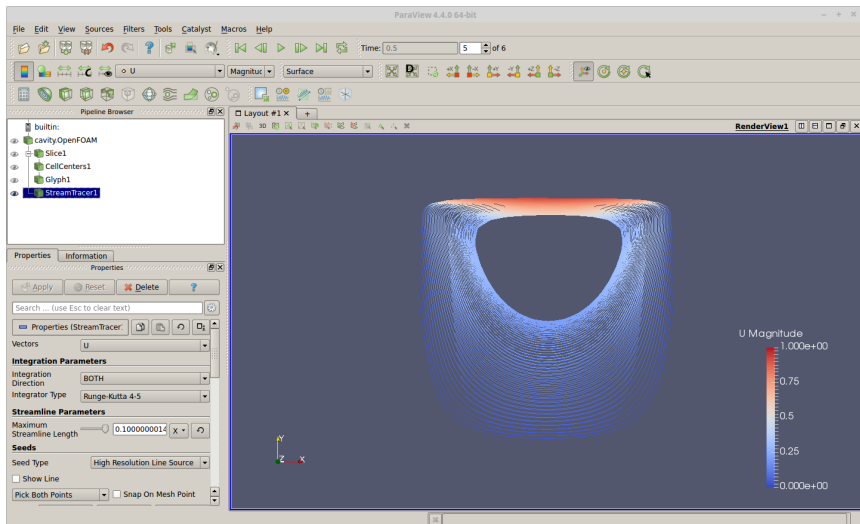
Cavity Tutorial: Post-Processing /9

-  **Threshold:** visualizza le celle aventi valori di un certo scalare entro un determinato range
 -  **Glyph:** crea plot di vettori. Di default i vettori sono creati nei punti di griglia, per crearli nei centri cella è necessario generare prima un oggetto con il filtro `Cell Centers`. È possibile variare le opzioni di creazione dei vettori nel tab `Properties`
 -  **Stream Tracer:** crea linee di corrente a partire da un punto, una linea, o una nuvola di punti
 -  **Warp:** distorce la geometria secondo il valor locale di uno scalare (utile, e.g., in meccanica dei solidi per visualizzare le deformazioni di un oggetto)
- ⇒ selezionare l'oggetto `Slice1` nel `Pipeline Browser`, applicare il filtro `Cell Centers`, e togliere la visualizzazione della legenda
- ⇒ selezionare l'oggetto `Cell Centers`, applicare il filtro `Glyph` con `scale mode` di tipo `vector`, e visualizzare la legenda. Otteniamo un plot di vettori colorati secondo la p e di lunghezza proporzionale al modulo della \mathbf{U}
- ⇒ selezionare l'oggetto `cavity.OpenFOAM` e selezionare il filtro `Stream Tracer`, impostare il `seed type` come `line source`, `X-axis`, colorare secondo la velocità, disattivare la visualizzazione della linea sorgente e cliccare `Apply`
- ⇒ disattivare la visualizzazione dei precedenti filtri `Cell Centers` e `Glyph`, selezionare l'oggetto `StreamTracer1` e visualizzare la legenda

Cavity Tutorial: Post-Processing /10



Cavity Tutorial: Post-Processing /11



Cavity Tutorial: Post-Processing /12

Output:

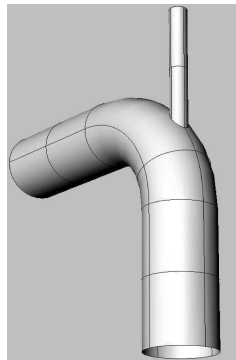
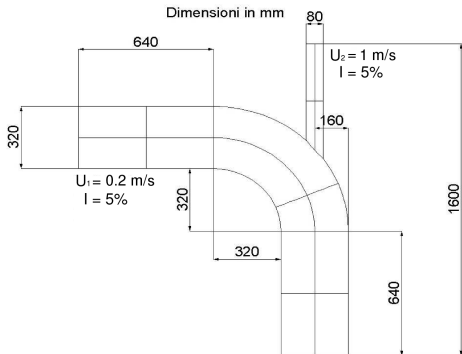
- l'opzione `File/Save Screenshot` permette di salvare immagini col contenuto della finestra di visualizzazione in vari formati
- la risoluzione dell'immagine è selezionabile a piacimento
- l'opzione `File/Save Animation` salva il frame del contenuto della finestra di visualizzazione per ogni istante di tempo disponibile. I frame potranno poi essere convertiti in animazioni mediante applicazioni esterne
- l'opzione `File/Save State` permette di salvare lo stato di visualizzazione (sequenza di oggetti nel `Pipeline Browser` e viste definite), che può poi essere ricaricato con l'opzione `File/Load State`. L'opzione è utile qualora vi sia la necessità di ripetere più volte la stessa visualizzazione

Sommario

- 1 Introduzione a OpenFOAM
- 2 Lid-Driven Cavity
- 3 **Mixing Elbow**
- 4 Hot Mixing Elbow
- 5 Micro-reference

Tutorial 2: Mixing Elbow

- Oggetto dell'analisi: flusso turbolento di due correnti di acqua che in un condotto si miscelano in corrispondenza di una curva a 90°
- Dimensioni e Boundary Conditions come da immagine
- $\nu = 1 \times 10^{-6} \text{ m}^2/\text{s}$
- Solutore simpleFoam



Mixing Elbow: Impostazione del Caso /1

A differenza del precedente caso, il caso in esame non è compreso tra i tutorial di OpenFOAM (sebbene se ne trovi una versione 2D tra i tutorial di icoFoam). Si procederà quindi al setup manuale del caso.

Per impostare un nuovo caso di OpenFOAM è consigliabile partire replicando la cartella di un tutorial relativo al solutore che si intende impiegare. In questo caso si partirà del tutorial pitzDailyExptInlet di simpleFoam.

- `cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDailyExptInlet \`
`$FOAM_RUN/elbow`
- `cd $FOAM_RUN/elbow`

A questo punto è possibile cancellare la cartella constant/boundaryData relativa al tutorial di partenza.

- `rm -r constant/boundaryData`

Mentre per il caso precedente, data la semplicità della geometria del dominio, era stato possibile creare la griglia mediante l'utility blockMesh. Per il caso in esame è consigliabile utilizzare strumenti di terze parti.

La generazione della mesh è lasciata all'utente.

Mixing Elbow: Impostazione del Caso /2

Si supponga di avere già nella propria home un file di griglia `elbow.msh`, compatibile con il software di calcolo ANSYS Fluent. Per prima cosa occorre spostare il file nella cartella del caso

- `mv $HOME/elbow.msh .`

Il file `msh` per essere leggibile da OpenFOAM deve essere salvato in formato ASCII, da ANSYS Meshing selezionare

- Tools → Options → Meshing → Export → Format of Input File → ASCII

Qualora fosse presente una mesh precedente occorre eliminarla

- `rm constant/polyMesh/*`

Possiamo quindi convertire la mesh `.msh` in formato OpenFOAM con l'utility

- `fluent3DMeshToFoam elbow.msh`

Nel caso la griglia sia costruita in un'unità di misura differente dal metro, occorre scalarla all'atto della conversione; e.g., se la mesh è costruita in millimetri

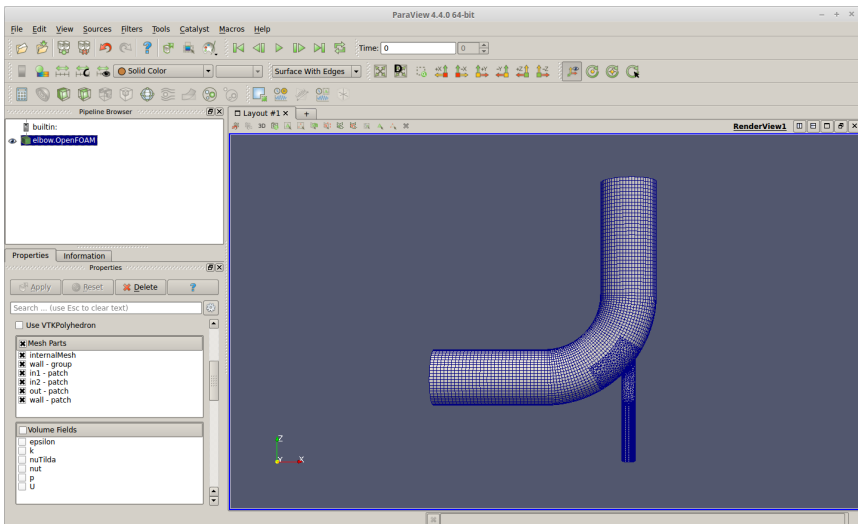
- `fluent3DMeshToFoam -scale 0.001 elbow.msh`

Si verifichi ora la mesh col comando

- `checkMesh`

e la si esamini mediante `paraFoam`, importando tutte le Mesh Parts, ma omettendo i Volume Fields (perché non ancora settati per il caso in esame)

Mixing Elbow: Impostazione del Caso /3



Mixing Elbow: Impostazione del Caso /4

Visualizzando il contenuto della cartella `constant/polyMesh` con

- `ls constant/polyMesh`

si nota che ora essa contiene alcuni files generati dalla conversione della mesh

- `boundary` `cellZones` `faces` `faceZones`
`neighbour` `owner` `points` `pointZones`

Il file `boundary` riveste particolare importanza perché contiene le informazioni relative alle `patches` di contorno della griglia, sulle quali andranno imposte le condizioni al contorno. Sulla base di queste andranno impostati i files nella cartella `0`

Mixing Elbow: Impostazione del Caso /5

Il file `constant/polyMesh/boundary` contiene 4 patches (definite durante la costruzione della griglia)

- la patch `wall` corrisponde alle pareti laterali del dominio
- la patch `out` corrisponde alla sezione di outlet
- le patches `in1` e `in2` corrispondono alle due sezioni di ingresso del fluido
- la patch `wall` eredita il `type wall` dalle impostazioni del pre-processore
- alle altre patches è assegnato il generico `type patch`

```

4
(
  in1
  {
    type patch;
    nFaces 525;
    startFace 331192;
  }
  in2
  {
    type patch;
    nFaces 525;
    startFace 331717;
  }
  out
  {
    type patch;
    nFaces 525;
    startFace 332242;
  }
  wall
  {
    type wall;
    inGroups 1(wall);
    nFaces 11586;
    startFace 332767;
  }
)

```

Mixing Elbow: Impostazione del Caso /6

Cartella 0:

- visualizzando il contenuto della cartella 0 (ls -l 0/), si vede che essa contiene i files `epsilon k nut nuTilda p U`
- i files `p` e `U` contengono le condizioni al contorno di pressione e velocità
- nella riga di descrizione del file `$FOAM_APP/solvers/incompressible/simpleFoam/simpleFoam.C` si legge:
`Steady-state solver for incompressible, turbulent flow`
- `simpleFoam` risolve flussi turbolenti, in regime stazionario, sotto l'ipotesi di incomprimibilità, utilizzando modelli di turbolenza `RANS`
- i files `epsilon`, `k`, `nut`, `nuTilda` rappresentano variabili associate ai vari modelli di turbolenza
- per la risoluzione del presente caso si utilizzerà il modello di turbolenza `k-ε` cui sono associate le variabili `epsilon`, `kappa` e `nut`
- nella cartella 0 si può quindi eliminare il file `nuTilda`
`rm 0/nuTilda`
- gli altri files andranno modificati come segue, in base alle patches presenti nel file `constant/polyMesh/boundary`

Mixing Elbow: Impostazione del Caso /7

File 0/U:

- alle patches `in1` e `in2` sono assegnate rispettivamente le velocità di 0.2 m/s e 1.0 m/s
- le normali alle patches entranti nel dominio sono allineate con le direzioni x e z positive, da cui i vettori $(0.2 \ 0 \ 0)$ e $(0 \ 0 \ 1)$
- alle patch di tipo `wall` si impone un vettore velocità nullo per l'ipotesi di aderenza
- sulla superficie di outlet `out` si impone un gradiente di velocità nullo
- il campo di velocità interno è inizializzato pari al valore alla patch `in1` per facilitare la convergenza

```

dimensions [0 1 -1 0 0 0 0];
internalField uniform (0.2 0 0);
boundaryField
{
    in1
    {
        type fixedValue;
        value uniform (0.2 0 0);
    }
    in2
    {
        type fixedValue;
        value uniform (0 0 1);
    }
    out
    {
        type zeroGradient;
    }
    wall
    {
        type fixedValue;
        value uniform (0 0 0);
    }
}

```

Mixing Elbow: Impostazione del Caso /8

File 0/p:

- alle patches `in1` e `in2`, essendo già stato specificato un valore di velocità, si impone che la pressione abbia un gradiente nullo
- anche alla patch `wall` si impone che il gradiente di pressione sia nullo
- sulla superficie di `outlet` (patch `out`) si fissa un valore di pressione nullo
- si noti che trattandosi di un flusso incomprimibile, la pressione è da intendersi come relativa

```

dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
{
    in1
    {
        type zeroGradient;
    }
    in2
    {
        type zeroGradient;
    }
    out
    {
        type fixedValue;
        value uniform 0;
    }
    wall
    {
        type zeroGradient;
    }
}

```

Mixing Elbow: Impostazione del Caso /9

Condizioni al contorno turbolente:

- il modello $k-\varepsilon$ prevede la risoluzione di due equazioni di trasporto di scalari:
energia cinetica turbolenta k
velocità di dissipazione turbolenta ε
- la risoluzione dei campi di k e ε permette di determinare il campo della eddy viscosity ν_t

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} = C_\mu^{\frac{1}{4}} \sqrt{\frac{3}{2}} UI \ell_t$$

dove $C_\mu = 0.09$ è una costante del modello

- in corrispondenza di un inlet si può definire k in base alla turbulent intensity I e alla velocità di ingresso U

$$k = \frac{3}{2} (UI)^2$$

mentre per ε si può specificare la turbulent mixing length ℓ_t

$$\varepsilon = C_\mu^{\frac{3}{4}} \frac{k^{\frac{3}{2}}}{\ell_t}$$

nel caso in esame (flusso interno) è ragionevole assumere ℓ_t pari a $0.07 D_h$,
dove D_h è il diametro idraulico dei tubi in cui entra il flusso

Mixing Elbow: Impostazione del Caso /10

Condizioni al contorno turbolente:

- ove non specificato diversamente, l'intensità turbolenta, nel caso di flussi interni, può essere stimata mediante la formula

$$I = \frac{0.16}{Re^{\frac{1}{8}}}$$

per flussi esterni invece non esiste una formula precisa, ma valori dell'ordine di 2×10^{-4} o 5×10^{-4} sono generalmente accettati

- altre grandezze che si possono trovare a seconda dei modelli di turbolenza e dei solutori adottati (anche se non nel presente tutorial) sono:
 - viscosità cinematica turbolenta modificata: $\tilde{\nu} = \nu_t / C_\mu^{\frac{1}{4}}$
 - viscosità dinamica turbolenta: $\mu_t = \rho \nu_t$
 - velocità di dissipazione turbolenta specifica: $\omega = \varepsilon / (C_\mu k)$
 - diffusività termica turbolenta: $\alpha_t = \nu_t / Pr_t$,
ove l'assunzione $Pr_t = 0.85$ è generalmente accettata

Mixing Elbow: Impostazione del Caso /11

Condizioni al contorno turbolente:

- all'outlet è sufficiente imporre, coerentemente con le condizioni cinetiche, che il gradiente di k e di ε sia nullo
- la variabile ν_t non necessita di condizioni al contorno specifiche per le sezioni di inlet e di outlet, poiché viene calcolata sulla base dei valori di k e di ε
- per quanto riguarda le pareti, per tutte e tre le grandezze è possibile impiegare le *wall functions* che implementano il profilo universale di parete per flussi turbolenti non separati
- Riassumendo, i `boundaryField` types di OpenFOAM corrispondenti alle condizioni sopra descritte sono:

```

k @ inlet    => turbulentIntensityKineticEnergyInlet
ε @ inlet    => turbulentMixingLengthDissipationRateInlet
νt @ inlet  => calculated
k @ wall     => kqfWallFunction
ε @ wall     => epsilonWallFunction
νt @ wall   => nutkWallFunction
k @ outlet   => zeroGradient
ε @ outlet   => zeroGradient
νt @ outlet => calculated
  
```

Mixing Elbow: Impostazione del Caso /12

File 0/k:

```

dimensions [0 2 -2 0 0 0 0];
internalField uniform 1.5e-4;
boundaryField
{
    in1
    { type
turbulentIntensityKineticEnergyInlet;
    intensity 0.05;
    value uniform 1.5e-4;
    }
    in2
    { type
turbulentIntensityKineticEnergyInlet;
    intensity 0.05;
    value uniform 3.75e-3;
    }
    out
    {
    type zeroGradient;
    }
    wall
    {
    type kqRWallFunction;
    value uniform 0;
    }
}

```

File 0/nut:

```

dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
{
    in1
    {
    type calculated;
    value uniform 0;
    }
    in2
    {
    type calculated;
    value uniform 0;
    }
    out
    {
    type calculated;
    value uniform 0;
    }
    wall
    {
    type nutkWallFunction;
    value uniform 0;
    }
}

```


Mixing Elbow: Impostazione del Caso /13

File 0/epsilon:

```

dimensions [0 2 -3 0 0 0 0];
internalField uniform 1.35e-5;
boundaryField
{
  in1
  {
    type turbulentMixingLengthDissipationRateInlet;
    mixingLength 0.0224; // D1=32cm
    value uniform 1.35e-5;
  }
  in2
  {
    type turbulentMixingLengthDissipationRateInlet;
    mixingLength 0.0056; // D2=8cm
    value uniform 6.74e-3;
  }
  out
  {
    type zeroGradient;
  }
  wall
  {
    type epsilonWallFunction;
    value uniform 1.35e-5;
  }
}

```

Mixing Elbow: Impostazione del Caso /14

Cartella constant:

- file `transportProperties`:
 - lasciare il `transportModel` impostato a `Newtonian`
 - cambiare il valore di `nu` a `1e-06` (acqua)
- file `turbulenceProperties`
 - definisce il modello di turbolenza da impiegare
 - lasciare il `RASModel` impostato a `kEpsilon`

Cartella system:

- file `controlDict`:
 - per un solutore stazionario come `simpleFoam`, il `deltaT` è unitario
 - `endTime` indica quindi il numero massimo di iterazioni
 - eliminare le righe dalla `keyword functions` in avanti
- file `fvSchemes`:
 - impostare il `gradSchemes` a


```
default cellLimited Gauss linear 1;
```
 - lasciare gli altri valori inalterati, le entries per la variabili turbolente non utilizzate dal modello $k-\varepsilon$ sono ininfluenti (e.g. `nuTilda`, `R`)

Mixing Elbow: Impostazione del Caso /15

Cartella system (continua):

- file `fvSolution`:

- il subdictionary `residualControl` individua i criteri di convergenza
- a convergenza raggiunta il calcolo si arresta
- le soglie di default ($10^{-2} \div 10^{-3}$) sono grossolane, impostarle a 10^{-5}
- fare in modo che le `tolerance` nella sezione `solvers` siano \leq dei residui scelti (e.g. 10^{-6})
- impostare le tolleranze relative `relTol` a 10^{-3}

Mixing Elbow: Esecuzione

- il solver `simpleFoam` per il caso `elbow` può essere avviato dalla cartella radice del caso col comando
`simpleFoam | tee log`
- in alternativa il ridirezionamento dell'output si può ottenere con lo script
`foamJob -screen simpleFoam`
 il quale scrive l'output di `simpleFoam` su un file `log` e, con l'opzione `-screen`, lo invia anche al terminale
- a calcolo ultimato nella cartella `elbow` saranno comparse le sottocartelle di output contenenti i valori delle variabili alle rispettive iterazioni
- lo script `foamLog` permette di analizzare il file `log` scomponendolo in diversi files contenenti la traccia dei residui delle diverse equazioni. Tali files vengono scritti nella sottocartella `logs` e possono essere utilizzati per plottare il grafico dei residui
`foamLog log`

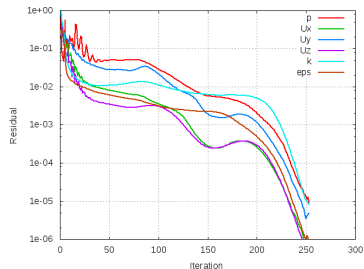
Mixing Elbow: Residui

I residui estratti dal file log con l'utility foamLog possono essere plottati con il programma gnuplot. Creare il file plotRes.plt contenente il seguente testo:

```
set terminal png
set output Residuals.png
set logscale y
set format y %1.0e
set yrange [1e-6:1]
set grid
set xlabel Iteration
set ylabel Residual
plot logs/p_0 w l lc 1 lw 2 t p ,\
logs/Ux_0 w l lc 2 lw 2 t Ux ,\
logs/Uy_0 w l lc 3 lw 2 t Uy ,\
logs/Uz_0 w l lc 4 lw 2 t Uz ,\
logs/k_0 w l lc 5 lw 2 t k ,\
logs/epsilon_0 w l lc 6 lw 2 t eps
```

ed eseguirlo col comando

● gnuplot plotRes.plt



Mixing Elbow: Post-Processing /1

Le funzionalità di ParaView per il post-processing dei casi di OpenFOAM sono già state illustrate all'interno del tutorial cavity.

Da notare che eseguire il post-processing di un insieme di dati non significa unicamente “produrre immagini colorate”.

In OpenFOAM esistono molte utilities per il post-processing dei dati da riga di comando: se ne richiamano alcune di seguito.

- foamCalc permette di eseguire operazioni sui dati per ricavare nuovi campi (visualizzabili anche in ParaView) in funzione dei campi esistenti. Utilizzo:

```
foamCalc calcType fieldName -options
```

dove calcType individua il tipo di operazione con una stringa tra: magGrad, magSqr, mag, addSubtract, randomise, div, interpolate, components

- e.g. per calcolare il campo del modulo di velocità

```
foamCalc mag U
```

in ogni cartella dei risultati comparirà il file magU

- e.g. per calcolare le componenti del vettore velocità solo all'iterazione finale

```
foamCalc components U -latestTime
```

nell'ultima cartella dei risultati compariranno i files Ux, Uy, Uz

Mixing Elbow: Post-Processing /2

- può essere utile calcolare il valore medio o l'integrale di una variabile su di una patch tramite le utilities `patchAverage` e `patchIntegrate`. Utilizzo:

```
patchAverage -options fieldName patchName
```

```
patchIntegrate -options fieldName patchName
```

- e.g. per calcolare la velocità media in uscita in direzione z all'iterazione finale

```
patchAverage -latestTime Uz out
```

- e.g. per calcolare la portata in volume in uscita all'iterazione finale

```
patchIntegrate -latestTime Uz out
```

```
patchIntegrate -latestTime phi out
```

nota: il campo `phi`, scritto ad ogni salvataggio, rappresenta il flusso attraverso le facce delle celle. Se il solutore utilizzato è incomprimibile questo valore è fornito in $\frac{m^3}{s}$, se comprimibile in $\frac{kg}{s}$. Le portate calcolate con la seconda opzione saranno quindi in volume o in massa a seconda dei casi

Mixing Elbow: Post-Processing /3

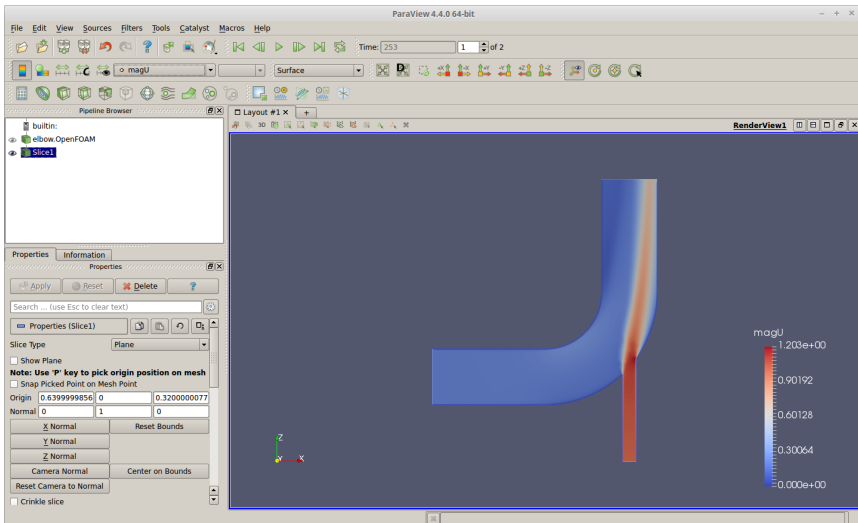
- l'utilizzo delle `wall functions` all'interno del modello di turbolenza prescrive che l'altezza Δy del centroide della prima cella in parete sia tale che

$$30 < y^+ < 300 \quad \text{dove} \quad y^+ = \frac{u_\tau \Delta y}{\nu}$$

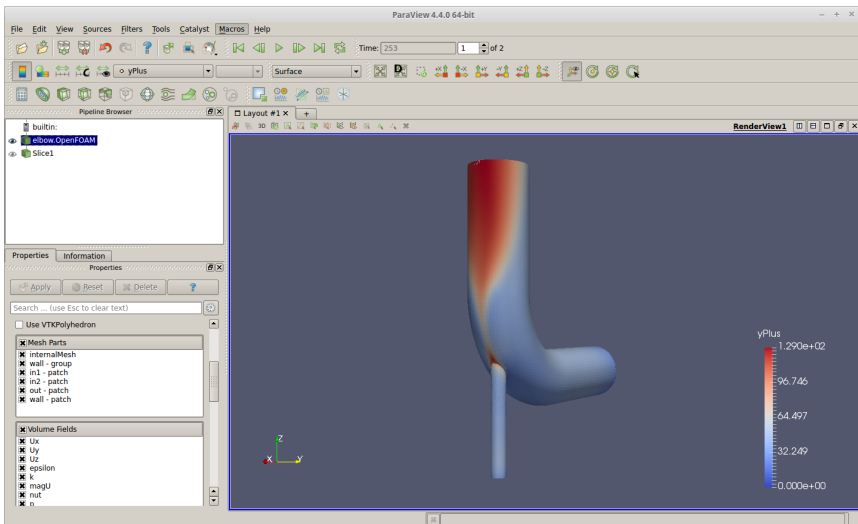
dove u_τ è detta `friction velocity`. Per verificare tale condizione si impiega l'utilità `yPlus` che visualizza i valori minimi, massimi e medi dell' y^+ sulle patch di tipo `wall` e crea il campo `yPlus` nelle cartelle dei risultati

- e.g. per scrivere il campo `yPlus` nella cartella dell'ultimo istante temporale `yPlus -latestTime`
- nel seguito si riportano due immagini ricavate da `ParaView`:
 campo del modulo di velocità sul piano di taglio mediano
 campo di y^+ alla parete
 essendo il calcolo stazionario, le immagini si riferiscono all'istante finale

Mixing Elbow: Post-Processing /4



Mixing Elbow: Post-Processing /5



Sommario

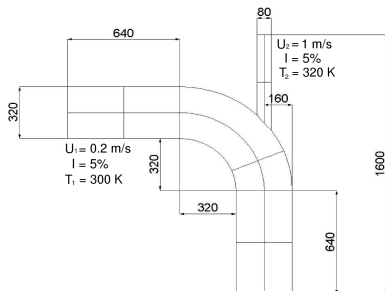
- 1 Introduzione a OpenFOAM
- 2 Lid-Driven Cavity
- 3 Mixing Elbow
- 4 Hot Mixing Elbow**
- 5 Micro-reference

Tutorial 3: Hot Mixing Elbow

- Oggetto dell'analisi: flusso turbolento non isoterma di due correnti di acqua che in un condotto si miscelano in corrispondenza di una curva a 90°
- Geometria analoga al caso precedente, ma con correnti che entrano a temperature differenti

- $\nu = 1 \times 10^{-6} \text{ m}^2/\text{s}$
- $\beta = 3 \times 10^{-4} \text{ K}^{-1}$
- $\text{Pr} = 6$
- Modello di turbolenza $k-\varepsilon$
- Solutore

buoyantBoussinesqSimpleFoam
 convezione con forze di galleggiamento (ipotesi di Boussinesq)



Hot Mixing Elbow: Impostazione del caso /1

Si procederà al setup manuale del caso, partendo da un tutorial del solutore `buoyantBoussinesqSimpleFoam` e riutilizzando ove possibile il setup del precedente tutorial

Copiare il tutorial `hotRoom` di `buoyantBoussinesqSimpleFoam` nella propria cartella `run` ed entrarvi

- `cp -r $FOAM_TUTORIALS/heatTransfer/buoyantBoussinesqSimpleFoam/hotRoom $FOAM_RUN/hotElbow`
- `cd $FOAM_RUN/hotElbow`

Rimuovere i files di mesh relativi al tutorial di partenza (se presenti) ed importare la mesh del caso precedente

- `rm constant/polyMesh/*`
- `cp -r $FOAM_RUN/elbow/constant/polyMesh constant`

Importare dal caso precedente anche alcuni files della cartella `0`: questo è possibile perché il solver `buoyantBoussinesqSimpleFoam` deriva da `simpleFoam`, col quale mantiene una certa compatibilità

- `cp $FOAM_RUN/elbow/0/U 0/`
- `cp $FOAM_RUN/elbow/0/k 0/`
- `cp $FOAM_RUN/elbow/0/epsilon 0/`
- `cp $FOAM_RUN/elbow/0/nut 0/`

Hot Mixing Elbow: Impostazione del caso /2

Cartella 0:

- la cartella 0 contiene anche i files `alphat`, `p`, `p_rgh`, `T`
- `alphat` è la diffusività termica turbolenta dinamica α_t . A parete si usa la `wall function` `alphatJayatillekeWallFunction`, che richiede la definizione di un numero di Prandtl turbolento ($Pr_t = 0.85$). Alle sezioni di inlet e outlet si impone gradiente nullo
- `p_rgh` è la pressione epurata dal contributo idrostatico

$$p_{rgh} = p - \rho g \Delta h$$

dove `p_rgh` non è fissata si impone la B.C. `fixedFluxPressure`, che richiede la definizione di una densità di riferimento. In accordo con l'ipotesi di Boussinesq si sceglie la densità cinematica `rho_k`

$$\rho_k = 1 - \beta(T - T_{ref})$$

dove T_{ref} è una opportuna temperatura di riferimento

- Il solutore ragiona in termini di `p_rgh`, il campo `p` è calcolato a posteriori (B.C. `calculated`)
- Per `T` si impongono le temperature agli inlet (B.C. `fixedValue`), la condizione di adiabaticità alla parete (B.C. `zeroGradient`), e la condizione derivata `inletOutlet` all'outlet. Quest'ultima corrisponde ad un `zeroGradient` con flusso in uscita, e ad un `fixedValue` con flusso in entrata

Hot Mixing Elbow: Impostazione del caso /3

File 0/alphat:

```

dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
{
    in.*
    {
        type calculated;
        value $internalField;
    }
    out
    {
        type calculated;
        value $internalField;
    }
    wall
    {
        type
        alphasatJayatillekeWallFunction;
        Prt 0.85;
        value $internalField;
    }
}

```

File 0/p:

```

dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
{
    .*
    {
        type calculated;
        value $internalField;
    }
}

```

Hot Mixing Elbow: Impostazione del caso /4

File 0/p_rgh:

```
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
{
  .*
  {
    type fixedFluxPressure;
    rho rhok;
    value $internalField;
  }
  out
  {
    type fixedValue;
    value $internalField;
  }
}
```

File 0/T:

```
dimensions [0 0 0 1 0 0 0];
internalField uniform 300;
boundaryField
{
  in1
  {
    type fixedValue;
    value uniform 300;
  }
  in2
  {
    type fixedValue;
    value uniform 320;
  }
  out
  {
    type inletOutlet;
    inletValue $internalField;
    value $internalField;
  }
  wall
  {
    type zeroGradient;
  }
}
```


Hot Mixing Elbow: Impostazione del caso /5

È anche possibile definire variabili ed eseguire dei calcoli all'interno dei files di setup di OpenFOAM utilizzando la sintassi del C++. Come esempio si riporta una versione modificata del file 0/epsilon in cui si impongono le condizioni iniziali per mezzo di opportuni calcoli

```

dimensions [0 2 -3 0 0 0 0];
D1 0.32; D2 0.08;
U1 0.2; U2 1.0;
I1 0.05; I2 0.05;
L1 #calc 0.07*$D1;
L2 #calc 0.07*$D2;
k1 #calc 1.5*pow($U1*$I1,2);
k2 #calc 1.5*pow($U2*$I2,2);
eps1 #calc 0.164*pow($k1,1.5)/$L1;
eps2 #calc 0.164*pow($k2,1.5)/$L2;
internalField uniform $eps1;
boundaryField
{
  inlet1
  {
    type turbulentMixingLengthDissipationRateInlet;
    mixingLength $L1;
    value uniform $eps1;
  }
}

```

...e così via ...

Hot Mixing Elbow: Impostazione del caso /6

Cartella constant:

- file `transportProperties`:
 - cambiare il valore di `nu` a $1e-06$
 - cambiare il valore di `beta` a $3e-04$
 - cambiare il valore di `Tref` a 310 (media delle temperature in ingresso)
 - cambiare il valore di `Pr` a 6
 - cambiare il valore di `Prt` a 0.85
- file `g`:
 - impostare il vettore gravità come $(0 \ -9.81 \ 0)$ (le forze di galleggiamento agiscono in direzione `y` negativa, trasversalmente al flusso)

Cartella system:

- per il file `controlDict` è possibile lasciare le impostazioni di default
- per il file `fvSchemes` impostare il `gradSchemes` a
`default cellLimited Gauss linear 1;`
- per facilitare la convergenza modificare il file `fvSolution` come segue:
 - porre tutte le `relTol` a 0.01;
 - impostare i `relaxationFactors`: `T 0.9; p 0.5; U 0.5;`
- si invita il lettore ad apportare ulteriori modifiche ai files della cartella `system` per verificarne l'effetto sulla convergenza del calcolo

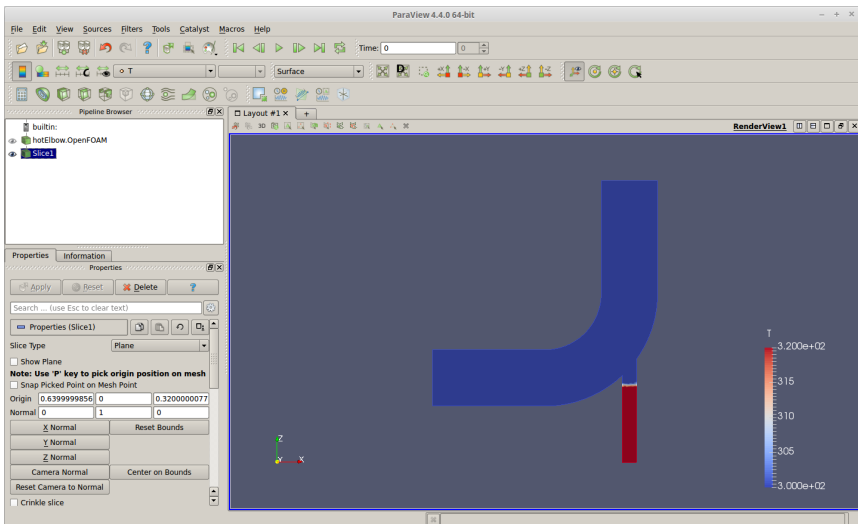
Hot Mixing Elbow: Impostazione del caso /7

Inizializzazione avanzata con l'utility `setFields`:

- l'utility `setFields` serve per settare le condizioni iniziali assegnando valori diversi alle variabili in regioni specifiche
 - inizializzare un calcolo in maniera più specifica contribuisce ad agevolarne una più rapida convergenza
 - per utilizzare `setFields` occorre compilare il dictionary `setFieldsDict` nella cartella `system` (vedi a destra)
 - e.g. si possono inizializzare la temperatura e la velocità all'interno del tubo secondario ai valori U_2 e T_2
- ⇒ compilare il `setFieldsDict`, quindi digitare `setFields`
- ⇒ controllare i campi iniziali con `paraFoam`

```
defaultFieldValues
(
  volScalarFieldValue
  T 300
  volVectorFieldValue
  U (0.2 0 0)
);
regions
(
  boxToCell
  {
    box (1.06 -0.06 -0.5)
    (1.18 0.06 -0.04);
    fieldValues
    (
      volScalarFieldValue
      T 320
      volVectorFieldValue
      U (0 0 1)
    );
  }
);
```

Hot Mixing Elbow: Impostazione del caso /8



Hot Mixing Elbow: Impostazione del caso /9

pyFoam:

- pyFoam è una libreria di scripts in linguaggio Python
- tali scripts permettono di automatizzare serie di operazioni in OpenFOAM
- pyFoam non è distribuito con OpenFOAM e va installato separatamente, per installarlo aprire un terminale e digitare:

```
sudo apt-get install python-numpy python-dev python-setuptools
svn co https://svn.code.sf.net/p/openfoam-extend/svn/trunk/\
  Breeder/other/scripting/PyFoam/
tar xzf PyFoam-0.6.5.tar.gz
rm PyFoam-0.6.5.tar.gz
cd PyFoam-0.6.5
sudo python setup.py install
cd ..
rm -r PyFoam-0.6.5
```

- alcune caratteristiche di pyFoam:
 - permette di eseguire applicazioni e analizzarne/modificarne l'output
 - permette di automatizzare runs parametrici dello stesso caso
 - permette di manipolare i dictionaries di OpenFOAM
- si utilizzerà pyFoam per plottare i residui con la simulazione in corso

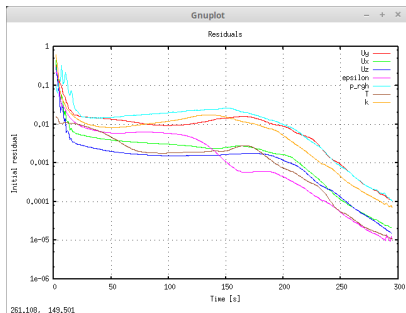
Hot Mixing Elbow: Esecuzione e Residui

Si avvia il solver `buoyantBoussinesqSimpleFoam` tramite lo script `foamJob` in background (ovvero, senza l'opzione `-screen`)

- `foamJob buoyantBoussinesqSimpleFoam`

Tramite lo script `pyFoamPlotWatcher.py` è possibile monitorare graficamente l'andamento dei residui in tempo reale

- `pyFoamPlotWatcher.py log`



```

Time = 295
GILPBMICG: Solving for ux, Initial residual = 2.12915e-05, Final residual = 6.41285e-06, No Iterations 1
GILPBMICG: Solving for uy, Initial residual = 0.000101161, Final residual = 4.71919e-06, No Iterations 1
GILPBMICG: Solving for uz, Initial residual = 1.46006e-05, Final residual = 1.27079e-06, No Iterations 1
GILPBMICG: Solving for T, Initial residual = 1.02726e-05, Final residual = 3.20229e-06, No Iterations 2
DICPCK: Solving for p_rgh, Initial residual = 0.000102040, Final residual = 9.76744e-07, No Iterations 112
time step continuity errors : sum local = 1.06702e-07, global = 2.14370e-09, cumulative = -9.90157e2
GILPBMICG: Solving for epsilon, Initial residual = 1.94601e-05, Final residual = 5.93536e-07, No Iterations 1
GILPBMICG: Solving for k, Initial residual = 6.75456e-05, Final residual = 4.12188e-06, No Iterations 1
ExecutionTime = 399.68 s  ClockTime = 418 s

Time = 296
GILPBMICG: Solving for ux, Initial residual = 2.07587e-05, Final residual = 4.9622e-06, No Iterations 1
GILPBMICG: Solving for uy, Initial residual = 9.69207e-05, Final residual = 4.31094e-06, No Iterations 1
GILPBMICG: Solving for uz, Initial residual = 1.39629e-05, Final residual = 1.15112e-06, No Iterations 1
GILPBMICG: Solving for T, Initial residual = 1.00148e-05, Final residual = 3.09959e-06, No Iterations 2
DICPCK: Solving for p_rgh, Initial residual = 0.000101937, Final residual = 1.94497e-06, No Iterations 24
time step continuity errors : sum local = 1.50959e-07, global = -2.51924e-08, cumulative = -9.00158704
GILPBMICG: Solving for epsilon, Initial residual = 8.8836e-06, Final residual = 0.8836e-06, No Iterations 0
GILPBMICG: Solving for k, Initial residual = 6.70977e-05, Final residual = 4.07766e-06, No Iterations 1
ExecutionTime = 400.14 s  ClockTime = 411 s

SIMPLE solution converged in 296 iterations
End

```

Hot Mixing Elbow: Post-Processing /1

Post-processing avanzato: l'utility sample

- l'utility `sample` serve ad estrarre dati dai risultati del calcolo, allo scopo di creare grafici e visualizzazioni avanzate
 - i due tipi di output sono i `sets` (profili delle variabili lungo linee o curve) e le `surfaces` (superfici, piani, o isosuperfici delle distribuzioni delle variabili)
 - per utilizzare l'utility `sample` occorre compilare il dictionary `sampleDict` nella cartella `system`
 - per saperne di più sulle opzioni di `sample` consultare il file `$FOAM_UTILITIES/postProcessing/sampling/sample/sampleDict`
 - per il presente tutorial vediamo come:
 - estrarre i profili di `U` e `T` su due diametri paralleli a `y` e `z` in prossimità dell'outlet
 - salvare `U` e `T` sul piano mediano
 - salvare due isosuperfici di `U` e `T` al valore medio di `T`
- ⇒ dopo aver compilato il `sampleDict` (vedi pagina seguente) digitare `sample -latestTime`
- ⇒ controllare il contenuto della cartella `sets`
- ⇒ con `paraFoam` aprire il file `surfaces/latestTime/U_iso_T.vtk`

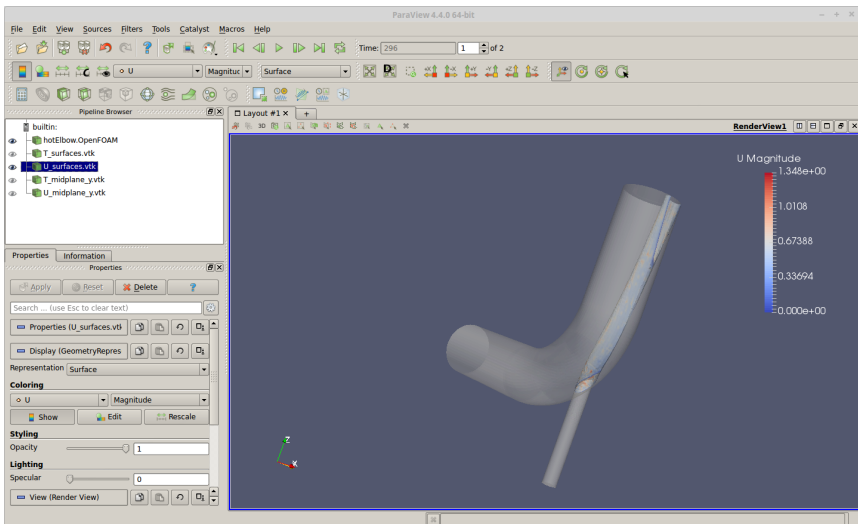
Hot Mixing Elbow: Post-Processing /2

File system/sampleDict:

```
set format raw;
surfaceFormat vtk;
interpolationScheme cellPointFace;
fields
(
  T
  U
);
sets
(
  outlet_x
  {
    type uniform;
    axis x;
    start (0.96 0 1.12);
    end (1.28 0 1.12);
    nPoints 30;
  }
  outlet_y
  {
    type face;
    axis y;
    start (1.12 -0.16 1.12);
    end (1.12 0.16 1.12);
  }
);
```

```
surfaces
(
  midplane_y
  {
    type plane;
    basePoint (0 0 0);
    normalVector (0 1 0);
    interpolate true;
  }
  surfaces
  {
    type isoSurfaceCell;
    isoField T;
    isoValue 310;
    interpolate true;
  }
);
```


Hot Mixing Elbow: Post-Processing /3



Sommario

- 1 Introduzione a OpenFOAM
- 2 Lid-Driven Cavity
- 3 Mixing Elbow
- 4 Hot Mixing Elbow
- 5 Micro-reference

Inlet/Outlet/Wall BCs: U

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per U:

Inlet

- `type fixedValue;`
`value uniform (1.0 0.0 0.0);` \\ m/s
- `type surfaceNormalFixedValue;`
`refValue uniform -1.0;` \\ m/s, <0 if entering
- `flowRateInletVelocity;`
`volumetricFlowRate constant 0.1;` \\ oppure `massFlowRate`
- `type freestream;`
`freestreamValue (1.0 0.0 0.0);` \\ m/s

Outlet

- `type inletOutlet;`
`value uniform (1.0 0.0 0.0);` \\ m/s, placeholder
`inletValue uniform (1.0 0.0 0.0);` \\ m/s
- `type zeroGradient;`

Wall

- `type fixedValue;`
`value uniform (0.0 0.0 0.0);` \\ m/s
- `type slip;`

Inlet/Outlet/Wall BCs: p , p_rgh

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per p , p_rgh :

Inlet/Outlet ● type freestreamPressure;

Inlet/Wall ● type zeroGradient;

- type fixedFluxPressure
rho rhok; \\ solo per solver di Boussinesq
value uniform 0.0; \\ Pa, $\neq 0$ if p_rgh

Outlet ● type fixedValue;
value uniform 0.0; \\ Pa, $\neq 0$ if p_rgh

- type totalPressure;
p0 uniform 0.0; \\ Pa, $\neq 0$ if p_rgh
U U;
phi phi;
rho none;
psi none;
gamma 1;

Inlet/Outlet/Wall BCs: T

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per T:

- Inlet**
- `type totalTemperature;`
`U U;`
`phi phi;`
`psi thermo::psi;`
`gamma 1.4;`
`T0 uniform 300; \\ K`
- Inlet/Wall**
- `type fixedValue;`
`value uniform 300; \\ K`
- Outlet**
- `type inletOutlet;`
`value uniform 300; \\ K, placeholder`
`inletValue uniform 300; \\ K`
- Outlet/Wall**
- `type zeroGradient;`
- Wall**
- `type compressible::turbulentHeatFluxTemperature;`
`heatSource flux; \\ (oppure power)`
`q uniform 10; \\ W/m2 se flux, W se power`
`kappa fluidThermo; kappaName none; Qr none;`
 - `type wallHeatTransfer;`
`Tinf uniform 300; \\ K`
`alphaWall uniform 20; \\ W/K coeff scambio termico`

Inlet/Outlet/Wall BCs delle Grandezze Turbolente: nuTilda

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per nuTilda:

- Inlet**
- `type fixedValue;`
`value uniform 0.14; \\ m2/s`
 - `type freestream;`
`freestreamValue uniform 0.14; \\ m2/s`
- Outlet**
- `type zeroGradient;`
 - `type inletOutlet;`
`value uniform 0.14; \\ m2/s, placeholder`
`inletValue uniform 0.14; \\ m2/s`
 - `type freestream;`
`freestreamValue uniform 0.14; \\ m2/s`
- Wall**
- `type fixedValue;`
`value uniform 0.0; \\ m2/s`
 - `type zeroGradient;`

Inlet/Outlet/Wall BCs delle Grandezze Turbolenti: k

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per k :

Inlet

- `type fixedValue;`
`value uniform 0.01; \\ m2/s2`
- `type turbulentIntensityKineticEnergyInlet;`
`intensity 0.05; \\ %`
`value uniform 0.01; \\ m2/s2, placeholder`

Outlet

- `type zeroGradient;`
- `type inletOutlet;`
`value uniform 0.01; \\ m2/s2, placeholder`
`inletValue uniform 0.01; \\ m2/s2`

Wall High-Re

- `type kqRWallFunction;`
`value uniform 0.01; \\ m2/s2, placeholder`

Wall Low-Re

- `type kLowReWallFunction;`
`value uniform 1e-12; \\ m2/s2, low value $\neq 0$, plchold`

Inlet/Outlet/Wall BCs delle Grandezze Turbolenti: epsilon

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per epsilon:

Inlet

- `type fixedValue;`
`value uniform 0.01; \\ m2/s3`
- `type turbulentMixingLengthDissipationRateInlet;`
`mixingLength 0.005; \\ m`
`value uniform 0.01; \\ m2/s3, placeholder`

Outlet

- `type zeroGradient;`
- `type inletOutlet;`
`value uniform 0.01; \\ m2/s3, placeholder`
`inletValue uniform 0.01; \\ m2/s3`

Wall High-Re

- `type epsilonWallFunction;`
`value uniform 0.01; \\ m2/s3, placeholder`

Wall Low-Re

- `type epsilonLowReWallFunction;`
`value uniform 0.01; \\ m2/s3, placeholder`

Inlet/Outlet/Wall BCs delle Grandezze Turbolenti: ω

Esempi di utilizzo dei più comuni tipi di condizioni al contorno per ω :

Inlet

- `type fixedValue;`
`value uniform 0.1; \ \ 1/s`
- `type turbulentMixingLengthFrequencyInlet;`
`mixingLength 0.005; \ \ m`
`value uniform 0.1; \ \ 1/s, placeholder`

Outlet

- `type zeroGradient;`
- `type inletOutlet;`
`value uniform 0.1; \ \ 1/s, placeholder`
`inletValue uniform 0.1; \ \ 1/s`

Wall High-Re

- `type omegaWallFunction;`
`value uniform 0.1; \ \ 1/s, placeholder`

Wall Low-Re

- `type omegaWallFunction;`
`value uniform 0.1; \ \ 1/s, placeholder`

Inlet/Outlet/Wall BCs delle Grandezze Turbolenti: nut, alphas

Esempi di utilizzo dei comuni tipi di condizioni al contorno per nut, alphas:

- | | |
|--------------|---|
| Inlet/Outlet | <ul style="list-style-type: none"> ● type calculated;
 value uniform 0.0; \\ m2/s [nut], kg/ms [alphas]
 \\ placeholder |
| nut | |
| Wall High-Re | <ul style="list-style-type: none"> ● type nutUSpaldingWallFunction;
 value uniform 0.0; \\ m2/s, placeholder
 \\ da usare con modello di turbolenza Spalart-Allmaras ● type nutUWallFunction;
 value uniform 0.0; \\ m2/s, placeholder, $k-\omega$ ● type nutkWallFunction;
 value uniform 0.0; \\ m2/s, plchld, $k-\epsilon$, $k-\omega$ |
| nut | |
| Wall Low-Re | <ul style="list-style-type: none"> ● type nutLowReWallFunction;
 value uniform 0.0; \\ m2/s, placeholder |
| alphas | |
| Wall High-Re | <ul style="list-style-type: none"> ● type alphasJayatillekeWallFunction; \\ oppure compr...
 value uniform 0.0; \\ kg/ms, placeholder ● type compressible::alphasWallFunction; \\ \neq incompr...
 value uniform 0.0; \\ kg/ms, placeholder |
| alphas | |
| Wall Low-Re | <ul style="list-style-type: none"> ● type calculated;
 value uniform 0.0; \\ kg/ms, placeholder |

Altri Tipi di Condizioni al Contorno Utili

- `type symmetry;`
per definire piani di simmetria
- `type empty;`
per definire le pareti nella terza dimensione in calcoli 2D piani
- `type wedge;`
per definire le pareti nella terza dimensione in calcoli 2D assialsimmetrici
- `type cyclic;`
per trattare due patches come fossero fisicamente connesse, la definizione di una `neighbourPatch` è necessaria

Le patches in questi casi vanno definite come tali sia nel file `constant/polyMesh/boundary` che nei file della cartella `0/`

fvSchemes: ddtSchemes e gradSchemes

ddtSchemes

```

{ default steadyState; }           \\ stazionario
{ default Euler; }                 \\ instazionario, I ordine
{ default crankNicolson 0.5; }    \\ instazionario, blending
{ default crankNicolson 1.0; }    \\ instazionario, II ordine

```

gradSchemes

```

{ default Gauss linear; }          \\ schema standard
{ default cellMDLimited Gauss linear  $\varphi$ ; } \\ schema standard limitato
{ default cellLimited Gauss linear  $\varphi$ ; }    \\ schema standard limitato
\\ ove  $0 \leq \varphi \leq 1$  indica l'entità del limitatore
\\  $\varphi = 0$  limitatore non attivo, +accuratezza, -stabilità
\\  $\varphi = 1$  limitatore al massimo, -accuratezza, +stabilità
\\ i limitatori sono utili qualora il calcolo presenti instabilità
\\ il cellMDLimited, ove sufficiente, è da preferirsi perché meno diffusivo

```

fvSchemes: divSchemes

```

\\ I ordine, ++diffusivo, ++stabile, --accurato
divSchemes
{ default      bounded Gauss upwind;
  div(phi,U) bounded Gauss upwind; ...
  div((nuEff*dev2(T(grad(U)))) Gauss linear; } \\ non modificare!!

\\ II ordine, +diffusivo, +stabile, -accurato
divSchemes
{ default      bounded Gauss linearUpwind default;
  div(phi,U) bounded Gauss linearUpwind grad(U); ...
  div((nuEff*dev2(T(grad(U)))) Gauss linear; } \\ non modificare!!

\\ II ordine con limitatore, -diffusivo, -stabile, +accurato
divSchemes
{ default      bounded Gauss limitedLinear  φ;
  div(phi,U) bounded Gauss limitedLinearV φ; ...
  div((nuEff*dev2(T(grad(U)))) Gauss linear; } \\ non modificare!!
\\ il limitatore ( $0 \leq \varphi \leq 1$ ) dà +stabilità e -accuratezza del linear puro

\\ II ordine, --diffusivo, --stabile, ++accurato
divSchemes
{ default      bounded Gauss linear;
  div(phi,U) bounded Gauss linear; ...
  div((nuEff*dev2(T(grad(U)))) Gauss linear; } \\ non modificare!!

```

fvSchemes: interpolationSchemes, laplacianSchemes e
snGradSchemes

interpolationSchemes

```
{ default linear; } \\ non modificare!!
```

laplacianSchemes

```
{ default Gauss linear uncorrected; } \\ mesh molto non-ortogonali (e.g. >80)
```

```
{ default Gauss linear limited  $\varphi$ ; } \\ mesh non-ortogonali
```

```
\\ aumentare  $\varphi$  con l'ortogonalità della mesh; 0=uncorrected, 1=corrected
```

```
{ default Gauss linear corrected; } \\ mesh ortogonali (e.g. <40)
```

```
{ default Gauss linear orthogonal; } \\ mesh molto ortogonali (e.g. <<40)
```

snGradSchemes \\ vedi sopra

```
{ default uncorrected; }
```

```
{ default limited  $\varphi$ ; }
```

```
{ default corrected; }
```

```
{ default orthogonal; }
```

Risoluzione della Mesh a Parete 1

L'utilizzo delle funzioni di parete (wall functions) all'interno del modello di turbolenza prescrive che l'altezza Δy del centroide della prima cella in parete sia tale che

$$30 < y^+ = \frac{u_\tau \Delta y}{\nu} < 300$$

La friction velocity u_τ è definita come

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

dove τ_w è la tensione tangenziale alla parete. Questa può essere stimata come

$$\tau_w = \frac{f \rho U^2}{8}$$

dove f è il fattore di attrito medio secondo Darcy del flusso in esame. Ne deriva che la altezza della cella a parete ($2\Delta y$) può essere stimata (come ordine di grandezza) con la relazione

$$2\Delta y = \frac{4\sqrt{2}y^+\nu}{U\sqrt{f}}$$

Risoluzione della Mesh a Parete 2

Resta ora da individuare un metodo per stimare il fattore di attrito secondo Darcy.

- Per flussi interni è possibile usare la formula iterativa di Colebrook, che per tubi lisci risulta essere

$$\frac{1}{\sqrt{f}} = -2 \log \left(\frac{2.51}{\text{Re} \sqrt{f}} \right), \quad \text{Re} = \frac{UD}{\nu}$$

Una formula alternativa, non iterativa e leggermente meno precisa è quella di Petukhov

$$f = [0.79 \ln(\text{Re}) - 1.64]^{-2}$$

- Per flussi esterni, ci si può servire della formula empirica di Schlichting

$$f = \frac{4 \times 0.455}{\log(\text{Re})^{2.58}}$$

Da non confondere il fattore di attrito secondo Darcy col fattore di attrito secondo Fanning (o anche skin friction coefficient)

$$f_{\text{darcy}} = 4 f_{\text{fanning}}$$

Risoluzione della Mesh a Parete 3

In alternativa all'uso della funzioni di parete, è possibile risolvere lo strato limite adottando un valore di $y^+ < 1$. Qualora questo richieda una mesh di dimensioni eccessive, è accettabile un valore di y^+ che sia comunque ≤ 5 . È invece da evitare il range tra 5 e 30, in quanto con queste condizioni non è possibile risolvere adeguatamente il comportamento del fluido a parete nè con l'utilizzo delle wall functions, nè risolvendo direttamente lo strato limite.